University of South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences (TNM) - Campus Porsgrunn

Industrial IT and Automation (IIA) master programme

Course IIA2017 - Industrial Information Technology

Lecture notes with sections for systems engineering, process control systems, autonomous systems, data communication, data security, digitalization, instrumentation, process database, embedded systems, multitasking systems and real-time requirements, alarm system, EEx/ATEX, testing, data recovery, and systems design.

© Nils-Olav Skeie (NOS)

January 3, 2022

Contents

Preface **xv**

Nomenclature **xvi**

I Introduction **1**

1 Introduction **2**
1.1 Overview 2

II Systems Engineering **5**

2 Systems Engineering **6**
2.1 Introduction 6
2.2 Scope 6
2.3 Complex systems 7
2.4 System design 7
2.4.1 Product realization 9
2.4.2 Technical Management 9
2.4.3 SysML 9
2.4.4 Interconnections 9
2.4.5 Ageing 10
2.4.6 History 10
2.5 Research and development 11

III Process control system **12**

3 Automation systems **13**
3.1 Automation 13
3.2 Control system 14
3.3 Process control system 15
3.4 Distributed Systems 16
3.5 System Reliability 17
3.5.1 Introduction 17
3.5.2 Estimation 18
3.5.3 Computation 18
3.6 Redundancy 19
3.6.1 Communication 20
3.6.2 Why may redundancy be important? 20
3.7 Cluster 20

4 Process Control System **21**
4.1 Introduction 21
4.2 SCADA History 21
4.3 SCADA Overview 22

4.4 SCADA modules . . . 24
4.4.1 User Interface (UI) . . . 24
4.4.2 Tag number conventions . . . 28
4.4.3 Data Engine . . . 28
4.4.4 Database . . . 31
4.4.5 Alarm system . . . 31
4.5 RTU devices . . . 32
4.5.1 PLC . . . 32
4.5.2 DCS . . . 33
4.5.3 PAC . . . 34
4.5.4 Use PLC or DCS? . . . 34
4.6 RTU Subsystems . . . 34
4.6.1 PID Controller . . . 34
4.6.2 CNC Machines . . . 36
4.6.3 Robots . . . 36
4.6.4 Instrumentation . . . 36
4.7 MDIS network . . . 36
4.8 Subsea systems . . . 37
4.9 Superior SCADA systems . . . 37
4.9.1 ERP . . . 38
4.9.2 MES . . . 38
4.9.3 IMS . . . 39
4.9.4 MOM . . . 39
4.10 Safety . . . 39
4.10.1 Safety system . . . 39
4.10.2 Shutdown system . . . 40
4.11 Life Time . . . 40
4.12 Documentation . . . 41
4.13 Planning . . . 41
4.14 Integrated Operations . . . 42
4.15 System Development . . . 42
4.16 System extension or replacement . . . 43
4.17 Oil & Gas: Integrated Operations . . . 43
4.18 Size of a Process Control System . . . 43
4.19 Security of Process Control Systems . . . 44

IV Autonomous systems 45

5 Autonomous Systems 46
5.1 Introduction . . . 46
5.2 Automation versus Autonomous . . . 46
5.3 Overview of building blocks . . . 47
5.3.1 Input information . . . 47
5.3.2 Perception . . . 48
5.3.3 Decision . . . 48
5.3.4 Actuation . . . 49
5.4 Industrial systems . . . 49
5.5 System development . . . 49
5.6 Examples . . . 49

V Digitalization 52

6 Introduction 53
6.1 Text Information . . . 54
6.2 Interface . . . 55
6.3 Network . . . 55
6.4 Protocol . . . 57

6.5 Information . . . 58
6.6 Cloud service . . . 58
6.7 Business models and work processes . . . 59

7 Information **61**
7.1 Introduction . . . 61
7.2 Handling . . . 61
7.2.1 Outliers . . . 61
7.2.2 Missing samples . . . 61
7.2.3 Filtering . . . 62
7.2.4 Merging data sets . . . 62
7.2.5 Storage of information . . . 62
7.2.6 Backup . . . 65

8 IoT and IIoT **66**
8.1 IoT protocols . . . 67
8.2 Edge devices and fog computing . . . 70

9 Process control systems **71**

10 Industry 4.0 **73**
10.1 Augmented reality . . . 76
10.2 Artificial intelligence . . . 76
10.3 Machine learning . . . 77

11 Autonomous systems **78**

12 Data security **80**

13 Digital21 **82**

VI Data Communication **83**

14 Introduction **84**
14.1 Process network . . . 84
14.2 Computer communication . . . 84

15 OSI model **87**
15.1 Introduction . . . 87
15.2 Protocol . . . 87
15.3 Directions . . . 89
15.4 OSI communication protocol model . . . 89
15.4.1 Layer 1: Physical Layer . . . 91
15.4.2 Layer 2: Data Link Layer . . . 93
15.4.3 Layer 3: Network Layer . . . 93
15.4.4 Layer 4: Transport Layer . . . 94
15.4.5 Layer 5: Session Layer . . . 94
15.4.6 Layer 6: Presentation Layer . . . 95
15.4.7 Layer 7: Application Layer . . . 95
15.4.8 Layer overview . . . 95

16 Local Area Network (LAN) **97**
16.1 Network Topologies . . . 97
16.2 IEEE 802 Standards . . . 100
16.3 Medium Access Control (MAC) . . . 100
16.4 Network communication devices . . . 101
16.5 Ethernet (IEEE 802.3) . . . 102
16.5.1 Ethernet History . . . 102
16.5.2 CAT Cable . . . 104

16.5.3 Standards and cables . . . 105
16.5.4 Ethernet Protocol . . . 106
16.6 Token Ring (IEEE 802.5) . . . 108
16.7 TCP/IP protocol . . . 109
16.7.1 Internet and address . . . 111
16.7.2 Programming . . . 117
16.7.3 Routing . . . 119
16.7.4 Security . . . 120
16.7.5 System . . . 120
16.8 Firewalls . . . 121
16.9 Time Synchronization . . . 122
16.9.1 Time Time Protocol (NTP) . . . 122
16.9.2 Time Sensitive Networking . . . 122
16.10Email system . . . 123
16.11Other network protocols . . . 123
16.12Virtual LAN . . . 124
16.13VPN . . . 124
16.14Proxy Server . . . 124

17 Physical layer 125
17.1 Parallel communications . . . 125
17.1.1 Centronics . . . 125
17.1.2 SCSI . . . 127
17.1.3 IDE / ATA . . . 128
17.2 Serial communication . . . 130
17.2.1 Asynchronous and synchronous communication . . . 130
17.2.2 Asynchronous communication . . . 131
17.2.3 Asynchronous communication protocols . . . 132
17.2.4 Current Loop . . . 140
17.2.5 Galvanic Isolation . . . 141
17.2.6 Optical or fibre cables . . . 143
17.2.7 Radio Data Communication . . . 144
17.2.8 5G wireless communication . . . 151

18 Communication Protocols 152
18.1 Serial communication protocols . . . 152
18.1.1 SATA (Serial ATA) . . . 152
18.1.2 USB . . . 154
18.1.3 FireWire . . . 160
18.1.4 Hart . . . 161
18.2 Network Communication Protocol . . . 164
18.2.1 Modbus . . . 165
18.2.2 XML . . . 169
18.2.3 SOAP . . . 170
18.2.4 OPC . . . 170
18.2.5 IoT . . . 170
18.2.6 MQTT . . . 171
18.3 Mobile Phones . . . 171
18.3.1 Short message service (SMS) . . . 172
18.4 SMS message types . . . 175
18.5 GPS . . . 178
18.6 Data Files . . . 179
18.6.1 CSV . . . 179
18.7 Network . . . 180
18.7.1 Internet Of Things (IoT) . . . 180
18.8 Buildings . . . 180
18.9 Select a Communication Protocol . . . 180
18.9.1 Node addressing . . . 181

18.9.2 Message or data oriented protocol . . . 181
18.9.3 Communication flow . . . 181
18.9.4 Data encoding . . . 182
18.9.5 Error Detection or Correction . . . 182
18.9.6 Data Flow . . . 182

19 Network Architecture 183
19.1 Architecture Components . . . 183
19.2 Middleware . . . 184
19.3 Firewall . . . 186
19.4 Sensor and actuator network . . . 187
19.5 Serial port server . . . 188
19.6 Virtual Machine . . . 190

20 Cloud based Systems 191
20.1 Introduction . . . 191
20.2 Type of cloud based systems . . . 192
20.3 Type of cloud based computing services . . . 192
20.3.1 Infrastructure as a Service (IaaS) . . . 192
20.3.2 Platform as a Service (PaaS) . . . 193
20.3.3 Serverless Computing . . . 193
20.3.4 Software as a Service . . . 193
20.4 IoT and IIOT . . . 193

21 Data communication in C# 194
21.1 Introduction . . . 194
21.2 Serial port data communication . . . 194
21.2.1 Introduction . . . 194
21.2.2 Physical layer . . . 195
21.2.3 Asynchronous serial communication . . . 195
21.2.4 Communication protocols . . . 196
21.3 Serial port programming in C# . . . 203
21.3.1 Introduction . . . 203
21.3.2 Three tier architecture . . . 204
21.3.3 Serial port library . . . 205
21.3.4 Terminal console application . . . 209
21.4 Network communication programming in C# . . . 213
21.4.1 Introduction . . . 213
21.4.2 OSI model . . . 213
21.4.3 TCP/IP protocol . . . 214
21.4.4 Socket programming in C# . . . 215
21.4.5 Console test application . . . 216
21.4.6 GUI Chat application . . . 222
21.4.7 Security . . . 230

VII Fieldbus 231

22 Introduction 232
22.1 Field bus overview . . . 232
22.1.1 Field bus or Serial Communication . . . 233
22.1.2 Description . . . 235
22.1.3 Interoperability . . . 235
22.1.4 Hot plugging . . . 235
22.1.5 Environment . . . 235
22.2 Advantages and disadvantages . . . 236
22.2.1 Advantages using a fieldbus . . . 236
22.2.2 Disadvantages using a fieldbus . . . 236
22.3 Types of Field buses . . . 236

22.4 Field bus selection . . . 237
22.4.1 Smart Factory . . . 238

23 Sensor buses (I/O buses) 239
23.1 AS-Interface . . . 239
23.2 CAN-bus . . . 240
23.3 EIB . . . 240
23.4 Interbus . . . 240
23.5 I^2C . . . 240
23.6 Modbus . . . 240
23.7 Hart . . . 241
23.8 IO-Link . . . 241

24 Process buses 242
24.1 PROFIBUS . . . 242
24.1.1 PROFIBUS FMS . . . 242
24.1.2 PROFIBUS DP . . . 243
24.1.3 PROFIBUS PA . . . 244
24.1.4 OSI model . . . 244
24.2 Field bus Foundation . . . 246
24.2.1 FF implementation . . . 246
24.2.2 FF Safety . . . 247
24.2.3 FF in OSI model . . . 247
24.3 PROFINET . . . 247
24.4 Controlnet . . . 248
24.5 LON works . . . 250
24.6 Interbus . . . 250
24.7 CAN . . . 251
24.7.1 OBD . . . 253
24.8 DeviceNet . . . 254
24.9 FDDI . . . 254
24.10 Modbus . . . 255
24.11 Industrial Ethernet . . . 255
24.11.1 Powerlink . . . 256
24.12 Fieldbus versus IoT . . . 256
24.13 Building Automation Systems . . . 256
24.14 Classification of serial bus systems . . . 257
24.15 Field bus summary . . . 257

VIII OPC 259

25 Introduction 260
25.1 Background . . . 260
25.2 Operating systems . . . 261
25.3 Software application . . . 262
25.4 Data models . . . 262
25.5 Communication models . . . 264
25.5.1 Client/server model . . . 265
25.5.2 Publisher/subscriber model . . . 265

26 OPC specification 268
26.1 Introduction . . . 268
26.2 Communication . . . 268
26.3 OPC Common . . . 270
26.4 OPC Data Access (OPC DA) . . . 271
26.5 OPC Alarms & Events (OPC AE) . . . 274
26.6 OPC Data eXchange (OPC DX) . . . 275
26.7 OPC Historical Data Access (OPC HDA) . . . 275

26.8 OPC Batch . . . 276
26.9 OPC Complex Data (OPC CD) . . . 277
26.10 OPC Security . . . 278
26.11 OPC XML-DA . . . 280
26.12 OPC UA . . . 281
26.12.1 Service Oriented Architecture . . . 283
26.12.2 Security . . . 284
26.12.3 MDIS network . . . 284
26.13 IoT and IIoT . . . 284
26.14 OPC Xi . . . 285

27 OPC systems **286**
27.1 Introduction . . . 286
27.2 Why use OPC . . . 286
27.3 OPC development . . . 286
27.4 OPC test . . . 287

IX Data Security **288**

28 Computer Security **289**
28.1 Introduction . . . 289
28.2 Information security . . . 290
28.2.1 Security triangle (triad) . . . 290
28.2.2 Security map . . . 291
28.2.3 Risk management . . . 292
28.2.4 Backup . . . 293
28.3 Grading of information . . . 293
28.3.1 General Data Protection Regulation (GDPR) . . . 294
28.3.2 Encryption . . . 294
28.3.3 Block chains and Bitcoins . . . 295
28.4 Network . . . 295
28.4.1 Firewall . . . 296
28.4.2 Internet . . . 296
28.4.3 Internet of Things (IoT) . . . 297
28.4.4 Mobile Phone . . . 299
28.4.5 Ping application . . . 299
28.4.6 Wireless network . . . 299
28.5 Cyber threats . . . 299
28.5.1 Passwords . . . 300
28.5.2 Emails . . . 301
28.5.3 Malware . . . 301
28.5.4 Denial of Service (DoS) . . . 303
28.5.5 Botnet . . . 303
28.5.6 Cyber attack . . . 304
28.6 Process control systems (PCS) . . . 304
28.6.1 Control system security standards . . . 306
28.6.2 Protocols . . . 307
28.7 Are the computer system hacked? . . . 308
28.8 Cyber security skills . . . 308

X Instrumentation **309**

29 Introduction **310**

30 Sensor and Actuator devices **312**
30.1 ON/OFF actuators . . . 313
30.2 Continuous actuators . . . 314
30.2.1 Mechanics . . . 314
30.2.2 Actuator types . . . 314
30.2.3 Motors . . . 315
30.2.4 Control valves . . . 317
30.2.5 Pumps . . . 318
30.2.6 Motor control . . . 318
30.2.7 Relay . . . 320
30.3 Sensors . . . 320
30.3.1 Smart sensors . . . 320

31 Data Acquisition (DAQ) Systems **321**
31.1 Introduction . . . 321
31.2 Digital representation of numbers . . . 324
31.2.1 Integers . . . 324
31.2.2 Floating numbers . . . 325
31.3 ASCII codes . . . 326
31.4 DAQ parts . . . 327
31.4.1 Counters . . . 327
31.4.2 Digital inputs . . . 327
31.4.3 Digital outputs . . . 327
31.4.4 Digital to Analog Converters . . . 328
31.4.5 Analog to Digital Converter . . . 330
31.4.6 Transfer of A/D conversion to system memory . . . 338
31.5 Range check of signal values . . . 338

32 Discrete Sampling **340**
32.1 A/D conversion . . . 340
32.2 Simultaneous Sample and Hold . . . 342
32.3 Aliasing . . . 342
32.4 Oversampling . . . 343
32.5 Folding diagram . . . 343
32.6 Selecting the sampling rate and filtering . . . 344
32.7 Filtering . . . 345
32.7.1 Low pass filter . . . 348
32.7.2 High pass filter . . . 350
32.7.3 FIR or IIR filter . . . 351

XI Process database **353**

33 Introduction **354**

34 Process data **355**
34.1 Configuration data . . . 355
34.2 Runtime and alarms . . . 357
34.3 Historical data . . . 358
34.4 Data compressions . . . 358

35 Database structure **359**
35.1 Database models . . . 359
35.1.1 Flat model . . . 360
35.1.2 Hierarchical model . . . 361
35.1.3 Network model . . . 361
35.1.4 Relational model . . . 362
35.1.5 Dimensional model . . . 364
35.1.6 Object database model . . . 364

35.2 Time series database . . . 365
35.3 Database interface (API) . . . 365
35.3.1 ODBC . . . 365
35.3.2 SQL . . . 365
35.3.3 WDB . . . 365
35.3.4 DBI . . . 366
35.4 Process database system . . . 366

36 SQL Database **367**
36.1 Standardization . . . 367
36.2 SQL data types . . . 367
36.3 SQL keywords . . . 368
36.3.1 Data retrieval . . . 368
36.3.2 Data manipulation . . . 369
36.3.3 Transaction Controls . . . 369
36.3.4 Data definition . . . 369
36.3.5 Data control . . . 370
36.4 SQL commands . . . 370

37 Database design **371**
37.1 Data modeling . . . 371
37.1.1 Entity-relationship model . . . 372

38 Data Privacy **373**
38.1 GDPR . . . 373

XII Alarm systems **375**

39 Introduction **376**

40 Management **380**
40.1 Introduction . . . 380
40.2 Alarm Philosophy . . . 381

41 Specifications **383**
41.1 Introduction . . . 383
41.2 Alarm System Functions . . . 383
41.2.1 YA-711 specifications . . . 383
41.2.2 IEC 62682 specifications . . . 384
41.3 Barriers . . . 385
41.4 Alarm Types . . . 385
41.5 Alarm Levels . . . 385
41.6 Alarm generation . . . 386
41.7 Alarm Priority . . . 386
41.8 Alarm colors . . . 386
41.9 Alarm States . . . 387
41.10 Alarm Shelving . . . 389
41.11 Response Time . . . 390
41.12 Performance . . . 391

42 Configuration **394**
42.1 Dead band . . . 394
42.2 Hysteresis . . . 394
42.3 Alarm limits . . . 394
42.4 Fail/Safe . . . 395
42.5 Alarm list . . . 395
42.6 Alarm devices . . . 395
42.7 Alarm logging . . . 396

43 Alarm system advises **397**

XIII Multitasking systems and Real-Time requirements **398**

44 Introduction **399**
44.1 Maximum time . . . 400
44.2 Real-Time Examples . . . 400
44.2.1 Example #1: Buffertank . . . 400
44.2.2 Example #2: Data Communication . . . 401
44.3 Operating System . . . 401
44.4 Synchronization and multitasking . . . 401
44.5 Resources . . . 402
44.6 Programming . . . 402
44.7 Embedded system . . . 403
44.8 CPU and micro controller . . . 404
44.9 Sequence Diagram . . . 404

45 Multitasking **406**
45.1 Introduction . . . 406
45.2 States . . . 408
45.3 Scheduler queues . . . 409
45.4 Strategies . . . 409
45.5 Performance . . . 410

46 Synchronization **412**
46.1 Mutex . . . 412
46.2 Semaphore . . . 413
46.3 Syncronization example . . . 414
46.4 Events . . . 416
46.5 Interprocess communication (IPC) . . . 416
46.5.1 Events . . . 417
46.5.2 Pipes . . . 417
46.5.3 Shared memory . . . 417
46.5.4 Message queue . . . 417
46.6 Communication Protocols . . . 417
46.6.1 Token Ring . . . 417
46.6.2 CSMA/CD . . . 418

47 Resources **420**
47.1 Deadlock . . . 420
47.2 Critical region . . . 422

48 Software modules **424**
48.1 Instruction time . . . 425
48.2 Software application . . . 427
48.2.1 Process . . . 427
48.2.2 Thread . . . 427
48.2.3 Task . . . 428
48.3 Core and Multicore . . . 429
48.4 Input monitoring . . . 429
48.4.1 Interrupt operation . . . 431
48.4.2 Interrupt example . . . 431
48.5 Priority Inversion . . . 431
48.6 Watchdog . . . 432

49 Real-time specifications **434**
49.1 Descriptions . . . 434
49.2 Properties . . . 435

50 Programming **436**
50.1 Introduction . . . 436
50.2 Memory allocation . . . 436
50.3 Posix.4 . . . 437
50.4 Multitasking examples . . . 437
50.4.1 Threading . . . 437
50.4.2 Traffic light controller . . . 439

51 Operating systems **448**
51.1 RTOS requirement . . . 448
51.2 Driver . . . 450
51.3 Windows / Linux . . . 450
51.3.1 Windows history . . . 452
51.3.2 Windows Embedded . . . 452
51.3.3 Windows CE, Windows Embedded or Linux . . . 452
51.4 QNX . . . 453

52 Design of RTOS tasks **454**
52.1 Benefits of any RTOS . . . 454
52.2 Selection of a real-time system . . . 454
52.3 Design of a real-time system . . . 455

XIV Embedded systems **456**

53 Introduction **457**
53.1 Properties . . . 459
53.2 Real-Time Systems . . . 460
53.3 Watchdog . . . 460
53.4 WEB user interfaces . . . 460
53.5 CPU platform . . . 461
53.6 Debugging . . . 461
53.7 Start-up . . . 462
53.7.1 Self Test . . . 462
53.7.2 Boot Loader . . . 462
53.8 Reliability . . . 463

54 Power **464**

55 Embedded software architectures **465**
55.1 Simple control loop . . . 465
55.2 Round Robin . . . 466
55.3 Round Robin with Interrupt . . . 467
55.4 Non preemptive multitasking . . . 467
55.5 Preemptive multitasking . . . 467

56 Operating system **469**
56.1 Device driver . . . 469
56.2 Linux . . . 469
56.3 Windows CE . . . 470
56.4 Windows Embedded . . . 470
56.5 eCOS . . . 471
56.6 Mobile Phones and PDA . . . 471
56.7 Storage . . . 472
56.8 Memory . . . 473
56.9 Misc . . . 473

57 Development **474**

XV Ex / ATEX 475

58 ATEX and Ex 476
58.1 Introduction . . . 476
58.2 Definitions . . . 479
58.2.1 Explosive atmosphere . . . 479
58.2.2 Hazardous area . . . 479
58.2.3 Explosion . . . 479
58.2.4 Potentially explosive atmosphere . . . 479
58.2.5 Equipment . . . 480
58.2.6 Gas explosion . . . 480
58.2.7 Risk . . . 480
58.2.8 Risk Assessment . . . 480
58.2.9 Hazard Identification . . . 480
58.2.10 Risk Estimation . . . 480
58.2.11 Risk Evaluation . . . 480
58.2.12 Risk Reduction Option Analysis . . . 480
58.2.13 Residual Risk . . . 480
58.2.14 Critical Ignition Energy . . . 481
58.2.15 Most Easily Ignited Concentration (MEIC) . . . 481
58.2.16 Ignition Sources . . . 481
58.3 Standards and Certifications . . . 481
58.3.1 IEC . . . 481
58.3.2 CENELEC . . . 481
58.3.3 BSI . . . 481
58.3.4 ATEX . . . 481
58.4 ATEX equipment directive . . . 482
58.5 Equipment protection . . . 483
58.6 Temperature classes . . . 484
58.7 Intrinsic safety . . . 484
58.7.1 Design of simple IS systems . . . 486
58.7.2 Maintenance, repair and test of IS systems . . . 487
58.7.3 IS systems standards . . . 487
58.8 Cetificate . . . 488
58.9 Process . . . 488
58.10 Ingress Protection (IP) Ratings . . . 490
58.11 IP coding . . . 490

XVI Testing 492

59 Testing 493
59.1 Introduction . . . 493
59.2 Test Plan . . . 493
59.3 Test Levels . . . 496
59.3.1 Unit or Device Testing . . . 496
59.3.2 Module testing . . . 496
59.3.3 System Testing . . . 497
59.4 Factory Acceptance Testing (FAT) . . . 497
59.4.1 Types of acceptance testing . . . 497
59.5 Positive or Negative Testing . . . 498
59.6 Accelerated testing . . . 498
59.7 Testing tools . . . 499
59.7.1 Simulation . . . 499
59.7.2 Emulation . . . 499
59.7.3 HIL . . . 499
59.7.4 HALT . . . 500
59.8 Software testing . . . 500

60 Maintenance **501**
60.1 Condition Monitoring . . . 501
60.2 Predictive maintenance . . . 503
60.2.1 Handling the data . . . 503
60.2.2 Machine Learning . . . 503
60.2.3 Challenges . . . 504

XVII Data Recovery **506**

61 Data Recovery Introduction **507**
61.1 Disk drives . . . 507
61.2 File systems . . . 508

62 RAID **509**
62.1 Introduction . . . 509
62.2 RAID levels . . . 510

63 Backup and Restore **513**
63.1 Introduction . . . 513
63.1.1 Unstructured . . . 513
63.1.2 Full + Incremental . . . 513
63.1.3 Continuous data protection . . . 513
63.2 Devices . . . 514
63.2.1 Magnetic tape . . . 514
63.2.2 Hard disk . . . 514
63.2.3 Optical disc . . . 514
63.2.4 Floppy disk . . . 514
63.2.5 Solid state storage . . . 514
63.2.6 Remote backup service . . . 514
63.3 Strategies . . . 515
63.3.1 Advises . . . 515
63.3.2 Reasons for backup . . . 515
63.3.3 Proposal . . . 515

64 UPS **517**
64.1 Introduction . . . 517
64.2 Function . . . 517
64.3 Rating . . . 519
64.4 Location . . . 519
64.5 Software . . . 520

65 Computer Room Design **521**

XVIII System Design **522**

66 Systems Design **523**
66.1 Introduction . . . 523
66.2 Collection of requirements . . . 525
66.3 Prototypes . . . 525
66.4 Pseudo coding . . . 526
66.5 Architecture . . . 526
66.5.1 SysML and UML diagrams . . . 526
66.5.2 Context diagram . . . 527
66.5.3 Logical Architecture . . . 527
66.5.4 Physical Architecture . . . 527
66.5.5 Big Data Architecture . . . 527
66.5.6 Hardware . . . 527

66.5.7 Grounding . . . 529
66.5.8 Interface . . . 530
66.5.9 Software . . . 530
66.5.10 Filtering . . . 531
66.5.11 Platform . . . 531
66.5.12 Service Oriented Architecture . . . 531
66.6 Development . . . 531
66.6.1 Development stack . . . 531
66.6.2 Communication Stack . . . 532
66.6.3 Database Modeling . . . 532
66.6.4 Hardware modules . . . 532
66.6.5 Software Modules . . . 532
66.6.6 Data Flow . . . 533
66.7 Models . . . 533
66.8 Documentation . . . 533
66.8.1 Flow charts . . . 534
66.8.2 UML diagrams . . . 535
66.8.3 N2 diagrams . . . 535
66.9 Summary of Systems Design Steps . . . 535
66.10 Project Management . . . 536
66.11 Advices . . . 536
66.12 Design of a measurement system . . . 536

XIX Documentation 540

67 Guidelines for reports and projects 541
67.1 Reports . . . 541
67.1.1 Informal report . . . 541
67.1.2 Formal report . . . 541
67.1.3 References . . . 542
67.1.4 Article or paper . . . 542
67.2 Presentation . . . 542
67.3 Meetings . . . 543
67.3.1 Non formal meetings . . . 543
67.3.2 Formal meetings . . . 543
67.3.3 Effective meetings . . . 544
67.4 Projects . . . 544

XX Appendix 546

A OPC Example 547

B MySQL Example 553
B.1 MySQL installation . . . 553
B.2 MySQL example . . . 553

C Alarm System Example 558

D Test Plan Example 563

Bibliography 567

Index 573

Preface

This document contains the main parts of the syllabus for the IIA2017 Industrial Information Technology (IT) course at University of South-Eastern Norway (USN). This is a 10 ECTS (credits) course given in the "Industrial IT and Automation" (IIA) master program.

The front page shows a picture of different processes that can be monitored and controlled by information technology, a house, an industrial process and a car. This document will give an overview of IT systems, modules and devices used in the industry for designing process control systems. The focus will be on interconnection between devices, modules and systems, data communication, PLC, databases, embedded system, multitasking system with real-time requirements, alarm system, simulation and testing. Both the information technology (IT) and operational technology (OT) parts of process control systems or industrial controls systems are covered.

This document consists of several sections important for understanding the usage of IT system in the industry, designing and making extensions of such systems and/or maintaining such systems. No single book covers these sections, the contents are a collection of several books and the industrial experience of the author.

The history of this document is:

Revision	**Changes / Extensions**	**Who**	**Date**
0.1	Merged all lecture notes parts into one single document.	NOS	$MAR-08$
0.2	New structure of the real-time part, and minor fixes.	NOS	$MAR-08$
0.3	Add alarm system section and minor extensions to other other sections	NOS	$DEC-08$
0.4	Add sections about testing, data recovery, and EEx	NOS	$JAN-09$
0.5	Minor changes to data communication, fieldbus, and real-time sections.	NOS	$FEB-09$
0.6	Add the System Engineering part, minor changes and extensions to other parts.	NOS	$JAN-11$
0.7	Minor updates/changes for most of the sections	NOS	$JAN-12$
0.8	Add the System Design part, minor updates/changes to other parts.	NOS	$JAN-13$
0.9	Minor updates/changes, moved some sections to appendix	NOS	$JAN-14$
1.0	Minor updates/changes, reduced the number of chapters.	NOS	$NOV-14$
1.1	Minor updates/changes, included Token Ring figures.	NOS	$DEC-15$
1.2	Minor changes to most sections, add Instrumentation part.	NOS	$DEC-16$
1.3	Add sub section about GDPR and "select protocol".	NOS	$DEC-17$
1.4	Extend the real-time section with multitasking.	NOS	$APR-18$
1.5	Add Digitalization part, and extend Security part	NOS	$JAN-19$
1.6	Extend digitalization and security parts, add alarm shelving, minor updates to other parts.	NOS	$DEC-19$
1.7	Add section about data communication in C# and autonomous systems, and minor updates/fixes.	NOS	$DEC-20$
1.8	Minor changes of multitasking, autonomy and security sections	NOS	$JAN-22$

©Nils-Olav Skeie: Permission is granted to distribute single copies of this lecture notes for non-commercial use, as long as it is distributed as a whole in its original form, and the name of the author is mentioned.

Nomenclature

This section gives a list of abbreviations used in this lecture notes.

Abbreviation	**Meaning/Explanation**
AC	Alternating Current
ATA	AT Attachment (same as IDE or PATA)
AWG	American Wire Gauge
CAN	Controller Area Network
CAT5	Category 5 cable, also known as "Cable and Telephone"
CODASYL	Conference on Data Systems Languages
COM	Component Object Model
CPU	Central Processing Unit
DBMS	Database Management System
DC	Direct Current
DCE	Distributed Computer Equipment
DCL	Data Control Language
DCOM	Distributed Component Object Model
DCS	Distributed Computer System
DDL	Data Definition Language
DML	Data Manipulation Language
ERP	Enterprise Resource Planning
EEMUA	Engineering Equipment Material Users Association
FF	Fieldbus Foundation
GMT	Greenwich Mean Time
GPOS	General Purpose Operating System
GUI	Graphical User Interface
HART	Highway Addressable Remote Transducer (Protocol)
HIL	Hardware in the loop
HMI	Human Machine Interface
I/O	Input and Output
IDE	Integrated Drive Electronics (same as ATA/PATA)
IIoT	Industrial Internet of Things
IO	Integrated Operations
IoT	Internet of Things
IPC	InterProcess Communication
ISA	International Society of Automation
LAN	Local Area Network
LED	Light Emitted Diode
LON	Local Operating Network
MAN	Metropolitan Area Network
MCU	MicroController Unit
MIME	Multipurpose Internet Mail Extensions
MMI	Man Machine Interface

Abbreviation	**Meaning/Explanation**
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
NOS	Network Operating System
OBD	On-Board Diagnostic
ODMG	Object Data Management Group
OLAP	On Line Analytic Processing
OLE	Object Linking and Embedding
OPC	OLE for Process Control (Open Process Control)
OSI	Open System Interconnection
PAC	Programmable Automation Controller
PC	Personal Computer
PDA	Personal Digital Assistant
PID	Proportional Integral Derivative (controller)
PLC	Programmable Logic Controller
POSIX	Portable Operating System Interface
PROFIBUS	Process Fieldbus
PWM	Pulse Width Modulation
RAM	Random Access Memory
ROM	Read Only Memory
RTD	Resistance Temperature Detectors
RTOS	Real-time operating system
RTS	Real-Time System
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SCSI	Small Computer System Interface
SIL	Safety Integrity Level
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SSID	Service Set Identifier
SQL	Structured Query Language
TSDS	Time Series Database Server
UML	Unified Modeling Language
UTC	Coordinated Universial Time
WAN	Wide Area Network
WEB	World Wide Web
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

Part I

Introduction

Chapter 1

Introduction

This part contains an introduction to the main areas of using and combining IT technology in the industry. The main focus will be Process Control Systems (PCS) like SCADA (Supervisory Control And Data Acquisition) systems and the main building blocks of such a system. These building blocks will be both hardware devices and software modules, and the interactions of these blocks. The motivations will be to have the knowledge to be able to combine the blocks to design a good PCS best suited for a specific system specification. Figure 1.1 indicated the process of developing such a system.

Industrial IT systems consists of hardware and software buildings blocks, these blocks can either be bought ready made or it must be developed for the PCS. Often Commercial off the self (COTS) devices are used for the hardware and the software is a combination of COTS and development especially for the PCS. This also means that knowledge about software development will be very important when developing and designing PCS.

Industrial IT used in Cybernetics, using IT systems to monitor and control in systems, including "systems" alive. Cybernetic means mate, defined by (Wiener 1961). Cybernetics is mainly about feedbacks, by measuring or estimation of the current state, combined with information about "noise" factors, the system will be able to calculate any wanted outputs. The outputs must be used to control the system, it will be a self adjusting system.

1.1 Overview

The focus will be an introduction for using information technology modules and devices for designing systems to be used in the industry, especially the process industry. The motivation is for designing and using systems for controlling processes in the industry, and requires measurements, computer systems and control elements like valves and actuators to mention some. Measurements meaning getting the sensor values from the sensor devices, mounted anywhere in the process, into a computer. The measurements have to be analyzed and evaluated, these values have to be trusted as they are used as input for controlling the system. The measurements started out as single devices with mainly $4 - 20\,\mathrm{mA}$ interfaces. However more and more information is needed, especially for maintenance and calibration, so more advanced interfaces are needed. The Hart (Highway Addressable Remote Transducer) protocol extend the $4-20\,\mathrm{mA}$ interfaces, but still limiting the amount of information that can be exchanged. The next step is a digital protocol, including process buses.

The measurements can be presented using some sort of graphical user interface[1] on the computer system(s), and/or used as input for the control system for the control elements in the process. The control system will also include an alarm system and a system for generating a set of reports. A database is

[1] GUI, HMI or MMI (Graphical User Interface, Human Machine Interface, or Man Machine Interface).

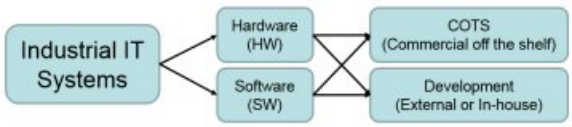


Figure 1.1: The process of developing a PCS.

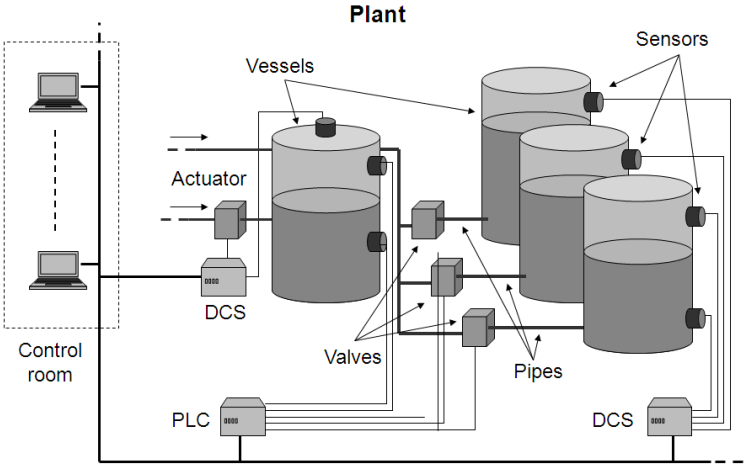


Figure 1.2: An overview of a small plant with some vessels with sensors, valves, and actuators.

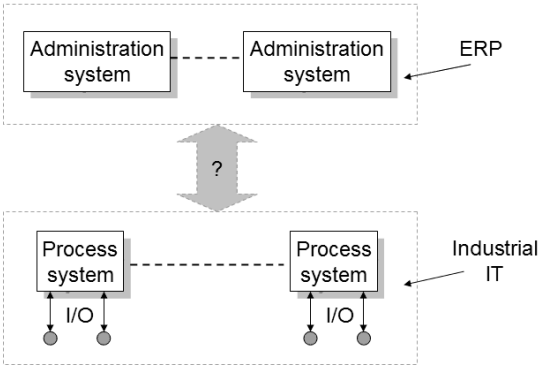


Figure 1.3: The connection between an industrial IT system and an ERP.

needed to be able to make reports with historical information. These computer systems can also referred to as a Process Control Systems (PCS), Distributed Control System (DCS) or SCADA system.

SCADA refers to a large-scale, distributed measurement and control system. SCADA systems are used to monitor and/or to control chemical or transport processes, water supply systems, control electric power generation, transmission and distribution, gas and oil pipelines, and other distributed processes (Olsson & Piani 1998). A SCADA system includes input/output signal hardware, controllers, human machine interface (HMI), networks, communication, database and software. It mainly comes in the branch of Instrumentation Engineering. A SCADA system can use Programmable Logic Controllers (PLC), Programmable Automation Controllers (PAC), and Distributed Control Systems (DCS). PLC is used mainly for digital I/O, DCS is used mainly for analog I/O, and PAC is a combination of a Personal Computer (PC) and a PLC and can be used for both analog and digital I/O.

An example of a plant is shown in Figure 1.2, showing these devices together with some vessels, pipes and valves. The system in the control room(s) are collection the sensor information from among others DCS, PAC, and PLC, displaying the information and using the information to control the process using among others the PLC and PID controllers. PID controllers can also be a part of a PLC, PAC or DCS.

It is more and more common that Enterprise Resource Planning (ERP) systems also want (or need) information from the industrial IT system as shown in Figure 1.3. The focus in this document will be on industrial IT systems, but will mention ways of exchanging informations with ERP. The reason for the focus on the industrial IT part will be in order to have validated values into the ERP section must the values and/or information from the industrial IT system be trusted. The focus will be on how to get trusted values within the industrial IT system.

The motivation for reading this document will be to know how the different "sub" systems are working and how they can be interconnected to be a complete control system. Figure 1.4 shows the

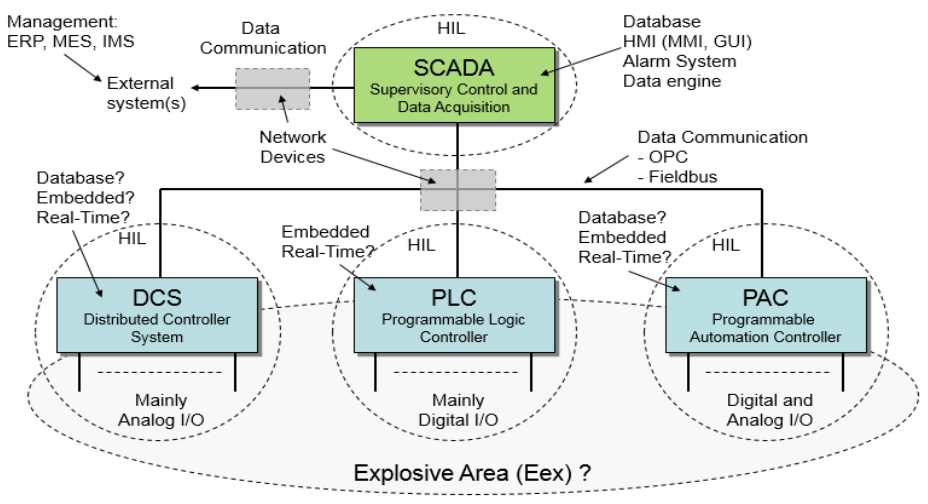


Figure 1.4: An overview of a Process Control System (PCS) indicating the contents covered by this document.

different sections of this document containing SCADA, PLC, PAC, and DCS systems, some technologies and functions of these sub systems, and different types of communications between these systems. The complete system is also known as a Process Control System (PCS) or an Industrial Control System (ICS).

Part II

Systems Engineering

Chapter 2

Systems Engineering

2.1 Introduction

Systems engineering is a methodical, disciplined approach for the design, realization, technical management, operations, and retirement of a system (Kapurch 2007). A "system" is a collection of different elements that together produce a result or results that are not possible by any of the elements alone (Kossiakoff 2003). The interconnection of the parts and the relationship among these parts give the added value as a whole system, and systems engineering is the logical way of think to achieve the goals for any system.

Systems engineering seeks a safe and balanced design in the multiple interests and requirements for complex systems. It consists of several fields that must be combined to be able to design complex systems and managed complex engineering projects. Systems engineering consists of several subject and areas like (Kapurch 2007):

- System designs; Requirements Definition and Technical Solution definition,
- Product Realization; Design realization, Evaluation, and Product Transitions,
- Technical Management; Planning, Control, Assessment, and Decision Analysis,
- Planning,
- Risk Management,
- Configuration Management,
- Data Management,
- Assessment,
- Decision Analysis.

2.2 Scope

The process of systems engineering is mainly product development and product maintenance. Product development and product maintaining is balancing organizational and technical interactions in developing new systems or improving existing systems. Systems engineering for a product will be active for the lifetime of the product, from collecting the first requirements of the first version until the end of the product. Systems engineering is not project management. Project management is the discipline of planning, organizing, securing and managing resources for a successful completion of specific project goals and objectives (Kossiakoff 2003). Systems engineering is not project control. Project control is that element of a project that keeps it on-track, on-time and within budget (Kossiakoff 2003).

Project management and project control will always be part of systems engineering as projects will be defined to develop the first version of a project, next version, and product maintaining. More details regarding relation between systems engineering and project is shown in Figure 2.1. This figure shows that system design, product realization and technical management being important parts of systems engineering will not be part of any project.

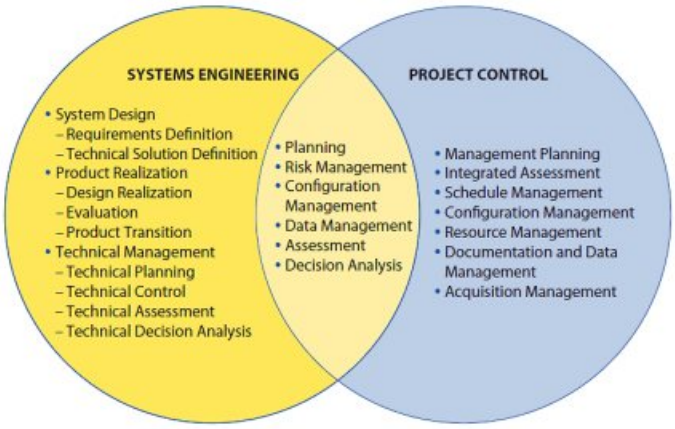


Figure 2.1: The relation between system engineering and project management (Kapurch 2007).

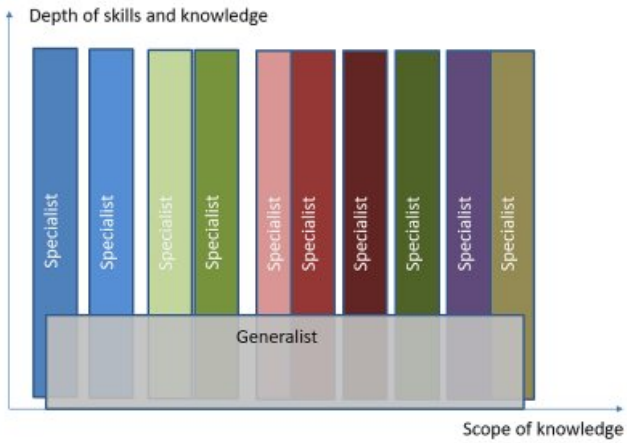


Figure 2.2: The combination of generalistics and specialists for designing larger systems (Muller 2012).

2.3 Complex systems

Systems engineering is useful for larger systems, especially complex systems, where several different areas must be combined to make a complete systems. In these cases a complete system will consists of a huge amount of sub systems that must be interconnected the correct way. One challenge with larger systems is the usage of several specialists for each sub section and combining the knowledge and experience from these specialists to design such a system. In these cases a combination of both one or several generalists and the specialists must be used, and system engineering is important for the design (Muller 2012). This combination is shown in Figure 2.2 where the systems engineering is covered by a combination of both generalists and specialists.

Systems engineering is mainly product development and product maintenance shown in Figure 2.3. The figure shows the steps for developing and maintaining a system consisting of a combination of hardware and software sub systems and/or modules. The same steps will also be used for the hardware and software parts in the system.

Figure 2.3 shows an overview model for system development and maintenance, several methods or models can be used like the waterfall method or agile methods. One model that can be used for both system design, hardware design and software design is the V model, shown in Figure 2.4.

2.4 System design

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems related to industrial information technology (IT) consists of both hardware and software modules, interconnected in many different ways. For such systems will the design be divided into the hardware design and software design parts, but both parts will

Figure 2.3: The life cycle of a system from planning to maintenece, and planning of either an upgrade or a new version of the system (www.svitla.com/blog: Nov-2019).

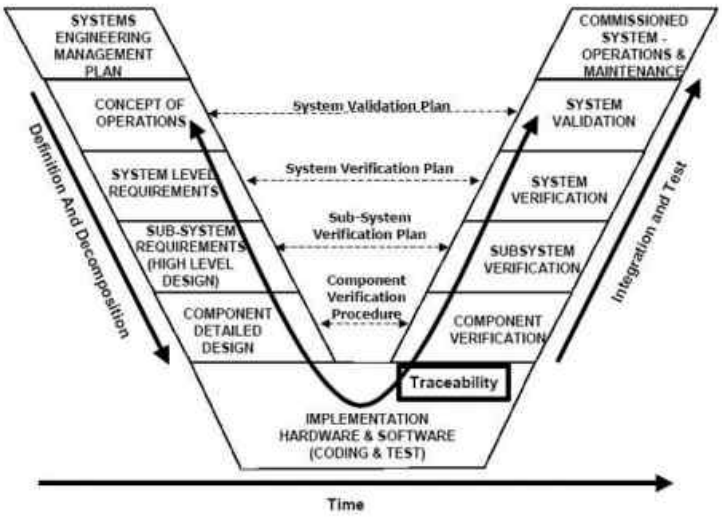


Figure 2.4: The V system design model that can be combined with different development process methods (www.engineering.com: Nov-2019).

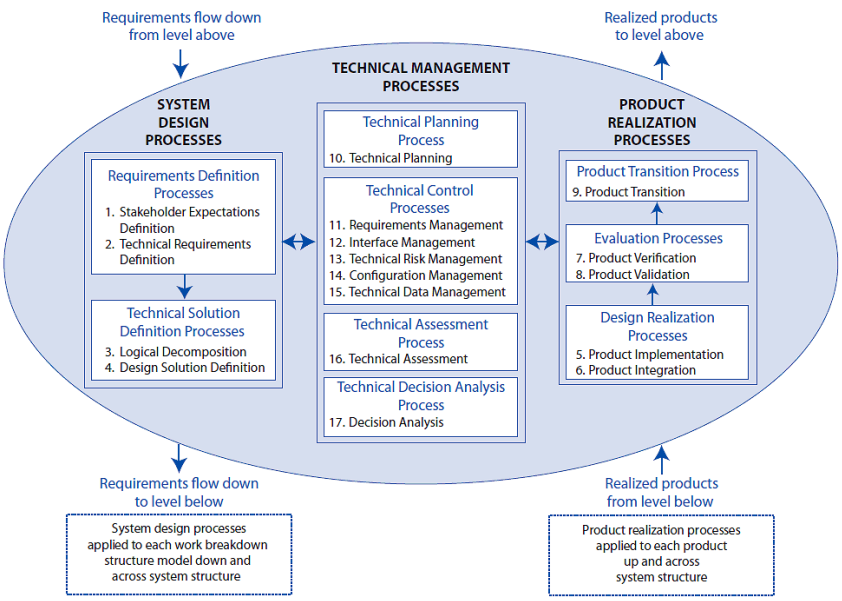


Figure 2.5: The functions, interactions, and flows of system engineering (Kapurch 2007).

normally be analyzed, designed, developed and tested in parallel. System design starts with collection of requirements, requirements describing the hardware, the software, and how the system is going to interact with the "rest of the world". Hardware devices normally end up with a ready made modules, commercial, off-the-shelf (COTS) modules, while software design normally end up with some development. Normally hardware devices contain one or several micro controllers, and will need software to fulfil the requirements. The system design process part of the systems engineering is shown in Figure 2.5, where systems engineering focus on complex systems.

2.4.1 Product realization

The product realization process consists of planning, collecting of requirements, design and development, purchasing, production and service provision, and control of monitoring and measuring equipment. The product realization process part of the systems engineering is shown in Figure 2.5. A system can have several project phases during the life time of the product, first a development project to make the first version, then several update/change projects to modify the product. These modifications can be due to change or extension of requirements, change or extension of functions, devices and/or modules not supported any more, or change of business. A project should contain the following steps; a) collection of requirements, b) analysis, c) design, d) construction, e) test, and f) deployment.

2.4.2 Technical Management

The management of resources. The technical management part of the systems engineering is shown in Figure 2.5.

2.4.3 SysML

System Modeling Language (SysML) is an extension to the Unified Modeling Language (UML) to support specification, collection of requirements, analysis, design and verification of complex systems. SysML provides two levels, Basic and Advance, adding elements to the model like use cases, constraints, and system structure to mention some. The advance level include more elements for physical structures and system properties to mention some.

2.4.4 Interconnections

A system consists of many devices and modules. The system can only fulfill the requirements if these devices and modules are exchanging information, and an interface is needed for this exchange of infor-

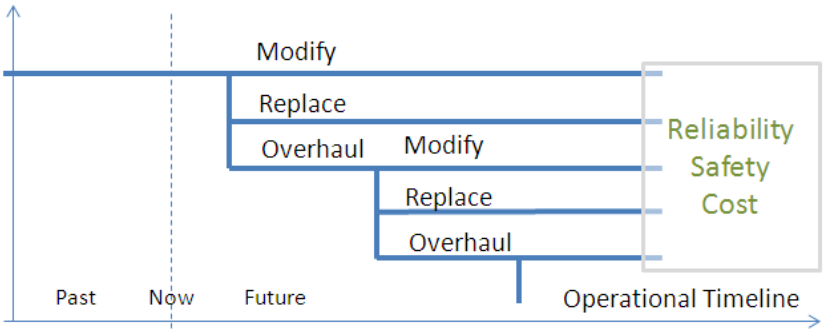


Figure 2.6: The ageing of systems.

mation. The interface will consists of both a hardware and a software description or specification, a set of rules that must be fulfilled for the systems to exchange information. An important challenge in system design, and systems engineering, are these interfaces and be sure that all the sub systems are communication. Normally the specialists, see Figure 2.2, will only focus on the sub systems within their domain. In this case this will be the responsibility of the project leader and the generalists to be sure that the sub systems will communicate across these domain borders. Each of the devices or modules work as expected, however the system is still may not working as expected due to missing interconnections or sharing of information.

2.4.5 Ageing

The management of ageing and life extension is also a part of systems engineering. What to do when a system is approaching the end of the design life?

A continued usage requires management of structures, systems and components. Ageing refers to changes in the operational state of these structures, systems and components which suffer from degradation, inefficiency or obsolescence. The question will be to maintain the current system or design a new system for the operational time horizon. This will result in an evaluation of several alternative strategies like:

- continue to overhaul and repair the existing system,
- replace the existing system with a new system,
- modify the existing system.

Aging is not a binary process, but at some time the system needs attention and where actions should be considered. These actions will be according to Figure 2.6 showing the actions of the strategies.

Downtime is always the biggest fear when companies make a decision about whether to upgrade or replace the system, or part of the system. So, the decision about whether upgrading is better than like-for-like replacing, will vary from company to company. In many cases the two solutions need to be used in parallel to minimize the downtime.

2.4.6 History

The ageing part must also consider the history of the system (and the company). A system contains both hardware and software, and often many elements. Should we replace all elements when developing a new system? Take the backward compatibility into consideration, can part of the elements from an old system be reused in a new system? This means that existing customer already installed the old system but installed at a newer date so an replacement is not necessary, want to upgrade the system to take advantage of new architecture and new feature.

Software that the customer has developed to run on the old system, can it be reused on the new system? A company wants to make innovation into the development but also keep new systems being backward compatible. History of the systems will complicate the development process, however both old and new customer will always be important for any company (or industry).

Figure 2.7: The research and development processes.

2.5 Research and development

Systems Engineering is part of development (Parnell, Driscoll & Henderson 2012). Research and Development is the process of bringing the world further on. The process are:

- Research; Unfocused effort that may lead to a focused design project, check if a solution is possible or not,
- Development; Making new or changes to a device or process, use a possible solution to make a product.
- Design; process for development.

Figure 2.7 shows the research and development process. The Figure shows that the design process is making a product, and the design process may or may not be depending on research and/or development. A new product can be developed only by a design process, just copying the an old design, or development can be added to enhance the product, and/or research can also be used.

Summary: Systems Engineering is the system approach to the design, development and maintenance of products and systems with focus on larger and/or complex systems.

Part III

Process control system

Chapter 3

Automation systems

The industry is using automation systems and information technology to reduce the human interaction. The reasons can be to reduce or remove boring tasks, to increase the "average" quality of the produced material, and to reduce the production cost. More and more devices and modules from IT systems are used, and a PC is preferred due to the low price and the software available. The learning time and cost of the operators are also important and using a system that the operators are familiar with will be an advantage.

An automation system can be either a monitoring system or a control system. A monitoring system is just for monitoring a process and contains only input modules and some sort of HMI (Human Machine Interface) devices for user information. The control system consists of the monitoring system used for input of informations and an output part for controlling the process, together with some sort of HMI devices for user information. The automation systems consists of hardware devices and software modules, and are design using information technology. These systems can be stand alone systems consisting of only a single computer or distributed systems consisting of several, tens or even hundreds of computers interconnected in different ways.

The automation system will interact with some type of process as shown in Figure 3.1. The process can be any type of process. Normally we think of an industrial process but the process can also be a human, a building or a car to mention some examples. Often there will be a requirement that the administration system should also be updated based on information from the automation system.

3.1 Automation

Automation is the use of control systems and information technologies to reduce the need for human work in production of goods and services. Information technologies are used for systems design like computers, communication devices, and software modules. Automation systems are today based on digital information so converting analog information into digital information (digitize) is an important part of automation systems. Some of the reasons for having an automation system can be to:

- perform operations that human are not able to do,
- perform routine procedures that are boring for humans,
- remove operations that are dangerous for humans,
- get a higher quality on the products (good repeatability),
- continuously monitor the processes,

Figure 3.1: An automation system is interacting with some type of process, monitoring and often controlling the process. The automation system may also be connected to the administration system.

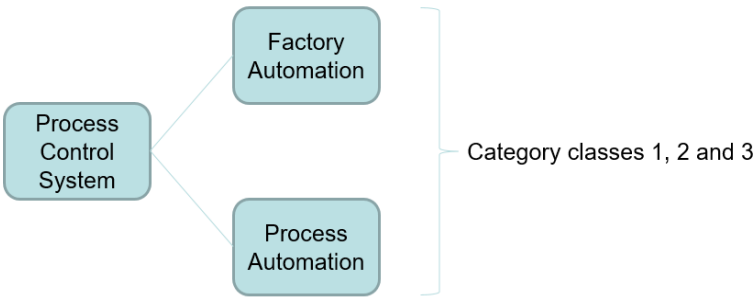


Figure 3.2: A control system within the category classes 1, 2, and 3 is divided into two different type of automation systems, process automation and factory automation.

- reduce the operational cost, but increasing the investment costs.

Automation systems or process automation systems are used to automatically control processes. Such an automation system can be divided into either safety, control and/or monitoring categories. The type of categories are:

Class	Category	Application	Description	Focus
0	Safety	Emergency action	Always critical,	
1	Control	Closed loop regulatory control	Often critical, regulations,	X
2		Closed loop supervisory control	Usually non-critical,	X
3		Open loop control	Human in the loop,	X
4	Monitoring	Alerting	Short term operational consequence,	
5		Logging and downloading/uploading	No immediate operational consequence.	

The focus in this document will be classes 1, 2, and 3 as a process control system is normally an automation system within these category classes.

Automation systems are divided into **1)** process automation and **2)** factory automation systems as shown in Figure 3.2.

Process automation can be characteristic of 1) continuous chemical processes with analog measurement signals and control, 2) part of the equipment and/or devices are located in explosive atmosphere, 3) relative slow response time, 4) typical measurements like temperature, pressure, flow, and level, 5) typical actuators like pumps and control valves, and 6) usage of PID control loops.

Factory automation can be characteristic of 1) discrete production with sequence of digital measurement signals, 2) relative fast response time, 3) typical measurements like on/off signals, positions, valid/not valid signals, 4) typical actuators like magnetic valves and on/off motors, and 5) usage of sequential control.

Some of the challenges for automation systems are:

1. increase investment costs,
2. decrease flexibility; difficult with individual adoption and fast changes of products,
3. decrease creativity; computers have difficulties solving unknown problems/challenges,
4. increase need for support and maintenance.

3.2 Control system

A control system or a process control system is a device or set of devices used for managing the behavior of another system or a process, normally based on automation systems the category 1 to 3. The control system will take the input from one or several sensing devices and perform some sort of action on a set of actuators, depending on input signals and algorithms (Stenerson 2003),(Olsson & Piani 1998). Two different types of control system exists, logic control and feedback control.

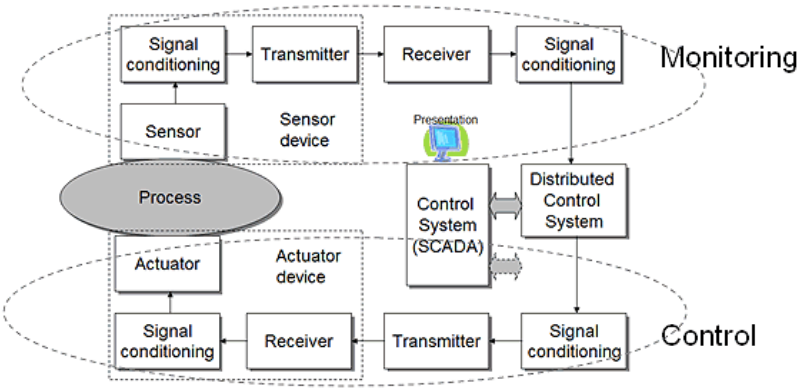


Figure 3.3: The main functions for a process control system; monitoring, control and interconnections.

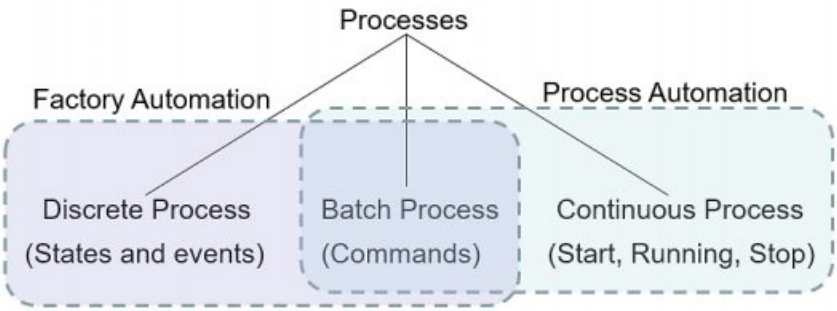


Figure 3.4: The type of processes that can be monitored and/or controlled by a process control system (PCS).

A logic control system, or sequential control system, is responding to "simple" input signals, often on/off signals. Normally these systems perform a sequence of operations due to an input signal, the reason why they are called sequential control systems. A feedback control system, or linear control system, is using "continuos" feedback signals from the system for the control algorithm. A PID controller is an example of such a controller where the difference between a setpoint signal and the feedback signals are used for control. Note that some physical systems may not be controllable. Fuzzy logic is a combination of logic control and feedback control to combine some of the design simplicity of logic control with the utility of feedback control.

An automated system is a collection of devices working together to accomplish tasks or produce a product or family of products. The main functions for a control system is shown in Figure 3.3 where a control system is connected to a SCADA system.

The main functions of a process control system is shown in Figure 3.3. These functions are:

1. process monitoring; modules for collecting and interpretation of data from the plant,
2. control; modules for controlling some parameters for the plant,
3. connections; interconnections between the process monitoring and control for processing of input and output data, for feedback and automatic control.

3.3 Process control system

A process control system will monitor and control some sort of process. Processes can be described by their starting and stopping points, and by the kinds of changes that take place in between. The type of processes are shown in Figure 3.4 and are:

- **discrete**; Manufacturing, motion and packaging systems. Robotic assembly are also characterized as discrete process control. Most discrete manufacturing involves the production of discrete pieces of product,

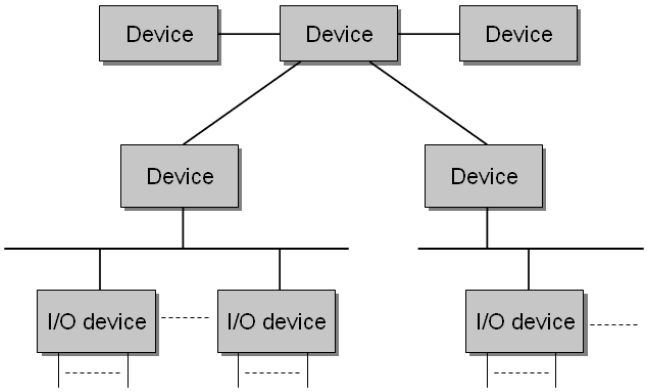


Figure 3.5: A distributed computer system.

- **batch**; Batch jobs are saved (queued) in the system during working hours and executed during the evening or whenever the computer is idle. Batch processing is particularly useful for operations that require the computer or a peripheral device for an extended period of time. Once a batch job begins, it continues until it is done or until an error occurs. Note that there is no interaction with the user while the batch program is being executed,
- **continuous**; Often, a physical system is represented through variables that are smooth and uninterrupted in time. Such systems will be running most of the time meaning that it will be active 24 hours a day, 7 days a week ($24x7$ system). Continuous processes, in manufacturing, are used to produce very large quantities of product per year,
- **hybrid**; Applications being some sort of combination of discrete, batch and continuous process control.

3.4 Distributed Systems

An industrial system can be a single computer system or a distributed system with several devices interconnected. A single system is often used on very small processes or plants, but normally several different IT systems are cooperating for solving the monitoring and/or controlling tasks. The reason for using several systems or devices are:

1. to exploit the functions of different systems or devices,
2. redundancy,
3. better overview and structure,
4. troubleshooting.

The drawback of using several systems are the price and a more complex system. No perfect single system exists for controlling purposes, so a cooperation between different systems is preferred. Such a system is shown in Figure 3.5.

A distributed control system (DCS) refers to as a control system usually of a manufacturing system, process or any kind of dynamic system, in which the controller elements are distributed throughout the system with each component sub-system controlled by one or more controllers. The entire system of controllers are connected by some sort of networks for communication and monitoring (Olsson & Piani 1998). The DCS is sometimes referred to as the first digital revolution, and has today computational power to handle several layers of the automation hierarchy (Isaksson, Harjunkoski & Sand 2018).

A distributed system will have a more complex system structure than a single system, but each sub device will end up with a more simple structure than a single system. The distributed system can also have a higher degree of redundancy.

One example of a distributed system is shown in Figure 3.6. The system contains several controllers for measurement and control of different parts in the plant, local displays, a local area network for sharing information and several display systems for remote operations.

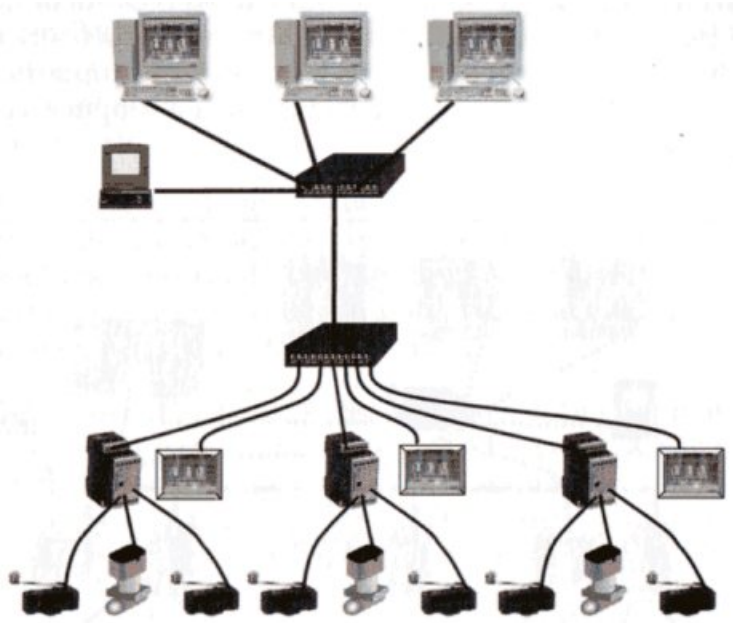


Figure 3.6: A distributes system using a set of local controllers, a local area network (LAN), and systems for remote operations (Caro 2004).

3.5 System Reliability

3.5.1 Introduction

Reliability, maintainability, and safety (RM&S) are important activities during product development. The main branches of reliability (Rausand & Høyland 2004), (Olsson & Rosen 2003):

1. hardware reliability,
   (a) the physical approach; technical items,
   (b) the actuarial approach; operating loads and strength,
2. software reliability,
3. human reliability.

Some system reliability definitions (Rausand & Høyland 2004), (Olsson & Rosen 2003):

1. Reliability; the ability of an item to perform the required function, under given environmental and operational conditions and for a stated period of time.
2. Quality; the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs.
3. Availability; the ability of an item (under combined aspects of its reliability, maintainability, and maintenance support) to perform its required function at a stated instant of time or over a stated period of time.
4. Maintainability; the ability of an item, under stated conditions of use, to be retained in, or restored to, a state in which it can perform its required functions, when maintenance is performed under stated conditions and using prescribed procedures and resources.
5. Safety; freedom from those conditions that can cause death, injury, occupational illness, or damage to or loss of equipment or property.
6. Security; dependability with respect to prevention of deliberate hostile actions.
7. Dependability; the collective term used to describe the availability performance and its influencing factors: reliability performance, maintainability performance, and maintenance support performance.

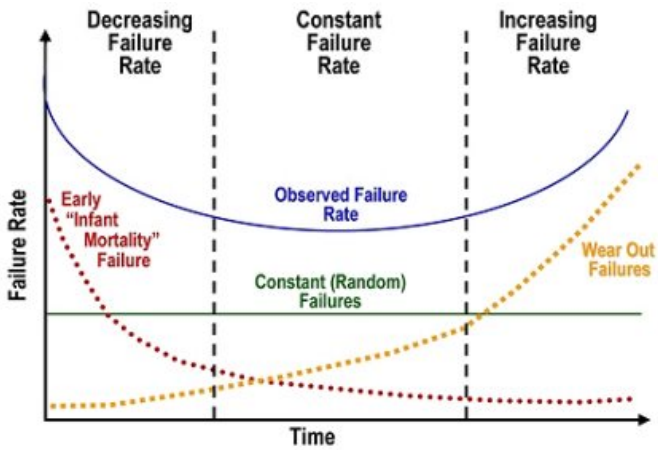


Figure 3.7: The fail rates over time for a given system or sub system. The shape is often known as the "bathtub" function. This fail rates are only valid for hardware failures.

3.5.2 Estimation

Applications for estimating system reliability (Rausand & Høyland 2004), (Olsson & Rosen 2003):

1. risk analysis,
2. environmental protection,
3. quality,
4. optimization of maintenance and operation,
5. engineering design,
6. verification of quality/reliability.

3.5.3 Computation

Computation of reliability (Olsson & Rosen 2003):

It is assumed that the possible errors are independent events, that they are not depending of each other. This assumption is correct as long as a faulty component does not influence the others and have a causal effect on their functionality (Olsson & Rosen 2003).

Using n components in the system, these components will operate or be faulty as:

$$n = n_o(t) + n_f(t)$$

where $n_o(t)$ is the number of operating components and where $n_f(t)$ is the number of faulty components, both numbers will be a function of time; n will be constant. The reliability function $R(t)$ is defined as (Olsson & Rosen 2003):

$$R(t) = \frac{n_o(t)}{n} = 1 - \frac{n_f(t)}{n}$$

A measure of the system is the MTTF (Mean Time To Failure) given as:

$$MTTF = \int_0^\infty R(t)\,dt = \frac{1}{\lambda}$$

where λ is the fault rate.

The fault rate for hardware devices often gives a shape known as the "bathtub" function, with several early faults in the beginning, a section with random faults (constant fault rate), and ends with wear-out faults. This shape is shown in Figure 3.7.

The reliability also depends on software and users, how will the curve in Figure 3.7 be for the software and the users?

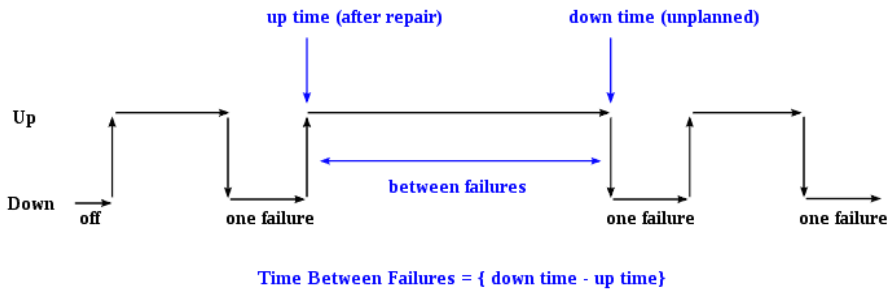


Figure 3.8: The mean time between failures for a device or a system (www.wikipedia.org 2006).

The availability of the system is measured as an average value in time intervals in which the system operates correctly, called the MTBF (Mean Time Between Failures). The average time intervals in which the system is not working is called the MTTR (Mean Time To Repair). The availability of a system A is defined as (Olsson & Rosen 2003):

$$A = \frac{MTBF}{MTBF + MTTR}$$

3.6 Redundancy

Redundancy is the duplication of critical components of a system with the intention of increasing the reliability, the stability and the operational reliability of the system, usually in the case of a backup and/or security (fail-safe) (Olsson & Rosen 2003). There can be 3 types of redundancy (Fortuna, Graziani, Rizzo & Xibilia 2007):

1. Physical redundancy; physically replicating the components to be used,
2. Analytic redundancy; the redundant source will be a mathematical model of the component,
3. Knowledge redundancy; The redundant source consists of heuristic information about the system.

Another way of dividing the forms of redundancy can be like:

1. hardware; 1) dual modular redundant (DMR) has duplicated elements which work in parallel to provide one form of redundancy, and 2) triple modular redundancy (TMR) is a fault tolerant form of N-modular redundancy, in which three systems perform a process and that result is processed by a voting system to produce a single output. If any one of the three systems fails, the other two systems can correct and mask the fault. If the voter fails then the complete system will fail. However, in a good TMR system the voter is much more reliable than the other TMR components.
2. information; 1) error detection and correction, 2) soft sensor, 3) disk arrays, and 4) time; i.e. time-outs and resending,
3. software; i.e. functions using the same type of algorithms based on different strategies,

Implementation of redundancy can be done in different ways, the goal is to have a backup system if the main system is failing. The type of implementation should be based on some sort of risk analysis, and very seldom the whole system needs to be duplicated. The most important components are power supply units (PSU), sensors, actuators and communication modules. An important property when deciding any form of redundancy is the Mean Time Between Failures (MTBF). MTBF is the mean time between failures of a system, see Figure 3.8. The calculation of the MTBF will be:

$$MTBF = \frac{\sum(downtime - uptime)}{number\ of\ failures}$$

Calculations of MTBF assume that a system is "renewed", i.e. fixed, after each failure, and then returned to service immediately after failure. The *downtime* and the *uptime* values are according to Figure 3.8 meaning that the E_{MTBF} will add up the time the system is not working dividing on the number of

times it is not working. The average time between failing and being returned to service is termed mean down time (MDT) or mean time to repair (MTTR).

Note that redundancy will 1) require extra functionality like automatic or manual switching between the duplicated components, 2) require support for maintaining and/or upgrading the duplicated components when the system is in use, 3) require that changes made for one component should automatically be transferred to the duplicated component, 4) also end up adding more components to the system that may fail, and 5) require good overview of the process when evaluating any need for redundancy.

3.6.1 Communication

Several nodes, computer with communication capabilities, are often connected to the same communication media. The communication media, called a bus, can be wire, optic, or wireless. A single failure of the wire or optic can stop all communication in the system. Several types of redundancy exists:

- ring redundancy; both ends of the communication media is connected to the master,
- sub ring redundancy; only parts of the network has redundancy,
- master redundancy; several masters monitoring the traffic of the network.

3.6.2 Why may redundancy be important?

Redundancy can be important for several reasons. 1) the vast majority of manufacturing companies depend on producing their products with as low a cost as possible at the right time and with the right quality. If machines, equipment and/or systems fail, it will be difficult to deliver on time. 2) redundancy can ensure security. Process security places demands on redundancy and sequential shutdown if something unforeseen should happen.

Redundancy can be implemented in several different ways. It is important to have some sort of back-up when something goes wrong. In all cases, the choice of redundant solutions should be the result of a risk analysis - must know what is critical. How long can the system be down? How long does it take to repair the system so that it is up and running again?

Redundancy is chosen to increase the uptime of a system. How a system gets the highest uptime and what should be duplicated varies from system to system. It is important to have a thorough understanding of the process to be managed.

3.7 Cluster

A computer cluster is a group of linked computers, working together closely so that in many respects they form a single computer. These computers are normally interconnected through a fast local area networks. Cluster of computers are usually deployed to improve performance and/or availability over that of a single computer. The solution with a cluster of computers are typically much more cost-effective than single computers of comparable speed or availability.

Redundancy is duplication of computers, while cluster is computers working together.

Chapter 4

Process Control System

4.1 Introduction

Different vendors are using different names on their process control systems. One of the name that is used a lot is Supervisory Control And Data Acquisition (SCADA). SCADA is an industrial control system for monitoring and controlling a process, or several processes. Other names are Industrial Control System (ICS) or Process Control System (PCS). In this document this type of systems are mainly named Process Control System (PCS) or SCADA systems, and such control systems are based only on software applications.

4.2 SCADA History

Automation system is divided into process automation and factory automation. Process automation consists of continuous processes with mainly analog signals and control loops. Factory automation consists of discrete production processes mainly with digital signals and sequential control. Traditionally there have been two choices when selecting a process control system, the DCS approach for process automation and the PLC/SCADA approach for factory approach.

The DCS approach was a complete software package with all modules from one supplier. The packages consist standard control objects that could be configured to the process system, a common configuration database, and other modules needed, all from the same supplier. Traditionally DCS has been expensive to support throughout the lifetime of the system with annual licensing and support costs. Most of the DCSs are engineered and supported by the DCS vendor leaving the end user dependent on this vendor.

The PLC/SCADA approaches uses several PLC devices and a SCADA system as the integration part. This may result in different configuration tools and configuration databases for the PLC and SCADA system. However several System Integrators with specific industry or application knowledge can make the integration and select a cost-effective solution. The weaknesses are that the system require different configuration databases for the PLC devices and the SCADA system as delivered from different vendors and that few PLCs have features required for demanding continuous process control applications. Another area of challenge is process optimization where many PLC is lacking the computational performance and system support when compared to DCS. These differences are shown in Figure 4.1.

Normally there is a need to choose between the usage of a DCS or a PLC/SCADA approach. However, today new systems are developed to take the best features from both approaches and integrate them into a single scalable solution. This allows for a single architecture to be used for a wide range of applications for both process and factory automation, and including new safety requirements. Programmable Automation Controllers (PAC) is such a device and has made the integration more flexible and more cost-effective. PAC is a control device with the functions of a PLC but has additional functions integrated such as 1) servo control (motion control), 2) more advanced processing for analog signals and 3) more advanced communications like gateways between networks to mention some. A PAC can often be looked at as a combination of the functions of both a PLC and a DCS.

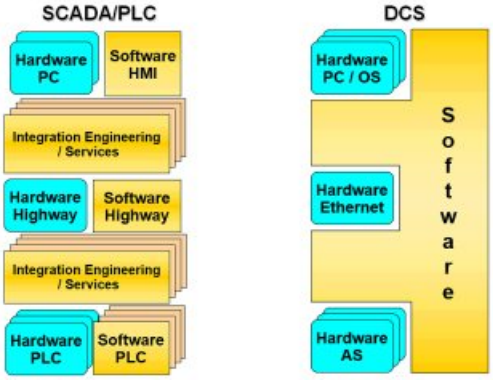


Figure 4.1: The configuration difference between a SCADA/PLC system and a DCS system. Several software applications, services and platforms for the SCADA/PLC solution, only one for the DCS solution. [copy from Siemens-PCS7: 2003]

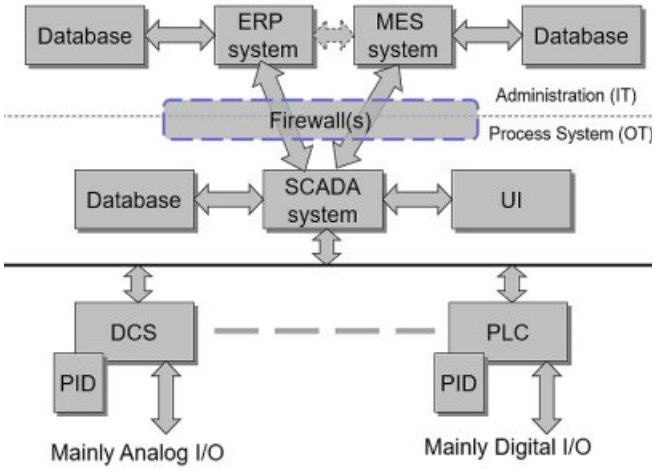


Figure 4.2: An overview of a set of devices and interconnections that can be used in an industrial IT system. The DCS, PLC and PAC are the RTUs. The UI devices are the user terminales for the system. Note that a network device is needed between the administration network and the process system.

4.3 SCADA Overview

Figure 4.2 shows how these devices, functions, and/or modules can be interconnected in an industrial IT system. An industrial IT system can however combine the devices and/or modules in any combination and this section will try to give a background for understanding the main functonality of these devices and the different ways of interconnect them. A SCADA system is a software system running on an industrial computer, normally with an alarm system, interfaces to database, external I/O devices and network, and an User Interface (UI). In Figure 4.2 will the UI devices be the user terminals while the UI software modules will be part of the SCADA system. The SCADA system will communicate with a set of external devices, denoted as RTU, being a DCS, a PLC, or a PAC depending on the I/O and control purposes. The RTU are physical devices, hardware devices, working as distributed modules in the process system. The PID module can be a software module of these RTUs. These RTUs will be standalone units, but the SCADA system can interact with these devices to monitor the process or changing the control parameters.

Figure 4.2, 4.3 and 4.4 show the connection to a management system. Different subsystems in the management system may use information from the SCADA system for planning and/or decisions. The layered interconnection between an ERP (Enterprise Resource Planning) system, a MES (Manufacture Execution System), and/or an IMS (Information Management System), and the SCADA system is shown in Figure 4.4. This figure shows the layer priorities of the systems and the information used in these systems. Figure 4.3 present the PCS/ICS as a pyramid structure with focus on information flow, but the same system as shown in Figure 4.2 and 4.4.

Figure 4.3: The PCS/ICS system presented as a pyramid structure where the process information is at the bottom and the business information on the top (from asitek.fi: Aug-19).

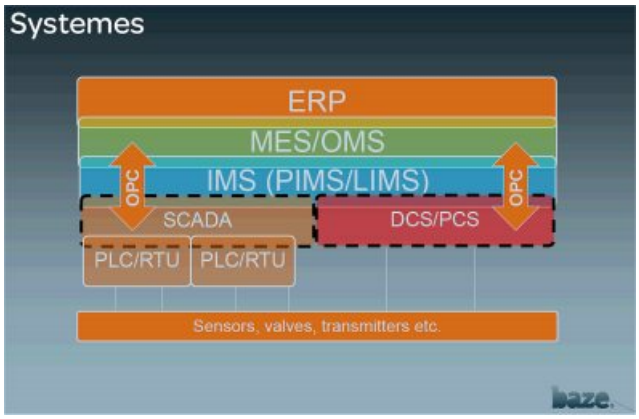


Figure 4.4: The layered connections between an ERP system, MES, IMS, and the SCADA system (from John Baaserud, Baze Technology).

The ERP, the MES, and the IMS all depends on the SCADA systems, the informations used and the decisions taken in the ERP, the MES, and the IMS depends on trustworthy informations from the SCADA system. The management system will be useless if the SCADA is not able to deliver trustworthy informations.

Note that Figure 4.4 shows a SCADA system connected to PLC only, while the DCS system in this case includes the SCADA system.

The difference between traditional IT systems and PCS is referred to as Information Technology (IT) for traditional IT systems and Operational Technology (OT) for PCS. In Figure 4.2 will the PCS be the OT while the administration system will be the IT. OT covers the hardware and software dedicated to monitor and control physical processes by the use of physical devices like sensors, valves, pumps, etc. (Isaksson et al. 2018) OT will include systems and modules like the PLC, DCS, and SCADA, and is all the equipment that the system will depend on to keep production going. What about the ERP system? The priorities for IT systems are 1) security, 2) data integrity and 3) downtime while the priorities for OT systems are 1) downtime, 2) data integrity and 3) security. It is important to understand these different priorities to achieve collaboration and integration between these teams.

An example of a plant using SCADA systems is shown in Figure 4.5. This is a gas power plant at Kårstø in Rogaland, in the western part of Norway, consisting of several SCADA systems monitoring and controlling the functions of the plant.

Many different SCADA systems exist in the market with different level of functionality and complexity. Some of theses systems are Wonderware, Citect, ClearCase, and iFix to mention a few.

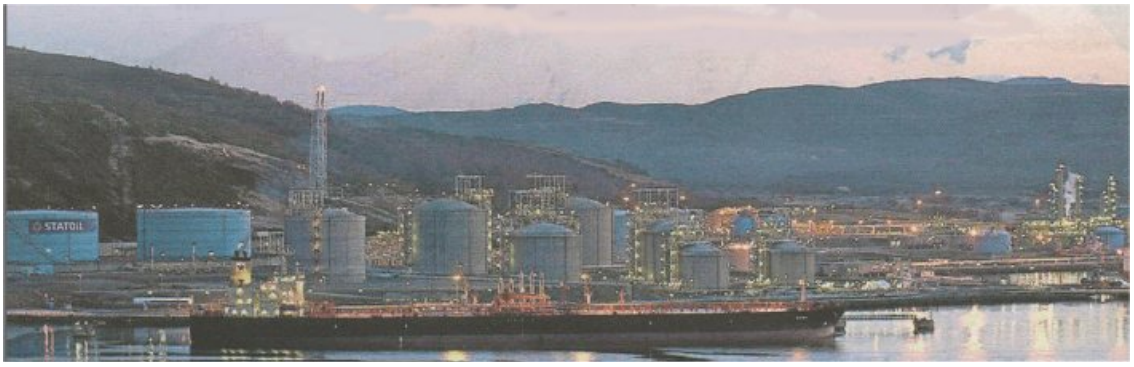


Figure 4.5: A gas power plant at Kårstø in Rogaland in the western part of Norway (Foto: Dag Magne Søyland/StatoilHydro).

4.4 SCADA modules

SCADA is an industrial control system monitoring and controlling a process, or separate control systems. A SCADA system is only a software application. A SCADA System usually consists of the following sub systems (www.wikipedia.org 2006):

1. a Human-Machine Interface or HMI (GUI) is the apparatus which presents process data to a human operator, and through this, the human operator controls the process,
2. a data engine, gathering (acquiring) all types of data from the process and sending any commands (control) to the process,
3. a database interface for storing the data, both for configuration, runtime, and historical information of the system. The database is normally a remote sub system, not an integrated part of the SCADA system,
4. an interface to Remote Terminal Units (RTUs) for connecting to sensors in the process, converting sensor signals to digital data and sending digital data to the supervisory system. RTUs are the collection of DCS, PLC and PAC.
5. communication infrastructure connecting the supervisory system to the Remote Terminal Units (and often to the administration network/system),
6. an integrated alarm system.

A SCADA system with these subsystems are shown in Figure 4.6.SCADA systems consist of several software modules, at least a monitoring module to get the information from the process, a control module to "write" information back to the process and a data engine. The data engine will be the "working horse" in such a system as this module will be responsible for the data flow in the SCADA system. Figure 4.7 shows the most common software modules (or sub modules) in a SCADA system. Note that the data engine is not part of the figure as the data engine is responsible for the data flow between all of these modules.

More details about some of the important SCADA subsystems (or sub modules) will follow in the next sections.

4.4.1 User Interface (UI)

A User Interface is the device or module which presents process data to a human operator, and through which the human operator controls the process. The User Interface (UI) is also known as the:

1. Graphical User Interface (GUI); today almost all user interfaces are graphical,
2. Man-Machine Interface (MMI), the same as a UI,
3. Human-Machine Interface (HMI), the same as a UI.

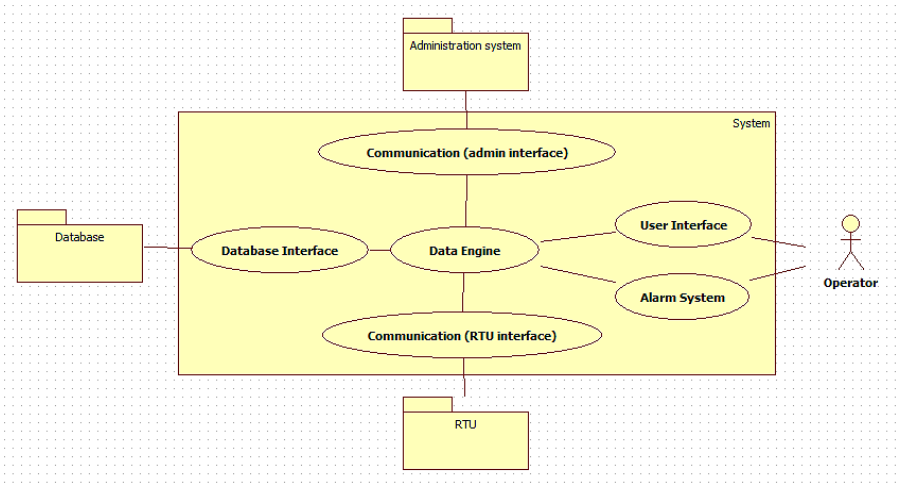


Figure 4.6: An overview of the main modules and subsystems in a SCADA system. The SCADA system communicates with 1) the RTU for process influence, 2) database for configuration, runtime and historical data, 3) administration systems and 4) any operators.

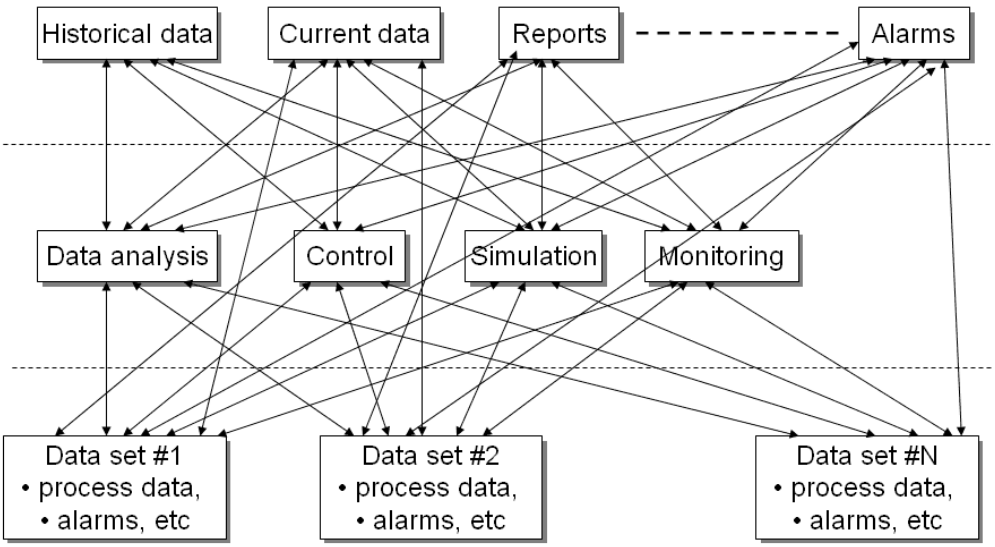


Figure 4.7: An overview of some of the software modules that can be part of a SCADA system. The Figure is shown as a three layered application structure with user interface layer on top, business lager in the middle, and data layer at bottom.

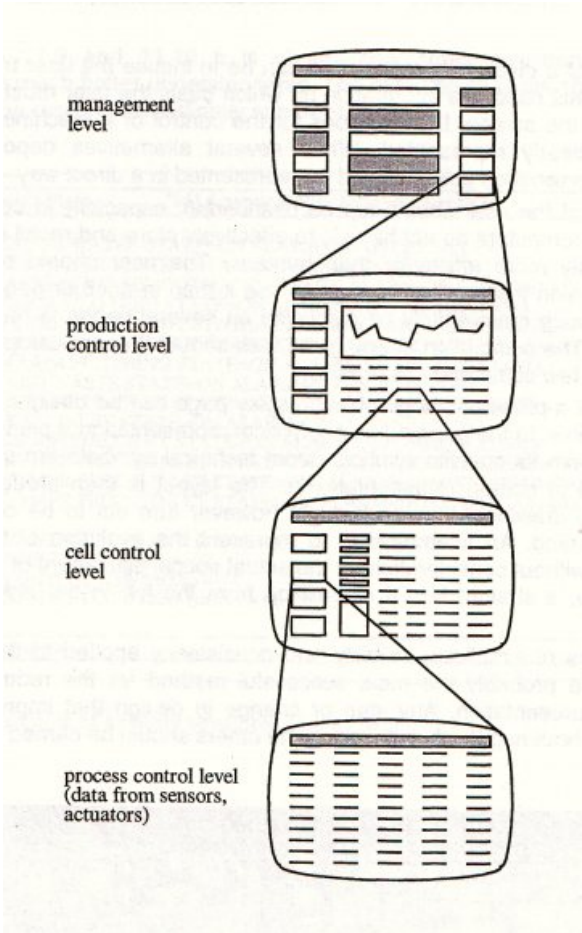


Figure 4.8: Several layers of information in a process system, from overview information about the plant down to detailed information about a sensor device (Olsson & Rosen 2003).

A UI can be simple devices like buttons and lamps up to complex systems on several computer screens or an overhead screen. The presentation of the process information is very important to let the operator focus on the important information only. The separation in information levels is a common solution for control system. The system has an overview presentation and the operator can select a specific part to get more detailed information. Depending on the size of the plant and type of information there can be several layers of information. An example of such a system is shown in Figure 4.8. The operator can also go back to the previous information, normally either the previous information or the first overview information. Figure 4.9 shows two computer screens with a graphical UI with process information and control options for a plant. Figure 4.10 shows an overview GUI of a process or plant, and a set of GUI with details from several parts of the same process or plant.

Figure 4.11 shows a large overview of the process or plant in the background, and each operator will have more details from several parts of the process on the screens in front of each operator.

The amount of information on the screen has to be adopted to the usage of the system and it is important not to overload the GUI with information. Figure 4.12 shows two different modes of presentation of the same information. Which presentation will be best for the operator?

The presentation to the left present all data, both overview and detailed data in the same view. The operator has to sort out a lot of detailed information to get an overview of the data. The presentation to the right has a focus only on the overview data. The operator can zoom in on a tag to get more detailed data about that specific tag. A tag is describing a single point of information, often an input and/or output signal.

The design of the UI is important for the usage, understanding and focus of the system. Guidelines for designing UI are described in several media including books as (Bullemer, Reising, Burns, Hajdukiewicz & Andrzejewski 2008), (Hollifield, Oliver, Nimmo & Habibi 2008) and (Olsson & Piani 1998).

Figure 4.9: A computer screen based UI for a plant (www.analogdevices.com: SEP-08).

Figure 4.10: The GUI of a SCADA system showing an overview of a total process (or plant) (www.abb.com: dec-09).

Figure 4.11: Another GUI from a SCADA system showing an overview of the total process (or plant) in the background. Each operator will have more detailed information about the process on the screen in front (Photo from Automatisering 05/2012).

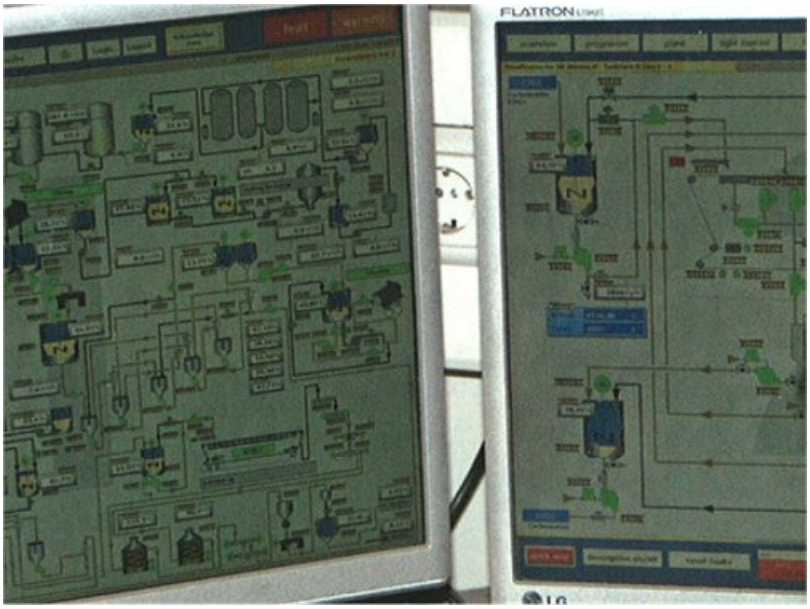


Figure 4.12: The same type of information presented in two different systems. Which system will you prefer to use? The system to the left present all details in the same view, while the system to the right using an overview view with an option to get any details by clicking on a tag.

Part of the UI can also be a decision support system. A decision support system requires updated and structured information in a readable form, and as much information as possible. The type of information may be number of items produced at fixed time intervals, tracability, number of items rejected, the reason for the rejection, running time and statistical process control to mention some. The amount of information has to be enough to be able to make any decision. The decision support system will also contain a set of reports generated automatically at fixed time intervals. A good decision support system is often a combination of an IT system and manual systems like blackboards for status and follow-up.

4.4.2 Tag number conventions

A set of field devices are selected for the process, based on the process design requirements. Tag names and numbers are assigned to these field devices for identification, and with an unique identifier to each device. The tag is defined based on some sort of documentation of the process, like a piping and instrumentation diagram (P&ID) and/or loop diagram. The International Society of Automation (ISA) has the standard "Instrumentation Symbols and Identification" (ISA-S5.1) (ISA 1992) defining standard tag number convention for valves, transmitters, other field devices, and calculation and control functions. In addition, this standard also defines expanded tag number conventions for larger installations. An overview of tag number convention is shown in Figure 4.13.

An example of a tag name or tag number can be TIC103 where T is the first letter indicating temperature, IC is the succeeding letters indicating a controller with some sort of indication, and 103 is the loop number. All field devices used together to perform a specific function are normally assigned the same loop number, and this loop number consists of three digits. This numbering will limit the number of field devices so the expanded tag number convention allows a number to be inserted in front of the function. This number is usually an indication of the process area number. The combination of the process area number, the function letters, and the loop number is unique within a plant. One or more characters may also be added after the loop number if several equal field devices are used for uniquely identify each measurement.

4.4.3 Data Engine

A data engine is the module in the SCADA system responsible for data/information exchange within the system. The SCADA system must collect information from the RTUs, handle the information, and update modules and external systems and RTUs. Figure 4.14 shows the main functionality of the data engine. The data engine manages the reading of the values from the process using the RTU

	FIRST-LETTER (4)		SUCCEEDING-LETTERS (3)		
	MEASURED OR INITIATING VARIABLE	MODIFIER	READOUT OR PASSIVE FUNCTION	OUTPUT FUNCTION	MODIFIER
A	Analysis (5,19)		Alarm		
B	Burner, Combustion		User's Choice (1)	User's Choice (1)	User's Choice (1)
C	User's Choice (1)			Control (13)	
D	User's Choice (1)	Differential (4)			
E	Voltage		Sensor (Primary Element)		
F	Flow Rate	Ratio (Fraction) (4)			
G	User's Choice (1)		Glass, Viewing Device (9)		
H	Hand				High (7, 15, 16)
I	Current (Electrical)		Indicate (10)		
J	Power	Scan (7)			
K	Time, Time Schedule	Time Rate of Change (4, 21)		Control Station (22)	
L	Level		Light (11)		Low (7, 15, 16)
M	User's Choice (1)	Momentary (4)			Middle, Intermediate (7,15)
N	User's Choice (1)		User's Choice (1)	User's Choice (1)	User's Choice (1)
O	User's Choice (1)		Orifice, Restriction		
P	Pressure, Vacuum		Point (Test) Connection		
Q	Quantity	Integrate, Totalize (4)			
R	Radiation		Record (17)		
S	Speed, Frequency	Safety (8)		Switch (13)	
T	Temperature			Transmit (18)	
U	Multivariable (6)		Multifunction (12)	Multifunction (12)	Multifunction (12)
V	Vibration, Mechanical Analysis (19)			Valve, Damper, Louver (13)	
W	Weight, Force		Well		
X	Unclassified (2)	X Axis	Unclassified (2)	Unclassified (2)	Unclassified (2)
Y	Event, State or Presence (20)	Y Axis		Relay, Compute, Convert (13, 14, 18)	
Z	Position, Dimension	Z Axis		Driver, Actuator, Unclassified Final Control Element	

Figure 4.13: An overview of the tag number convetions in the ISA-S5.1 standard (ISA 1992).

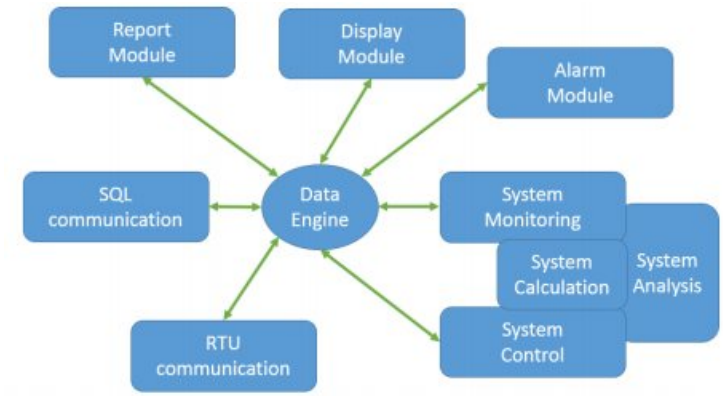


Figure 4.14: An overview of the functionality for a data engine. The responsible of the data engine is to handle the data flow between all the other modules of a SCADA system.

communication, and handling monitoring, calculation, analysis and any estimation of control derived values based on these RTU values, store any values in the database, display, alarm checking and send any control values back to the RTU. The data engine also handles report requests and any requests from the administration network, however note the figure has no connections to an administration network. How to add connection to the administration network? The data engine module will normally be an imaginary module being part of all the other modules where each module has a limited responsibility in handling the data flow in the system.

Pseudo coding or a data flow diagram can be a good way of documenting the functions of a data engine. Pseudo code is defined as an informal high-level description of the operating principle of a computer program or other algorithm. The pseudo coding can be based on a virtual high-level programming language, no need of defining of variables, but with "human" understandable structures and methods name. No need to define the methods, pseudo coding is used mainly for documenting the main software structure, and describing the data flow. Using pseudo coding based on the C# programming language, the coding of the data engine can be like:

```
while (TRUE)                                        // Endless loop
{
    RtuDevice = 1;                                  // start with the first communication device
    while (RtuDevice < MaxRtuDevice)
    {
       // Get the values from a specific RTU
       RtuVal[] = getValuesFromRtu(RtuDevice);
       // Do the handling of the RTU values: Monitor, calc, analysis and control
       RtuCtrl[] = handleRtuValues(RtuVal[]);
       writeValuesToRtu(RtuDevice, RtuCtrl[]);
       displayRtuValues(RtuDevice, RtuVal[], RtuCtrl[]);
       // Alarm checking of the values
       alarmChkRtuValues(RtuDevice, RtuVal[], RtuCtrl[]);
       // saving the current process values in the process database
       SaveDatabaseValues(RtuDevice, RtuVal[], RtuCtrl[]);
       // Check for any report request from the user
       if (checkReportRequest() == FALSE)
       {
          // Release some CPU power if no user request
          waitSystem();
       }
       // Next RTU device
       RtuDevice++;
    }
}
```

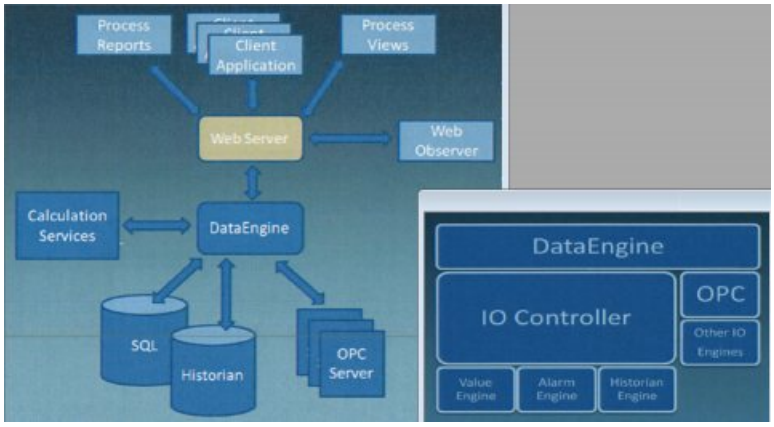


Figure 4.15: An overview structure of a data engine with web support. On the left side an overview of the moduls where the data engine collect information from I/O devices, use the Calculation Services for any estimation/calculation/prediction based on the information, stores the information, and update the web servers. Users and operators are using web clients for access to the information. The right side shows how the data engine is using more specific data engines for handle information in specific modules. These specific data engines are distributes data engines working together with the main data engine (from John Baaserud, Baze Technology).

This pseudo coding describe the data flow of the data engine. *Is this the best way of solving the data flow?* Why doing the alarm checking after writing the values back to the RTUs and displaying the values? May be the alarm checking should be done before writing and displaying the values? In that case the alarm status will be updated before the values are displayed on the screen. That may be a better solution. Why reading one and one RTU, why not read all the RTU's first before handling the values, alarm checking and saving the values? There will not be a perfect way of managing the data flow, but by using pseudo coding, or another way of documenting the data flow, the specific structure can be documented, and also evaluated for any "better" solutions.

All SCADA systems must have some sort of a data engine to handle the information within the system, the execution of estimations, calculations and predictions, and transferring the information to, and between, the necessary modules within the system. The data engine for a specific system is shown in Figure 4.15.

4.4.4 Database

A database is a structured collection of data that stored in a computer system. The structure is achieved by organizing the data according to a database model(www.wikipedia.org 2006). The model used most today is the relational model, but other models exist such as the hierarchical model or the network model. In a process system is the database used for:

1. configuration or setup data; how is the process system structured with references to the input and output of values,

2. runtime data; the current values of the process system. The current value should be the measured or derived value, the time stamp and the status,

3. historical data; a history of the "current" values of the process system.

4.4.5 Alarm system

A process system needs an alarm system for presenting the alarms and any alarm actions from the operators. The alarm system should be an integrated part of the monitoring and control systems to give an overview of all the alarms. A SCADA system should only contain one alarm module, integrating all the alarms from the SCADA system and any submodules and devices. An alarm system is stressing the operator and minimizing this stress by only one alarm system (handling only one user interface) is

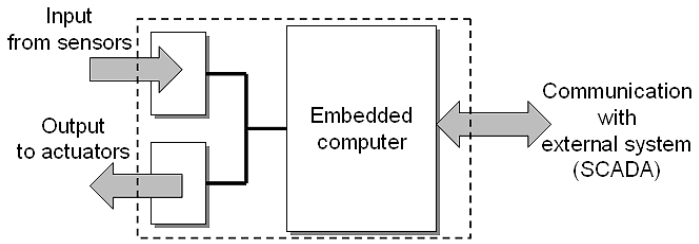


Figure 4.16: The modules of the RTU, with I/O for reading and writing values to process equipment like sensors and actuators and communication to some sort of SCADA system.

a good solution. An alarm require attention from the operator, the operator should acknowledge any alarms.

4.5 RTU devices

A SCADA system is normally using a set of sub devices for monitoring and controlling a plant. The size of the plant will decide the distribution of these devices, but normally should the SCADA system be independent of the "physical" monitoring and control of the plant. Remote Terminal Units (RTU) are used as distributed computer systems in a larger control system where each RTU will control and/or monitor only a part of the plant. An RTU will be used for each open or closed control loops or on/off control for better maintaining the control strategies of the plant. The RTU will be an industrial computer system consisting of (See Figure 4.16):

- inputs; for analog and/or digital inputs,
- outputs; for analog and/or digital outputs,
- communication; for communication with the SCADA system (serial lines or network, wired or wireless),
- processor and memory,
- software dedicated to the functions of the RTU.

The RTU can use a real-time operating system or an operating system customized to the operation of the RTU. The RTU is a standalone physical unit working without any interactions from any remote systems, except for comunication of information.

This information can be data from the RTU or commands to the RTU so the remote system (SCADA) can monitor the device and also change any control parameters of the RTU. Available RTUs are the DCS, the PLC, and the PAC. A DCS can be a computer running an operating system like Windows or Linux, and applications developed using tools like LabVIEW or programming languages like C#. A DCS can also be a network of computers, include PLCs. A PAC is often a combination of a DCS and PLC.

4.5.1 PLC

A Programmable Logic Controller (PLC) is a RTU used mainly for digital I/O for the control system. These devices are primarily used for sequence control based on on/off inputs and outputs, but today these devices may also include analog I/O (Mackay, Wright, Park & Reynders 2004). The PLC is often a single computer. The PLC will often depend on a cycle time, at a specific time a copy of the input states will be copied into the system memory (or I/O registers), the PLC program will run in working memory, and depending on the input states and the program, the output states will be generated based on the information in system memory only. The input states will be stored as an image when starting a new cycle time, keeping the input signal constant during the cycle time, and generating the output state. At the end of the cycle time the output state image will be copied to the output pins (registers). This way the input states will be constant when the program is running, the output will only be updated at the end of each the cycle time, and there will always be a delay from the input states to the output states in a PLC system. This is shown in Figure 4.17.

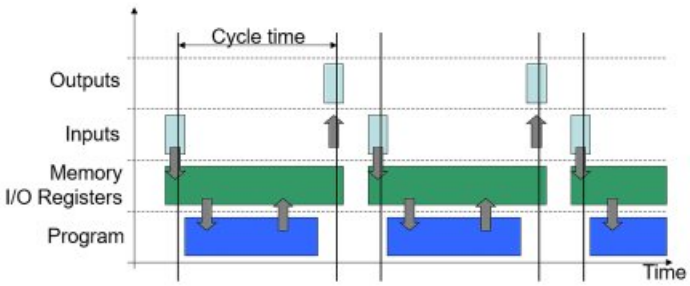


Figure 4.17: The operation of a PLC program. An image of the input bits will be copied to the memory, the program will perform the operations depending on the input image in memory, generating an output image. At the end of the cycle time will the output pins be updated.

A PLC application can be developed using a different types of special programming languages and tools. The application can be developed on the PLC or in a development environment on a PC and uploaded to the PLC. Some of the programming languages for a PLC are, all part of the IEC61131-3 standard:

- Instruction List (IL); low level textual language (assembler type),
- Structure Text (ST); High level procedural programming language (Pascal syntax),
- Ladder Diagram (LD); evolution of electrical wiring diagrams, the main programming language for PLC based on relay logic,
- Functional Block Diagram (FBD); A graphical method based on grouping function blocks,
- Sequential Function Chart (SFC); A graphical method based on states and transitions, describing the behavior of the processing elements.

Despite the popularity of PLCs, there are some limitation of these devices. These are limited memory capacity, the low level programming languages, no support for object oriented programming, limitation to application architectures, and updates of the application software.

4.5.2 DCS

A Distributed Control System (DCS) is a RTU used mainly for analog I/O in the control system. The DCS often has a modular, distributed and open architecture (Mackay et al. 2004). The DCS is dedicated for a specific task in the control system, often analog monitoring and/or control loop. These devices may also have a small display for local UI (Mackay et al. 2004). The DCS is used for control purposes in a distributed system, an industrial process will consists of a set of DCS controlling only a part of the process. The DCS itself can be a single computer or a set of computers in network depending on the complexity and/or requirements of redundancy for the control system. To complicate the structure a PLC can also be part of the DCS network.

An example of a plant controlled by a number of DCS is shown in Figure 4.18. The DCS's are connected in a network. Each DCS can consist of a single computer or several computers interconnected in a private network or the common network. The network of computers for a DCS can be any combination of computers for measurement and control, both DCS and PLC.

Some vendors also use DCS as a description of a SCADA system having three main elements:

1. the operator console (HMI),
2. the distributed controllers with integral input/output (I/O) modules,
3. a proprietary communications protocol.

This mean that a DCS can be anything from a single controller up to a complete SCADA system, depending on the vendor. A DCS application can be developed using standard programming tools like MatLab, LabView, C, C++, C# and/or Java to mention some. The operating system and DCS tools will decide the type of Integrated Development Environment (IDE) and programming language.

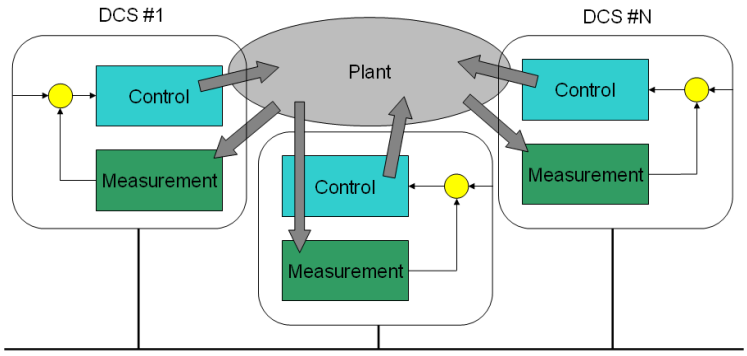


Figure 4.18: A set of DCS are used for monitoring and controlling a plant. Each DCS can consist of a single computer or a set of computers in network. The DCS or the network of DCSs are connected to the same netwrk as the SCADA system.

4.5.3 PAC

A Programmable Automation Controller (PAC) is a compact controller that combines the features and capabilities of a PC-based control system with that of a typical PLC. A PAC provides the reliability of a PLC and the task flexibility and computing power of a PC.

4.5.4 Use PLC or DCS?

PLC and DCS technologies were invented to serve different industrial processes, PLC mainly for factory automation and DCS mainly for process industry. During the last 15 years these technologies have merged and now the selection can be confusing. The combination of a SCADA system and PLC can deliver the same functionality as a DCS. A new system should be easy to maintain and extend, improve the reliability and minimizing the downtime. However a PLC may not have enough memory to meet the requirements while a traditional DCS may be too large, complicated and costly. Still the DCS technology offers several advantages for industrial operations. What about a distributed mixed system based on both PLC and DCS? A six step selection process has been proposed to help determine which technology is best suited for the application, see (Yehia 2016). A flow chart, shown in Figure 4.19, indicates these steps.

Summary of the six step selection process and the flow chart(Yehia 2016):

1. If the process require an implementation of advanced process control (APC/MPC), select a DCS.
2. If a high number of regulatory control loops are needed, select a DCS.
3. If an operator control room and/or high speed discrete control is required, select a PLC (with User Interface).
4. If system modification is needed, select a DCS.
5. If both advanced control or many control loops, and high speed discrete control is required, select both DCS and PLC.

4.6 RTU Subsystems

4.6.1 PID Controller

A proportional–integral–derivative controller (PID controller) is a generic control loop feedback mechanism (controller) widely used in industrial control systems. A PID controller attempts to correct the error between a measured process variable and a desired set point by calculating and then outputting a corrective action that can adjust the process accordingly(www.wikipedia.org 2006).

The PID controller calculation (algorithm) involves three separate parameters; the Proportional, the Integral and Derivative values. The Proportional value determines the reaction to the current error, the Integral value determines the reaction based on the sum of recent errors, and the Derivative value

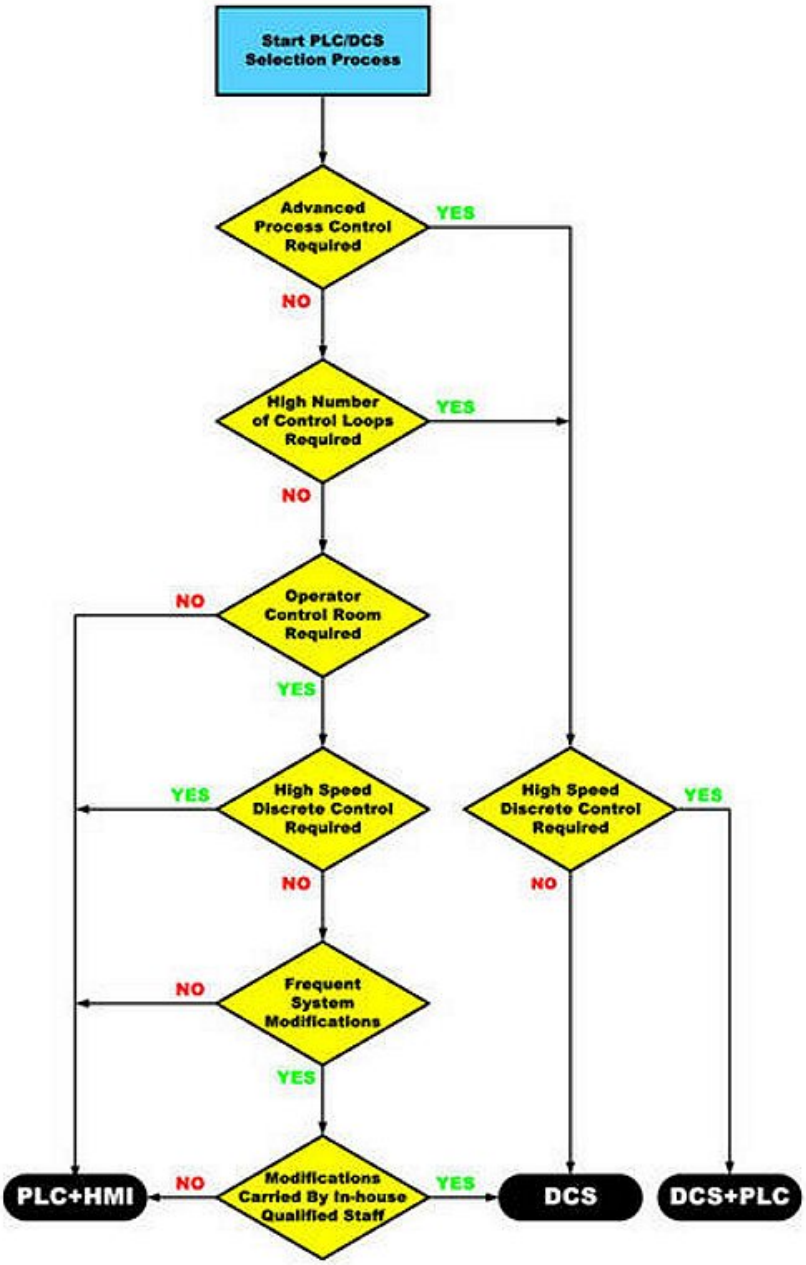


Figure 4.19: The six step PLC or DCS selection process to help determine the best technology (Yehia 2016).

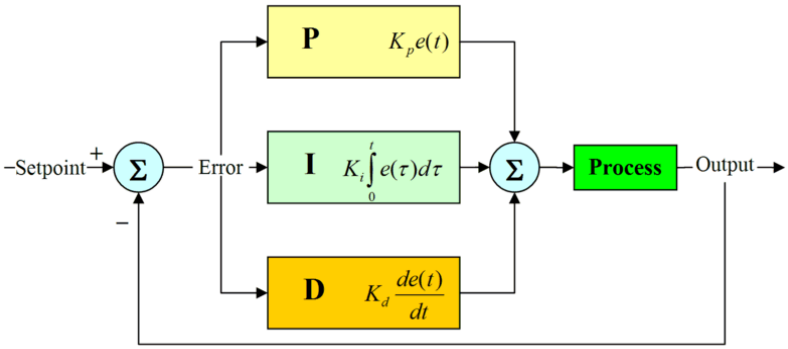


Figure 4.20: An overview of the PID controller (www.wikipedia.org 2006).

determines the reaction based on the rate at which the error has been changing. The weighted sum of these three actions is used to adjust the process as shown in Figure 4.20 where K_p is the proportional factor, K_i is the Integral factor, and K_d is the derivative factor.

The PID function or controller is controlling the process using some type of analog actuator(s). The PID function is often a part of the functionality of the RTU in the SCADA system, only a software module. Some applications may require using only one or two modes to provide the appropriate system control giving a PI, PD, or just a P controller. The PID function is often a software module of the DCS, the PLC, and the PAC devices.

4.6.2 CNC Machines

Computer Numerical Control (CNC) machines are using programmed operations for machine tools. These machine tools, powered mechanical devices, are operated only by the software program downloaded into the device.

4.6.3 Robots

Robots are flexible tools in automation systems handling a fixed sequence of operations. The robots are programmed using specific programming tools, as a sequence of movements for each part of the robot. We often think of robots as a mechanical device but there exist also software robots. Robotic Process Automation (RPA) is a software robot that performs the tasks of one human through interaction with the user interface to one or more IT systems or computer programs. RPA can be useful in digitalization.

4.6.4 Instrumentation

Instrumentation consists of both sensors and actuators; sensors for monitoring the process and actuators for controlling the process. Sensors and actuators have either an analog or a digital interface for connection to the RTU. An analog interface will require some type of data acquisition (DAQ) devices for connection between the sensor devices, actuator devices and the computer system. The DAQ device will be used for analog to digital conversion of the signals from any analog sensor devices, and for digital to analog conversion for any analog actuator devices. Sensors and actuators with a digital interface, a communication protocol, will require some type of gateway for communicating with the RTU or even the SCADA system. Internet of things (IoT) devices are instruemntatiuon with a digital interface, using the internet protocol as the digital interface. Devices with a digital interface will have an internal A/D converter and CPU to convert the sensor signals to a digital value, and using the protocol for communication. The communication media can be wire, fibre and/or wireless media.

4.7 MDIS network

The MCS-DCS Interface Standardization (MDIS) network has been formed with involvement from many important actors within the subsea controls area, topsides specialists and oil companies. A standard interface will be developed, including a standard communication protocol, for communication between the

Figure 4.21: The administration system will be a superior system to the PCS and SCADA module.

Figure 4.22: The main modules in the administration system that will depend on information from the PCS.

Master Control System (MCS) and Distributed Control System (DCS) on topside systems. Standardization of the interface simplifies implementation of data communication links and increasing the data quality.

4.8 Subsea systems

Some systems have some of the process components on the seabed. These components are located on the seabed to optimize the production, especially on small distributed production "points". The components are mainly modules as separators, pumps, power distribution, and gas compression. Such subsea systems are used among others in the oil production in the north sea.

The separators is used to separate water and mud from the oil production, inject the water and sand back into the reservoir and just pump the oil and gas for further processing. This module will increase the production and reduce the energy consumption.

The pumps are used for pumping oil and gas from the separators to the production section. Gas compression are used to compromise the gas on seabed before pumping it onshore (or to a platform).

Power distribution is used for distributing power to all other modules, and is also responsible for communication. Communication will be both between the modules on the seabed, and between the seabed and process control system onshore or on a platform.

The challenges with a subsea system are among others:

- power distribution and communication,
- all rotating systems must be able to work without any maintenance for a long time,
- optimize the separation, normally the separation is solve in several steps with only the first step on the seabed.

4.9 Superior SCADA systems

The administration system needs information from the PCS to be able to better handle the inputs and outputs from the PCS systems. These inputs and outputs can any materiel and personell needed for production or manufacturing, and any outcome of the production or manufacturing. Figure 4.21 shows how the administration system is connected, it is connected only to the PCS, not the process. The administration system will depend on reliable information from the PCS for making any decision.

The administration system will require different type of information from the SCADA system, mainly about the status of the production or manufacturing. The administration system consists of several sub systems as shown in Figure 4.22.

These systems are the Enterprise Resource Planning (ERP), the Manufaturing Execution System (MES) and the Information Managaement System (IMS). The Figure indicate the type of information needed given that the IMS will require more technical information from the PCS than the MES, while the MES will require more technical information from the PCS than the ERP. Most administration systems have the ERP module, there is often a requirement for the MES module, while few systems have the IMS module included.

There must also be some sort of network device to limit or restrict the communication between the administration system and the PCS. Normally the administration system can read information from the PCS, but with limitation in writing information back to the PCS.

4.9.1 ERP

Enterprise resource planning (ERP) is an enterprise-wide information system designed to coordinate all the resources, information, and activities needed to complete business processes such as order fulfillment or billing. The main purposes is to plan the resources for the production, and handle orders and invoices.

An ERP system supports most of the business system that maintains in a single database the data needed for a variety of business functions such as Manufacturing, Supply Chain Management, Financial, Projects, Human Resources and Customer Relationship Management. The following are steps of a data migration strategy that can help with the success of an ERP implementation:

1. Identifying the data to be migrated,
2. Determining the timing of data migration,
3. Generating the data templates,
4. Freezing the tools for data migration,
5. Deciding on migration related setups,
6. Deciding on data archiving.

Names of ERP system are SAP, IFS, Movex/Lawson, Navision, Microsoft Dynamics, Visma to mention some.

4.9.2 MES

Manufacturing execution systems (MES) serve as the intermediary between a business system such as ERP and a manufacturer's plant floor control equipment such as PCS. MES is a database system for an efficient production and optimizing the production from start to ending product. MES is helping to manage production scheduling and sequencing, creating an audit trail for track and trace, and delivering work instructions to shop floor workers. Functions in MES can be scheduling and resource allocations, startup and execution, prescription handling, tracability, data collections, quality control, reports and analysis. MES normally contains more details about the production than the ERP, can be used for updating the ERP system and used for optimization of the production. MES can also be named as Operations Management Software (OMS). The standards ISA 95 and IEC 62264 are defining five levels for production systems where ERP is level 4 and MES is level 3, and the actual production process is level 0. These levels are shown in Figure 4.23.

The focus with a MES system is optimum usage and traceability of the production, to be able to figure out :

1. where a product is manufactured,
2. when a product is manufactured,
3. any sub devices used to manufactured this product,
4. any claims, warnings, or errors of this product,
5. the type of raw materials that have been used in the product or production.

Note that a MES system is not designed to be a PCS as the information is changing about ten times faster on level 2 then on level 3 in the production system. The ERP system knows "why" while the MES system knows "how" when dealing with the production system.

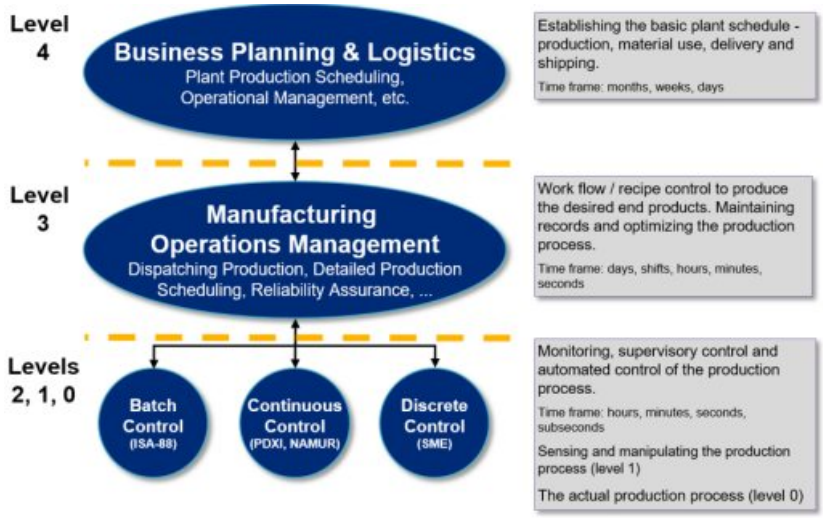


Figure 4.23: The levels in production systems where the actual production process i level 0, according to the standards ISA 95 and IEC 62264.

4.9.3 IMS

Information management system (IMS) is an information that makes low level information available on all levels in the organization. This system is often divided into the subsystems:

- Laboratory Information Management System (LIMS); Information system for lab data; samples, analysis, results, instruments managing etc.,
- Process Information Management System (PIMS); Information system with focus on process data and data acquisition to a real-time database, in the process industry is the PIMS and the IMS the same system.

4.9.4 MOM

Manufacturing operations management (MOM) is a collection of systems for managing end-to-end manufacturing processes for optimizing efficiency. The MES and MOM terms are often used interchangeably and both are refering to the ISA-95 Level 3 architecture and functionality. MOM is including some extensions to the level 3 modell including the usage of IoT and IIOT devices, and advanced configuration options. Still many companies have different understanding of the terms so it is important to define any differences, if any. There are many types of MOM software, including for production management, performance analysis, quality and compliance, and human machine interface (HMI).

4.10 Safety

A SCADA system must also contain control logic for safety and shutdown functions. Such a system can also be named a Safety and Automation System (SAS). The safety systems can be part of the SCADA system, but often there is a requirement for an external safety system for redundancy.

There exists a lot of different standards and regulations regarding the safety matter for equipment, we the humans, and the environment. Some international, other local to different countries, or local extensions or limitations to international standards and regulations. Some of these standards are the IEC 61508 and the IEC 61511. These standards are normative for design, installation, operation and maintenance of system both onshore and offshore.

4.10.1 Safety system

Safety Integrity Level (SIL) is requirements for the processing chain; reading, evaluating, and responding. Regarding a process system or a SCADA system will the SIL be looked upon as the sensor devices, the processing units (RTUs), and the actuators. The SIL consists of a set of numbers with 1 as the lowest

level. The weakest part in a chain will decide the SIL for the whole chain. One way to have a higher SIL is to use fail safe controllers. These controllers will evaluate safety relevant field signals and switch to or stay in a safe condition in the event of faults (PROFIsafe 2009). In a fail safe controller will safety oriented operations be processed in two different paths (algorithms) and the results are compared at the end of the algorithms. If any deviations, a fault has occurred in one of the paths, and the controller will switch to a safe condition. These controllers must have extensive self-diagnostic facilities (PROFIsafe 2009).

The higher the degree of automation, the more the control needs to be monitored for safety, but this is only possible if the system is fail-safe (PROFIsafe 2009). Safety will be regarding hardware, software, and communication modules and devices in the system.

The reason for SIL is to avoid undetected errors on instruments that can damage people and/or the environment. The consequence of these errors can be reduced by mapping the risk of damage and defining the security level, 1 is the lowest level and 4 is the highest level.

IEC61508 Safety Standard

IEC61508 is an international standard focusing on safety-related systems that incorporate electrical, electronic and/or programmable electronic (E/E/PE) instruments and devices. This standard is mainly used in the automation and process control industry, but is more and more accepted for applications in other industries.

The IEC 61508 standard is divided into 7 parts:

1. General requirements,
2. Requirements for (E/E/PE) safety-related systems,
3. Software requirements,
4. Definitions and abbreviations,
5. Examples of methods for the determination of safety integrity levels,
6. Guidelines on the application of IEC 61508-2 and IEC 61508-3,
7. Overview of measures and techniques.

4.10.2 Shutdown system

The function of a shutdown system is to protect environment, plant and humans in case any state of the process goes beyond predefined boundaries. A plant normally have several levels of protection like:

- Process Control System (PCS) for daily operation of the plant, using the alarm system of the SCADA system,
- Process Shutdown System for controlled process shutdown (PSD),
- Emergency Shutdown System in case of an emergency situation (ESD),
- Fire & Gas System (FGS) to detect fire and initiate automatic shutdown,
- Mechanical devices like Pressure Safety Valves (PSV) to avoid overpressure when previous systems fail.

4.11 Life Time

The expected life time of a process control system may vary depending on the type of devices or modules in use. There will also be a technical life time and an economical life time. The typical life time of the modules in a PCS is:

- HMI (Human-Machine interface) or UI (User Interface); the typical life time is 8 years,
- Network; the typical life time is 10 years,
- RTU and RTU applications; the typical life time is 20 years,

- I/O solutions; the typical life time is 30 years,
- Field devices and equipment; the typical life time is 40 years,
- Safety systems; the typical life time is 40 years.

The life time can vary depending on the maintaining and service contracts, however the expected life time should be taken into consideration when using IT devices and modules in a PCS.

4.12 Documentation

A SCADA system can be documented in different ways. Some of the diagrams are:

1. P&ID (Process and Instrumentation Diagram); a family of functional one-line diagrams showing hull, mechanical and electrical systems like piping, instrumentation and cable block diagrams,
2. P&ID (Piping and Instrumentation Diagram/Drawing); a schematically diagram showing piping, equipment and instrumentation connections within process units (www.wikipedia.org 2010),
3. SCD (System Control Diagrams); mainly for SAS, integration of the control logic in order to more easy be able to check the logic "connection" to other areas in the system.

The documentations are using different types of levels and symbols to show the contents of the control and monitoring systems.

4.13 Planning

The steps of production in an automation system are:

1. order; the registration of the order, part of the ERP,
2. coarse planning, part of the ERP,
3. detailed planning,
4. production,
5. storage, part of the ERP,
6. logistics, part of the ERP,
7. economics. part of the ERP.

One of the important step in this planning will be to make a more detailed planning regarding the time intervals and resources like equipment, materials, and peoples. Different tools are used for the detailed planning as paper, black- or white board or Microsoft Excel. The disadvantage of such tools is no or little flexibility of changes. Any changes will have consequences, as:

- how will the change influence of the production of the other orders?
- how will the change influence of the staff, any usage of overtime?
- how will the change influence of the bottlenecks in the production?

The integration of the detailed planning is important and should be better integrated with the SCADA and the superior systems as ERP and MES. Figure 4.24 shows how detailed planning can be integrated into an automation system containing SCADA, ERP, and MES.

The detailed planning will be based on the current data, historical data, and configuration data from different type of databases. The amount of information can be huge and different types of data analysis must be used to be able to make the correct or best conclusions.

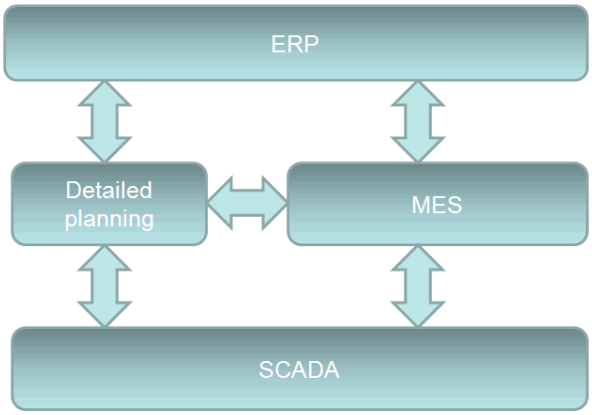


Figure 4.24: Integration of detailed planning in an automation system (www.techpress.no: AUG-09).

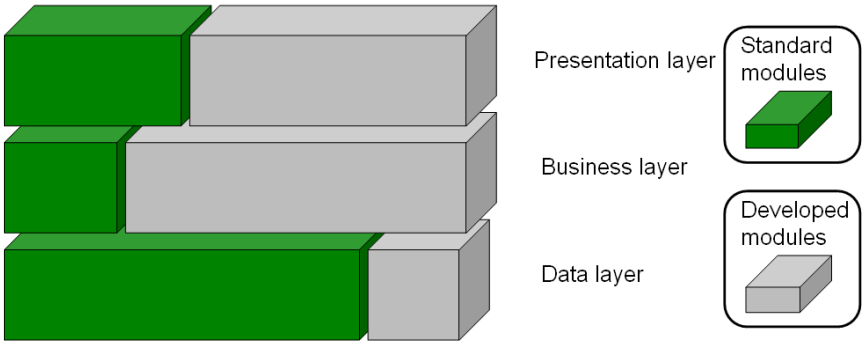


Figure 4.25: The development system for a SCADA system consists of a set of standard modules, and can be extended with developed modules for a specific plant, process, or production system.

4.14 Integrated Operations

In the Oil and Gas industry, Integrated operations (IO) refers to work processes facilitated by new information and communication technology. Collecting of information, multi-discipline collaboration and communication is important when performing oil and gas exploration and production. IO is all about linking professionals and experts across disciplines, systems, organizations, and geographic location. Access of information, in real time, providing better and faster decisions at all levels in the system, which will lead to increased production and higher security. The goal will be to have the right data to the right people at the right time. The data can be any combination of information regarding the people, the process and the technology used. IO means that the right data will be available on any location, in a plant, on a rig, in an office, or in a meeting room.

4.15 System Development

A SCADA system has to be developed and configured to a specific plant, process or production systems. A modern SCADA system like System Platform from Wonderware (2010), Proficy iFIX (Interlution) from General Electric (GE)(2010), CitectSCADA or ClearSCADA from Schneider Electric (2010) consist of a set of standard modules and building blocks. The building blocks are normally object oriented building blocks meaning that a new system consists of the standard modules and any combinations of the building blocks. New or extended building blocks can normally easy be developed using object oriented development principles. The developed building block consists of system functions (business layer) and specific HMI elements. Figure 4.25 shows a SCADA system with the standard modules and the developed modules, often the developed modules are mainly for the business layer, but also for communication with special hardware devices and specific HMI elements.

Since SCADA systems can be any size and combination of devices, the SCADA software must be

configured to each process systems.

The design of such a system starts with a requirement and analysis phase defining the system integration between GUI, alarm system, controller communications, control loops, discrete applications and configuration system(s). The type of system to use must be decided, selecting the DCS or PLC approach, or a combination. Part of the presentation layer, business layer and may be the data layer must be programmed together with a tag mapping and integrating the configuration.

The developed system must have a commissioning and start-up phase for a specific plant including some troubleshooting. The system must also contain some sort of operator training modules and system documentation.

The system must also have the ability to change to meet process requirements over the life time of the system.

4.16 System extension or replacement

Process plants today need to be able to react quickly to factors such as:

- the demand for product customizing,
- shorter, more frequently changed production runs,
- the need to produce products faster and more cost-efficiently.

There is a growing need for an installation of more flexible processes solutions. There are three basic upgrade paths to extend a process plant:

1. Complete replacement of the system; Find a system, shut down for an extended period of time, tear out the old system and wire in the one, often the most expensive path,
2. transitional upgrade of elements of the system; Replace part of the system, can be any combination of hardware and software,
3. a 'plug in' system to obtain the required functionality; Extend a current sub system or module, can be any combination of software and hardware.

4.17 Oil & Gas: Integrated Operations

In the Petroleum industry production will always be in focus. Integrated Operations (IO) is a way to enhance collaboration by using new information and communication technology. IO is defining a new working process by using multi discipline cooperation. The main contribution is a communication link between offshore platforms and land-based offices. This communication link needs to support high speed communication allowing for video conferences and sharing data. This way it is possible to share knowledge faster between the offshore platforms and land-based offices, and between the offshore platforms. Human experts can also be available for several or all offshore platforms at the same time.

4.18 Size of a Process Control System

A process Control System (SCADA system) can be from small systems containing only one computer up to large network of computers. Some examples of large systems are:

1. The oil drilling platforms on Gullfaks in the North Sea consists of three platforms Gullfaks A, Gullfaks B and Gullfaks C. These platforms have been operating since the second part of the 1980s. The SCADA system was based on Unix in the 1990s, but newer systems are based on Windows. These platforms have about 17.000 I/O signals (tags) with about 200.000 connection points. The Windows based system consists of about 100 servers and about 100 operator terminals. The platforms are using Modbus TCP (Industrial Ethernet) for communication between the platforms.
2. A large process system in Norway is the gas power plant at Kårstø, see Figure 4.5, containing more than 550 km of instrumentation cabling and more than 65.000 control tags (expanding upwards 100.000). The process systems make use of more and more standard IT systems, and contains more then 60 servers and the operators need more qualifications than just knowledge about $4 - 20$ mA I/O devices.

4.19 Security of Process Control Systems

Today most of the PCS is mixed within the network of the company to be able to share the information in this network. PCS has some properties which differ from standard IT systems such as:

- installed in locations that can be difficult with physical access,
- using proprietor operating systems and/or protocols that have not been tested for security reasons,
- the software is not updated due to previous points,
- using an external protocol "port" for configuration and maintenance. The same port can be used by a hacker,
- illegal access to one of the computers within the network for accessing the PCS.

Part IV

Autonomous systems

Chapter 5

Autonomous Systems

5.1 Introduction

Different people may have different meaning of what an autonomous system is. Most people think about a self driven vehicle, like a car. An autonomous system can be defined as 1) a system that can take decision and action without any human intervention or 2) a system that can automatically perform a predefined set of tasks under real world conditions. These systems can be autonomous vehicles such as cars, ships, buses and clearing devices to name a few. However, autonomous systems can also be a collection of routers and Internet Protocol (IP) networks managing the communication traffic on a network like an intranet or Internet, robots which can perform desired tasks in unstructured environments without continuous human guidance, and predictive maintenance in an industrial system indicating when the next maintenance period is necessary without risking an accidental interruption in production to name a few other types as well.

The reason for an autonomous system is often refereded to as the four Ds; 1) dangerous, 2) distant, 3) dirty and 4) dull, and should increase the uptime of the system, reduce the cost and increase the safety. Autonomy consists of several levels of complexity as mission and environment complexity and can be divided into 1) system autonomy, 2) task autonomy, and 3) mission autonomy.

5.2 Automation versus Autonomous

A system can not be defined to be either of an automation type or an autonomous type, in stead four keys aspects represent the system can be used (Murphy 2019). These four keys aspects are 1) plans, 2) actions, 3) models and 4) knowledge representation, as shown in Figure 5.1.

The plan key aspect is how to handle the plan for the system. An automation system has normally a plan, made by a human, and the system will only execute the plan. An autonomous system will also need

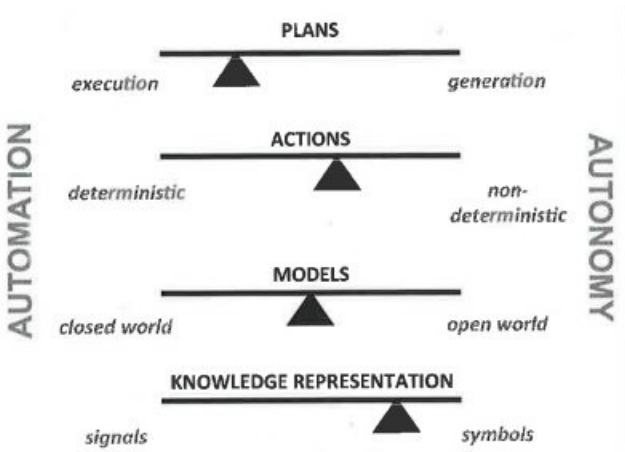


Figure 5.1: The four key aspects of an automation and autonomous system (Murphy 2019).

to generate the plan first before the system can execute the plan. Generation of the plan will include mission planning, resource planning, scheduling and when to use the plan.

The action key aspect is about deterministic and nondeterministic actions. Deterministic and non-deterministic systems should be tested differently as guidance, navigation and control often relied on deterministic algorithms. Testing algorithms for non-deterministic systems is still an open research question (Murphy 2019). A smart system will often be a non-deterministic system, based on artificial intelligence (AI) techniques.

The model key aspect defines how much a system knows about the "world", a closed world approach means that everything possible is known a prior. This means that a "world" model has been programmed into the system and any outcome should be known. If the outcome is not known, a deviation like an alarm or a shutdown may be the result. An open world approach assumes that the outcome for the system, as a possible list of states, objects and/or conditions can not be completely specified (Murphy 2019). In this case must the system use algorithms, like machine learning, to update the model(s) with the missing states, objects and/or conditions. A challenge is the frame problem (Murphy 2019) referring to the challenge of currently identifying what is changing and what is not changing in the world. The system needs computational power for updating the status for items and/or states that are changing but should reduce the update frequency for each item and/or state to limit the need for computational power. The frame problem has to be included when designing such a system as the designers have to be aware of how assumptions about the world impact the adaptability of the system and who the system should monitor the validity of these assumptions.

The knowledge representation key aspect defines the type of information that the systems handle. An automation system is mainly based on signals or raw data from sensor devices, while an autonomous system is based more on higher level of information. The higher level of information is mainly symbols, based on fusion of signals and raw data from sensor devices often combined with external information converted into labeled objects and/or areas.

So the difference between these two type is by the amount of adoption, learning and decision making integrated into such a system. Automation systems are instructed to perform a set of specific tasks with well understood parameters that are known ahead of time while autonomous systems are advised to make the right decision or action in an evolving, non-deterministic environment. The decisions made by an automation system are often based on predefined heuristics. An autonomous system learns and adapts to dynamic environments and evolves as the environment around it changes. Based on historical data sets, such systems will also be able to learn faster and more reliably then what will be reasonable for humans.

5.3 Overview of building blocks

Autonomous system is systems that can make decisions and actions without any human intervention. In these cases must the system have access to as much information as possible to be able to make these decisions and/or actions, the information must be structured in such a format that the system can understand it and utilize it to be able to make the necessary (and correct) choices. Such a system must have a set of building blocks (or modules) for collection the information, structure the information in such a way that one or several decisions can be made and actions performed based on any decisions. An overview of one way of structuring these building blocks is shown in Figure 5.2.

This figure shows the input of information from sensor device to collect information from the environment and external communication for additional information. This information needs to be structured and formatted in such a way that the system can use this information to make any useful decision and/or action, this is the main function of the perception block. The decision block must define any goal for the system based on the current situation, and define how to reach that goal also based on the current situation. This information will be the input to the actuation block for performing any action to reach the new goal.

An autonomous system will often be some type of robot or unmanned aerial vehicle (UAV) consisting of some type of body, sensing and intelligense, according to Figure 5.2. Body for the physical outline of the system, sensing for situational awareness, and intelligense for perception and decisions.

5.3.1 Input information

An autonomous system is defined as a system that can make decisions without human intervention and such a system will need as much information as possible to make a decision. The system needs information

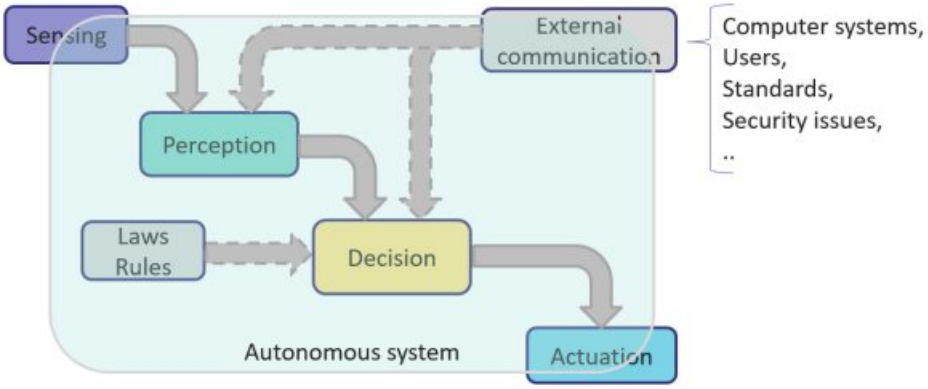


Figure 5.2: One way of defining the building blocks of autonomous systems. Input information from sensor devices and external systems, structure the information in a useful format (perception), define a goal and make a plan to reach that goal based on any decisions, and control the system for that goal.

about the environment which is normally used by collecting information from sensor devices, like radar devices, GPS devices, ultrasonic devices or lidar devices to mention a few. A radar device will give information about surrounding objects and the movement of these objects, using radio signals. A GPS device will give information about the position for this system as long a GPS signals can be received from the satellites. So GPS will normally not work inside buildings, in underground parking lots nor in tunnels. Ultrasonic devices can be used to check for surrounding objects and the distances to these objects, using sound in the range of 50 kHz. Ultrasonic devices can detect both moving and not-moving objects. Lidar devices uses laser light to check for surrounding objects and the distances to these objects.

Information can also be connected by external communication like with other computer systems, with users (by some sort of user interface (UI)) and according to standard and security issues.

5.3.2 Perception

The system will receive a lot of information, and in different formats and structures. The perception module will need to structure and organize this information so that the autonomous system can utilize information. This will be both raw data from sensors, at several information levels, and more structured information from external systems. The module must analyze, preprocess and structure the information so that it can be used further by the decision-making processes in the system. The data analysis will be to extract as much valuable information as possible from the data sets. The preprocessing will include scaling the data, adapting it to the time and place of each individual information source. The structuring of the data will be to store all the information in a common format and structure so that the information can be utilized further by the decision-making process.

This is a large research area for autonomous systems involving artificial intelligence (AI), machine learning (ML), big data and sensor fusion.

5.3.3 Decision

The decision module consists of two main blocks, planning and goal management. The planning is to make plans so that the autonomous system can achieve its goals. Goal management is to check that the system reaches its goals, which will often also involve optimization challenges.

The autonomous system must be self adaptive so that it can adjust behavior through learning and reasoning. Reasoning will be the ability to explain actions and decisions. The self adaptive capability of the autonomous system will affect both the planning processes and the goal management. The adaptive part will depend on the uncertainty of the environment.

This will be a complex module as the planning process will depend highly on the environment and the goals for the system. The environment will be based on all the information that the system will use as input to the perception module, any additional information from external systems like standards, and any laws and/or rules affecting the system. The goals for the system will depend on the goal management which may need to choose or select among a set of goals.

This is a large research area for autonomous systems involving artificial intelligence (AI), machine learning (ML), big data and sensor fusion.

5.3.4 Actuation

The actuation module is based on controllers for controlling the autonomous system. These controllers will be closed loop controllers with feedback from the perception and decision modules. They can be either proportional integral derivative (PID) based controller or model predictive controllers (MPC).

5.4 Industrial systems

The comparison between the degree of automation and autonomy of an industrial system is not easy and using 5.1 can be a good starting point. The paper (Müller, Müller, Talkhestani, Marks, Jazdi & Weyrich 2021) makes some definitions of such a systems and define three levels of such systems as 1) Industrial Automation System, 2) Intelligent Industrial Automation System and 3) Industrial Autonomous System (see Figure 1 in the paper). The industrial autonomous system will be an industrial automation system with the expansions 1) adaptability without human intervention, 2) self-governance and 3) self-containedness. Adaptability will be an automatic change of shape and/or behavior based on the environment. Self-governance is managing the system resources without external interference. Self-containedness is that the goals and scope of the system are formulated precisely.

The paper (Gamer, Wallstadter, Kloepper, Bauer & Isaksson 2019) defines five level of autonomy for an autonomous plant and how these levels are affecting the control room operation, field operation and maintenance, and planning and scheduling the plant. These levels are (Gamer et al. 2019):

0 No autonomy, humans are in complete control without assistance.

1 Assistance with or control of subtasks. Humans are always responsible, and specifying set points.

2 Occasional autonomy in certain situations. Humans are always responsible, specifying intent.

3 Limited autonomy in certain situations. System alerts to issues, humans confirm proposed solutions or act as a fallback.

4 System in full control in certain situations. Humans might supervise.

5 Autonomous operation in all situations. Humans may be completely absent.

5.5 System development

The development of autonomous systems require knowledge in many disciplines / areas such as 1) cognition like planning, learning and adaptation, 2) motion and maniplulation, 3) sensing, 4) perception, 5) localization and mapping, 6) mechatronics, 7) data processing and 8) energy. Also konowledge about 9) data communication with external systems and not least 10) human-machine interaction. Figure 5.2 highlights many of these areas.

5.6 Examples

Autonomous vehicles

SAE International, also known as the Society of Automotive Engineers, a globally active professional association and standards developing organization for engineering professionals in various industries, has defined six levels $(0-5)$ for autonomous vehicles. These levels are defined as the standard $J3016$, named "Levels of driving automation", and is shown in Figure 5.3.

A vehicle will need sensors to get information about the environment around the vehicle and one set of sensors is shown in Figure 5.4. A set of radars and cameras on level 2, increase the number of radars and cameras on level 3, and include lidars (light detection and ranging) on level 3, 4, and 5.The number and type of sensors will increase when increasing the automation level to get more information about the environments, and will often include sensor fusion on higher levels to get as much information from both sensor devices and by combining the information from different sensor devices. Cameras take images and vision processing is needed. Radars use radio waves to determine the range and angle of objects, while lidars uses light in the form of a pulsed laser to measure range and angle of objects.

The information from the sensor devices must be handles by the perception module to convert the information into understandable objects so the decision module can make some sort of action based on

Figure 5.3: The SAE standard $J3016$, also known as "Levels of driving automation". (www.sae.org - JAN-20).

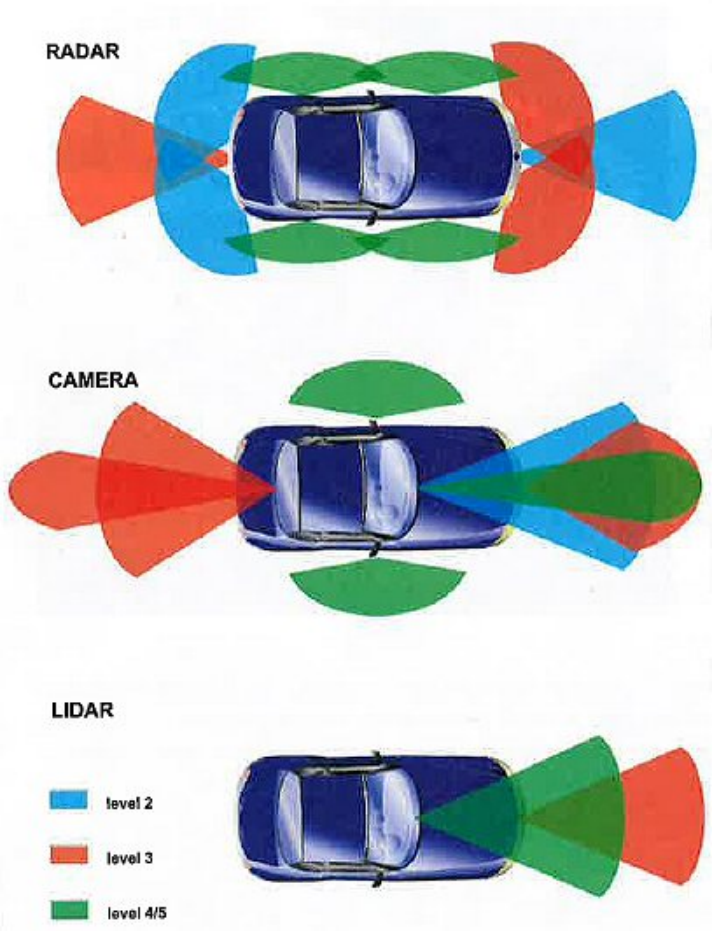


Figure 5.4: An increase in automation level should increase the number and types of sensor devices. Radar and camera on level 2, more radars and cameraes on level 3, and adding lidar on both level 3, 4 and 5. (www.infineon.com - JAN-20).

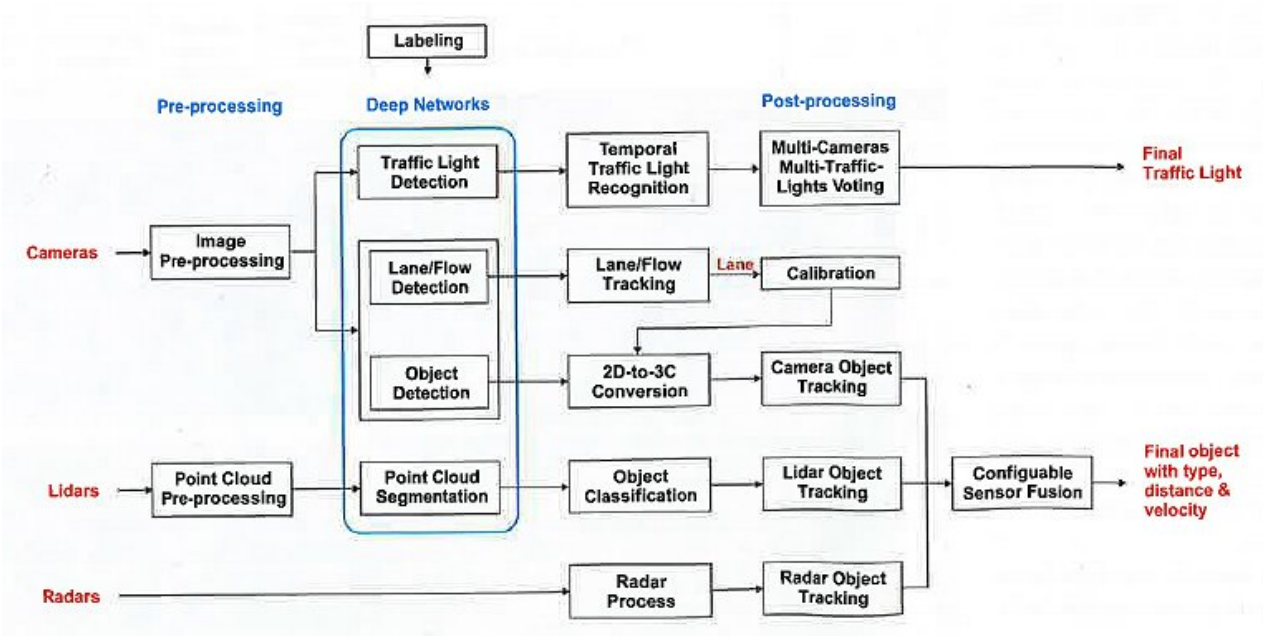


Figure 5.5: The structure of a perception module converting sensor information from cameras, radars and lidars into objects (www.apollo.auto - FEB-20).

the surrounding objects. An example of a perception module for an autonomous vehicle, using radars, cameras and lidars, is shown in Figure 5.5. First some preprocessing of the information from cameras and lidars, then machine learning using neural network with deep learning to make labels for features in the information, and post processing to convert the label information into objects. The post processing includes sensor fusion to combine the information from several sensor sources to better make the objects.

Part V

Digitalization

Chapter 6

Introduction

Digitalization may have different meaning to different people so a good start will be to define digitalization. Digitalization is the adoption or increase in use of information stored in computer systems by sharing, reusing and/or combining the information (Columbus 2019), (Westerman, Bonnet & McAfee 2014), (Regjeringen.no n.d.), (Isaksson et al. 2018), (Yokogawa 2018), (Fortinet 2018). This means to use technology, often with some sort of "intelligence", to renew, simplify and/or improve existing actions or processes. Digitization is the action or process of digitizing, converting data into a digital form so that it can be combined and/or reused by other computer systems. Digitization includes storing the digital information in such a format that the information can be reused by different types of software application and may include information from both text, sound, images and/or video sources. Digital transformation is based on digitalization and is the process of creating new digital products, digital business or business models. The relationships between digitization, digitalization and digital transformation is shown in Figure 6.1.

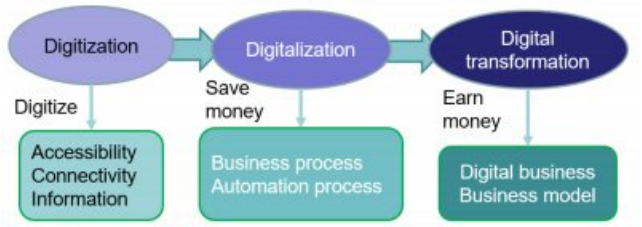


Figure 6.1: Digitization, digitalization and digital transformation.

Figure 6.2 gives an overview of digitalization where collected information (from digitization) can be used for analyses, modelling and/or prediction for improving actions and/or processes.

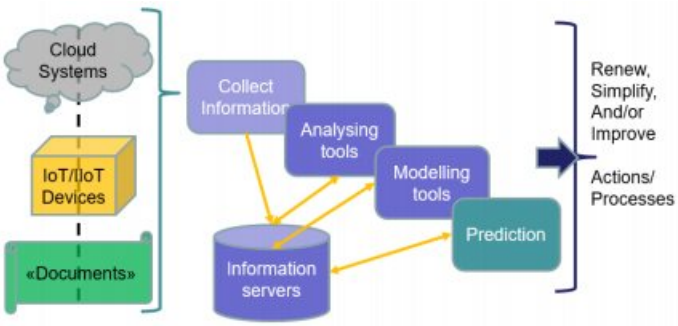


Figure 6.2: An overview of digitalization where information is collected and digitized into standard formats that can be reused by different tools for improving existing actions or processes.

The key questions regarding digitalization are 1) how to collect the information, 2) how to store the information, 3) where to store the information, 4) how often should the information be stored (sampled), and 5) in what kind of format to store the information. This storage format must be in some sort of data

file stored on a disk, and will affect both the internal data format of this file, the file type (text, binary or database), and the file size. However, an important part with digitalization is to use a data format so the information can be reused. Any information that can be reused, can be combined and structured in many different ways and stored again and again.

So digitalization is important when there is a need to gather and analyze information, use the information for calibrating of mechanistic models, use the information to develop, calibrate and validate data driven models, or extract information for making some sort of decisions and/or optimizations. Digitalization is also important when collecting information using data driven models for prediction and simulation. Often this information will be collected over time, so the information should be stored in a structured way and in a standard data format. The focus regarding digitalization should be to reuse the information, so always have in mind how to make your information available for others.

6.1 Text Information

This section will deal with digitization of text information only, as this may be an easy way to show the start of digitalization. Digitization of text information can store the information in different file formats. The file format define how the information is structured when storing on a file. Some of these file formats are MS-Word (docx), PDF (pdf), MS-Excel (xlsx), Comma Separated Values (csv) and Extensible Markup Language (xml). Text file formats like docx, pdf and xlsx focus on the presentation of the information while formats like csv and xml focus on the information itself. The csv format is used mainly for storing data sets while the xml file format is used mainly for information in web systems. Selecting an appropriate file format when storing the information can be important for reuse of the information. Digitalization should use a file format focusing on the information itself, not the presentation of the information. Any software applications must support the chosen file formats to be able to reuse the information. It will be more difficult to reuse information from a file format where the focus is presentation as extra information must be included on how to extract the information from the presentation information. This means that information stored in .docx, xlsx or pdf file formats may not be digitalization, while information stored in xml or .csv file formats may be digitalization, depending on the supporting file formats in the software applications. An example showing the differences of these file types can be by making and storing the following table:

Timestamp	Temperature	Level
11 : 45	22.5	1.56

in MS-Word and MS-Excel, and save the table in the docx, pdf, xlsx, xml and csv file formats. Figure 6.3 shows the table in MS-Word, and MS-Excel to the left, and the table presented from the pdf and xml files to the right.

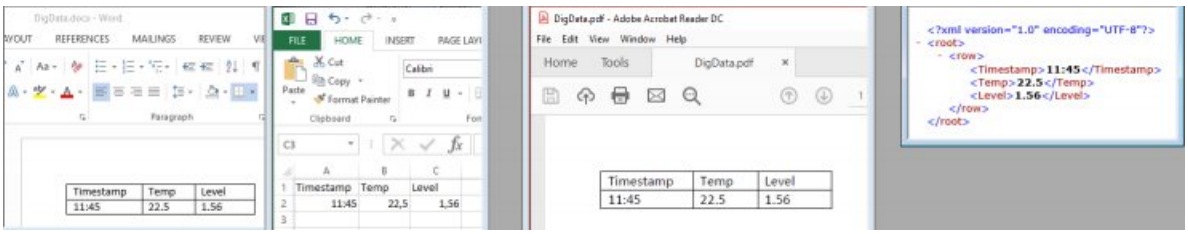


Figure 6.3: The table created in MS-Word to the left, and in MS-Excel in the second image from the left side. The third image from the left side shows the table from the pdf file, and the image on the right side shows the table from the xml file.

The *Save As* function in MS-Word was used to create the pdf and docx file types, while MS-Excel was used to create the xlsx, xml and csv file types. The table data was saved in files named *DigData* with different extensions (file types) and Figure 6.4 shows the information from the File Explorer in Windows. Pdf, docx and xlsx file formats focus on formatting and presentation of the information, and is the reason why these files consists of more bytes than the csv and xml file. Computer systems are storing the information in blocks and the block size for this Windows system is 1024 bytes, giving a minimum file size of 1 kB. More information about a file like the file type, when the file was created and the exact file size is also available by placing the mouse cursor over the file name, as shown Figure 6.4.

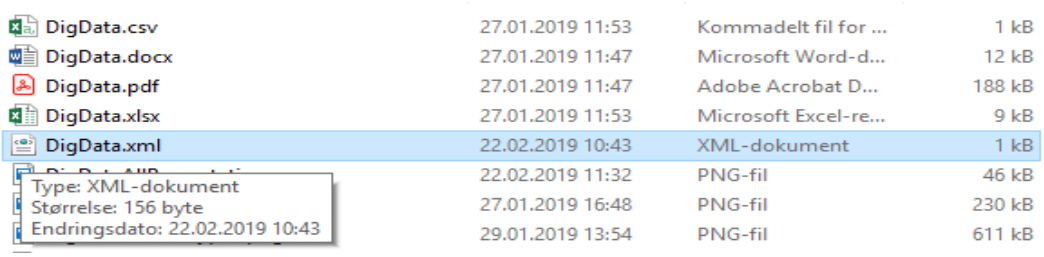


Figure 6.4: The different file types for the DigData file, csv, docx (MS-Word), pdf, xlsx (MS-Excel) and xml. The right column indicates the size of each of the file types. More information is available for a specific file by placing the mouse cursor on the file name (Type, size (størrelse), and last changed (endring)).

Based on the information from Figure 6.4 the pdf, xlsx and docx are the files containing the most information. The contents of these files are shown in Figure 6.5 and it is not easy for a human to find the data from the tables based on the contents. This is the reason for the software applications to have support for these file formats, to be able to transform the information from the file format into an internal format for the applications. A challenge can be to specify what will be the actual information that the application should use, and a manual operation can in some cases be necessary to select the type of information that should be transformed from the file format. The contents of these files are shown using Notepad, a standard Windows application for showing the "raw" contents of a file.

The contents of the xml and csv file are shown in Figure 6.6 where the focus is the information and it is more easy to get the table data, both the data types and the values. Any software application should at least support one of these file formats (csv or xml) as these file formats are easy to implement. Some file formats may also be proprietary so it can be difficult and/or expensive to get a specification for implementing these file formats.

Note that the separator in the csv file in Figure 6.6 is semi colon and not comma because comma in this case is used as the separator field for numeric values. The csv file may have a header line describing the data fields in the file. The xml file has the description as field commands for every line. Implementing the csv or xml file format is strait forward in programming languages like MATLAB, Python, or C#.

6.2 Interface

An interface is necessary as a connection point between two or more pieces of electric equipments, or between a human and an electric equipment, in order to exchange information. The human interface is normally called a human machine interface (HMI) or a graphical user interface (GUI), but is not that important for digitalization and will be covered by this document. The interface for connection between for two or more computers to exchange information consists at least of the following types:

- file; the applications must be able to read and write files with the same type of format,
- database; the applications must contain an interface to the database manager for sending/receiving database requests. The two main types of database types are the relational type based on relations between tables and the time series based having independent tables with time series data,
- application programming interface (API) defining how two or more applications can exchange information using operating system functions, either locally or distributed in a network.

6.3 Network

In digitalization, the information will be reused by different type of applications, and these applications can run on different computer systems. These computer systems must be interconnected in a local area network (LAN) where the information can be stored in one or several servers in this LAN. The information can then be reused by any computer system in that network (LAN) with access rights to the server. Such computer systems in a LAN may also be named computer nodes (or just nodes) as a computer node is a computer system with interconnection capabilities. A server is a computer node

Figure 6.5: The contents of the pdf file on the top, the docx file in the middle and the xlsx file at the bottom. Most of the data is for presentation of the information, and it is not very easy for a human to find the wanted information.

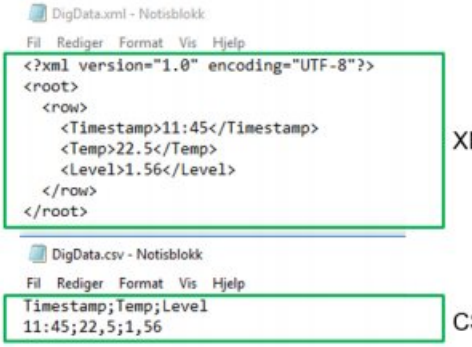


Figure 6.6: The contents of the xml and cvs file, the xml file on the top and the csv file at the bottom. Both file types has focus on the information, the xml file with commmands while the csv file used semicolon to separate the data fields.

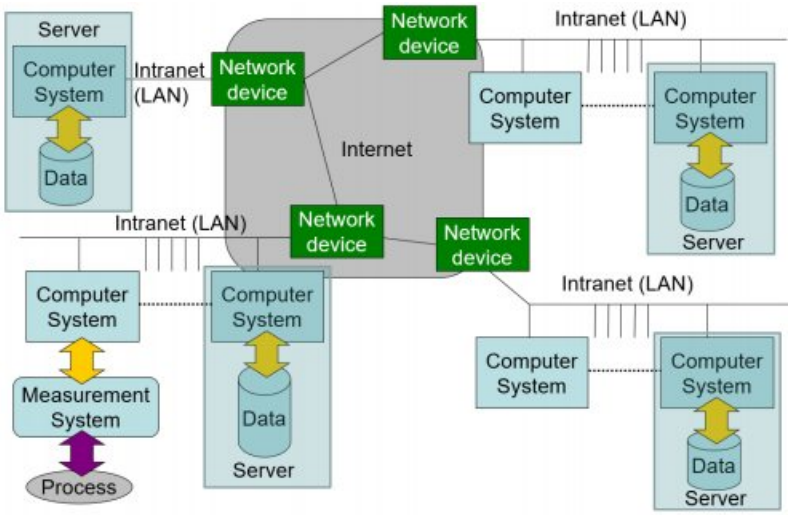


Figure 6.7: A set of computer system interconnected in an intranet, and a number of intranets interconnected to internet.

with a specific responsibility or ownership, like the owner of a specific resource such as a database (or file). In this case the resource will be the information (data) and it can be a good practice to name it a data server (or information server) to specify the type of server. Normally there will exist several types of servers in a LAN like communication servers, data servers, file servers, mail servers and print servers to mention some.

Internet is a network based on interconnection of many LANs, and each of these LANs are named intranet to separate them from Internet. A company or an organization is the owner of a specific intranet and the information in this intranet, and the nodes in an intranet are grouped in domains like *usn.no*. The cloud is other intranets in Internet so a cloud computer is just a computer system in another intranet. A cloud service is a software application running on a computer system in the cloud. An example of several computer systems interconnected in an intranet is shown in Figure 6.7. This figure also shows that these intranets are interconnected into Internet using network devices. The main functions of the network device are as a router and a firewall; router to handle routing of messages in the network and firewall to handle security issues. The router will optimize the routing of messages within the Internet while the firewalls will contain a set of rules of which messages that are allowed to enter the intranet or a computer system in the intranet.

A large challenge with digitalization is that these systems, mainly the software applications on the different computer systems, are not able to communicate. System integration is the main success factor for digitalization! The system design and system structure, both for hardware and software, is important for digitalization.

6.4 Protocol

Computer systems will need a communication protocol for interconnection. A protocol is a set of rules defining how the computer system will share the communication media and how to decode the information sent on this media. The communication media is the physical connection like a wire, optical fibre or radio channel (wireless). One popular protocol is TCP/IP (Transport Control Protocol/Internet Protocol), the protocol used for interconnection on the Internet. There are many different types of protocols with focus on different goals for the connection(s) like speed, throughput, number of nodes, security and infrastructure to mention some. Protocols are important for exchanging information both internally in a computer system and externally between computer systems. In order to use and share digital information protocols are important. Digitalization will depend on a robust infrastructure involving the computers, intranet and protocols.

Interfaces must also have a protocol to describe how information can be exchanged between applications, or applications and cloud services, using files, databases or API's.

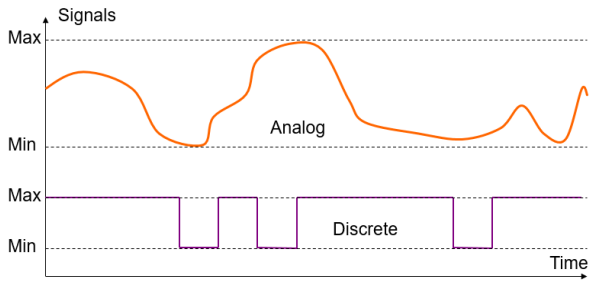


Figure 6.8: An analog signal and a discrete (digital) signal shown in the time domain. The discrete signal can be triggered by the state changes, while the analog signal must be sampled at a fixed time interval.

6.5 Information

The information can be raw data, structured data and unstructured data. Raw data is information from the real world as either analog or "discrete", most of it being analog. The analog information is a continuously changing signal (for instance a temperature) while "discrete" information is a signal changing between two states (on/off signal). The "discrete" information can be stored as is (only two states), while the analog information must be digitized first. For the "discrete" information, only the state changes are of interest, while the analog information changes all the time and another approach must be used. For analog signals measurement systems will use some time to convert the analog information into a digital representation of the value, using a sampling time. Sampling time will decide how often a digital representation of the analog signal can be recorded. Figure 6.8 shows an analog signal and a "discrete" (digital) signal in the time domain. The digital signal can be checked and logged at the state changes. However, the analog signal must be sampled at fixed time interval, and sufficiently frequent that the curve information is kept.

Unstructured and structured data is information already stored in a computer. Structured data is often stored in a database or in documents with meta data. The meta data is a description of the unstructured information, a way of describing the data. The integrity of the information is important to reuse and/or combine the information.

6.6 Cloud service

Cloud services are scalable services that are delivered via Internet. The reason for these services are a change of the business model. Traditional services inhouse requires an IT department for running and maintaining the computer systems, while the cloud services will require less IT knowledge and cost. The cloud services are rented from another company with the IT department for running and maintaining the computer system, as part of the business model for that company. These services will run on a cloud computer, in an intranet connected to Internet. You will access these services using some sort of client via Internet. These services will also be a type of a server in the network, that can be accessed by a client software like a web browser.

Services based on cloud computers can be either an infrastructure, a platform or software application. Digitalization may depend on such services so cloud services can be important modules in digitalization strategies. These services, infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) are shown in Figure 6.9.

IaaS is useful for data servers and storage, PaaS for software development, and SaaS for software applications and services.

Infrastructure as a service (IaaS) is the computer resources you normally have in your own data center or computer room like servers, networks and storage. IaaS will provide the physical servers (hardware) (including memory, hard drives, and CPU) and the infrastructure, and keep it up to date. You need to provide the operating system, the software that you want to run, and all data that you are using. You will run your own virtual servers with your software and data on their physical servers. An example can be that you want to run your own data collection system on a specific operating system and save the data in a specific format on a specific storage device, but you will not manage the computer system nor the infrastructure.

Platform as a service (PaaS) is the computer resources you normally need to develop software or

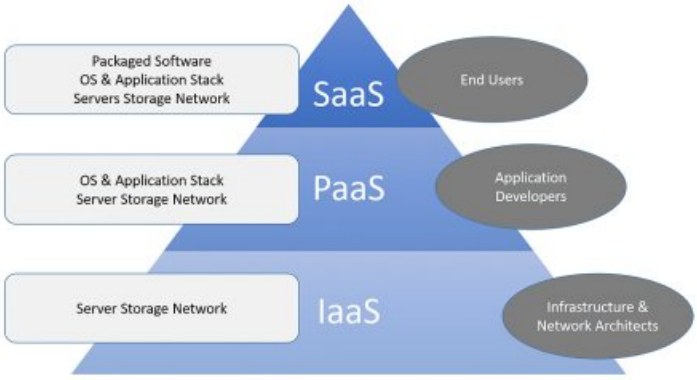


Figure 6.9: The different types of cloud services, infrastructure, platform and software. (www.medium.com: oct-18).

Responsible	Private	IaaS	PaaS	SaaS
Data				Customer
Access rights				
Connections				
Access control				
User accounts				
Infrastructure for identification				
Application				
Network control				
Operating system				
Servers				
Network				
Computer centre				Supplier

Figure 6.10: The responsiblities of the customer (orange) and supplier (violet) for the different cloud models. Private means that the customer has a private cloud system (Source: ComputerWorld:19).

systems like IaaS plus the operating system and the development tools. You may provide the extra development tools that you need and the data while the rest is automatically maintained for you including the newest versions of the platform software. You will pay only for the resources that you are using. You develop your software, define the storage structure and deploy and run it on the platform. An example can be that you want to develop your own data collection system and saving the data in a specific format on the storage device, but will not manage the development tools nor the computers used.

Software as a service (SaaS) is all the computer resources as a complete solution. These services offer little or no customizing, but in return you don't have to worry about configuration, maintenance, updates, nor hardware. You rent the software and will always run the latest version and pay only for the resources that you are using. An example can that you do not want to buy the Microsoft Office license and upgrade to a new version every other year because you are not using the software that often, in stead you just want to rent the latest version whenever you need the software.

Figure 6.10 shows the responsibilities of the customer and the supplier for the different type of cloud models.

The cloud service requires Internet access and some client software for accessing the service. This client software can be an application like web browsers, thin clients, terminal emulators and/or mobile applications.

6.7 Business models and work processes

Companies have some sort of business model, and work processes to fulfill the business model. Part of the work processes may be production of some kind. These work processes may also include the ordering and collection of materials for the production, the distribution of the production material, and any feedback from the market regarding these products.

The technological developments are moving so fast that new areas are digitized all the time giving more and more information about both the working processes and production processes. Companies are already using digital tools in one form or another and digitalization is not the goal itself but a means of achieving a goal. Digitalization is nothing new, what is new is that the technological development goes so fast that new areas are digitized all the time, and the digitalization is again increasing the development speed and give more information available for better decisions. The result is a large amount of information about the processes, and the companies can reassess any working processes, and may be also evaluate any business model, based on this information. During this reassessment it is important to involve people, as people and technology must work together.

Examples of some modern business models where digitalization is used are:

- Airbnb: Providing a platform (cloud system and services) for free, but takes a 3% commission on the host's earnings and asks for a booking fee of $6 - 12\%$ from guests for every booking they make,
- Uber: Takes a $5 - 20\%$ commission on all the rides made through its cloud service. Payments are done automatically using the Uber application,
- Tesla: As the drivers prepaid their energy for the next 10 years, Tesla invests the money in the Supercharger network and makes sure the energy provided becomes cheaper. Furthermore, competitors pay Tesla for every car using their network,
- Nike: customers can order personalized shoes online in hundreds of color combinations. Nike uses digital tools making product design and manufacturing faster and more efficient supporting a global supply chain,
- Electricity company: In Norway the electricity consumption is measured by AMS (automatic electric meter system) devices measuring the consumption every hour and the information is transmitted to the electricity company. Every consumer will be billed every month based on the actual consumption for that month, based on digitalization of the electricity consumption.

Changes will often include the usage of mobile devices like smart phones or tablet computers as an interface between working processes and the users for digitalization of working process information. This is the reason why the users should be involved as early as possible when planning digitalization of working processes.

Chapter 7

Information

7.1 Introduction

The information can be collected from sensor devices, mathematical models like mechanistic models or data driven models, other type of models, from a laboratory, or external systems. External systems can be from manual systems like smart phones or tablet computer, or from sources on intranet or Internet. Sources on Internet can be external services like weather forecast, traffic overview, etc. The information should not only be from the production lines, but also from the work processes, the market using the product, and if possible, from the business model. As the information is digitized, software applications can be used to evaluate the information, propose changes, and predict any responses. More information can make the analysis, evaluation, proposals and prediction even better.

7.2 Handling

The information must be collected, converted to a digital form, and validated. This validation involves preprocessing of the data sets. When the data sets have been validated they can be merged, if several sets. Before merging these sets they should be aligned in time and space. This preprocessing of the information involves marking of outliers, check for missing data, filtering and merging.

7.2.1 Outliers

An outlier is an observation (sample) or variable that lies outside the overall pattern of a distribution, or is abnormal compared to the major part of the data (Esbensen 2002). Outliers can be due to some sort of malfunction or error in a system and should be marked (kept out) in the data sets. Outliers can still be useful and one can decide later on either to use them or not. Outliers are observations or variables that are not belonging to the same populations as the other observations or variables, however extreme observations or variables are not necessarily outliers. Outliers can for example be useful for a "fault detection" system.

7.2.2 Missing samples

Missing samples are often due to some sort of malfunction or error in a system not giving any samples at all at a given time. There are several approaches for handling missing samples (or data):

1. The missing samples can be kept out of the data set by removing all corresponding samples in the data set.

2. The missing samples can be replaced by interpolation, like linear interpolation. For linear interpolation we need to know the previous sample (x_1, y_1), the current x_2, and the next sample (x_3, y_3). The missing sample y_2 can be estimated as:

$$y_2 = \frac{(x_2 - x_1)(y_3 - y_1)}{(x_3 - x_1)} + y_1$$

Never replace missing samples with any fixed value like 0 which will lead to very unreliable interpretations. If you are not sure, keep these samples out of the data set.

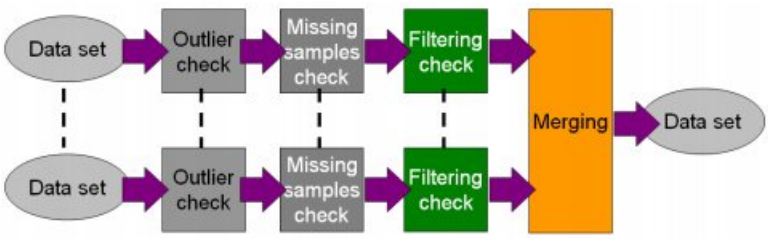


Figure 7.1: The handling of different data sets by checking for outliers and missing samples, filtering and merging of the data sets.

7.2.3 Filtering

Information from sensor devices must be filtered by a low pass filter (Wheeler & Ganji 2004), either a Finite Impulse Response (FIR) or Infinite Impulse Response (IIR) filter (Ifeachor & Jervis 2002). The FIR filter is without any feedback from the output signal, just an average filter where the output signal y_k will be an average of a set of previous input signals like:

$$y_k = \frac{\sum_{i=1}^{N} x_i}{N}$$

where N is the number of samples. Depending on the time constant in the system a useful value for N is in the range of 4 to 256. The IIR filter, or exponential filter, has feedback from the previous output value, and is using only the last y value and the new x value, no need to store several x values like the FIR filter. The output signal y_k is:

$$y_k = \theta x_k + (1 - \theta) y_{k-1}$$

where θ is a filter parameter between 0 and 1 indicating the weight of the filter. This weight defines how much of the new value (x_k) should affect the new result (y_k). As an example will $\theta = 0.1$ give a new output signal combined by 10% of the new value (x_k) and 90% of the old filtered signal (y_{k-1}). The filter parameter is a function of a time constant T_f and the sampling interval T_s as (Haugen 2014):

$$\theta = \frac{T_s}{T_f + T_s}$$

Filtering is mainly used for information from sensor devices, but a good advice is to always do some sort of low pass filtering of information recorded from an external system. Note that in some special cases it is the gradient that is of interest and in such cases the signal should not be low pass filtered.

7.2.4 Merging data sets

Information often comes from multiple sources like different types of sensor devices connected to distributed measurement systems, external sources of different formats, and even manual measurements. With digitalization the probability for information from different sources is even higher and before this information can be analyzed, these data sets must be merged together as one data set. Merging means that each set should be aligned to the same time instant and spatial frame valid for all the data sets. Figure 7.1 shows the data handling process of checking the data sets from external sources like sensor devices.

Having only one external source, the merging part will not be necessary.

7.2.5 Storage of information

The information must be stored in some sort of software container, stored on a disk drive as a file. This file may be an ordinary data file, like csv, or a database file. The information can be stored on the local computer node, on a data server in the intranet or on a cloud computer system. However the physical storage should be of the same type independent of the location. The storage is a file, a database or a database combined with a data warehouse infrastructure. Database technologies includes “standard” databases, Hadoop, CouchDB, Cassandra, Pig, Hive, MongoDB and AsterData to mention some.

There exists many different database types, the two most used types for storing information are time series databases and relational databases. In time series databases, the sampled data sets are the main

information, stored related to the sample time and with no relations between the items in the data sets. A relational database consists of a set of two-dimensional tables where the tables are related by primary and foreign keys. In a relational database all type of information can be stored. However, an important aspect is not to store duplicated data. Why is duplicated data a challenge? When updating the duplicated data, how to be sure to update all copies? Databases are using a standard protocol like Structured Query Language (SQL) or Open Process Control (OPC) for communication. Storing the information in a relational database SQL is a good choice, while SQL or OPC is a good choice for a time series database. Using SQL or OPC based databases means that all applications must support this protocol in order to take advantage of digitization, otherwise text files should be used.

A standard file can also be an option for sharing information, but an open format of the file should be used, preferably text-based. Comma Separated Values (csv) or eXtensible Markup Language (XML) are formats used a lot for these types of files, see Figure 6.6. These formats allow standard text so these files can be created or updated automatically or manually by some sort of text editor. A csv file may have a header indicating the type of information where each information field is separated by a separation character like comma or dot. As comma or dot can be a challenge when exchanging between different languages (separator in floating point numbers), other types of separation characters like semicolon, space or tab are also used. Each line will contain samples taken at a specific time. A header will be optional, and will always be the first line if included.

An extract of such a file can be:

```
No;Time;Interval(mi);Indoor Humidity(%);Indoor Temperature(DegC);Outdoor Humidity(%);...
1;25-04-2016 21:35;30;48;10.9;26;0.9;952.3;1.1;2.5;N;996.4;-16.5;0.9;70.8;70.8;70.8;....
2;25-04-2016 22:05;30;48;10.7;35;-1.2;952.5;1.1;2.5;N;996.6;-14.7;-1.2;70.8;70.8;70.8;..
3;25-04-2016 22:35;30;47;10.5;33;-2.2;952.7;1.1;2.5;NE;996.8;-16.3;-2.2;0.0;70.8;70.8;..
4;25-04-2016 23:05;30;47;10.3;41;-2.5;953.0;2.5;3.6;NE;997.1;-14.0;-2.5;0.0;70.8;70.8;..
5;25-04-2016 23:35;30;47;10.1;41;-2.3;953.1;2.5;3.6;NE;997.2;-13.8;-2.3;0.0;70.8;70.8;..
.....
```

where the first line is the header line with a description of each column (field) and then one row for each sample. Each row contains the sample time and the data according to the header. This means that the number of columns will be fixed, while the number of rows depends on the number of samples in each file. Each file from the same collection or system should have the same header, and the same number of columns.

Extensible Markup Language (XML) has a set of rules for structuring the information. Part of these rules are the commands defining the type of information. These commands will be encapsulated with < and >, however the naming of the commands are free and should be an agreement between the transmitter and receiver(s) of the information. Each part of the information will start with the command (<command>) and end with a slash (/) in front of the command (< /command>). Example: <Level> 717 < /Level> defining the current level value as 717 without the unit. Unit can be fixed or defined with another command. These commands can also be in a selectable hierarchy with new commands inside another command. XML will focus on information, not presentation, and can be used in both messages (protocols) and files. File based XML information has the file extension *xml* and an extract of such a file can be:

```
<DevConfData>
   <DeviceConfigData>
      <Id>0</Id>
      <Description>NI-USB-Device</Description>
      <Type>NI-6218</Type>
      <Driver>dev1</Driver>
      <Address>0</Address>
      <Sampling>1,0</Sampling>
      <AnInp>1</AnInp>
      <AnOutp>1</AnOutp>
      <DigInp>1</DigInp>
      <DigOutp>1</DigOutp>
      <IpAddress>0.0.0.0</IpAddress>
```

```
   </DeviceConfigData>
</DevConfData>
<TagConfData>
   <TagConfigData>
      <Id>0</Id>
      <Description>Mobile Phone</Description>
      <Type>AI: GSM Power (%)</Type>
      <DevId>dev1</DevId>
      <DevAddress>1</DevAddress>
   </TagConfigData>
</TagConfData>
```

An important purpose of digitalization is to store the information in such a format that the information can be reused by other applications. In an analyzing phase for a system, the applications that are going to be used should be checked for the type of protocol(s) for sharing information, and the more suitable protocol should be selected. When analyzing and selecting a protocol, focus on file or database based protocols.

The JavaScript Object Notation (JSON) format can also be used in text files as JSON is text based and similar to the XML format. JSON will often separate combined fields in more specific fields than the XML format making it more suitable for defining the contents of objects.

Regarding information from external systems, especially sensor devices, the value, the sampling time and the status should all be stored. The value should be the filtered unit converted value (not the raw sensor value), the sampling time is the time when the value was read, and the status is an indication on how good this reading is (quality). A temperature value is often useless if the time is not included to know if this temperature was gathered now, today or last week. The status is an indication of the quality of the value, if it was read without any error, or with some type of error(s). A good advice can be to also log both the raw value (the value from the DAQ system), the unit converted value, and the filtered unit converted value from the sensor devices. These values can be useful for checking and/or debugging if getting invalid or wrong information into the system. The filtered unit converted value is the output value that should be used for digitalization.

GDPR

The General Data Protection Regulation (GDPR) is a regulation protecting the data of citizens within the European Union and other countries in Europe including Norway. It will affect every company making business with in these countries. This regulation is protecting the information about the citizens that every computer system is storing, saying that the owner of the computer system should protect the data, know the type of data stored, and be able to delete all the data whenever a specific citizen request to be removed from a specific system. The type of information that is protected by GDPR is name, photo, email addresses, social media posts, personal medical information, IP addresses, bank details to mention some. The GDPR covers any information that can be classified as personal details or that can be used to determine the identity of a citizen. Parental consent will be required to process any data relating to children younger than 16. You should have the GDPR in mind when designing a system used for digitalization.

Big Data

Big data is the technology that makes it possible to analyze larger and more complex amount of information fast and with a good accuracy. One way of defining big data is "information that is characterized by a large volume, high speed and/or high degree of variation, and which requires new methods of processing and adaptation to serve as a basis for decisions, increased insight and optimization of processes" (Simon 2013). Big data is based on both internal information in a company and external information from many different source, both structured and unstructured, and used for analysis for giving new information (Andersen & Bakkeli 2015). Structured information will be information from databases and files, and internal systems in a company like the planning and control systems, while unstructured information like images, messages in emails and social medias are information that can not be structured that easy in some sort of tables. So Big Data is about how to combine the structured and unstructured information

for analysis for decision basis and to get new knowledge. Digitalization is the reason for getting both the structured and unstructured information.

Big data can be described with four V's characterized as (Simon 2013), (Andersen & Bakkeli 2015):

1. Volume; The quantity of generated and stored data. The size of the data determines the value and potential insight, and whether it can be considered big data or not.

2. Variety; The type and nature of the data. Big data draws from files, databases, text, images, audio, video and it may complete missing pieces through data fusion. Data fusion is a process for combining multiple data sources to produce more useful, consistent and accurate information than that provided by any individual data source (Hall & Llinas 2001), (Hall & McMullen 2004).

3. Velocity; Two kinds of velocity related to Big Data are the frequency of generation and the frequency of handling, recording, and publishing. Big data are producing information more continually compared to small data.

4. Veracity; An extended definition for big data, and refers to the data quality and the data value. The data quality of captured data can vary greatly, affecting the accurate analysis.

The knowledge is based on the analysis of the information, but you must first define what kind of knowledge you are looking for. It is not possible to analyses the information without knowing what you are looking for or what your goals from the information are. By using big data you should have some judgement about which part of the information that are important to you, and what you are looking for in the information, and tools and knowledge to do the analysis.

Compression

Data compression is the process of reducing the amount of data needed for the storage or transmission of information, typically by the use of different encoding techniques. Compression can be important for digitalization to reduce the amount of information stored on the data servers. An example is the zip file type reducing the size of ordinary text files.

7.2.6 Backup

Data backup is the process of duplicating data to allow retrieval of the duplicate set after a data loss event. Digitalization will increase the amount of data and backup will ensure that data is more secure and that critical information is not lost in case of a disk failure, natural disaster, theft situation or other kinds of emergency. The backup data should be stored on a separate medium and stored in another physical location.

Chapter 8

IoT and IIoT

Internet of Things (IoT) and Industrial Internet of Things (IIoT) are sensor and/or actuator devices connected to an intranet or Internet, ether wired or wireless. IoT and IIoT devices consist of a small computer system containing one or several sensor and/or actuator devices, an input/output module including a measurement system, and a module for Internet communication, often by wireless communication. These devices will also depend on an IoT protocol, an application programming interface (API) or a cloud service for communication to access the information collected by the IoT devices. An overview of an IoT device is shown in Figure 8.1.

Using intranet in industrial applications will require industrial Internet solutions, named as industrial Internet. An industrial Internet type consists of more robust devices, both hardware and protocols names as IIoT devices. An IIoT device may have a wider temperature working range, more robust hardware, shielding, mechanical packing, data security focus and/or more robust communication protocols or API than IoT devices. The cloud service will contain both a data collection module and a service oriented architecture (SOA) where SOA can include several analysis tools for handling and/or presenting the collected data. SOA is an architecture where adding additional tools is an option. These tools can for instance be for preprocessing, data fusion and/or extended presentation of the information and can be selected from an internal base of tools, or external tools.

Figure 8.2 shows different IoT devices that can be used at home, in front a weight[1], in the power socket in upper right corner a temperature controller for electrical panel heaters[2], at front to the left an indoor module for a home weather system[3], and behind the weather station two power switches[4] for turning on and off electrical power devices. The power switch is using a low power wireless network, while the other devices are using the Wi-Fi LAN at home to communicate with a cloud service, and you can get the information from a web server on Internet using a web browser. The weight can be used to record any variations in your weight and keep track of the variations depending on your activities and seasons. The power switch system has a gateway connecting the low power wireless network to the Wi-Fi LAN and a cloud service can be used to turn on and off electrical devices based on specific events like time, sunset or sunrise. The temperature controller is using a temperature sensor in the IoT device

[1] see www.withings.com (OCT-18).
[2] see www.millheat.com (OCT-18).
[3] see www.netatmo.com (SEP-18).
[4] see www.nexa.com (SEP-18).

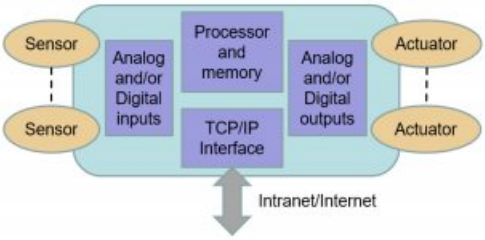


Figure 8.1: An overview of an IoT device, zero, one or several both sensor and actuator devices, an I/O part handling both analog and digital information, processor, memory and a TCP/IP interface for connection to intranet or Internet. An IoT dvice will have at least one sensor or actuator connected.

Figure 8.2: The image shows an IoT device in the background controlling the tempearture of the electrical heather using the WiFi network, in front the weight for monitoring and logging the weight for one or several persons using the WiFi network, to the left an indoor weather station module measuring the temperature, sound level, CO_2 and pressure, and back on the sift side two actuators turning electrical devices on and off.

and a cloud based temperature controller to adjust the electrical heater.

The home weather system gives information about the indoor environment by measuring the temperature, humidity, CO_2, noise level, and barometric pressure. Other IoT devices can be added to the system to measure outdoor temperature, humidity, amount of rain, wind speed and wind direction. An example of the information for a specific day is shown in Figure 8.3.

These devices are connected to the wireless network at home, an intranet, connected to Internet and a cloud service. The cloud service will use Internet and your intranet for collecting the information from these devices, store all the information in cloud server, and you must use an application (app) on your mobile phone or a web browser on your computer to access the information. The information will always be historical data, you will not be able to have "real-time" values in these systems as there will always be some delay in collecting, storing and analyzing the information. The figure also shows that the cloud service includes an extra service (SOA) to estimate the indoor comfort based on the sensor readings, and presented as a colored bar graph.

IoT and IIoT devices are part of the big data technology in collecting information, and an important part of digitalization. IoT devices will collect information from the environment, and/or control the environment based on the information. The information collected by these devices will be sent using the intranet, often wireless connection, via the gateway on to Internet to a specific cloud service hosted and controlled by the manufacture of the IoT device. The information will be stored by the cloud service, and the cloud service will present the information for the owner using a web client or an app on the mobile phone. The cloud service may also contain a controller estimating the control signal for the IoT device if used as a controller. Figure 8.4 shows the interconnections of these modules.

The information available on the web client or on the app on the mobile phone will be historical values. Many of the cloud services will also offer an API for other application to collect "real-time" information from the cloud service for use in other systems. One way of showing the relationship between IoT, IIoT, M2M (machine to machine) and Industry 4.0 is shown in Figure 8.5, with references to type of items.

8.1 IoT protocols

The communication protocol between the IoT device and any cloud service will be a specific IoT protocol, either proprietary protocol or an open protocol like OPC UA, Modbus TCP, XMPP (Extensible Messaging and Presence Protocol), MQTT (Message Queuing Telemetry Transport) or CoAP (The Constrained Application Protocol) to mention some. The challenge is the TCP/IP protocol as this protocol only defines a header and a data field like:

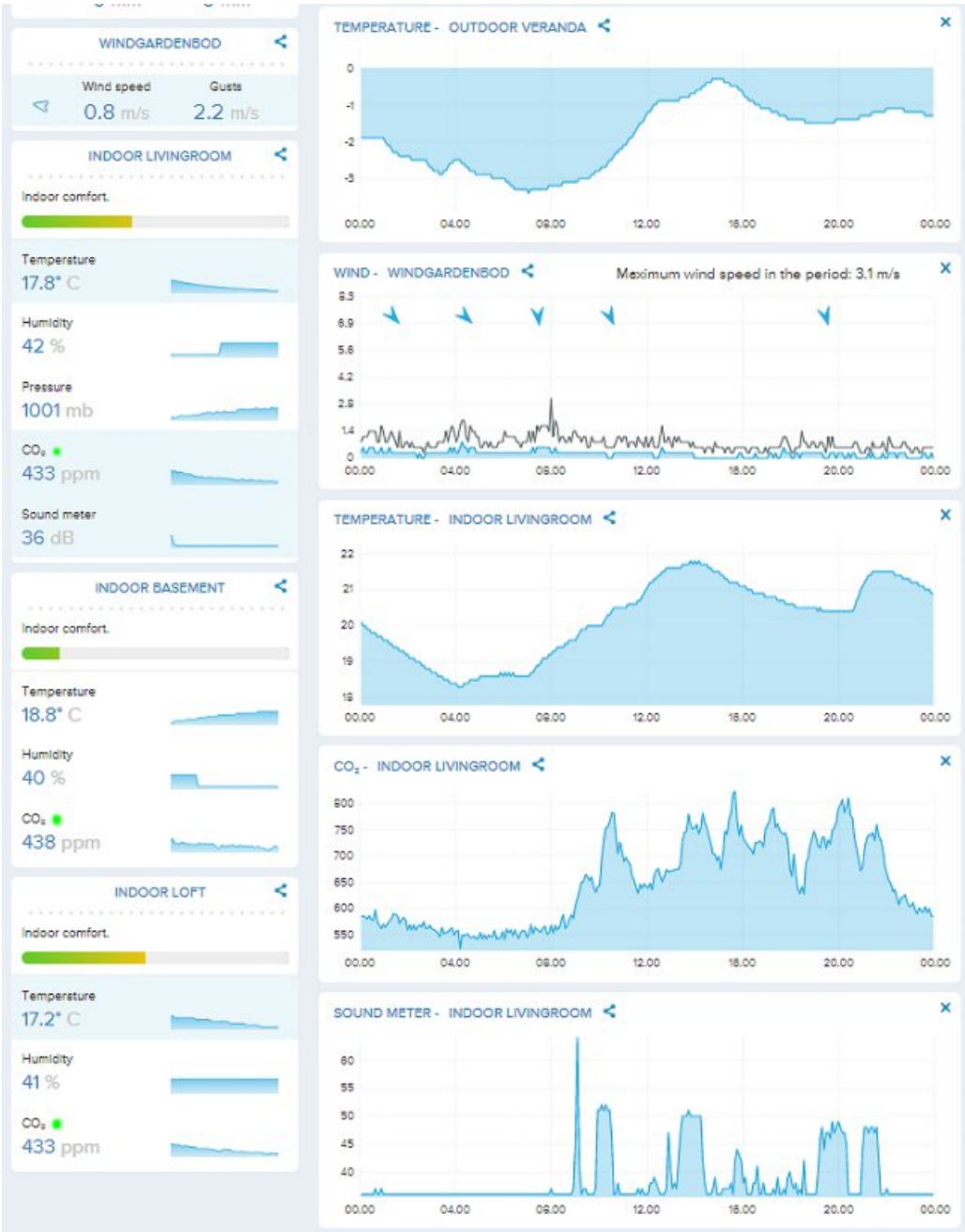


Figure 8.3: The information from the outdoor, wind and indoor IoT devices, showing the outdoor temperature, wind speed and direction, and indoor temperature, CO_2 and sound meter for a specific day. Left side of the figure includes an extra service (SOA) indicating the indoor comfort.

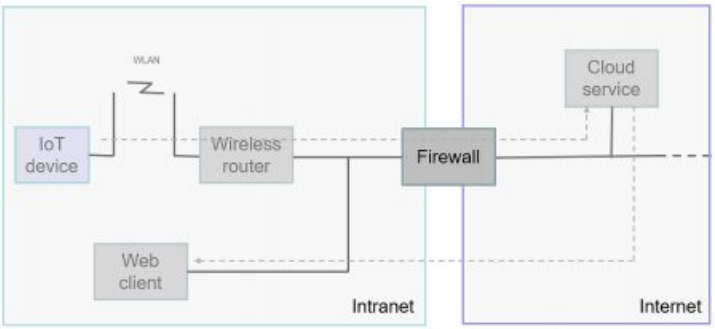


Figure 8.4: The interconnections of modules using an IoT device with the collected information transmitted from the IoT device via the wireless router and gateway to the cloud service. The cloud service will save the information, and estimate any controller signals to be used by the IoT for adjusting the environment. The user can access the information by a web client or an app on a mobile phone.

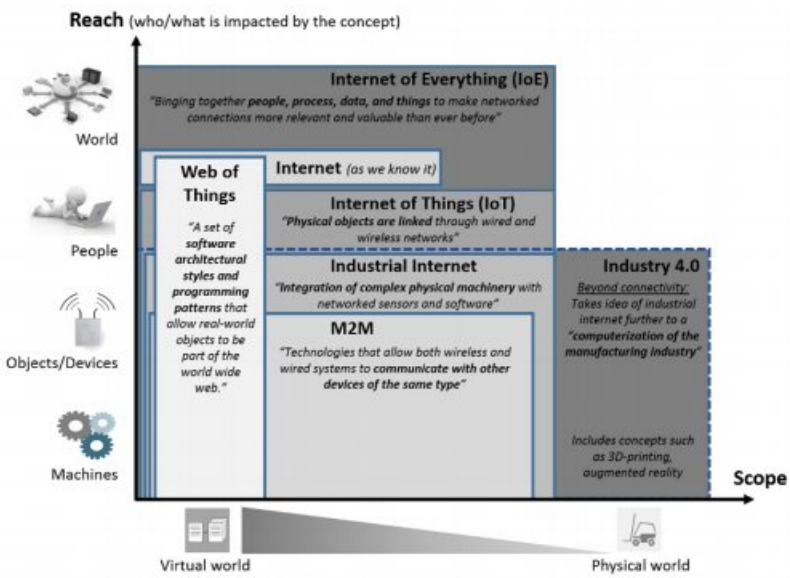


Figure 8.5: One way of showing the relationship between IoT, IIoT, M2M and Industry 4.0 (Iot-Analytics.com: Oct-20).

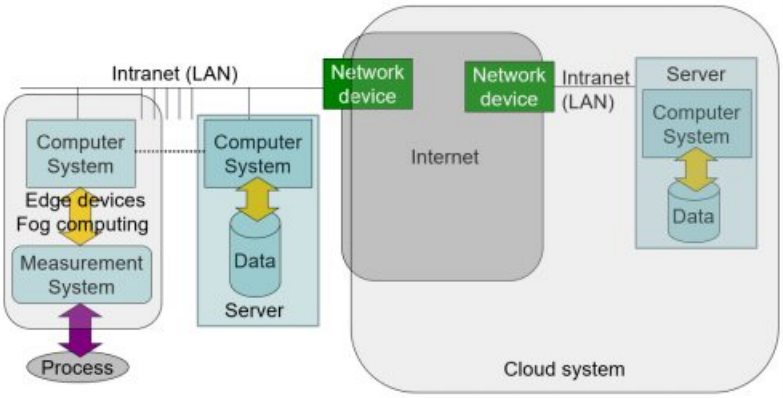


Figure 8.6: Edge devices and fog computing to the left in the figure, for communicating with the environment. The edge devices are conected to the LAN, interconnected to the cloud system.

Header	Data

where all the bits in the header field are defined while the data field are not defined. Any data can be included in the data field, but the receiver needs to have the knowledge of the strucure of this data. One way to solve this challenge is to use an IoT protocol is needed on top of the TCP/IP protocol to define the contents of the data field like:

Layer	OSI model reference
IoT protocol	Layer 7 - application
TCP/IP	Layer 4 and 5 - network and transport

The analog to digital (A/D) conversion is included in any IoT device and information exchange using the TCP/IP and IoT protocols is of digital format only.

8.2 Edge devices and fog computing

The usage of IoT and IIoT devices will include local computing as they must include a computer system to be able to perform the data handling and communication on Internet, as shown in Figure 8.1. A remote computer in the network can collect the information from several IoT or IIoT devices, do some local processes like compression or data fusion to minimize the information sent over the network and storage in the global data system. These remote computers located at the edge of the network are named edge devices and the activity performed is called fog computing. Fog computing can also be done directly on an IoT device, so a powerful IoT device can be used instead of the remote computer. Note that these concepts, edge devices and fog computing, are sometimes mixed into the concept *edge computing*.

Using Figure 6.7 as a reference, Figure 8.6 shows the edge devices, fog computing and the cloud system. A local computer system, as shown to the left in the figure, will collect information from the environment, do local processing of the information, and send the computed information to the local server and/or to the cloud system.

Chapter 9

Process control systems

Production companies have both an administration system and an automation system where the automation system is controlling the production processes. The administration system communicates with the automation system for information about the process and production, as shown in Figure 9.1.

Process control systems (PCS) or industrial control systems (ICS) are a collective term for automation types like factory automation and process automation. A PCS consists of different types of remote terminal units (RTU) for interaction with the industrial process, a supervisory control and data acquisition (SCADA) module or system as the heart of the PCS, user interface and a database, as shown in Figure 9.2. The administration system consists of several modules like the enterprise resource system (ERP) and manufacturing execution system (MES) systems for communication with the PCS, shown on the top of the figure.

There will be digitalization on all levels in such a system, however the ERP system will be the main module for digitalization (Columbus 2019) as this module will contain information about the business model, working processes and PCS. A process system is designed to produce some sort of materials or products, hopefully with a profit. Design limits exist in such system giving a limit for the maximum production. One goal of the operation of such process systems is to exploit the system as much as possible, but the operation target is often far below the design limits, as shown in 9.3. Digitalization can help in optimizing the production and improving the target for operation, based on the information mainly in the ERP system. The ERP system can better exploit and combine the information from the different modules and sub systems to increase the profit by moving the operating target closer to the design limits of the systems.

Digitalization will be to combine and exploit the information already available in the sub systems better, but also to evaluate for more information to be included. The usage of IoT devices and cloud services in digitalization will require more IT knowledge in combination with the operation of the PCS. With the usage of IoT devices, edge computing and cloud systems/services, a mixing of the terms operational technology (OT) and information technology (IT) has become more common. The use of IT at the operational level of a process control system (PCS) has become more common removing the borders between the OT and IT layers. The OT and IT layers are indicated in Figure 4.2, showing that the IT layer is merging slowly into the OT layer. Today process technologists should have skills combining both of these areas. The OT and IT areas have the same goals but different ways of combining them. The OT business area will priorities 1) avoid downtime, 2) maintain data integrity and data accessibility, and 3) data security, while the IT business area will swap the priorities as 1) data security, 2) maintain data integrity and data accessibility, and 3) avoid downtime of the system. Figure 4.2 also indicated the layers that will be affected by data security knowledge. Figure 9.4 shows how OT and IT can be combined for better digitalization in a PCS.

Figure 9.1: The interconnections between the administration system, automation system and the production process.

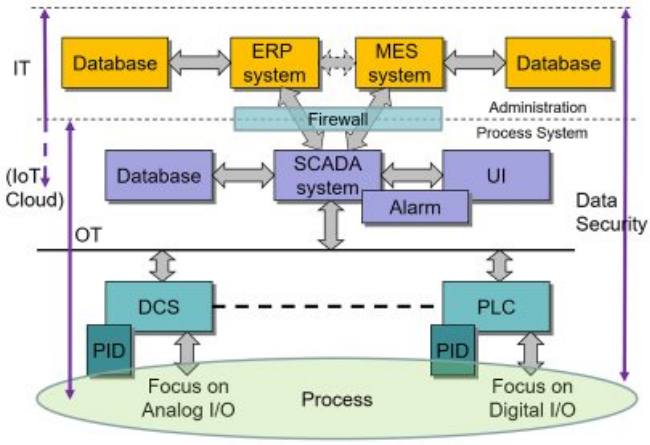


Figure 9.2: The building blocks of a process control system (PCS) and a connected administration system. The heart of the PCS is the SCADA software module (or system) collecting the information from the process, presenting, storing, and estimating new settings. Some of the information should/must be shared with the administration system.

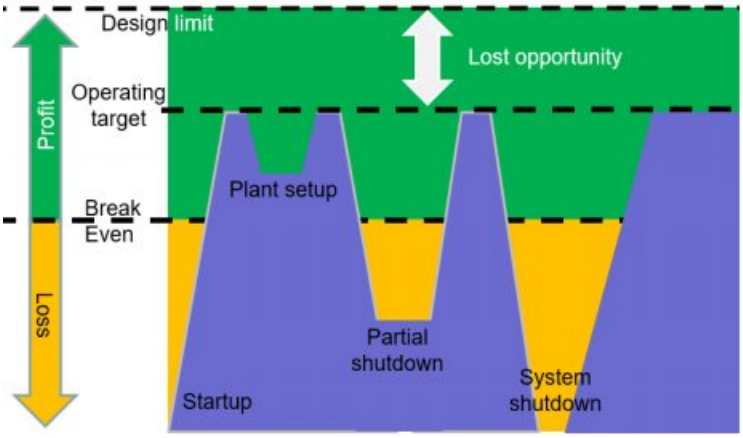


Figure 9.3: The operating target of a process is between the break even and the design limits of that process. Digitalization can be a way of increasing the profit by moving the operating target upwards (Isaksson et al. 2018).

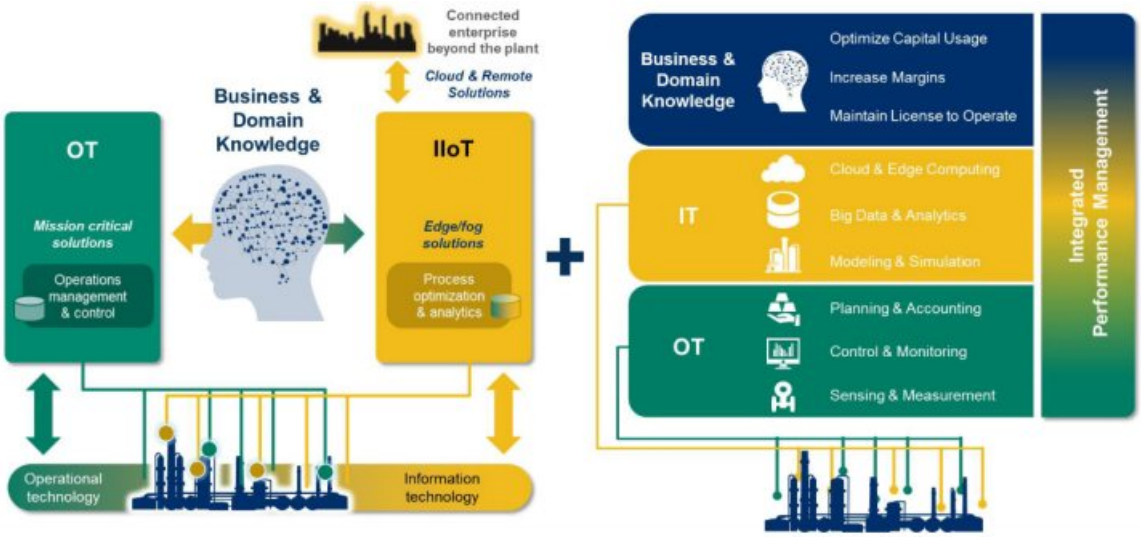


Figure 9.4: The combination of operational technology (OT) and information technology (IT) in process control systems (PCS) (Yokogawa 2018).

Chapter 10

Industry 4.0

Digitalization is a key feature of Industry 4.0 as industry 4.0 is the path to get more information about the working processes, the production and the market. Industry 4.0 is said to be the fourth industrial revolution and it combines automation and informatics and communication technology (ICT). The first industrial revolution was about using mechanical automation, the second industrial revolution was about implementing more streamlined working process, like production lines, the third industrial revolution was about electronic automation, and this fourth industrial revolution is about combining the electronic automation with information and communication technology (ICT) on both the production and working processes. An illustration is shown in Figure 10.1.

Digitalization is the method for combining information using automation and ICT. Electronic automation is about controlling the processes by using sensors and actuators, sensors for getting information from the process, and actuators for controlling and influencing the process by Proportional Integral Derivative (PID) controller and/or Model Predictive Controllers (MPC) amongst others. MPC will also include an optimizer and a set of boundary conditions to control the process. Information from the PCS and automation systems is distributed using the network and shared and/or reused with the information from the working processes for Industry 4.0 as shown in Figure 10.2.

The fourth industrial revolution can be summed up by digitizing a company's products and services, and taking technological integration into the value chain, both horizontally and vertically. The aim is to 1) satisfy customers' expectations, needs, wishes and requirements, 2) increase cost efficiency, productivity and efficiency, 3) increase the quality in all stages of the value chain and 4) reduce the error rate. And may be find new profitable business models and/or create new customer platforms.

A production company will have both a horizontal value chain and a vertical value chain. A production system and automation system is part of the vertical value chain and there is a wish that this value chain should increase the capacity. The production data is today often located only as part of the PCS and not shared with the rest of the systems. By sharing the information (availability) with the ERP system it may be better to optimize a system, or combining several systems for optimization. Systems today are based very much on silo of information, and not sharing this information. The information from the vertical value chain should be more available to the ERP system, and digitalization is an important process here.

The horizontal value chain is dealing with the planning of the production, allocating the resources and the feedback from the market and the requirements of the users. A faster feedback can be solved by using IoT (and/or IIoT) devices getting a more instant feedback. This feedback can only be utilized by digitalization, by storing the information from the IoT and IIoT devices in data servers and using analyzing tools checking this information on a regular basis. Based on the results from the analyzing tools, the resources and planning of the production can be adjusted to the current need in the market. This faster feedback will also require a more flexible production process for faster adoption to the market changes. A more flexible production process should also give a faster changeover and lower cost of these changes. Figure 10.3 shows the horizontal value chain with feedback using IoT devices, and a vertical value chain for one production line.

Example of digitalization of the horizontal value chain can be electrical cars like Tesla and the measurement of electricity using AMS. A Tesla includes wireless updates that will improve the control system of the car over time, remote diagnostics and support from mobile service technicians that will reduce the need to visit a service center. In cases when the car requires a visit at the workshop, the service should be fast as the workshop should have all the information and parts available before the visit.

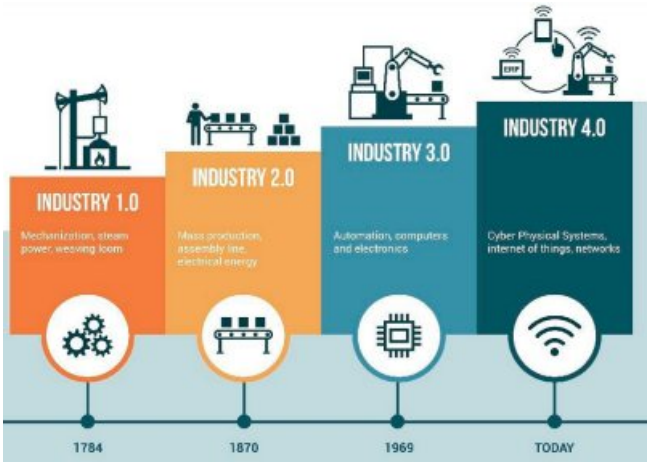


Figure 10.1: The industrial revolutions where Industry 4.0 is said to be the fourth revolution. The fourth revolution combines computers, automation, networks and cyber-physical systems (www.cadm.com: JAN-19).

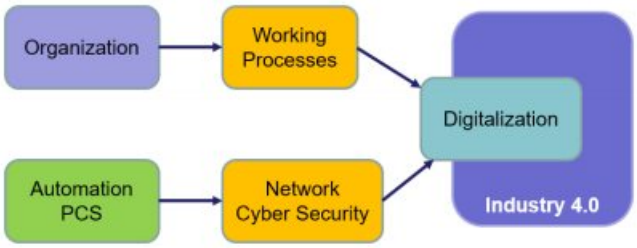


Figure 10.2: The information combines, shared and/or reused in Industry 4.0.

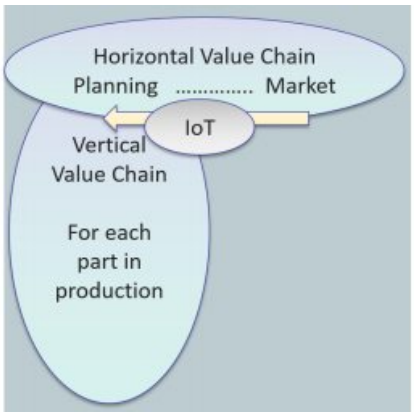


Figure 10.3: Digitalization of both the working process (horizontal value chain) and the production process (vertical value chain).

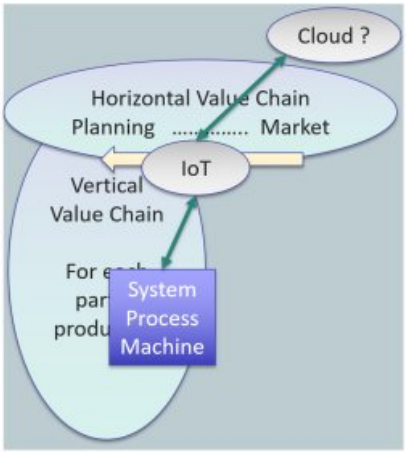


Figure 10.4: The production processes (vertical value chain) contain automation systems for the devices, and the information can be collected by the PCS, including IoT and IIoT devices. A cloud service can be used for collecting and storing the information from these systems, including the IoT and IIoT devices.

AMS will automatically monitor the usage of electricity, bill you for the correct amount of electricity that you have used the last month, and inform or warn you of any failure in your electricity system.

The automation system have the information for the vertical value chain. This vertical chain consists of processes, systems and/or machines, and the automation system will have current values for these items. Using digitalization these values can be stored in a data server and used for control, optimization and/or simulation. Figure 10.4 shows the automation system using IoT (or IIoT) devices collecting the information, and a cloud service for retrieving and storing this information. The IoT devices can be connected directly to the automation system without any cloud service. Actions like testing, adjusting, optimizing, and/or simulation of the devices in the vertical value chain are executed. Services like predictive maintenance can also exploit the information in making the production more robust.

These actions will require a model of the system, and industry 4.0 uses the concepts *cyber-physical systems(CPS)* and *digital twins* for describing the use of these models. CPS are the integration of computer based algorithms, networking and physical processes. The computer based algorithms, as part of the model, is used for monitoring and controlling some sort of item or device, often with mechanical parts, and integrated with the Internet and users. The Internet is used for collecting information used for the model. These models can then be used for checking and verifying any operation of device in "real-time". A digital twin is more a virtual description of the system and how it can be maintained during the lifetime. The digital twin may also include models of the items and devices, like 3-D models, mechanistic models, data driven models and CPS to mention some. These digital twins can be models combined into machines, processes and production lines, and digitalization is required to have data for developing the models and "real-time" data for using the models. Figure 10.5 shows the digital twins using information from the IoT devices and cloud services (or data servers).

Digitalization involves information from the production of the devices or items, and historical data, all information in a format that can be reused by other applications. The models will be developed by using this information and must be calibrated. After developing the model, the model will use the current information for optimization, simulation and/or prediction. The information must be combined from production data, historical data and current data, meaning that this information must be combined from many different sources. Some of these sources can be IoT (or IIoT) devices and these devices will involve cloud services as many of these devices will require an API for collecting the information. The amount of IoT devices and other type of information will result in big data, huge amounts of data from different type of data servers or data sources, both structured and unstructured information. The usage of cloud services may include SOA to get preprocessed and/or analyzed information instead of just raw data. The SOA can be different type of tools that can be included with the cloud services or that can be used at an extra cost. Some of the IoT devices can collect information from the market or users giving the system an option for faster response to changed customer requirements as well.

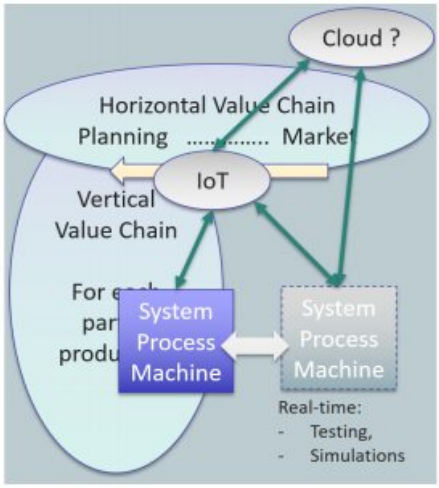


Figure 10.5: The use of digital twins in making a system for real-time simulation of a process. The digital twins can also be used in control systems for optimizing.

Figure 10.6: The QR code for "www.usn.no".

10.1 Augmented reality

Augmented Reality (AR) is often linked to gaming and the use of AR glasses. However, AR can be very useful in a production system for service and documentation. AR strongly depends on digitalization as all the information regarding devices and systems must be available in a shareable digital form. The devices or systems are marked either using a quick response code (QR code) or a radio-frequency identification (RFID) and using a smart phone or a tablet computer to read the QR code or RFID can give you all the wanted information about the device or system when located close to a device or a system.

QR codes can easily be created, printed and glued to any system. The QR code for www.usn.no is shown in Figure 10.6.

RFID is a small electronic device, consisting of a computer device, transmitting a specific code on request that can be used to identify any device or system in the same way as the QR code. The QR code requires an optical reader while the RFID requires a radio transmitter and receiver to be able to read the code or identification. A production system often has a lot of tags (ISA 1992), a code describing an input or an output device or system in the production system, and this tag information can be linked to the QR code or RFID.

AR can improve the quality of the production system as it will provide all the information available at the location, both information for configuration, historical and current values, and updated documentation. The system can also improve knowledge transfer as extra information from experienced personnel can be stored dealing with special errors and/or conditions for a specific tag. The system may also reduce the service cost and improve the service or maintenance working processes as all the information should always be available for tags in the production system. AR systems may be tablet computers or smart phones.

10.2 Artificial intelligence

Artificial intelligence (AI) is the simulation of human intelligence processes by machines, especially computer systems. These processes include learning, reasoning and self-correction. Learning is the

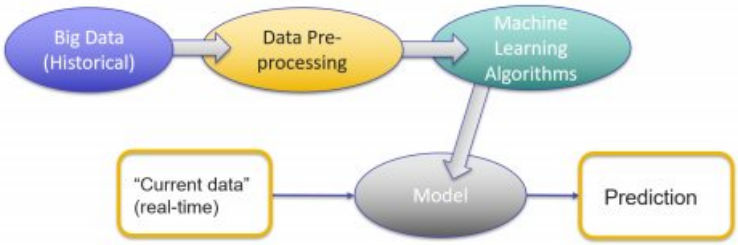


Figure 10.7: The machine learning approach where a data driven model is developed based on historical data and used for prediction.

acquisition of information and rules for using this information. Reasoning is the use of rules to reach approximate or definite conclusions. Both learning and reasoning require information, and digitalization is used for both collecting and using this information.

Artificial Intelligence (AI) and the Internet of Things (IoT) are in a process of merging into what is define as AIoT. With edge computing, computational power is moving to the edge where IoT devices are gathering data. AI is the next logical step in efficient data handling and lowering latency while opening up for innovative solutions at the edge.

10.3 Machine learning

Machine learning is an application of artificial intelligence (AI) that provides the systems with the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning (ML) requires data, often a huge amount of data, and digitalization is a way of getting the data that is needed for machine learning methods. ML methods are used for developing data driven models, instead of programming, where the models are based on historical data, and the models are used for prediction based on current information. The knowledge is included in the model, from the ML method and the historical information. Figure 10.7 shows the ML approach where a ML algorithm is used for developing a data driven model. The model is developed based on a large amount of historical data covering the whole range of variation in the system, and it is used for prediction. It is important to note that the historical data must contain enough information about all the variation in the system as the system can only deal with knowledge learned by information available in the historical data sets.

The data preprocessing part is important for the ML algorithms and different analysis tools from pattern recognition and classification are used like the Principal Component Analysis (PCA) (Næs, Isaksson, Rearn & Davis 2004), (Esbensen 2002). The structure of the data pre-processing is shown in Figure 7.1. Machine learning algorithms can be principal component regression (PCR), partial least squares regression (PLSR), artificial neural network (ANN), deep learning, support vector machines (SVM), genetic algorithm (GA), decision trees, nearest neighbor methods and other probabilistic methods. (Marsland 2015), (Tufféry 2011), (Kuhn & Johnson 2016). Machine learning algorithms require three data sets, one for training the model, another one for calibrating the hyper parameters for the model and a third one for validating the model. Overfitting will always be a challenge when dealing with data driven models meaning that the model will be fitted too much to the data sets used for training and calibration, and can make poor prediction for new data sets (Kuhn & Johnson 2016):

Machine learning algorithms are often categorized as either supervised learning, unsupervised learning or reinforcement learning (Marsland 2015). The differences are:

- Supervised machine learning algorithms use data sets with the responses and an algorithm generalizes to respond correctly to all possible inputs.
- Unsupervised machine learning algorithms use data sets without the correct responses, the algorithm tries to find similarities between the inputs. The algorithm will try to categorize inputs together that has something in common.
- Reinforcement machine learning algorithms are learning methods in between the supervised and unsupervised approaches. The algorithm is told when the answer is wrong, but not how to correct the answer. The algorithm must try out different possibilities until the answer is not wrong.

Chapter 11

Autonomous systems

Autonomous systems are systems which can perform desired tasks in unstructured environments without continuous human guidance. Any system will need some sort of actions for performing the desired tasks. A manual system will require these actions from humans, as shown in Figure 11.1.

The third industrial revolution is based on electronic automation. In electronic automation information is collected from sensor devices (digitalization) and the information is used by a controller for doing the actions needed for the system performing the desired tasks. However, the automation system needs set points for defining the desired states of the system. These automation systems, based on a type of controller (P, PI, PID or analogous) will have the set points set by a human, but will perform the desired tasks within the set points ranges. A controller based system is shown in Figure 11.2.

An extended automation system is an automation system based on a Model Predictive Controller (MPC) where the controller is combined with a mathematical model of the system including a set of boundaries for performing the desired tasks. The controller will use the output from the system, collected by sensor devices or estimators as input to the model for prediction of the actions, within the set of boundaries. Using the model, the automation will have included some knowledge.

An autonomous system may use the MPC approach with a model and a set of boundaries from the automation system, but requires more rules and a module for decisions based on the inputs and rules. As the system should perform the desired tasks without continuous human guidance, it must have some knowledge from similar situations. This knowledge can be in form of data driven models developed by machine learning algorithms on similar situations. Autonomous systems will therefore include some sort of AI system for handling the learning and reasoning part of the system. This is shown in Figure 11.3. The figure also shows how the human can guide the system in new areas or situations, and that the model can be updated with the new information. In autonomous system digitalization is required, both for training and guiding the system, and for unguided operations.

An autonomous system is based on a set of building blocks including perception from sensor devices and rules, security aspects for both the system, environment and humans. The predicted decision can be optimized, a good optimization module should be an important module in such systems. Figure 11.4 shows the building blocks of an autonomous system, with optimization. Optimization will also depend on information so digitalization will be important here as well.

Figure 11.1: The opertation of a manual system, all action performed by humans.

Figure 11.2: Electronic automation of a system using a controller. The controller optimizes the performance of the system according to the setpoints, set by humans.

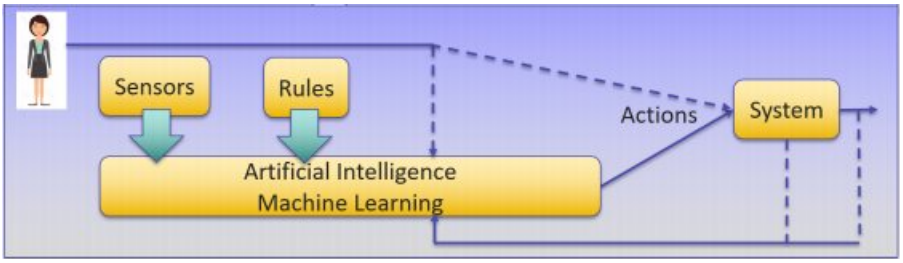


Figure 11.3: The structure of an autonomous system where sensors and rules together with a data driven model define the desired action for the system.

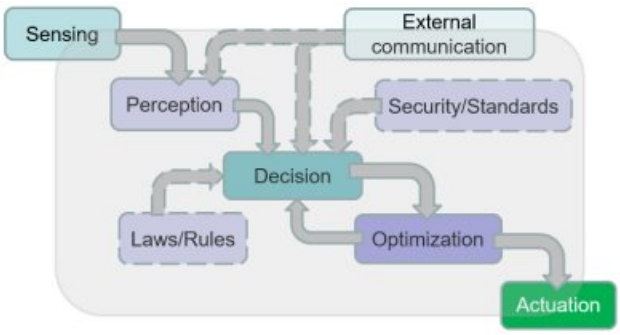


Figure 11.4: The main building blocks of autonomous systems required to make the optimum decision.

Chapter 12

Data security

Digitalization is collecting, storing and reusing information. Part of this information can be private and should be protected in some way. Historical data stored in a computer should also be protected as the data can be destroyed or altered. Data security is more then just preventing unauthorized access to computer systems and databases. Data security will affect anything that can destroy your computer systems, either the computer system itself or information related to the computer system. This includes fire, explosion, power loss, theft, user error and cyber attack to mention the most normal reasons. So security threats affect both the location where the computer system is used, the environment of this location, the connection to external devices and networks, and the computer system itself. Figure 12.1 shows many of the threats that can affect a computer system, and the information available in such systems.

Note that external devices can also be USB disks (memory disks), not only connections to other computer systems. We often think of data security as mainly preventing the hackers from accessing the network, however data security involves 1) network errors/downtime, 2) loss of data and handling backups, 3) misuse of data, 4) operations inside the company by employees, and 5) operations outside the company by hackers. Backup will always be important when dealing with data information.

Start with an overview where the information is stored in your system; local computers and servers in your intranet, and data stored in the cloud. Second, figure out who has access to the data, type of access, how they are using the data, and what they are using the data for. You will need a good understanding of the business models to understand how to protect this information. Also remember that your company is responsible even if the information is stored ina cloud system. The next step is to make logs on access and/or use of the information. These logs are very useful after an attach to understand when it happend, how it happend and what type of information was involved. A good advise is to train on attacks to check that it can be seen in the logs, and learn how to protect the information even better. This means that a company must invest money and time in advance to prevent an attach, and to learn from an attach. If your server is attached, and you restore your server, will the server be atttached again. So you need to learn from the attach and make your system better. More restrict acces rights? Separate the different systems into segments with more restrict acces rights? And you need a

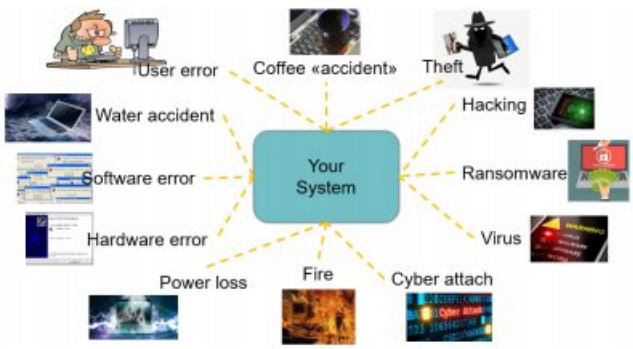


Figure 12.1: Many of the security threats that may affect a computer system, and the information in such systems. These threats varies from physical threats, user threats, to cyber threats. However, cyber threats are only a small part of possible threats.

Figure 12.2: Any computer systems in a LAN is important for data security, even the computer system for the boss. The bosses often have access to most of information in a computer system.

system that are checking the logs regulary to discover an attach as early as possible. In EU, with the new GDPR, companies must report within 72 hours about all data burglaries. The use of digitalization gives companies more possibilites both for saving information and logging the use of this information.

One way of attaching the information is the use of ransomware, a software that will encrypt your storage device and make the information unavailable to the company. A ransomware can be in an attachment in an email to one of the employers in the company. Another way of attach is the use of (DoS) where a computer, or a network of computers send an huge amount of illegal network messages to a computer to overload that computer, preventing form being unable to perform any other tasks. First a lot of computers are attached to install the DoS software (attachment to emails etc.) and then all these computers can be triggered to start the DoS at the same time. In an industrial control system or process control system the information technology (IT) systems often consists of updated systems (thee to four year cycles) while the operational technology (OT) systems often consists of outdated systems (ten to twenty year cycles). This means that the IT infrastructure will generally be better protected and equipments still in operation in the OT part were not originally designed for external connectivity making them a softer target for cyber-attacks. The IEC 62443 standard has a set of guidelines that covers technical specifications as well as the maturity levels and processes that are required within the OT domain, such as passwords and how to control them

Access to the computer system can either be a physical login at your computer system, or a remote login using the network. Remember that a password is meant to protect your computer system from strangers, not to make it as difficult as possible for you to access your own computer system. The password is part of the login process for getting access to the computer system. This login process is either an one stage process based only on the password, or a two stage process where the login process is only the first validation stage. The second stage is often sending a code to your mobile phone for input as the second validation part. If using only a one stage login process, use a proper password. Please do not select any of the most used passwords as in 2018 were: 1) 123456, 2) password, 3) 123456789, 4) 12345678, 5) 12345, 6) 111111, 7) 1234567, 8) sunshine, 9) qwerty, and 10) iloveyou. These will be the first options for anybody trying to access your computer system. Your password is an important part for the data security in the whole system (the intranet), as shown in Figure 12.2.

Chapter 13

Digital21

The Norwegian government established "Digital21" in 2017 to speed up the digitalization process. Digital21 shall promote the business community's ability and opportunity to develop and implement new technology and knowledge in step with the increasing digitization. A conclusion from "Digital21", late in 2018, by input from more than 600 participants, is that Norway, as a country, needs a strategic investment in four key technology areas. These areas are 1) artificial intelligence (AI) , 2) big data, 3) the Internet of things (IoT) and 4) autonomous systems. The proposal is a strengthened, targeted and holistic initiative within education and competence enhancement, research and instruments for innovation.

Part VI

Data Communication

Chapter 14

Introduction

This part will focus on the communication between any "sub" systems in an industrial IT system, meaning the communication between:

1. the sensors and the PLC, DCS, PAC, and/or SCADA, the input devices and modules,
2. the valves and the PLC, DCS, PAC, and/or SCADA, the output devices and modules,
3. the control room system(s) (SCADA system) and the I/O devices,
4. the systems in the control room(s) (process control systems and industrial control systems),
5. the system (process control systems or industrial control systems) and external system(s).

This part will focus on the communication between the sensors and the rest of the monitoring or control system, not the sensors and the actuators itself. The communication will be between different computer systems, but also within the devices included in each computer system. This communication can be between a single computer and I/O devices, or a large control systems consisting of several servers, separate local area networks, and remote terminal units as I/O devices.

14.1 Process network

The process network is the network for distributed devices like SCADA, PAC, DCS, and PLC in the process. These devices are computers designed and used for process control. These computers were connected in a private network for the process control, but lately the administration want to have information from the process as well. An example of a process network is shown in Figure 14.1.

14.2 Computer communication

A personal computer (PC) is using a lot of data communication, both internally and for external communication. Figure 14.2 shows an overview of some of the devices in a PC using both parallel and serial data communication.

A computer can be a personal computer for operators, but can also be a computer for monitoring and/or control of the process. The computer still needs internal and external communication for monitoring and/or controlling a process. All the I/O devices needs some sort of communication facilities and the communication will be either of parallel or serial form. The I/O devices for a computer will mainly be for storage, user interface, and network. The computer will use serial communication for:

1. RS-232C and RS-485 for communication with external sensors, measurement systems, control systems, and other computer systems,
2. network for communications with other computer systems,
3. SATA for communication with storage devices like disks, CD-ROMs, DVDs, and backup devices,
4. USB and FireWire for communication with external sensors, measurement systems, control systems, and other computer systems,

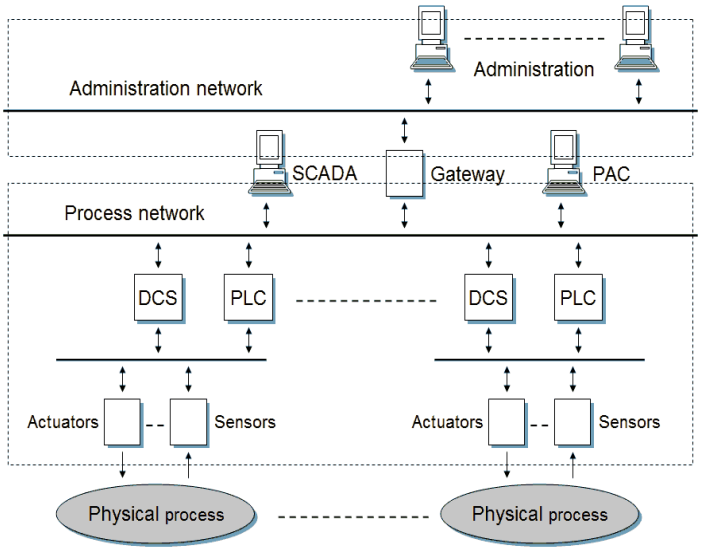


Figure 14.1: A process network with a SCADA system and a PAC system, and connected to the administration network.

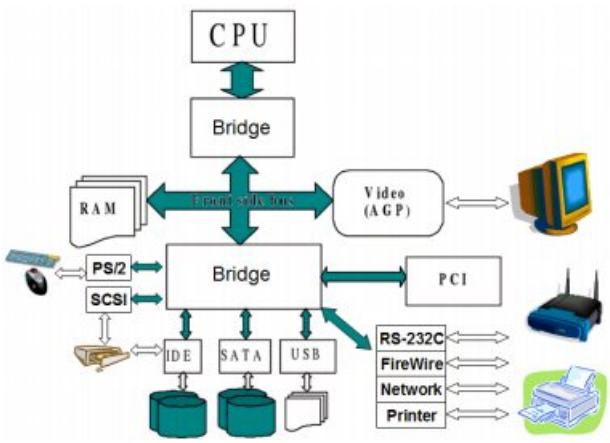


Figure 14.2: A computer with internal devices and I/O devices using data communication.

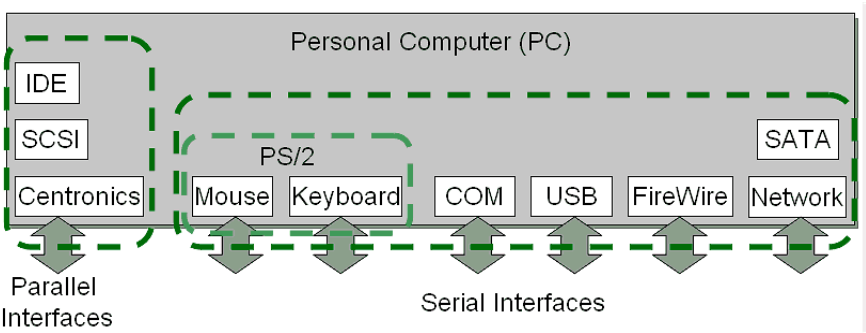


Figure 14.3: The main communication ports for a personal computer (PC).

5. PS/2 for communication with I/O devices like keyboard and mice.

The computer will use parallel communication as:

1. IDE (ATA or PATA and SCSI for communication with storage devices like disks, CD-ROMs, DVDs, and backup devices,

2. printer port for communication with printer etc.

Figure 14.3 shows the main communication ports for a personal computer, both internal and external communication ports. Remember that internal devices like disks and DVDs also need some sort of communication and protocols to communicate with the CPU(s) of the PC.

Chapter 15

OSI model

15.1 Introduction

Communication is vital for all type of information system, including industrial systems. A computer system is only a system taking some data as input, doing some computation(s) and output new data. A computer system must use some sort of communication to receive the input data from some other system, and some sort of communication for transmitting the new data.

Communication can either be with another computer system, or with humans (or both). Communication requires some sort of I/O[1] devices both for communication to other computer systems and with humans. I/O devices for humans are different types of UI (User Interface) devices like keyboards and graphical devices like screens, while the I/O devices for communication with other computer systems are according to different type of protocols. An overview of the I/O devices and the communication is shown in Figure 15.1.

A communication system device is often called a node. A node is defined as a computer system with communication capabilities.

15.2 Protocol

A protocol defines how a set of computers can communicate. An analogy can be two persons talking together, they have to talk the same language, they have to talk with almost the same speed, the distance between the persons have to be controlled, and if one of the persons are not understanding, the other person may respond in some way. See Figure 15.2.

A communication protocol is a set of rules for data representation (format, buffer sizes, encryption etc.), signalling, synchronization (procedures for transmitting and sending, time control, etc.), authentication and error detection (and correction?) required to send information over a communication channel. The protocol may be implemented in hardware or software, or both. The communication protocols for digital computer network communication have many features intended to ensure reliable interchange of data over an imperfect communication channel. Communication protocol is basically following certain rules so that the system works properly (www.wikipedia.org 2006).

[1] I/O: Input/Output

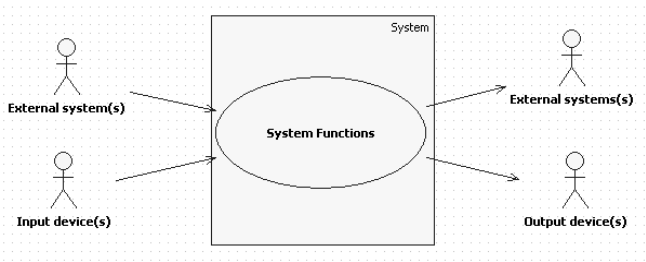


Figure 15.1: An use case model showing the I/O communication for a computer system.

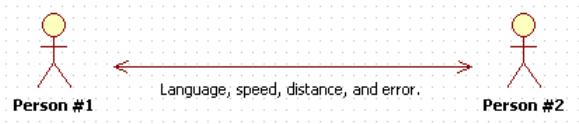


Figure 15.2: The protocol is important when communicating.

Protocols vary, from a simple description of the electrical interface of the medium and the data format of the medium, to huge specifications, depending on the services provided.

A protocol normally consists of a set of bytes, a frame format, that might be made up as follow:

1. The first byte(s) used for synchronization (of the frame),
2. The header, consisting of frame information like:
   (a) A byte with the destination address,
   (b) A byte with the source address,
   (c) Option bytes may include
      i. a command byte to indicate the type of message,
      ii. the length of the message (or frame),
3. the actual data,
4. end of frame indication, which may include some type of error detection.

The protocol can vary from simple ASCII protocols up to complex binary protocols used for high speed and transferring large amount of data per second. ASCII protocols are transmitting the ASCII conversion of the fields, while the binary protocols are transmitting the binary values of the fields. The value 124 will be transmitted as 1, 2, and 4 (3 characters or bytes) using a ASCII protocol and as the binary value 124 (hex=$0x7C$, only one byte) in a binary protocol.

ASCII protocols are normally used for serial line communications, while binary protocols are used for both serial line communications and network communications. Protocols are also used for internal communication between applications on the same computer, using middleware as a form of Application Programming Interface (API) for data exchange between applications.

The main types of protocols are:

1. Output protocol: The master node is just transmitting data in a cyclic or period way. The slave node(s) have to read and capture the data needed. There will be no acknowledge from the slaves. Often a single transmitter access protocol. The access type can be publisher and subscriber.
2. Output protocol with receive check: A node is transmitting data, but checking that it is receiving the same as transmitted. Multiple transmitter access protocol. The access type can be deterministic, stochastic, or token ring.
3. Master / slave protocol: In a master/slave protocol will one of the nodes be a master for the communication and the usage of the communication media. Normally will the usage of the communication media be divided into time slots. The master will decide who can use the communication media by sending a message to one of the slave nodes. This specific slave node will have access to the communication media for the next time slot and the master node will expect a response from the slave node. There will be either a response or a time-out. Then the next slave node will receive a message and can use the communication media for the next time slot. A server / client protocol however is a protocol with focus on the data, not on the communication media. This type of protocol is used a lot in industrial communication systems.
4. Token ring protocol: A master/slave protocol with several masters. The current master will be the owner of the token, and the current master will send the token to the next master when a set of operations are finished.

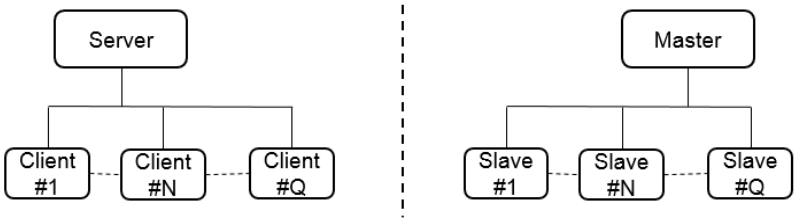


Figure 15.3: The server/client principle to the left and the master/slave principle to the right.

5. Multiple Access protocol: All the nodes will be masters and some sort of arbitration is needed to decide which master should use the communication resource. Only one master can use the communication resource at a specific time interval. The access type can be deterministic or stochastic. Peer to peer communication protocol is such a protocol where every node has the same rights to use the communication resource.

The master/slave principle is often mixed with the server/client principle. These two principles are not the same, but they have several similarities. The master/slave principle is normally used in the industry and communication is the focus. Often the slaves contains the information in the system (RTUs) and the master wants to collect the information from the slaves. The master be the owner of a resource and the slaves will have some sort of data or information that can be shared using the resource. Dealing with a master/slave principle it is important to agree on what the meaning of the master is, master of what (data, type of information, communication resource)?

The server/client principle is based where the data or information is in focus. The server will have the data, and the clients can send request to the server to get a copy of the data. The server/client principle is used with type of databases or web pages where the server will be the owner off the information, and the clients will request information from the server. The server/client principle is used most in IT systems where the clients can run of different nodes.

In the server/client principle will the information or data be in focus and the server will be the owner of the data. In the master/slave principle will the communication resource be in focus and the slaves will normally be the owner of the data. So the server in a server/client principle will be the slave in a master/slave principle, see Figure 15.3.

15.3 Directions

The direction of communication can be simplex, half duplex or full duplex. Simplex communication means that the communication direction will be only one way meaning that one device with either receiver data or transmit data. Half duplex means that the communication direction can be both ways, but only one way at a time. Any device can both receive data and transmit data, but not simultaneously. Full duplex means that the communication direction can be both ways at the same time. Any device can both receive data and transmit data simultaneously. A simplex and half duplex communication requires only one communication line (or channel), but need a type of master/slave protocol. A full duplex communication requires two communication lines or channels, meaning either physical or virtual connections. A full duplex connection can be emulated using time multiplexing on one physical connection or frequency modulated using two different frequency channels (and higher bandwidth).

Summary; the communication directions between computers can be:

1. simplex; communication only one way (one channel),
2. half-duplex; communication both ways, but only one way at a time (one channel),
3. duplex; communication both ways at the same time (two channels).

15.4 OSI communication protocol model

In the 1970 when the data communication started, the programming language was in assembler and the concern was about every bit, so a protocol was not that important. When programming in a higher level language, and especial using OOP, more data and higher speed, the OSI model is used for making

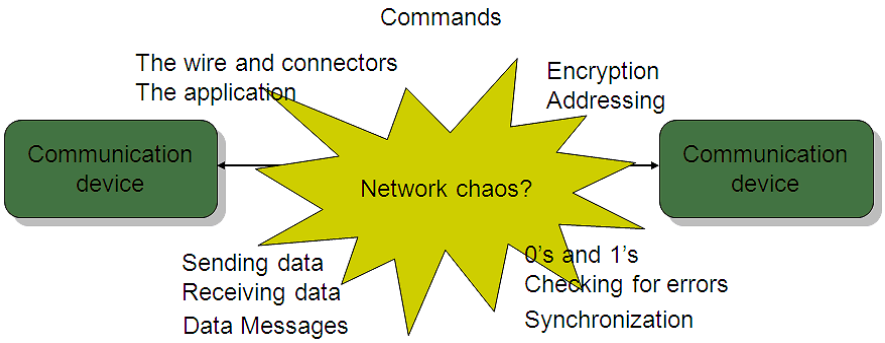


Figure 15.4: Functions and services needed for communication between communication devices (nodes).

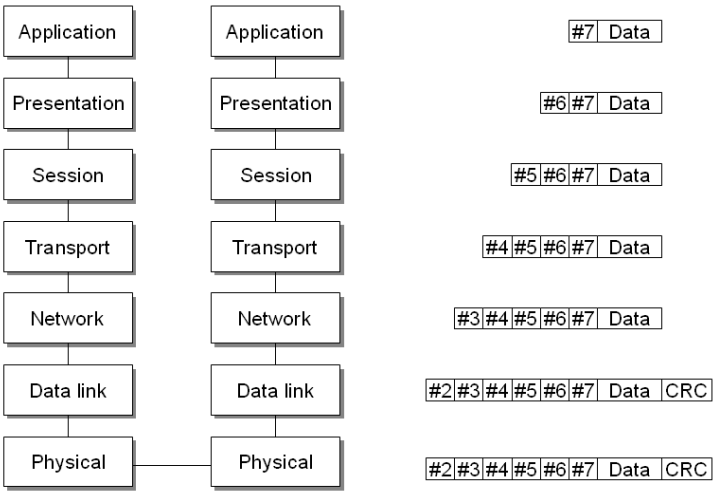


Figure 15.5: The OSI model for communication protocols (Olsson & Rosen 2003).

different abstraction layers for the communication. The network or communication between computer systems consist of a lot of functions and services as shown in Figure 15.4. These functions and services are structures in layers. The OSI communication model is dividing these functions and services into 7 layers, as shown in Figure 15.5.

A more general description of the services in a protocol, the OSI model is used. The Open Systems Interconnection Basic Reference Model (OSI reference model or OSI model for short) is a layered, abstract description for communications and computer network protocol design, developed as part of the Open Systems Interconnection initiative. It is also called the OSI seven layer model (Houldsworth 1990), (Levermore 2000). A OSI model is shown in Figure 15.5.

The OSI model is a reference tool for understanding data communications between any two networked systems. The model divides the communications processes into seven layers. The purpose of each layer is to offer certain services to the higher layers, shielding those layers from the details of how the offered services are actually implemented. The three lowest layers focus on passing traffic through the network to an end system. The top four layers come into play in the end system to complete the process.

The OSI model is a conceptual model defining a set of services in each layer. The protocol will define a set of the layers describing the services available in the protocol. Figure 15.5 also show the message, shown as data, and the additional data added from each layer. Each layer in the model can only communicate with the nearby layers and the same layer on the remote computer. For example layer #4 (transport layer) can only communicate with layer #3 and layer #5 on the same computer, and only layer #4 on the remote layer. This is also shown in Figure 15.6, showing the message passing through all the layers on one computer, over the physical medium and through all layers on the remote computer.

Each of layers in the model relies on the layers below it to provide supporting capabilities and performs support to the layers above it. Such a model of layered functionality is also called a *protocol stack* or *protocol suite*. A stack means a layered, abstract description for network protocol design. The protocols can do the work in either hardware or software, or normally in a combination of the two. The lower

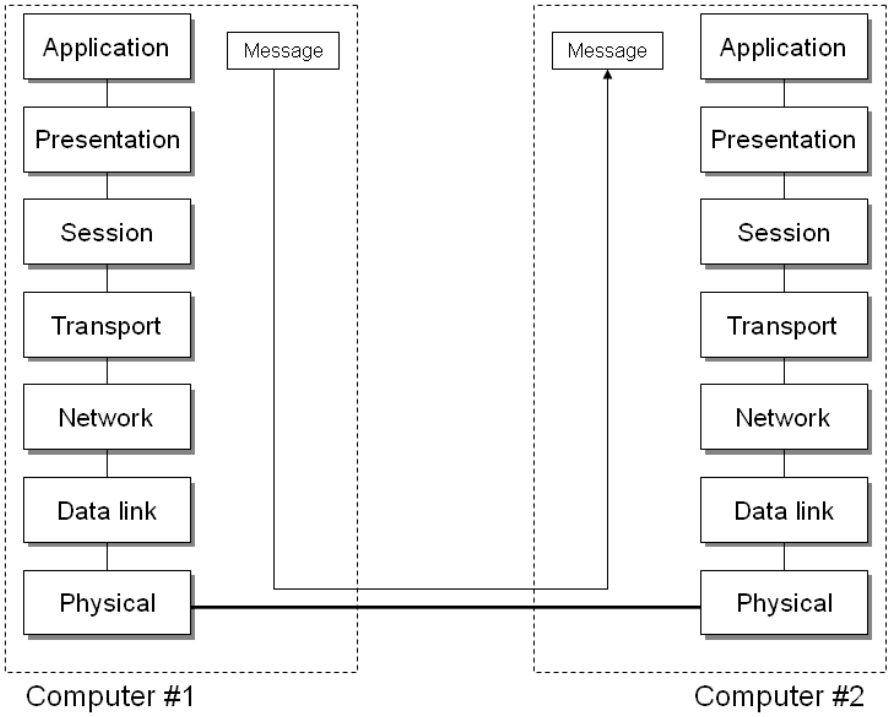


Figure 15.6: The OSI model on two computers (Olsson & Rosen 2003).

layers mainly in hardware, while the upper layers mainly in software.

The main benefits of the OSI will be:

1. more easy to understand the main functionality of networking,
2. more easy to understand the cooperation between hardware and software in networking,
3. more easy to compare functionality between protocol stacks,
4. more easy to understand the technologies used in different networks,
5. more easy to troubleshoot by separating networks into manageable pieces,
6. more easy to understand the functionality of a network (or different networks).

An more detailed overview of the communication layers are shown in Figure 15.7, showing the size of message and the functions. The size of the message to be transmitted on the physical layer often has a maximum size, so the message has to be split is smaller messages and merged on the remote computer.

A description of these layers are:

15.4.1 Layer 1: Physical Layer

The Physical layer defines all the electrical and physical specifications for devices. This includes the layout of pins, voltages, and cable specifications. Hubs, repeaters, network adapters and Host Bus Adapters (HBAs used in Storage Area Networks) are physical-layer devices. Components of the physical layer include:

1. Cabling system components,
2. Adapters that connect media to physical interfaces,
3. Connector design and pin assignments,
4. Hub, repeater, amplifiers, multiplexers, and patch panel specifications,
5. Wireless system components,

	Data unit	Layer	Function
Host layers	Data	Application	Network process to application
		Presentation	Data representation and encryption
		Session	Interhost communication
	Segments	Transport	End-to-end connections and reliability
Media layers	Packets	Network	Path determination and logical addressing (IP)
	Frames	Data link	Physical addressing (MAC & LLC)
	Bits	Physical	Media, signal and binary transmission

Figure 15.7: The data, layer, and functions information in the OSI model (www.wikipedia.org 2006).

6. Network Interface Card (NIC); the physical connection.

The major functions and services performed by the physical layer are:

1. Electrical and mechanical interfaces for a communication medium,
2. Establishment and termination of connection to a communication medium.
3. Sending and receiving the data bits across the physical media,
4. Participation in the process of effectively sharing the communication resources among multiple users.
5. Modulation, or conversion between the representation of digital data in user equipment and the corresponding signals transmitted over a communications channel. These are signals operating over the physical cabling (such as copper and fibre optic) or over a radio link.
6. "Simple" error detections like parity checking,
7. Fixed or maximum size of the data messages.

The physical layer consists of three sub layers, the physical layer convergence procedure (PLCP), the physical medium dependent layer (PMD), and the physical management layer (PML)(Tektronix 2013). The PLCP act as the adoption layer towards the upper layer, the data link layer, and building data packets for different physical layer technologies. PMD specifies the modulation and coding techniques for the physical layer. Station Management handle channel tuning and is responsible for cooperation between the data link layer and the physical layer. Figure 15.8 shows these sub layers in the physical layer and the connection to the upper layer in the OSI model, the data link layer. This figure focus on the IEEE 802 type of local area networks.

Each physical layer packet consists of a preamble field, header field and payload data field(Tektronix 2013). The preamble field is the first part of the packet to allow the receiver for time and frequency synchronization and estimate channel characteristics, consisting of a bit sequence. The number of bits is defined in the protocol. The header field is the next field and contains information about the packet configuration like format, data rates to mention some. The specific information must be defined in the protocol. The payload data field is the user data. Figure 15.9 shows the structure of the packet with the preamble field first, than the header field and the payload data field at the end of the packet.

The physical layer supports of a set of frames where each frame contains the packet with preamble, header including the addressing and frame sequence control, and the payload data including an error check. The layer may supports several types of frames like management frames, control frames, and data frames. Some of these frames do not have any payload data field.

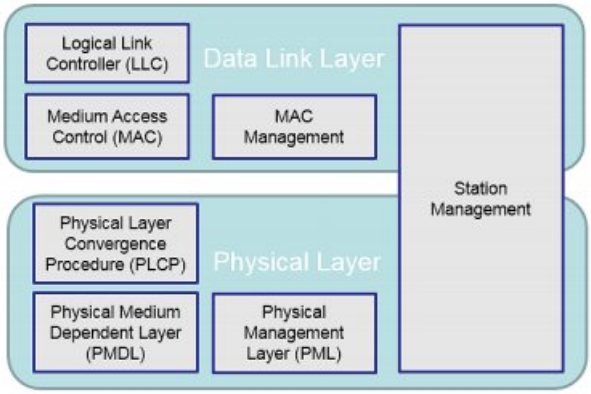


Figure 15.8: The sub layers of the physical layer in the OSI model. The Figure also shows the connection to the upper layer, the data link layer (Tektronix 2013).

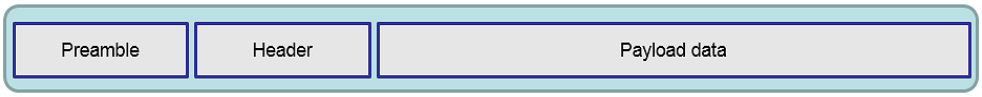


Figure 15.9: The packet from the physical layer showing the syncronization field, preamble, header field and the payload data field. The first field is the preamble field (Tektronix 2013).

15.4.2 Layer 2: Data Link Layer

The Data Link layer provides the functional and procedural means to transfer data between network entities and to detect and possibly correct errors that may occur in the Physical layer. The best known example of this is Ethernet. Other examples of data link protocols are HDLC and ADCCP for point-to-point or packet-switched networks and Aloha for local area networks. On IEEE 802 local area networks, and some non-IEEE 802 networks such as FDDI, this layer may be split into a Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. It arranges bits from physical layer into logical chunks of data, known as frames.

Functions of the Data Link Layer:

1. Assign a physical address to the network devices; identifies a specific computer on the network,
2. Let a device access the network for sending and receiving messages (frames, packets of data bits); provides the access and flow of data over a single link, the messages will be of fixed length,
3. Provides error detection and correction capabilities; this layer can request retransmission if communication fails,

This is the layer at which the bridges and switches operate. Connectivity is provided only among locally attached network nodes forming layer 2 domains for unicast or broadcast forwarding. Unicast addressing is sending information packets to a single network destination, multicast addressing is sending information packets to a group of network destinations, and broadcast is sending information packets to all network destinations.

Other protocols may be imposed on the data frames to create tunnels and logically separated layer 2 forwarding domain (www.wikipedia.org 2006).

Components of the data link layer include:

1. Network Interface Card; MAC addresses,
2. Switches and bridges; using the MAC address to filter and forward traffic, relieve the amount of messages.

15.4.3 Layer 3: Network Layer

The Network layer provides the functional and procedural means of transferring variable length of data sequences from a source to a destination via one or more networks while maintaining the quality of service (QoS) requested by the Transport layer. The Network layer performs:

1. network routing functions and finding the“best” route for devices not connected to a single link,
2. might also perform segmentation/desegmentation,
3. report delivery errors,
4. logical addressing; software addressing based on hardware addressing,

Routers operate at this layer, sending data throughout the extended network and making the Internet possible (also existing at layer 3 (or IP) are switches). This is a logical addressing scheme – values are chosen by the network engineer. The addressing scheme is hierarchical. The best known example of a layer 3 protocol is the Internet Protocol (IP). Perhaps it's easier to visualize this layer as the actual Air Mail or Consolidated Carrier that transfers the mail from Point A to Point B (www.wikipedia.org 2006). This layer provides an end-to-end logical addressing system so that a packet of data can be routed across several layer 2 networks. When passing packets between different networks, it may become necessary to adjust their outbound size to one that is compatible with the layer 2 protocol that is being used. The network layer accomplishes this via a process known as fragmentation. Some basic security functionality can also be set up by filtering traffic using layer 3 addressing on routers or other similar devices. Quality of Service (QoS) involves mechanism for quality and priority control of traffic in the network.

15.4.4 Layer 4: Transport Layer

The Transport layer provides transparent transfer of data between computers, thus relieving the upper layers from any concern while providing reliable data transfer. The transport layer controls the reliability of a given link through flow control, segmentation/desegmentation, and error control. Some protocols are state and connection oriented. This means that the transport layer can keep track of the packets and retransmit those that fail. The best known example of a layer 4 protocol is the Transmission Control Protocol (TCP). The transport layer is the layer that converts messages into TCP segments or User Datagram Protocol (UDP), Stream Control Transmission Protocol (SCTP), etc. packets. Perhaps an easy way to visualize the Transport Protocol is to compare it with a Post Office, that deals with the dispatching and classification of mail and parcels sent (www.wikipedia.org 2006).

Some of the functions offered by the transport layer include:

1. Application identification,
2. Client-side entity identification,
3. Confirmation that the entire message arrived intact; ensures error free delivery, ensure proper sequencing of data packets,
4. Segmentation of data for network transport, use sliding windows to avoid to many duplications of the data packets,
5. Control of data flow to prevent memory overruns,
6. Establishment and maintenance of both ends of virtual circuits,
7. Transmission-error detection,
8. Realignment of segmented data in the correct order on the receiving side,
9. Multiplexing or sharing of multiple sessions over a single physical link.

15.4.5 Layer 5: Session Layer

The Session layer controls the dialogues (sessions) between applications on computers. It establishes, manages and terminates the connections between the local and remote application. It provides for either duplex or half-duplex operation and establishes check pointing, adjournment, termination, and restart procedures. The OSI model made this layer responsible for "graceful close" of sessions, which is a property of TCP, and also for session check pointing and recovery, which is not usually used in the Internet protocols suite (www.wikipedia.org 2006).

Session layer functionality includes:

1. Virtual connection between application entities; responsible for opening, using, and closing sessions (application connections),
2. Synchronization of data flow,
3. Creation of dialog units,
4. Connection parameter negotiations,
5. Partitioning of services into functional groups,
6. Acknowledgements of data received during a session,
7. Retransmission of data if it is not received by a device.

15.4.6 Layer 6: Presentation Layer

The Presentation layer transforms data to provide a standard interface for the Application layer. MIME[2] encoding, data compression, data encryption and similar manipulation of the presentation is done at this layer to present the data as a service or protocol developer sees fit (www.wikipedia.org 2006).

Examples of presentation layer functionality include:

1. Encryption and decryption of a message for security,
2. Compression and expansion of a message so that it travels efficiently,
3. Graphics formatting,
4. Content translation,
5. System-specific translation.

15.4.7 Layer 7: Application Layer

The Application layer provides a means for any application to access information on the network. This layer is the main interface for any application or the operating system to interact with the network. Some examples of application layer protocols include Telnet, File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and Hypertext Transfer Protocol (HTTP). Applications built to utilize a protocol, such as FTP, should not be confused with the protocols themselves, which often reside at the session layer (www.wikipedia.org 2006).

Examples of application layer functionality include:

1. Interface between applications on the computer and the operating system like a) support for file transfers, b) ability to print in the network, c) electronic mail, d) electronic messaging, d) Browsing the World Wide Web,
2. Any API (application programming interfaces),
3. Defines a set of "message commands" for data exchange between applications on different computers.

15.4.8 Layer overview

A short overview of the layers in the OSI model will be:

1. the physical layer; concerned with the physically communication,
2. the data link layer; access to the communication medium, logical transfer of data, and physical addressing,
3. the network layer; concerned with logical addressing of the nodes and routing schemes,

[2] MIME: Multipurpose Internet Mail Extensions is an Internet Standard that extends the format of e-mail to support among others different character sets, attachments, and multipart messages.

4. the transport layer; responsible for end to end communication control,
5. the session layer; concerned with the execution of any remote actions,
6. the presentation layer; responsible for any data interpretations,
7. the application layer; provide the services that are required by the applications.

Each layer contains a collection of related functions, provides services to the layer above it, and receives services from the layer below it. A layer will only communicated with the layer above, the layer below, and the same layer on the remote node.

Chapter 16

Local Area Network (LAN)

Connecting several computers in a local area and if each one of them should be able to communicate with all the others, a local area network (LAN) is needed. A LAN is basically a cable, twisted pair, coax or optical fibre, interconnecting all the computers through a dedicated interface. The length of a LAN can be from some meters up to a few kilometers and the supported data rates up to several Gbit/s (Olsson & Piani 1998).

The interconnection of LANs in wider area is often named Wide Area Network (WAN). Interconnection of LANs in specific area is often named Metropolitan Area Networks (MAN). See Figure 16.1 for the relationship between LAN, MAN, and WAN.

16.1 Network Topologies

The computers can be connected in different ways, being the topologies of the network. The topology describe either the physical topology or the logical topology. Physical topology is how nodes are physical interconnected, logical topology is how information is exchanges between the nodes. Two nodes can communicate either directly or through one or more other nodes. The most important topologies are (Olsson & Piani 1998):

1. Bus; all computers share a common line for communication,
   (a) Advantages:
      i. Easy to implement and extend,
      ii. Easy to install,
      iii. Well-suited for temporary or small networks not requiring high speeds (quick setup), resulting in faster networks,
      iv. Cheaper than other topologies (But in recent years has became less important due to devices like a switch),
      v. Cost effective; only a single cable is used,
      vi. Easy identification of cable faults,

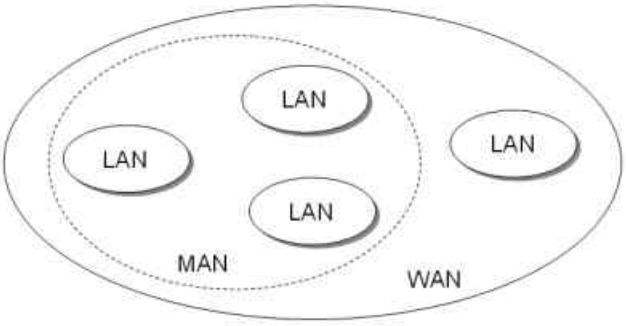


Figure 16.1: The relationship between LAN, MAN, and WAN.

vii. Reduced weight due to fewer wires.

(b) Disadvantages:

i. Limited cable length and number of stations,
ii. If there is a problem with the cable, the entire network breaks down,
iii. Maintenance costs may be higher in the long run,
iv. Performance degrades as additional computers are added or on heavy traffic (shared bandwidth),
v. Proper termination is required (loop must be in closed path),
vi. Significant Capacitive Load (each bus transaction must be able to stretch to most distant link),
vii. It works best with limited number of nodes,
viii. Commonly has a slower data transfer rate than other topologies,
ix. Only one packet can remain on the bus during one clock pulse.

2. Star; all computers are connected to a central point, like a hub, switch or concentrator,

(a) Advantages:

i. Better performance: star topology prevents the passing of data packets through an excessive number of nodes,
ii. Isolation of devices: Each device is inherently isolated by the link that connects it to the hub,
iii. Benefits from centralization: As the central hub is the bottleneck, increasing its capacity, or connecting additional devices to it, increases the size of the network very easily,
iv. Simplicity: This topology is easy to understand, establish, and navigate,
v. Easy to install and wire,
vi. Easy to detect faults and to remove parts,
vii. No disruptions to the network when connecting or removing devices.

(b) Disadvantages:

i. High dependence of the system on the functioning of the central hub,
ii. Failure of the central hub renders the network inoperable,
iii. The performance and scalability of the network depends on the capabilities of the hub,
iv. Network size is limited by the number of connections that can be made to the hub,
v. Other nodes may see a performance drop if traffic to another node occupies a significant portion of the central node's processing capability or throughput,
vi. Wiring up of the system can be very expensive.

3. Ring, all computers are sharing a common line for communication, connected as a ring,

(a) Advantages:

i. Very orderly network where every device has access to the token and the opportunity to transmit,
ii. Performs better than a bus topology under heavy network load,
iii. Does not require network server to manage the connectivity between the computers.

(b) Disadvantages:

i. One malfunctioning workstation or bad port in the MAU can create problems for the entire network,
ii. Moves, adds and changes of devices can affect the network,
iii. Network adapter cards and MAU's are much more expensive than Ethernet cards and hubs,
iv. Much slower than an Ethernet network under normal load.

4. Tree, computers are connected in cascade to other hubs or computers,

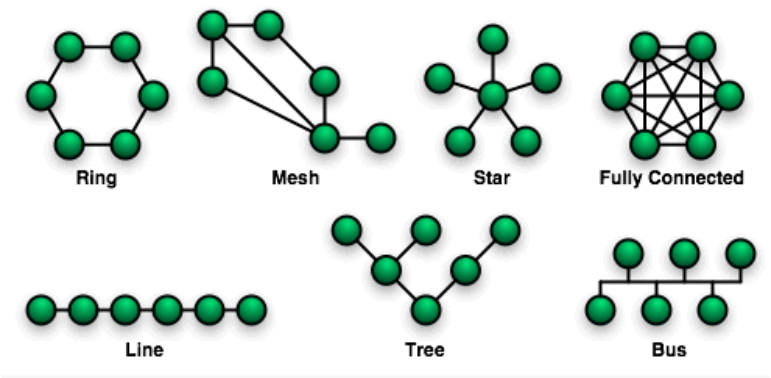


Figure 16.2: Different local area network topologies. The Mesh network topology shows a partially connected network.

(a) Advantage: Central hub (repeater) increases the distance a signal can travel between devices.
(b) Disadvantages:
    i. More cabling is required in a tree than in other topologies,
    ii. Entire network collapse if central Hub fails.

5. Mesh, connect to several neighbor nodes, partially or fully connected. Used mostly for wireless connections, connecting to the closest neighbors only, to save energy.

6. Mixed, mixing of several topologies.

The different network topologies are shown in Figure 16.2.

The topologies consist of either point-to-point configuration or multidrop configuration. In point-to-point configuration each computer has to receive a message, check the receiver of the message, pass the message on to the next computer if the address is not correct. In multidrop configuration will all computers receive the same message at the same time, then they only need to check the address of the message and just forget it if the address is not correct. Electrical connection is also different, in point-to-point connection only one receiver is connected to one transmitter, in multidrop connection a lot of receivers are connected to one transmitter.

Important factor for the communication topology and layout will be:

1. point-to-point or multidrop,

2. type of message traffic between the computers,

3. the distance between the computers,

4. the requirement for speed (message size, message numbers and response),

5. the disturbances in the environment.

The bus access can be:

1. deterministic / stochastic,

2. master / slave,

3. token ring

4. publisher / subscriber

and the protocol will define how the nodes can access the bus. Only one node can use a specific part of the bus for communication at a specific period of time.

16.2 IEEE 802 Standards

IEEE 802 referees to a family of IEEE standards dealing with LAN and MAN. Some of the standards are:

	IEEE 802 Standards
802.1	Bridging and Management
802.2	Logical Link Control
802.3	Ethernet - CSMA/CD Access Method
802.4	Token Passing Bus Access Method
802.5	Token Ring Access Method
802.6	Distributed Queue Dual Bus Access Method
802.7	Broadband LAN
802.8	Fiber Optic
802.9	Integrated Services LAN
802.10	Security
802.11	Wireless LAN
802.12	Demand Priority Access
802.14	Medium Access Control (MAC)
802.15	Wireless Personal Area Networks
802.16	Broadband Wireless Metro Area Networks

16.3 Medium Access Control (MAC)

LANs are covered by the layers 1 and 2 in the OSI model. Layer 1 is the physical media, and layer 2, data link, is often divided[1] into two layers (Olsson & Piani 1998):

1. Medium Access Control (MAC),
2. Logical Link Control (LLC).

MAC is responsible for the coordination of the access to the network and all hardware accessing a IEEE802.2 type of network needs MAC address. All MAC addresses are unique, 48 bits (6 bytes), and located in hardware of each network device. These MAC addresses can not be changed, and must be set by the maker of the hardware device (chip).

LLC is responsible for the protocol and error detection/correction of the layer.

This separation means that MAC is the interface to layer 1 and LLC is the interface to layer 3.

OSI model	Layers	Data link layer
3	Network	
		Logical Link Control
2	Data link	
		Medium Access Control
1	Physical	

In wireless communication will all nodes use the same space; the air. If all nodes communicate at the same time, collisions of communication will occur and loss of data will be the result. The media access must be handle in some way, and the two most used principles are:

- TDMA (Time Division Multiple Access); each node will have a given time slot for communication.
- CSMA (Carrier Sense Multiple Access); use a random time after a free channel is detected before starting the communication..

[1] The IEEE 802.2 layer.

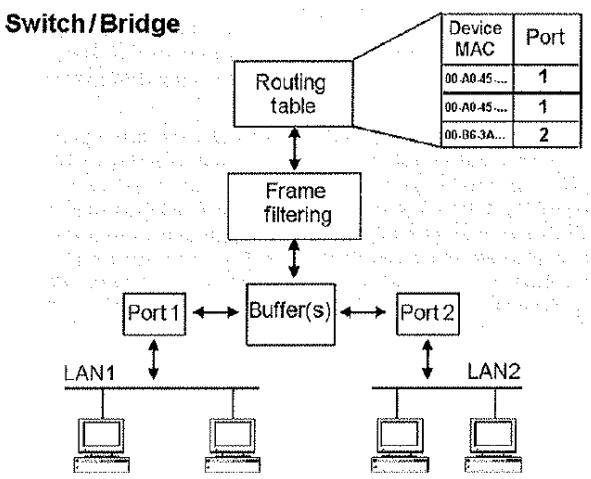


Figure 16.3: A switch used for routing of messages between 2 different network segments (Bormann & Hilgenkamp 2006).

16.4 Network communication devices

A set of devices exist for a network, some of these devices are, with reference to the OSI model:

1. Repeaters; a signal regenerator for connecting two or more network segments, layer 1 in the OSI model, often used with coax cables (old type, not used anymore),
2. Hubs; a multi port repeater, regeneration one input signal on all other ports, layer 1 in the OSI model, often used with TP cables (old type, not used anymore),
3. Bridges; a multi port repeater but using level 2 in the OSI model to filter the traffic using the MAC addresses, used to connect two networks (old type, not used anymore),
4. Switch; used to link together network segments using the level 2 in the OSI model (linking based on the MAC address). Designed to divide the network into separate collision domains to increase the number of nodes and the length of the LAN. Two types of switches exist, unmanaged and managed switches. The difference is:
   (a) Unmanaged switches; These switches have no configuration interface or options. They are plug-and-play and least expensive devices.
   (b) Managed switches; These switches have an configuration interface like a serial port, a web server or a Simple Network Management Protocol (SNMP). Configuration options are
      - enable features such as Spanning Tree Protocol;
      - set port speed;
      - create or modify VLANs;
      - security settings.
5. Routers; used to link together network sections using the level 3 in the OSI model (linking based on the logical IP address). To make it easier to manage the network and control the flow of packets, many organizations separate their network layer addressing into smaller parts known as sub nets. Routers use the network or sub net portion of the IP addressing to route traffic between different networks. Each router must be configured specifically for the networks or sub nets that will be connected to its interfaces.
6. Gateways; used to connect different networks and applications using all levels in the OSI model. The gateway is often only a router with added software support for layer 7 of the OSI model.
7. Firewall; a set of rules defining the network traffic passing trough the device. These rules are based on the network addresses, protocols and port numbers,
8. DMZ; A demilitarized zone; a subnetwork with services for an external (not trusted) network,

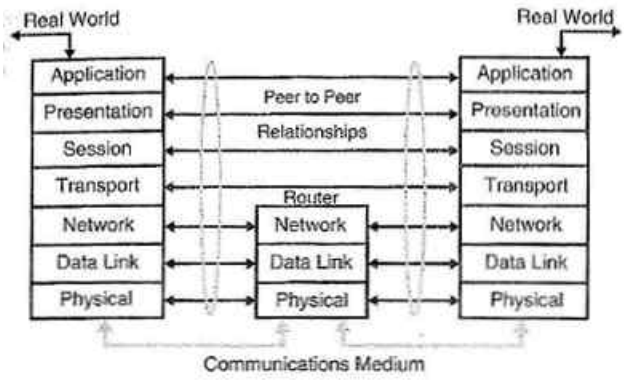


Figure 16.4: The OSI model with the services of a router (Mackay et al. 2004).

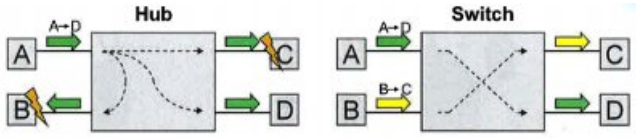


Figure 16.5: The difference of a hub and a switch. A hub, to the left, will copy any messages on one port to all other ports, while a switch, to the right, will only transmit the message on the right port. A switch will increase the message troughput in any network with respect to a hub (https://www.phoenixcontact.com: oct:12).

9. Node; a computer with communication capabilities.

These devices can be a combination of both hardware and software as layer one and two is mainly hardware, while layer three and above is mainly software. A router as a hardware device will then support both layer one, two and three. So what is the difference of a hub and a switch? A hub is working only on layer 1 and can just copy a message from one port to all the other ports. The message will then be duplicated on all the sub networks connected to the hub and increase the message traffic in the network. A switch is working on both layer 1 and 2, and can use the MAC addressing information to route a specific message from one port to another specific port. The switch needs to contain some software to be able to interpret the protocols in use, be able to extract the MAC address from the protocol and store the MAC address in an address table. As the messages needs to contain both the transmitting and receiving MAC addresses the switch will be able to maintain the address table whenever a device is disconnected or a new device is connected. Figure 16.5 shows the difference of a hub and a switch regarding the message throughput in a network.

Figure 16.6 shows an example of a LAN using a firewall for connected to the internet. The LAN contains a firewall, a server, a set of clients including a wireless connection.

A Firewall is a device defining a set of rules controlling the usage of network addresses, protocols and port numbers of the applications. As long as a network is behaving well and disciplined, these rules is OK. The next generation of firewalls has to be more intelligent and should be based on identifying the applications independent of addresses, protocols, ports, or encryption. The firewalls should also be able to protect the system from attacks from installed applications and check of the data stream in any network without a major degradation of the speed.

16.5 Ethernet (IEEE 802.3)

Ethernet is one of the protocols that are used for communication between nodes on a LAN.

16.5.1 Ethernet History

The topology first started as a bus topology, being a multidrop solution. Today the solution is still multidrop, but using a mixed topology in a combination of bus, star and tree. The number of different cables for the physical layer is:

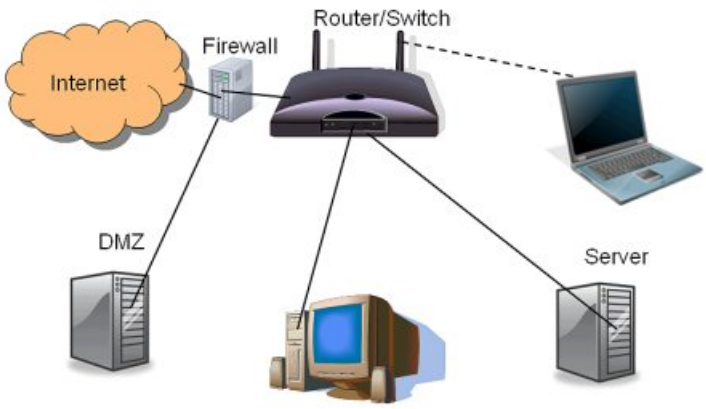


Figure 16.6: An example of a LAN with an router, connected to internet using a firewall. One computer is connected to the DMZ port in stead of the LAN.

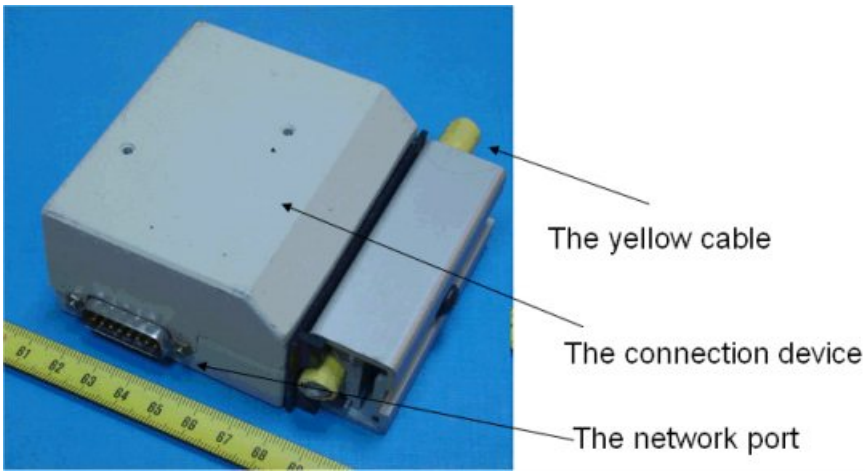


Figure 16.7: A 10Base-5 network port connected to the "yellow" Ethernet cable (www.wikipedia.org 2006).

Specification	**Cable**	**Code**	**Bus type**	**Year**	**Note**
Thick Ethernet	"Yellow cable"	10Base-5	Bus	1980..	
Thin Ethernet	Coax cable	10Base-2	Bus	1990..	RG-58
Fibre Ethernet	Optic fibre	10Base-F			
Twisted pair	CAT-x[2]	10Base-T	Star	2000..	
		100Base-T	Star		
		1000Base-T	Star		

These 10Base-x codes, defined by yyBase-x, where yy indicateds the speed in MBits/s, BASE is short for baseband signalling as opposed to broadband, and x stands for the maximum segment length in multiply of 100 meters. Ethernet started in the early 1980s with the thick yellow Ethernet cable, about 1 cm in diameter, thick and stiff, not very practical in narrow areas. The maximum length between two nodes on thick Ethernet can be 500 m, the recommendation (from IEEE) is in intervals of 2.5 m. The number of nodes (network computers) are unlimited, but the recommendation is maximum 100 in each segment. New nodes can easily be mounted on the cable. An example of a 10Base-5 connection is shown in Figure 16.7.

As this cable was not very practical in use, the thin Ethernet cable using a standard RG-58 cable become more popular in the end of the 1980s. As the RG-58 cable was cheaper, the thin Ethernet was also called Cheapernet. Thin Ethernet cable has *a*) a diameter of 0.5 cm, *b*) must have terminator

[2] x being $4, 5, 6$, or .., meaning among others the number of twisting pr. inch.

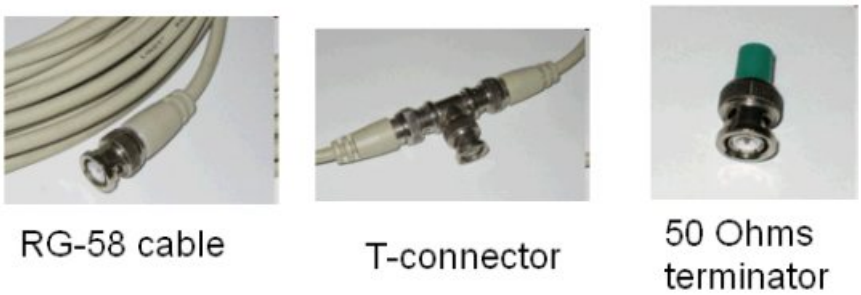


Figure 16.8: The connection devices for the 10Base-2 Ethernet (www.wikipedia.org 2006).

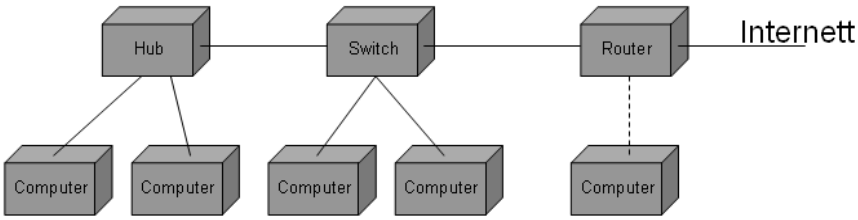


Figure 16.9: Ethernet with 10Base-T using a combination of start and bus topology.

of 50 Ω at each end of the cable to avoid signal reflections (See Figure 16.8), *c*) a maximum length of each segment of 185 m. The number of nodes (computers) are unlimited, but the recommendation is maximum 30, connection of nodes (computers) using T connections (See Figure 16.8), and repeaters can be used for connecting several segments.

The T connector, the cable and the 50 ohms terminator are shown in Figure 16.8.

The main challenges using Thin Ethernet:

- The T-connector (See Figure 16.8),
- The BNC connectors on the T-connector (See Figure 16.8),
- The cable from the T-connector to the computer,
- All communication was lost when problems with a T-connector, due to the bus topology,
- Difficult to find the problem when all communication was lost.

These challenges, especially the T connectors, were some of the reasons for developing the 10Base-T standard, using twisted pair cables, and changing the topology. The topology now changed to a mixture of bus, stars, and tree from the stars, shown in Figure 16.9. 10Base-T is using a form of RS-485 as a physical layer, and using separate pair of cables for Rx and Tx.

The number of nodes (computers) and the communication traffic will decide the throughput of the network, and as the number of nodes (computer) rise the number collision will rise. The number of collision will be an important factor of the trough-put of the network.

16.5.2 CAT Cable

The network cables used today are called CAT cables and are twisted pair cable type designed for high signal integrity. Normally these cables are unshielded but they can also be shielded in environment with EMC or other type noise. The codes CAT-4, CAT-5, and CAT-6 differs among others in the number of twists of the cable pairs per. inch. The CAT5e is an extension of the CAT5 specification and is often used in structured cabling for computer networks such as Ethernet. The maximum length of CAT cables is 100 m, normal practice is to use maximum 90 m to allow for about 5 m (patch cable) at each ends of the connections. Another difference between the cables will be the maximum frequency for the cable as shown:

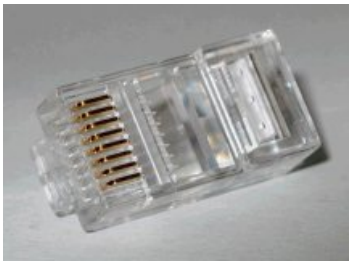


Figure 16.10: The 8 Position 8 Contact (8P8C) modular plugs and jacks are communications connectors. These connectors are often mistakenly called RJ45 plugs and jacks (www.wikipedia.org 2006).

Cable	Maximum Frequency	Mbit/s	Pairs	Note
Cat4	20 MHz	16	2	
Cat5	100 MHz	100	2	
Cat5e	100 MHz	100	2	
Cat6	250 MHz	1000	4	
Cat7	600 MHz			

The CAT cable for 10Mbit/s and 100Mbit/s are using only 2 pairs, while the 1000Mbit/s cables are using 4 pairs. The 1000Mbit/s network is also often referred to as Gigabit network.

The CAT cable is also referring to the AWG standard (American Wire Gauge) using the sizes AWG26 to AWG22. The number is an indication of the cross-section of the cable, a higher number is a lower cross-section. This cross-section of the cable as shown:

AWG	Cross-section	Diameter
26	$0,128\,\mathrm{mm}^2$	$0,404\,\mathrm{mm}$
25	$0,162\,\mathrm{mm}^2$	$0,455\,\mathrm{mm}$
24	$0,205\,\mathrm{mm}^2$	$0,511\,\mathrm{mm}$
23	$0,259\,\mathrm{mm}^2$	$0,574\,\mathrm{mm}$
22	$0,324\,\mathrm{mm}^2$	$0,643\,\mathrm{mm}$

Cable types, connector types and cabling topologies are defined by TIA/EIA568A/B. The connectors, the 8P8C modular connectors, is often incorrectly referred to as "RJ-45", are used for connecting category 5 cable. The RJ45 refers to a standardized physical interface as well as a standard wiring scheme, 8P8C refers only to the connector. The 8P8C connector is shown in Figure 16.10.

The wiring scheme is defined by TIA/EIA568A/B and shown in Figure 16.11. A "normal" cable will have a T568B in both ends, a straight through connection. A patch cables, however, will have a T568A connector in one end of the CAT cable and a T568B connector in the other end of the CAT cable.

16.5.3 Standards and cables

Ethernet standard	**Cable**	**Speed (Mbit/s)**
10Base-5	Yellow, thick	10
10Base-2	Coax, RG-58	10
10Base-F	Fiber optic	10
10Base-T	CAT3 or better	10
100Base-T(X)	CAT5 or better	100
100Base-F(X)	Fiber optic	100

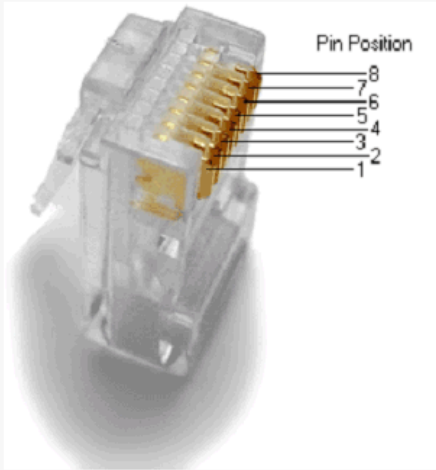
Pin	T568A Pair	T568B Pair	Wire	T568A Color	T568B Color	Pins on plug face (jack is reversed)
1	3	2	tip	white/green stripe	white/orange stripe	
2	3	2	ring	green solid	orange solid	
3	2	3	tip	white/orange stripe	white/green stripe	
4	1	1	ring	blue solid	blue solid	
5	1	1	tip	white/blue stripe	white/blue stripe	
6	2	3	ring	orange solid	green solid	
7	4	4	tip	white/brown stripe	white/brown stripe	
8	4	4	ring	brown solid	brown solid	

Figure 16.11: The pins, cable pairs and cable color for the CAT cables (www.wikipedia.org 2006).

16.5.4 Ethernet Protocol

The protocol used for Ethernet is Carrier Sense Multiple Access With Collision Detection (CSMA/CD), a network control protocol in which a carrier sensing scheme is used. A transmitting data station that detects another signal while transmitting a frame, stops transmitting that frame, transmits a jam signal, and then waits for a random time interval before trying to send that frame again. An simplified algorithm of the CSMA/CD is shown in Figure 16.12 using an UML[3] activity diagram.

CSMA/CD description:

1. Carrier Sense (CS): Every node in the network must always check if a carrier is present on the transmission media as an indication if the transmission media is busy or not. If no carrier present, the transmission media is not in use, and any node can use the media. However, all nodes should wait a random time and rechecking if it is available, before using it. The reason is that several nodes can be waiting for the communication media, and there will be a collision if all nodes try to use it at the same time.

2. Multiple Access (MA): Any node can use the communication media meaning that the network don't have any master controlling the access to the media. As long as every node is using the CSMA/CD protocol, any node can use the communication media when no carrier is detected. The carrier is a communication signal used for transmitting data on any media, including wireless.

3. Collision Detect (CD): Transmit a bit and check the bit received from the network. If the same bit, continue sending the rest of the message. If not equal, wait a random time and try to resent the message.

Figure 16.13 shows both situations using the CSMA/CD protocol, in the upper rows a transmission without collision, and in the bottom row a transmission from several nodes that will result in a collision.

Figure 16.14 shows the flowchart of a data collision. Node #2 determines that the medium is available and starts transmitting at moment A. Shortly before the data reaches node #3, node #3 also determines a available medium and starts transmitting data (moment B). At moment C and D will the data from node #2 reach node #1 and node #3. Node #3 will now detect a collision, at moment D, transmit a

[3] UML: Unified Modeling Language

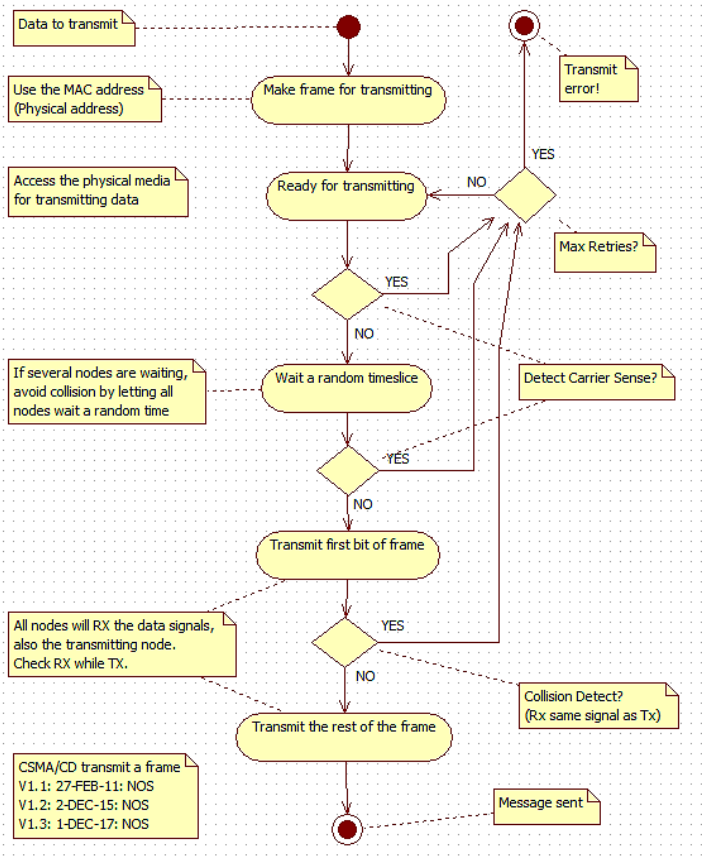


Figure 16.12: An UML activity diagram describing the algorithm for sending a frame using the CSMA/CD protocol.

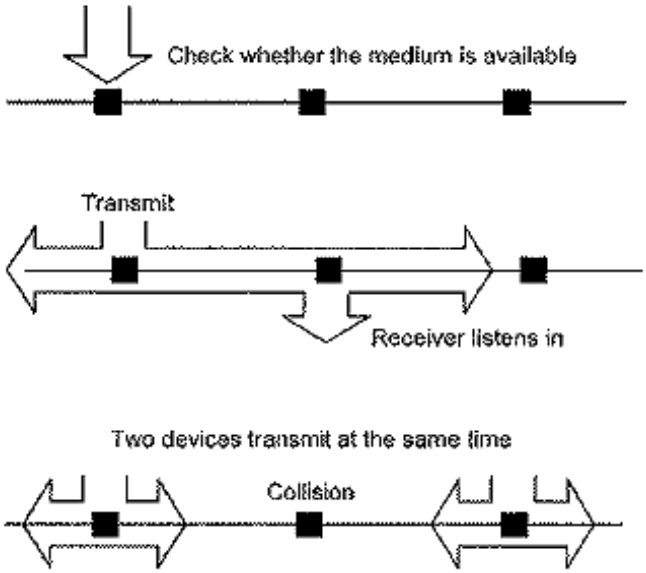


Figure 16.13: The upper rows show a situation without a collision, one node is checking the medium and will transmit a message. The bottom row shows a collision situation where 2 nodes are transmitting at the same time (Bormann & Hilgenkamp 2006).

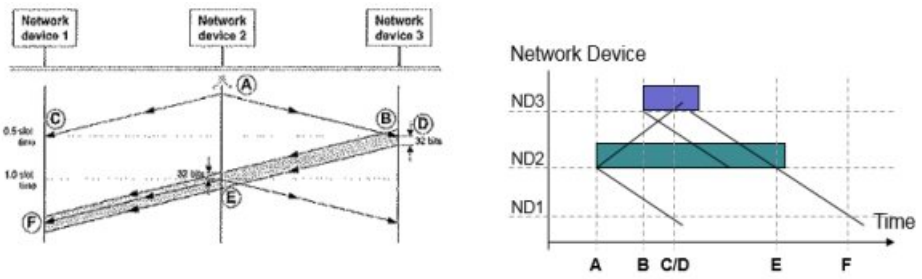


Figure 16.14: The transmission of messages on a communication media using CSMA/CD with respect to time. (Bormann & Hilgenkamp 2006).

jam signal, and abort the transmission. Within the slot time of 51.2 μs will the jam signal reaches the other nodes, moment E and moment F.

If the message is shorter than the slot time, will the node #3 not detect the collision and will not retransmit the data. Each node will wait a random time when detecting a collision before retransmit the message. A node will try to retransmit the data up to 16 times according to the standard.

If the number of nodes in a segment is high the probability for a collision is higher, organizing the network in several segments can lower the collision rate. If retransmission more then 16 times, the node will indicate a transmission error to the above layer.

In computer networking a Media Access Control address (MAC address) is a unique identifier attached to most network adapters (NICs). It is a number that acts like a name for a particular network adapter, so, for example, the network cards (or built-in network adapters) in two different computers will have different names, or MAC addresses, as would an Ethernet adapter and a wireless adapter in the same computer, and as would multiple network cards in a router.

CSMA/CD is a modification of pure Carrier Sense Multiple Access (CSMA). Collision detection is used to improve CSMA performance by terminating transmission as soon as a collision is detected, and reducing the probability of a second collision on retry.

Methods for collision detection are media dependent, but on an electrical bus such as Ethernet, collisions can be detected by comparing transmitted data with received data. If they differ, another transmitter is overlaying the first transmitter's signal (a collision), and transmission terminates immediately. A jam signal is sent which will cause all transmitters to back off by random intervals, reducing the probability of a collision when the first retry is attempted. CSMA/CD is a layer 2 protocol in the OSI model.

How many nodes can be connected to a CSMA/CD type of network? MA indicates Multiple Access, in theory this means infinity. In practise, an increasing number of nodes will give a higher probability for CS and CD in Figure 16.12 giving a higher delay and again giving more CS and CD. A practical advice will be less then 30 to 50 nodes, depending on the topology and requirement for speed.

16.6 Token Ring (IEEE 802.5)

Token ring is another protocol used for communication between nodes on a LAN. Token ring is based on different approach than CSMA/CD, but both protocol can be used for layer 2 in a network. Token Ring is a solution introduced by IBM using a ring topology, in stead of the bus topology, and a software "token" on the ring. The token is a special three-byte frame that travels around the ring. When a computer receives the token, this computer is the master of the communication line and can transmit any messages. When the transmission is finished, or no transmission is needed, the token must be passed on to the next computer. This protocol is a multi master protocol. This topology must be a point-to-point solution as only one computer at a time can receive the token. Is CSMA/CD a point to point or a multidrop topology?

An simplified algorithm of the Token ring receive function is shown in Figure 16.16 using an UML[4] activity diagram. The figure shows the algorithm for receiving a message. Note the special case if receiving the token, if receiving a token the transmit section is activated and this node will be the master.

[4] UML: Unified Modeling Language

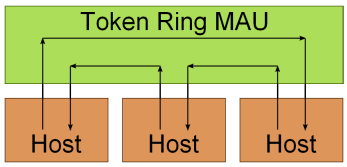


Figure 16.15: Three hosts connected in a token ring nework. A Multi-station Access Unit (MAU) is used for connection of the hosts (www.wikipedia.org 2010).

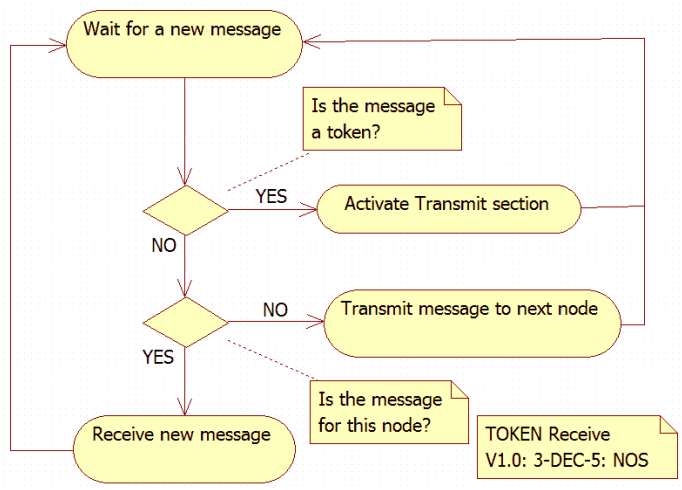


Figure 16.16: An UML activity diagram describing the algorithme for receiving messages in a Token ring system.

An simplified algorithm of the Token ring receive function is shown in Figure 16.17 using an UML[5] activity diagram. The figure shows the algorithm for transmitting a message, and is activated from the receive section when receiving a token.

IBM was a competitor to the Thick and Thin Ethernet bus topology, but today token ring is almost unused comparing to the 10Base-T standard.

16.7 TCP/IP protocol

The TCP/IP (Transmission Control Protocol / Internet Protocol) is a set of communication protocols to operate across complex networks. The full TCP/IP is more correct called Internet Protocol Suite (IPS) where TCP and IP are just two of the protocols included in IPS. The protocol suite may also be called a protocol stack as the protocol suite is a stack based on the OSI layers. TCP/IP has a layered structure similar to the OSI model, but with fewer layers.

Figure 16.18 shows the TCP/IP protocol suite or TCP/IP protocol stack where the protocol is based on the Ethernet protocol for the lower layers and other application protocols for layer 7. The node will transmit a message from one node to one or several other nodes. The messages in a network are often referred to as datagrams. Datagrams are packets of information that can be destined for one, many or all the computers in the network, having a unique, multicast**??** or broadcast address. The datagrams contains information about the protocol, tx and rx addresses, protocol control information and the data (message). Figure 16.19 shows the addressing modes where the addressing mode for a specific message will be part of the protocol control information in the datagram. So basically will all nodes receive all datagrams, but based on the protocol control information and rx address will either layer 2 or 3 reject the datagram if not as a valid address.

The protocol control information is normally part of the header for the protocol, and the header structure of the TCP/IP protocol will be shown later in this chapter.

Looking more into the layered structure of the TCP/IP protocol, Figure 16.20 shows the TCP/IP protocol where layer 3, the IP layer, is the "heart" of the protocol. The IP layer is responsible for

[5] UML: Unified Modeling Language

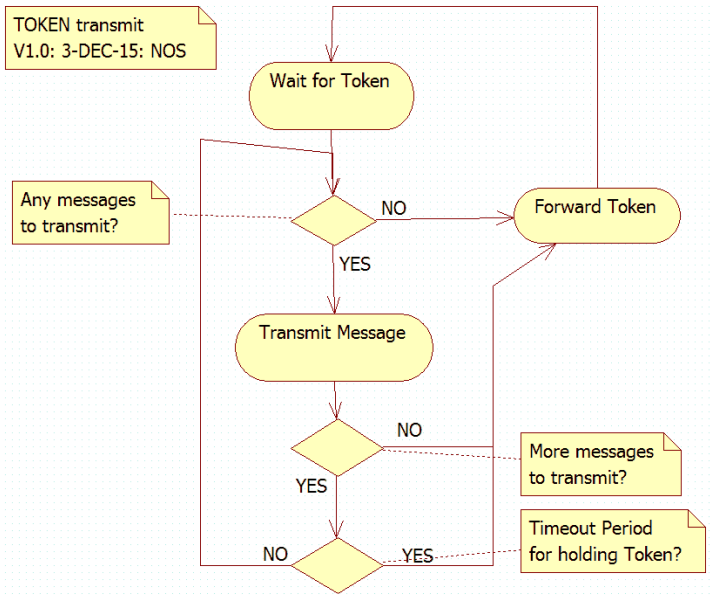


Figure 16.17: An UML activity diagram describing the algorithme for transmitting messages in a Token ring system.

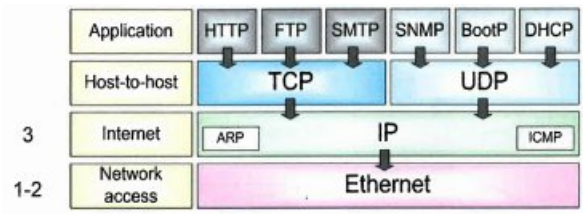


Figure 16.18: The TCP/IP protocol stack based on the Ethernet protocol in layer 1 and 2, and supporting IP for layer 3 and either TCP or UDP as part of layer 4. Note also that an application protocol is needed to be able to the TCP/IP protocol (www.phoenixcontact.com: oct-12).

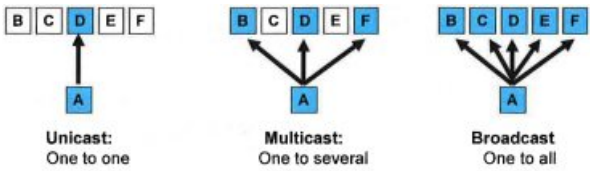


Figure 16.19: The unicast, multicast and broadcast addressing in a network. Node A wants to transmit a datagram and any of the nodes B, C, D, E or F can be the receivers of the datagram (https://www.phoenixcontact.com: oct:12).

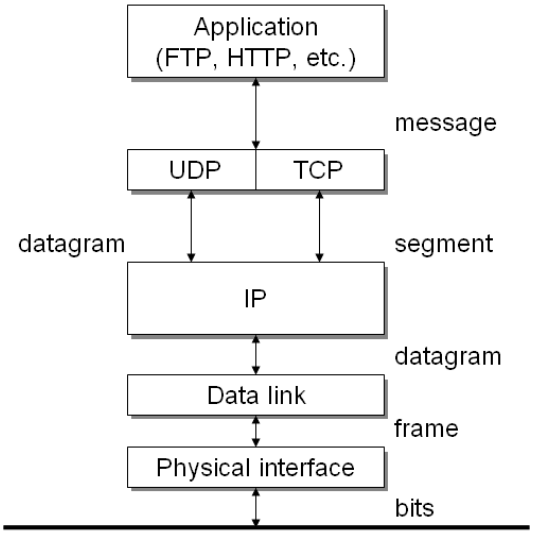


Figure 16.20: The basic structure of the TCP/IP protocol (Olsson & Piani 1998).

moving the datagram from one computer in the network to another computer in the network. IP layer does not check the delivery of the datagram nor the sequence of the datagram, higher layers have this responsibility.

The transport layer, consisting of User Datagram Protocol (UDP) and Transmission Control Protocol (TCP), has the responsibility for delivery checking and sequence control of the datagram forming the messages.

UDP is used for applications with several recipients (multicast) being unpractical and expensive to acknowledge the reception of every datagram, instead the information can be updated often. UDP can also be used for transmission where speed is more important than the reception of every datagram like video etc., in video some of the images can be lost without destroying the information.

TCP is the full-fledged protocol with handshaking, error detection and correction, and flow control. This protocol has a higher overhead giving a high reliability of the service.

The application layer can not be compared with the layers 5 to 7 in the OSI model as this layer only include specific application programs that uses the TCP/IP protocol like the File Transfer Protocol (FTP) and the HyperText Transfer Protocol (HTTP) used for the World Wide Web. Figure 16.21 shows the internet protocols with references to the OSI model.

Addressing on different layers

The addressing will be available on several layers based on the network structure:

- The Media Access Control (MAC) addressing is based on a 48 bits unique physical address and will be limited to the segment where the devices is physical connected, working on layer #2 of the OSI model,
- Internet Protocol (IP) based on a 32 bits logical address for the version 4 protocol (IPv4 or just IP) and a 128 bits logical address for the version 6 protocol (IPv6), working on layer #3 of the OSI model,
- Address Resolution Protocol (ARP) is based on the IP address, and will use a IP to MAC address lookup system, working as a connection point (link) between layer #2 and layer #3 of the OSI model,
- Domain Name System (DNS) is based on a logical name (for a domain) and will use a Name to IP address lookup system, working as a service in layer #7 of the OSI model.

16.7.1 Internet and address

An internet (TCP/IP) address consists of 4 bytes, xxxx:xxxx:xxxx:xxxx, for example 192.168.0.1. Two computers using the TCP/IP protocol can communicate using repeaters, hubs, bridges and routers if

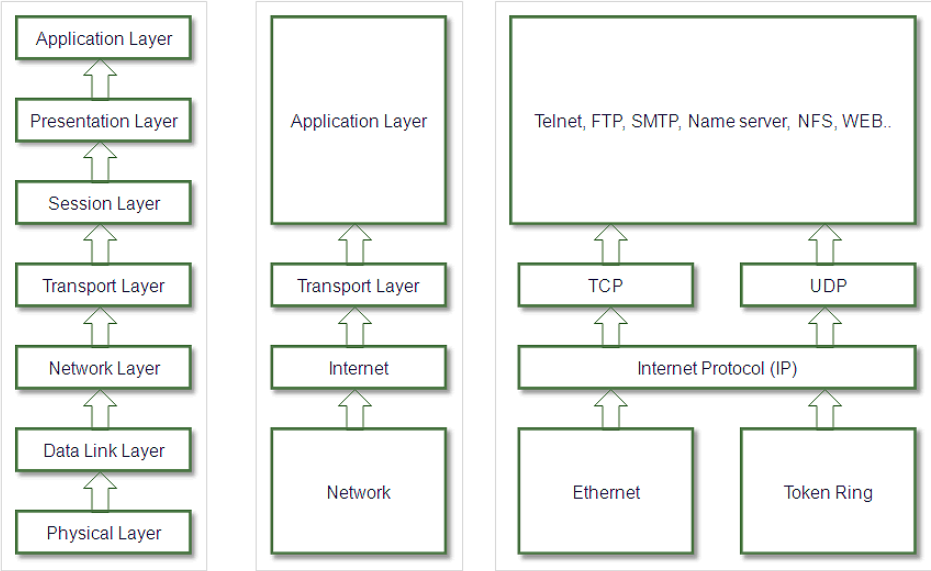


Figure 16.21: Internet protocols with references to the OSI model.

they are assigned the same address class. Note that two computers on internet can NOT have the same address so the internet addresses must be assigned to an organization by a valid authority. The Internet Assigned Numbers Authority (IANA) manages the IP address space allocations globally and delegates five regional Internet registries (RIRs) to allocate IP address blocks to local Internet registries (Internet service providers) and other entities.

The OSI model is supporting physical addressing in layer 2 and logical addressing in layer 3. The internet address is a logical address used in layer 3. The first part of the address identifies the network on which the host resides, while the second part identifies the particular host on a given network. This two level addressing hierarchy is illustrated as:

Network number Host number

All host on a given network share the same network number but must have unique host numbers. Any two hosts on different networks must have different network numbers, but can have the same host number. The Internet was designed to support different sizes with different classes. The TCP/IP protocol is using the following classes:

Class	**Network size**	**First byte**	**First bits**	**Maximum nodes**
A	big	$0-127$	$0xxx$	16 million
B	medium	$128-191$	$10xx$	65535
C	small	$192-223$	$110x$	256
D	IP Multicasting		1110	
E	Experimental		1111	

The Internet address is displayed as a dotted decimal notation, to make it easier for people to read and write. This means that the addresses are expressed as four decimal numbers, each separated by a dot. As the IPv4 consists of 2^{32} bits (4 bytes), this will be four decimal numbers each of eight bits. The conversion is:

	#0			#31
Bits	10010001	00001010	00100010	00000011
Hexadecimal	$0x91$	$0x0a$	$0x22$	$0x03$
Decimal	145	10	34	3

The IP address will be 145.10.34.3. The address ranges for the class A, B, and C with dotted decimal notation will be :

Class	**Start**	**End**
A	$0.xxx.xxx.xxx$	$127.xxx.xxx.xxx$
B	$128.0.xxx.xxx$	$191.255.xxx.xxx$
C	$192.0.0.xxx$	$223.255.255.xxx$

Submask

Submask was added to extend the two level addressing hierarchy. Due to growing of routing tables and administration of network domains the address hierarchy was extended from two levels to three levels addressing hierarchy. The extension is to divide the host number into a sub net number part and a host number part as:

	#1	#2	#3	#4
Two levels	Network number	Host number		
Three levels	Network number	Sub net number part	Host number part	

The "old" host number was divided into a sub net number and a "new" host number. In this way a specific network can be divided into a set of sub networks. The administrator of the network can now divide the network into a set of sub nets using the sub net level. The sub nets are not visible outside the private network of an organization and therefor not included in the routing tables. The introduction of sub nets will therefor not affect the challenges regarding the expanding routing tables. The routing from the Internet to the organization will always be the same, but the routers inside the organization need to differentiate between the individual sub nets.

The address of the sub nets is using an extended network prefix consisting of the network number and the sub net number, identified by the sub net mask. Assuming that you have a class B network of 134.14.0.0 and want to use the third number to represent the the different sub nets, the sub net mask will be 255.255.255.0. The corresponding bits between the IP address and the sub net mask will define the extended network prefix. Assuming a message is sent to IP address 134.14.245.45, giving the following numbers:

IP address	134.14.245.45	10000110	00001110	11110101	00101101
Sub net mask	255.255.255.0	11111111	11111111	11111111	00000000
Network prefix	134.14.245.0	10000110	00001110	11110101	00000000
Host number		00000000	00000000	00000000	00101101

This solution gives any organization with a class B address the options to have 256 sub nets with 256 host numbers in each sub net. Some considerations when designing sub nets are (3COM 2001):

- how many sub nets does the organization need today?
- how many sub nets does the organization need in the future?
- how many hosts are on the organization's largest sub net today?
- how many hosts will there be on the organization's largest sub net in the future?

The first step in the process of defining the sub net mask will be take the maximum number of sub nets required in the future and round up to the nearest power of two. If the organization consists of 5 departments, a normal need will be 5 sub nets. The number of sub nets will rounded up will 8. The second step will be to estimate the maximum number of hosts in the largest sub net. Assume that the maximum number of hosts today is 15, this number be 16. As a system designer you will however always add some room for growth, so you will may be select 32. The product of the sub nets and the maximum number of hosts is:

$$Nodes = 8 \cdot 32 = 256$$

giving that a network of class C can be used. The number of bits for the sub nets will be 3 $\left(8 = 2^3\right)$ and the number of bits for the hosts will be 5 $\left(32 = 2^5\right)$. Assume that your class C internet address will be 193.14.245.x giving the sub nets and sub net mask:

					Sub net number bits	Hosts number bits
IP address	194.14.245.x	11000010	00001110	11110101	xxx	$xxxxx$
Sub net mask	255.255.255.224	11111111	11111111	11111111	111	00000
Department #1	255.255.255.0	11111111	11111111	11111111	000	00000
Department #2	255.255.255.32	11111111	11111111	11111111	001	00000
Department #3	255.255.255.64	11111111	11111111	11111111	010	00000
..	255.255.255.x					
Department #7	255.255.255.192	11111111	11111111	11111111	110	00000
Department #8	255.255.255.224	11111111	11111111	11111111	111	00000

The sub net addresses are 194.14.245.0, 194.14.245.32, 194.14.245.64 and so on, but it will be difficult to see the difference of a sub net address and a IP address. The solution is to append the number of bits in use at the end of the sub net to define the sub net length (sub net mask length). This is defined as:

$$subnetmask \ / \ bits$$

giving the sub net addresses for your system 194.14.245.0/27, 194.14.245.32/27, 194.14.245.64/27 and so on. The /27 indicates that the first 27 bits are used for the sub net mask.

The Internet practices a host number with only zeros identifies the base network number, or the base subnetwork number. A host number with only ones identifies the broadcast address for the network or subnetwork. A broadcast address is an address used to indicate that information being sent out should be delivered to every host number in the network or subnetwork. The broadcast address for each subnetwork will be the address with all the hots number bits set to ones ($x.x.x.255$). A broadcast message is a message that every hosts (nodes) in the network will see. Broadcast messages refers to transmitting a packet that will be received (conceptually) by every device on the network and are used in some situations where:

1. you don't know the address of a specific node. One example can be the address of the gateway, the node for communication outside the local network segment.

2. you want to give the same information to several computers in the network,

Remember that the broadcast message is limited to a broadcast domain (subnetwork) and that the broadcast service is not supported by all network types.

A multicast address is a logical class identifier for a group of hosts in the network, defined as a class D address (224.$x.x.x$: see the class overview table).

Private Internet Address Spaces

The grow of the number of hosts connected to the Internet and the limit numbers of Internet classes lead to a private Internet address space. This private address space should take some of the load of the routing tables and allow organizations to use Internet communication equipment without being connected to the Internet. The connection to the Internet can be solved by using a gateway and only one IP address, assigned by the Internet service provider. This way you can use the private address space without the need to check if any other computer on the Internet is using the same address. The private network address spaces are:

- Class A: 10.0.0.0 to 10.255.255.255

- Class B: 172.16.0.0 to 172.31.255.255

- Class C: 192.168.0.0 to 192.168.255.255

The messages with these addresses will not be copied through routers, so these hosts can not communicate directly with the Internet. A Network Address Translator (NAT) is used to convert the private addresses to an address for use on the Internet. NAT must keep a track of every accesses on the Internet from any local host number and route the answer back to the right host number. Note that all answers will go back to the address of the gateway, and not the address of the host. The gateway and the NAT must route the message from the Internet to the right host on the intranet.

Communication Methods

In a network all local nodes should be able to communicate with each other. There exists several ways of communication methods between nodes in a network and the IP addressing allow for these communication methods:

- Unicast meaning that one node can send a message to another specific node,
- Multicast meaning that one node can send a message to many other nodes,
- Broadcast meaning that one node can send a message to all other nodes,
- Anycast meaning a special case of unicast where only the nearest (physical distance) will respond only. Used mostly in wireless network to save power when transmitting messages,
- Geocast meaning a special case of multicast used by some types of ad-hoc mobile networks.

Loopback

Loopback is normally a way of routing electronic signals, digital data streams, or other flows of items from a source and directly back to the same source without any form of processing. The primarily function of loopback is for testing the transmission or transportation infrastructure.

In TCP/IP a loopback device is a virtual network interface implemented in software only, and not connected to any hardware device. The software interface is fully integrated into the network infrastructure and is used for testing the testing of the network infrastructure of a specific host. The Internet Protocol (IP) specifies the address 127.0.0.1 as the loopback network.

DHCP

Configuration and maintaining a TCP/IP network can be a challenging task, depending on the number of nodes that should be connected in the network. If the number of nodes are fixed, and connected all the time, the configuration can be OK. If the number of nodes is varying, and especially if new nodes need to be add for a period of time, the configuration can be challenging. The challenge will be almost the same for the network at home as for the network for your PCS. There exists a server solution for handling the dynamic assignment of addresses in a TCP/IP network. The TCP/IP address can be assigned manually to each node, or a DHCP, Dynamic Host Configuration Protocol, server can be used for automatic assignment. DHCP can be used for assigning addresses to nodes in the network when a node is logging onto the network. This is software running as a service in the network. In larger networks there can also be several DHCP servers running.

The allocation of a TCP/IP address using the DHCP service is a sequence of messages. A description of the message flow:

1. When a client computer (node) is started or connected to the network, it broadcasts a DHCPDISCOVER message on the local sub net, requesting an answer from a DHCP server. This request includes the hardware address of the requesting client (MAC address).
2. Each DHCP server receiving the broadcast sends a DHCPOFFER message to the client, offering an IP address for a set period of time, known as the lease period.
3. The client receives one or more DHCPOFFER messages from one or more servers and selects one of the offers received. Normally only one DHCP server will be used. If several DHCP servers are used, the client will look for the longest lease period.
4. The client broadcasts a DHCPREQUEST message indicating the client has selected an offered leased IP address and identifies the selected server.

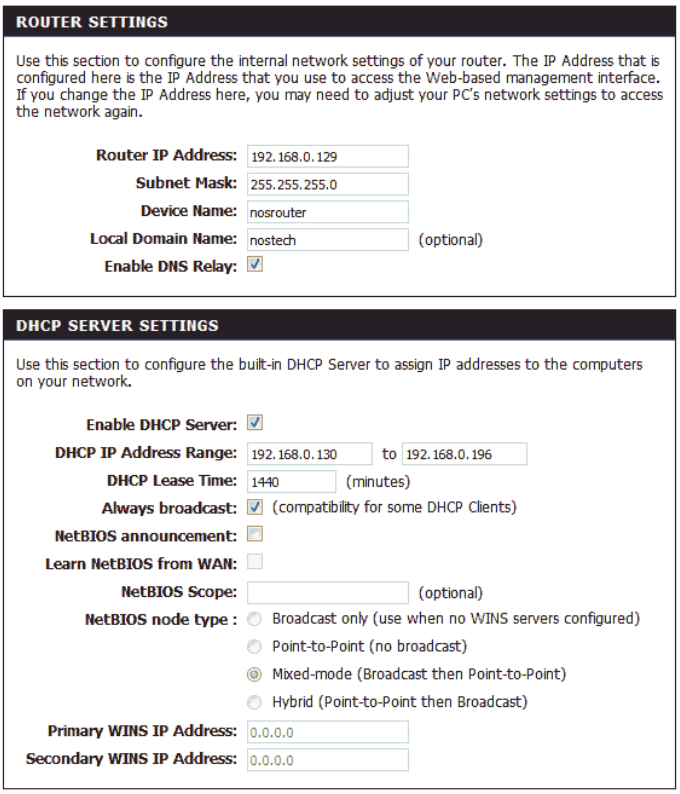


Figure 16.22: The setup of a router with a DHCP server with the private address range of 192.168.0.x. The upper part shows the setup of the router and the lower part shows the setup of the DHCP server.

5. Those servers not selected by the DHCPREQUEST message return the unselected IP addresses to the pool of available addresses.

6. The selected DHCP server sends a DHCPACK acknowledgment that includes configuration information such as the IP address, sub net mask, default gateway, and the lease period.

7. The client receives the DHCPACK message with configuration information. The setup process of the IP address allocation is complete.

DHCP supports both dynamic bindings and static bindings. For dynamic bindings, IP addresses are assigned to clients from a pool of free addresses within the sub net. Static bindings provide configuration information for a specific client and can include one or more fixed IP addresses for the client. Static bindings are used allow a specific host to get the same address every time it is running, and that no other computer in the network can use this address. This can be useful for printer servers and other type of servers in the network. You can configure a DHCP server to include several address pools and static bindings. For any individual client, static bindings take priority over address pools. DHCP will use the MAC address of the host computer for assigning the addresses, and the MAC address must be used when assigning the static bindings of the DHCP server.

Figure 16.22 shows the setup of a router with a DHCP server. The lease time of the DHCP server is the time a host can use a address until the address is returned to the pool of available addresses and the host has to request a new address. A long time means less network traffic of requesting new addresses, but also a longer time an address can be used when a host is turned or removed from the network.

Figure 16.23 shows the fixed binding in the upper part, and the current addresses in use in the lower part. The upper part shows the addresses with fixed binding and the corresponding MAC addresses. These addresses can only be assigned to these MAC addresses. Remember to reconfigure the DHCP server when changing any network hardware of these computers. The lower part shows the hosts that have an address assigned at the moment, and the expire time of these addresses. Notice that the hosts with a fixed binding will never expire. Some of the computers are not connected anymore, but the address stays assigned until the lease time expire.

DHCP RESERVATIONS LIST

Enable	Computer Name	MAC Address	IP Address		
☑	B410-42A1DF	00:80:87:42:a1:df	192.168.0.195		
☑	nosfiles	00:10:5a:14:de:45	192.168.0.194		
☑	noslidev	00:07:95:27:36:60	192.168.0.192		
☑	OKI C5200	00:80:87:64:3c:70	192.168.0.191		
☑	nos_devicemaster	00:c0:4e:12:02:f2	192.168.0.193		

NUMBER OF DYNAMIC DHCP CLIENTS:10

Hardware Address	Assigned IP	Hostname	Expires		
00:1d:7d:d7:d6:a6	192.168.0.141	noshdwdev	20 Hours 35 Minutes	Revoke	Reserve
00:25:d3:86:71:6b	192.168.0.181	skeie	23 Hours 36 Minutes	Revoke	Reserve
00:24:d2:86:60:f4	192.168.0.188	Eva-Toshiba-PC	22 Hours 41 Minutes	Revoke	Reserve
00:1b:77:24:c5:ff	192.168.0.190	fiolvn-laptop	17 Hours 27 Minutes	Revoke	Reserve
00:80:87:64:3c:70	192.168.0.191	OKI C5200	Never		
00:07:95:27:36:60	192.168.0.192	noslidev	Never		
00:c0:4e:12:02:f2	192.168.0.193	nos_devicemaster	Never		
00:10:5a:14:de:45	192.168.0.194	nosfiles	Never		
00:80:87:42:a1:df	192.168.0.195	B410-42A1DF	Never		
00:11:09:d4:58:18	192.168.0.196	noswindev	21 Hours 50 Minutes	Revoke	Reserve

Figure 16.23: The upper part of the Figure shows the fixed binding, the IP addresses that are locked to a fix MAC address. The lower part shows the hosts allocated an address and the expire times of these addresses.

A node running the Windows operating system that will not be able to get a TCP/IP address from the DHCP server, will assign itself a TCP/IP address in the $169.254.0.0 - 16$ range.

IPv6

The standard IP protocol (IPv4) supports only 2^{32} addresses (about 4.3 billion) which is not enough addresses for the growing amount of people and hosts. The wireless area and the Internet of Things (IoT) area are pushing these limits as well. The solution is to extended the IP protocol. Version 6 (IPv6) supports 2^{128} addresses being more then enough for the almost 7 billion people alive today together with all the things connected to the Internet. An overview of the header of each protocol is shown in Figure 16.24 showing the different fields and sizes of the header. The only fields that are of interest is the source and destination addresses, showing a larger header in IPv6 to support more addresses. The preferred form for the IPv6 address is hexadecimal values with a colon between group of four digit hexadecimal integers (16 *bits*). This way it is easy to see the difference between an IPv4 and IPv6 address. Figure 16.24 also shows that an IPv4 protocol can be used inside an IPv6 protocol, but an IPv6 can not be used in an IPv4 protocol. The IPv4 address of 192.168.0.129 will be converted to the IPv6 address of $0:0:0:0:0:0:0:0:192.168.0.129$. Note that a colon indicates a hexadecimal value, while a comma indicates a decimal value. The IPv4 can of course be converted to an network number as long as the IPv4 remains as the lowest bytes. A compressed form of IPv6 substitutes zero strings with double colons (::) to compress a complete 16 bit group. The IPv6 address above will in the compressed form be $::::::::192.168.0.129$.

There exists several differences between the IPv4 and the IPv6 protocols, but will not be part of this document.

16.7.2 Programming

TCP/IP consists of two different protocols, TCP and UDP. Both of these protocols consists of the server/client model where the client sends a request to the server and the server reply with the requested data. So any software requires both the server program and the client program.

The Transmission Control Protocol (TCP) protocol requires a specific IP address and a specific port number so only one node will receive such a request. The port number is a number between 1 and 65535, how ever port numbers below 1024 are often reserved for specific purposes. The server will make a socket connection and any client can connect to this socket for communication with the server. The server will define the maximum number of clients that can connect to the socket at the same time and must also handle these simultaneous client connections. The TCP connection is a secure connection as

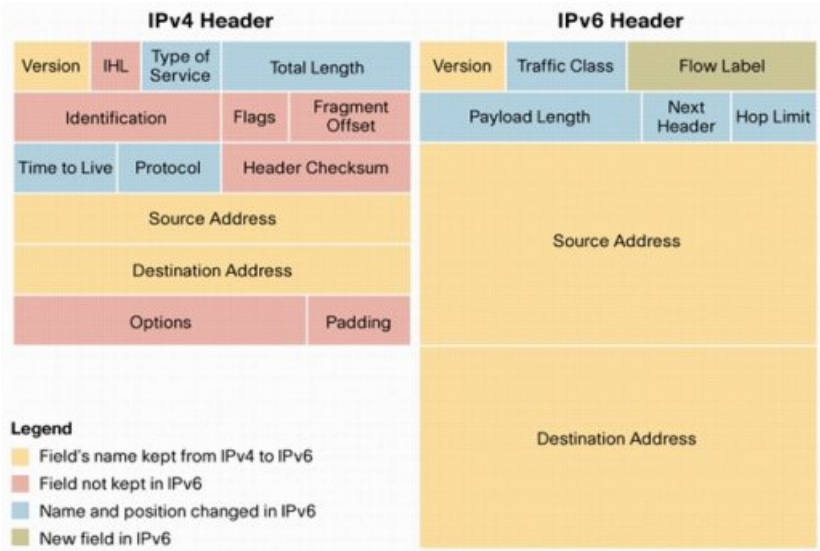


Figure 16.24: Comparison of the IPv4 header and the IPv6 header. The IPv4 consists of 12 bytes ($2x6bytes$) and the IPv6 consists of 20 bytes ($2x10$ bytes). The first bits are used for the version number. (www.cisco.com: Dec-10).

all the message packages are guaranteed to arrive at the recipient, otherwise there will be an error. The TCP protocol is used in C# examples later in this document.

The User Datagram Protocol (UDP) protocol can be used with broadcast packages, but the broadcast packages will not be sent to other network segment, only to nodes in the same network segment. An UDP connection is not a secure connection as there is not guarantee that a data package will arrive at the recipient, but for the same reason a faster protocol than TCP. So UDP can be used instead of TCP if speed is more important then loss of packages.

The pseudo coding for TCP/IP programming (both TCP and UDP):

Server:

```
ServerSetup(iServerIpAdress, iSocketNumber);
while (true)
{
    iIpAdress = WaitForClientConnection();
    MakeClientConnection(iIpAdress, iSocketNumber);
    while (ClientConnected() == true)
    {
        bData = HandleClientrequest();
        SendClientAnswer(bData);
    }
    DisconnectSocket();
}
```

Client:

```
ClientSetup(iClientIpAdress, iSocketNumber);
if (ServerConnection(iServerIpAdress, iSocketNumber) == true)
{
    while (bServerConnection == true)
    {
        bData = MakeRequest();
        SendRequest(bData);
        bData = HandleServerData(bData);
```

```
        bServerConnection = CheckMoreCommunication();
    }
    DisconnectServer();
}
```

Note that the TCP/IP protocols is just handling a data buffer (bData in pseudo coding), no description of the content or structure of this buffer. This is the challenge of any TCP/IP protocol, needs another protocol to define the structure of this buffer.

16.7.3 Routing

A router is used for connecting two or several different networks, and is working on layer 3 in the OSI model. Routing is the process of selecting paths for transmission of data packets through a network. The process will always try to select the best path, fastest and/or shortest, and is based on large tables. The transmission is achieved by using different routing protocols. The routing protocol specify how routers communicate with each other, exchange information to be able to select routes between the networks, and the selection of a route. Different routing algorithms are used for different protocols. A router needs a protocol to get information about the topology of the network. A router is using static and/or dynamic routing to define how the messages should be sent from one network to another network. Dynamic routing can among others be RIP (routing information protocol) or OSPF (Open Shortest Path First).

All routers will use tables for the routing information. Static routing algorithms are using fixed tables configured by the administrator of the network. Static routing will only be possible for small networks with predictable traffic, dynamic routing is a must in larger networks. The dynamic routing is using the information in the messages to analyze, build and/or update the routing tables, and will be scalable. The routing tables will grow when an external network is connected and the memory size of these routers will be important. The routing process is inspecting the header of the IP packets to get the addresses, making or updating the routing table and using the routing table to select the best path for sending any new message. The best path may change from one message to the next message depending on several factors like traffic load, speed, etc.

Important tasks for the router are security and traffic load. The industrial network is normally without any security strategies and should give access only to authorized users from the office network. The router will give access only to authorized users and let these users access only specified components in the industrial network, restricting both the number of users and the traffic.

A challenge caused by the growth of internet addresses is the size of routing tables. The number of routing tables will grow with the number of routers and the size of these tables will grow with the number devices with an IP address. At the end of 1990 there were about 2200 routers, at the end of 1995 there were more than 30000 routers, and at the end of 2000 there were more than 100000 routers. Each router will need a routing table in memory, and enough network information to keep the table updated.

The routing problem can not be solved by simply installing more memory in the routers, and increasing the size of the routing tables. More CPU power will be needed to compute the routing tables from the topology, and more network status information is needed to keep track of connections managed by people and machines. The size of the tables has to be without boundaries, otherwise limitations will force routers to drop part of router tables and portions of the Internet will become unreachable.

Services that are used by routing system are:

- NAT (Network Address Translation), translating many private IP addresses into one public IP address,
- DHCP (Dynamic Host Control Protocol), making network configuration changes easier,
- DNS (Domain Name Server), connections between a domain name and an IP address,
- VPN (Virtual Private Network), making a secure connection between two secure networks,
- SNMP (Simple Network Management Protocol), a OSI level 7 service for network management.

DNS (Domain Name Server)

All nodes on internet has an IP address and is using the IP addresses for communication. However, the IP addresses has a data format that is not easy for human to remember. DNS (Domain Name Server) is a

distributed database for adding node names in addition to the IP addresses of the nodes. This database contains several information about each node and the structure is a set of data type codes with a test string belonging to each data type codes. A DNS is structured in domains (zones) and around the world there exists a set of root servers (13 servers in 2006) keeping track of the high level domains around the world. The structure of the DNS uses a hierarchical tree based name structure and at top of the tree there is the "root" followed by the Top Level Domain (TLD) and followed by the domain-name and any number of lower level sub-domains, all separated by a dot. The TLD are divided into the categories: 1) Generic TLD (gTLD) and 2) Country Code TLD (ccTLD).

Some of the common gTLD are:

- .com – commercial web sites,
- .org – non profit organizations web sites,
- .edu – restricted to schools and institutions,
- .net – originally for network infrastructures, now unrestricted.

The ccTLD are country codes like 1) .us - United States, 2) .no - Norway, 3) .lk - Sri Lanka, and 4) .np - Nepal to mention some.

All DNS are interconnected meaning that a web address like www.xxxx.yy can be used instead of the IP address for the same node. The web client will ask the DNS for the IP address and send the web request to correct IP address.

16.7.4 Security

The TCP/IP protocol consists of the IP protocol for layer three and either TCP or UDP protocol for layer four in the OSI model. The protocol consists of the type , TCP or UDP, and a port number, 1 to 65535. Any communication using the TCP/IP protocol, used for some type of intra net or internet communications, must specify a protocol type and port number. Some of the port numbers are:

Type			**Port**
FTP	File Transfer Protocol	TCP	20, 21
SSH	Secure SHell	TCP	22
HTTP	HyperText Transfer Protocol (web browsing ..)	TCP	80
MSN	Messenger	TCP	1863

A computer with a TCP/IP protocol will have 65535 ports for both TCP and UDP, a total number of 131070 ports. Each port will be a communication channel that any remote computer can connect. This will be like a house with 131070 doors and all doors should be secured.

16.7.5 System

Internet as a system consists of interconnected computer networks using the TCP/IP protocol. Internet is the global system of all these interconnected computer networks serving all the users connected to these networks. Each computer must have a specific TCP/IP address in this network. Routers, gateways, and firewalls are used to connect these networks together, and the routers can map locale TCP/IP addresses to global TCP/IP addresses based on services from internet service providers. The TCP/IP message contains the address of the transmitter and the receiver giving every routers in the path the information where to send the message. Figure 16.25 shows a part of internet with a set of interconnected networks.

The user request some sort of action by input of an address, either a specific TCP/IP address or a WEB address that will be converted to a specific TCP/IP address. The action will be sent to the server, performed, and the response will be sent back to the user. All the information about where to send the message is contained in the header of the TCP/IP message. Any action like surfing on the internet, downloading a file etc., one or several messages will be sent on internet. These messages:

- will be observed at any routers along the path between the server and the client,
- will be observed by the internet service provider, logging your actions on internet,

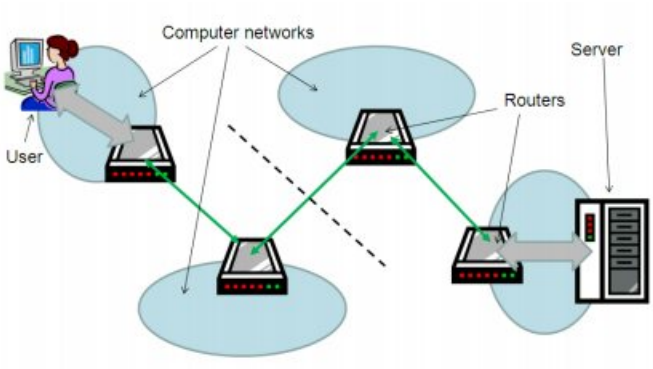


Figure 16.25: The internet is consisting of interconnected networks using the TCP/IP protocol.

- will be observe by the server having information about your network address,
- will be observed by every router along the path that will know the network address of the sender and the receiver,
- can be encrypted, but every router still knows who you are communicating with, how you are communicating, and how much data you are exchanging.

The information about these messages will be logged and can be there forever.

16.8 Firewalls

Firewalls are important for the network security, as the antivirus software is important for the computer security. A firewall controls the traffic flow between a trusted network (LAN) and an untrusted or public network (like Internet) (Miller 2011). A firewall is a collection of componetens placed between two networks with the following properties (Cheswick, Bellovin & Rubin 2003): 1) all trafic between the networks must pass through the firewall, 2) only authorized trafic, defined by the local security policy, are allowed to pass and 3) the firewall itself is immune to penetration.

The basic type of firewalls are based on filtering messages using ports or packets. Firewalls based on port filtering use the source and destination IP addresses in the TCP/IP message header and the TCP/UDP port information to validate the exchange of information between the trusted and not trusted networks.

When a node in the trusted network starts a session with a node in the not trusted network, a connected is created and the firewall will create a temporary rule to allow communication for this session. Any session started from a node in the not trusted network needs an already created rule in the firewall to allow for a connection.

Firewalls are based on rules and as long as all nodes and applications plays by the rules, everything works ok with these basic type of firewalls. The challenge is the nodes not playing by these rules, as the rules are only guidelines. The biggest challenge in communication between trusted and not trusted networks is the exposure of sensitive or private data. To deal with this challenge any organization should (Miller 2011):

1. gain control over the applications using the firewalls,
2. scan the applications they want to use the firewalls, for sensitive or private data,
3. understand the process of using these applications (why using and type of transactions).

More advanced firewalls will check the behavior of the applications using the firewalls. Applications can change the “standard” behavior like (Miller 2011):

- port hopping; changing randomly the ports/protocols in a session,
- usage for non-standard ports,
- tunneling within commonly used services,

- hiding within SSL encryption (like the TCP port 443 used for HTTPS).

Advanced firewalls should in addition to filter messages based on ports and packets, also identify the applications sending the messages, the users using these applications and the content of the messages. The first step of identification of an application is the usage of port(s) and protocol(s). The protocol(s) should also be decrypted and decoded to figure out the content of the message. Is it the "real" message or just used as a tunnel for hiding the "real" application. The firewall could also have a type of application "signature" to correctly identify the unique properties and transaction characteristics of the application (Miller 2011). The firewall should also be able to identify the user running the application and the content of the messages for the application. These firewalls should also handle mobile and remote users.

An advanced firewall should (Miller 2011):

1. identify the applications, not the ports in use,
2. identify the users, not the IP addresses,
3. identify the content of the messages, not only the packets.

16.9 Time Synchronization

16.9.1 Time Time Protocol (NTP)

The protocol Network Time Protocol (NTP) is a standard protocol for synchronization of computer clocks in a network. Several methods exist for synchronization in network where accuracy and precision are the differences. NTP is based on a server and a set of clients where the clients will adjust the time to the server.

When the client sends a request to the server, there will be some time delay before the client will receive the answer. The client will ask the server about the time at t_1, the server will receive the request at t_2, the server will send the answer at t_3, and the client will receive the answer at t_4.The offset O will be:

$$O = \frac{((t_2 - t_1) + (t_3 - t_4))}{2}$$

The client can then use the time from the server plus the offset as the correct time. How to adjust the time on the client? If the deviation is large, this has to be done in smaller step to destroy any type of logging. Should the logging be according to a relative time or an absolute time? Relative time means that all computers in the network are synchronized, but they do not need to be synchronized with other networks. Absolute time means that all computers in the network are synchronized with the rest of the world. The usual reference for absolute time is Coordinated Universal Time (UTC).

Several NTP servers can be used as a redundant time synchronization. The client can then use an average of the times from the servers. The accuracy will depend on the transport time from the server to the client, using many servers can make several routes giving different timing on the requests.

Time synchronization is important for all type of time stamping and/or logging. Any change of time will make unwanted steps in the time stampling or logging. What happens if the time has to be changed backwards?

NTP messages are using the port 123 in the firewall. Using NTP means that this port has to be open, and the port can be used by hackers to access the computer.

A NTP client uses an algorithm to make an average from several servers to estimate the correct time. The Precision Time Protocol (PTP) (IEEE1588) is using a different approach, a more advanced algorithm called the Best Master Clock (BMC) strategy if several clients are available. The PTP is also consider the "offset" in the infrastructure of the system having a more accurate time, as long as the devices giving delays are supporting PTP. PTP is based on a separate hardware module used for time stamps giving that PTP is also much more expensive as NTP. The accuracy of the time stamps decide the protocol, an accuracy of several seconds is supported by NTP, an accuracy of better than one second is supported by PTP.

16.9.2 Time Sensitive Networking

In an Ethernet network some messages need to be delivered faster, or within a maximum time. This is not possible in a standard Ethernet network, but Time Sensitive Networking (TSN) standards has

been defined. These standards contain three components required for time based communication, and are extensions to the Ethernet standards. In real-time systems a maximum communication time is important so TSN will be an important extension for communication systems in PCS. The three basic components for nodes depending on maximum time types of communication are:

- Time synchronization: All nodes need to have a common understanding of time,
- Scheduling and traffic shaping: All nodes need to adhere to the same rules in processing and forwarding communication packets,
- Selection of communication paths, path reservations and fault-tolerance: All nodes need to adhere to the same rules in selecting communication paths, in reserving bandwidth, time slots, and possibly utilizing simultaneous paths to achieve fault-tolerance communication.

16.10 Email system

An electronic mail system, or email, is part of a network used mainly for communication between humans. Emails can also be used by PCS for automatic generation of information that can be sent to humans. An email can be looked upon as a communication protocol between computer systems and humans, or between humans. The first email was sent in 1971 and in 2015 the number was about 205 billion emails every day, 112 billion emails related to work, the rest is not work related. In 2015 the average received number of emails for an user was 122 and the average number of transmitted emails was 34.

The email system consists of several email servers exchanging emails and a set of email clients. A user is using an email client to connect to a specific email server connected to a domain. Your email address will indicate your email server as the email address is:

name@domain.type

where name is the user name, domain is network domain, and type is normally a country code, just like the node internet address. The protocols used for email communications between the email servers, or between the email server and email clients are:

- IMAP (Internet Message Access Protocol); used between the server and clients, a protocol for connected with limit data traffic, download only the headers, must request a specific email to download all the information, supports also management of emails and folders on the server,
- POP3 (Post Office Protocol 3); used between the server and clients, a protocol for downloading all the messages with the contents, an option to leave a copy of the email on the email server,
- SMTP (Simple Mail Transfer Protocol); used between servers, and between the server and clients, a protocol that can only be used for sending emails, not to receive them. If you are using the SMTP for sending emails from the email client to the email server, you need to use another protocol for receiving the emails. This is part of the configuration of the protocol settings for incoming and outgoing mail servers for the email client.

16.11 Other network protocols

Some of the protocols used in a process network:

- SNMP; Simple Network Management Protocol is an Internet-standard protocol for managing devices on IP networks (www.wikipedia.org 2010). SNMP operates in the Application Layer of the Internet Protocol Suite (Layer 7 of the OSI model) using the UDP protocol. The protocol can be used for communication with devices like printers, UPS, routers, and gateways to mention some. This protocol is useful to both IT professionals and plant operators for managing the network infrastructure and diagnosing problems.
- RSTP; Rapid Spanning Tree Protocol is used for checking the routing, prevent looping bridges, and broadcast control.
- LLDP; Link Layer Discovery Protocol is a Link Layer protocol in the Internet Protocol Suite used by network devices for advertising their identity, capabilities, and neighbors on a IEEE 802 local area network (www.wikipedia.org 2010).

16.12 Virtual LAN

A Virtual LAN has the same attributes as a physical LAN, but it allows for end stations to be grouped together even if they are not located on the same network switch. Reasons can be security, redundancy, and/or reliability.

16.13 VPN

A virtual private network (VPN) is a computer network in which some of the links between nodes are carried by open connections or virtual circuits in some larger networks, such as the Internet, as opposed to running across a single private network (www.wikipedia.org 2006). VPN can also be a "tunnel" trough an existing network with a cryptic connection, a secure connection between two end points in a network. VPN will then depend on the protocol used and the type of cryptic algorithms used.

16.14 Proxy Server

A proxy server is a type of "virtual" server that acts as an intermediary server for requests from clients (www.wikipedia.org 2010). The clients will request information from a server, a proxy server will be able to check these requests and either answer directly or feeding the requests further to the real server. Proxy servers can be used for:

- filtering purpose; filter the traffic, if a valid request the proxy server will connect to the relevant server and requesting the information on behalf of the client,
- balancing the workload; the proxy server can "cache" the traffic and be able to answer the client without connecting the relevant server,
- simulation; simulating servers for testing purposes.

Chapter 17

Physical layer

Communication can be either of parallel form, sending a set of data bits in each "time slot", or serial form, sending a data bit in each "time slot". A "time slot" here means the time for sending the minimum number of information (bit). Receiving of data is an external event, the remote computer don't know when the next information will be received. Synchronization is therefore necessary as well, both for serial and parallel forms. See Figure 17.1 for the difference of parallel and serial forms. Parallel is faster, but require more "channels". A "channel" will be the location for transmission of one bit of data in one time slot.

The physical layer can be divided into:

1. Electrical data transmission; serial or parallel,
2. Optical data transmission; serial (or parallel),
3. Radio data transmission; serial (or parallel).

Optical and radio data transmission will be of serial form mainly, while electrical data transmission can be divided into both parallel form and serial form.

17.1 Parallel communications

In parallel form a set of data bits is transferred together, meaning that it has to be a separate physical connection for each data bit. The number of data bits is often a multiple of 8 bits. In addition some sort of synchronization is needed, solved using an extra strobe signal. The strobe signal can be changing from high to low, or low to high, telling the remote system that data is available on the communication lines. An example is shown in Figure 17.2, shown a parallel layer consisting of 8 data bits and one strobe bit. The data bits are 10010010 ($0x92$) and the strobe is low to high.

The data bits are digital information limiting the length of the physical connection and the communication speed. The advantage is faster data transfer; the disadvantage is the length of the physical connection. Also note that the transfer direction is unidirectional, for bidirectional transfer direction more physical connections are needed, either more data bits or more control signals.

17.1.1 Centronics

A Centronics parallel port is a type of socket found on personal computers for interfacing with various peripherals. It is also known as a printer port or only parallel port. The IEEE 1284 standard defines the bidirectional version of the port. The IEEE 1284 standard allows for faster throughput and bidirectional data flow with a theoretical maximum throughput of 4 megabits per second, with actual throughput around 2 megabits, depending on hardware. The maximum distance is 2 m or 10 m depending on the cable. The connectors are DB25 types or Centronics types, the number of bits are 8, and the direction is half-duplex.

The Centronics port allow only for one device to be connected to parallel port, and an example of a printer connected is shown in Figure 17.3.

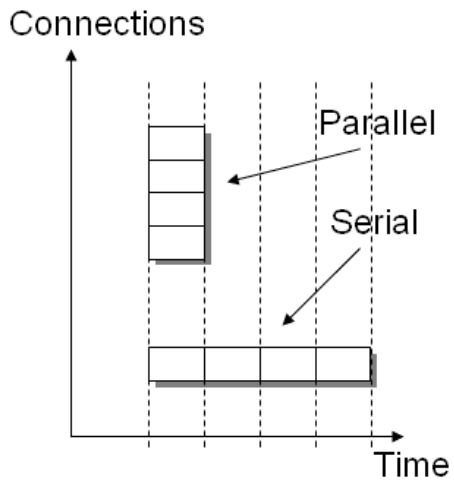


Figure 17.1: The time and connections for parallel and serial form of communication.

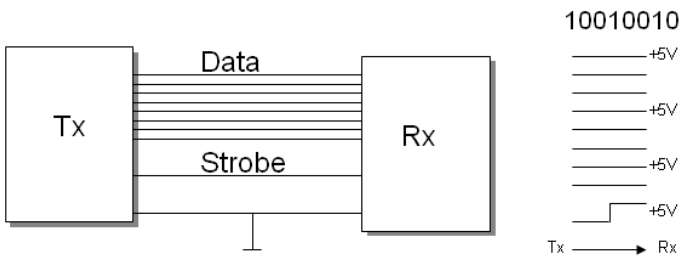


Figure 17.2: A parallel pysical layer of 8 databits and one strobe bit.

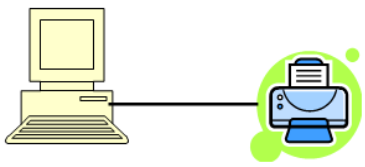


Figure 17.3: Printer connected using the Centronics parallel port.

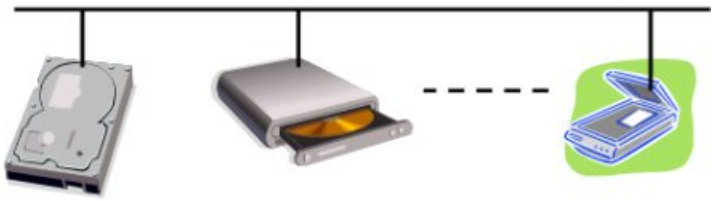


Figure 17.4: Devices connected to the SCSI bus.

Interface	Width (bits)	Clock	Maximum		
			Throughput	Length	Devices
SCSI	8	5 MHz	5 MB/s	6 m	7
Fast SCSI	8	10 MHz	10 MB/s	1.5-3 m	7
Fast-Wide SCSI	16	10 MHz	20 MB/s	1.5-3 m	15
Ultra SCSI	8	20 MHz	20 MB/s	1.5-3 m	7
Ultra Wide SCSI	16	20 MHz	40 MB/s	1.5-3 m	15
Ultra2 SCSI	8	40 MHz	40 MB/s	12 m	7
Ultra2 Wide SCSI	16	40 MHz	80 MB/s	12 m	15
Ultra3 SCSI	16	40 MHz	160 MB/s	12 m	15
Ultra-320 SCSI	16	80 MHz	320 MB/s	12 m	15

Figure 17.5: Some of the SCSI standards.

17.1.2 SCSI

Small Computer System Interface (SCSI) is another parallel protocol standard used a lot in computers, lately especially in servers. The parallel communication is a physical layer for making connections between computers and peripheral devices. The SCSI protocol define the commands and the electrical interfaces. The SCSI standard is a complex specification with maximum 8 or 16 devices that can be connected. The SCSI standard was standardized in 1986, but due to the complex structure, a simpler (low cost) interface was wanted for low priced PC. The result was the ATA specification in the beginning of the 1990s.

The SCSI bus allow many devices as shown in Figure 17.4.

Standards

Parallel SCSI consists of a lot of standards, and some them are listed in Figure 17.5 showing the data bits, speed, and maximum devices connected. The number of physical connections is 50 or 68 (pins cable).

Note that the SCSI bus must be terminated at each end of the bus using a combination of 220 Ohms and 330 Ohms resistors on all lines. Normally these termination resistors are included in the SCSI devices and can be activated using a jumper.

SCSI command protocol

In addition to many different hardware implementations, the SCSI standards also include a complex set of command protocol definitions. The SCSI command architecture was originally defined for parallel SCSI buses but has been carried forward with minimal change for use with iSCSI and serial SCSI.

In SCSI terminology, communication takes place between an initiator and a target. The initiator sends a command to the target which then responds. SCSI commands are sent in a Command Descriptor Block (CDB). The CDB consists of a one byte operation code followed by five or more bytes containing command-specific parameters.

At the end of the command sequence the target returns a Status Code byte which is usually 00h for success, 02h for an error (called a Check Condition), or 08h for busy. When the target returns a Check Condition in response to a command, the initiator usually then issues a SCSI Request Sense command in order to obtain a Key Code Qualifier (KCQ) from the target. The Check Condition and Request Sense sequence involves a special SCSI protocol called a Contingent Allegiance Condition.

There are 4 categories of SCSI commands: N (non-data), W (writing data from initiator to target), R (reading data), and B (bidirectional) and there are about 60 different SCSI commands in total. Some of the SCSI commands can take some time, and the target can release the SCSI bus to let other devices use the bus until the command is finished and an answer can be sent back to the initiator.

SCSI device identification

In the modern SCSI transport protocols; there is an automated process of "discovery" of the IDs. On a parallel SCSI bus, a device (e.g. host adapter, disk drive) is identified by a "SCSI ID", which is a number in the range 0-7 on a narrow bus and in the range 0–15 on a wide bus. You usually set the SCSI ID of the initiator (host adapter) with a physical jumper or switch on early models. On modern (since about 1997) host adapters, you set the SCSI ID by doing I/O to the adapter; for example, the adapter often contains a BIOS program that runs when the computer boots up and that program has menus that let you choose the SCSI ID of the host adapter. Or the host adapter may come with software you can install on the computer to do this. The traditional SCSI ID for a host adapter is 7, as that ID has the highest priority during bus arbitration (even on a 16 bit bus). This gives the maximum number of devices to be 7 for a “narrow bus” and 15 devices for a “wide bus”.

The complexity of the command sequences results in a “simpler” bus for the personal computers in the beginning of the 1990s, called the ATA (or IDE).

17.1.3 IDE / ATA

AT Attachment (ATA) or Integrated Drive Electronics (IDE) is a standard parallel interface for connecting storage devices such as hard disks and CD-ROM drives inside personal computers (IBM AT), other terms like IDE, ATAPI, and UDMA are also used. ATA standards only allow cable lengths up to 46 centimeters (18 inches) (although cables up to 90 cm (36 inches) can be purchased), and the data bits are 8 or 16.

The ATA standard allow for only two devices to be connected to each ATA port, and normally a PC has two ATA ports allowing for maximum four ATA (or IDE) devices to be connected. The ATA port has 40 connections (pins) showing in Figure 17.6, and pins 33 and 35 are the limitation for the number of devices connected.

ATA’s ribbon cables had 40 wires for most of its history, but a 80-wire version appeared with the introduction of the Ultra DMA/66 (UDMA4) mode. All of the additional wires in the new cable are ground wires, interleaved with the previously defined wires. The interleaved ground wire reduces the effects of capacitive coupling between neighboring signal wires, thereby reducing crosstalk. Capacitive coupling is more of a problem at higher transfer rates, and this change was necessary to enable the 66 megabytes per second (MB/s) transfer rate of UDMA4 to work reliably. The faster UDMA5 and UDMA6 modes also require 80-conductor cables.

Though the number of wires doubled, the number of connector pins and the pinout remain the same as on 40-conductor cables, and the external appearance of the connectors is identical. Internally, of course, the connectors are different: The connectors for the 80-wire cable connect a larger number of ground wires to a smaller number of ground pins, while the connectors for the 40-wire cable connect ground wires to ground pins one-for-one. 80-wire cables usually come with three differently colored connectors (blue, gray & black) as opposed to uniformly colored 40-wire cable’s connectors (all black). The gray connector has pin 28 CSEL not connected; this makes it the slave position for drives configured cable select.

The ATA bus can connect two slave devices and shown in Figure 17.7, a disk and a DVD connected to the bus.

Communication

The communication is command based; the host is sending a message to the ATA device containing a command like read data, write data, get status and so on. The requested ATA device then performs the command and sent a status command back to the host.

Parallel ATA Pins

Pin	Function	Pin	Function
1	Reset	2	Ground
3	Data 7	4	Data 8
5	Data 6	6	Data 9
7	Data 5	8	Data 10
9	Data 4	10	Data 11
11	Data 3	12	Data 12
13	Data 2	14	Data 13
15	Data 1	16	Data 14
17	Data 0	18	Data 15
19	Ground	20	Key
21	DDRQ	22	Ground
23	I/O Write	24	Ground
25	I/O Read	26	Ground
27	IOC HRDY	28	Cable Select
29	DDACK	30	Ground
31	IRQ	32	No Connect
33	Addr 1	34	GPIO_DMA66_Detect
35	Addr 0	36	Addr 2
37	Chip Select 1P	38	Chip Select 3P
39	Activity	40	Ground

Figure 17.6: The ATA connection pins.

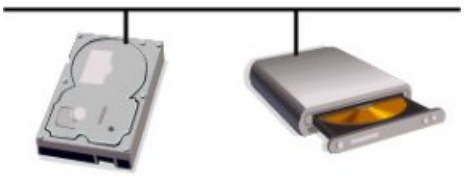


Figure 17.7: Two ATA (IDE) devices connected to the ATA bus, a disk and a DVD.

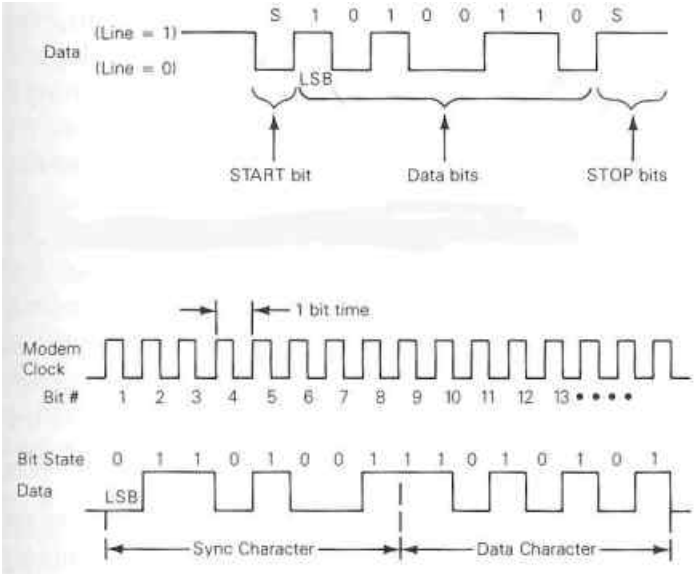


Figure 17.8: Asyncron communication on the top of the figure, and syncron communication at the bottom of the figure (McNamara 1988).

Master and slave

In the ATA specifications the two devices are referred to as device 0 (master) and device 1 (slave), respectively. However it is the host device controlling the arbitration of the devices, so both devices are really only a slave.

Transfer speeds

The transfer speed depends on the different ATA standards and cables, in the range from 3.3MB/s up to 133MB/s.

PATA

With the market introduction of Serial ATA in 2003, the original ATA was renamed Parallel ATA (PATA).

17.2 Serial communication

Only one physical connection can be used and the data bit can be sent in serial, in stead of using several physical connections.

17.2.1 Asynchronous and synchronous communication

Asynchronous serial communication describes an asynchronous transmission protocol in which a start signal is sent prior to each byte, character or code word and a stop signal is sent after each code word. The start signal serves to prepare the receiving mechanism for the reception and registration of a symbol and the stop signal serves to bring the receiving mechanism to rest in preparation for the reception of the next symbol (www.wikipedia.org 2006).

Sending a lot of data, the extra bits will limit the bit rate, synchronous communication can be used instead. In synchronous communication an extra physical connection is needed, as the connection is using an extra hardware connection for synchronization. The communication is using the extra hardware synchronization, so no start and stop bit is needed. Even if hardware synchronization is used, a synchronization character is needed before the rest of the data is transmitted. However all the characters can be transmitted without any extra bits for each character. Asynchronous and synchronous communication is shown in Figure 17.8.

Figure 17.9: Asyncron serial communication.

On top the figure asynchronous communication is shown with the start and stop bits for every characters. At the bottom synchronous communication is shown with the extra hardware synchronization signal (clock signal). The first byte is used for synchronization, however no start or stop bits are added to any of the characters. So using a synchronous communication another connection is needed for synchronization, and full duplex communication needs at least four physical connections (plus grounding).

17.2.2 Asynchronous communication

Asynchronous communication is probably most used today to transfer a small amount of data. The remote system doesn't know when the data will be sent, so some sort of synchronization is still needed. Both hardware and software synchronization can be used, and probably most used is software synchronization communication with a start and stop bit together with the data bits, shown in Figure 17.9. The synchronization is also called asynchronous communication.

The start bit is "0" and the length of each bit is according to the speed or data rate that all systems connected to the physical layer must agree on. The data rate is defined in bits per. second and can among others be 1200, 9600, and 115200 bits per second. The sequence is always starting with a start bit, being a low signal. This also means that the signal is high when no data is transmitted. The start bit is for synchronization of the transmitter and receiver, and then data bits are transmitted, starting with the least significant bit (LSB). Then all data bits are transmitted, normally 7 or 8, all systems connected must agree on the number of data bits as well. Serial communication can be used over long distances, including distances with noise, so a parity bit can be added. The parity bit is optional, used as an extra error check of the data received. The parity bit can be none, even or odd, and all systems connected must agree on the parity settings. An even parity means that the number of high data bits including the parity bit must be a even number. The stop bit ends the communication and the number of stop bits can be 1, 1.5, or 2.

Communication parameters:

Type	**Meaning**	**Normal range**
Baud	The data rate (actually the number of symbols)	300 to 115200
data bits	The number of data bits (LSB to MSB)	5 to 8
Parity bit	The usage of the parity bit	None, Odd, Even
Stop bits	The number of stop bits	1, 1.5 or 2

Serial communication can be bidirectional using only 3 physical connection, transmit, receive and ground. Hardware circuits exist making the conversion from a byte to the serial communication and from the serial communication back to the byte. These circuits are called UART, Universal Asynchronous Receiver and Transmitter, doing all the logic, control and communication.

The synchronization between the two computers is solved using an internal clock of 16 times the bit rate, synchronize this internal clock with the start bit, and using the signal level in the middle of each bit. The middle of each bit will then be after 8 cycles of the internal clock.

The speed is defined as baud or data bit per seconds (bit/s). The serial data communication is adding a set of control bits so the baud rate will be higher than the actual data rate.

Parity can be added to make the communication more immune to errors. None parity means no extra bit information in this location, even parity means that the number of "1" shall be a even number. Odd parity means that the number of "1" shall be an odd number. The number of bits is including the parity bit.

Transmitting the ASCII character **A** ($0x41$) and **a** ($0x61$) using different types of parity is according to the table shown below:

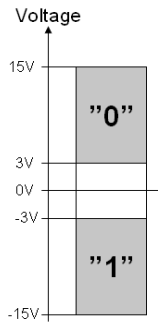


Figure 17.10: The voltage ranges for the RS-232C protocol.

A (LSB:MSB)	Parity	Transmitting	a (LSB:MSB)	Parity	Transmitting
1000 0010	EVEN	1000 00100	1000 0110	EVEN	1000 01101
1000 0010	ODD	1000 00101	1000 0110	ODD	1000 01100
1000 0010	NONE	1000 0010	1000 0110	NONE	1000 0110

17.2.3 Asynchronous communication protocols

The most used protocols for the physical layer of the OSI model is RS-232C, RS-422. and RS-485. These are protocols for transferring digital informations, on long distances the data has to modulated to an analog signal and demodulated back to a digital signal using a modem.

RS-232C

In telecommunications, RS-232 is a standard for serial data interconnection between a DTE (Data terminal equipment) and a DCE (Data Communication Equipment). It is commonly used in computer serial ports (Gofton 1986). The DTE is often a computer system. The RS-232 protocol specification was first released in 1960, however the specification has been revised several times as RS-232A, RS-232B and now RS-232C (Levermore 2000). The RS-232C is one of the most used connection for asynchronous communication, but has the limitation that the communication can only be between 2 systems. Physical interface is a 25 pins or 9 pins D-sub connector with the following pining:

Signal	Description	Pin id (25pins)	Pin id (9pins)	From	To
TD	Transmit data	2	3	DTE	DCE
RD	Receive data	3	2	DCE	DTE
GND	Ground	7	5		
RTS	Request To send	4	4	DTE	DCE
CTS	Clear To Send	5	5	DCE	DTE
DSR	Data Set Ready	6	6	DCE	
DCD	Data Carrier Detect	8	1	DCE	
DTR	Data Terminal Ready	20	4	DTE	

Physical connection The RS-232C specification define the signal on physical layer to be between $-15V$ and $-3V$ to be "1" and between $+15V$ and $+3V$ to be "0". See Figure 17.10.

Note however that the control signals have a positive voltage $[3\,\mathrm{V} - 15\,\mathrm{V}]$ when turned on and a negative voltage when turned off.

Cables RS-232C cables are often a mess due to the DTE and DCE equipment, and the number of connection used. Often only TxD and RxD are used (together with GND). Connecting 2 DTE devices, then the cable must emulate the remote DTE as a DTE (null modem).

The reason that a minimal two-way interface can be created with only 3 wires is that all the RS-232 signals share a common ground return. The use of unbalanced circuits makes RS-232 susceptible to problems due to ground potential shifts between the two devices. RS-232 also has relatively poor control of signal rise and fall times, leading to potential crosstalk problems. RS-232 was recommended for short connections (15 meters or less), however the limit is actually defined by total capacitance and low capacitance cables allow reliable communications over longer distances exceeding 50 m. RS-232 interface cables are not usually constructed with twisted pair because of the unbalanced circuits (Campbell 1984).

The connections are:

1. Transmitted Data (TxD): Data sent from DTE to DCE,
2. Received Data (RxD): Data sent from DCE to DTE,
3. Request To Send (RTS): Asserted (set to 0) by DTE to prepare DCE to receive data,
4. Clear To Send (CTS): Asserted by DCE to acknowledge RTS and allow DTE to transmit,
5. Data Terminal Ready (DTR): Asserted by DTE to indicate that it is ready to be connected,
6. Data Set Ready (DSR): Asserted by DCE to indicate an active connection,
7. Data Carrier Detect (DCD): Asserted by DCE when a connection has been established with remote equipment, often a modem.

Handshaking The signals in addition to the TxD, the RxD, and the ground can be used for different levels of handshaking between the sender and the receiver. These signals can be used for a more reliable connection. The most used handshaking signals are the RTS (Request To Send) and CTS (Clear To Send) where the transmitter will set the RTS to request a transmission and the receiver will respond with the CTS when ready for receiving the data. This type of handshaking is used for controlling the data stream between the devices.

Using these control signals are the hardware handshaking, but software handshaking can be used as well. In software special characters are used. A software handshaking protocol that are used a lot is the XON/XOFF protocol. The receiver will send XON when it is ready to receive new data and XOFF when the transmitter should stop sending data. These characters are defined as:

Character	Transmission status	ASCII	Decimal	Hex	Keyboard
XON	Pause in transmission	DC3	19	$0x13$	Ctrl+S
XOFF	Resume transmission	DC1	17	$0x11$	Ctrl+Q

The handshake signals are just like a control system with feedback control, the receiver will give feedback to the transmitter of when more data can be received. The following signals are used:

- Hardware; The transmitter will set the RTS when ready to transmit, the receiver will set the CTS when data can be received; using two different pins and extra wires;
  - the transmitter activate the RTS electrical signal and waits for activation of the CTS from the receiver before sending the next data,
  - the transmitter will check for an active CTS signal for every byte to send,
- Software; The receiver will send a XON when more data can be received, and a XOFF when data can not be received; using both ways communication; avoid the XON/OFF codes in your normal data streams of data.
  - the transmitter will check the receive buffer for any handshake codes; if the last code was XON the next byte is sent; if the last code was XOFF the transmitter will wait for a XON code before transmitting the next byte;
  - the protocol will define the codes (single bytes or sequence of bytes) for the XON/OFF,
  - the ASCII (American Standard Code for Information Interchange) codes for XOFF/XON are:

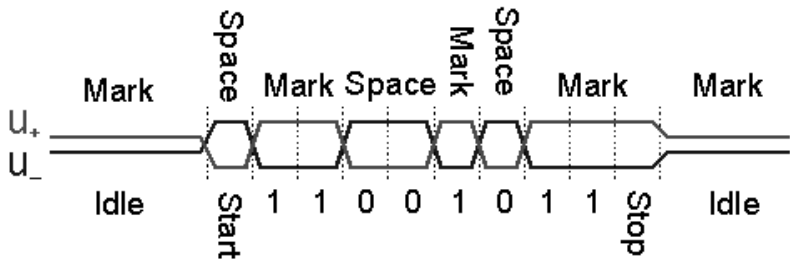


Figure 17.11: RS-485 communication.

Code	Meaning	ASCII	Decimal	Hexadecimal	Keyboard
XOFF	Pause transmission	DC3	19	$0x13$	Ctrl+S
XON	Resume transmission	DC1	17	$0x11$	Ctrl+Q

Test Equipment A multimeter or a LED together with a resistor can be a useful tool for checking the data and control signals. A scope may be necessary to check the data transmitted on the wire.

RS-422

To overcome the problem with only 2 devices and the length limitation, RS-422 was defined. RS-422 is using balanced or differential signalling, but does not allow multiple drivers, only multiple receivers.

Several key advantages offered by this standard include the differential receiver, a differential driver and data rates as high as 10 Megabyte at 12 metres (40 ft). The specification itself does not set an upper limit on data rate, but rather shows how signal rate degrades with cable length (Gofton 1986).

The maximum cable length is 1200 m. Maximum data rates are 10 Mbit/s at 12 m or 100 kbit/s at 1200 m. RS-422 cannot implement a truly multi-point communications network, however one driver can be connected to up to ten receivers.

RS-485

RS-485 was designed to overcome the problem with only one driver on RS-422, so RS-485 is a electrical specification of a two-wire, half-duplex, multipoint serial connection. The electrical communication is differential form of signalling using the voltage difference between the A and B cables. One polarity of voltage indicates logic 1 level, the reverse polarity indicates logic 0. The difference of potential must be at least 0.2 volts for valid operation, but any applied voltages between +12 V and -7 volts will allow correct operation of the receiver.

The data transmission speeds can be 35 Mbit/s up to 10 m and 100 kbit/s at 1200 m. In contrast to EIA-422, which has a single driver circuit which cannot be switched off, EIA-485 drives need to be put in transmit mode explicitly by asserting a signal to the driver. This allows EIA-485 to implement linear topologies using only two lines.

The pins are labeled A and B, where A is the inverting pin being negative compared to B when the line is idle. The communication is showed in Figure 17.11, where U_- is A and U_+ is B.

Using differential form only 2 connections are used, both transmitting and receiving on the same connection. RS-485 is only the physical layer, a higher layer in the OSI model must take care of the master/slave principles allowing only one device to transmit at a time. The maximum numbers of devices that can be connected are 32 using the standard, but new hardware allow up to 256 devices. The communication line will be operated in three states as:

1. Logic 1,
2. Logic 0,
3. High impedance.

The RS-485 has to be terminated with a 120 Ohm resistor at the devices at each end on the cable. See Figure 17.12.

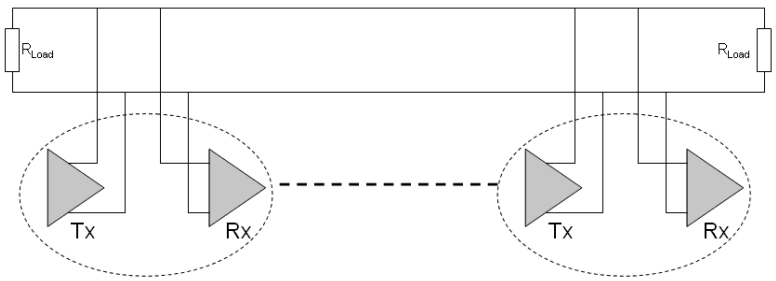


Figure 17.12: The RS-485 bus with Tx/Rx devices and the termination resistors.

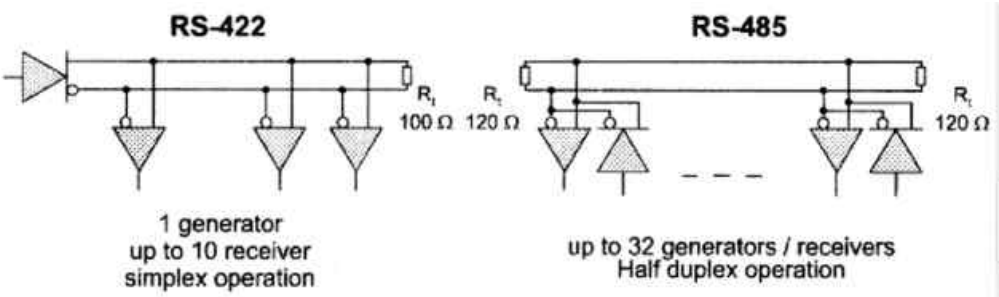


Figure 17.13: The difference between RS-422 and RS-485. RS-422 with only one transmitter and up to 10 receivers. RS-485 with up to 32 transmitters and receivers.

RS-485 can be used with a 2 wire solution in a master/slave system. The master is controlling the RS-485 bus where every device is addressable. The master is controlling the communication with the addressable devices, the timing of the bus, and the data flow.

The difference between RS-422 and RS-485 is shown in Figure 17.13. This Figure shows the bus topology, the number of transmitters and receivers, the simplex/duplex operations, and the termination of the bus.

A typical RS-485 network (or bus) is shown in Figure 17.14. This figure shows a 2 wire network with 4 nodes connecting the A and B lines between the nodes. The nodes at each end of the network must also include a network termination, a resistor of $120\,\Omega$. In this figure also the groundings are interconnected using a set of resistors, but this may depend on the interconnection using common power supply among others.

RS-485 connection can also be extended using repeaters shown in Figure 17.15.

Error Detection

Electrical communication systems are effected by noise. Noise is any unwanted signal in a system. Noise can be electromagnet fields inducing currents in nearby electrical components and cause them to produce unexpected and/or incorrect behavior. The inducing currents can produce errors such as toggled, missed or duplicated bits on a communication line, both parallel and serial. The data received will not be the same as the data transmitted and the data should be rejected. The solutions are:

1. add an error detection system like a parity check or a checksum. Note that the layer #1 of the OSI model is only handling bits, while layer #2 is handling bytes so the error detection systems may be covered by different layers,

2. forward error correction (FEC) where redundant data is added to the transmitted message so the receiver can both identify and correct transmission errors. The correction methods are based on bytes, so this must be covered by layer #2 (or higher) in the OSI model.

Error detection system is to add some sort of check of the transmitted data, either a parity check or a checksum. Parity check is normally a single bit for checking each byte or character, while the checksum is normally a check of several bytes or characters.

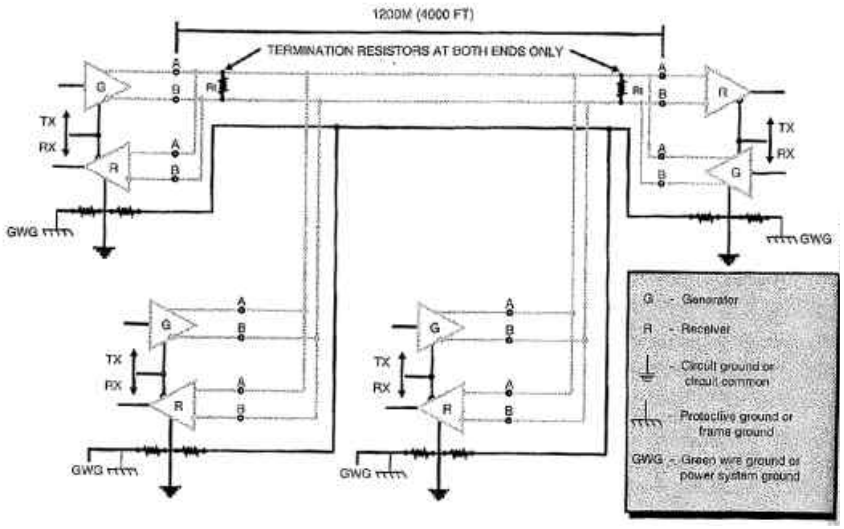


Figure 17.14: A typical two wire RS-485 multidrop network with 4 nodes and the termination resistors (Mackay et al. 2004).

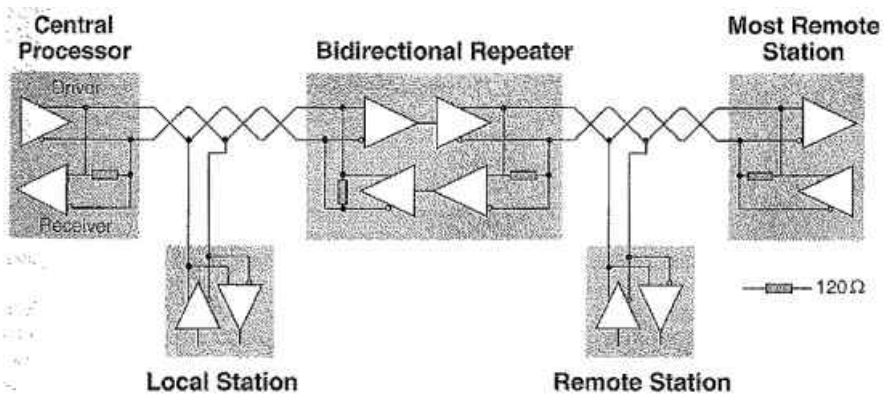


Figure 17.15: A RS-485 connection extended using a repeater (Mackay et al. 2004).

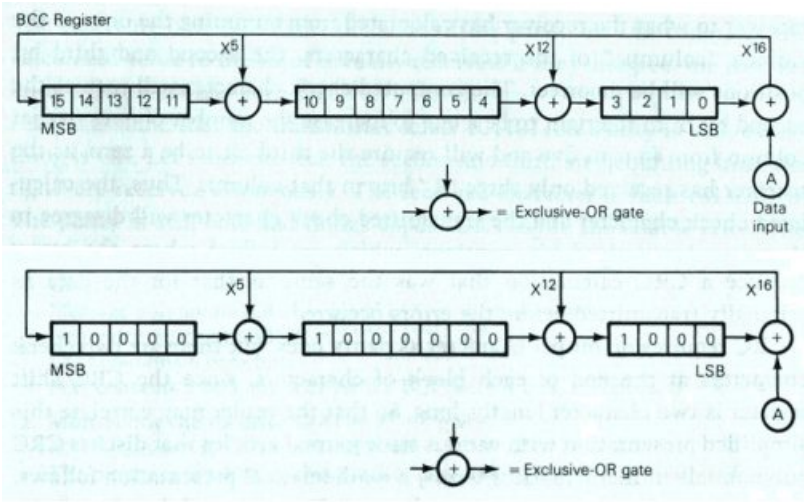


Figure 17.16: CRC register using the generator polynomial $P(X) = \left(X^{12} + X^5 + 1\right)$. The upper part shows the block check calculation (BCC) register, the lower part shows the register filled with 0 and the first bit as 1 (McNamara 1988).

In their most basic form, checksum are a form of a redundancy check which allow the receiver of the data to protect the integrity of the system. The transmitter typically will calculate the checksum of the data to be sent with the data to the receiver, and the checksum will be added to the data message. It is the job of the receiver to verify that the checksum matches with the data and the original checksum.

The most basic checksum algorithms entail adding up components or patterns in a message and then comparing these totals to the checksum value of the original message to provide a form of authentication. Often there is a three-way trade-off between the computing power used on the checksum, the size of the checksum itself and the probability of not detecting an error. Basic checksum algorithms can fail to detect complex errors like if two bytes are transposed, while more complex algorithms such as a Cyclic Redundancy Check (CRC) can solve such errors.

A checksum algorithm is based on adding all the bytes in a message and sending the sum as the last byte in the message. A CRC algorithm has two main advantages over a checksum algorithm:

1. reveals a higher proportion of errors,
2. bit oriented, not character oriented; can work with protocols that produce both stream of bits and streams of bytes.

The CRC algorithm is based on calculations and there exist different standards for these calculations, some are known as CRC-12, CRC-16, and CRC-CCITT. The common solution is that the message is treated as one huge binary number. The CRC algorithm is (McNamara 1988):

1. Multiply the message $G(x)$ by X^{n-k} where $n-k$ is the number of bits in the block check calculation. n is the total number of bits in the message and k is the number of data bits,
2. Divide the resulting product $X^{n-k}[G(x)]$ by the generator polynomial $P(x)$,
3. Disregard the quotient and add the remainder $C(x)$ to code message polynomial $F(x)$. $F(x)$ is represented as $X^{n-k}[G(x) + C(x)]$

The polynomial $F(x)$ is transmitted to the receiver and the receiver is using the $P(x)$ to regenerate the message $G(x)$. Different $P(x)$ are used, known as CRC-12, CRC-16, and CRC-CCITT. CRC can detect several bit errors and is used a lot for serial communication protocols. CRC algorithms are implemented in hardware shift registers shown in Figure 17.16. The Figure shows a register for a total message length of 16 bits and the generator polynomial $P(X) = \left(X^{16} + X^{12} + X^5 + 1\right)$ being the CRC-CCITT standard. The lower part of the Figure shows the block check calculation (BCC) register filled with $0's$ and the first bit of the message bit stream fed into the register.

A simple example (McNamara 1988):

1. The total number of the message length: $n = 10$,
2. The message polynomial: $G(X) = 110011$ giving $\left(X^5 + X^4 + X + 1\right)$,
3. The generator polynomial: $P(X) = 11001$ giving $\left(X^4 + X^3 + 1\right)$,
4. The number of data bits: $k = 6$ giving $n - k = 4$,
5. $X^{n-k}\left[G(x)\right] = X^4\left[\left(X^5 + X^4 + X + 1\right)\right] = \left(X^9 + X^8 + X^5 + X^4\right)$,
6. $C(X) = \frac{1100110000}{11001} = 1001$,
7. $F(X) = X^{n-k}\left[G(x) + C(x)\right] = 1100111001$,
8. $F(X)$ is transmitted,
9. $G(X) = \frac{F(X)}{P(X)}$ if the reminder is 0.

FEC algorithms are working without retransmissions as long as the number of errors stays reasonable low. The advantages of FEC are that half or full duplex can be avoided and a lower bandwidth, in average, is used. FEC can then be applied in situations where retransmissions are relatively costly or impossible. In particular, FEC information is usually added to most mass storage devices to protect against damage to the stored data(www.wikipedia.org 2010). Several FEC algorithms exist, some of these are used:

1. for CD encoding (Reed-Solomon),
2. in GSM mobile phones transmission.

The FEC algorithms are split into two categories:

1. linear block codes, work on fixed length blocks of data,
2. convolutional codes manage a continuous stream of bits.

In a linear block code will the most basic approach be repetition, just repeating the block code a number of times. A checksum added to each message or with a majority voting method the correct block code can be assumed. Some of the linear block codes are Golay, BCH (initials of the inventors' names), Reed-Solomon, and Hamming codes.

Comparing RS-232C, RS-422, and RS-485

	RS-232C	RS-422	RS-485
Operation	Single ended	Differential	Differential
Transmitters	1	1	32
Receivers	1	10	32
Max. length	15m	1200m	1200m
Max. speed	115kbit/s	10Mbit/s	10Mbit/s
Termination	None	None	120 Ω
Topology	Point to point	Bus	Bus
Handshaking	Hdw and Stw[1]	Stw	Stw

Converters exist for converting the electrical signal between any combinations of the RS-232C, the RS-422, and RS-485 signals.

[1] Hdw: Hardware based / Stw: Software based

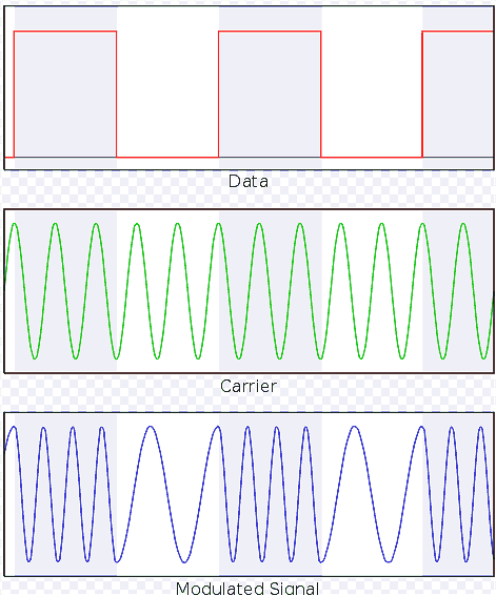


Figure 17.17: Frequency modulation of a digital signal (www.wikipedia.org 2006).

Modem

RS-232C, RS-422, and RS-485 are digital signals using an UART for sending the data on the wire. The length of the cable will attenuate and alter the signal, especially at high speeds. Sending the data over long distances a modem should be used. A modulation/demodulation device (modem) is transforming the signal by modulating at the sender and demodulating the signal at the receiver. Modulation is the process of varying a periodic waveform, often a sine wave, in order to use that signal to convey a message (www.wikipedia.org 2006). The message can be either an analog or a digital signal. A connection will need two modems, one at each connection points. Most often a digital connection is used and the frequency used are 1.07 and 1.27 kHz for one side and 2.025 and 2.225 kHz for the other side representing 0s and 1s (Levermore 2000). Using different frequencies for each side means that the communication channel will be full duplex. The length of a modem connection can be more than 100 km.

Digital modulation is to transfer a digital bit stream over an analog bandpass channel and analog modulation is to transfer an analog lowpass signal over an analog bandpass channel. The 3 parameters of a periodic waveform that can be used for modulation are the amplitude, the phase, and the frequency. All of these parameters can be modified in accordance with a low frequency information signal to obtain the modulated signal.

The 3 modulation parameters for digital modulation are:

1. amplitude modulation (AM); using the strength of the signal or amplitude peaks for representing a 1 and a 0. This is called Amplitude-shift keying (ASK) where a finite number of amplitudes are used. See Figure 17.18.

2. frequency modulation (FM); using different frequencies for representing a 1 and a 0. This is called Frequency-shift keying (FSK) where a finite number of frequencies are used. See Figure 17.17 for the principles for FSK where 2 different frequencies are used for representing a digital 1 and 0. See also Figure 17.18.

3. phase modulation; using the phase of the carrier sine wave for representing a 1 and a 0. This is called Phase-shift keying (PSK) where finite number of phases are used. Phase modulation requires more complex hardware circuits and is not used as much as AM and FM. See Figure 17.18.

Frequency modulation and phase modulation are in some litterateur mention as angle modulation.

An example of a modem can be the cable modem used for internet access at homes. A cable modem is a type of modem that provides access to a data signal sent over the cable television or telephone infrastructure. The modem gives you the internet access at home, using the modem for communication with the server at the internet service provider (ISP). Cable modem is taking the advantage of the

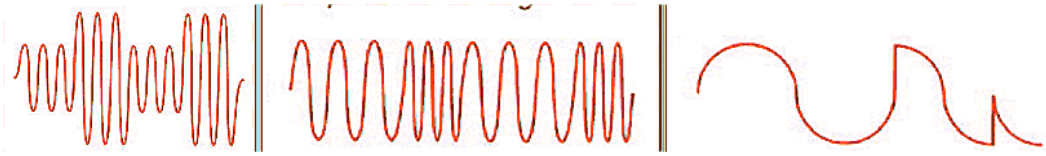


Figure 17.18: Different modulation types: amplitude modulation to the left, frequency modulation in the middle, and phase modulation to the right.

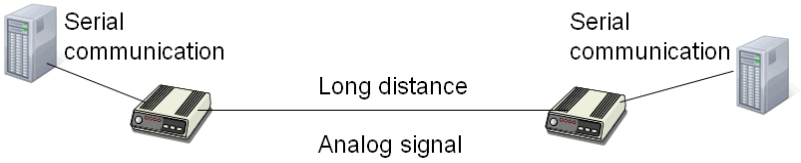


Figure 17.19: The usage of a set of modems for data communication.

high bandwidth of a cable television or telephone network and the principle is shown in Figure 17.19. Regarding the OSI model will a cable modem be a data link layer (or layer 2) forwarder.

17.2.4 Current Loop

A current loop describes two different electrical signalling schemes, both digital and analog. A current loop is using the current in stead of the voltage, and can be used over moderately long distances (tens of kilometers), and can be interfaced with optically isolated links (www.wikipedia.org 2006). Digital signalling is working very much the same way as the RS-232C or RS-485 signalling.

Using RS-232C or RS-485, the voltage potential is used for the communication, but then the devices have to be power external. Current loop can be used for both power and communication, and especially the analog current loop which is called the $4-20$ mA. The analog current loop is mainly used for external devices, and for devices working with a maximum current of 4 mA. The $4-20$ mA device contains a current regulation to adjust the current between 4 and 20 mA depending on the environment, but only one value can be transferred. Note that the $4-20$ mA protocol is only for analog signals.

The current loop device can be either an input device or an output device, and Figure 17.20 shows a sensor device. The resistor has to be adjusted for the power supply and is also used for current to voltage converter for the current loop value. A current loop is "low" resistance, while a standard serial communication line is "high" resistance due to this resistor. A "high" resistance connection may be more affected by noise.

Using a power supply of 24V and a resistor of 250 Ohm gives 1V for 4mA and 5V for 20mA. A/D[2] converter with 5V as input can then convert the current to a digital value. Easy to troubleshoot, just

[2] A/D: Analog to Digital

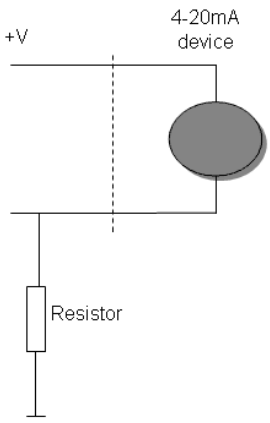


Figure 17.20: Current loop sensor ($4-20mA$)

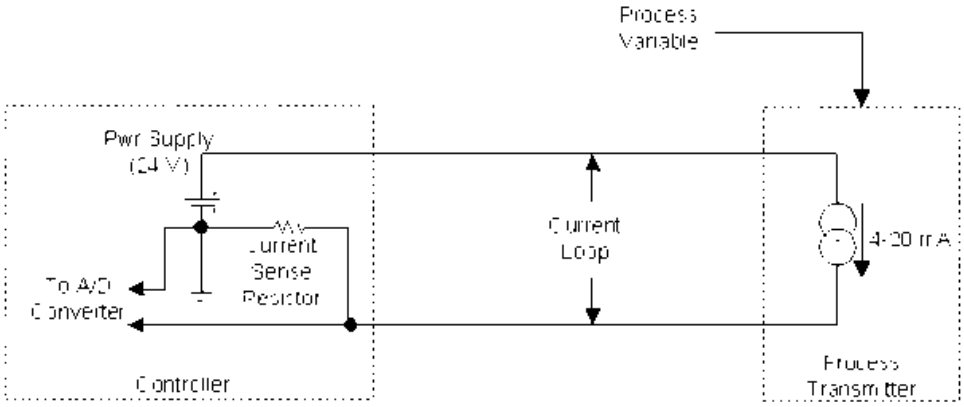


Figure 17.21: A $4-20$mA sensor device connected to a DAQ system. The current loop consisting of only two wires contains both the power and the analog signal for the sensor device (www.analogservices.com: feb-2010).

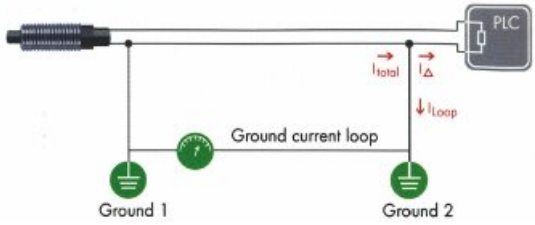


Figure 17.22: The challenge using ground as reference in distributed systems (www.wago.com: jun-09).

measure the current, if no current no connection, otherwise convert the value of the current to the measured value. The devices can also use a current outside the $4-20$ mA range as indication of an error with the device (internal device error), for instance 3.5 or 20.5 mA.

The sensor device contains a current generator adjusting the current according to the process variable. The sensor device itself is using 4 mA. Figure 17.21 shows a 4-20 mA sensor device (process transmitter) connected to a DAQ system (controller). The DAQ system contains the 24 VDC power supply, the current sensor resistor for converting the analog current signal to a voltage signal, and the A/D converter.

Current loop can also be used for serial communication letting the process transmitter sending a digital signal in stead of an analog signal. The current generator will generate two different current values in the range of $4-20$ mA for "0" and "1", and the controller needs to convert the current signal into a serial communication.

Just remember that the sensor devices need a minimum voltage to work. This voltage together with the DAQ input voltage must be taken into consideration when selecting the power supply. This means if a sensor device needs minimum 9V to operate and the maximum DAQ input voltage is 5V, the minimum power supply must be 14V, see Figure 17.21.

17.2.5 Galvanic Isolation

Interconnecting several systems may result in an earth fault. Interconnecting these systems include connections between the grounding of each system, introducing a current flow between the systems because of small differences in the ground potentials. This current flow will be noise for the communication and may affect the communication, both analog and digital, in a bad way. Of course, analog communication will be affected as wrong signal levels. However, digital communication may also be affected, by changing the digital bit streams, but depends on the error checking/corrections.

Figure 17.22 shows the challenge with ground reference in distributed systems. Depending on the length of the cables, the potential difference will result in measurement error and noise.

Galvanic isolation is a type of a safety barrier that can let the data and/or power pass, while blocking charge or current flow. Normal solutions will be to use a signal transformer or an optical isolator (opto coupler). Figure 17.23 shows the usage of galvanic isolation modules.

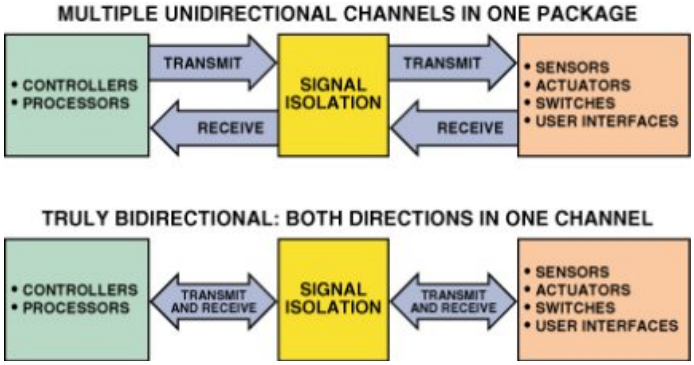


Figure 17.23: Isolation of unidirectional and bidirectional communication channels. Isolation of bidirectional channles will be more complex as signal must be able to pass both ways. (www.analog.com: NOV-14).

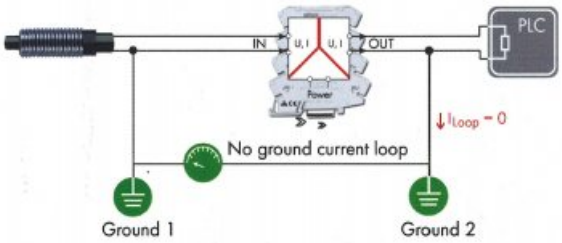


Figure 17.24: Using galvanic isolation to avoid the measurement error and noise due to potential difference in a distributed system (www.wago.com: jun-09).

The galvanic isolation module consists of isolation inputs, often a an isolation amplifier, with transient overvoltage circuits on both inputs and outputs. Using a galvanic isolation module will transfer the input signal to the output unchanged, independent of the potential differences. Figure 17.24 shows the usage of a galvanic isolation module to reduce the impact of measurement errors and noise due to potential differences in distributed systems.

The potential differences may be a challenge already when the length of the cable is 10 meters, and the potential difference can be in the range of volts.

There exists two different type of galvanic isolators, active and passive. The active isolator requires an external power supply, often 24 VDC. Figure 17.25 shows an active galvanic isolator module, while Figure 17.24 shows a passive galvanic isolator module. A passive module will use any power from the input signal for the operation, so the input signal needs to contain enough energy for the module, like in the $4 - 20$ mA signal.

Protocols including the ground signal in layer #1 of the ISO model must have special focus on the isolation strategy between the systems.

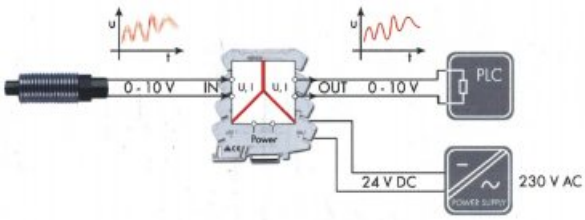


Figure 17.25: An active galvanic isolation module powered by 24 VDC. This module may also, depending on the configuration, remove noise as well (www.wago.com: jun-09).

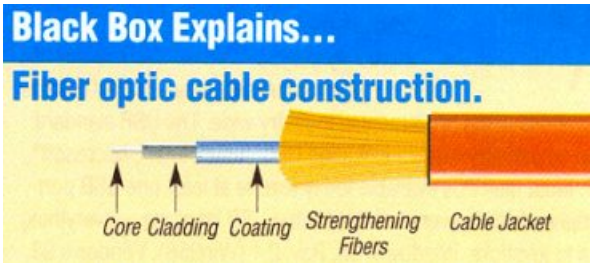


Figure 17.26: The protection of the glass fibre (www.blackbox.com:nov-08)

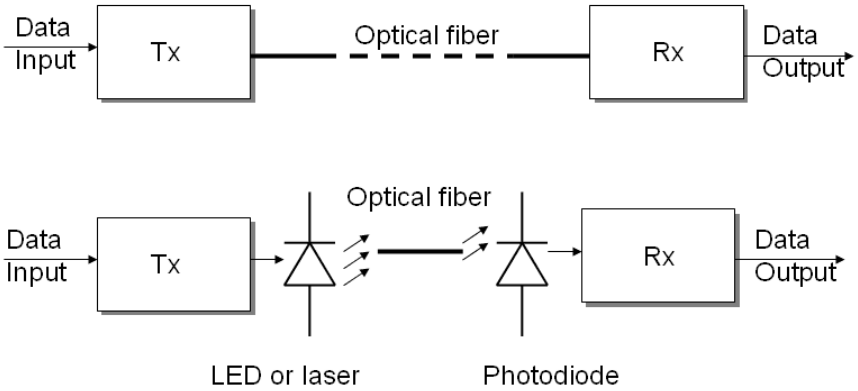


Figure 17.27: Devices used for a optical communication.

17.2.6 Optical or fibre cables

One way of avoid earth fault is to convert electrical signals into light pulses. Signal transmission by light pulses in optical cables can also be used as a physical layer for communication protocols. The optical cables are thin glass fibre with a diameter between 10 μm and 125 μm, and this thin glass fibre, the core, is protected as shown in Figure 17.26. There exists two different types of optical cables or optical fibres, single mode and multimode, where single mode has the smallest diameter. An overview of the devices needed for optical data communication is shown in Figure 17.27 and consists of:

1. optical signal transmitter; convert the electrical signal into light pulses,
2. optical cable (optical fibre), transmitting the optical pulses, either single mode or multimode cable (fibre),
3. optical signal receiver, converting the light pulses into electrical signals,
4. signal processing units for the optical signal transmitter and the optical signal receiver.

Optical signal transmission has several advantages over electric transmission, the principal advantages are (Olsson & Piani 1998):

1. High bandwidth (100 Mbit/s or higher),
2. Lower signal attenuation per unit length than electric wires giving tens of kilometers between optical repeaters,
3. Immune to electromagnet disturbance (noise),
4. Total isolation,
5. Thin and lightweight giving small sizes,
6. More secure.

Among the disadvantages must be mentioned (Olsson & Piani 1998):

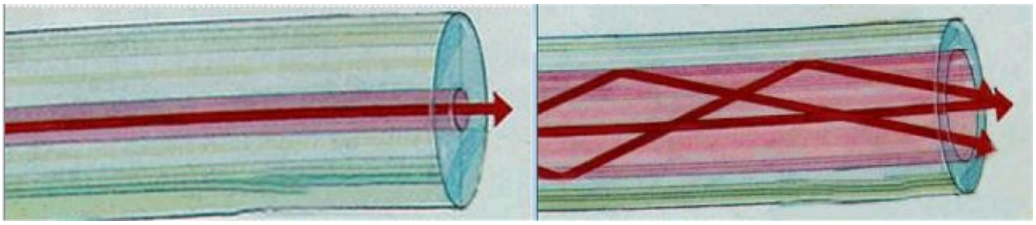


Figure 17.28: Single mode fibre to the left, and multimode fibre to the right (www.arcelect.com: jan-09).

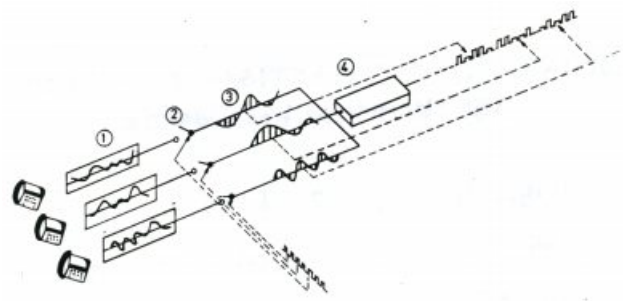


Figure 17.29: Multiplexing of different data on only one optical fiber.

1. The work for connecting several fibers is difficult,
2. Optical fibers connect two points only,
3. The interface devices (electrical/optical) are complex and therefore more expensive than for electrical cables.

As shown in Figure 17.27 the optical transmitter can be a light-emitting diode (LED) or a laser diode for converting the electrical signal to light pulses. A LED has a limited optical power output up to 0.1mW giving a maximum transmission speed of 8 Mbit/s (Olsson & Piani 1998). The more expensive laser diodes have an optical power output of 100 times higher than the LED, up to 10 mW. The transmission speed of laser diodes can reach a transmission speed > 10 Gbit/s.

A distinction is made between single mode and multimode fibres. Single mode fibre has a very thin core which is used together with a laser for long range high speed communication. The diameter of the fiber is normally between 8 to 10 microns. A multimode fibre is more coarse giving lower data rates and shorter ranges. Multimode fibre is most commonly used in local data communication applications. The diameter of the fiber is normally between 50 to 100 microns. The difference of single mode and multimode is shown in Figure 17.28.

Due to the high speed data can easily be multiplexed on one optical fibre, an overview is shown in Figure 17.29. 3 different electrical signals (1) are merged (4) using a multiplexer (2). The essence of the sampling theorem is that the samplings frequency has to be at least twice as high as the highest frequency component in the signal, and using a multiplexer, the frequency of the multiplexer must be high enough for getting the highest frequency component for all the signals.

17.2.7 Radio Data Communication

In stead of a media like a wire or fibre, radio waves can be used. Radio waves are an alternative when fixed links cannot be installed, for example in communication with moving objects like vehicles and trains. Radio waves are using one or several dedicated frequencies and these frequencies are usually in the VHF (Very High Frequency) or UHF (Ultra High Frequency). VHF is in the area of 30 to 300 MHz and UHF is in the area of 300 MHz to 3 GHz.

Full duplex protocol requires at least two different frequency ranges, with only one frequency range available and a two-way communication is needed only half-duplex protocol can be used. The transmitter and receiver have to use the same frequency range to be able to communicate, often called a channel. Half-duplex means using the same channel, full duplex means using two different channels, one for transmitting data and one receiving data. Since the data consists of digital information, different frequencies are used for sending zeros and ones. Each channel must have a minimum bandwidth to transfer

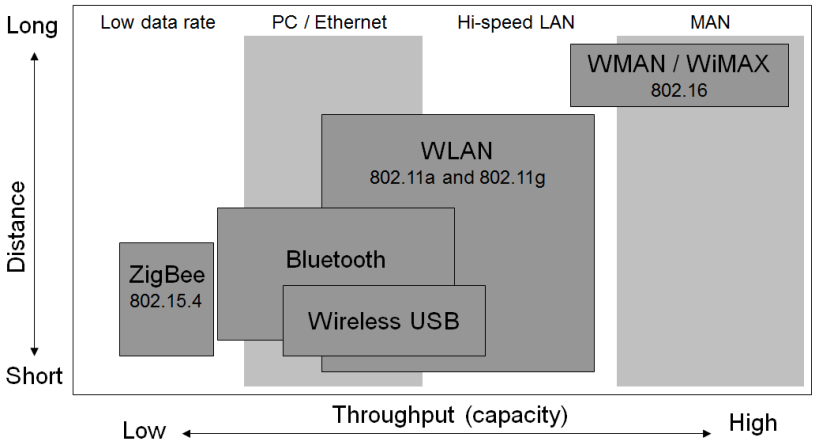


Figure 17.30: An overview of the throughput and distances for some of the wireless protocols.

the digital information. Two transmitters can not use the same frequency range within the same area for transmission, this will cause collisions. A technique can be frequency hopping (Frequency Hopping Spread Spectrum - FHSS) where the channel frequency is changed continuous at fixed time intervals. Any loss of data will then be retransmitted in the next time interval.

Radio communication is working without any cable or wire, and is named wireless communication. Wireless communication uses a radio transmitter/receiver with radio waves as the physical layer, and some of the protocols used are Bluetooth, ZigBee (802.15) and Ethernet (802.11). Most of these protocols are based on the IEEE 802.15.4 standard. An overview of these protocols with respect of distance and throughput is shown in Figure 17.30.

The Wireless USB is described in subsection 18.1.2. WMAN is Wireless Metropolitan Area Network, also known as WirelessMAN or WiMAX. WiMAX is acronym for Worldwide Interoperability for Microwave Access. WMAN or WiMAX will not be part of this course.

Power usage may also be an important issue for wireless nodes as they may run on batteries. The speed and length of the wireless communication will affect the power usage of the nodes. Energy harvesting where the nodes harvest the energy or part of the energy from either light, temperature changes, or linear motion, may offer many benefits. There exists several energy saving devices that can harvest energy from the surroundings that can reduce the operational and maintenance costs of the nodes running on batteries.

ZigBee

ZigBee is the name of a specification for a suite of high level communication protocols using small, low-power digital radios based on the IEEE 802.15.4 standard for wireless personal area networks (WPANs) (www.wikipedia.org 2006).

The raw data rate is 250 kbit/s per channel in the 2.4 GHz band, 40 kbit/s per channel in the 915 MHz band, and 20 kbit/s in the 868 MHz band. Transmission range is between 10 and 75 meters (33˜246 feet), although it is heavily dependent on the particular environment. The maximum output power of the radios is generally 0 dBm (1mW).

The basic channel access mode specified by IEEE 802.15.4 – 2003 is "carrier sense, multiple access/collision detection" (CSMA/CD).

ZigBee protocols are intended for use in embedded applications requiring low data rates and low power consumption. ZigBee's current focus is to define a general-purpose, inexpensive, self-organizing, mesh network that can be used for industrial control, embedded sensing, medical data collection, smoke and intruder warning, building automation, home automation, , etc. The resulting network will use very small amounts of power so individual devices might run for a year or two using the originally installed battery.

The software is designed to be easy to develop on small, cheap microprocessors. The radio design used by ZigBee has been carefully optimized for low cost in large scale production. It has few analog stages and uses digital circuits wherever possible.

All the routers are normally mains-powered devices (lamps, heat pump, lighting fixtures, smoke alarms) and the "end" devices are normally battery-powered (switches, thermostats, motion detectors).

Protocol	Type	Range	Datarate	Devices
ZigBee	2.4 GHz		250 kbit/s	
(IEEE 802.15.4)	915 MHz	10m-75m	40 kbit/s	65535
	868 MHz		20 kbit/s	

The number of ZigBee devices in a network depends on the topology, but can be up to 65535 nodes. The topology is normally star or mesh.

Wireless HART

Wireless HART specification was released late in 2007 and was the first international standard for industrial wireless sensor networks. The protocol is using only the 2.4 GHz frequency band, consists of 15 channels, each channel requires a bandwidth of 3 MHz, and the spacing between the channels is 5 MHz. The network is a mesh network topology where every sensor nodes are routers. The protocol is using frequency hopping across all 15 channels, using Frequency-division multiple access (FDMA). The communication is based on time slots using Time division multiple access (TDMA) where each time slot is 10 ms. The data packets are frames consisting of 1024 time slots.

Bluetooth

Bluetooth is an industrial specification for wireless personal area networks (WPANs). Bluetooth provides a way to connect and exchange information between devices such as mobile phones, laptops, PCs, printers, digital cameras and video game consoles via a secure, globally unlicensed short-range radio frequency (www.wikipedia.org 2006).

Class	Power	Range
Class #1	100 mW	$\approx$ 100 m
Class #2	2.5 mW	$\approx$ 10 m
Class #3	1 mW	$\approx$ 1 m

A Bluetooth device playing the role of the "master" can communicate with up to 7 devices playing the role of the "slave". This network "group of up to 8 devices" (1 master and 7 slaves) is called a piconet. A piconet is an ad-hoc computer network of devices using Bluetooth technology protocols to allow one master device to interconnect with up to seven active slave devices. Up to 255 further slave devices can be inactive, or parked, which the master device can bring into active status at any time.

At any given time, data can be transferred between the master and one slave; but the master switches rapidly from slave to slave in a round-robin fashion. (Simultaneous transmission from the master to multiple slaves is possible, but not used much in practice). Either device may switch the master/slave role at any time.

Bluetooth specification allows connecting two or more piconets together to form a scatternet, with some devices acting as a bridge by simultaneously playing the master role in one piconet and the slave role in another piconet. These devices have yet to come, though are supposed to appear in 2007.

The Bluetooth protocol operates in the license-free ISM band at 2.45 GHz. In order to avoid interfering with other protocols which use the 2.45 GHz band, the Bluetooth protocol divides the band into 79 channels (each 1 MHz wide) and changes channels up to 1600 times per second. Implementations with versions 1.1 and 1.2 reach speeds of 723.1 kbit/s. Version 2.0 implementations feature Bluetooth Enhanced Data Rate (EDR), and thus reach 2.1 Mbit/s. Technically version 2.0 devices have a higher power consumption, but the three times faster rate reduces the transmission times, effectively reducing consumption to half that of 1.x devices (assuming equal traffic load).

Version	Data rate	Note
1.1	723.1 kbit/s	
1.2	723.1 kbit/s	
2.0	2.1 Mbit/s	

The topology is normally star.

Bluetooth SMART

Internet of Things (IoT) requires a protocol that can run for a long time on a set of batteries. Several low power protocol exists, one of these protocols is the Bluetooth SMART. This protocol has other names like Bluetooth SMART, Bluetooth Low Energy (LE) or Bluetooth 4.0. This protocol supports very short packets between 8 and 27 octets transfered at 1 MBps, with frequency hopping. The latency is very short, down to 3 ms, to allow for fast connections and data transfer. The range can be up to 100 m with a 24 bit CRC on all packets for robustness against interference. A full AES-128 encryption is used on all packets for security and 32 bits are used for addressing.

SMART phones normally include this protocol allowing for local communication. One type of devices may be the Beacon, or iBeacon, small devices transmitting local information. These beacons can work for years on a single battery device. The distance is normally up to 30 meters.

SP100

SP100 is the wireless standard from ISA (International Society of Automation) that can be used for both process industry and factory automation. SP100 is designed for the industry with focus on reliability, predictability and security. Reliability includes enhanced error detection and frequency hopping. Predictability includes TDMA (Time Division Multiple Access) and different QoS (Quality of Service) levels. TDMA means that the frequency range is divided into time slots and allocating nodes to different time slots. SP100 supports multiple industrial protocols via a single wireless infrastructure. These industrial protocols include HART, Profibus, ProfiNet, Modbus TCP, Fieldbus Foundation to mention some. The protocol is also designed for low power usage to include applications based on battery including battery powered routers.

The field devices are either Full Function Devices (FFD) or Reduced Function Devices (RFD) together with backbone devices as routers and gateways. The backbone is to interconnect several wireless networks and/or connection to the plant network.

The SP100 protocol supports layer $1, 2, 3, 4$ and 7 in the OSI model, and will be using the 2.4 GHz frequency band.

Ethernet

Wireless Ethernet consisting of the 802.11 family which currently includes six different modulation techniques all using the same protocol. The most popular (and prolific) techniques are those defined by the b, a, and g amendments to the original standard; security was originally included and was later enhanced via the 802.11i amendment. 802.11n is another modulation technique that has recently been developed (2012). Other standards in the family (c–f, h, j) are service enhancements and extensions or corrections to previous specifications. 802.11b was the first widely accepted wireless networking standard, followed by 802.11a and 802.11g.

802.11b and 802.11g standards use the 2.40 GHz (gigahertz) band. Because of this choice of frequency band, 802.11b and 802.11g equipment can incur interference from other wireless devices like Bluetooth devices using this same band. The 802.11a standard uses the 5 GHz band, and is therefore not affected by products operating on the 2.4 GHz band.

Protocol	**Type**	**Range**	**Data rate**
Ethernet	802.11		2 Mbit/s
(IEEE 802.11)	802.11.a	≈ 30 m	54 Mbit/s
	802.11.b	≈ 30 m	11 Mbit/s
	802.11.g	≈ 30 m	54 Mbit/s
	802.11.n	≈ 125 m	540 Mbit/s

The topology is normally star.

6LoWPAN 6LoWPAN is IPv6 over Low power Wireless Personal Area Networks, one way of fulfilling the internet everywhere wishes! The 6lowpan group has defined encapsulation and header compression

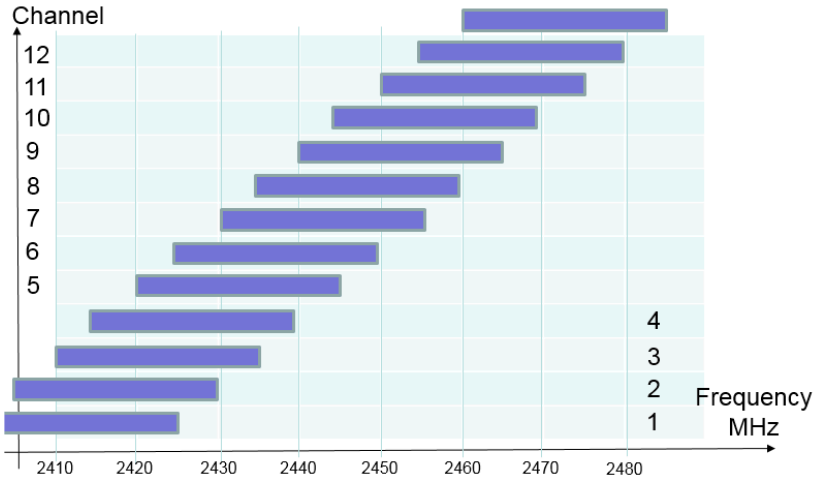


Figure 17.31: The wireless Ethernet (Wi-Fi) frequency ranges and overlapping channels.

mechanisms that allow IPv6 packets to be sent to and received from over IEEE 802.15.4 based networks. 6LoWPAN is another protocol developed for the Internet of Things (IoT).

Wireless Frequency Wireless nodes using Ethernet (IEEE 802.11b/g) communicate with each other using radio frequency signals in the ISM (Industrial, Scientific, and Medical) band between 2.4 GHz and 2.5 GHz. The frequency difference between each channel is 5 MHz, while a node using a particular channel will utilize frequency spectrum 12.5 MHz above and below the center channel frequency. This means that two separate wireless networks using neighboring channels in the same physical area will interfere with each other, giving fewer usable channels[3]. The radio frequency channels used for wireless Ethernet are listed in the table below:

Channel	Center Frequency	Frequency Spread
1	2412 MHz	2399.5 *to* 2424.5 MHz
2	2417 MHz	2404.5 *to* 2429.5 MHz
3	2422 MHz	2409.5 *to* 2434.5 MHz
4	2427 MHz	2414.5 *to* 2439.5 MHz
5	2432 MHz	2419.5 *to* 2444.5 MHz
6	2437 MHz	2424.5 *to* 2449.5 MHz
7	2442 MHz	2429.5 *to* 2454.5 MHz
8	2447 MHz	2434.5 *to* 2459.5 MHz
9	2452 MHz	2439.5 *to* 2464.5 MHz
10	2457 MHz	2444.5 *to* 2469.5 MHz
11	2462 MHz	2449.5 *to* 2474.5 MHz
12	2467 MHz	2454.5 *to* 2479.5 MHz
13	2472 MHz	2459.5 *to* 2484.5 MHz

The preferred channel separation between the channels in neighboring wireless networks is 25 MHz (five channels) meaning that it is only possible to apply up to three different channels within a wireless network. The table is presented graphically in Figure 17.31 showing the frequency range and the overlapping frequencies for the available channels.

Security Wireless Ethernet is often referred to as WLAN and gives a freedom of location both inside and outside an organization. The range of these WLAN's is at least 30 m and more than 100 m on the newest standard. Using WLAN in an organization then it will be important to have some sort of security policy to be able to control the access to the network.

An open WLAN will distribute the name of the WLAN and allow everybody to connect to the network, especially when using DHCP. Open means that everybody can see the name of the network

[3] Regulations in the United States prohibit using channels greater than channel 11.

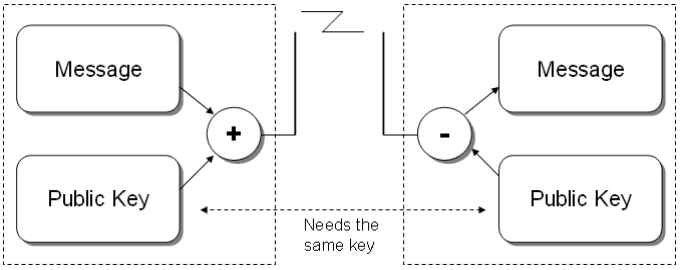


Figure 17.32: The figure shows the usage of a public key for encryption of a message in a WLAN system.

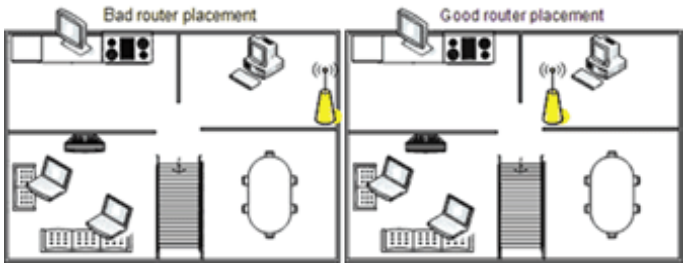


Figure 17.33: Bad and good wireless router placement (www.microsoft.com:nov-08).

and that the network is not using any form of encryption.

The name of the WLAN is the SSID, Service Set Identifier, and is broadcasted to be able to connect to the network. The list of WLANs shown in the setup section of the computer will be the SSID.

Two important parameters for the security of the WLAN will be the SSID and the encryption.

The SSID should be selected so that it is not easy to see the connection between the organization and the SSID. In some network it is also possible to use hidden SSID, you will not see the SSID in the list of networks, you will just need to know the SSID to connect to the WLAN.

The encryption is the encryption of the wireless traffic between the computers and the access point. The Wireless Equivalent Privacy (WEP) encryption will be one option, but is a weak, standalone encryption method with different standards. Using WEP will not guarantee that the computer will connect to the access point due to different standards and if the computer connects, the encryption will be weak.

The Wi-Fi Protected Access (WPA) protocol was created in response to several serious weaknesses in the WEP. WPA is administered by the Wi-Fi Alliance as a security protocol created to secure wireless computer networks. WPA is using a password key for access to the network. Pre-shared key (PSK) mode (also known as Personal mode) is designed for home and small office networks that don't require the complexity of an 802.1X authentication server. Each user must enter a passphrase to access the network. The passphrase may be from 8 to 63 printable ASCII characters or 64 hexadecimal digits (256 bits) (www.wikipedia.org 2006).

WPA2 is replacing WPA. The IEEE 802.11i encryption, also known as WPA2, is defined as a standard and using the Advanced Encryption Standard (AES) for encryption. The AES is the current highest standard for encryption giving a much better encryption than WEP. The predecessor to WPA2, WPA, is using Temporal Key Integrity Protocol (TKIP) which is an significant improvement over WEP. Both WPA and WPA2 can be used for WLANs. The WPA2 and WPA encryption standards require encryption keys, either enter manually when setting up the computer, or using a server for distributing of the encryption keys. The encryption methods will always need some sort of key, and both the transmitter and receiver need to use the same key, see Figure 17.32.

Extend/Improve the wireless network Some tricks/tips for extending your wireless range and improving your wireless network performance (www.microsoft.com:nov-08):

1. Place the wireless router (or wireless access point) in a central location. See Figure 17.33.

2. Move the router off the floor and away from walls and metal objects (such as metal file cabinets). See Figure 17.33.

3. Replace the router's antenna. See Figure 17.34.

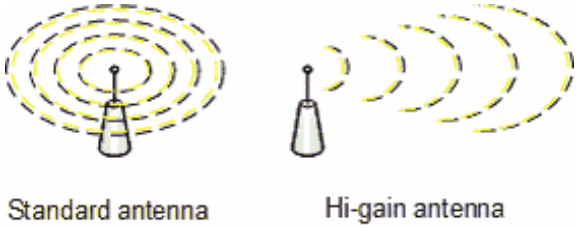


Figure 17.34: Use a directional antenna (www.microsoft.com:nov-08).

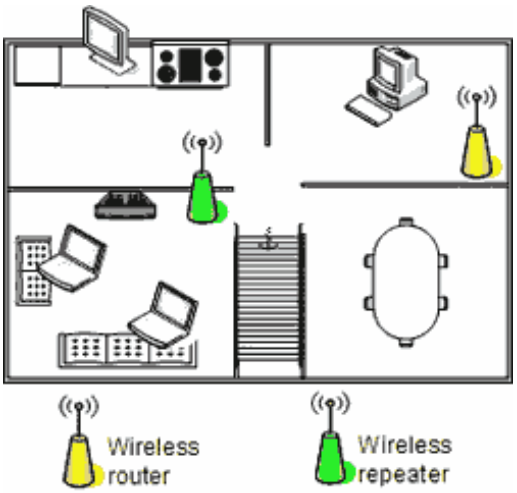


Figure 17.35: The usage of a wireles repeater to extend the network range (www.microsoft.com:nov-08).

4. Replace the wireless network adapter of a computer having communication problems. Remember that the wireless network signals must be sent both to and from the computer back to the router device.
5. Add a wireless repeater. Wireless repeaters extend the wireless network range without requiring any extra wiring. See Figure 17.35.
6. Change the wireless channel.
7. Reduce wireless interference. Check other wireless equipment like cordless phones etc.
8. Use wireless equipment from a single vendor.
9. Upgrade 802.11b devices to 802.11g.

Industrial Wireless Technologies

Commercial wireless and industrial wireless technologies often use the same radio technology. Standards like ZigBee, 6LowPAN, Wireless HART and ISA100 uses all the 2.4 GHz radio. This means that almost the same radio hardware can be used on different technologies. So the same hardware can be used, but the industrial solution may require:

- higher reliability,
- lower latency (close to real-time),
- more secure systems.

Wireless solutions have been used in monitoring applications especially where it was not possible or difficult to install a wired solution. Two major wireless technologies have been used in the industry, Wireless HART at the field level and Wireless LAN (WiFi) or Industrial Wireless LAN (IWLAN) at a higher level in the PCS. Security is integrated into Wireless HART and IWLAN. As a wireless solution is more flexible and transparent it will also move into the control areas, not only monitoring areas.

Subsea communication is based on acoustic signals, not radio signals. Acoustic signals are much faster in water than air, about 1500 m/s in water, but still much slower than a radio signal. This will be an challenge for any protocols regarding the amount of data and the speed especially since most of the protocols are based on handshaking and routing.

Wireless summary

A detailed overview of the range and data rate for the Bluetooth, ZigBee and WLAN protocols are:

Protocol	**Type**	**Range**	**Data rate**	**Devices**	**Frequency**	NOTE
Bluetooth	Class #1	≈ 100 m	v1.1=720 kbit/s	8 (255)	2.4 GHz	
	Class #2	≈ 10 m	v1.2=720 kbit/s	8 (255)	2.4 GHz	
	Class #3	≈ 1 m	v2.0=2.1 Mbit/s	8 (255)	2.4 GHz	
ZigBee		10m−75m	250 kbit/s	65535	2.4 GHz	
(IEEE 802.15.4)		10m−75m	40 kbit/s	65535	915 MHz	
		10m−75m	20 Kbit/s	65535	868 MHz	
Ethernet	802.11		2 Mbit/s	≈ 30		
(IEEE 802.11)	802.11.a	≈ 30 m	54 Mbit/s	≈ 30	5 GHz	
	802.11.b	≈ 30 m	11 Mbit/s	≈ 30	2.4 GHz	
	802.11.g	≈ 30 m	54 Mbit/s	≈ 30	2.4 GHz	
	802.11.n	≈ 125 m	540 Mbit/s	≈ 30		

17.2.8 5G wireless communication

The fifth generation (5G) network is using frequencies between 24 and 100 GHz for communication while the 4G network is using frequencies between 2 and 8 GHz. Higher frequencies means higher speed of information but also a reduction of wave length and signal coverage distance. The antenna and the locations of the antennas will be a huge challenge for 5G communication. The 5G network will also be important for machine to machine (M2M) communication for wireless communication especially for moving objects like cars, buses and trains to mention a few. More and more internet of things (IoT) devices will use the 5G wireless network. The three main areas for the 5G network will be:

- eMBB; enhanced mobile broad band - data speeds up to 20 Gbps will be available with 5G,
- mMTC; massive machine type communication - more stabile and availability for M2M communication,
- uRLLC; ultra-reliable and low latency communication - shorter delay times will give real-time options for wireless communication. The delay time can be less than 1 ms.

Chapter 18

Communication Protocols

A huge number of protocols exist, and this chapter will only deal with a very few of these protocols. The protocols with focus in this section will be SATA, USB, FireWire, Hart, Modbus and XML. Several protocols are used in the industry. Many of the protocols are proprietary, the most used open protocols are Modbus, XML and OPC to mention some. Remember that a lot of protocols exists and new protocols are entering the field from time to time. Some protocols can only be used on proprietary hardware, some on standard serial lines, some on network systems, and some on any combination.

18.1 Serial communication protocols

Serial communication protocols are using serial lines with limitation in speed capabilities and interconnections. Such protocols often support layer OSI layer #1, the physical layer, layer #2 if supporting more then two devices on the same bus, and layer #7 if a common set of commands are available. The protocol defines the capabilities, both the advantages and disadvantages. Remember, no perfect protocol exists.

18.1.1 SATA (Serial ATA)

SATA is the serial protocol for the ATA interface for storage devices on computers. The standard interface for SATA controllers is Advanced Host Controller Interface (AHCI). If AHCI is not enabled by the motherboard and chip set, SATA controllers typically operate in "IDE emulation" mode which does not allow features of devices to be accessed that are not supported by the ATA/IDE standard (www.wikipedia.org 2006).

SATA offers performance as high as 3.0 Gbit/sec per device with the current specification. SATA uses only 4 signal lines, allowing for much more compact (and less expensive) cables compared with PATA. SATA drives may be plugged into Serial Attached SCSI (SAS) controllers and communicate on the same physical cable as native SAS disks. SAS disks, however, may not be plugged into a SATA controller (www.wikipedia.org 2006).

First-generation SATA interfaces, also known as SATA/150 or (unofficially) as SATA 1, communicate at a rate of 1.5 gigabits per second (Gbit/s). SATA/150 devices were scarcely faster than legacy parallel ATA/133 devices. So a 3 Gbit/s signaling rate was added to the Physical layer (PHY layer), effectively doubling data throughput from 150 MB/s to 300 MB/s. SATA/300's transfer rate is expected to satisfy drive throughput requirements for some time, as the fastest desktop hard disks barely saturate a SATA/150 link. This is why a SATA data cable rated for 1.5 Gbit/s will currently handle second generation, SATA 3.0 Gbit/s sustained and burst data transfers without any loss of performance. SATA's road map includes plans for a 6.0 Gbit/s standard. In current PCs, SATA 3.0 Gbit/s already greatly exceeds the sustainable (non-burst) transfer rate of even the best hard disks (www.wikipedia.org 2006).

The SATA protocol, like the PATA protocol, defines the contents of the data messages between the devices and must be at least levels 1 and 7 in the OSI model. SATA supports communication only between two nodes, while PATA supports communication between three nodes (computer, master and slave). The PATA protocol must also support layer 2 in the OSI model.

Figure 18.1: The 7 pins SATA connector.

Data Message

The main data message is called a FIS (Frame Information Structure) containing the following fields:

FIS type	1 byte
Port Multipliert	1 byte
Reserved	1 byte
Type	1 byte; 0=control, 1=command
Command	1 byte
Register	1 byte
Data	18 bytes
Reserved	4 bytes

No information about the address since the protocol is not supporting layer 2 in the OSI model.

Cable and connectors

The SATA standard defines a data cable with seven conductors (3 grounds and 4 active data lines in two pairs) and 8 mm wide wafer connectors on each end. SATA cables can be up to 1 m (39 in) long, and connect one motherboard socket to one hard drive. Figure 18.1 shows the 7 pins SATA connector and the connection of the 7 pins cable is:

Pin#	Function
1	Ground
2	A+
3	A-
4	Ground
5	B-
6	B+
7	Ground

The connector in Figure 18.1 has only signal and ground, no power. A SATA connector with power can be used in addition to the SATA connector, called SATA/power connector. This is a 15 pins connector containing pins for 3.3V, 5V, 12V, and ground.

External SATA

External SATA, eSATA, defines separate cables, connectors, and revised electrical requirements for external applications/connections. The maximum cable length of eSATA is 2 m.

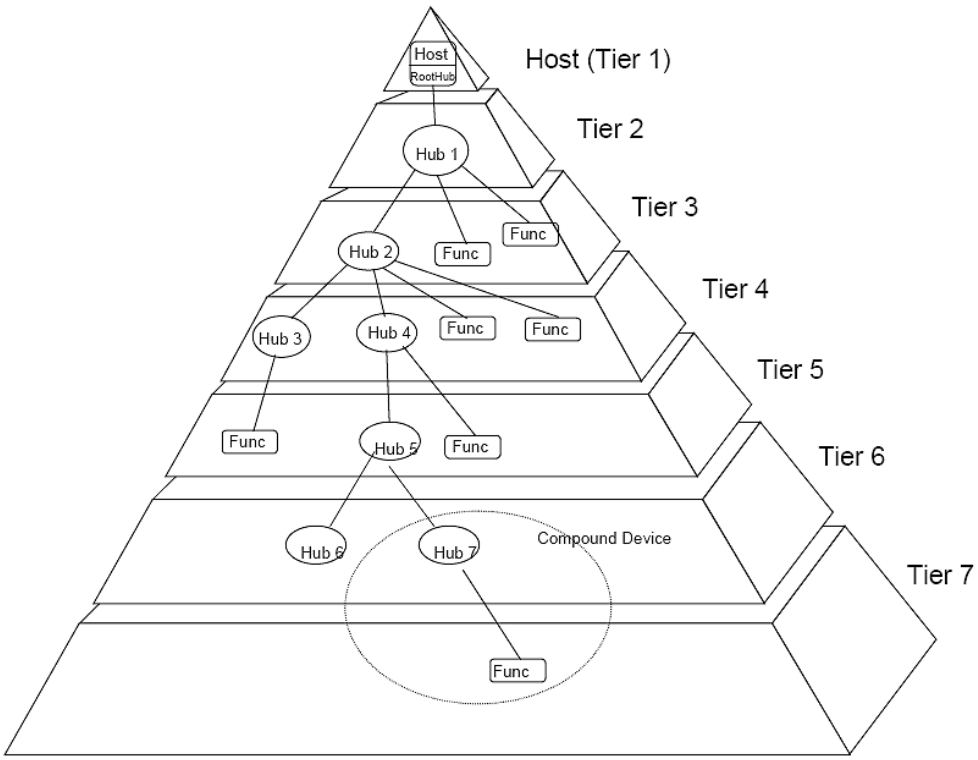


Figure 18.2: USB bus topology (Gamey 2000).

18.1.2 USB

Introduction

Universal Serial Bus (USB) is a serial bus standard being more and more common on computers, often in stead of the $RS-232C$ communication port. Normally is the maximum numbers of $RS-232C$ ports are 4 giving a maximum 4 external serial devices using the $RS-232C$ ports on a PC. The maximum number of external devices for the USB is 127. The connection is a root tree with a main star hub in each tier for the next tier. The maximum number of tiers is 7 giving that the maximum number of hubs that can be cascade is 6. The connection is shown in Figure 18.2 showing the host, a set of hubs and a set of devices (nodes).

A computer can have a number of USB roots where each root can connect up to 127 devices. The USB devices is either a hub device or a function device, an example is shown in Figure 18.3.

The topology is a tiered star topology where each hub is the center of a star connecting to a function device or additional hubs. Each device connected to a root is assigned an address at power up or when connected to the bus. The root is the master of the bus and manages the data flow. Available time is divided into segments, called micro frames, where each device on the bus can transmit data.

The benefits of USB are:

- ease of use; one interface for many devices, automatic configuration, no settings, simple cables, often no extra power supply required,
- fast and reliable data transfer; from low speed on cheaper and/or simple devices up to high speed on more expensive devices,
- reliability; hardware design and the data transfer protocol,
- low cost; using inexpensive components and cables,
- low power consumption; power saving circuits and automatically power down logic.

There exist no perfect interface or protocol and the downside of USB are:

- lack of support for older hardware,

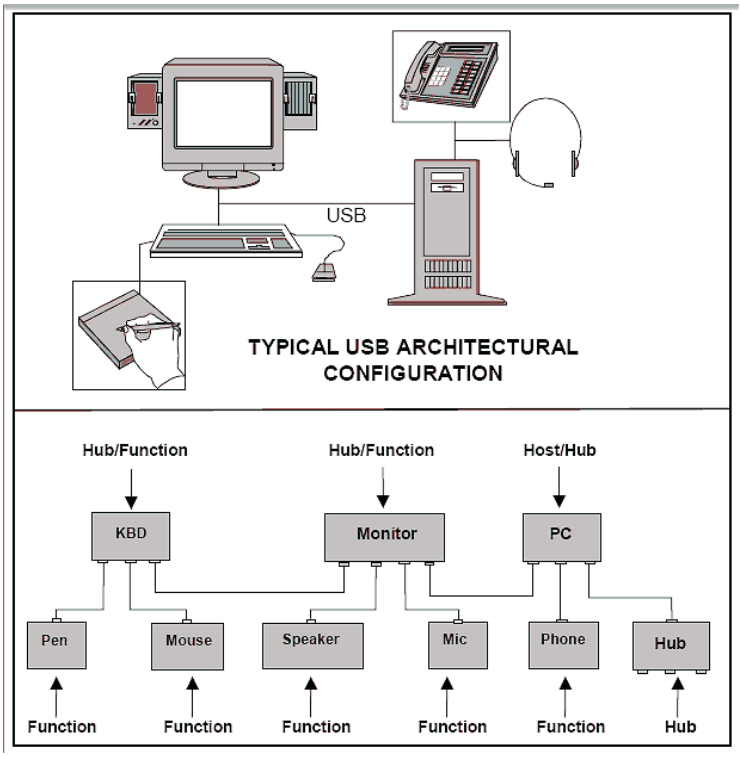


Figure 18.3: Hubs in a Desktop Computer Environment (Gamey 2000).

- speed limits; other protocols like FireWire are faster,
- distance limits; designed as a desktop bus,
- master slave communication where only the master can communicate with the slaves, the slaves can not communicate with each other directly.

Communication

The communication speed of the USB is depending on the version, but is backward compatible. The speeds are according to the table:

USB	**Version**	**Speed Maximum Theoretical**	**Maximum Practice**	**Text Description**
1996	1.0	1.5 Mbit/s		low speed
1998	1.1	12 Mbit/s		full speed
2000	2.0	480 Mbit/s	320 Mbit/s	hi speed
2008	3.0	4.8 Gbit/s	3.2 Gbit/s	super speed

Using USB 2.0 the maximum speed is 480 Mbit/s giving a theoretical speed of 60 MByte/s, but in practice the maximum speed will be $\approx$ 40 MByte/s. The USB is speed backward compatible meaning that any version 1.1 devices can be connected to a version 3.0 port, but the maximum speed will still be 12 Mbit/s. CRC−5 is used for error detection.

The Low Speed rate of 1.5 Mbit/s (187.5 kB/s) that is mostly used for Human Interface Devices (HID) such as keyboards, mice, and joysticks.

USB communication is in form of packages (messages) from the root device to any USB device. The different types of packages are:

1. Handshake package; The responses are *ACK*, *NACK* (retry), *STALL* (device error), *NYED* (v. 2: Not yet completed), *ERR* (v.2: comm. error),

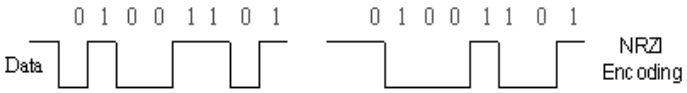


Figure 18.4: The NRZI encoding.

Figure 18.5: The USB A and B connectors, B connector to the left and A connector to the right.

2. token package; Always from the host, never from an USB device, *IN* (response from a device), *OUT* (followed by a data package), *SETUP* (followed by setup data), *PING* (v.2: Ready for a *OUT/DATA* package?).

3. data package; Two data packages, *DATO0* and *DATA1*, and the receiver must handshake the data package. If it receives another *DATAx* packet of the same type, it is acknowledged but ignored as a duplicate. Only a *DATAx* packet of the opposite type is actually received.

USB devices support 4 data flow types:

1. control; synchronous communication,

2. interrupt, asynchronous communication,

3. bulk; combination of control and interrupt, can be used for larger amount of data,

4. isochronous; used commonly for streaming data types like video and audio. It is used in processes where data must be delivered within certain time constraints, video and audio must be delivered together with correct time synchronization.

The USB signal uses the NRZI [Non-Return-to-Zero-Inverted Encoding]: A '0' is encoded as no change in the level. However a '1' is encoded depending on the current state of the line. If the current state is '0' [low] the '1' will be encoded as a high, if the current state is '1' [high] the '1' will be encoded as a low, and is shown in Figure 18.4. NRZI should give better noise immunity then no encoding.

Cabling and connectors

The USB connectors are named A and B and the A connector is the DTE connector and the B connector is the DCE connector. The connectors are shown in Figure 18.5 and Figure 18.6.

The maximum length of the USB cable according to the specification is 5m (but is normally also working up to 10m). The cable contains 4 physical connections, the power and the communication as shown in Figure 18.7.

Note that the cable has only one serial line for communication (D+/D-) meaning that the communication can only be simplex or half duplex. USB is using half duplex with the root as the master device.

Version 3.0 is using full duplex and requires two serial lines. Using USB in SuperSpeed mode (v.3.0), full-duplex signaling occurs over two differential pairs separate from the non-SuperSpeed differential pair. The result of this solution is that USB 3.0 cables is containing two wires for power and ground, two wires for non-SuperSpeed data, four wires for SuperSpeed data, and a shield. A USB 3.0 cable contains nine wires while a USB 1.1 or USB 2.0 cable only contains four wires.

The voltage on the USB is $5V$ and the maximum power is 2.5W from each port giving a maximum current of 500 mA. Initially, a device is only allowed to draw 100 mA. It may request more current from the upstream device in units of 100 mA up to a maximum of 500 mA. In practice, most ports will

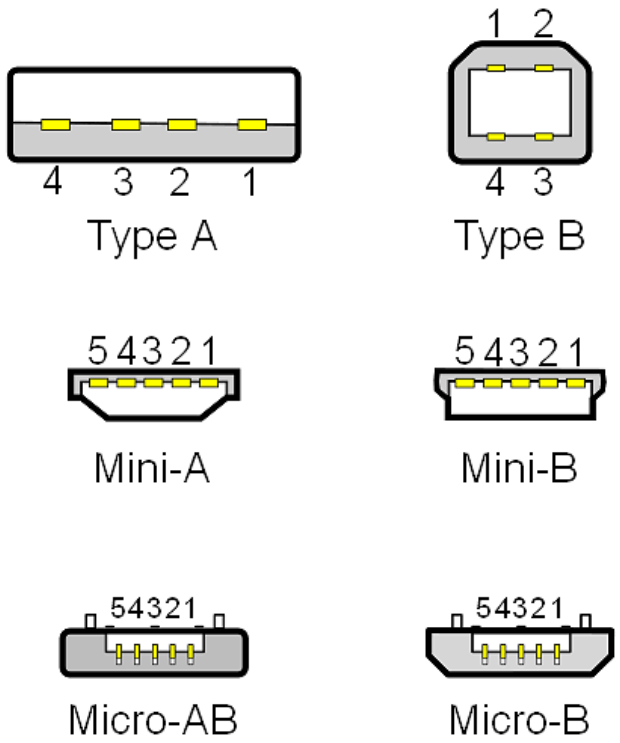


Figure 18.6: The pinning of the USB connectors, a A connector type to the left and a B connector type to the right.

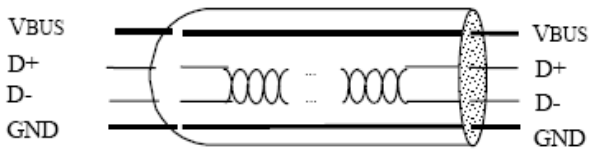


Figure 18.7: USB cable (Gamey 2000).

deliver the full 500 mA or more before shutting down power, even if the device hasn't requested it or even identified itself.

The USB 3.0 has extended the maximum current for one device to be 150 mA and the maximum current from a port is 900 mA.

The pinning for USB 1.1 and USB 2.0 A and B connectors are (see also Figure 18.6 and Figure 18.7) :

Pin	**Function**	**Color**	**Note**
1	VBUS / VCC	Red	4.75V to 5.25V
2	D-	White	Data -
3	D+	Green	Data +
4	GND	Black	
Shell	Shield		

The pinning for USB 1.1 and USB 2.0 mini/micro A and B connectors are (see also Figure 18.6 and Figure 18.7) :

Pin	**Function**	**Color**	**Note**
1	VBUS / VCC	Red	4.75V to 5.25V
2	D-	White	
3	D+	Green	
4	ID	(None)	
5	GND	Black	
Shell	Shield		

The ID pin permits the distinction of a mini/micro plug as; Type A: connected to Ground, Type B: not connected.

USB signals are transmitted on a twisted pair of data cables, labelled D+ and D-. These collectively use half-duplex differential signalling to combat the effects of electromagnetic noise on longer lines. D+ and D- usually operate together; they are not separate simplex connections. Transmitted signal levels are 0.0–0.3 volts for low and 2.8–3.6 volts for high.

Version 3.0 has extended the speed and power usage, and a new type of connector is used. However the connectors can be interchanged since the extra connection are located between the original pins. The pinings are:

Pin	**Function**	**Color**	**Note**
1	Power	Red	
2	D-	White	
3	D+	Green	
4	GND	Black	Ground for Power return
5	SSRX-	Blue	SuperSpeed receiver differential pair (ARX- or BTX-)
6	SSRX+	Yellow	SuperSpeed receiver differential pair (ARX+ or BTX+)
7	GND	N/A	Ground for signal return
8	SSTX-	Purple	SuperSpeed transmitter differential pair (ATX- or BRX-)
9	SSTX+	Orange	SuperSpeed transmitter differential pair (ATX+ or BRX+)
			The USB 3.0 "Powered-B" connector has two additional pins
10	DPower		Power provided by device (Powered-B only)
11	DGND		Ground return to DPWR (Powered-B only)

The advantages of using USB instead of RS-232C and RS-485 will then be:

1. define DTE and DCE connectors,
2. power included in the connectors,
3. more devices can be connected,
4. standard software gives more services according to the OSI model,

The disadvantage is the length of the cable, at least compared to the RS-485 standard. The USB standard has often support for more layers of the OSI model, while the RS-232C and RS-485 standards are only supported by the physical layer, layer 1. The USB standard will support at least the physical layer, the datalink layer and the application layer, the comparison is shown in the table

OSI model	**RS-232C / RS-485**	**USB**
Application		Application
Presentation		
Session		
Transport		
Network		
Datalink		Datalink
Physical	Physical	Physical

USB devices

1. HUB: Can connect several USB devices to a computer,
2. Switch: Can connect several computers to a USB device.
3. Wireless: Can connect several wireless USB devices to a computer.

USB advises

1. Prefer a USB HUB with external power,
2. A USB 1.1 HUB is working on a USB 2.0 port, but satisfy only the USB 1.1 specification,
3. In practice only 2 USB HUBs can be cascade connected,
4. Some USB devices are working only on the root hub, not on external HUBs.

USB summary

Type	USB 1.0	USB 1.1	USB 2.0	USB 3.0	NOTE
Transfer rate	1.5 Mbit/s	12 Mbit/s	480 Mbit/s	4.8 Gbit/s	
Length	3 m	5 m	5 m	3 m	(10m)
Devices	< 128	< 128	< 128	< 128	

Wireless USB (WUSB)

Wireless USB is a short-range, high-bandwidth wireless radio communication protocol created by the Wireless USB Promoter Group. Wireless USB is sometimes abbreviated as "WUS", although the USB Implementers Forum discourages this practice and instead prefers to call the technology "Certified Wireless US". Wireless USB is based on the WiMedia Alliance's Ultra-WideBand (UWB) common radio platform, which is capable of sending 480 Mbit/s at distances up to 3 meters and 110 Mbit/s at up to 10 meters. It was designed to operate in the 3.1 to 10.6 GHz frequency range, although local regulatory policies may restrict the legal operating range for any given country.

A wireless USB connection will use a star connection instead of the tree connection used by the cable USB connection. The number of devices will still be 127 but using a star connection there will be no hubs. A new device called *Device Wire Adapter* (DWA) will be used instead of the hubs. The DWA will communicate with a *Host Wire Adapter* (HWA). The usage of WUSB together with HWA and DWA is shown in Figure 18.8.

Note that the WUSB must support more layers in the OSI model than USB due to more services for network communication (transport and session layers).

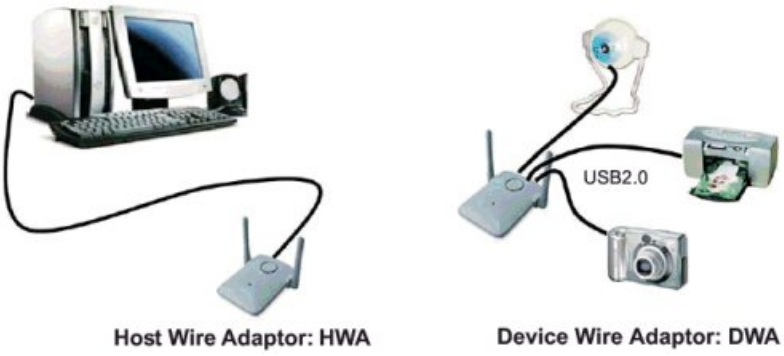


Figure 18.8: Wireless USB with a Host Wire Adapter (HWA) and a Device Wire Adapter (DWA).

USB Versions

The different versions of USB is shown in the following table:

USB Year	Version	Speed	Current	Communication
1996	1.0	1.5 Mbit/s	100 mA	half duplex
1998	1.1	12 Mbit/s	100 mA	half duplex
2000	2.0	480 Mbit/s	100 mA	half duplex
2008	3.0	4.8 Gbit/s	150 mA	full duplex

18.1.3 FireWire

FireWire is a serial bus protocol and defined in the IEEE 1394 standard, from 1995 (1994?), and used a lot in Macintosh computers and digital camcorders. The IEEE 1394a standard from 1998 starts showing up on PC as an alternative to the USB.

FireWire Specifications:

	Firewire				
Name	**S400**	**S800**	**S800T**	**S1600**	**S3200**
IEEE standard	1394a	1394b	1394c	1394b	1394b
Year (approximate)	1998	2002	2006	2008	2008
Transfer rates	400 Mbps	800 Mbps	800 Mbps	1.6Gbps	3.2Gbps
Max. distance between devices	4.5m	100m	100m	100m	100m
Devices	63 / 1023	63 / 1023	63 / 1023	63 / 1023	63 / 1023
Cable	Firewire	Firewire	CAT5e	Firewire	Firewire

Like the USB, FireWire 400 also has 3 different speeds being 98304, 196608, and 393216 kb/s (kilobits per second), from now on abbreviated as 100, 200, and 400 Mb/s respectively. These high speeds (up to 30 times faster than USB 1.1) are achieved by good shielding of the individual wire pairs and additional shielding of the entire cable.

Up to 63 devices can be connected to each other, without the need for any hubs. Additionally, so-called bridge devices can be used to connect up to 1023 of these buses with each other. A chain of devices (so-called daisy chain) can be up to 16 devices long. With the maximum cable length between two devices restricted to 4.5 meters, this sums up to a total distance of 72 meters. Loop connections are not allowed and stop the operation of the bus.

1394 is a true peer-to-peer based bus, meaning that every device can independently communicate with every other device on the bus without the need for a dedicated host (like a computer). IEEE 1394 is self-configuring in which a unique identification number is assigned to every device automatically. Also, 1394 devices are "hot pluggable", meaning that they can be plugged on and off during operation of the bus ("Plug & Play").

Figure 18.9: FireWire connectors.

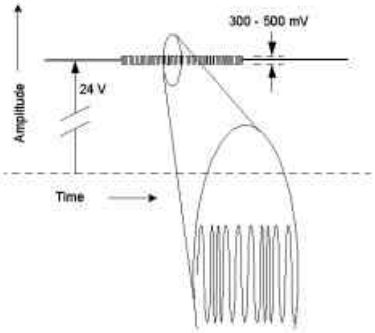


Figure 18.10: The HART communication on a $4-20$ mA transmission line (www.analogservices.com: feb-2010).

IEEE 1394 supports two transfer modes: Asynchronous and isochronous transfer. In the asynchronous transfer mode, data is guaranteed to be delivered to the receiver, where in case there are some disturbances in the bus, the data is sent again. However, the latency of delivery is unknown, because the bus might be busy with other operations. Contrary to that, the isochronous transfer mode allows for guaranteed access to the bus after a certain time interval. This way, devices that need to send their data in real-time can ensure a continuous stream of data, an important feature in Audio/Video (A/V) applications.

FireWire cables come in two flavours: either 4-pin or 6-pin. While the 4-pin version uses all 4 wires for the transfer of data, the 6-pin version has two additional wires to supply power (8-40V, up to 1.5A) to connected devices. There exist special adapter cables to connect 4-pin and 6-pin cables to each other.

The four data wires are arranged in two twisted pairs that are separately shielded to minimize distortions due to environmental noise. Additionally, the entire cable is shielded again, resulting in very good protection which makes the high transfer rates possible.

18.1.4 Hart

The HART (Highway Addressable Remote Transducer)[1] protocol is a communication technology used with smart process instrumentation, and used a lot in the industry. HART is field-proven, easy to use, and provides two-way digital communication simultaneously with the $4-20$ mA analog signaling (current loop) used by traditional instrumentation equipment. HART is digital communication being added to the analog $4-20$ mA physical communication, showing in Figure 18.10.

The $4-20$ mA transmission line contains only one analog signal, by adding HART more information can be transmitted between the device and the computer. The $4-20$ mA solution is used a lot in the industry and the HART Protocol provides a unique communication solution that is backward compatible with the installed base of instrumentation in use today. This backward compatibility ensures that investments in existing cabling and current control strategies will remain secure well into the future.

The digital signal is ± 0.5 mA bit stream added to the analog signal and a filter is needed to separate the analog and digital signal. The analog signal is in the frequency range $0-10$ Hz and the HART signal (bit stream) is in the frequency range $1200-2200$ Hz. A logic 1 is transmitted as a 1200 Hz frequency signal and a logic 0 is transmitted as a 2200 Hz frequency signal. The HART filter is a two part filter, one low pass filter for the analog signal and a high pass filter for the HART communication. The cut off

[1] HART® is a registered trademark of the HART Communication Foundation

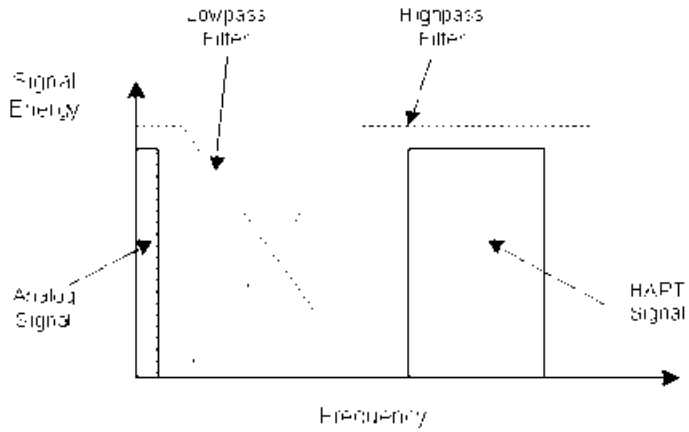


Figure 18.11: The HART filter for separating the low frequency analog signal and the high frequency digital signal (www.analogservices.com: feb-2010).

	OSI Layer	Function	HART
7	Application	Provides the User with Network Capable Applications	Provides the User with Network Capable Applications
6	Presentation	Converts Application Data Between Network and Local Machine Formats	
5	Session	Connection Management Services for Applications	
4	Transport	Provides Network Independent, Transparent Message Transfer	
3	Network	End to End Routing of Packets. Resolving Network Addresses	
2	Data Link	Establishes Data Packet Structure, Framing, Error Detection, Bus Arbitration	A Binary, Byte Oriented, Token Passing, Master / Slave Protocol.
1	Physical	Mechanical / Electrical Connection. Transmits Raw Bit Stream	Simultaneous Analog & Digital Signaling. Normal 4-20mA Copper Wiring

Figure 18.12: The OSI model used for the HART protocol.

frequency is in the range of $400 - 800$ Hz. The HART filter frequency ranges are shown in Figure 18.11. The speed of the digital signal is 1200 bits/second. The coding used is FSK (Frequency Shift Keying).

Protocol

Designed to compliment traditional 4-20mA analog signalling at frequencies usually under 10 Hz, the HART Protocol supports two way digital communications for process measurement and control devices. Specification of the HART protocol is based largely on the OSI 7-Layer Communication Model, see Figure 18.12.

The HART protocol specifications directly address 3 layers in the OSI model: the Physical, Data Link and Application Layers. The Physical Layer connects devices together and communicates a bit-stream from one device to another. It is concerned with the mechanical and electrical properties of the connection and the medium (the copper wire cable) connecting the devices.

While the Physical Layer transmits the bit stream, the Data Link Layer is responsible for reliably transferring that data across the channel. It organizes the raw bit stream into packets (framing), adds error detection codes to the data stream and performs Media Access Control (MAC) to insure orderly access to the communication channel by both master and slave devices.

The bit stream is organized into 8-bit bytes that are further grouped into messages. Each byte is transmitted as a character consisting of 11 bits; a startbit, 8 databits, a parity bit and a stop bit. A HART transaction consists of a master command and a slave response. Media access consists of token passing between the devices connected to the channel. The passing of the token is implied by the actual

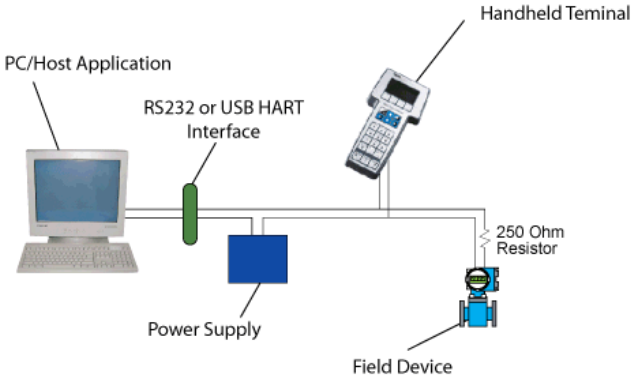


Figure 18.13: An overview of a HART system with a computer and a Field Device with a HART protocol.

message transmitted. Timers are used to bound the period between transactions. Once the timer expires, control of the channel is relinquished by the owner of the token. The HART message consists of the following characters:

Part of message	**Length (bytes)**	**Purpose**
Preamble	5 to 20	Synchronization and Carrier detect
Start Delimiter	1	Synchronization and shows which is master
Address	1 or 5	Choose slave and indicate the master
Command	1	Tell slave what to do
Number of data bytes	1	Number of bytes before the checksum
Status	Master = 0 Slave = 2	Slave indicate the health and status of command
Data	0 to 253	
Checksum	1	Error control

Several HART devices can be connected to the same pair of cables giving a network of HART devices. None of the devices can use the wire for analog value, only digital values can be transmitted. The protocol is a master/slave protocol and the OSI layers 3 and 4 is used in addition to the layers shown in Figure 18.12. The network layer provides routing, end to end security and transport services. The transport layer provides a successful end to end communication between the devices on the network. The address field consists of either a short address (4 bits) or a long address (38 bits).

The Application Layer defines the commands, responses, data types and status reporting supported by the Protocol. In addition, there are certain conventions in HART (for example how to trim the loop current) that are also considered part of the Application Layer. An overview of a HART system is shown in Figure 18.13.

Communication

HART communication is FSK (frequency-shift-keying), with a frequency of 1200 Hz representing a binary one and a frequency of 2200 Hz representing a binary zero. These frequencies are well above the analog signalling frequency range of 0 to 10 Hz, so that the HART and analog signals are separated in frequency and ideally do not interfere with each other. The HART signal is typically isolated with a high-pass filter having a cut-off frequency in the range of 400 Hz to 800 Hz. The analog signal is similarly isolated with a low-pass filter. The bit rate is 1200 bps.

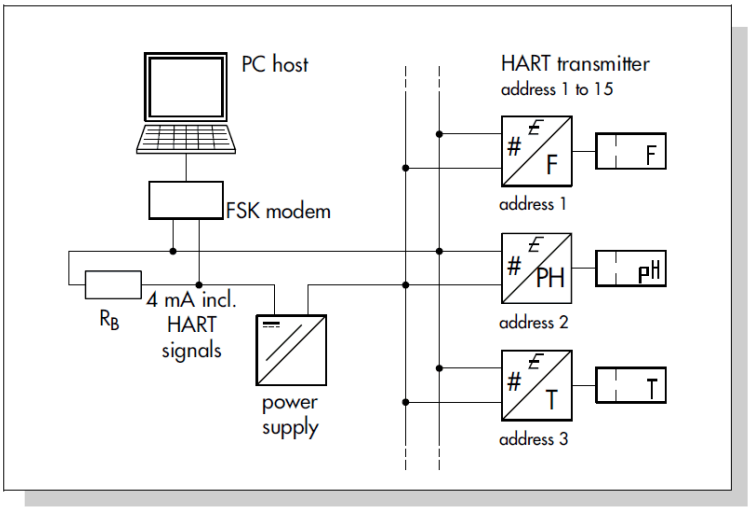


Figure 18.14: A HART multidrop connection (www.samson.de: feb-12).

Multidrop connection

In a multi-drop connection several HART instruments are connected together and all HART instruments are only using 4 mA. A unique address is assigned to each instrument and only the HART protocol is used for communication. The token are used for controlling the access to the communication medium. The protocol allow for maximum 15 devices in multidrop connection. Figure 18.14 shows the connection of several HART devices.

Wireless HART

WirelessHart (HART 7 Specification) enables capabilities for communication with wireless intelligent field devices. The WirelessHART consists of a gateway between a HART nework and wirelessHART, and a set of access points communication with the wireless field devices. The wireless network can also consists of Wireless adapters connecting wired HART field devices to the wireless network. The gateway also includes a network manager and a security manager for the allowing wireless filed devices to connect to network.

HART IP

HART-IP has been added to the HART specification to allow for better system integration. This specification serves as better connection between the HART protocol and the Internet Protocol (IP). HART-IP seems as a possible protocol for integration with the Internet of Things (IoT) in the future. HART-IP works over standard Ethernet and can co-exist with other IP protocols, both wired and wireless. The HART-IP application layer (Layer #7) is based on the same commands as used for the $4-20$ mA/HART and WirelesHART protocols. As the HART-IP is based on IP it will work with the standard infrastructure components such as switches, routers, cables and connectors.

18.2 Network Communication Protocol

Network communication protocols are also using serial lines for communication, but normally not the same limitation in speed capabilities and interconections as for serial lines communication. Such protocols support layer OSI layer #1, the physical layer, layer #2 supporting more then two devices on the same bus (physical addressing), layer #3 for routing and logical addressing, layer #4 for flow control of data and entity identification, and layer #7 if a common set of commands are available. The protocol defines the capabilities, both the advantages and disadvantages. Remember, no perfect protocol exists.

18.2.1 Modbus

Modbus is a serial communication protocol published by Modicon in 1979 for use with its programmable logic controllers (PLCs). Today Modbus is communications protocol used a lot in the industry, both as a serial line protocol and a network (TCP/IP) protocol. The main reasons for the use of Modbus as a "standard" industry protocol are:

1. it is openly published and royalty-free,
2. easy to implement,
3. defining a set of commands for the layer #7 of the OSI model moving raw bits or words between nodes without placing many restrictions on any vendors.

Modbus allows for communication between many devices connected to the same network, for example a system that measures several temperature and humidity parameters from different devices, and communicate the results to a common computer (server). Modbus can be used to connect a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems. Versions of the Modbus protocol exist for serial port (Modbus RTU and Modbus ASCII) and Ethernet (Modbus TCP).

Modbus	**Connection**	**Data Representation**	**Devices**	**243 as Value**
ASCII	Serial	ASCII	247	2,4,3
RTU	Serial	Binary	247	0xF3
TCP	Network	Binary	Ethernet	0xF3

Two variants exist, with different representations of numerical data and slightly different protocol details. Modbus RTU is a compact, binary representation of the data. Modbus ASCII is human readable, and more verbose. Both of these variants use serial communication. The RTU format follows the commands/data with a cyclic redundancy check checksum, while the ASCII format uses a longitudinal redundancy check checksum. Nodes configured for the RTU variant will not communicate with nodes set for ASCII, and the reverse. Modbus/TCP is very similar to Modbus RTU, but transmits the protocol packets within TCP/IP data packets. A TCP/IP connection must be used for Modbus TCP. The different Modbus standards are shown in Figure 18.15. This Figure also shows the reference to the OSI model where ASCII and RTU is supporting layer #1, layer #2, and layer #7. The Modbus TCP is also supporting the layer #3 to layer #5 functions. The Modbus Plus is a proprietary protocol based on token passing between the devices.

Layer #1 is defining the physical connections (wire/wireless, electrical signals, connectors++). Layer #2 is defining multidevice bus, meaning that several devices can be connected to the same bus. Since layer #3 functions are not defined, means that several segments can not be interconnected as not routing exists. The protocol defines a master controlling the communication bus. Layer #7 is defining a set of commands and/or message structures for exchanging information between the devices. Note that Modbus RTU supports only layer #2 and #7 and used RS-485 as layer #1, and Modbus TCP supports only layer #7 and using TCP/IP for layer #3 and #4.

Each device intended to communicate using Modbus is given a unique address, either a specific slave address ($1 - 247$) for Modbus RTU or a TCP/IP address for Modbus TCP. The master will send a message to any of the slaves, and the slave will respond to the message. A Modbus command contains the Modbus address of the device it is intended for. Only the intended device will act on the command, even though all other devices on the bus will receive it. The maximum number of Modbus RTU addresses is 247 giving the maximum number of Modus devices on the bus. The protocol is a master/slave protocol meaning that one device (master, address=0) must request all the other devices for data or informations (slaves, address=1 to 247).

A Modbus command consists of a frame and the Modbus frame is divided into an Application Data Unit (ADU) and a Protocol Data Unit (PDU). The PDU contains the same information for all the Modbus protocols meaning that the Modbus RTU and Modbus TCP can be interconnected. However, the ADU will be different. The ADU contains the PDU pluss additional information for the specfic protocol. Figure 18.16 shows the general Modbus frame on the top, the frame for the Modbus RTU in the middle, and the frame for the Modbus TCP frame at the bottom.

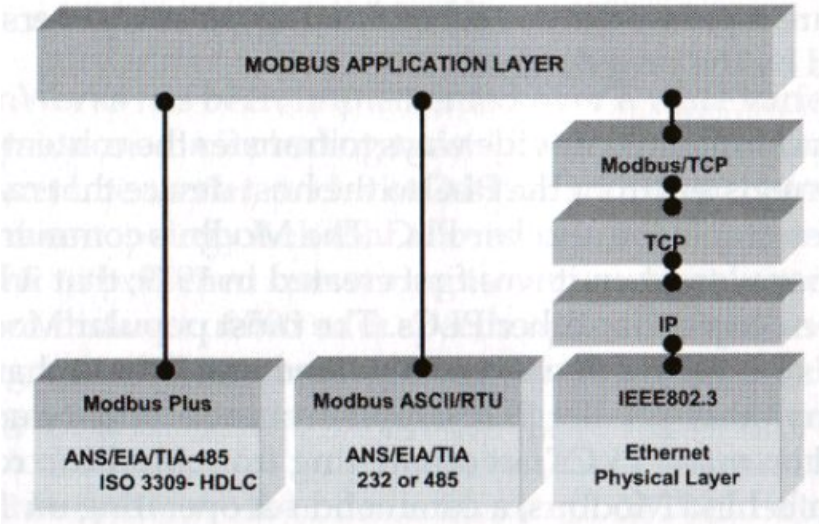


Figure 18.15: The different Modbus types with reference to the OSI model. The Modbus Plus is a proprietary protocol based on token passing (Caro 2004).

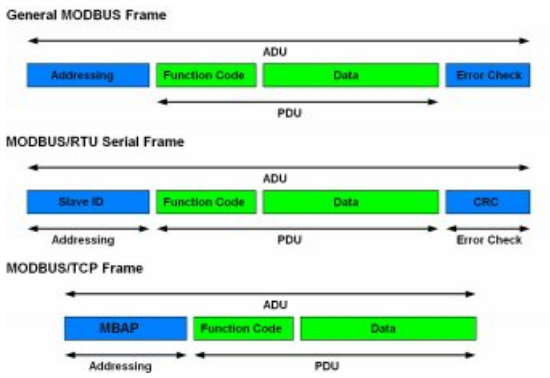


Figure 18.16: The Application Data Unit (ADU) and Protocol Data Unit (PDU) in a Modbus mesage frame.

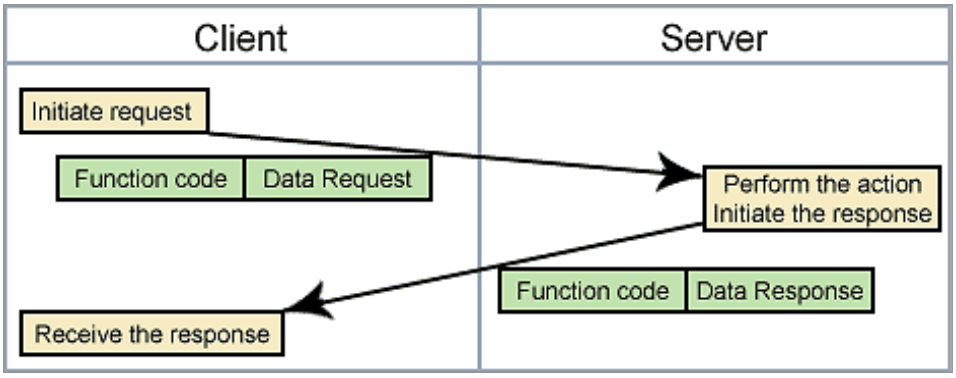


Figure 18.17: The Modbus communication model using a function code as the command and a data field.

Modbus TCP do not contain the error check as this is already a part of the TCP protocol, and no addressing as this is part of the IP protocol. However the Modbus Application Protocol (MBAP) header contains additional information if several TCP clients are communication with the TCP server at the same time. Note that the PDU is handling the total data range of the registers from 0 to 65535, while the ADU is handling the Modbus model from 1 to N where N is the maximum number of registers for a specific application. Important to note the offset of one register address between the ADU and PDU.

A Modbus RTU message consists of the following fields where the Function Code and Data is the PDU:

Address Function Code Data ErrorCheck

The Data field will consists of register address information, normally the start register address and a number of continuous register addresses. The data field contains the start register address and number when requesting information, and the contents of these registers when receiving the information.

An examples: The master is transmitting:

5 3 246 4 "CRC"

meaning that the slave address = 5, function code = 3, start register address = 246 and the number of register addresses = 4. The last byte, errorcheck, is only used for ASCII and RTU, not TCP. The slave address 5 may respond:

5 3 101 202 34567 22113 "CRC"

meaning that the answer is from slave 5, the answer is from function code 3, register address $[246] = 101$, $[247] = 202, [248] = 34567$ and $[249] = 22113$. Modbus is a "Big Endian" protocol measning that the high byte is always handled first. If the value of a register is $0x1234$, the first byte will be $0x12$ then the second byte ($0x34$).

The concept of the Modbus protocol is to synchronize the register address buffers between the nodes. The buffer consists of a set of registers, one set of bit registers and one set of 16 bits registers. The bit registers are used for binary signals and the 16 bits registers are used for analog values represented normally by a short integer value. The binary registers are divided into input (read only) and output (read and write) bits, the output bits are called the coil registers and the input bits are called the input discrete. The 16 bits registers are both input (read only) and output (read and write), and the output registers are called the holding registers. The Modbus protocol will update the registers and by configuration different registers the values can be updated on different nodes. Note that Modbus is only supporting single bits and 16 bits integrers, no floating point numbers.

Figure 18.17 shows the structure of Modbus communication. A function code is used as the command indicating the requested response from the remote node. A data field is used if additional data is needed for the function code. A client will request some sort of information from a data server and the data server will response with the requested function code and data. Note that the Figure is refering to client and server, being the same as slave and master. The Figure is refering to the Modbus slave as the server,

why? In this case the master/slave refers to the information, the Modbus slave is the master (or server) of information, but a slave of the communication line. Always define the purpose of the master or the server. Even the Modbus specification is confusing about the master/slave and client/server concepts.

Since the Modbus is based on a request and response message, it will take some time to get an answer from the data server. A good advice is to group the register in a continuous range so all the registers can be read by only one message. Note that the maximum number of registers in one message is less than 128 giving a maximum message length on about 256 bytes.

The function code is a single byte and is normally a command to read or change the value in one or more registers. The most used function codes are:

Function code	Description	Datasize
01	Read coil status	output bit
02	Read input status	input bit
03	Read holding registers	16 bits
04	Read input registers	16 bits
05	Force single coil	output bit
06	Preset single register	16 bits
07	Read exception status	bit
15	Force multiple coils	output 16 bits
16	Preset multiple registers	16 bits
17	Report slave ID	address

The basic Modbus commands can instruct a RTU to change a value in one of its registers, as well as commanding the device to send back one or more values contained in its registers. The Modbus protocol is based on a register model where the contents of these registers are exchanged between the master and slave devices. The maximum number of regsiters is 256.The registers are:

Table	Classic Terminology	Register Address	Characteristics
Discrete outputs	Coils	00000	16-bits quantity, alterable by an application program, read-write
Discrete inputs	Inputs	10000	Single bit, provided by an I/O system, read-only
Input registers	Input registers	30000	16-bits quantity, provided by an I/O system, read-only
Output registers	Holding registers	40000	Single bit, alterable by an application program, read-write

The master node will request data or write data to the slave, meaning that the register contents on the slave will be used for syncronization between the master and the slave. Note that the Mdobus master is the communication master, not the data master. Normally the master slave is the data master. The Modbus TCP is dealing with TCP client and TCP server, being the data master. Be aware of the master/slave and client/server concepts when working with the Modbus protocol.

One example can be a set of sensors connected to the slave and the slave will update the local registers from the sensors. The master node will use the Modbus protocol to read updated sensor values from the slave. The registers are 16 bits using 8 or 16 bits values as characters or short values. The scaling of the values has to be done outside the Modbus protocol, and must agreed between thetransmitter and receiver of data. If one of the values is a temperature using a sensor with an accuracy of $\pm 0.5\,^{\circ}$C, one soluting can be to multiply the tempearture with 10 before storing the value. A temperature of $16.5\,^{\circ}$C will be stored and read as 160.

The function code can indicate only one address or a range of addresses. Almost all implementations have variations from the official standard. Different varieties may not communicate correctly between different suppliers equipment. Some of the most common variations are:

1. Data Types
   (a) Floating Point IEEE
   (b) 32 bit integer

(c) 8 bit data
(d) mixed data types
(e) bit fields in integers
(f) multipliers to change data to/from integer. 10, 100, 1000, 256 ...

2. Protocol extensions

   (a) 16 bit slave addresses
   (b) 32 bit data size (1 address = 32 bits of data returned.)
   (c) word swapped data.

The serial Modbus will send and receive data in frames, also called messages. A frame is a buffer with the bytes (or characters) describing the information needed for the communication between two nodes. These frames are:

ASCII	**mode**				
Start	Address	Function	Data	Check	End
1 byte	2 bytes	2 bytes	n bytes	2 bytes	CR+LF

RTU	**mode**				
Start	Address	Function	Data	Check	End
$>$ 28 bits	8 bits	8 bits	$nx8$ bits	16 bits	$>$ 28 bits

All protocols will be designed for a special purpose and always contain some limitations. This mean that no protocol will be perfect for any purpose, and this is the reason why a huge number of protocols exist. Modbus was designed in the late 1970's to communicate to programmable logic controllers (PLC). The number of data types is limited to those understood by PLCs at the time. The limitations for this protocol are:

1. Large binary objects are not supported.
2. No standard way exists for a node to find the description of a data object, for example, to determine if a register value represents a temperature between 30 and 175 degrees.
3. Since Modbus is a master/slave protocol, there is no way for a field device to "report by exception" - the master node must routinely poll each field device, and look for changes in the data. This consumes bandwidth and network time in applications where bandwidth may be expensive, such as over a low-bit-rate radio link.
4. Modbus RTU is restricted to addressing 247 devices on one data link, which limits the number of field devices that may be connected to a master station,
5. The maximum number of registers in one message is limited to 128 registers,
6. The Modbus standard is only supporting bit and 16 bits integers.

18.2.2 XML

Extensible Markup Language (XML) is a text based protocol with focus on data, not on presentation. The XML based protocol consists of several parts:

1. an envelope defining what is in the message and how to process it,
2. a set of encoding rules for expressing instances of application and defined datatypes,
3. a convention for representing procedure calls and responses,
4. structuring of the data with focus of information, not presentation.

Using <CMD> as a start code for command information, </CMD> will be the end code for command information.

Using XML instead of the Modbus for accessing a set of registers, the answer from a node may look like:

<REGISTERS>
<REGREADADR>10, 11, 12, 13, 45, 56</REGREADADR>
<REGADRVALUE>14, 12.4, 15, 14.8, 57, 2876, 4</REGADRVALUE>
</REGISTERS>

The receiver must know the connection between the register address and tags, and the units for each address. Part of these information can be included in a XML message. Using XML for sending values from a tag (Boiler_1) in a process system, including unit information, may look like:

<PLANT_x>
<BOILER_1>
<TEMPERATURE> 67.8 </TEMPERATURE>
<TEMPUNIT>C</TEMPUNIT>
<LEVEL> 12056.8 </LEVEL>
<LEVUNIT>MM</LEVUNIT>
<PRESSURE> 243.2 </PRESSURE>
<PRESSUNIT>MBAR</PRESSUNIT>
</BOILER_1>
</PLANT_x>

The receiver must know the XML commands when using XML.

18.2.3 SOAP

Simple Object Access Protocol (SOAP) relies on eXtensible Markup Language (XML) as its message format, and is normally based on other Application Layer protocols.

The SOAP specification defines the messaging framework which consists of (www.wikipedia.org 2010):

1. The SOAP processing model defining the rules for processing a SOAP message,
2. The SOAP extensibility model defining the concepts of SOAP features and SOAP modules,
3. The SOAP underlying protocol binding framework describing the rules for defining a binding to an underlying protocol that can be used for exchanging SOAP messages between SOAP nodes,
4. The SOAP message construct defining the structure of a SOAP message.

18.2.4 OPC

Open Process Control (OPC) is a standard based on server/client communication with focus on process control.

18.2.5 IoT

Internet of Thing (IoT) devices are based on the TCP/IP protocol and is not supporting the application layer (layer 7) in the OSI model. An additional protocol must be added to be able to communicate with the IoT devices. The most used solution is the use of a proprietor protocol and a cloud service, however several standard IoT protocols are available. These protocols are OPC UA, Modbus TCP, Bluetooth, Zigbee, MQTT (Message Queue Telemetry Transport), CoAP (Constrained Application Protocol), DDS (Data Distribution Service), XMPP (Extensible Messaging and Presence Protocol) and NFC (Near Field Communication) to mention some. One reason for so many protocols is that an IoT device can have several functions and no single protocol can support all the functionallity.

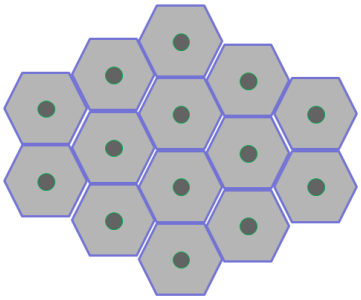


Figure 18.18: The cells in a mobile network. Each cell has an antenna connected to a receiver and transmitter controling that specific cell zone.

18.2.6 MQTT

MQTT (Message Queue Telemtry Transport) protocol is designed as an lightweighted publish/subscribe messaging protocol. MQTT consists of a server, called the broker, and a set of clients that will communicate with the broker. The broker is the communication centre, and any client can connect to the broker, publish and subscribe to messages.

An MQTT broker is able to buffer messages can't be dispatched to MQTT clients that are not connected, an useful feature for unreliable networks. For this reason is MQTT supporting three types of QoS (quality of services) delivery messages: 0) at most once, 1) at least once, and 2) exactly once.

MQTT brokers and clients are available as open source solutions for several programming languages so implementing the protocol into an existing system is possible. The published data consists of a hierarchical topic paths represented as plain text strings. An example, with reference to the XML example above, can be: Plant_x/Boiler_1/Temperature.

18.3 Mobile Phones

The mobile communication is based on a set with base stations where the mobile phone will always be connected to the nearest base station. GSM (Global System for Mobile Communications) is the second generation of mobile phone standards and used a lot world wide. GSM is a cellular network meaning that mobile phones connect to the network by searching for cells in the immediate vicinity. The cellular network is based cells, overlapping circular zones covering a geographical area as shown in Figure 18.18. Each cell contains a receiver, a transmitter, and an antenna, marked as the dark point in the middle of each cell. Each cell contains the Base Transceiving Station (BTS). A number of BTS are connected to a Base Station Controller (BSC). Figure 18.19 shows an overview of the GSM network. The phones or the terminals in the network are called mobile stations. The mobile station is using a Subscriber Identity Module (SIM) card to identify the user. A PIN code can be used to get a unique identification number of the SIM, called International Mobile Subscriber Identity (IMSI). The phones or the terminals are identified by a unique 15 digit number called International Mobile Equipment Identity (IMEI). The IMEI code can be shown by entering the code **#06#*, and often the code is located underneath the battery. The SIM card can be used to identify the user and the IMEI code can be used to identify the mobile station.

The mobile station will be connected to the nearest base station (BTS). If the mobile station is moving they will always switch to the nearest base station (BTS). The mobile station will reserve information from the closest base stations and select the base station based on the informations from these base stations. Different operators will use different base stations, or they can cooperate reusing the same base stations.

Each cell will have one base station (BTS). The base station controller (BSC) will administrate a set of base stations, and set of base station controllers will be administrated by a mobile switching centre (MSC). The mobile switching centre has a set of registers to validate the users of the mobile stations and should also know the IMEI code of the mobile stations. The mobile switching centre is also the connection point between the mobile network and the landline network.

The registers are used to know the locations of the users. Any call from the landline network the network provider needs to know the location of the subscriber and the status of the mobile station (ON/OFF). These informations are stored in the Visitor Location Register (VLR) as shown in Figure

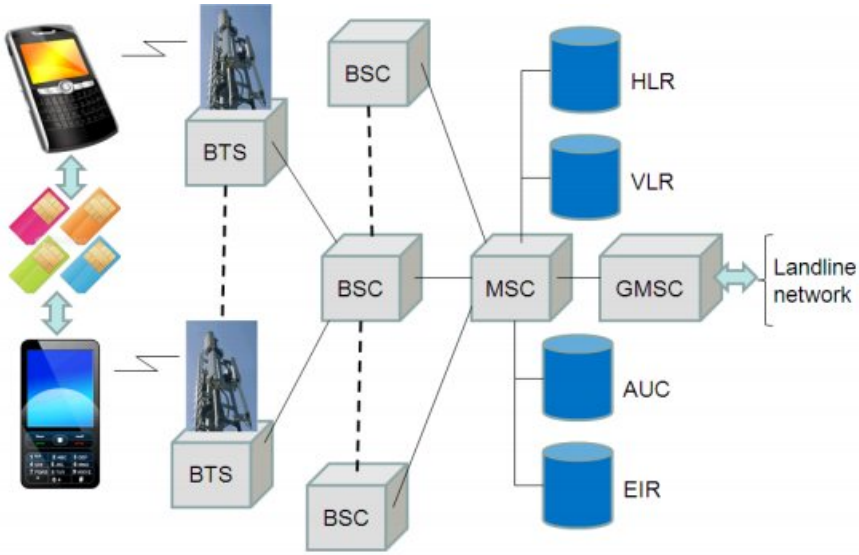


Figure 18.19: The structure of a GSM network with Base Transceiving Stations (BTS), Base Station Controllers (BSC), and Mobile Switching Centre (MSC).

18.19. The Home Location Register (HLR) contains the name of the subscribers, the number, and the required serves in the network. The Authentication Centre (AUC) contains the information as whether the subscriber is able to use the network and subscriber related keys amongst other things required during authentication. The Equipment Identity Register (EIR) contains information about the mobile stations using the network. This register can also keep track of stolen mobile stations using the IMEI code, but this register is optional.

Turning on a mobile station, the mobile station will try to connect to the nearest base station. The network provider will have the identification data of the user and the mobile station, and the base station. If the user can connect to the network, the MSC will use this information to update the registers. This connection is done when the mobile station is turned on, not when making or receiving a call or SMS.

The GSM standard consists of GSM900 and GSM1800 using the frequencies:

GSM	**Frequency** MHz
900	Up: 880 – 915
	Down: 925 – 960
1800	Up 1710 – 1785
	Down: 1805 – 1880

The third generation is called 3G or UMTS (UMTS2100) using the frequencies 1920 MHz – 1980 MHz for up communication and 2110 – 2170 MHz for down communication. The fourth generation is called LTE (Long Term Evolution) and using the frequencies 2500 MHz – 2570 MHz for up communication and 2620 – 2690 MHz for down communication.

GPRS (General Packet Radio Service) and EDGE (Enhanced Data Rates for GSM Evolution) are a few of the standards for data communication in mobile network.

18.3.1 Short message service (SMS)

Short message service (SMS) was added to the GSM (Group System Mobile) specification from 1981 and can be used for communication with mobile phones.

The infrastructure of SMS is shown in Figure 18.20, showing a mobile phone, a set of base stations for communication with the mobile phones, the message center (MC) and Short Message Entity (SME). SME are devices for generating and/or reading SMS. The mobile phone will also be a SME as long as it is using the GSM standard. The MC is also called SMSC, the SMS control center.

The SMS is using a store and forward service, meaning:

1. The SMS message will always be stored in the MC before sending to a SME,

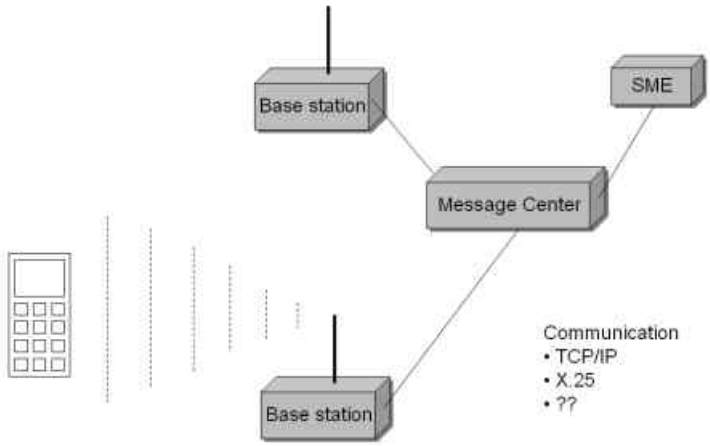


Figure 18.20: The mobile infrastructure for SMS.

2. The SME will only communicate with the MC, never other SME's.

The sequence of sending a SMS message:

1. The SME will send the SMS message to MC,
2. The MC will store the SMS message,
3. The MC will check if it has contact with the receiver SME,
   (a) The MC will send the SMS message if it has contact with the receiver SME,
   (b) The MC will store the SMS message for a specified time if no contact with the SME.

Addressing

The mobile network consists of a set of physical and locical numbers, both to identify the user and the mobile device. The mobile network is the interconnections of base stations dividing the area into cells. A cell may have one or several base stations and the mobile device will select the base station with the best signal strength and quality. This is the reason why mobile devices also are called cellphones.

IMEI number All mobile phones connected to the GSM or UMTS networks have a unique number, an International Mobile Equipment Identity (IMEI) number. This number consists of 15 or 17 numbers and just like the MAC address for Ethernet equipment, this number is specific for the phone device. This number is connected to the phone, not the SIM card or the user. The number can be displayed by typing the code

$$*\#06\#$$

on the phone. This number is used by the GSM, UTMS, and SMS as the address of the device, not the phone number as the phone numbers can be the same in different countries. This number can also be used to track stolen phones so it may be a good advace to make a notice of this number. The Central equipment Identity Register (CEIR) are used for keeping track of stolen or not wanted mobile phones.

SIM card The SIM card (subscriber identification module) securely stores the service-subscriber key (IMSI) used to identify a subscriber. The SIM card contains:

1. its unique serial number,
2. internationally unique number of the mobile user (International Mobile Subscriber Identity; IMSI),
3. security authentication and ciphering information,
4. temporary information related to the local network,

5. a list of the services the user has access to,
6. two passwords (PIN for usual use and PUK for unlocking).

The SIM card allows users to change phones by simply removing the SIM card from one mobile phone and inserting it into another mobile phone or broadband telephony device. The IMEI contains information about the mobile phone (hardware device), while the SIM card contains information only about the user.

SMS message

The SMS message consists of different fields, like:

1. The number of the receiver; mandatory,
2. The number of the transmitter; mandatory,
3. The message; optional,
4. The validity period; optional,
5. Status report request; optional,
6. Other optional parameters depending on the protocol.

The optional fields can be mandatory in some of the protocol used for communication between the message center and the SME.

Validity period The validity period parameter defines how long the message should be stored in the SMSC (MC) if no contact with the receiving SME. This option can be programmed on some mobile phones and PDAs.

The validity period can be according to the table:

Value	Period	Note
-1	Making only one attempt to deliver the message	
$0-143$	5 minutes intervals for up to 12 hours	
$144-167$	30 minutes intervals from 12 hours to 24 hours	
$168-196$	1 days intervals from 2 days to 30 days	
$197-255$	1 weeks intervals from 5 weeks to 63 weeks	

Status report request The status report request is a deliver status message when the receiving SME has received the SMS message. This option can be set on many some mobile phones. The status value will be according to the table:

Value	Status	Note
1	Temporary error	
2	Validity Period Expired	
4	Delivery Failed	
8	Delivery Successful	
16	Message Cancelled	
32	Message deleted by the operator	

The status report request very often contains more options/settings on newer mobile phones.

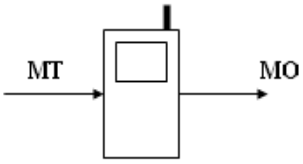


Figure 18.21: Mobile Originated (MO) and Mobile Terminated (MT) communication.

18.4 SMS message types

1. MO; Mobile Originated, SMS message from the mobile phone (SME), see Figure 18.21,
2. MT; Mobile Terminated; SMS message to the mobile phone (SME), see Figure 18.21,
3. PUSH; Request message from the mobile phone (SME),
4. POP; Subscriptions of information from a server (mobile contractor),
5. SMS messages
   (a) Text messages, short or long,
   (b) Binary messages, short or long.

Binary SMS messages will be (to mention some):

1. Logo,
2. Ringtone,
3. Picture message,
4. FLASH message,
5. Other types ..

Text message The text message must contain the receiving phone number and the message text as a minimum, and is organized as:

Phone number	**Message**
Maximum 14 numbers	Maximum 160 characters

Long messages will be divided in several SMS message with maximum 160 characters in each message.

Binary message The text message must contain the receiving phone number, a header and the binary message as a minimum, and is organized as:

Phone number	**Header**	**Message**
Maximum 14 numbers	Header code	Maxium length of header + message is 280 characters

The header code for a short binary message will be:

$$< LH > 0504 < DPORT1 >< OPORT1 >< OPORT2 >$$

and the header code for a long binary message will be:

$$< LH > 0003 < REFNO >< NOFSUB >< SEQNO > 0504 < DPORT1 >< OPORT1 >< OPORT2 >$$

where:

Code	Meaning	Note
<LH>	Length of header: Short=06: Long=0B	
0003	00=Submessages + 3 control bytes	
<REFNO>	Referance number module 256	
<NOFSUB>	Number of submessages: 0-255	
<SEQNO>	Sequence of current submessage	
0504	05=IEI + 4 control bytes	
<DPORT>	Destination port (Ringtone, Logo, Etc)	
<OPORT>	Originator port	

SMSC protocols

The SME and SMSC can communicate using among others the internet, but need a protocol. The protocols between the SMSC and SME can one of the following:

1. CIMD;
   (a) Computer Interface to Message Distribution,
2. EMI/UCP;
   (a) External machine Interface / Universal Computer Protocol,
   (b) An extension of the ETSI European Radio Messaging System.
3. SMPP
   (a) Short Message Peer-to-peer Protocol.
4. YAP/CPA/MSB
   (a) SMPP "look'n like",
5. Soniq MQ
6. SEMA2000

SMS usage

SMS can be used for any remote or movable systems using the already existing infrastructure of SMS system. Examples of such systems can be:

- alarm system for moving devices like operators, cars, etc.,
- important informations from monitoring and control system,
- important informations from alarm systems,
- position information from moving devices like trains and busses.

Mobile phone communication

The physical layers for reading or writing a SMS or MMS messages from/to a the mobile phone:

1. IR communication (IrDA)(Infrared link),
2. Serial communication (RS-232C),
3. Bluetooth (wireless),
4. WLAN (wireless).

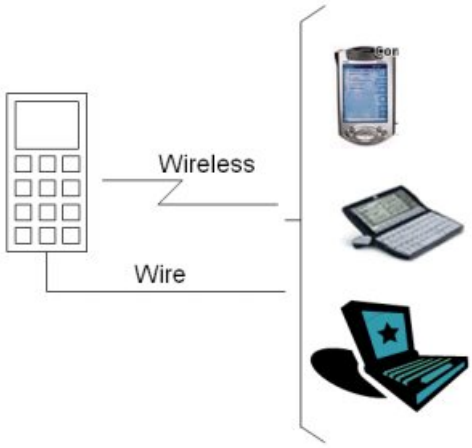


Figure 18.22: Devices connected to the mobile phone for reading and/or writing SMS messages. Popular protocols are Hayes command set (AT command set) or TCP/IP based.

Figure 18.22 shows some devices that can be used together with a mobile phone for reading and/or writing SMS messages. In a control system a phone can be used to send important messages from the control system to the guard on duty meaning that the guard do not need to be present all the time (24x7 systems).

The Hayes command set can be used for communication to the mobile phone, the Hayes command set is also called AT commands. Common AT commands include AT+CMGS (send message), AT+CMSS (send message from storage), AT+CMGL (list messages) and AT+CMGR (read message). Send a message:

$$AT + CMGS = \text{“}PhoneNumber'' < CR > \text{“}Messagetext'' < EOM >$$

where <CR> is the carrige Retrun code ($0x0D$) and <EOM> is the End Of Message code.

MMS (Multimedia Message System)

Multimedia Messaging Service (MMS) is a standard for telephony messaging systems that allows sending messages that include multimedia objects (images, audio, video, rich text) and not just text as in Short Message Service (SMS) (www.wikipedia.org 2006). MMS should not be confused with Enhanced Messaging Service (EMS), which is simply Short Message Service (SMS) with additional payload capabilities, allowing a mobile phone to send and receive messages that have special text formatting (such as bold or color), animations, pictures, icons, sound effects, and special ring tones (www.wikipedia.org 2006). The MMS is a combination of text and multimedia, combining text and binary messages.

Security

In a cell a false base station (BTS) can be installed for monitoring the traffic, jamming and hacking. The false base station may just monitor the traffic before routing the traffic to the real base station. This means that neither the user nor the telecommunication operator is able to detect the illegal monitoring of the traffic (both speech and data).

Roaming

Roaming is a term defining how an end user of a mobile phone can access other mobile networks when outside the home network's geographical coverage area. The end user must have a customer relationship to a specific mobile network, also named the home network. Other mobile networks is often named the visited network. Roaming is used for access of mobile services like making and receiving calls and SMS/MMS, and use data services like Internet. Roaming can be both national and international roaming. National roaming means that a mobile provider may use the mobile network from several operators for its traffic, without including any base stations or other equipment into the network for the operators. International roaming means that the home network and visited network are in different countries giving mobile customer access to mobile networks also when traveling abroad.

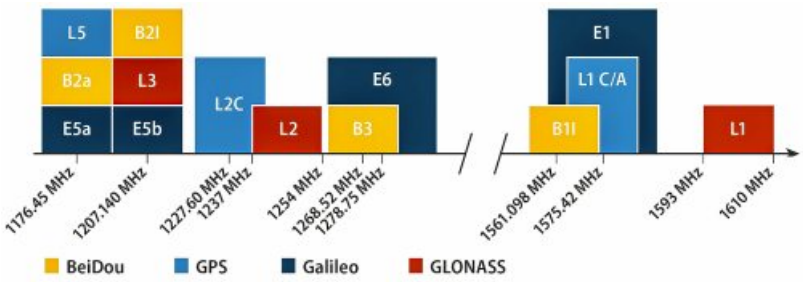


Figure 18.23: The frequences used by the different satellite navigation systems (TU-2019).

18.5 GPS

The global positioning system (GPS) is a satellite navigation system with up to 32(24) satellites orbiting the earth in 12 hours orbits (paths). The satellites are distributed such that, on average, there are 12 satellites visible in each hemisphere (horizon). The satellites are time synchronized using an onboard atomic clock, and continuously transmitting the time and other information. The transmission is using a common carrier, but each satellite has its own pseudo random number sequence. An another encrypted carrier is available for military purposes using a "super secret GPS password". The non encrypted carrier can be used by any GPS receiver.

The GPS receiver is a satellite receiver which listens for the signals and measures the time of the arrival, comparing it to the GPS time when the data was sent. This information provides a pseudo-range to each satellite that is received, and the range is used to compute the position. Four satellite signals are needed to get a three-dimensional position.

GPS receivers typically listen for all 12 satellites that should be in the hemisphere where the receiver is located and uses the strongest signals to compute the position. The more satellites it uses, the better will the accuracy be. Triangulation is used for calculating the position, based on an accurate time signal. By knowing the receiving and transmitting times, the distance can be estimated for each of the satellites.

In order to know which satellite is where, will all satellites from time to time transmit a "database" containing the orbital data for all the satellites. The GPS receiver should store this database in nonvolatile memory to preserve this information between power cycles. The accuracy will be from 30 m down to 1 m depending on the number of satellites and noise. A normal accuracy will be less than 10 m.

At a cold start (power up) must the GPS receiver cycles through all possible satellites codes until it receives a satellite signal with sufficient signal to noise ratio to download the orbital information and current time. The cold starts can take up to 15 minutes, depending on the GPS receivers. A warn start, for example a watchdog reset, or moved a large distance, the startup time is often less than 2 minutes. Note that the communication is only one way, simplex.

The protocol used for many GPS receivers is the NMEA-0183 (National Marine Electronics Association). This protocol uses a simple ASCII, serial communication protocol that defines how data is transmitted in a "sentence" from one transmitter to one receiver at a time. The communication paramters are 1) 4800 baud rate (typical), 2) 8 databits, 3) no parity, 4) one stop bit, and 4) no handshake. Typical CPS commands are GGA, GLL, GSA, GSV, RMC and VTG. The format of a typical NMEA-0183 command is:

$$\$Command, DataField1, DataField2, .., DataFieldN, *NN$$

where $ is the start of the message, the type of command defines the number of datafields, and * is the field for the checksum, represented as a two-digit hexadecimal number. The checksum is the bitwise exclusive OR of ASCII codes of all characters between the $ and * in the message.

There exists several satellite navigation systems, the most know is GPS, the US system. GLONASS is the Russian alternative based on 24 satellites, Galileo is the European alternative based on 11 active satelittes, BeiDou Navigation Satellite System (BDS) is the Chinese alternative (active in Asia only), and IRNSS is the Indian alternative (active in Asia only). The European system is a civilian system only, while both GPS and GLOSNASS are both military and sivilian systems. The satelittes for Galileo are located in a higher path giving, together with other properties, a higher resolution. The current accuracy for Galileo is about eight meters. These satelitte systems are using the frequency range from about 1.1 GHz to about 1.7 GHz for transmission, as shown in Figure 18.23.

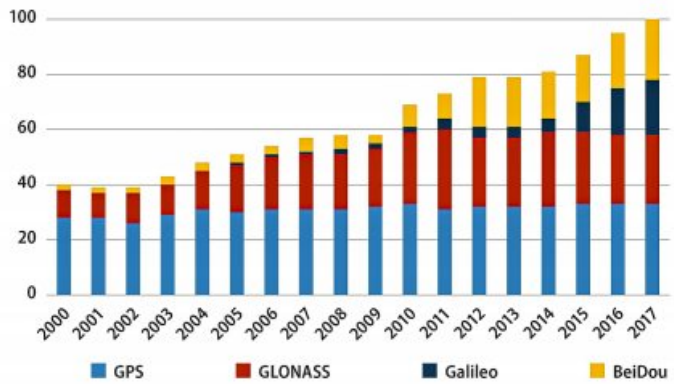


Figure 18.24: The number of satellites installed for the different navigation systems (TU-2019).

The number of installed satellites for the different navigation systems is shown in Figure 18.24.

Today a lot of systems are using GPS and several systems have services based on GPS. Note that GPS signals can be tampered with, either by jamming or spoofing. Jamming is normally not a problem because the transmitter seems to be dead and no more information is received. Spoofing is a larger challenge as the contents of the message can be changed. By changing the time information the received will estimate a wrong position. As the satellite signals are very weak, it is possible to receive these weak signals, alter the timing information and transmit the new information as a stronger signal. Any GPS receiver will use the strongest signal. A good choice will be to use data fusion, combine different systems to estimate and verify any positions.

18.6 Data Files

Data Files can also be used for communication between applications, also known as interprocess communication (IPC). One application can write the information to a file at a specific location on the disk or in the network, and another application can check the same location at fixed time intervals. Different type of data formats exists, both ASCII and binary. The data format of these data files will be the exchange protocol between the applications. The most used formats for data files is the comma separated values (CSV) format and the XML format. The XML format is explained in section 18.2.2. The JavaScript Object Notation (JSON) format can also be used in text files as JSON is much like the XML format. JSON will often separate longer fields in more specific fields than the XML format making it more suitable for defining the contents of objects.

18.6.1 CSV

The comma separated values (CSV) format of data files consists of a separator and a set of values. The format will be a set of lines, all with ASCII text, and a new line for a new set of values. The format may include a header line with information about each column in the file. If a value is missing, the field will often be empty or have a fixed value. However this has to be part of the protocol specification. The separator started out as the comma, as the name is describing. Today different types of separator can be used, and most be part of the protcol specification if selectable. The most used separators are comma (,), colon (:), semi colon (;), space ( ) and tab ( ). An example of a CSV file with semi colon as the separator :

```
Day;Month;Year;Hour;Minute;Second;SampleTimeObj;[TempOutsideSouth]:RawDAQVal#1;RawUnitVal#1;
TempOutsideSouth;TagStatus#1(OK=0);[RelHumOutsideSouth];RawDAQVal#2;RawUnitVal#2;..
28;10;2014;19;7;22;28okt14_190722;0,644;11,799;11,828;0,000;3,814;100,992;100,560;0,00;
0,014;0,293;0,291;0,000;4,640;92,787;92,768;..
28;10;2014;19;17;22;28okt14_191722;0,644;11,799;11,818;0,000;3,838;101,056;100,453;0,000;
0,015;0,303;0,296;0,000;4,639;92,794;92,770;..
28;10;2014;19;27;22;28okt14_192722;0,643;11,764;11,787;0,000;3,798;100,748;100,732;0,000;
0,015;0,303;0,302;0,000;4,647;92,945;92,944;..
```

.....

Note: Each line in this CSV file is cut/ truncated (indicated with two dots) after 22 fields, the number of fields in such a file can be unlimited.

18.7 Network

In network a protocol supporting layer #7 is needed for extracting valid information from the protocol. Using TCP/IP will only give you large buffer of data, no structure or information about the data unless you have knowledge about the structure before sending the data (a proprietary protocol). Modbus TCP is such a protocol.

18.7.1 Internet Of Things (IoT)

Internet of Things (IoT) means having embedded devices with their own IP addresses. Using the new IP6 protocol makes it possible to include internet in all type of devices. The challange is the cpu power, the memory space and the protocol. Internet will require more cpu power and the TCP/IP protocol must be included. IoT will also require a protocol on top of TCP/IP to support layer #7 in the OSI model. Devices from different vendors are meant to be able to exchange information, and the only way is to include a protocol supporting layer #7. Devices can also support several protocols for the layers below the TCP/IP protocol like WiFi, Bluetooth, Bluetooth Low Energy, and ZigBee.

A cloud service or a layer #7 protocol must be used for communication with the IoT device. A cloud service will communicate with the IoT device using the layer #7 protocol. This protocol can be either an open or a proprietor (closed) protocol, while the users will use the cloud sevice for getting information from the devices.

18.8 Buildings

Building Automation Systems (BAS) and Building Energy Management Systems (BEMS) are system used for monitoring and control of buildings. These systems must be distributed as a building consists of several floors and rooms, and therefor use a communication protocol. Several protocols exist like FND, Profibus, LonWorks, EIBus and Batibus (Levermore 2000).

18.9 Select a Communication Protocol

The first dilemma will be to choose between a standard protocol or design a new protocol. Choosing a standard protocol can make your life much easier, especially if you need to go with a complex protocol such as TCP/IP, USB, CANopen, or DeviceNet to mention quite a few. There exists a lot of good libraries and open-source code for such protocol that can be integrated into your system. The important part is to focus on layer 7, the application layer, which defines how much communication support you need to include in your own source code. If you are going to make you own protocol, try to reuse as much as possible from the underlying layers by using an existing protocol. If you need to design a protocol for the application layer you should consider using protocols like TCP/IP or RS-485 for underlying layers if several nodes are needed. TCP/IP can be used for network protocols and RS-485 for serial communication protocols. Always think about the communication protocol as a layered structure based on bricks where each brick can be changed or replaced without affecting the other bricks. The number of bricks in your protocol depend on the type of service you need, normally you need at least the physical layer (layer #1) and the application layer (layer #7). The next layers support depends on the node addressing management and the communication flow management.

The protocol can be either based on ASCII type or binary type data transfer, depending on the nodes. The binary type of data is more efficient as the data is stored more compact. Also evaluate the Endianess management when dealing with different hardware architecture of the nodes in the network (big or little endian data formats). Due to different CPU architectures the high and low part of a data word (16 bits) will be handled different.

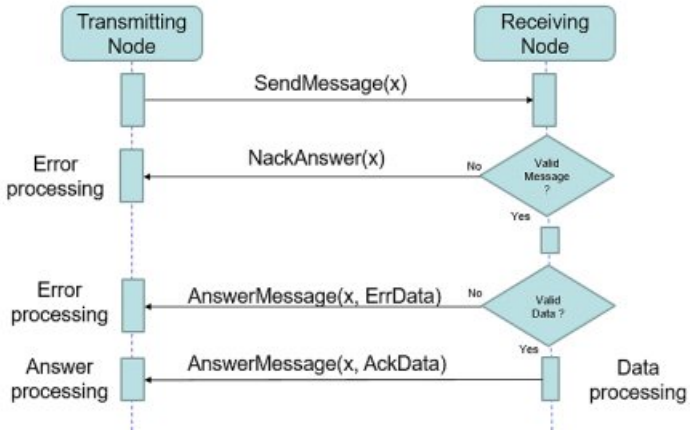


Figure 18.25: The communication flow between two nodes, and possible types of answers.

18.9.1 Node addressing

Often a communication protocol is used for communication between more then two nodes in a network, and some sort of node addressing is necessary. These addresses have to be unique, and can be related to either the node (physical device) or to the data (a data-sharing abstraction model). The address information is normally an early information in the protocol so it is easy for each node to first check the node addressing information to be able to reject the rest of the message. A good solution can be to use an address from an underlying layer or brick like the MAC or TCP address if using a TCP/IP underlying protocol.

18.9.2 Message or data oriented protocol

Message oriented protocols are used if you need a lot of specialized commands with specific parameters. This can be an useful protocol if only a small amount of the data needs to be transmitted between the nodes. In this case will the nodes need a set of commands to describe the type of data included and the type of operation for the data. The protocol will end up with a lot of different type of messages to handle different type of commands and different type information.

Data oriented protocol consists of a shorter list of messages, one for every of data to transfer, and some control messages.

18.9.3 Communication flow

The network architecture defines how many nodes that will be communicating, how these nodes are connected, the relationship between the nodes, and defining the rules for exchanging data between the nodes. Many industrial networks use master/slave and multi master architectures, while many PC networks use the client/server architecture. Figure 18.25 shows the communication between two nodes, and all of the possible types of answer. Only the SendMessage(x) can be used on data that is updated often. If any error, just wait for the next message for updated data. In this case a time-out alarm must be included for the receiving node to give an alarm if no data has been received for a specific time (for instance 120 seconds).

The next option is just to send a short message, NackAnswer(x), to indicate an error on the last message, message x. The transmitting node must then retransmit message x. Further on the message is OK, but the data may be corrupt. An answer message may be used to indicate if the received data was OK or corrupted. The number of feed backs for each message will complicate the message flow for the protocol, but will also make the protocol more robust. In environment with a lot of factors affecting the data communication, a robust protocol should be considered. A robust protocol will use more time for the communication resource for each package to be transmitted.

The answering time will also depend on the network architecture and in a larger network the communication time will depend on the number and type of nodes. Figure 18.26 shows a larger network where the messages are handled by several nodes before reaching the destination node. Both the transmitting time and the answering times will depend on the network architecture, and allow for some delay in these

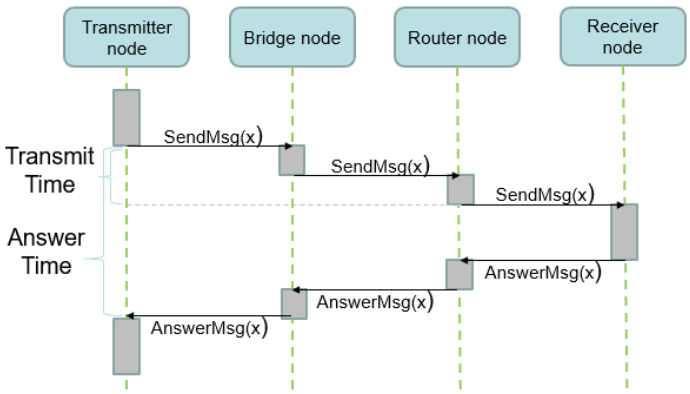


Figure 18.26: Communication messages route between two nodes in a larger network with sub networks.

times as the time will depend on the computing time in each node. The answer time can be several seconds, take this into considerations when designing the network architecture. The network architecture will also define the type of distributed system, however the timing will depend mainly on the software solutions for each of the nodes in the network.

The timing will also depend on the data encoding and the type of error detection in the network.

18.9.4 Data encoding

The data message is sent as a frame through the communication line. However, the frame must be defined a starting part and an ending part so the nodes know when receiving a new message and the length of the message. In addition, the frame also need some information about the addressing and some type of error checking. A normal way of making such a frame is:

Start of Frame Addressing Message End of Frame Error Checking

The “Start of Frame” may also include the length of the “Message”, or it may be the first part of the “Message”. The first part of the “Message” can also include a command field. The “Addressing” and ”Message” parts can be divided into:

RxAddres TxAddress Length Command Data

The data encoding can be either ASCII or Binary based, however it is important that the "End of Frame” indicator can not exist in these fields. The normal way is to use a specific sequence of bytes for the “Start of Frame” and “End of Frame” and then add an extra byte if the same sequence exist in any of the fields between these sequences. ASCII encoding will make a larger frame using more time for transmitting and receiving, but makes it easier to handle and debug.

18.9.5 Error Detection or Correction

Parity, check sums and cyclic redundancy check (CRC) can be used for detecting errors in a data stream. Parity is adding a bit to every byte, but can only detect singe bit errors. Check sum is to make a 8 bits (256) or 16 bits (65535) sum of all the bytes in the frame, and add this sum to the end of the frame (“Error Checking” field). The receiving node will sum up all the bytes and compare the sum with the last byte or bytes in the frame. CRCs are more sophisticated error detection algorithms. CRC will add more information to the “Error Checking” field. Error Correction will require even more information in this field as the information can be used to correct errors in the frame.

18.9.6 Data Flow

In many network there will be different computing power for different nodes. In such network architecture either a maximum lengths of the frame must be defined, or a data flow control must be implemented. A data flow control can be based on a fixed length in the network, or some sort of sequence that can be sent from the receiver node to the transmitter node when starting and stopping the communication.

Chapter 19

Network Architecture

The network architecture will normally not either be correct or not correct, it is more about how good the architecture is. The clue is to make a structure based on technologies with more advantages than disadvantages. Most systems will be distributed and the network architecture will be important in such systems. Price, availability and scalability will be important factors when designing the network architecture. As always in design requirements and analysis are important steps. Regarding network architecture requirements like network services, network traffic, network traffic patterns, system data, and system complexity can be important factors.

The network architecture is defined as the superior goal or plan of the network together with a set of specifications. The architecture must also define the solutions for the physical architecture (connectivity and cabling), interoperability (protocols and services), and systems/procedures for monitoring and controlling the network.

The network architecture contains no deployment information, this should be part of separate design documents, based on the architecture.

19.1 Architecture Components

The architecture can be divided into three main components. These are:

1. Basic rules or principles,
   - based on vision, behavior, strategies, and goals,
   - should be short and concise (ie. the main protocol should be TCP/IP),
   - group the rules in 1) paramount rules, 2) rules for selection of suppliers, and 3) rules for usage.
2. Technical instructions,
   - a framework for technical solutions,
   - transport protocols, integration with mainframe computers, routing protocols, quality of service (QoS), addressing, domains, and cabling,
   - type devices to use to get a robust network,
   - combination of local network (LAN) and wide area network (WAN),
   - wired or wireless?
   - security.
3. Templates and sketches
   - cabling, central, buildings, servers, WAN, internet.
   - control room,
   - firewall.

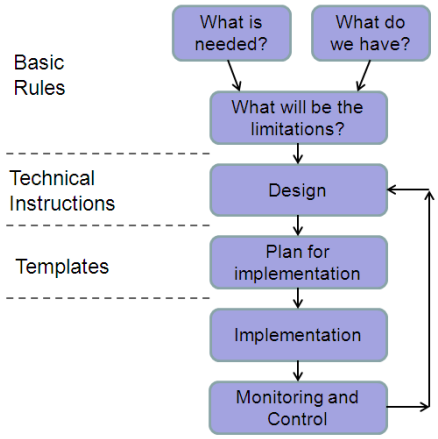


Figure 19.1: The process of the network architecture (Skrivervik 2000).

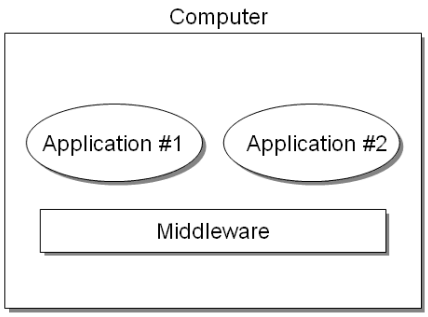


Figure 19.2: The middleware software in a computer used for communication between the applications.

The process of using these main components are shown in Figure 19.1 where the first four steps are part of the architecture components.

An important goal of the architecture is that the network system should be able to be monitored, to be controlled, automatic and centralization control and management. Different types of tools and protocols must be used for management.

19.2 Middleware

Middleware is computer software that connects software components or applications. It is used most often to support complex, distributed applications. It includes web servers, application servers, content management systems, and similar tools that support application development and delivery. Middleware is especially integral to modern information technology based on XML, SOAP, Web services, and service-oriented architecture (www.wikipedia.org 2006). Figure 19.2 shows a computer with two applications using a middleware software for communication.

Other words of middleware are Inter-Process Communication (IPC) which is a set of techniques for the exchange of data between two or more threads in one or more processes. Processes may be running on one or more computers connected by a network. IPC techniques are divided into methods for message passing, synchronization, shared memory, and remote procedure calls (RPC). The method of IPC used may vary based on the bandwidth and latency of communication between the threads, and the type of data being communicated.

In modern computers a MMU (Memory Management Unit) is used to let every process or application on a computer to execute in a separate address space giving that one application can not interfere with any of the other applications on the computer. Very good when one of the applications are trying to access part of the address space of another application by accident, but a problem when trying to make two applications to communicate on the same computer.

Some of the middleware services:

1. File: Write and read information to file (in memory or on disk).

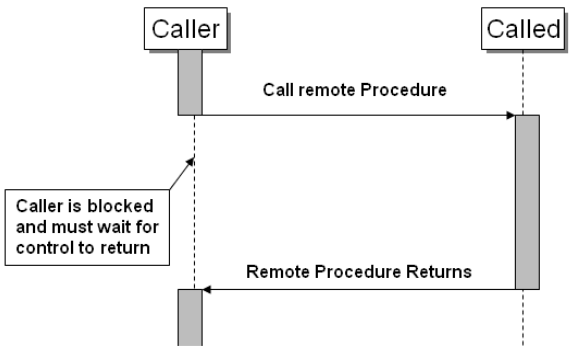


Figure 19.3: Syncronous RPC call (Mahmoud 2004).

2. Remote Procedure Call (RPC): a technology that allows a computer program to cause a subroutine or procedure to execute in another address space (commonly on another computer) without the programmer explicitly coding the details for this remote interaction. That is, the programmer would write essentially the same code whether the subroutine is local to the executing program, or remote. See Figure 19.3.

3. Shared memory: A common memory area maintained by the OS as the application can access using a set of system functions.

4. Message queue: An advanced shared memory where messages can written and read asynchronous from the applications. A message must first be written by an application before another application can read the message.

5. Named pipes: A byte stream connection (FIFO) that can be used as communication between two applications on one computer. On some system this is file based, meaning that the applications don't have to be running simultaneous.

6. Sockets: A byte stream connection (FIFO) that can be used as communication between two applications on one computer or on different computers.

7. Distributed Component Object Model (DCOM) is a Microsoft proprietary technology for software components distributed across several networked computers to communicate with each other. DCOM, which originally was called "Network OLE", extends Microsoft's COM, and provides the communication substrate under Microsoft's COM+ application server infrastructure.

8. The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on the same or multiple computers to interoperate.

9. The Extensible Markup Language (XML) is a W3C-recommended general-purpose markup language that supports a wide variety of applications. XML provides a text-based means to describe and apply a tree-based structure to information. At its base level, all information manifests as text, interspersed with markup that indicates the information's separation into a hierarchy of character data, container-like elements, and attributes of those elements. The basic syntax for one element in XML is: <name attribute="value">content</name>

10. SOAP (Simple Object Access Protocol) is a protocol for exchanging XML-based messages over computer network, normally using HTTP. Because of the lengthy XML format, SOAP can be considerably slower than other middleware technologies such as CORBA. This may not be an issue when only small messages are sent.

11. The Microsoft .NET Framework is a software component that can be added to the Microsoft Windows operating system. Programs written for the .NET Framework execute in a software environment that manages the program's runtime requirements. This runtime environment, which is also a part of the .NET Framework, is known as the Common Language Runtime (CLR). The CLR provides the appearance of an application virtual machine, so that programmers need not consider the capabilities of the specific CPU that will execute the program. The CLR also provides other

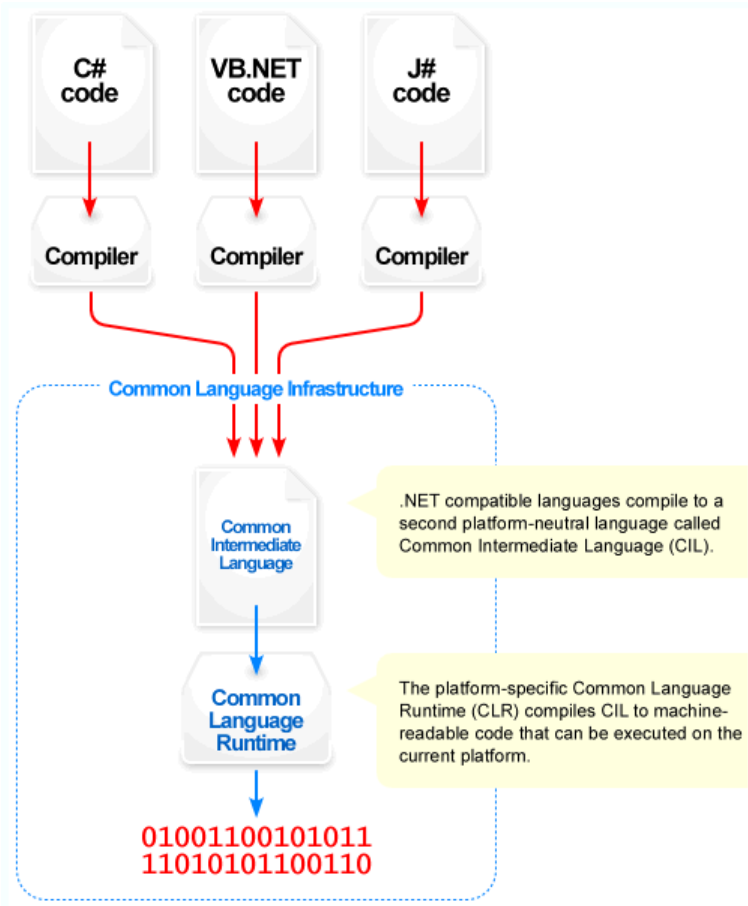


Figure 19.4: The .NET framework with CIL and CLR.

important services such as security mechanisms, memory management, and exception handling. The class library and the CLR together compose the .NET Framework. The framework is intended to make it easier to develop computer applications and to reduce the vulnerability of applications and computers to security threats. See Figure 19.4.

19.3 Firewall

A firewall is an important component in a network architecture for an industrial process control system. The firewall is an important building block in designing and maintaining a stabile and secure production network, including separating the IT and OT networks. A firewall is used to make segments for different areas or zones, and controlling the communication traffic between these areas or zones. The firewall will be a sort of a filter between two network segments with rules for the type of communication messages between these segments.

A firewall is normally used to separate the IT and OT networks in a company, but can also be used for different areas or zones within both the IT and OT network segments. The rules is about to control the access to the different areas or zones, a type of an access control between these areas or zones. This way will any problems in one area or zone not be able to propagate to other areas or zones. One way of segmentation is the usage of virtual networks (VLAN) where it is possible to group production assets based on functions and types, like at set of PLCs in one segment and SCADA software in another segment. The messages between the PLCs and the SCADA system can then be controlled by the firewall.

Industrial firewalls is often used in OT network while "standard" firewalls are used in IT network. Industrial firewalls are based on some extensions of "standard" firewalls 1) an industrial design to put up with dust and vibration, work without a fan and mounting in industrial racks, 2) optimized for industrial communication (protocols) like Modbus TCP, Profinet and Ethernet/IP to mention some. These devices will also have a "Deep Packet Inspection" (DPI) to be able to analyze the communication messages

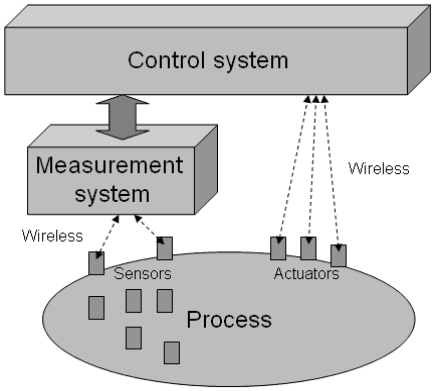


Figure 19.5: A process control system with wireless sensor network and wireless communication to actuators.

(packets) in more detail to allow a message to pass between the areas or zones. Normally a firewall will only check the addresses (to and from labels in the message header), while the DPI will analyze the contents of the message as well, and 3) minimizing the communication delay or latency. Since industrial firewalls are using DPI, the analysis of every packet will take more time and the delay or latency will be more important as packets in an OT network may be time critical for the production systems.

The listing of rules for firewalls are often divided into white listing and black listing. The white listing focus the packets that are allowed to pass through the firewall, while the black listing focus on the packets that are not allowed to pass through the firewall. The black listing will be based on definitions of virus, worms and other suspicious messages that are not allowed to enter the network. A good knowledge to the industrial processes and the OT systems are important when configuring these rules and an advice is to combine personnel from both IT and OT systems for configuring these systems.

19.4 Sensor and actuator network

Sensor network types:

1. Process sensor network,
2. Wireless sensor network.

This section will focus on wireless sensor network where each sensor or a small group of sensors are connected to a wireless device. See Figure 19.5 for an example of a wireless sensor network.

The type of layers in a wireless sensor network; physical layer, medium access control layer, network layer and application layer. The physical layer is based on the IEEE 802.15.4 specification. A wireless sensor network consists of a gateway as the connection point to the network, a set of fully functional devices, and end devices. The end devices is a sensor and/or an actuator, but can only communicate with one node. A fully functional device is a combination of an end device and a router meaning that it can communicate with several nodes. Figure 19.6 shows a WSAN with a set of devices and a gateway as an connection point to the rest of the process network.

A wireless sensor device contains one or more sensing devices, data acquisition system, computer, and a radio device for data communication. The reasons for using a sensor network are for:

1. minimize the usage of cable,
2. use information from different sensors of different manufactures,
3. combining information from several sensors,
4. cost.

Important matters for a wireless sensor device are 1) small physical device, 2) battery powered, 3) communication range should be at least tens of meters, and 4) able to connect to other wireless sensor devices.

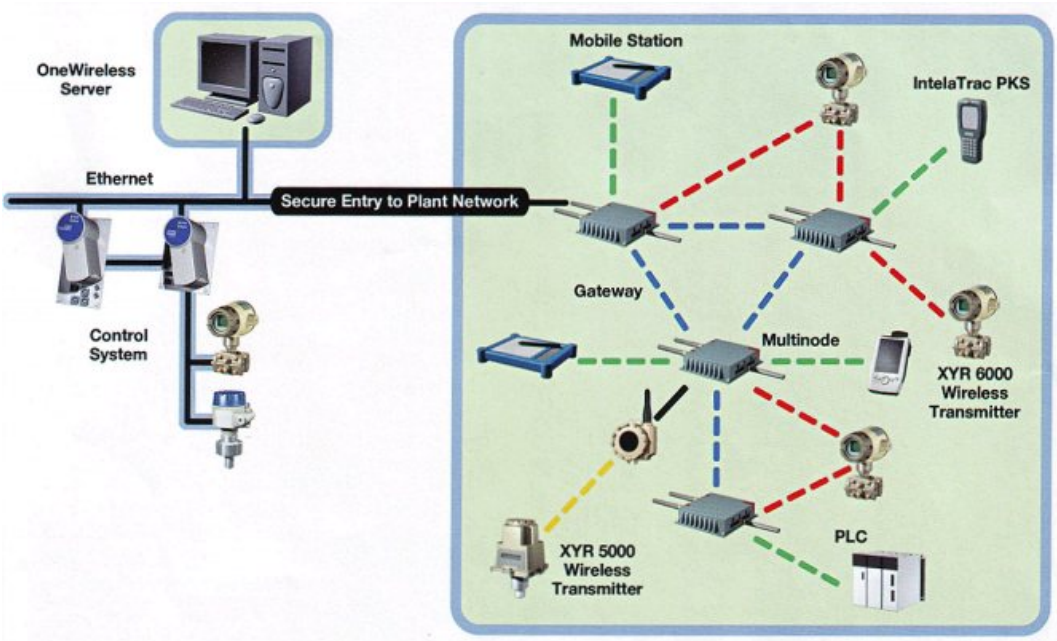


Figure 19.6: A wireless sensor and actuator network with a gateway, routers (multinodes), and end devices like sensors and actuators (AMNYTT#4 – 2009).

Protocols for wireless sensor networks:

Protocol	OSI support	Note
Bluetooth	1, 2, 3, 7	
ZigBee	1, 2, 3, 7	
WLAN	1, 2, 3 ,4	
Wireless HART	1, 2, 3, 7	
Wireless USB	1, 2, 3, 7	
6LoWPAN		Type of WLAN
Proprietary protocols	1, 2, 3	Dust Networks, Sensicast, ++

A wireless sensor network normally contains 3 or 4 layers from the OSI model:

1. Physical layer; controls the RF transceiver, performs frequency selection, and transmitting raw data bits,
2. Datalink layer; handles access to the physical radio channels, synchronization, and provides a reliable link between to MAC entities. The Datalink layer consist mainly of the Medium Access Control (MAC) layer,
3. Network layer; responsible for joining and leaving the network, and routing,
4. Application layer; provides services to user defined application processes, not to end users, handles fragmentation and reassembly of data packets, defines the role of the device within the network.

19.5 Serial port server

Today more and more industrial systems are using a local area network (LAN), the computers may miss the serial ports, and the transfer speed is better on LAN than serial lines. A serial port server, also called a serial server or port redirector, is a device that transfers data between a computer serial port (COM port) and an Ethernet local area network (LAN). This makes it possible to use Ethernet in place of serial cables, minimizing workstation clutter and also allowing serial devices to be placed far away from the computers with which they are used. Figure 19.7 shows the usage of serial port servers in a LAN to have a set of transparent serial ports. The LAN can be both wired and wireless.

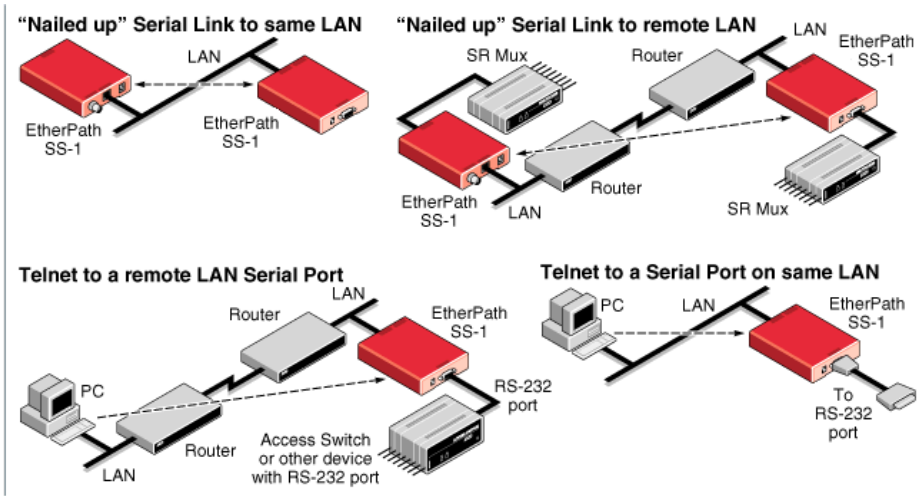


Figure 19.7: The usage of serial port servers for transparent usage of serial ports in a network.

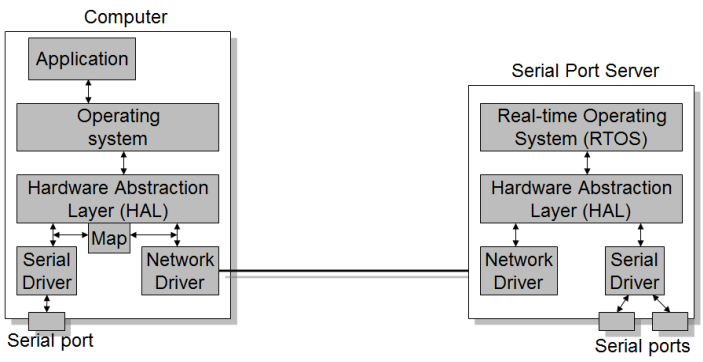


Figure 19.8: The figure shows how a software driver on the computer is mapping the serial port I/O to the network driver for I/O on the serial port server.

This solution requires extra software to be installed on the computers to map all serial port I/O to network traffic and a network device to transform the network traffic to serial port I/O. The serial port servers support from 1 and up to 128 serial ports, normally selectable between RS-232C and RS-485 interfaces. Figure 19.8 shows the mapping of the serial port I/O.

The serial port I/O is mapped from the Hardware Abstraction Layer (HAL) from the I/O to the serial port driver to the network driver. The mapping, or the redirections, must convert the I/O serial port message to a network message, before sending the message to the serial port server. This solutions gives:

1. less I/O load of the computer,
2. more software drivers (mapping) on the computer,
3. more network traffic on the computer,
4. more complex and expensive serial ports,
5. more flexible serial port I/O in the system,
6. less cabling for the I/O ports,
7. less depending on serial I/O on the computers.

The software is not a protocol according to the OSI model, but a redirector redirect the serial data between the serial protocol and the network protocol. A set of network messages added to the application layer is used for sending the data between layer one in the serial protocol and layer seven in the network protocol.

19.6 Virtual Machine

A virtual machine (VM) is a software emulation of a hardware platform and operating system that executes software applications for that platform like the physical computers. The VM can be run on any computers running operating systems Windows, Linux or Mac OS (OS X), meaning that these computers can support different operating systems simultaneously.

VM can be used for:

- testing; testing different operating systems,
- applications; using applications requiring different operating systems,

A VM will not affect the other VM on the same computer. Some VMs are:

- VMWare (www.vmware.com),
- Xen (opensource).

Chapter 20

Cloud based Systems

20.1 Introduction

Cloud based system are computer systems connected to internet, normally outside the firewall of your organization or company. The internet is the cloud but remember that internet is just a network of computers. Cloud based systems refer to computers, applications, services and/or resources that are made available to users on demand. Companies are using cloud based systems as a way to increase the capacity, enhance the functionality, and/or add additional services on demand without investing in an expensive infrastructure and/or increase / train in-house staff. Figure 20.1 shows a cloud based system where the cloud system is located at the data center, outside the firewall or the organization or company. The cloud based system can be accessed from anywhere, including in the office or at home. As seen from the Figure, security is important as all the information is available outside the firewall and available from anywhere. Security is needed to restrict the access rights from anywhere.

These cloud based systems are accessed via the web on a pay-for-use basis. Your organization or company will just pay the services that are used and “cloud” company will be responsible for having the cloud based systems available whenever you need them. Your organization or company will just pay for the usage while the “cloud” company will be responsible for having the resources available when ever you need them. This means the infrastructure, the network bandwidth, storage, security and computer processing capacity. Note that many users from many different organizations or company will share the cloud based systems with you. The “cloud” company can be anything from small companies having only an application and up to large data centers having applications, storage, and different type of services.

These cloud based systems can be servers, storage, databases, analytic services, applications or system services to mention some of these systems. The benefit of cloud based systems are 1) cost; no need to invest in expensive infrastructure, computers and IT experts, 2) speed; the systems are available online and updated hardware and software, 3) scaling; you pay for the computing power, storage and bandwidth that you want and/or need, 4) performance; the "cloud" company needs to upgrade to the

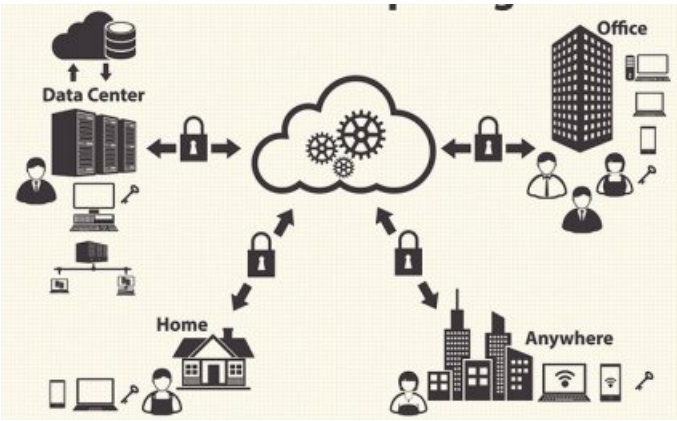


Figure 20.1: The data center is the physical location for the cloud based system, the computer system used from the office, home or anywhere. (www.fool.com: oct-18)

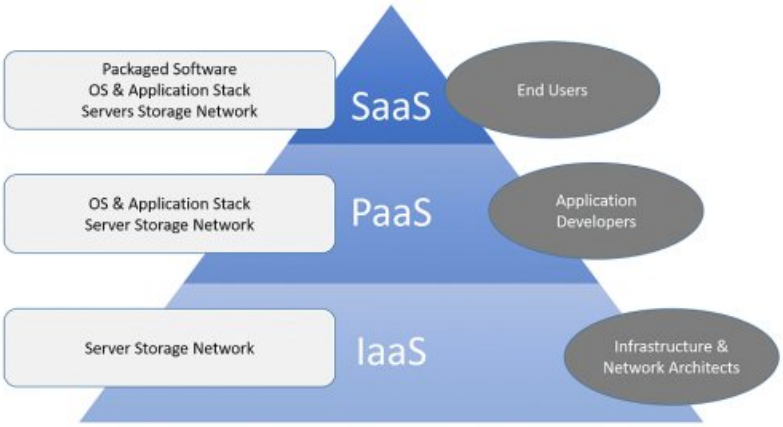


Figure 20.2: The cloud services as a stack with the IaaS, PaaS and SaaS (www.medium.com: oct-18)

latest generation of fast and efficient computing hardware and software to maintain their business, 5) security; the "cloud" company needs to offer a broad set of policies, technologies and controls that protect the applications, services, and information available on the system from potential threats.

20.2 Type of cloud based systems

There exists several types of cloud based systems. These types are private cloud, public cloud, and hybrid cloud based systems.

Private cloud based systems refers to systems that are used exclusively by a single company or organization, often located on the on-site data centre for the company. Some companies also pay "cloud" company to host their private cloud based systems, but often the services and infrastructure are maintained on a private network.

Public cloud based systems are owned and operated by the "cloud" company. These companies must deliver the computing resources, like servers, applications, and storage over internet, with focus on computer performance, network performance, storage and security. The "cloud" company is the owner of all the hardware, software, application and supporting infrastructure and must be kept in updated condition to be able to deliver to the customers.

Hybrid cloud refers to a combination of private and public cloud based systems. Technology is used that allow applications and information to be shared between them giving a greater flexibility and deployment options for many businesses giving more options for optimize the existing infrastructure, security and compliance.

20.3 Type of cloud based computing services

There exists several types of cloud based computing services like Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) to mention some of the available service types. These services are at different levels, or stack, for computer systems where IaaS is the lowest level giving an infrastructure like cpu capacity, network connections and storage as a service and SaaS is the highest level giving a running application as a service. These levels are also called the cloud computing stack as these levels will be on top of each other. Figure 20.2 shows the most used categories for cloud computing services, the infrastructure as a service at the lowest level, the platform as a service in the middle and the software as a service on the top. Often the serverless computing is also included as a category overlapping the PaaS.

20.3.1 Infrastructure as a Service (IaaS)

The basic category of cloud based computing services defined as IaaS are servers and virtual machines, storage, networks and operating systems. A virtual machine is an application environment (operating system) that is installed and running on software, which imitates dedicated hardware. IaaS means that you can rent your IT infrastructure without the need of the IT hardware.

20.3.2 Platform as a Service (PaaS)

Platform as a Service (PaaS) refers to cloud based computing services where the environment is defined for developing, testing, deploying and managing software applications. This level assumes that the underlaying IaaS is up and running, making it easier for developers to creating and managing applications.

20.3.3 Serverless Computing

Serverless Computing is overlapping with PaaS, focuses on developing and managing applications without spending time on managing servers and the infrastructure.

20.3.4 Software as a Service

Software as a service (SaaS) refers to a method for delivering software applications over the Internet, using a IaaS. The cloud provider will also need the PaaS on top of the IaaS to maintain the application, while the user only will use the application on demand and typically on a subscription basis. The cloud providers host and manage the software application and underlying infrastructure and handle any maintenance, like software upgrades and security patching.

20.4 IoT and IIOT

Internet of Things (IoT) and Industrial Internet of Things (IIoT) are devices connected to Internet, or an Intranet, using the TCP/IP communication protocol. The challenge using the TCP/IP protocol is the missing application layer with reference to the OSI model. This means that many IoT and IIoT devices will use a proprietary protocol and a SaaS meaning that you will need to run a cloud application to access the information from your IoT device. Many manufacturers will offer an API, Application Programming Interface, to access the information from the IoT device. The API is often based on a more specific protocol like Message Queue Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), Extensible Messaging and Presence Protocol (XMPP), Modbus TCP or OPC UA to mention a few, all based on the TCP/IP, but via the SaaS application. Access directly, using the TCP/IP protocol, is impossible since the data structure is not defined on the TCP layer, with reference to the OSI model. In this case must the cloud based computing service be used as the application layer, with reference to the OSI model.

Chapter 21

Data communication in C#

21.1 Introduction

The most used method for external communication with a computer is by serial communication. Serial communication means that there is a communication media where the data is transformed in serial form, each bit in every byte is transformed one by one on a single communication medium. The medium can be an electrical wire, in the air (wireless) or an optical cable (using light), and a protocol is needed to describe how the information is transformed and coded on the medium. A protocol contains a set of rules describing how to convert the information in the data between any applications and the communication media. USB, TCP/IP, MQTT, RS-485, CAN and RS-232C are a few of the communication protocols used where a protocol description is documenting the structure of the data and the usage of the communication media. This section is an introduction for using C# for accessing serial communication functions on a computer, for external communication with other communication devices, based on some sort of protocols.

21.2 Serial port data communication

21.2.1 Introduction

Data communication is important for communication between computers, for stand-alone devices like a GPS device, on local area network (LAN) and on internet. Every time you want to send or receive an online message, send or receive a SMS (short message service) send or receive an email, access a web page on internet or download any information from internet, data communication is needed. If you connect a device to your computer, like a GPS device, an USB memory stick, a mobile phone or a camera, data communication is also needed. A computer with external communication capabilities is often referred to as a node and a set of nodes need to use the same structure of the data and the communication media to be able to communicate. However, this structure can be very complex, as shown in Figure 21.1.

This structure is defined by a protocol document describing the details for how to use the communication media and how to interpret the data information. The protocol is a set of rules trying to structure sending and receiving of information. This can be compared with two humans that are trying

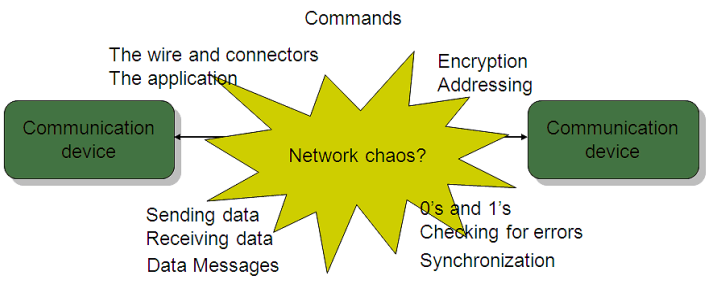


Figure 21.1: An overview of the challenges when two or more nodes (computers) need to communicate using a communication media.

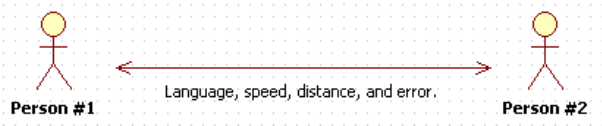


Figure 21.2: Compare a protocol with the situation where two people should be able to talk with and understand each other.

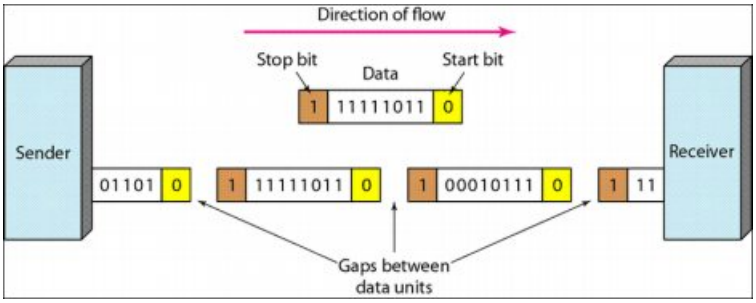


Figure 21.3: Asyncronous transmission using a start and stop bit for syncronization (copy from NetworkEncyclopedia.com / Aug-20).

to communicate, they also need to have a set of rules to understand each other like using a language that both understand, not talk at the same time, and not being too far away from each other, as indicated in Figure 21.2.

Since data communication contains many different states and rules, the Open System Interconnection reference model (OSI model) (Houldsworth 1990), (Levermore 2000), (Stevens 1990), (Stallings 2001) is often used to better structure the situation. The OSI model is a seven layer model defining different types of functionality that must be fulfilled by the protocol. The lowest layer in the OSI model is the physical layer describing the functionality for a physical connection between two or more nodes. This layer is describing the main functionality of the serial port communication. This connection may also be named M2M communication in some literature, short for machine to machine communication.

21.2.2 Physical layer

The physical layer may be wired, optical or wireless (Stallings 2001), and describes how to connect to the physical communication media, how to transfer the data on the communication media and how to handle the data flow on the communication media. The wired connection will be either parallel or serial communication. Parallel is using one wire for each bit, while serial is handling each bit in time slots on the same wire. Parallel is faster but serial is cheaper and can be used over longer distances. So serial communication is used most. Optical is using light (laser) as the communication carrier instead of electrical current as for a wired connection. Wireless can be protocols like Bluetooth and WiFi (Stallings 2001), using radio frequencies as the communication carrier. This chapter will focus further on wired connections only, for serial data communication.

21.2.3 Asynchronous serial communication

Serial data communication will be either synchronous or asynchronous communication where asynchronous is most used. In synchronous communication there will be a continuos stream of data bits and an external synchronization signal is needed to synchronize the sender and the receiver. This synchronization signal must use an extra wire. The asynchronous communication will divide the data stream into characters and add synchronization bits for each of the characters in the data stream. No external synchronization signal is needed however the transmission will take a little more time as the synchronization bits must be transmitted along with the data bits. Figure 21.3 shows asynchronous communication where each byte has a start bit and a stop bit for synchronization between the transmitter (sender) and the receiver. Each byte in this figure consists of eight bits.

The RS-232C, RS-422 and RS-485 protocols are some examples of asynchronous protocols used a lot on computer systems, and these protocols need a set of communication parameters to be configured

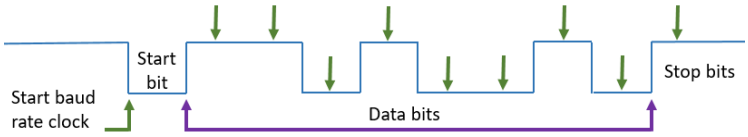


Figure 21.4: The transmiision of the bits in asyncronous serial communication. It will start with a start bit, then the data bits, and a stop bit at the end.

before starting transferring the data stream. RS-232C has normally only one transmitter node and one receiver node, RS-422 has normally one transmitter node and several receiver node, while RS-485 has normally several both transmitter and receiving nodes.

The number of data bits and stop bits can be configured while one start bit is fixed. The communication parameters that can be configured are:

- Baud rate; defining the transmission speed, the number of bits per. second. The baud rate is normally either $300, 600, 1200, 2400, 4800, 9600, 19200, 38400$ or 115200.
- Data bits; defining the number of bits in each character to transmit. The number of data bits are either 5, 6, 7 or 8.
- Parity check; single bit summing up as either None, Even or Odd parity. Even or odd parity will add an extra bit to the data stream, before the stop bit.
- Stop bits; number of bits after the data bits and any parity bit. The number of stop bits are either 1, 1.5 or 2.

The start bit is used for synchronization as the receiver(s) do not know when any node is transmitting. The start bit is normally changing the electrical carrier voltage level from high to low indicating that new data will be transmitted. The start bit will also synchronize the baud rate system for the receiver as the baud rate system is used to define when a new bit is to be transmitted and when the receiver can read this next bit. Figure 21.4 shows the message package, the transmission of one byte, starting with a start bit and ending with the stop bit. The figure also shows when the receiver should check for a new bit, indicated by the vertical green arrows.

The parity bit will be added after the data bits, and before the stop bit(s) if even or odd parity is configured. The extra bit will be 0 or 1 to fulfill the parity configuration. If the data bits are 1101 1111, as shown as one byte in Figure 21.4 will the number of "1" be 7, an odd number of bits. If odd parity is selected will the extra parity bit be "0", or if even parity is selected will the extra parity bit be "1". Parity bit checking can be useful if communication in a noise environment, or over a long distance.

The conversion between the data byte and the data bits, and including communication parameters, is handled by a hardware chip named universal asynchronous receiver and transmitter (UART). The UART will control the baud rate, the number of data bits, any optional parity check, and include the start and stop bits. The frame is the data bits and is part of the total package as shown in figure 21.4. The UART will handle any framing errors meaning any errors of the number of data bits. The reasons can be configuration of wrong baudrate or wrong number of data bits. The UART also contains a first in first out buffer (FIFO) for the receiving part, for example 16 bytes, in addition to the parity and framing error checking. The FIFO buffer allows the serial port to receive more characters before any software driver needs to read these characters. The UART is shown in figure 21.5 where UART1 is part of the transmitter node transmitting a character from the TX (transmitter) part to the RX (receiver) part of the receiver node. So UART 2 is part of the receiver node.

The control of the UART is handled by the drivers of the operating system, and most programming languages have ready made libraries for communication with these drivers. This document will explore the serial port library in C#.

21.2.4 Communication protocols

Before looking into the serial port programming, two asynchronous serial protocols will be exploited, that will be used as examples in the programming section. These protocols are the National Marine Electronics Association (NMEA) 183 protocol that is often used for GPS (Global Position System), and the on-board diagnostics (OBD) used as a network inside most cars produced after 2000. The NMEA

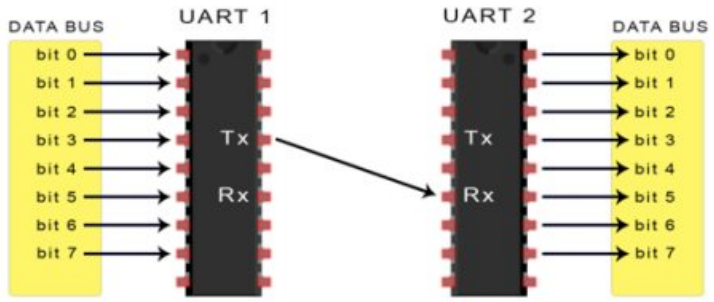


Figure 21.5: The UART hardware chips converting between data bytes and data bits for asyncronous communication between two nodes.

Figure 21.6: One version of an USB to serial port adapter, including the pins for a 9 pins serial port connector. The connector and pinning is defined in the RS-232C protocol.

183 is a one way protocol only so the transmitter is cyclic transmitting all necessary information and any receiver needs to listen to the information and store only the information needed. So only one node can be the transmitter, the rest of the nodes will be receivers. The OBD protocol is a two-way protocol where all the nodes can be either a transmitter or a receiver. Any node can send a request on the network and the node with the information will answer within a specific time frame. However, before digging into these two different protocols, a short introduction on connecting to the serial port is needed.

Serial port

Older computers had one or several RS−232C or RS−485 ports while modern computers only have USB ports. However, many versions of USB to serial port adapters exist, one of these is shown in Figure 21.6.

The three most important pins of the serial port connector is the receive data (RD) pin (rx: pin 2), transmit data (TD) pin (tx: pin 3) and ground (GND) pin (pin 5). Pin 2 is the Rx (receive) connection and pin 3 is the Tx (Transmit) connection as shown in Figure 21.5. This is shown for 9 pins connectors, but there exists 25 pins connectors as well. One way of testing such an adapter is by interconnecting pin 2 and pin 3 and make a test option in the software just to transmit a dummy text and check receiving the same text. This test option is often named loopback testing, and will test both the hardware and the software for that node.

NMEA 183 protocol

The NMEA 183 protocol is a cyclic protocol where the transmitting node is sending a set of messages in a cyclic manner, and is the protocol used for many GPS devices. One is these devices is the Pharos iGPS−180, shown in Figure 21.7. This is a standalone device, the power requirement is 5 V (normally from the USB port) and the protocol is based on RS-232C, the baud rate is 4800, 8 data bits, no parity and one stop bit.

The GPS device has an integrated antenna and is using the NMEA protocol for transmitting the GPS information. The NMEA protocol messages have the following format:

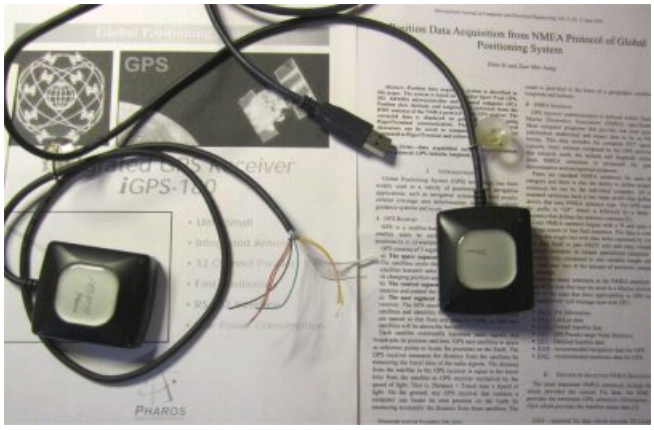


Figure 21.7: GPS devices are hardware devices available in many sizes and forms. The Pharos iGPS-180 is the device used for collecting the data in this document. The figure shows two such devices, one with an USB connection and one with just power and communication connection.

```
Rx[91]=<$GPGGA,000028.997,0000.0000,N,00000.0000,E,0,00,0.0,0.0,M,,,,0000*03
$GPGSA,A,1,,,,,,,,,,,,,0.0,0.0,0.0*30
$GPGSV,3,1,12,06,90>
Rx[92]=<,000,,22,45,000,,24,43,000,,17,42,000,*77
$GPGSV,3,2,12,10,31,000,,26,22,000,,23,20,000,,05,13,000,*78
$GPGSV,3,3,12,25,12,000>
Rx[93]=<,,29,07,000,,04,04,000,,09,02,000,*79
$GPRMC,000028.997,V,0000.0000,N,00000.0000,E,,,051299,,*1F
$GPGGA,000029.996,0000.0000,N>
Rx[94]=<,00000.0000,E,0,00,0.0,0.0,M,,,,0000*03
$GPGSA,A,1,,,,,,,,,,,,,0.0,0.0,0.0*30
```

Figure 21.8: An example of some of the NMEA commands like GPGGA, GPGSV, GPRMC and GPGSA.

$ Command , Param#1 , Param#2 , .., Param#N * ChkSum

where all the fields are based on printable ASCII characters. The $ indicates the start of a new message, the Command indicates the type and number of parameters (1 to N), comma is the standard field separator, and * (asterisk) indicates the checksum of the message. The asterisk is immediately followed by a checksum represented as a two-digit hexadecimal number, indicating the end of one NMEA message. The checksum is the bitwise exclusive OR of the ASCII codes of all characters between the $ and *. The message ends with the <CR><LF> codes. An example of some of the NMEA messages is shown in Figure 21.8.

Figure 21.8 shows one of the challenges with asynchronous serial communication as one and one character is sent. The figure shows that the receive function has been used four times, Rx[91] to Rx[94]. The first read function, Rx[91], contains three NMEA messages. However, only the first part of the last NMEA message is read, the $GPGSV command. This is quite common when reading asynchronous serial communication protocols and the software must include logic to merge the last part of this read function and the first part of the next read function. As shown in the figure, the NMEA messages start with the $ sign and end with checksum field starting with the * sign where each field is separated by the , sign.

The most used commands for GPS information in the NMEA protocol are, where some of them are also shown in Figure 21.8:

```
>
 Time=204005.999  Latitude=5901.3925  Longtude=942.7862 Quality=1
Rx[7762]=< Cmd[9911]:<GPGGA,204005.999,5901.3925,N,00942.7862,E,1,05,1.7,132.0,M,,,,0000*02>
```

Figure 21.9: An example of the GPGGA command with valid information, at a location at Stathelle in Norway.

Command	Description	Fields
GPGGA	Global Positioning System Fix Data	10 (12)
GPGLL	Geographic Latitude and Longitude	4
GPGSA	Satellite status	6
GPGSV	Satellites in view	7
GPRMC	Recommended Minimum sentence C	8
GPVTG	Track made good and ground speed	4

The first two characters of the command field defines the sending device and the last characters are actually the NMEA command. The sending device can be either GP, GL or GN. The NMEA GGA command (GPGGA) can be used to receive the location of a moving device like a car. Figure 21.8 shows a GGA command, but before the GPS device has received data from enough satellites, the contents is therefor marked as invalid (field #6 is 0). Also note that one of the GGA messages are broken into two parts (in Rx[93] and Rx[94]), and must be merged into one message by logic in the software. The GPS device will use some time to get information from a set of satellites before valid information can be added to these messages. Figures 21.9 shows a message (Rx[7762]) with valid information for a GGA command (field #6 is 1).

The GGA command description in the NMEA protocol documentation has the following information for the different fields (in the message):

Field	Description	Example
1	UTC time from satellite	20:40:06 UTC
2	Latitude	5901.3925
3	Latitude direction	N
4	Longitude	942.7862
5	Longitude direction	E
6	GPS quality: 0=invalid, 1=GPS fix, 2=DGPS fix (0-8)	1
7	Number of satellites	05
8	Horizontal dilution of position	1.7
9	Altitude, Meters, above mean sea level	132.0 M
10	Height of geoid (mean sea level) above WGS84 ellipsoid	
11	Empty field, time of seconds since last DGPS update	
12	Empty field, DGPS station ID number	
	checksum after the * sign	02

The example column (to the right) contains the values from Figures 21.9. The first field, after the command, is an accurate UTC time showing that the information was received at 20:40:06 UTC time, being 22:40:06 local time (Norwegian summer time). The format of latitude and longitude coordinates in the NMEA format is (d)ddmm.mmmm where d=degrees and m=minutes. The degree part can be calculated as an integer by dividing with 100, and minutes part by dividing the rest with 60. The GPS latitude position based on the values from the example column will be:

$$\begin{aligned} 5901.3925/100 &= 59 \\ 1.3925/60 &= 0.023208 \end{aligned} \tag{21.1}$$

and the GPS longitude position will be:

$$\begin{aligned} 942.7862/100 &= 9 \\ 42.7862/60 &= 0.713103 \end{aligned} \tag{21.2}$$

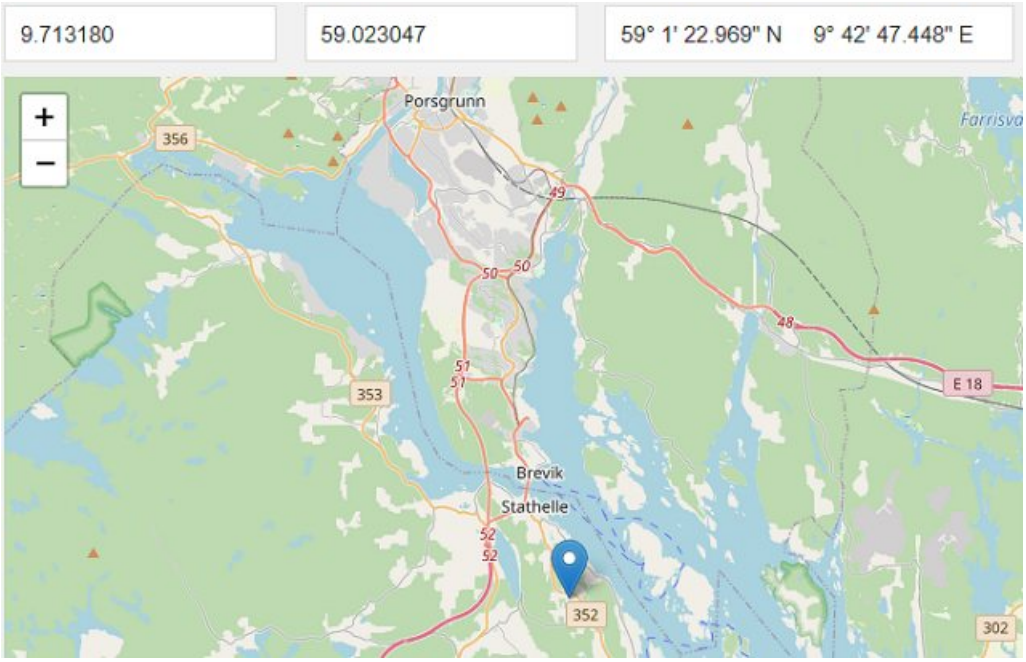


Figure 21.10: A map of the Grenland area in Norway, with the Porsgrunn, Brevik and Stathelle cities. The blue mark is the location of the GPS device at Stathelle, with the position on top of the map.

The direction factor for the latitude position is N=+1 and S=−1, and the direction factor for the longitude position E=+1 and W=−1. The address position for the GPS device is shown at the top of the map in Figure 21.10 with the longitude position = 9.713180 and the latitude position = 59.023047, very close to the position received from the GPS device. These positions are:

Type	Latitude	Longitude
Map	59.023047	9.713180
GPS	59.023208	9.713103

The deviation is due to that the position in Figure 21.10 is based on a street address while the GPS device has a specific position within that address area. Older GPS devices, like this device, have an accuracy of about 5 m while newer GPS devices (newer than 2018) may have an accuracy of about 30 cm. A GPS device can then be used to receive and store the position of a moving object, like a car.

GPS devices using the NMEA protocol has either a serial port or an USB port for connection to the computer, and the serial port (COM port) library can be used in C#. To get more information for a specific moving object, other protocols must be used. The OBD protocol is such a protocol for a car.

OBD-2 / EODB protocol

The On-Board Diagnostic (OBD-2 or OBD-II) standard is a requirement for all cars sold in the United States from 1996 and is a protocol for a network of sensors, actuators and controllers inside cars. European On-Board Diagnostics (EOBD) is the European equivalent of the US OBD II standard, which applies to petrol cars sold in Europe from 2001 (and diesel cars three years later). Cars are using the Controller Area Network (CAN bus) to let microcontrollers and devices to communicate without the need for a server, so the CAN bus is the local area network (LAN) for all the devices installed in a modern car. The OBD protocol is defining the connection port, the signals and the commands. A modern car includes an industrial control system with several distributed control systems like engine controller, braking system (ABS) controller, transmission controller, suspension controller and air conditioner controller to mention some. One possible option for an external communication with this network (LAN) is by using the OBD port.

The CAN protocol is a complex protocol that can be hard to implement in software. A better option is to use the ELM327 interface which is a protocol converter converting between the CAN commands and type of Attention (AT) commands. AT commands is normally used for communication with modems

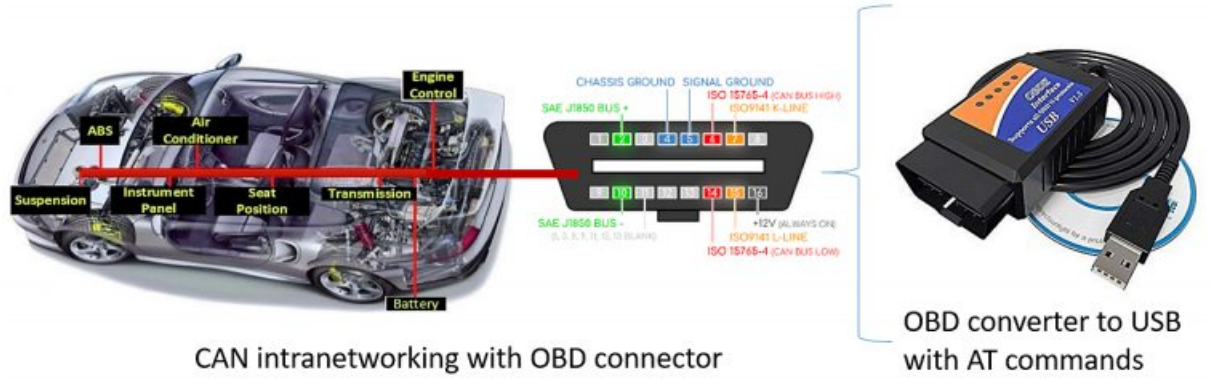


Figure 21.11: The OBD connector, and an example of the ELM327 interface.

where the commands normally start with AT. The ELM327 implementation has customized a set of these commands (Elm 2016). The OBD port and the ELM327 interface is shown in Figure 21.11.

The ELM327 interface can connect to a computer using either USB, Bluetooth or WiFi connection[1]. The easiest way is using an USB connection as this connection can be used as a serial port (COM port) in C#, just like the NMEA protocol. Unlike the NMEA protocol, the OBD protocol is a two-way protocol meaning that the software must first send a request for a specific information, wait for the system to respond, and then send a new request. In this case must the software also handles any timeout situation in case the system is not responding. The protocol baud rate is either 9600 or 38400 depending on the setting of the ELM327 device, 8 data bits, no parity and one stop bit.

One possible startup sequence of AT commands for communicating with the ELM327 interface is:

	Tx	**Rx**	**Comments**
1	ATZ	ATZ ELM327 v1.5	Device reset
2	ATE0	ATE0 OK	Message echo On=1/Off=0
3	ATH0	OK	Message header On=1/Off=0
4	ATL0	OK	Message Line feed On=1/Off=0
5	ATSP0	OK	Set protocol: 0=Auto (Different CAN protocols)
6	ATDP	AUTO	Display protocol type
7	ATI	ELM327 v1.5	Device identification
$8-N$			Looping getting the values

Note that some of these commands may take some time before an answer is received. As shown in Figure 21.11 the car has several controllers monitoring and controlling a set of devices as sensors and actuators. The OBD protocol is handling a set of parameters and these parameters are linked to values handled by these controllers. Many of the these parameters are defined as standard parameters in OBD and are assigned a parameter id number (PID) that will be used by the protocol to get the parameter value.

The OBD protocol is defining different service types or modes and the most important service types are:

Type	**Description**
01	Show current data
03	Show stored diagnostic trouble codes
04	Clear diagnostic trouble codes and store values

The most used service type is 01, to get the current value for a specific parameter. The service types 03 and 04 are used to read any error codes (often combined with a warning on the dash board), and to clear these error codes. The OBD protocol defines at least 10 different service types, check the protocol documentation for more information.

[1] Many ELM327 devices contain old hardware and will not be supported by standard Windows 10 or 11 drivers. In these cases, add an older serial driver supporting either the CH340 (Arduino) or PL2303HXA hardware depending on your ELM327 device.

Using the service type 01, a set of parameters can be read from the engine controller. The OBD protocol is defining the parameter ids to read these parameters and a few of these parameters ids are:

PID Hex	PID Dec	Description	Min Value	Max Value	Unit	Rx bytes	Equation
04	4	Calculated engine load	0	100	%	1 (A)	$\frac{A*100}{255}$
05	5	Engine coolant temperature	−40	215	°C	1 (A)	$A - 40$
$0C$	12	Engine speed	0	16383.75	rpm	2 (A+B)	$\frac{(256*A)+B}{4}$
$0D$	13	Vehicle speed	0	255	km/h	1 (A)	A
$0F$	15	Intake air temperature	−40	215	°C	1 (A)	$A - 40$
11	17	Trottle position	0	100	%	1 (A)	$\frac{A*100}{255}$

Information about more PID can be found in the OBD protocol documentation, a lot is available on internet, like the OBD-II PIDs web site[2]. Many car manufacturers may also have defined their own PIDs in addition to the standard PIDs.

The serial port driver in C# can be used together with the ELM327 device to get information from the OBD port. The commands will be to set up and initialize the ELM327 device and start requesting information. The transmit commands will be:

$$ServiceType\ PIDCode\ (Hex)\ \ < CR >< LF >$$

and the receive information will be:

$$0x40 + ServiceType\ AnswerBytes\ (Hex)\ \ < CR >< LF >$$

An example of the startup communication with the ELM327 device is shown in the following list, where Tx is the command sent from C# application and Rx is the data received by the C# application. The list of the first commands are:

```
16-Apr-2020;16:03:16;Tx;ATZ
16-Apr-2020;16:03:17;Rx;ELM327 v1.5
16-Apr-2020;16:03:17;Tx;ATE0
16-Apr-2020;16:03:18;Rx;ATE0 OK>
16-Apr-2020;16:03:18;Tx;ATH0
16-Apr-2020;16:03:18;Rx;OK>
16-Apr-2020;16:03:18;Tx;ATL0
16-Apr-2020;16:03:19;Rx;OK>
16-Apr-2020;16:03:19;Tx;ATSP0
16-Apr-2020;16:03:23;Rx;OK>
16-Apr-2020;16:03:23;Tx;ATDP
16-Apr-2020;16:03:24;Rx;AUTO>
16-Apr-2020;16:03:24;Tx;ATI
16-Apr-2020;16:03:25;Rx;ELM327 v1.5
16-Apr-2020;16:03:26;Tx;010D
16-Apr-2020;16:03:26;Rx;41 0D 08
16-Apr-2020;16:03:26;Tx;0104
16-Apr-2020;16:03:26;Rx;41 04 67
```

A timestamp is included and note that the respons of the ATSP0 command, set the specific protocol between the ELM327 device and the CAN bus, takes almost four seconds. However, the normal timing is faster, about 200 to 300 ms for each message sequence. The last four lines are OBD information from the vehicle where (notice that all values are hexadecimal values):

[2] OBD-II PIDs web site: https://en.wikipedia.org/wiki/OBD-II_PIDs (Oct-19).

```
//////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;
using System.IO;
using System.IO.Ports;
//
namespace NosComTerm
```

Figure 21.12: Include the serial port library with the "using System.IO.Ports;" statement.

Id		Service	Command	Value	Description
1	Tx	01	$0D$		Request speed
2	Rx	41	$0D$	08	Speed response
3	Tx	01	04		Request engine load
4	Rx	41	04	67	Engine load response

The first request is for the vehicle speed, the response starts with $0x41$ which is $0x40 + 01$ which is a valid service type 01 response. The command is $0x0D$ so the current speed is 8 km/ h. The second request is the engine load, and the engine load will be (see the PID list):

$$\frac{0x67 * 100}{255}\% = \frac{103 * 100}{255}\% = 40,4\%$$

The application must, in a cyclic manner, request the wanted parameters from the car. Parameters that are changing often should be requested more frequently than parameters not changing that much. Parameters changing often can be the vehicle and engine speed while parameters not changing that often can be the engine coolant temperature and the intake air temperature. An evaluation of which parameters that are important for your application and how often each parameter should be recorded should be decided in the analysis section of the software development.

21.3 Serial port programming in C#

21.3.1 Introduction

Serial port programming in C# is quite straightforward by using the standard serial library. Include the serial port library in your code, as shown in Figure 21.12, and the serial port can be used like a text file meaning that the port can be open, read from, written to and be closed.

Note that open is exclusive so only one application can open a specific serial port at a given time and therefor it is important to check for a valid open response. For the same reason remember to close any serial port no longer in use in your application, including when changing from one serial port to another serial port.

In this chapter a console application will be developed to test the basic serial port functions with some snippets of source code of using the serial ports and decoding the protocols. As always when developing a software application, a use case diagram should be made and then choose the type of tier architecture. An use case diagram for the application is shown in Figure 21.13. The use case shows the user functions as 1) select a serial (COM) port, 2) set the parameters for this COM port, 3) select the protocol for decoding the information on this COM port and 4) handling the data communication on the COM port.

A three tier architecture is choosen, one tier for the user interface (user interaction), one tier for the business logic of the application meaning handling the logic of the serial protocols and the information in the application, and one tier for the data layer to handle the serial port. This tier should also handle any logging of data to a file, if needed. A tier can also be named a layer in the literature. The reason for a tier architecture is to get a better structure of the application, and to better separate the code for user interface, the business logic and the data handling.

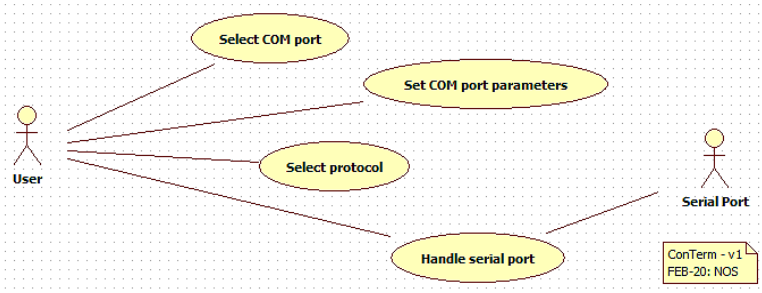


Figure 21.13: A use case diagram for the terminal application, with the user functions of selecting a COM port, set the COM port parameters, select the protocol for that COM port, and handling the data communication on that specific COM port.

21.3.2 Three tier architecture

The application was developed as a C# console application using the Visual Studio IDE. The IDE creates a console template with the Program class and the static Main() method. The class ProgramSetupConsoleTerm was created to make the three tier architecture as shown in the C# code snippet:

```
//////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace NosComTerm
{
    /// <summary>
    class Program
    ///
    /// </summary>
    {
        static void Main(string[] args)
        {
            ProgramSetupConsoleTerm rProgramSetupConsoleTerm;

            rProgramSetupConsoleTerm = new ProgramSetupConsoleTerm();
        }
    }
    /// <summary>
    class ProgramSetupConsoleTerm
    ///
    /// </summary>
    {
        LayerData rLayerData;
        LayerBusiness rLayerBusiness;
        LayerUser rLayerUser;

        public ProgramSetupConsoleTerm()
        {
            /// include a try/catch block?
            rLayerData = new LayerData();
            rLayerBusiness = new LayerBusiness(rLayerData);
            rLayerUser = new LayerUser(rLayerBusiness);
            rLayerUser.MainLoop();
        }
```

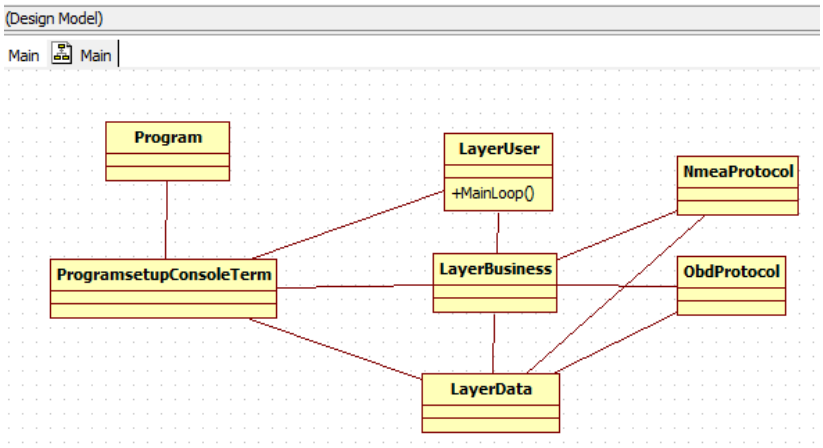


Figure 21.14: The main classes of the terminal application with a three tier architecture.

```
    }
}
////////////////////////////////////////// EOC ////////////////////////////////
```

The ProgramSetupConsoleTerm class contains only the constructor running when the object was created. The object creates the tier classes, starts with the LayerData tier, then the LayerBusiness tier including a reference to the LayerData object, then the LayerUser tier object including the reference to the LayerBusiness object, and finally starts the MainLoop() method in the LayerUser object. The programming code standard used for development defines that any objects should start with the 'r' character to indicate a reference. The MainLoop() method will display the menu and wait for any input selection by the user.

A try/catch block could have been included as the new statements are system operation and it is a good advice to have a general try/catch block on a "high level" to catch any errors not catched by other try/catch blocks.

The class diagram for the application is shown in Figure 21.14 where the main classes are included. Two classes to handle the NMEA and OBD protocol are included by the LayerBusiness class. Normally a separate class for handling the SerialPort should have been included by the LayerData class, but the SerialPort methods are included in the LayerData to simplifying the application.

The main menu has selections for handling either the NMEA or the OBD protocol that will call an appropriate method in the LayerBusiness class. The method in the LayerBusiness class will call either the method in the NmeaProtocol or in the ObdProtocol class depending on the selected protocol, which will use the serial port methods in the LayerData class.

21.3.3 Serial port library

The serial port library has a set of predefined methods that can be used for handling any serial port on the computer. According to Figure 21.13, the use case diagram, the first option should be to get a list of available serial port, select one of the available serial port(s), and the second option should be to open the serial port with the right communication parameters as baudrate, data bits, parity and stop bits before handling any protocol information.

Available serial ports

The serial port library has a method that will return a string array with the available serial ports in the system. The source code snippet for a method getting the available serial ports in your system:

```
/// <summary>
private string[] GetListOfAvailableComPorts()
/// Purpose: fill a string array with available COM ports, null if no com ports.
/// Version: 1.0: 26-JAN-20: NOS
```

```
/// </summary>
{
    string[] saComPortNames;

    try
    {
        saComPortNames = SerialPort.GetPortNames();
    }
    catch
    {
        saComPortNames = null;
    }
    return saComPortNames;
}
```

The method will return the string array filled with the available serial ports, and saComPortNames.Length will define the number of serial ports in the system. Try/catch is used since this is a system command, and the string array will be null if an error.

Open and Close

Select an available serial port from the list, and open the serial port using the serial port name from the list, the baudrate and with 8 data bits, no parity and one stop bit as default. The serial port names in Windows are on the form *COMx* and in Linux on the form /dev/ttyx, where x is a number from 1 to N (Linux from 00 to N-1). The source code snippet for opening the serial port is:

```
/// <summary>
public bool Open(string sPortName, int iBaudrate, out string sOpenMsg)
/// Purpose: open a COM port with selectable baudrate, fixed data bits, parity and stop bits.
/// Version: 1.0: 26-JAN-20: NOS
/// </summary>
{
    if (bOpenPort == true)
    {
        rSerialPort.Close();
    }
    try
    {
        rSerialPort = new SerialPort(sPortName, iBaudrate, Parity.None, 8, StopBits.One);
        rSerialPort.ReadTimeout = 500;
        rSerialPort.WriteTimeout = 500;
        rSerialPort.Open();
        sOpenMsg = "Open <" + sPortName + "> serial port OK!";
        bOpenPort = true;
    }
    catch (Exception e)
    {
        sOpenMsg = "Error open <" + sPortName + "> serial port: " + e.Message;
        bOpenPort = false;
    }
    return bOpenPort;
}
```

The rSerialPort object reference is a global variable for the class as the object reference must be used for any read() and write() methods for serial port. The binary flag bOpenPort is used for defining if a serial port is open or not. This flag can also be used in the Read(), Write() and Close() section to verify that the serial port is already open. So the first operation when opening a serial port is to close any

already opened serial port. Then make a new serialPort object with the correct settings of the baudrate, data bits, parity and stop bits. These parameter can also be changed after the serial port is opened. The parameters ReadTimout and WriteTimeout are timeout parameters for letting the serial port driver of the operating system to timeout if no characters are read or written within the timeout period. These timeout parameters are in ms, in this case the timeout parameters are 500 ms. The Read() operation will try to read a specific number of bytes and will not finish unless the number of characters are read or if the timeout parameter is set. The application will wait for ever ("hang") if no valid if the ReadTimout parameter value is not set (used). An ok open message will be returned to the calling method with the binary open flag set to true, if the open() method was ok. An error message with information will be returned to the calling method with the binary open flag set to false if the open() method fails.

The source code snippet for closing the serial port:

```
/// <summary>
public void Close()
/// Purpose: close an open serial port
/// Version: 1.0: 26-JAN-20: NOS
/// </summary>
{
    if (bOpenPort == true)
    {
        rSerialPort.Close();
        bOpenPort = false;
    }
}
```

The binary flag bOpenPort is used to indicate an already open serial port, see the code snippet for open the serial port.

Read and Write

The Read() method will read a number of bytes from the serial port, the parameters of the Read() method are:

public int Read(byte[] buffer, int offset, int count);

where buffer is a byte array, offset is starting point in the buffer array (normally 0), and count is the number of bytes that you want to read from the serial port. Offset is used if you have read only part of a message and want to read the rest of the message. The Read() method supports several type of parameters, check the C# serial port documentation for more information. The source code snippet for an example of reading from the serial port:

```
/// <summary>
public string Read(int iCntMax, bool bTimeoutMsg)
/// Purpose: read a serial port with a maximum number of bytes, with timeout support.
/// Version: 1.0: 26-JAN-20: NOS
/// </summary>
{
    int iLen, iMsgCnt, iOffset, iMaxLoop;
    string sMsgBuf;
    byte[] bMsgBuf;

    iLen = 0;
    sMsgBuf = "";
    try
    {
        bMsgBuf = new byte[iCntMax + 16];
        iOffset = 0;
```

```
        try
        {
            iMaxLoop = 0;
            while (iOffset < iCntMax && iMaxLoop < 64)
            {
                iLen = rSerialPort.Read(bMsgBuf, iOffset, (bMsgBuf.Length - iOffset));
                iOffset += iLen;
                iMaxLoop++;
            }
            for (iMsgCnt = 0; iMsgCnt < iOffset; iMsgCnt++)
            {
                sMsgBuf = sMsgBuf + Convert.ToChar(bMsgBuf[iMsgCnt]);
            }
        }
        catch (TimeoutException)
        {
            // Do nothing
            if (iOffset > 0)
            {
                for (iMsgCnt = 0; iMsgCnt < iOffset; iMsgCnt++)
                {
                    sMsgBuf = sMsgBuf + Convert.ToChar(bMsgBuf[iMsgCnt]);
                }
                if (bTimeoutMsg == true)
                {
                    sMsgBuf = sMsgBuf + "<Timeout>";
                }
            }
        }
        catch (Exception e)
        {
            // Any other exceptions
            sMsgBuf = sMsgBuf + "<Exception=" + e.Message + ">";
        }
    }
    catch (Exception e)
    {
        sMsgBuf = sMsgBuf + "<Serial port error=" + e.Message + ">";
    }
    return sMsgBuf;
}
```

The source code snippet for an example of writing to the serial port:

```
/// <summary>
public bool Write(string sBuf)
/// Purpose: write a string buffer to the serial port
/// Version: 1.0: 26-JAN-20: NOS
/// </summary>
{
    bool bWriteOk;

    try
    {
        rSerialPort.Write(sBuf);
        bWriteOk = true;
    }
```

```
---- Console Terminal : Main Menu ----
Current COM port: COM7
--------------------------------------
N: Select a COM port
O: Open a selected COM port
T: Input the loopback test string
R: Raw dumping of COM port data
L: Test loopback of COM port data
G: GPS data from NMEA protocol (S:Start/Stop)
B: OBD protocol (S:Start/Stop)
M: Modbus (dummy) protocol (S:Start/Stop)
S: Serial Terminal
Q: Quit console terminal

Select a function:
```

Figure 21.15: The output from the MainMenu() method in the application, showing the options that can be selected.

```
        catch (Exception e)
        {
            sBuf = "Error write to the serial port: " + e.Message;
            bWriteOk = false;
        }
        return bWriteOk;
    }
```

21.3.4 Terminal console application

A console application was developed in C# for testing out the serial port, and NMEA and OBD protocols are two of the protocols that are included. A console application is useful for making a test version and the developed application is developed for testing different type of serial port operations. These operations can be to test the hardware of software of the computer by loopback testing, just dumping all the information transmitted and received regardless of any protocols, a general terminal application able to transmit commands, and test out different types of data communication protocols. The application was developed according to the class diagram shown in Figure 21.14.

Menu selections

The output from the MainLoop() method is shown in Figure 21.15, showing the selectable options in the application.

The following options are available in this terminal console application, but not all of these options will be explained in this section:

Key	Description	Explained
N	Select a serial port, based on available ports in the system	X
O	Open the selected serial port, choosing the right baud rate	X
D	Input the test string used for the loopback testing of the port	
L	Loopback testing of the selected serial port	X
R	Dumping of raw data from the serial port	(X)
G	Display GPS data from the NMEA protocol (S: Start/Stop button)	(X)
B	Display OBD data from the OBD protocol (S:Start/Stop button)	
M	Display Modbus data from the Modbus protocol (S:Start/Stop button)	
S	Generic serial port terminal, transmit commands and receive data	
Q	Quit the terminal console application.	

The first option should be to select a serial port. In this case the application is running on a Windows system so the serial port names will be COMx where x is a number from 1 to N, depending on the number of available serial ports. The user must use the 'N' button to make this selection, and the output from the application is shown in Figure 21.16.

On top of the screen is the current serial port selected, in this case None. At bottom is a list of avaialble serial ports, COM1 and COM7, and the '2' button is used to choose serial port COM7. The

```
---- Console Terminal : Main Menu ----
Current COM port:  None
----------------------------------------
N: Select a COM port
O: Open a selected COM port
D: Input the loopback test string
L: Test loopback of COM port data
R: Raw dumping of COM port data
G: GPS data from NMEA protocol (S:Start/Stop)
B: OBD protocol (S:Start/Stop)
M: Modbus protocol (Limited) (S:Start/Stop)
S: Serial Terminal (Tx commands)
Q: Quit console terminal

Select a function:n
Number of available COM ports =
1: COM1
2: COM7
Select id for COM port:
```

Figure 21.16: The 'N' option in the software for selecting a serial port. Available serial ports in this system is COM1 and COM7.

```
---- Console Terminal : Main Menu ----
Current COM port: COM7
----------------------------------------
N: Select a COM port
O: Open a selected COM port
D: Input the loopback test string
L: Test loopback of COM port data
R: Raw dumping of COM port data
G: GPS data from NMEA protocol (S:Start/Stop)
B: OBD protocol (S:Start/Stop)
M: Modbus protocol (Limited) (S:Start/Stop)
S: Serial Terminal (Tx commands)
Q: Quit console terminal

Select a function:o Open serial port ..

Available baudrates:
1: 2400
2: 4800
3: 9600
4: 19200
5: 38400
6: 115200
Select baudrate for the COM port:
```

Figure 21.17: The selection of a baud rate for open the COM7 serial port.

next operation is to open this serial port with the appropriate baud rate like the snippet code shown for the Open() method. It is of course possible to combine the selection of a serial port and open a serial port in the same menu selection if wanted. The menu option 'O' will give the output as shown in Figure 21.17 with the selected serial port on the top, and a list of available baud rates at the bottom.

Use the button for the number in front of the baud rate to select the valid baudrate, and the respons shown in Figure 21.18 will be the results. In this case the serial port COM7 is now opened ok and can be used by the application. An error will be displayed if the open() statement fails, as shown in the open() source code snippet.

The port is now open with the selected baud rate and the serial port can now be used for data communication.

Loopback testing

The first time using a serial port, a loopback testing can be useful to check that both the hardware and software is working. Loopback testing means that an interconnection between the transmitter pin and the received pin for the serial port must be inserted, and all the data transmitted for serial port should be received by the application. In this way both the transmitter and receiving section of the application, and the serial port will be testing. Figure 21.19 shows a serial port connector with a loopback test connector (to the left, Rx and Tx pins connected) and the NMEA protocol connector (in front, only Rx and Gnd pins connected).

The loopback connector is inserted and the 'D' option is selected to make the loopback test string to be "SerialPortTesting". Then the 'L' option is selected and the output from the application is shown in Figure 21.20. The application will add a counter id after each time reading the loopback test string, so the output should be "SerialPortTestingx" where x is a number from 1 and increasing. The read opeation is defined to read 16 characters from the serial port for each operation, the reason for the figure to display

```
Current COM port: COM7
---------------------------------------
N: Select a COM port
O: Open a selected COM port
D: Input the loopback test string
L: Test loopback of COM port data
R: Raw dumping of COM port data
G: GPS data from NMEA protocol (S:Start/Stop)
B: OBD protocol (S:Start/Stop)
M: Modbus protocol (Limited) (S:Start/Stop)
S: Serial Terminal (Tx commands)
Q: Quit console terminal

Select a function:o Open serial port ..

Available baudrates:
1: 2400
2: 4800
3: 9600
4: 19200
5: 38400
6: 115200
Select baudrate for the COM port:2 4800
Message=<Open <COM7> serial port OK!>
Press any key to clear ..
```

Figure 21.18: The '2' button is used to select 4800 as the baud rate, and the serial port is now open OK. The COM7 serial port can now be used by the application.

Figure 21.19: A serial port with a loopback serial port connector that can be inserted to test the serial port. The loopback connection is just interconnecting pin 2 and pin 3 of the serial port, the transmitter and receiver pins.

```
---- Console Terminal : Main Menu ----
Current COM port: COM11
---------------------------------------
N: Select a COM port
O: Open a selected COM port
D: Input the loopback test string
L: Test loopback of COM port data
R: Raw dumping of COM port data
G: GPS data from NMEA protocol (S:Start/Stop)
B: OBD protocol (S:Start/Stop)
M: Modbus protocol (Limited) (S:Start/Stop)
S: Serial Terminal (Tx commands)
Q: Quit console terminal

Select a function:l Loopback testing ..
Rx[1]=<SerialPortTestin>
Rx[2]=<g1SerialPortTest>
Rx[3]=<ing2SerialPortTe>
Rx[4]=<sting3SerialPort>
Rx[5]=<Testing4SerialPo>
Rx[6]=<rtTesting5Serial>
Rx[7]=<PortTesting6Seri>
Rx[8]=<alPortTesting7Se>
Rx[9]=<rialPortTesting8>
Rx[10]=<SerialPortTestin>
Rx[11]=<g9SerialPortTest>
Rx[12]=<ing10SerialPortT>
Rx[13]=<esting11SerialPo>
Rx[14]=<rtTesting12Seria>
Rx[15]=<lPortTesting13Se>
```

Figure 21.20: The console output from the loopback testing option.

only 16 characters on each line. However, all the characters are read so the serial port and the application is working. The screen update will stop when the loopback test connector is disconnected, or use the 'Q' button to stop this operation. This output also shows the challenges on receiving information on a serial port and merge the information into a valid structure. The receiving method may stop the reading at any time and position of the receiving information, depending on the impementation of reading the serial port. If reading too few characters, the application need logic to merge these characters into a valid message. If reading too many characters, the application will timeout and use more time in collecting the information. Also, reading too many characters may end up reading several messages at the same time, as shown in Figure 21.8.

Dumping of raw data

This can be a very useful option just to dump the all the caracters received from the serial port. Convert the byte[] buffer from the read() method to a string, and print the string on the console. This is shown in Figure 21.8 but remember that this requires an ASCII protocol with mainly printable characters. Any non-printable characters must be converted to printable characters like 0x08, 0x0a and 0x0d can be converted to <TAB>, <LF> and <CR>. For a binary protocol, like Modbus RTU, the information must be converted to printable characters before this option can be used.

Implement NMEA protocol

A console application can be a good tools to test out the first version of the protocol coding and decoding, to get some experience how to merge and extract the protocol information. Figure 21.8 shows the raw data for the NMEA protocol showing the structure of the message, and Figure 21.9 show the extraction of some of the information from the NMEA command GPGGA. The list in section 21.2.4 shows a sequence of OBD messages, with timestamps, both transmitted and received, for the OBD protocol. Both of these protocols are ASCII protocols meaning that the information consists of ASCII characters, where many of the ASCII characters are printable characters. The decoding of such protocols will be to use different string operations in C#. The general structure is 1) to find the start of the message, 2) find the end of the message to be sure that the whole message has been received, 3) any error checking if checksum is available, 4) subtract the different fields of the message depending on the field separator, and 5) decode the information wanted from these fields.

A source code snippet for one way of extracting information from a NMEA message. Just part of the method is included to get an idea of one possible way of extracting such information:

```
/// <summary>
int decodeNmeaMsg(string sNmeaMsg, out string sErrMsg)
/// Purpose: decode a NMEA message
/// Version: 1.0: 29-JAN-20: NOS
/// </summary>
{
    string[] sNmeaFields;
    string sNmeaCmd, sNmeaChksum;
    double dGpsTime, dLatitude, dLongitude;
    int iGpsQuality;

    /// Spilt the fields based on comma and star
    sNmeaFields = sNmeaMsg.Split(',','*');
    sErrMsg = "";
    /// The last field is the checksum, after the star
    sNmeaChksum = sNmeaFields[sNmeaFields.Length - 1];
    /// No test of the checksum - must be included in the final solution!
    /// Get the NMEA commands
    if (sNmeaFields[NMEA_CMD_OFFSET].Length > 2)
    {
        /// Remove the GP part of the message command
        sNmeaCmd = sNmeaFields[NMEA_CMD_OFFSET].Substring(2);
        if (string.Equals(sNmeaCmd, "GGA", StringComparison.OrdinalIgnoreCase) == true)
        {
            try
            {
                dGpsTime = Convert.ToDouble(sNmeaFields[NMEA_GGA_CMD_GPS_TIME_OFFSET]);
```

The message format of any protocol is important to understand when implementing such a protocol and a console application can be a very good way of testing such an implementation before integrating the protocol support into the final application.

21.4 Network communication programming in C#

21.4.1 Introduction

Data communication is important for communication between computers, both on local area networks (LAN)(intranet) and on Internet. The previous chapter looked into serial port communication, using serial ports for external communication. In this chapter will the focus be network data communication, meaning data communication over a network. Serial port data communication has a limited number of transmitters and receivers on the same communication media, and normally the communication media has also a limited length as well. The maximum length is also often depending on the baud rate for serial data communication. Network data communication is using a more complex structure with hardware devices like switches, routers, gateways and firewalls to link network system together, route the messages between these network systems, and using firewalls to protect for which messages that can be received by each sub network.

This means that a network protocol will be more complex than a serial port protocol, however the OSI model can also be used to structure the functionality of these protocols. The serial port protocols depends mainly on the lowest layer in the OSI model, while network protocols will depend on several layers in this protocol model. Still, network communication must still depend on some sort of serial data communication, like the RS-485 protocol used in many network protocols, as the physical layer. The physical layer can still use either wired, optical or wireless physical connection for data communication.

21.4.2 OSI model

The OSI model is an abstract model used to structure any communication protocol into the functionality of seven layers, as shown on the left side in Figure 21.21. The serial port communication described in the

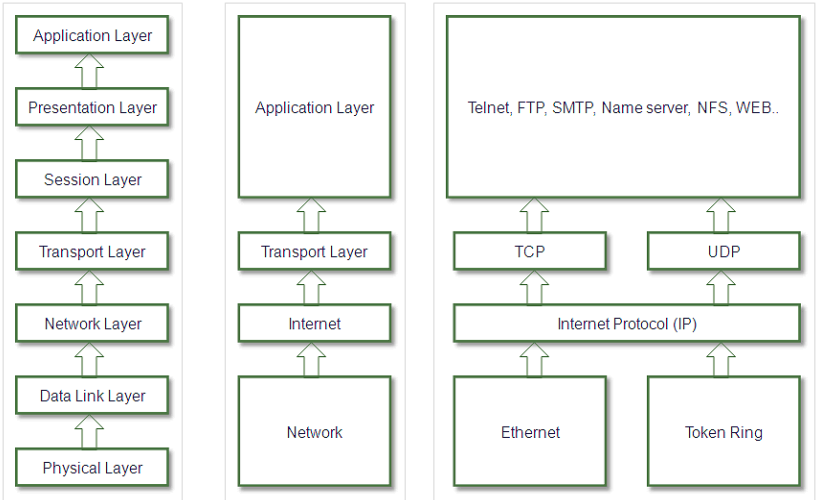


Figure 21.21: The OSI model, the general model to the left, the Internet structure in the middle, and the TCP/IP solution to the right.

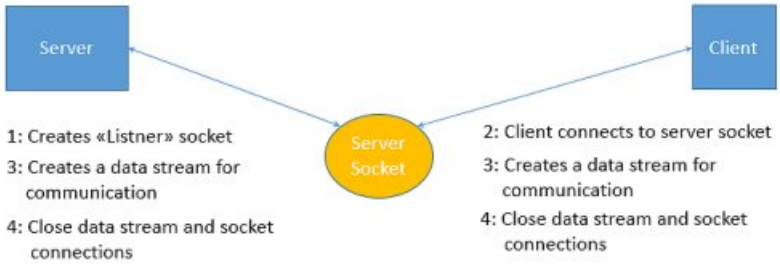


Figure 21.22: The socket connection steps for communication for server and client objects in C#.

previous chapter describes the functionality in the physical layer of the OSI model, while the network data communication is described by the functionality of the first four layers, physical, data link, network and transport layers.

This chapter will not focus on the OSI model(Stevens 1990)(Stallings 2001), but on the TCP/IP protocol (Comer 1991) (Stevens 1990) (Crowcroft & Phillips 2002) (Bentham 2002) used as one protocol for the transport layer and corresponding support from C# libraries. The structure of the TCP/IP protocol is indicated to the right in Figure 21.21. The figure shows that the TCP/IP protocol is using the functionality of the Internet Protocol (IP) for communicating with the nodes on the network, where the main functionality is logical addressing. A node in one local area network (LAN) connected to the Internet can communicate with another node connected to another LAN on the Internet using an IP protocol. The IP protocol will use either the Ethernet or Token Ring protocols for handling the functionality of the two lowest layers in the OSI model, as shown to the right in the figure. The Ethernet protocol is the most used protocol today for handling the Internet communication on these lowest layers.

21.4.3 TCP/IP protocol

The TCP/IP protocol is a network protocol and is based on the client/server concept. This concept define there must be an existing server with some sort of information that a client wants get access to. In data communication a socket is used to establish a connection between two or more nodes (computer with communication capabilities), and these sockets will be used to send data between the nodes. For the TCP/IP protocol will the server first create a listener socket, and any client will create a socket for connecting to the listener socket of a specific server. For the TCP/IP protocol will the IP addresses be used by the both the server and clients to create the socket for communicating, so the IP address of the server should be known. When the socket connection is established, the nodes can exchange information using the socket connection (read and write methods). Figure 21.22 shows the socket connection where the server first make a listener socket and waits for any client socket to connect to that socket (step 2).

When the connection is fulfilled will both systems make a data stream for exchanging information, by reading and writing. Note that this will be a multitasking system as these to applications will run independent of each other. Ending the connection by closing the data stream and socket connection at both ends.

21.4.4 Socket programming in C#

Socket programming (Stevens 1990) (Crowcroft & Phillips 2002) (Bentham 2002) (Mahmoud 2004) is based on a bidirectional communication link between two applications, a server application and a client application, see Figure 21.22. Socket programming is using the IP address so the TCP/IP protocol needs to be implemented in the system, but most nodes today have the TCP/IP protocol included. The C# socket library is included in the System.Net and System.Net.Sockets namespaces, so these two libraries must be included in your application, as shown in the following list::

```
using System;
using System.Text;
using System.Net.Sockets;
using System.Net;
```

The first step is to get the IP address of the, either the running node or the remote node. Use the following commands to get a list of the available IP addresses for the node:

```
ipHostInfo = Dns.GetHostEntry(Dns.GetHostName());
ipAddress = ipHostInfo.AddressList[0];
```

The first line gets the information from the Domain Name System, and the information is using the name of this node. The second line gets a list of available IP addresses for this node. The reason for several IP addresses is that the system has support for both IPv4 and IP v6 protocols with a different type of IP addresses. In this case is the first IP address used.

For the server will the next steps be:

```
tcpListener = new TcpListener(ipAddress, iSocketPortId);
tcpListener.Start(5);
tcpClient = tcpListener.AcceptTcpClient();
tcpNetworkStream = tcpClient.GetStream();
```

First make a listener socket that is used for a client to connect, and start this listener socket. In this case it allows for up to five clients to connect at the same time. The AcceptTcpClient() command will wait (block) until a client is connecting to this socket, with the correct IP address and socket port number. Accepting the client connection, a network data stream is created to allow for reading and writing data between the server and client.

The commands will close the connected after finishing the communication:

```
tcpNetworkStream.Close();
tcpClient.Close();
```

For the client will the next steps be:

```
tcpClient = new TcpClient(ipAddress.ToString(), iSocketPortId);
tcpNetworkStream = tcpClient.GetStream();
```

The TcpClient() command will connect to an existing socket connection, with the correct IP address and socket port number. Accepting the server connection, a network data stream is created to allow for reading and writing data between the server and client.

The network data stream can now be used for transmitting and receiving bytes using the Read() and Write() methods for the data stream references. The Read() method will read a number of bytes, and return the number of bytes read. The Read() method can also be extended with the buffer offset and maximum number of bytes to read. The Read() method reading a number of bytes and converting to a printable string:

```
byte[] baBuffer = new byte[64];
int iLen = tcpNetworkStream.Read(baBuffer);
String sBuffer = Encoding.ASCII.GetString(baBuffer);
sBuffer = sBuffer.TrimEnd('\0');
```

The Write() method will write a number of bytes, no return. The Write() method can also be extended with the buffer offset and maximum number of bytes to write. The Write() method making a string, converting the string into a byte array, and write the byte array:

```
string sBuffer = DateTime.Now.ToString() + ": Server command=<" + sBuffer + ">";
byte[] baBuffer = Encoding.ASCII.GetBytes(sBuffer);
tcpNetworkStream.Write(baBuffer);
```

The commands will close the connected after finishing the communication:

```
tcpNetworkStream.Close();
tcpClient.Close();
```

Note that these socket methods are system methods and try/catch blocks must be used to make the implementation more robust. The Read() and Write() methods handling only a byte array is the challenge for TCP/IP data communication as no standard protocol exists for handling the data structure in these byte arrays. If your application is receiving the byte stream $160A326790B386D2B4A8324447$ you will not have any information about this information unless informed by the sender application system. This is also one of the reason for many proprietary protocols for Internet of Thing (IoT) systems, using the TCP/IP protocol do NOT define any data structure of the byte buffers. The solution will often be to define a new protocol on top of the TCP/IP protocol for handling the application layer of the data communication, see Figure 21.21.

These socket methods will be used in a console test application to show the implementation of socket communication, and a simple protocol will be defined on top of the TCP/IP protocol to handle the messages between the socket server and client.

21.4.5 Console test application

A simple console application will be developed to test out the concept of client/server programming using sockets in C#. The application will also show the challenges with the TCP/IP protocol as this protocol is just defining a data buffer for exchanging information, no interpreting of the information in the data buffer. A very simple protocol has been define, two string commands, for communication between the server and the client. These commands are string based, "LIST" and "Quit"; "List" will display the IP address(es) of the server node and "Quit" will terminate the server/client connection. The TCP/IP protocol is also used for IoT and IIoT devices and these devices will have the same challenge with the TCP/IP protocol, and is the reason for many different IoT protocols, both proprietary and open protocols. These protocols will define the interpretation of the TCP/IP buffer, and support the three upper layers in the OSI model, normally only the "Application layer". The "presentation layer" contains any security functionality and should be integrated into any of these protocols.

The first task when designing a new software application is to make a use case diagram, and this diagram is shown in Figure 21.23.

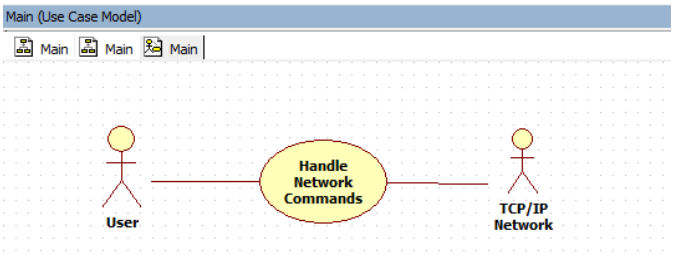


Figure 21.23: The use case diagram for the console socket test application, only one use case for testing commands between a local socket server and socket client.

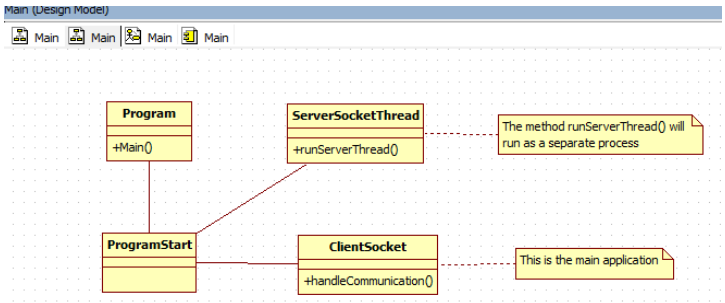


Figure 21.24: The one tier architecture shown in a class diagram, for the network test application.

Since this is a only an application for testing/showing the socket functions in C# for network communication, only an one tier application is developed. If the application is going to be extended with new functionality, a two or three tier application should have been developed. The console application consists of four classes, as shown in the class diagram in Figure 21.24.

The main method of the server class (*RunServerThread()*) will run as a separate process, making this a multitasking system. The main method in the client class (*HandleCommunication()*) will contain the functionality of the client, with interface to the user selecting any commands. Program class is generated by the Visual Studio IDE, the ProgramStart class is added to make an object, and this object will first start the server thread as part of the server object, and then the main loop in the client object. The output from this multitasking system is shown in Figure 21.25.

The user first send the command "test" to the server, and the server respond with an invalid command message. Then the user tries the "verify" command with the same response from the socket server. The next command is "list", listing the IP addresses of the server, and finally the "quit" command first terminating the socket connection, then the server process and then the client application. Note that for simplicity will the server write the list messages to the console, the best solution is however for the server to send this information as text messages to the client, and for the client to display this information on the console. Why is this a better solution?

The simple protocol implemented in the application is the two text commands "*LIST*" and "*QUIT*" checked for in the server thread. Also note that the information is sent as plain text on the socket stream meaning that the information can be seen by anybody with access to the communication media and the TCP/IP protocol. Part of a data security policy can be to crypt/decrypt the byte buffers when transmitting and receiving the data. Wireshark[3] is one of the free applications that can be used to visualize data from external TCP/IP messages. The source code for the network console test application:

```
 ////////////////////////////////////////////////////////////////////////////////////
///
/// Purpose: make a console application showing the usage of TCP/IP socket programming,
///          one thread running as a server, and the application running as a client
/// Version: 19-OCT-20: NOS
/////////////////////////////////////////////////////////////////////////////////////
using System;
```

[3] Wireshark, a free and open-source network data packet analyzer, www.wireshark.org, Nov-19.

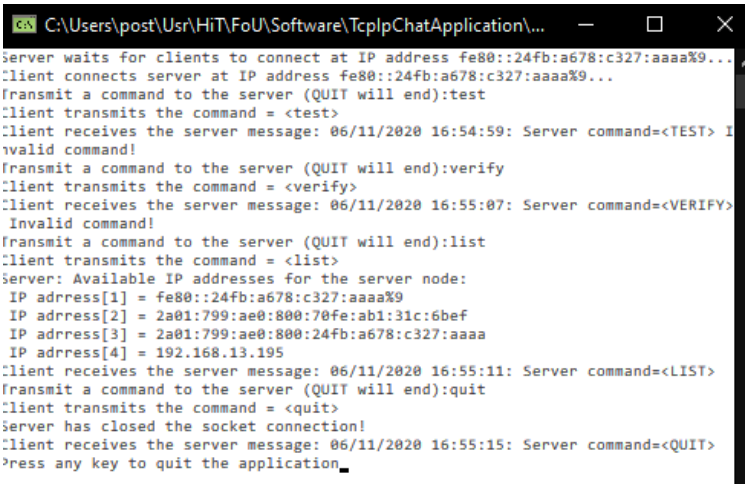


Figure 21.25: The output for the network test application, the socket server running as a background prosess (thread) and the socket client running as the foreground application with user interaction.

```
using System.Text;
using System.Net.Sockets;
using System.Net;
using System.Threading;
///
namespace NosConsoleSocket
{
    ////////////////////////////////////////////////////////////////////////////////////////
    class Program
    {
        /// <summary>
        static void Main()
        /// Purpose: The main entry point for the application.
        /// Version: 1.0: 20-OCT-20: NOS
        /// </summary>
        {
            /// Convert from a static to a dynamic object
            new ProgramStart();
        }

    ////////////////////////////////////////////////////////////////////////////////////////
    class ProgramStart
    {
        const int iSocketPortNumber = 9999;
        ServerSocketThread rServerSocketThread;
        ClientSocket rClientSocket;
        /// <summary>
        public ProgramStart()
        ///
        /// Purpose: the startup object for handling the server thread and client
        /// Version: 21-OCT-20: NOS
        /// </summary>
        {
            try
            {
                /// start the socket server first as a separate thread
                rServerSocketThread = new ServerSocketThread(iSocketPortNumber);
                /// Wait to let the server create the connection point for clients
```

```
                Thread.Sleep(500);
                /// start the socket client to communicate with the server
                rClientSocket = new ClientSocket(iSocketPortNumber);
                rClientSocket.handleCommunication();
            }
            catch (exception ex)
            {
                Console.WriteLine(" Application error: " + ex.Message());
            }
        }
    }
    ////////////////////////////////////////////////////////////////////////////////
    class ServerSocketThread
    {
        int iSocketPortId;
        bool bServerThreadFlag;
        Thread thServerSocket;
        /// <summary>
        public ServerSocketThread(int iSPN)
        ///
        /// Purpose: constructor making the server thread using sockets
        /// Version: 22-OCT-20: NOS
        /// </summary>
        {
            iSocketPortId = iSPN;
            bServerThreadFlag = true;
            thServerSocket = new Thread(new ThreadStart(this.runServerThread));
            thServerSocket.Start();
        }
        /// <summary>
        private void runServerThread()
        ///
        /// Purpose: the server thread running as a separate process, handling the server
        ///          functions.
        /// Version: 22-OCT-20: NOS
        /// </summary>
        {
            IPHostEntry ipHostInfo;
            IPAddress ipAddress;
            TcpListener tcpListener;
            TcpClient tcpClient;
            NetworkStream tcpNetworkStream;
            byte[] baBuffer;
            string sBuffer;
            int iCnt;
            bool bCmdValid;

            try
            {
                /// get the IP address information this node
                ipHostInfo = Dns.GetHostEntry(Dns.GetHostName());
                ipAddress = ipHostInfo.AddressList[0];
                /// make a TCP connection point based on the IP address and the port number
                tcpListener = new TcpListener(ipAddress, iSocketPortId);
                /// Start the listner connected, allow up to five clients at the same time
                tcpListener.Start(5);
                /// wait for a client to connect to this conenction point (port ID)
                Console.WriteLine("Server waits for clients to connect at IP address " +
```

```
                    ipAddress.ToString() + "...");
        tcpClient = tcpListener.AcceptTcpClient();
        /// a client has connected, make a data stream for communicating with the client
        tcpNetworkStream = tcpClient.GetStream();
        /// loop for communicating with the client, let the client close the connection.
        while (bServerThreadFlag == true)
        {
            /// wait for receiving a message from the client
            baBuffer = new byte[64];
            tcpNetworkStream.Read(baBuffer);
            sBuffer = Encoding.ASCII.GetString(baBuffer);
            sBuffer = sBuffer.TrimEnd('\0');
            /// check for a valid message from the client
            /// Note that the TCP/IP protocol just define a byte data buffer, no data
            /// structure. You must define your own protocol based on this buffer. The same
            /// for IoT communication, often defining a propritary protocol on top of TCP/IP
            /// Define a simple protocol, based on string commands; LIST and QUIT.
            /// LIST shows the IP addresses for the server and QUIT terminates the server
            /// connection.
            sBuffer = sBuffer.ToUpper();
            if (string.Equals(sBuffer, "QUIT") == true)
            {
                bServerThreadFlag = false;
                bCmdValid = true;
            }
            else if (string.Equals(sBuffer, "LIST") == true)
            {
                Console.WriteLine("Server: Available IP addresses for the server node:");
                for (iCnt = 0; iCnt < ipHostInfo.AddressList.Length; iCnt++)
                {
                    Console.WriteLine(" IP adrress[" + (iCnt + 1).ToString() + "] = " +
                        ipHostInfo.AddressList[iCnt].ToString());
                }
                bCmdValid = true;
            }
            else
            {
                bCmdValid = false;
            }
            /// Send an answer back to the client
            sBuffer = DateTime.Now.ToString() + ": Server command=<" + sBuffer + ">";
            if (bCmdValid == false)
            {
                sBuffer = sBuffer + " Invalid command!";
            }
            baBuffer = Encoding.ASCII.GetBytes(sBuffer);
            tcpNetworkStream.Write(baBuffer, 0, baBuffer.Length);
        }
        /// The connection is closed, close the data stream and socket.
        tcpNetworkStream.Close();
        tcpClient.Close();
        Console.WriteLine("Server has closed the socket connection!");
    }
    catch (Exception ex)
    {
        /// Display any errors for the server socket thread
        Console.WriteLine("Server error: ", ex.Message);
    }
```

```
        }
    }
    ////////////////////////////////////////////////////////////////////////////
    class ClientSocket
    {
        const int MAX_RX_LEN = 128;
        int iSocketPortId;
        IPHostEntry ipHostInfo;
        IPAddress ipAddress;
        TcpClient tcpClient;
        NetworkStream tcpNetworkStream;
        string sTxBuf, sRxBuf;
        byte[] baBuf;
        /// <summary>
        public ClientSocket(int iSPN)
        ///
        /// Purpose: constructor for the client class
        /// Version 23-OCT-20: NOS
        /// </summary>
        {
            iSocketPortId = iSPN;
        }
        /// <summary>
        public void handleCommunication()
        ///
        /// Purpose: the client application, communicating with the server process
        /// Version: 23-OCT-20: NOS
        /// </summary>
        {
            try
            {
                /// Get the IP address of this node
                ipHostInfo = Dns.GetHostEntry(Dns.GetHostName());
                ipAddress = ipHostInfo.AddressList[0];
                /// connect to server running on localhost at the specific port number id
                tcpClient = new TcpClient(ipAddress.ToString(), iSocketPortId);
                Console.WriteLine("Client connects server at IP address " + ipAddress.ToString() +
                                  "...");
                /// make the data stream for reading and writing with the server
                tcpNetworkStream = tcpClient.GetStream();
                Console.Write("Transmit a command to the server (QUIT will end):");
                sTxBuf = "";
                /// handle the simple string based protocol for the server communication
                while (string.Compare(sTxBuf, "QUIT", true) != 0)
                {
                    /// get the next command from the user
                    sTxBuf = Console.ReadLine();
                    Console.WriteLine("Client transmits the command = <" + sTxBuf + ">");
                    baBuf = Encoding.ASCII.GetBytes(sTxBuf);
                    /// transmit the command to the server, and wait for any answer
                    tcpNetworkStream.Write(baBuf, 0, baBuf.Length);
                    /// release cpu time for other processes while waiting for server response
                    Thread.Sleep(100);
                    /// Read any answer from the server
                    baBuf = new byte[MAX_RX_LEN];
                    tcpNetworkStream.Read(baBuf, 0, baBuf.Length - 1);
                    sRxBuf = Encoding.ASCII.GetString(baBuf);   // convert byte array to string
                    Console.WriteLine("Client receives the server message: " + sRxBuf);
```

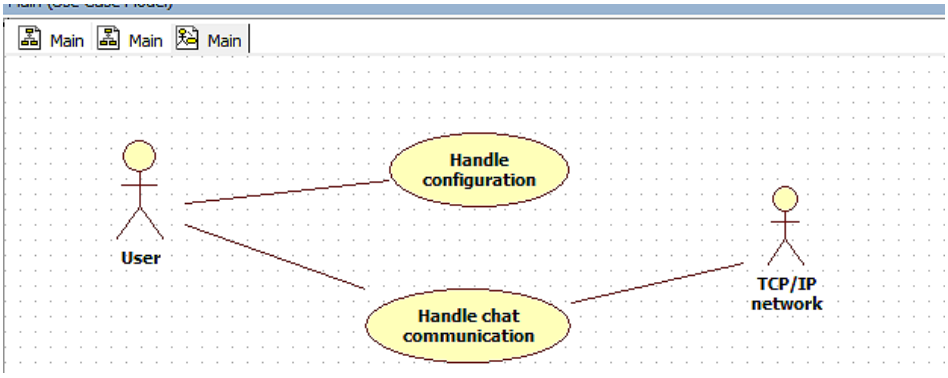


Figure 21.26: The use case diagram for the main functionality of the chat application, with two use cases. One use case for handling configuration and one use case for handling the chat communication.

```
                    if (string.Compare(sTxBuf, "QUIT", true) != 0)
                    {
                        Console.Write("Transmit a command to the server (QUIT will end):");
                    }
                }
                /// Close the socket communication with the server
                tcpNetworkStream.Close();
                tcpClient.Close();
                Console.Write("Press any key to quit the application");
                Console.ReadKey();
            }
            catch (Exception ex)
            {
                /// Handle any client erros
                Console.WriteLine("Client error: ", ex.Message);
            }
        }
    }
}
///
////////////////////////////////////////////// EOC ////////////////////////////////////////////
```

21.4.6 GUI Chat application

The console test application is running both the server and the client on the same node which does not take any advantages of network communication. A Graphical User Interface (GUI) application that can be used as a chat application between two nodes is developed to show full utilization of network communication. With this application two user should be able to communicate (chat) running the same application on two different nodes in a local area network (LAN). It should also be possible to use the application over Internet but requires that the selected port number is available for communication through any firewalls. Note that the usage of this application should trigger your "antivirus/internet security" protection as this application is trying to make a TCP/IP connection on a non-standard port number.

The use case diagram for application is shown in Figure 21.26, showing the main user functions for the application.

The first use case will be handle the configuration of the application (server/client mode, IP address and socket port number) and the second use case for the chat function where the users can chat with each other by writing messages. The connection will be part of the second use case as this will involve both users, where the remote user is shown as the TCP/IP network actor. The structure of the application will be a three tier architecture, one tier for the user interface, one tier for the business logic for handling the communication modes and exchange of messages, and the tier for the data layer handling the socket

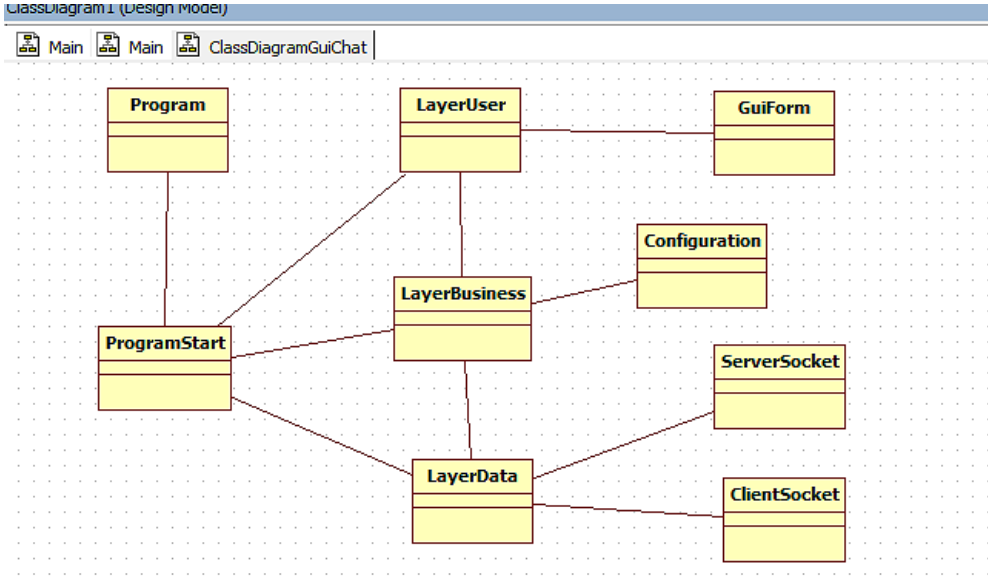


Figure 21.27: The three tier architecture class diagram of the GUI chat application, with the program startup classes, the tier classes, socket mode classes and a class for configuration parameters.

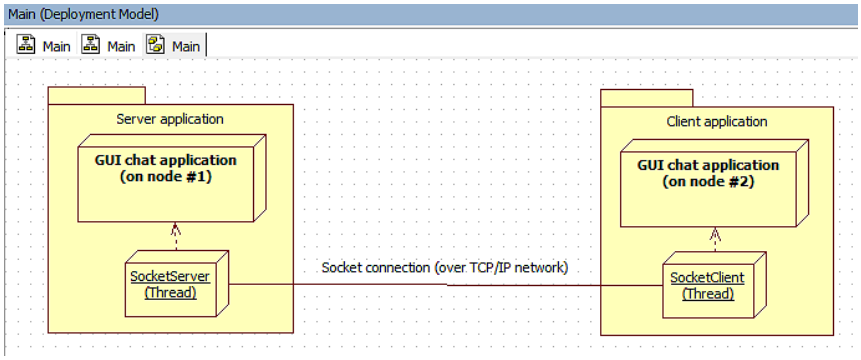


Figure 21.28: The deployment diagram for the application, running two applications. One application as the server, one application as the client, and using a socket connection as the TCP/IP communication.

communication. The data layer can also be extended with a log feature saving the messages on a text file. The class diagram for this structure is shown in Figure 21.27.

The most important classes is shown in this class diagram, two classes (Program and ProgramStart) for starting the application, three classes for handling the tier functionality of the application (LayerUser, LayerBusiness and LayerData), one class for the configuration parameters, and two classes handling the threads for server and client socket communication. Only one of these threads will be running in a specific application, depending on the configuration. A thread is very useful to let the socket communication be a background process independent of any user interaction. Figure 21.28 shows the deployment diagram, showing the two instances of the application running in a network. One application configured as the socket server, with the ServerSocket thread object activated, and one application configured as the socket client, running the ClientSocket thread object.

The C# library and methods are the same as used for the console application, however this application is extended with a configuration section and a three tier architecture. The user interface for the application is shown in Figures 21.30 and 21.29, where Figure 21.30 shows the configuration and some messages for the client application and Figure 21.29 shows the configuration and some messages for the server application.

The user interface consists of five sections, one section with the menu bar and the host name on the top, two message sections with the transmitted messages to the left and the received messages to the right. The section in the lower left corner is status information from the application, in this case the connection and disconnection to any remote nodes. The last section, in the lower right corner, is the

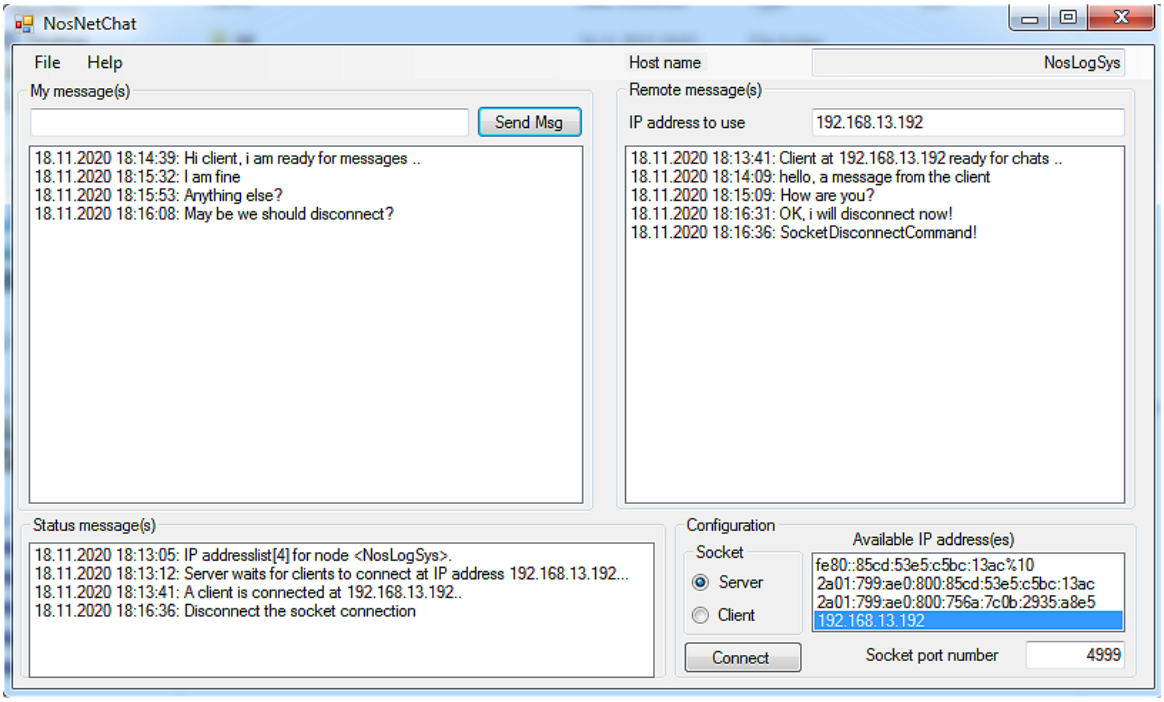


Figure 21.29: The configuration and messages for the server application.

configuration section, where the user selects the socket (server or client), the port number and the IP to use.

The setup of the system should start with the server application, and the configuration section. Select an unused port number, a good choice is to select a port number above 1024. The valid port numbers are 0 to 65535. Note that using an unused port number above 1024 should result in a warning from the Antivirus/Internet protection of your system as these ports will also be used for a cyber security attach of computers. Select the socket type, server or client, and an IP address. A socket connection consists of a socket number and an IP address. In the upper right corner of the application in Figure 21.29 the name of node is shown, the server node name for the socket the connection. The IP addresses are shown in the list in the lower right corner, shown both IPv4 and IPv6 addresses for this node. A good choice can be to use the IPv4 addresses as this is more easy for the user of the client application to input the IP address. Click on the address that you want to use, and the IP address will be shown in the right upper corner. Use the *Connect* button to start the server socket thread. Messages in the status window, lower left corner, shows that IP addresses are listed, and that the server has started with a specific IP address. The Connect button will change the text to *Disconnect* when connected.

The setup of the client application must use the same setting as the server application, as shown in Figure 21.30. The node name in the upper right corner is the name of the client node, not the same as the server node. The next field in the upper right corner must be the IP address of the server, so the same IP address as in Figure 21.29. Use the same IP address and socket number as the server, and use the *Connect* button to connect to the server socket. Information in the status windows shows the status of this connection. When connected, input a message in the input field in the upper left corner and use the *Send Msg* button to the send the message to the remote node. The upper left list window shows the messages sent from the node, and the upper right list window shows the messages received from the remote node.

The source code for this application will not be included, but is using the same methods as shown for the console application. Some comments with code snippets will however be included for specific part of this application, as a multitasking (multithreading) GUI application will include some challenges with sending events between the threads. The software architecture is a three tier architecture for this application, as shown in Figure 21.27 and the normal message flow is to use methods to get information from a lower tier. The challenge is to get information from a lower tier up to a higher tier when receiving external information and in this application is the publish/subscriber concept (best practice - design pattern) used. The publisher/subscriber concept consist of one publisher, active on a lower tier, and activated when receiving external information. Any method in a higher tier can subscribe on these

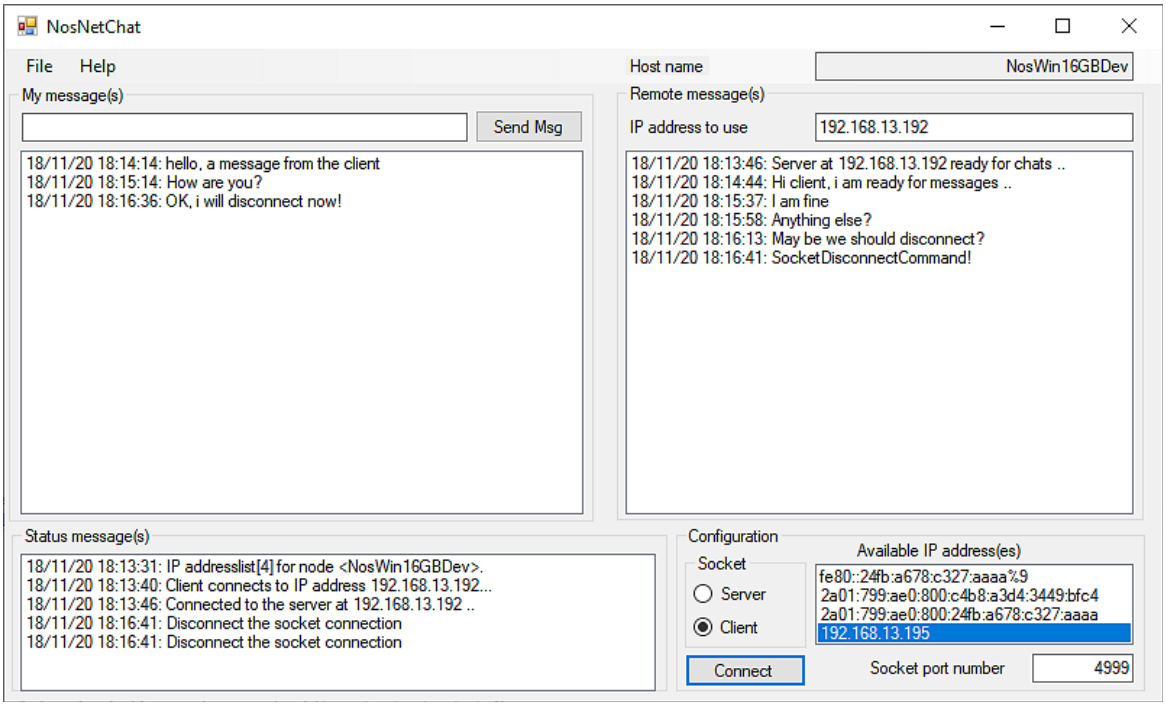


Figure 21.30: The configuration and messages for the client application.

publisher, and be informed when new external information is received.

The data flow in this application is, with reference to Figure 21.27, is when sending a message is to make the message in the user tier, send the message buffer down to business tier, further down to the data tier, and the active thread for sending to the remote node. The challenge is receiving a message from the remote node. One solution can be to let the user tier poll the socket thread all the time but this will waste cpu power and memory, here the publisher/subscribe concept should be used. The threads will be the publishers, and the user tier will be the subscriber, receiving an event whenever new information is received from the remote node.

The source code snippets for the publisher/subscriber logic:

1 : define the callback methods as delegate methods in the header of the LayerUser class:

```
/// <summary>
/// Purpose: event handler with the UiEventArgs object as information
/// </summary>
public delegate void HandlerUiMsg(object rSender, UiEventArgs eUi);
delegate void AddReceivingMsgCallback(string txt);
delegate void AddStatusMsgCallback(string txt);
```

2 : define the event handler in both the LayerUser and LayerData classes:

```
HandlerUiMsg eListenUiMsg = new HandlerUiMsg(OnHandlerUiMsg);
rDataLayer.ePublishUiMsg += eListenUiMsg;
```

3 : define the event handlers. Note that the events has to be sent between different threads so these handlers must be thread safe:

```
/// <summary>
public void OnHandlerUiMsg(object rsender, UiEventArgs eArgs)
///
/// Purpose: handling UI event commands
```

```
        /// Version: 14-NOV-20: NOS
        /// </summary>
        {
            switch (eArgs.iCmd)
            {
                case (int)Constants.UiEventCmd.MsgRxList:
                    addReceivingMsgThread(eArgs.sMsg);
                    break;
                case (int)Constants.UiEventCmd.MsgStatusList:
                    AddStatusMsgThread(eArgs.sMsg);
                    break;
                case (int)Constants.UiEventCmd.MsgSocketButton:
                    rGuiForm.buttonSocketConnect.Text = eArgs.sMsg;
                    break;
                default:
                    addStatusMsg("Error: Rx invalid UI event cmd = " + eArgs.iCmd.ToString());
                    break;
            }
        }

        /// <summary>
        public void AddStatusMsg(string sStatusMsg)
        ///
        /// Purpose: add a messge to the status window
        /// Version: 13-NOV-20: NOS
        /// </summary>
        {
            string sMsg;

            sMsg = DateTime.Now.ToString() + ": " + sStatusMsg;
            rGuiForm.listBoxStatusMessages.Items.Add(sMsg);
        }
        /// <summary>
        public void AddStatusMsgThread(string sRxMsg)
        ///
        /// Purpose: update the status window in the GUI, threadsafe
        /// Version: 1.0: 11-OCT-20: NOS
        /// </summary>
        {
            // InvokeRequired compares the thread IDs
            if (rGuiForm.listBoxStatusMessages.InvokeRequired == true)
            {
                try
                {
                    AddStatusMsgCallback rCallback = new AddStatusMsgCallback(addStatusMsgThread);
                    rGuiForm.Invoke(rCallback, new Object[] { sRxMsg });
                }
                catch
                {
                    // No support, normally at startup or close time
                }
            }
            else
            {
                AddStatusMsg(sRxMsg);
            }
        }
        /// <summary>
```

```
    private void AddReceiveMsg(string sRxMsg)
    ///
    /// Purpose: add a messge to the receive message window
    /// Version: 13-NOV-20: NOS
    /// </summary>
    {
        string sMsg;

        sMsg = DateTime.Now.ToString() + ": " + sRxMsg;
        rGuiForm.listBoxRxMessages.Items.Add(sMsg);
    }
    /// <summary>
    public void AddReceivingMsgThread(string sRxMsg)
    ///
    /// Purpose: update the receive window in the GUI, threadsafe
    /// Version: 1.0: 11-OCT-20: NOS
    /// </summary>
    {
        // InvokeRequired compares the thread IDs
        if (rGuiForm.listBoxRxMessages.InvokeRequired == true)
        {
            try
            {
                AddReceivingMsgCallback rCallback = new
                                        AddReceivingMsgCallback(addReceivingMsgThread);
                rGuiForm.Invoke(rCallback, new Object[] { sRxMsg });
            }
            catch
            {
                // No support, normally at startup or close time
            }
        }
        else
        {
            AddReceiveMsg(sRxMsg);
        }
    }
```

4 : define the event class for the event:

```
/// <summary>
public class UiEventArgs : EventArgs
///
/// Purpose: a public class for event messages for UI publisher/subscribe
/// Version: 14-NOV-20: NOS
/// </summary>
{
    public int iCmd;
    public int iCode;
    public string sMsg;
}
```

5 : define the event handler in the LayerData class:

```
    /// <summary>
    public void publishUiEvent(UiEventArgs eArgs)
```

```
///
/// Purpose: publisher event methods for sending events to the UI layer
/// Version: 13-NOV-20: NOS
/// </summary>
{
    if (ePublishUiMsg != null)
    {
        ePublishUiMsg(this, eArgs);
    }
}
```

The souce code snippets for making the either the server socket thread or the client socket thread:

```
/// <summary>
private bool makeSocketConnection(int iSocketMode, int iSockId, string sIpAddress)
///
/// Purpose: make a socket connection, ether server or client
/// Version: 12-NOV-20: NOS
/// </summary>
{
    bool bValid;

    ipAddress = IPAddress.Parse(sIpAddress);
    iSocketPortId = iSockId;
    switch (iSocketMode)
    {
        case (int)Constants.Socket.Server:
            publishEventToUi((int)Constants.UiEventCmd.MsgStatusList,
                "Server waits for clients to connect at IP address " + sIpAddress +
                "...", iSocketMode);
            iActiveSocketConnection = (int)Constants.Socket.Server;
            /// Start the server thread
            thSocketThread = new Thread(new ThreadStart(this.ServerSocketThread));
            bValid = true;
            break;
        case (int)Constants.Socket.Client:
            publishEventToUi((int)Constants.UiEventCmd.MsgStatusList,
                "Client connects to IP address " + sIpAddress + "...", iSocketMode);
            iActiveSocketConnection = (int)Constants.Socket.Client;
            /// Start the client thread
            thSocketThread = new Thread(new ThreadStart(this.ClientSocketThread));
            bValid = true;
            break;
        default:
            publishEventToUi((int)Constants.UiEventCmd.MsgStatusList,
                "Socket error: Invalid socket mode = " + iSocketMode.ToString() + "!",
                iSocketMode);
            iActiveSocketConnection = (int)Constants.Socket.None;
            bValid = false;
            break;
    }
    thSocketThread.Start();
    publishEventToUi((int)Constants.UiEventCmd.MsgSocketButton, "Disconnect",
                iActiveSocketConnection);
    return bValid;
}
```

The souce code snippets for the server socket thread, including the receive method:

```
/// <summary>
public void ServerSocketThread()
///
/// Purpose: thread for handling the server thread
/// Version: 16-NOV-20: NOS
/// </summary>
{
    byte[] baBuffer;
    /// make a TCP connection point based on the IP address and the port number
    tcpListener = new TcpListener(ipAddress, iSocketPortId);
    /// Start the listner connection to let clients to connect
    tcpListener.Start();
    /// wait for a client to connect to this conenction point (port ID)
    tcpClient = tcpListener.AcceptTcpClient();
    /// a client has connected, make a data stream for communicating with the client
    tcpNetworkStream = tcpClient.GetStream();
    /// send a start message
    publishEventToUi((int)Constants.UiEventCmd.MsgStatusList,
        "A client is connected at " + ipAddress.ToString() + "..", 0);
    sendSocketMessage("Server at " + ipAddress.ToString() + " ready for chats ..");
    /// Connected, wait for messages
    while (bRunFlag == true)
    {
        /// wait for receiving a message from the client
        baBuffer = new byte[64];
        if (tcpNetworkStream.Read(baBuffer, 0, 64) > 0)
        {
            receiveSocketMsg(baBuffer);
        }
        Thread.Sleep(100);
    }
    /// close the data stream channel
    if (tcpNetworkStream != null)
    {
        tcpNetworkStream.Close();
    }
    /// close the socket connection
    if (tcpClient != null)
    {
        tcpClient.Close();
    }
}

/// <summary>
private void receiveSocketMsg(byte[] baBuffer)
///
/// Purpose: receiving a data buffer from the socket connection
/// Version: 13-NOV-20: NOS
/// </summary>
{
    string sBuffer;

    sBuffer = Encoding.ASCII.GetString(baBuffer);
    sBuffer = sBuffer.TrimEnd('\0');
    publishEventToUi((int)Constants.UiEventCmd.MsgRxList, sBuffer, 0);
    if (sBuffer.Equals(SOCKET_DISCONNECT_CMD) == true)
```

```
        {
            closeSocketConnections();
        }
    }
```

The souce code snippets for the writing method, using the either the server or client socket:

```
    /// <summary>
    public bool sendSocketMessage(string sSendMsg)
    ///
    /// Purpose: the message to send on an active socket connection
    /// Version: 13-NOV-20: NOS
    /// </summary>
    {
        bool bValid;
        byte[] baBuffer;

        switch (iActiveSocketConnection)
        {
            case (int)Constants.Socket.Server:
            case (int)Constants.Socket.Client:
                if (tcpNetworkStream != null)
                {
                    baBuffer = Encoding.ASCII.GetBytes(sSendMsg);
                    tcpNetworkStream.Write(baBuffer, 0, baBuffer.Length);
                    bValid = true;
                }
                else
                {
                    bValid = false;
                }
                break;
            case (int)Constants.Socket.None:
            default:
                bValid = false;
                break;
        }
        return bValid;
    }
```

21.4.7 Security

Both the reading and writing functions are just sending plain strings (text buffers) that can easy can be read by other applications, and these applications can also easy change the contents of the strings. Adding any type of security requires another protocol on top of TCP/IP as TCP/IP is only handling data buffers and the receivers must have some sort of protocol for decrypting the information. Applications like Wireshark can very easly get all the data buffers for all TCP/IP connections on your computer.

Looking at Figure 21.21 the Presentation layer is normally containing any data security function. One way to extend this application can be by crypting the strings before sending them and decrypt them when receiving them. One easy way of doing this can be to add a number to each character in the string, for example 2, meaning that the character A will be C, B will D and so one. The sender and receiver must both agree on the coding so the receiver will, in this case, subtract 2 from each of the characters. The string "Hello world" will then be transmitted as "Jgnnq yqtnf" as long as characters outside the range A to Z and a to z will not be crypted/decrypted.

Part VII

Fieldbus

Chapter 22

Introduction

Field buses will be an important part of a process system or a control system. A system can consist of one or several field buses, most used is several field buses. These field buses can be used for communication between the sensors, valves and actuators in a system, between the DCS's, and PLC's, interconnections for the SCADA system, or between all of these devices.

The field bus is a control network used in process control and industrial automation. Field buses are bidirectional, digital serial networks that offer services at layers 1 and 2 of the OSI model (physical and data link). Most field buses offer services at layer 7 as well, layer 7 is defined commands (message structure) for reading and/or writing information to and from the bus. A mixture of devices from several vendors on the same field bus require layer 7. Field bus is an industrial digital communications network intended to replace the existing $4 - 20$ mA analogue signal standard.

22.1 Field bus overview

In industrial applications there is a big interest for sharing of the devices in the physical process, several systems want to use the same devices for monitoring and control. One way of sharing these devices among several systems is the usage of a digital technology for communication between the sensor devices. This low level digital communication between the sensor devices is known as a field bus.

The OSI communication model describes an abstract and general layered communication model, but normally not all layers are needed in all applications. When connected to a common physical bus normally only layers #1 (physical layer) and #2 (data link layer) in the OSI model are needed together with an application layer (layer #7). See Figure 22.1.

Layer #1 defines the physical interconnection of the devices (wire or wireless, voltage levels, connectors, and so). Layer #2 defines that several devices can be connected to the same bus and how they can be addressed. Since layer #3 is not supported means that the devices can not use the bus as a network, but that one of the devices should some type of masters of the bus. The application layer, layer #7, will be defined for most field buses. Layer #7 defines a set of commands or message structures for exchanging information on the bus.

The main reasons for a field bus are:

1. to save the amount of cable. In a physical process there will be a lot of sensors and actuators

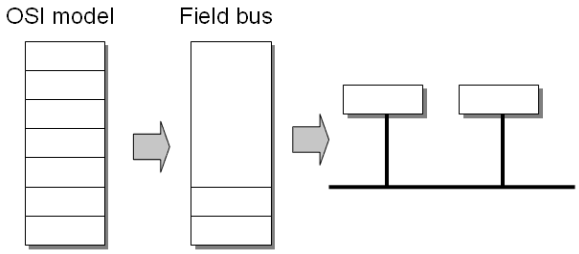


Figure 22.1: The number of communication layers for a field bus. Note that some field buses may also support layer #7.

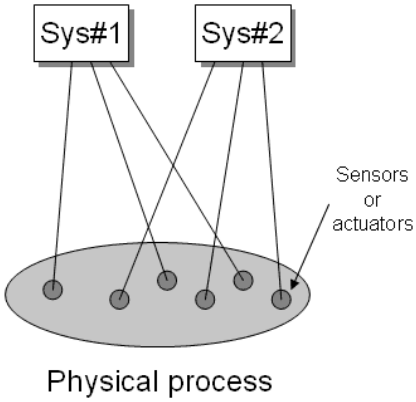


Figure 22.2: A set of sensors and actuators connected to computer systems using separate cables.

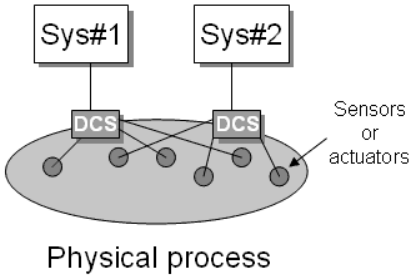


Figure 22.3: A set of sensors and actuators connected using a Distributed Computer System (DCS) as a concentrator.

connected to different systems using cables. The cable cost a lot of money, the space may be a problem, and the weight may be a problem. Figure 22.2 shows a standard system using separate cables for each sensor and actuator in the physical process,

2. exchange data between devices,

3. for an easy maintenance of the system. Using a field bus it is more easy to configure the system, regular maintaining of the devices and troubleshooting of the devices if any problems,

4. interoperability. Using devices from different suppliers can make the control system more flexible, easier to maintain, and extend the lifetime of the system.

A way of reducing the amount of cables is to use a Distributed Computer System (DCS) as a concentrator located in the physical process. Figure 22.3 shows the system from Figure 22.2 using a number of DCS instead. The DCS's are used as concentrators reducing the amount of cable, but what about the cost? The cost of the DCS will most probable be higher than the cable, depending on the number of sensors and/or actuators connected, but the maintenance of the system will be cheaper.

Still the systems have to use separate sensors and actuators, Figure 22.4 shows the system from Figure 22.2 using a field bus to reduce the amount of cable and also reducing the number of sensors/actuators in the system.

A SCADA system can contain several field buses depending on the size of the system. Figure 22.5 shows a SCADA system with a field level, a process level, a plant level and an enterprise level. The process level and the field level consists of field buses.

22.1.1 Field bus or Serial Communication

Digital industrial networks, field buses, device- and sensor- level networks (also USB) all support many vendors' products and can offer advanced features for configuration and diagnostics. Any need for serial communications like RS-232, RS-422, and RS-485? Some of the arguments for serial communication may be:

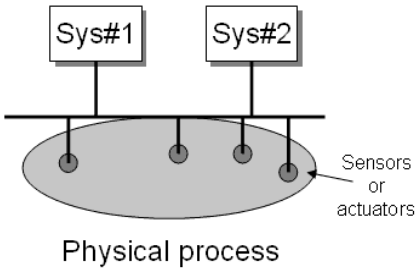


Figure 22.4: A set of sensors and actuators connected using a common field bus.

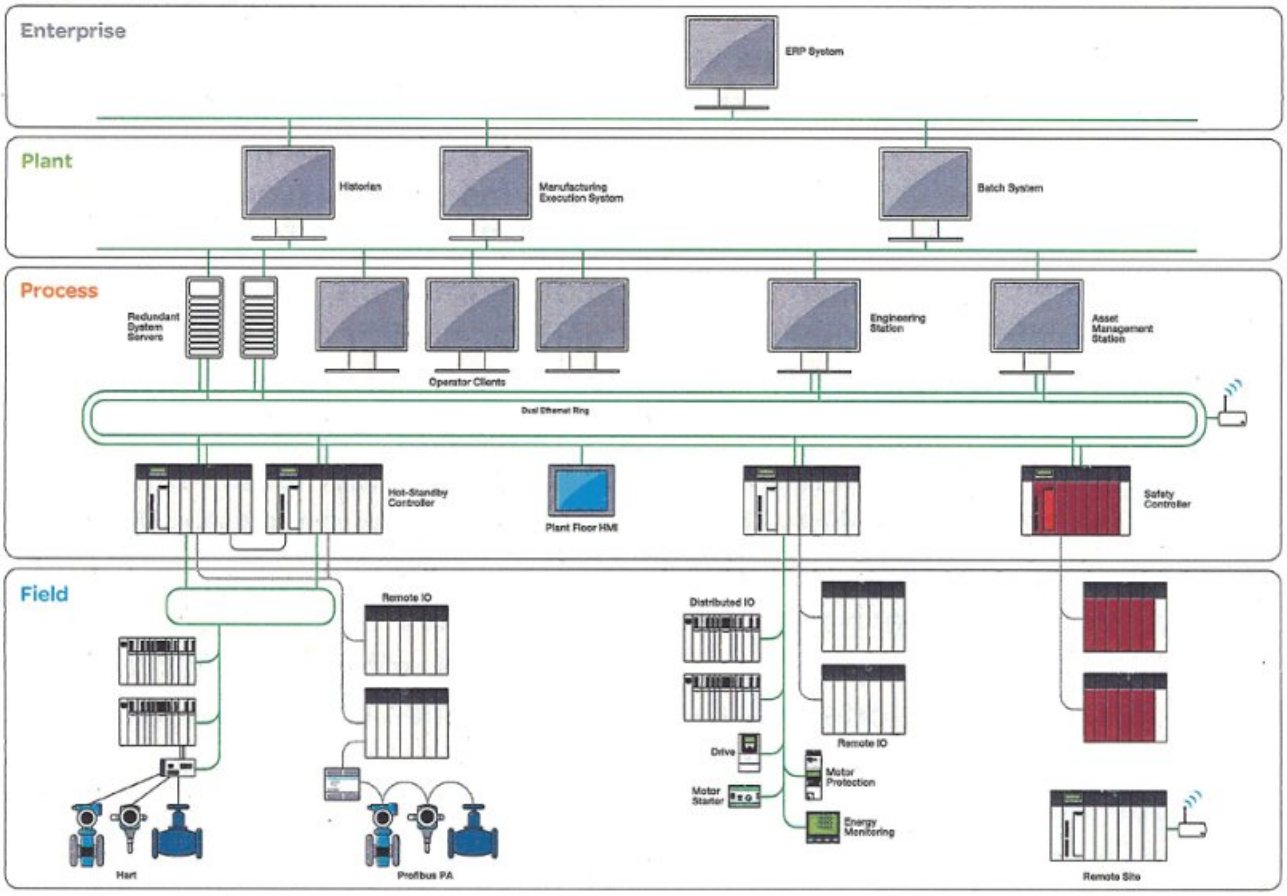


Figure 22.5: An example of different type fieldbuses used in a SCADA system (AMNYTT #3/2010).

1. low cost, simplicity, and familiarity. Industrial serial communications, largely RS-232, RS-422, and RS-485, are still included on many products and machines because they provide simple, bit-level, point-to-point communications, and they are widely understood.

2. an easy and inexpensive way to communicate with devices. Most vendors will offer a serial-only version as an entry-level product, offering an Ethernet-enabled version as a higher end offering;

3. if a product has been in production for many years and is proven, it could be cost- prohibitive for a device manufacturer to redesign and recertify a network version of the same product.

4. serial communication can be the least expensive and simple solution when your communication is only point to point, but with a limited set of information. Serial protocols are excellent for data transfer but do not conveniently enable increased information and control flexibility.

22.1.2 Description

The field bus is a digital, bidirectional, multidrop, serial-bus, communications network used to link isolated field devices, such as controllers, transducers[1], actuators and sensors together. Each field device has low cost computing power installed, making each device a "intelligent" device. Each device is able to execute simple functions on its own such as (www.wikipedia.org 2006):

1. diagnostic,

2. control,

3. maintenance functions,

4. providing bidirectional communication capabilities.

Field bus will replace centralized control networks with distributed-control networks. Therefore field bus is much more than a replacement for the 4–20 mA analogue standard. The field bus technology will:

1. improve quality; digital communication, local control, local maintenance local diagnostic, better failure information,

2. reduce costs; sharing communication cable, better assortment, several suppliers,

3. boost efficiency; digital communication, faster connection and configuration, better failure and calibration functions.

22.1.3 Interoperability

The interoperability may be both an advantage and a disadvantage. Devices from several suppliers can be interconnected, but they need to be tested for the same response to all control signals. Some field buses have compatibility tests or certifications, but then the prices will be higher.

22.1.4 Hot plugging

Hot plugging means that devices can be connected and/or disconnected when the communication system is activated. No restart of the master or the other nodes is necessary, and the internal tables should be updated as well.

22.1.5 Environment

A bus or a network in an industrial environment will always be affected by other factors than an office network. Any buses or networks to be used in a plant have to take care of these factors during the design phase. Some of these environmental challenges are(Caro 2004):

[1] Transducer is sometimes defined as any device that converts a signal from one form to another (www.wikipedia.org 2006).

Factor	Effect	Solutions
Heat	expansion/contraction of connectors	pin and socket connectors better connector materials
Vibration	intermittent connections	pin and socket connectors
Corrosion	oxides at contact points	sealed connectors
Electrical noise	common mode induce false signals	high common mode noise rejection
Magnetic field	induce current	shielding, isolation, fiber optics

22.2 Advantages and disadvantages

The field bus has several advantages from which end users can benefit, but also some disadvantages.

22.2.1 Advantages using a fieldbus

Cost savings;

Several cost savings issues when using a field bus:

1) Cable elimination as multiple devices share the same cable for communication, 2) Speedier commissioning, 3) More easy upgrade of existing equipment, 4) more easy for future extensions, 5) Maintenance (remote error, calibration, and configuration), 6) Larger assortment of devices due to several suppliers and 7) Lower system cost.

Performance;

Several way for increasing the performance when using a field bus:

1) More distributed monitoring and control; control programs can now reside in the individual field devices, reducing the overall size of the main control system, 2) Simplification of the collection of information from field devices giving that measurement and device values will be available to all field and control devices, 3) Two-way communication between field devices and the control system as two field devices can communicate directly rather than via the control system. This also enables several related field devices to be combined into one device, 4) Setup and calibration of the devices, 5) Less influenced by noise and 6) Higher degree of reliability; better check with warnings and failures.

Vendor independent:

It is more easy to interconnect devices from different vendors when using a field bus. Field buses should include the application layer (layer 7) if the OSI model giving that interconnection is more easy.

22.2.2 Disadvantages using a fieldbus

A field bus has some disadvantages like 1) a more complex system; need higher qualification to maintain the field bus, 2) higher price of devices as more "intelligence" is build into them, 3) testing devices may be more complex, 4) depending on the system, slightly longer reaction times and 5) due to the number of different field buses the sensor/actor manufacturers are forced to offer several of them, which adds cost.

22.3 Types of Field buses

Industrial communication is divided into two types of automation and in different levels. The type of automation are:

1. process automation; continuous processes with mainly analog signals and control loops,
2. factory automation; discrete production processes mainly with digital signals and sequential control.

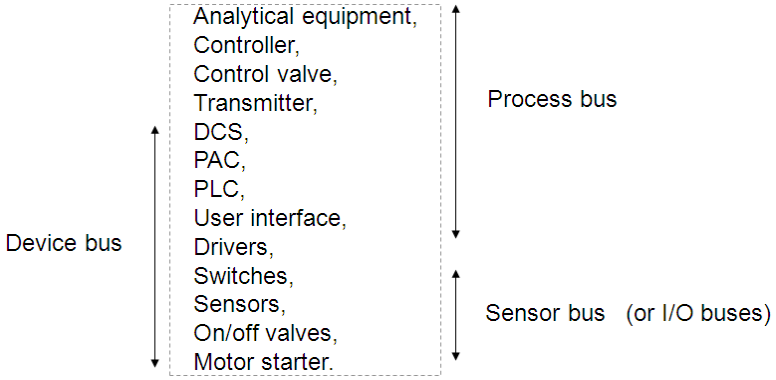


Figure 22.6: The type of devices connected to the sensor bus, device bus, and field bus (from M.Pedersen/TUC: jan-09).

The levels are automation, production, and administration. This document will focus on the field buses being at the automation level. The field buses must be digital serial communication buses, and these buses can be divided into the following categories:

1. Sensor buses or I/O buses; contains only I/O devices (sensors and actuators) and transfer small amount of data (binary data),

2. Device buses; contains mainly digital I/O and frequency controllers/starters. These buses are normally used for factory automation,

3. Process buses (field buses and industrial control buses); contains all sort of devices for monitoring and control a process and transfer great amount of data. The buses are used for process automation, often meaning control devices in addition to the fabric automation. Process automation may also include EEx requirements, safe operation of electric equipment in explosive atmospheres,

4. Safety buses; a reliable and deterministic bus that often will be a sub bus of the device bus or the process bus. The important function of the safety bus is always to detect a failure of the bus or a node on the bus.

The type of devices connected to the different types digital communication buses are shown in Figure 22.6. The amount of data between the devices is describing the type of bus. The sensor bus is using information on bit level, the device bus is using information on byte level, and the process bus is using information on message (block or several bytes) level.

Figure 22.7 shows the relationship between the message size between the devices, the type of information, and the type of application.

22.4 Field bus selection

Several field buses exist with different network topologies and architectures, but how to define the right field bus for a new project? Important choices when selecting a field bus:

1. information level; issues for selecting the information level in the network like 1) the type of devices used in the network and the information between these devices (computers, PLC, email, message based, file based ..), 2) focus on the size of data messages and communication speed, 3) security level and 4) openness; interoperability with different types of devices, vendors, and levels.

2. control level; issues that may affect the control level like 1) using devices like PLC, actuators and DCS for control, 2) focus on high communication speed and smaller data messages and 3) cycle times.

3. device level; issues for selecting the device level are 1) using different type of devices depending on the “intelligence” build into these devices and 2) focus on cost and small data messages.

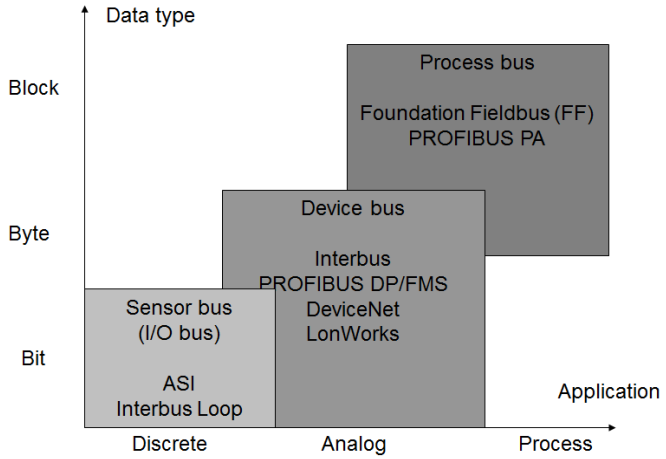


Figure 22.7: The relationship between the amount of data in digital serial communication buses, the type of buses, and the type of application (from M.Pedersen/TUC: jan-09).

The selection of a field bus should be based on technical, practical, and cost decisions like: 1) data messages like size of messages (header and data) and transmission speed, 2) cable type, number of manufactures, and number of devices available, 3) the number of devices connected to the network, 4) the length of the network, 5) the cost and 6) the configuration and maintaining of the network.

22.4.1 Smart Factory

Several project has been started to make a factory smarter. The goal is to make the factory more intelligent, characterized by integration with the process, customers and business partners. The process integration is based on adaptability, resource efficiency and ergonomics. One concept is the "*Industry* 4.0" where integration of computers, network and the physical process is defined as the Cyber Physical System (CPS). The concept is based on sensors to collect information, software to exchange information (communication) and perform multivariate analysis to "optimize" any results. The system will be based on adaptive sub systems and Internet of Things (IoT), so the network will be internet. The industry has even defined the Industrial Internet of Things (IIoT) where the process bus will be internet. Systems of today are based on different levels of information and control (company level, operational control level, process control level, and field level), while the "Industry 4.0" will be based on independent devices performing an "optimized" operation based on information from the system. China and USA has their own concept named "Made in China 2025" and "Industrial Internet".

Chapter 23

Sensor buses (I/O buses)

This section will only list some of the I/O buses available today. I/O bus, or sensor bus, offer few capabilities and are the lowest grade of existing field buses.

23.1 AS-Interface

Actuator Sensor Interface[1] (ASI) is an I/O bus used for networking of sensors and actuators. The ASI is an open standard based on the standards IEC 62026 – 2 and EN 50295. ASI specifications:

- ASI has a two wire interface including both power and data. The unshielded 2-wire Yellow cable is unterminated and ungrounded,
- The network is master/slave network with a single master,
  - With maximum 4 outputs on each slave up to 31 slaves can be interconnected, giving maximum 32 devices in such an ASI network,
  - With maximum 3 outputs on each slave up to 62 slaves can be interconnected, giving maximum 63 devices in such an ASI network,
  - Slaves can have analog inputs of up to 16 bits resolution,
- Two type of slaves can be used, either a slave with 4 inputs and 4 outputs or a slave with 4 inputs and only 3 outputs to allow addressing of more slave nodes,
- Automatic addressing of slave nodes,
- The Topology may be either: 1) Bus, 2) Ring, 3) Tree, and 4) Star,
- The maximum length of the bus can be up to 100 meters without a repeater and 300 meters with repeaters,
- Power is provided by a 29.5V to 31.6V floating DC supply, which can supply at least 8 A over the network,
- The master is controlling the communication in the network and the response time from each node is maximum 150 μs. This means that the master will be updated with information from all slaves within maximum 4.7 milliseconds,
- The baud rate is 167 Kilobits/second and the message size is 8 bits,
- Includes safety solutions where a master will be the safety monitor.

ASI is the simplest of the industrial networking protocols used in PLC, DCS and PC-based automation systems. It is designed for connecting binary (ON/OFF) devices such as actuators and sensors in discrete manufacturing and process applications using a single cable. Safety devices can also be connected directly to the bus as shown in Figure 23.1.

[1] AS-Interface: www.as-interface.com (MAR-07)

Figure 23.1: ASI network with a master and a set of sensor nodes as switches and measurements, including safety devices. Notice the yellow cable containing both the network signal and the power. (www.Pepperl-Fuchs.us: des-13).

ASI is a highly efficient networking alternative to the hard wiring of field devices. It is an excellent partner for field bus networks such as PROFIBUS, DeviceNet, Interbus and Industrial Ethernet, for whom it offers a low-cost remote I/O solution. It is proven in hundreds of thousands of applications, including conveyors, process control valves, bottling plants, electrical distribution systems, elevators, and food production lines to mention some.

23.2 CAN-bus

CAN (Controller Area Network) bus can be both an I/O bus and a process bus and will be described in the process bus section.

23.3 EIB

European Installation Bus is an I/O bus transmitting data up to 9600 bits/sec using RS-485 for building and home automation. The physical layers can be power lines, radio frequency (RF) and infrared.

23.4 Interbus

Interbus is a ring based I/O bus based on the RS-422 electrical interface. The network consists of one master and a set of slaves communicating at a baud rate of 500 Kbps (Maximum 2 Mbps) at full duplex. The maximum length between 2 devices is 400 m and the total length of the network is limited to 13 km. This gives a maximum number of 32 devices in the network. Each slave connects to the bus using 2 full-duplex RS-422 transceivers, one on each side of the slave. The topology can be Tree, Ring, Star or point to point bus

23.5 I^2C

Inter IC bus can be used as a sensor bus, but is used mainly inside computer systems. The I^2C bus uses a bidirectional Serial Clock Line [SCL] and Serial Data Lines [SDA] and 3 speed modes are specified:

1. Standard; 100kbps [Bits per Second],
2. Fast mode; 400kbps,
3. High speed mode 3.4Mbps.

I^2C, due to its two-wire nature (one clock, one data) can only communicate half-duplex. The maximum length is about 10 meters.

23.6 Modbus

Modbus can be both an I/O bus and a process bus and will be described in the process bus section.

23.7 Hart

The Highway Addressable Remote Transducer (Hart) Protocol is an early implementation of Field bus. Hart is using a digital communication on the standard $4 - 20\,\mathrm{mA}$ analog communication, where the digital information is added to the analog current signal. Due to the huge installed base of 4-20 mA systems throughout the world, the HART Protocol is used a lot in the industry.

There are two main operational modes of HART instruments: analog/digital mode, and multidrop mode. In analog/digital mode the communication is point to point giving only one master and one slave. In multidrop mode the number of devices can be up to 15.

23.8 IO-Link

Point to point communication to sensors and actuators, a digital communication using 3 wires (power, signal, ground) to each device. The communication speed is 187 kBaud and the packet size is up to 32 bytes. Includes configuration and diagnostics. Each device is delivered with a configuration file with all parameters included.

Chapter 24

Process buses

Process buses can be divided into control buses and field buses where the control buses are the most complex buses. This section will try to list the most used process buses including some technical description for each of the process buses.

24.1 PROFIBUS

PROFIBUS (Process Field Bus) was defined in 1991/1993 as DIN[1] 19245 standard, moved in 1996 to EN 50170 standard, and is since 1999 included in IEC 61158/IEC 61784 Standards. IEC 61158 is also called the Octobus[2] standard.

PROFIBUS is the most popular type of field bus with more than 14 million nodes in use worldwide (www.wikipedia.org 2006) in 2006 and over 22 million nodes in 2009. PROFIBUS was developed in 1989 as the result of a German research project involving 15 firms and research institutions, and dominates in Europe with more than 60% of the factory automation market (www.wikipedia.org 2006). The PROFIBUS standard is maintained, updated and marketed by PROFIBUS International[3], a non-profit organization administered from Karlsruhe in Germany. PROFIBUS International is also responsible for the development of PROFINET, an Ethernet based field network.

The PROFIBUS Industrial Bus Protocol Specifications consist of:

1. PROFIBUS-FMS (Field bus Message Specification or Field bus Management System) is a control bus generally used for communications mainly between PROFIBUS masters. A complex protocol specification,
2. PROFIBUS-DP (Decentralized Peripherals) is a device level bus that supports both analog and discrete signals,
3. PROFIBUS-PA (Process Automation) is used for process level instrumentation,
4. ProfiDrive is used for motor control.

The PROFIBUS is able to transmit a large amount of data at a high speed. PROFIBUS can be used on large installation, but remember that the protocol is using a large header of each data message giving much overhead if transmitting small amount of data.

24.1.1 PROFIBUS FMS

PROFIBUS can be from a simple master/slave I/O network up to a complex multi-master token ring network. The Field bus Message Specification is a protocol used for communication between masters, between masters and slaves, and to communicate directly between the slaves without the usage of any master. Today more and more intelligence is build into the devices given the possibility to communicate directly with other devices without the usage of the FMS specification. The PROFIBUS-DP is the result of the FMS specification and more intelligence build into the I/O devices.

[1] DIN: Deutsches Institut für Normung
[2] The standard describes 8 different field buses, hence the name.
[3] www.profibus.org (mar-07).

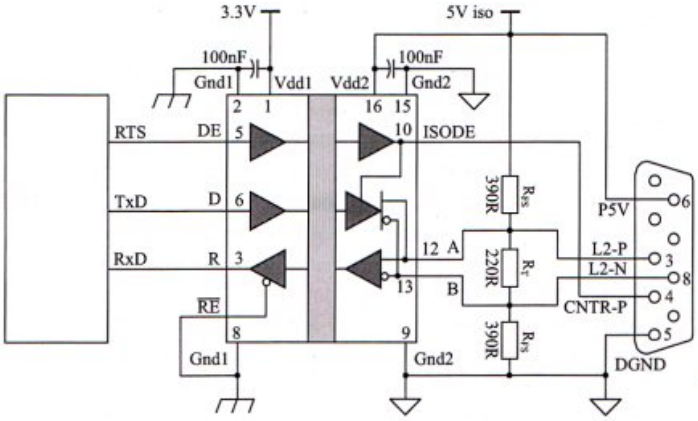


Figure 24.1: The RS-485 interface of a Profibus node with a insolation circuit and the termination resistors R_{FS} and R_T. The node should be isolated from the RS-485 bus. A 9 pin connector is used for connection to the RS-485 bus (www.IsoLoop.com, oct-09).

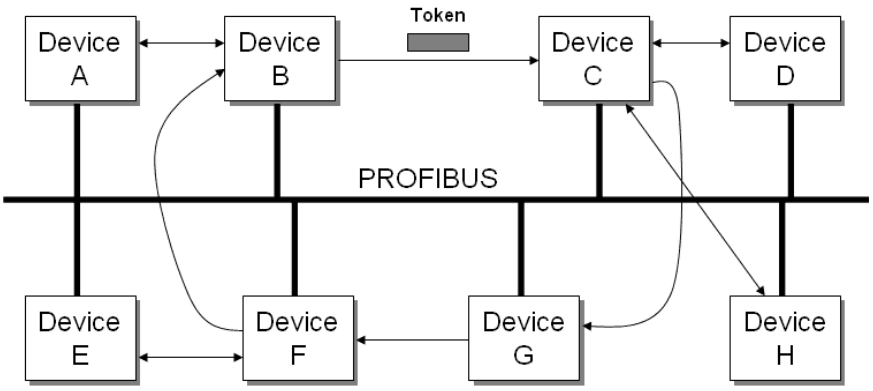


Figure 24.2: The PROFIBUS operating principle. The master devices B, C, F, and G is communicating using a token, while the slave devices A, D, E, and H are answering requests from the master device (Olsson & Piani 1998).

24.1.2 PROFIBUS DP

PROFIBUS-DP (Decentralized Peripherals) is the network for distributed (remote) I/O devices and is using RS-485 or fibre as the physical layer. A PROFIBUS DP node with RS-485 interface is shown in Figure 24.1, with the control signals (RTS, RxD, and TxD), an isolation circuit, and termination resistors (R_{FS} and R_T).

The PROFIBUS-DP protocol consists of 3 types:

1. DP-V0: a master/slave protocol, cyclic polling of the I/O devices. A fast and simple protocol,
2. DP-V1: an extension of the V0 protocol using multiple masters and token ring communication.
3. DP-V2: an extension with the possibility of slave to slave communication. The communication structure will be based on broadcasting.

An example of the DP-V1 protocol is shown in Figure 24.2 where devices B, C, F, and G are the master devices using the token and communicating with the slaves.

The communication of the DP network is from 9600 kbit/sec up to 12Mbit/sec, the maximum communication speed depends on the length of the network. The relationship between the maximum communication speed and the length is:

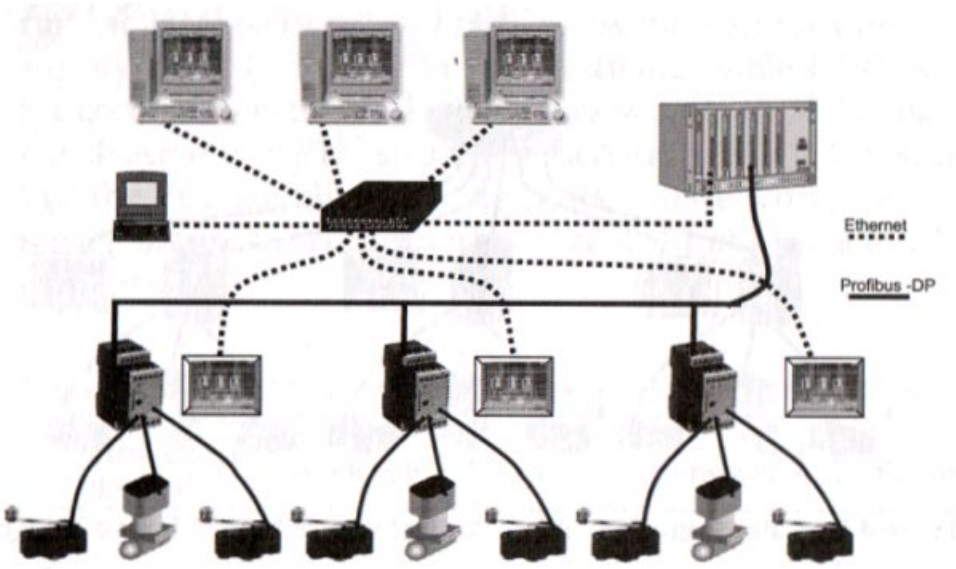


Figure 24.3: A plant measurement and control with Profibus-DP, and standard Ethernet for display (Caro 2004).

Length of segment $[m]$	Communication speed $[kbit/\,\text{sec}]$
1200	$9.6, 19.2, 45.45,\ or\ 93.75$
1000	187.5
400	500
200	1500
100	$3000, 6000,\ or\ 12000$

The number of devices on a DP network is 32 DP devices without repeaters and 126 DP devices with repeaters. The maximum number of repeaters are 9. Using 9 repeaters will the maximum length of the network be 9.6 km.

The communication speed has to be selected when the whole network is configured and/or connected.

The DP network can be configured as a bus, tree, ring or star.

PROFIBUS-DP can also be used in hazardous area (explosive) using the RS485-IS protocol, an extension of the RS-485 protocol. IS is short for Intrinsically Safe. Using the RS485-IS protocol the maximum speed will be 1500 kbit/sec.

Figure 24.3 shows a plant measurement and control system where the PROFIBUS-DB is used for measurement and control, and a standard LAN based on Ethernet is used for user operations.

24.1.3 PROFIBUS PA

PROFIBUS-PA is a full-function field bus that is generally used for process level instrumentation. The PROFIBUS-PA must always be part of a DP bus, can not be a independent bus. The maximum number of PA devices are 32 for each PA segment, and the maximum number of PA segments connected to a DP segment is 125. PROFIBUS-PA communicates at 31.25 kbit/sec and has a maximum distance of 1.900 meters per segment. Figure 24.4 shows the interconnection of a DP and PA buses.

The protocol DP-V2 is not supported in PROFIBUS PA. PROFIBUS PA can also be used in explosive areas.

24.1.4 OSI model

The PROFIBUS protocol with reference to the OSI reference model is shown in Figure 24.5.

The PROFIBUS protocol consists of the following layers (www.wikipedia.org 2006):

1. Application layer with 3 different methods as;

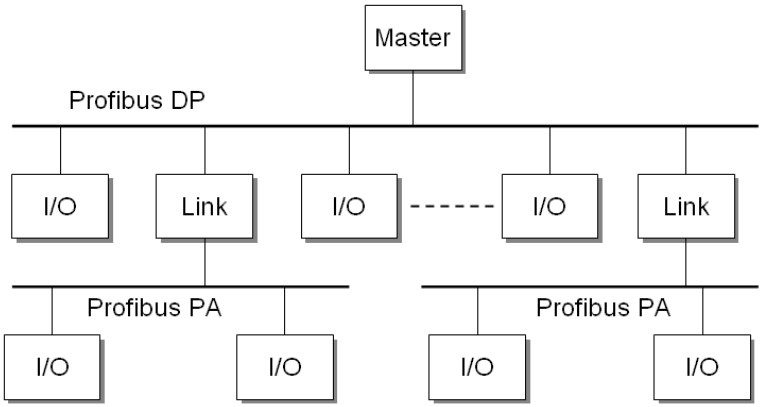


Figure 24.4: Connection of DP devices and PA devices in a PROFIBUS network.

PROFIBUS Protocol (OSI reference model)

OSI-Layer		PROFIBUS			
7	Application	DPV0	DPV1	DPV2	Management
6	Presentation	--			
5	Session				
4	Transport				
3	Network				
2	Data Link	FDL			
1	Physical	EIA-485	Optical	MBP	

Figure 24.5: The PROFIBUS protocol with references to the OSI reference model (www.wikipedia.org 2006).

(a) DP-V0 for cyclic exchange of data and diagnosis,

(b) DP-V1 for acyclic and cyclic data exchange and alarm handling,

(c) DP-V2 for isochronous mode and data exchange broadcast (slave-to-slave communication). Isochronous mode means time-dependent, and refers to processes where data must be delivered within certain time constraints.

2. Data Link layer where the security functions are implemented. The security layer FDL (Field bus Data Link) works with a hybrid access method that combines token passing with a master-slave method. In a PROFIBUS DP network, the controllers or process control systems are the masters and the sensors and actuators are the slaves.

3. Physical layer with 3 different methods for data bit transmission as:

   (a) RS-485 (EIA-485); twisted pair cables with wave impedances of 150 ohms are used in a bus topology. Bit rates from 9.6 kbit/s to 12 Mbit/s can be used. The cable length between two repeaters is limited to 100 to 1200 m, depending on the bit rate used. This transmission method is primarily used with PROFIBUS DP,

   (b) Optical; fiber optics, star-, bus- and ring-topologies are used. The distance between the repeaters can be up to 15 km. The ring topology can also be executed redundantly,

   (c) MBP (Manchester Bus Powered) transmission technology, data and field bus power are fed through the same cable. The power can be reduced in such a way that use in explosion-hazardous environments is possible. The bus topology can be up to 1900 m long and permits branching to field devices (max. 60 m branches). The bit rate here is a fixed 31.25 kbit/s. This technology was specially established for use in process automation for PROFIBUS PA .

24.2 Field bus Foundation

The IEC/ISA SP50 Field bus committee[4] was formed by the Instrument Society of America (ISA), the International Electrotechnical Commission (IEC), PROFIBUS (German national standard) and FIP (French national standard).

Fieldbus Foundation (FF) is an all-digital, serial, two-way communications system that serves as the base-level network in a plant or factory automation environment. Developed and administered by the Fieldbus Foundation.

In a FF network a Link Active Scheduler (LAS) is controlling the communication by polling the control devices. The LAS is the master in the network sending requests to the control devices in the network, and the control devices will answer the request with a broadcast message that all devices in the network will be able to read. The function of the LAS will be just to inform when the different slaves are able to share the information on the network. The LAS will not receive any information for sending to the different slaves. The different devices must be configured to what kind of data is should use from other devices in the network. A LAS can communicate with maximum 240 devices. Figure 24.6 shows a FF network using a LAS for communication between the devices connected to the network.

24.2.1 FF implementation

Two related implementations of FF have been introduced to meet different needs within the process automation environment. These two implementations use different physical media and communication speeds. These implementation are:

1. H1 works at 31.25 kbit/s and generally connects to field devices. It provides communication and power over standard twisted-pair wiring. H1 is currently the most common implementation. Using LAS as the master device in the network. The number of devices are 32 in each segment and the total length of each segment is 120 m. The total length of the network is 1900 m. The network can be used as bus, star or tree topology.

[4] Fieldbus Foundation: www.fieldbus.org (MAR-07)

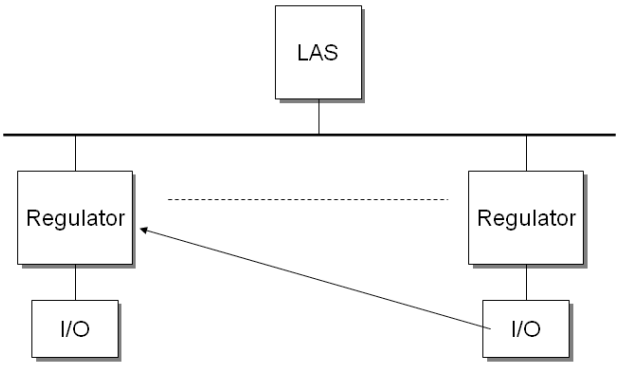


Figure 24.6: Link Active Scheduler (LAS) in a Fieldbus Foundation network controlling the communication between the devices.

2. HSE (High-speed Ethernet) works at 100 Mbit/s and generally connects input/output subsystems, host systems, linking devices, gateways, and field devices using standard Ethernet cabling. The communication is using the Ethernet protocol without the need for a LAS being a peer to peer communication, not master/slave communication. Peer to peer means that any node can be a master or a slave (Ethernet protocol)(P2P). A peer is a software application running on a node being both supplier and consumer of resources. The network can only be a star topology, using one or more hubs.

24.2.2 FF Safety

Fieldbus Foundation Safety Instrumented Functions (FF-SIF) is an automation network for the hazardous area where safety systems and emergency shutdown can be incorporated into the same fieldbus.

24.2.3 FF in OSI model

Fieldbus Foundation is defining layer #1, #2, and #7 with reference to the OSI model, and an additional user layer. This is shown in Figure 24.7.

The SP50 committee decided to concentrate on four layers for the fieldbus solution:

1. Physical Layer: This defines the media that communication occurs over and could be viewed as the 4–20 mA standard replacement (1992). Using the same physical layer as PROFIBUS PA.

2. Data Link Layer: This monitors the communications taking place among the various devices and detects errors (1996).

3. Application Layer: This formats the data into messages which all devices connected to the network can understand and provides the services for process control, supplying them to the user layer (1996).

4. User Layer: This connects the individual plant areas and provides an environment for applications. It is implemented using high level control functions (1996).

24.3 PROFINET

The PROFINET architecture is defined by the PROFIBUS user organization and defines a communication model for vendor independent engineering for plant-wide automation. PROFINET is an Industrial Internet, starting the standardization in 1983. It started as communications between master control room and controllers, today addressing the total field level. Note that PROFINET is not PROFIBUS over Ethernet, PROFINET is using a different protocol but with elements from the PROFIBUS protocol.

The communication is based on DCOM using TCP/IP and Ethernet using mostly standard Ethernet components like hub, switches, and bridges. PROFINET also supports the DHCP (Dynamic Host Configuration Protocol). PROFINET has also real-time support on Ethernet. Its component-based

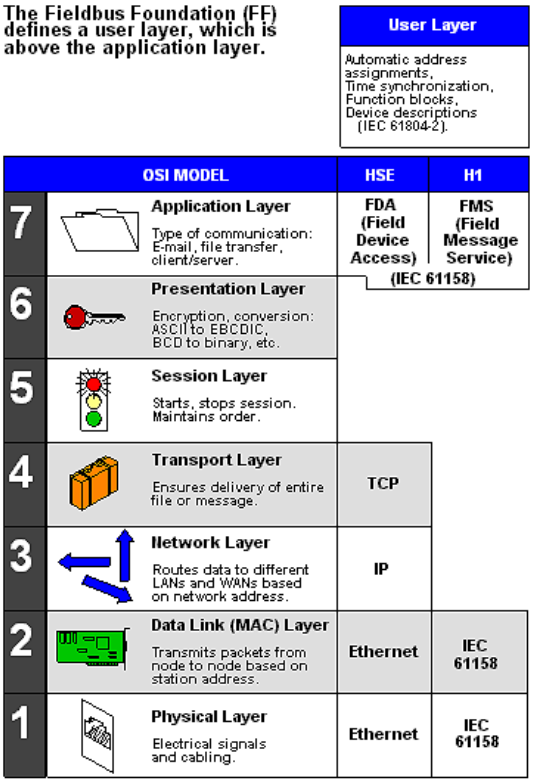


Figure 24.7: The Fieldbus Foundation and the OSI model (Computer Desktop Encyclopedia: 2004).

approach goes beyond OPC's tag-based approach. Functionally PROFINET extends the configuration and diagnostic features of PROFIBUS, employing established IT standards like XML. The component based structure gives a modular structure where the designer can select the functions.

The PROFINET standard consists of two different modes, these are (www.wikipedia.org 2006):

1. PROFINET CBA; PROFINET CBA is suitable for the component-based communication via TCP/IP and the real-time communication for real-time requirements in modular systems engineering. Both communication options can be used in parallel.

2. PROFINET IO; PROFINET IO was developed for real time (RT) and isochronous real time IRT (Isochronous Real Time) communication with the decentralize peripherals.

PROFINET CBA and PROFINET IO can communicate at the same time on the same bus system. They can be operated separately or combined so that a PROFINET IO subsystem appears as a PROFINET CBA system from a system perspective.

The communication model of PROFINET with reference to the OSI model is shown in Figure 24.8 and 24.9. Figure 24.9 also shows how the real-time extension in the PROFINET IO is implemented with reference to the OSI model.

24.4 Controlnet

ControlNet[5] is developed by Allen Bradley, a large manufacturer of PLCs. The communication is using Concurrent Time Domain Multiple Access (CTDMA) where each node will have a network interval where the node can use the network. This network interval will be configured as a Network Update Time (NUT) on each device.

Each node can transmit maximum 512 bytes in each interval and the data can be both configuration data and measurement data (real-time). The measurement data will normally have the highest priority.

[5] ControlNet: www.controlnet.org (MAR-07).

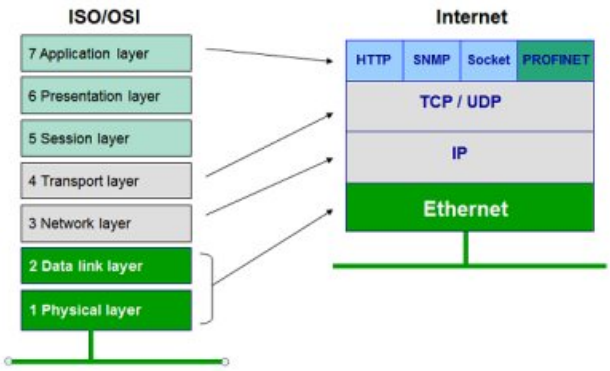


Figure 24.8: PROFINET protocol with reference to Internet and the OSI model. PROFINET is based on the TCP/IP protocol used for Internet.

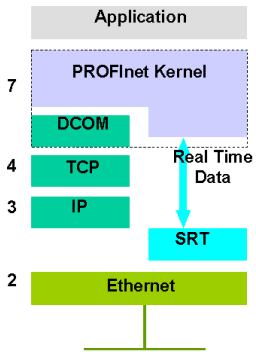


Figure 24.9: The OSI model for the PROFINET with the real-time extension.

Several masters can be used in the network and peer to peer communication can also be used. Multicast messages is used when several devices want data from one device and for synchronization in the network.

The network is using a standard RG-6 coax cable and BNC connectors. The communication speed is 5 Mbit/sec and the maximum length of a segment can be 1000 m. The maximum length depends on the number of devices connected to the segment, with 2 devices the maximum length is 1000m while with 48 devices the maximum length will be 250 m. Repeaters can be used, but the maximum number of devices in the whole network will 99.

Controlnet can also be used as a redundant system using two different networks, named as network A and network B. Using the redundant network devices can be connected to different networks or the devices must have network connectors. The network B is optional so the devices can use either Network A or both networks. Figure 24.10 shows two ControlNet devices connected to the networks.

Controlnet is using Ethernet bus connection.

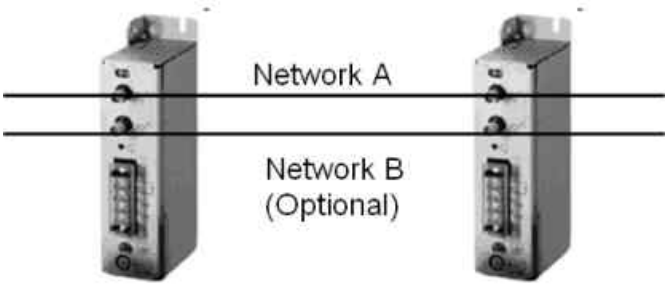


Figure 24.10: Two ControlNet devices connected to the redundant network, network A and network B.

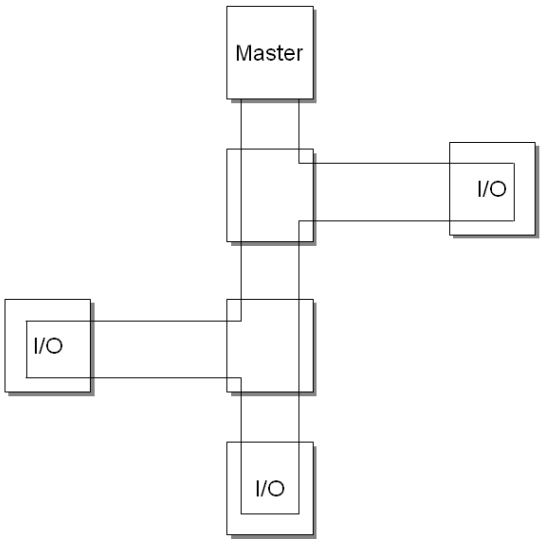


Figure 24.11: The Interbus devices and interconnection.

24.5 LON works

Lon Works (Local Operating Network) is a networking platform where the local network should be independent (distributed control) of the rest of the control system. A hardware chip, called the "neuron chip", was developed containing all the 7 layers from the ISO model. This chip is the only "connection" between the bus and the devices using the bus.

The physical layer can be cable, power line, wireless or fiber optic. The network consists of a set of subnets with maximum 127 devices in each sub net, maximum 255 sub nets, giving a maximum of 32000 devices in the network. The communication is peer to peer.

The communication speed depends on the topology and the maximum length between the devices as:

Physical layer	Length $[m]$	Speed $[kbit/s]$
Cable	2000	78
Cable	130	1250
Wireless	15000	15
Free topology	500	10

24.6 Interbus

The INTERBUS system was developed by Phoenix Contact and has been available since 1987. The INTERBUS is said to be a "big" shift register where all the bits are shifted out on the bus to the slave devices.

The INTERBUS is using the ring topology with a master controlling the communication. Each device has at least two connections, one connection transmitting the data and one connection receiving the data. Each device will then receive the data on one connection, amplify, and transmit on the other connection. Figure 24.11 shows an Interbus.

The number of devices in the bus is 512 using 16 sub levels in the network. The devices do not have a fixed address in the network, but will get a dynamically address depending on the location in the network.

The physical layer is RS-485 (EIA-485) and based on a ring network the speed is 500 kbit/s (maximum speed is 2 Mbit/s) at full duplex. The maximum bus length between two slaves are 400m and the maximum bus length is 13 km. Each slave connects to the bus via two full-duplex RS-422 transceivers, one on each side of the slave.

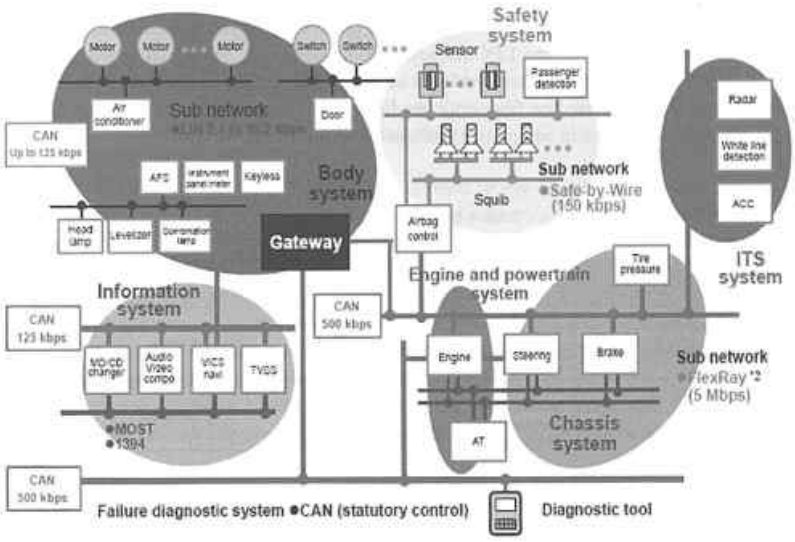


Figure 24.12: An overview of a CAN network in a car with different systems using different communication speeds, and interconnected by gateways (Ibrahim 2011).

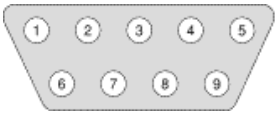


Figure 24.13: The 9 pins CAN bus connector.

24.7 CAN

Controller Area Network (CAN) is a broadcast, differential serial bus standard, originally developed in the 1980s by Robert Bosch GmbH, for connecting electronic control units (ECUs). The CAN bus was specifically designed to be robust in electromagnetically noisy environments and can utilize a differential balanced line like RS-485 (Ibrahim 2011). It can be even more robust against noise if twisted pair cable is used. Although initially created for automotive purposes (as a vehicle bus), nowadays it is used in many embedded control applications (e.g., industrial) that may be subject to noise.

The main purpose is automotive purposes and the main focus is cars. The CAN bus was first introduced in the BMW 850 in 1986 saving more then 2 km of cabling in the car. The number of connectors was significantly reduced, and the estimated weight savings for the cables and connectors was 50 kg. A modern car contains more than 50 microcontrollers that need to exchange information from a single byte and up to large data packets, and from low speed (5 $kbit/$s) to faster speend (≥ 1 $Mbit/$s). Unfortunately, all large car manufactors developed their own bus concepts and try to get their system adopted as an international system. Four of these systems survived, the CAN, VAN, J1850SCP and J1850DLC. Today CAN is the world leader and is used in most cards and vans. However, maintance system often support all the four protocols. Figure 24.12 shows an example of usage of CAN networks in a car.

The CAN bus protocol is using a 9 pins connector for connection to the bus, see Figure 24.13, using the following pins:

9 pin male D-SUB	Name	Description
1		Future use
2	CAN_L	Dominant low
3	CAN_GND	Ground
4		Future use
5	CAN_SHLD	Shieled, optional
6	GND	Ground, optional
7	CAN_H	Dominant High
8		Future use[6]
9	CAN_V+	Power, optional

The messages it sends are small (8 data bytes max) but are protected by a CRC-15 (polynomial 0x62CC) that guarantees a Hamming bit length of 6 (so up to 5 corrupted bits in a row will be detected by any node on the bus). The communication is based on broadcasting, a device is sending a request and another device is answering using broadcast meaning that all devices in the network can read the answer. The request is a command, defines as a *Message Identifier*.

When the bus is unused, any device can use the bus, but the device with the highest priority can use the bus first. Bit rates up to 1 Mbit/s are possible at network lengths below 40 m. Decreasing the bit rate allows longer network distances like:

Length $[m]$	Speed $[kbit/s]$
40	1024
100	500
200	250
500	125
6000	10

Only practical limitation on the number of devices connected to the bus and no configuration of device addresses. The address is assigned when the device is connected to the bus. This means that there will be no changes of hardware or software when a device is added or removed.

The bus is based on a two wire bus where the information is sent as a voltage difference, very much like the RS-485 standard. The signals are named CAN_L and CAN_H, and voltage difference is:

$$U_D = CAN_H - CAN_L$$

The voltages of the CAN bus is:

Voltage	recessive	dominant
CAN_H	$2.5V$	$3.5V$
CAN_L	$2.5V$	$1.5V$
$U_D = CAN_H - CAN_L$	$0 - 0.5V$	$0.9V - 2.0V$

and like the RS-485 the CAN bus must be terminated at each end with a resistor of $124\,\Omega$.

The CAN bus protocol is implemented in a set of micro controllers making it easy to make CAN bus devices. The CAN bus protocol defines the physical layer and the data link layer, layer 1 and 2 in the OSI reference model, all the other protocol layers are left to the network designer's choice. Protocols like CanOPEN, SDS and DeviceNet are protocols that can be used for layer 3 to 7 in the OSI model.

The CAN bus is multimaster protocol where any device can start sending a message when the bus is free. A collision will be detected if several devices attempt to use the bus at the same time. The devices has no addresses nor IDs meaning that there is no node to node communication, instead broadcast is used and it is up to each node to use the information transmitted on the bus. In stead of device addreses, message identifiers are used for the type of information. These message identifiers are also used for priorities, the message with the lowest message identifier has the highest priority (Ibrahim 2011). The communication speed is not fixed, the communication speed will be decided based on the devices attached to the bus.

[6] Pin 8: Some systems may use this pin as an error line, to indicate an error on the net.

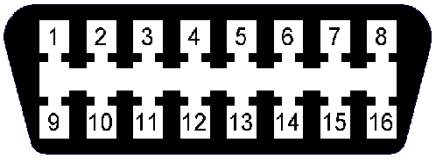


Figure 24.14: The OBD-II connector.

The CAN bus also offers remote transmit request (RTR) meaning that a node is able to request data from another node. A CAN system can then be configured to let a node broadcast any values at fixed time intervals or only upon request. This request may reduce the bus trafic and increase the performance while maintaining the integrity (Ibrahim 2011).

The CAN protocol consists of four message frames:

1. Data frame; data transfer between nodes. The length of the data filed is 0 to 8 bytes,
2. Remote frame; request data from a remote node,
3. Error frame; a message from any node signaling an error condition,
4. Overload frame; a messaeg used for a node to indicate that it is not yet ready to receive frames.

The CAN protocols consists of two types, 2.0A and 2.0B. CAN 2.0A is the earlier standard with a message identifier of 11 bits, while CAN 2.0B is the extended protocol with a message identifier of 29 bits. The CAN 2.0B standard is backward compatible with the 2.0A standard meaning that 2.0B controllers can handle both the 2.0A and 2.0B standards.

24.7.1 OBD

On-Board Diagnostics (OBD) is a generic term referring to a vehicle's self-diagnostic and reporting capability. OBD systems give the vehicle owner or a repair technician access to state of health information for various vehicle sub-systems. The amount of diagnostic information available via OBD has varied widely since the introduction in the early 1980s of on-board vehicle computers, which made OBD possible. Early instances of OBD would simply illuminate a malfunction indicator light (MIL), while modern OBD implementations use a standardized fast digital communications port to provide realtime data in addition to a standardized series of diagnostic trouble codes (DTCs), which allow one to rapidly identify and remedy malfunctions within the vehicle (www.wikipedia.org 2006).

After 1-JAN-96 all cars and light trucks built and sold in the US were required to have an OBD system. The requirements are:

- An OBD connector should be available within 1 meter from the driver,
- a stricker or nameplate should state that the system is "OBD II compliant".

OBD II is the second generation of the OBD system being build around the CAN bus. The OBD connector is shown in Figure 24.14,

using the following pins:

Figure 24.15: A stand alone OBD tool that can also be connected to a PC using RS-232C or USB (www.elektor-electronics.co.uk).

Pin	Description	Note
1		
2	Bus positive Line	SAE-J1850
3		
4	Chassis ground	
5	Signal ground	
6	CAN high	ISO 15765-4 and SAE-J2234
7	K line	ISO 9141-2 and ISO 14230-4
8		
9		
10	Bus negative Line	SAE-J1850
11		
12		
13		
14	CAN low	ISO 15765-4 and SAE-J2234
15	L line	ISO 9141-2 and ISO 14230-4
16	Battery voltage	

Different tools exist that can be connected to the OBD-II port for reading the status of the vehicle, resetting different parameters like warnings etc., and parameters can be changed as well. An example of a OBD tool is shown in Figure 24.15.

There are 5 protocols in use with the OBD-II interface, and often it is possible to make an educated guess about the protocol in use based on which pins are present on the OBD connector (www.obd.dk: nov-09).

24.8 DeviceNet

DeviceNet is a communication protocol used in the automation industry to interconnect control devices for data exchange. It uses Controller Area Network (CAN) as the backbone technology, and defines strong application layer to cover a vast range of device profiles. The communication speed is 125 kbits/sec, 250 kbits/sec, or 500 kbits/sec, the maximum length is 500 meters, and the maximum number of devices is 64. The cable also include 24 VDC for power to the devices.

24.9 FDDI

FDDI (Fiber Distributed Data Interface) is a set of standards for data transmission on fiber optic lines in a local area network (LAN). The LAN can be extend in range up to 200 km. The FDDI protocol is based on the token ring protocol. In addition to being large geographically, an FDDI local area network can support thousands of users. FDDI is frequently used on the backbone for a wide area network (WAN).

A FDDI network contains two token rings, one primary ring and one secondary ring for possible backup in case the primary ring fails. The primary ring offers up to 100 Mbps capacity. If the secondary ring is not needed for backup, it can also carry data extending the capacity up to 200 Mbps.

24.10 Modbus

Modbus is a serial communications protocol published by Modicon in 1979 for use with its programmable logic controllers (PLCs). It has become a de facto standard communications protocol in industry, and is now the most commonly available means of connecting industrial electronic devices. The main reasons for the extensive use of Modbus over other communications protocols are:

1. it is openly published and royalty-free,
2. it can be implemented in days, not months,
3. it moves raw bits or words without placing many restrictions on vendors.

The Modbus consists of 3 types:

1. Modbus ASCII; using ASCII communication on a serial line,
2. Modbus RTU; using binary communication on a serial line,
3. Modbus/TCP; using the TCP/IP protocol on a network.

The Modbus/TCP is the protocol that can be used as a fieldbus. The Modbus is a master/slave protocol giving that the master must poll all the slave devices. The maximum number of slave devices are 247 on one data link. Modbus consists of either 1 bit registers (coil) or 16 bits registers (holding) meaning that it is suitable for digital values (coil registers) and integers (holding registers). The number of registers are 65535. Using floating numbers the resolution has to be defined between the master and the slaves. Using temperature a resolution of 0.1 °C can be used so a measured temperature of 36.8 °C is stored in the holding register as 368. Using 16 bits the range will still be ±32767.

The Modbus communication is based on a start register and a number of following registers. The optimum configuration is to use only continuous address ranges, this will minimize the number of Modbus messages on the data link. The maximum number of registers in one message is in the range of 125 to 127, and the maximum length of a Modbus message is 256 bytes.

The Modbus/TCP protocol is a protocol that can be used for IoT devices, defining the layer 7 in the OSI model for these devices. The layer 7 in the OSI model is very often missing from IoT devices as these devices are based only on the TCP/IP protocol.

24.11 Industrial Ethernet

Ethernet has in the past been considered as not suitable for high performance process control, leading to design of other type of protocols, often company specific version of "Ethernet" types. Lately this has been changing, the popularity of Ethernet across ranges of industries have adopted Ethernet as an industry standard like IEEE 1588. By using "standard" Ethernet as a fieldbus means that several vendors can communicate with each other, an advantage for the manufactures and system integrators. Ethernet already contains a lot of network monitoring and diagnostic tools that can be used in a fieldbus as well. Industrial Ethernet or Ethernet/ip (Industrial Protocol) is a name given to the use of the Ethernet protocol in an industrial environment, for automation and production machine control.

Some of the advantages are:

1. Increased speed, up from 9.6 kbit/s with RS232 to 1 Gbit/s with IEEE 802 over Cat5e/Cat6 cables or optical fiber,
2. Increased overall performance,
3. Increased distance,
4. Ability to use standard access points, routers, switches, hubs, cables and optical fiber, which are immensely cheaper than the equivalent serial-port devices,

5. Ability to have more than two nodes on link, which was possible with RS485 but not with RS232,
6. Peer-to-peer architectures may replace master-slave ones,
7. Better interoperability,
8. Support for TCP/IP and UDP,

The most used protocols for industrial ethernet is Ethernet/IP, PROFINET, Ethercat, Powerlink, and Modbus TCP. Several of these protocols also support each other so the these protocols may be mixed in the same network. Some difficulties of using industrial Ethernet are:

1. Migrating existing systems to a new protocol (however many adapters are available),
2. Real-time uses may suffer for protocols using TCP (but some use UDP and layer 2 protocols for this reason),
3. Managing a whole TCP/IP stack is more complex than just receiving serial data.

24.11.1 Powerlink

Ethernet Powerlink is a deterministic real-time protocol for standard Ethernet communication systems. It is an open protocol managed by the Ethernet POWERLINK Standardization Group (EPSG)(www.wikipedia.org 2010). It is based on the standard Ethernet, but extent the protocol with a mixed Polling- and Times-licing mechanism. The protocol also support hot plugging.

The specifications:

- 0.1 μs system synchronization,
- 100 μs cycle time,
- 100 Mbit/s (CAT6),
- 240 nodes,
- 100 m (wire) / 2 km (fibre) segment length between nodes.

24.12 Fieldbus versus IoT

Can Internet of Things (IoT) replace Fieldbus? Using the OSI model will a Fieldbus support the physical layer (layer #1) and the data link layer (layer #2) while the IoT will support the four lower layers up to the transport layer (layer #4). This means that the functionality of the Fieldbus is more on local data busses while the IoT main functionality is for external network. This means that the security and speed will always be higher for a Fieldbus then an IoT system, and the cost will be lower for a Fieldbus as the communication cost will be lower supporting less functionality according to the OSI model. An IoT system will also normally depend on a cloud system as a storage and communication site for the collected information from the IoT devices.

24.13 Building Automation Systems

Building Automation Systems (BAS) are also based on field buses and these buildings are often refered to as smart buildings. Systems in such buildings can be: 1) HVAC; heating, ventilating, and air conditioning, 2) Elevator Control System, 3) Alarm system (security system), 4) Light control system, 5) Eldercare, 6) Childcare and 7) Entertainment to mention a few.

The field buses used in BAS and based on a specific protocol are amoung others:

1. EIB; European Installation Bus,
2. KNX; Konnex;
3. LON works; Local Operating Network works,

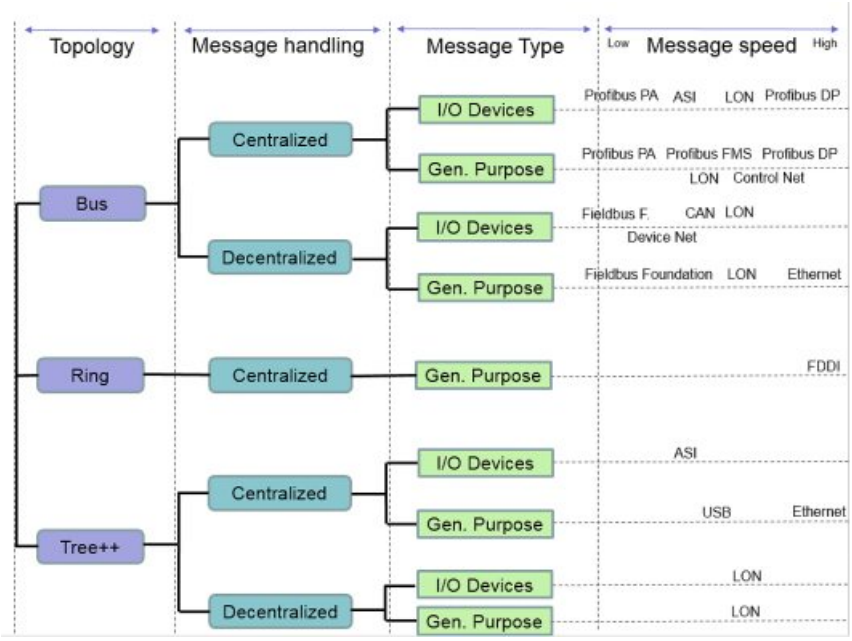


Figure 24.16: The classification of the Fieldbuses based on topology, message handling, message type and message speed.

4. DALI; Digital Addressable Lighting Interface; Light control,

More general protocols for BAS are:

1. OPC (Open Process Control); server/client based, object based,
2. BACnet (Building Automation and Control network); server/client or peer to peer based, object based,
3. Modbus, master/slave based, register based.

24.14 Classification of serial bus systems

A field bus can be classified based on topology, location for processing of information, size of data messages and speed of transmitting and receiving data messages in the network. Note that all buses as based on digital serial communication. By combining some of the classification parameters as topology, processing and message speed, an overview of the field buses mention can be as shown in Figure 24.16. Most of the field buses can be used to most of the tasks in a process control system, but by evaluating the classification parameters a specific fieldbus can be used to optimize the communication and throughput of a system.

24.15 Field bus summary

A summary of some field buses:

Name	**Devices**	**Speed (kbit/s)**	**Length**	**Note**
PROFIBUS	126	9.6 – 1200		PROFIBUS-DP
Fieldbus Foundation	240	31.25		H1
				HSE
Controlnet	99	5000		Redundency
Devicenet	64	500		
LON works	32000	10 – 1250		
Interbus	512	500		
CAN	∞	10 – 1024		
Modbus	247	9.6 – 110		
HART	16	1.2		

A study from Advisory Group (ARC), an American Analysis company, indicates that in 2010 the usage of industrial ethernet is including about 4 million nodes. The major part of these nodes are based on the following protocols:

- 32% of the nodes are using Ethernet/IP,
- 20% of the nodes are using Modbus TCP.
- 16.5% of the nodes are using Ethernet PowerLink.
- 13.9% of the nodes are using PROFINET.

The same analysis also checked the trend toward 2015 indicating that Ethernet/IP will still be the leader, however the usage of PROFINET seems to have increased most.

Part VIII

OPC

Chapter 25

Introduction

25.1 Background

In the industry there is a move from proprietary solutions to open vendor independent standards. As well as reducing costs it allows the choice of components according to their performance and reduces the dependence on suppliers. The important part in data communication is the protocol defining a set of rules for data exchange between software applications. Figure 25.1 shows two computers exchanging information using a wired communication link. The protocol should define how the computers will exchange this information and the specification of the information in the data message sent between the computers. The Field Bus defined how several computers can be connected to the same network in a industrial system. The bus protocol must define the structure of the data message sent between the computers, but also how to use the "bus" as a common communication link. Only one computer can transmit data on the "bus" at a time. Figure 25.2 shows several computers connected on a bus using a bus protocol for controlling the communication between the computers. Normally the protocol will define the structure of the communication between the computers and the contents of data messages to some degree. However are the details about specific data values and the meaning of specific bits in the protocols not defined clearly. How can the software applications in a network system exchange a specific data value and know the status of this data value? This is one of the reasons for the OPC standard.

The specification of the Dynamic Data Exchange (DDE) protocol from 1987 provide a first solution for the data exchange between MS-Windows based applications. The main drawback of this solution was low bandwidth not very well suited for real-time systems. High bandwidth will be a major requirement for automation systems where exchange of data will be important. See Figure 25.3.

The specification of the Object Linking and Embedding (OLE) protocol, from 1990, a distributed object system and protocol, provided better bandwidth. OLE is said to be the evolution of the DDE. While DDE was limited to transferring limited amounts of data between two running applications, OLE was capable of maintaining active links between two documents or even embedding one type of document within another. The main benefit of using OLE, next to reduced document size, is the ability to create a master document. References to data in this document can be made and the master document can then have changed data which will then take effect in the referenced document. See Figure 25.3.

The OLE protocol later evolved to become an architecture for software components known as the component object model (COM) given that the documents can be objects as well. Both OLE and COM was developed for communication on a single computer, but the *Network OLE* protocol as later evolved to the Distributed Component Object Model (DCOM) protocol for software components distributed across several networked computers to communicate with each other.

The usage of DCOM and OLE was used for developing the open standard "*OLE for Process Control*" (OPC), as the original name for an open standard specification developed in 1996 by an industrial au-

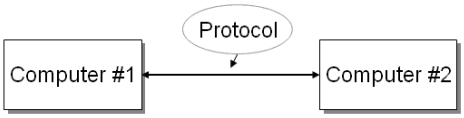


Figure 25.1: A protocol is needed when 2 computers are going to exchange information.

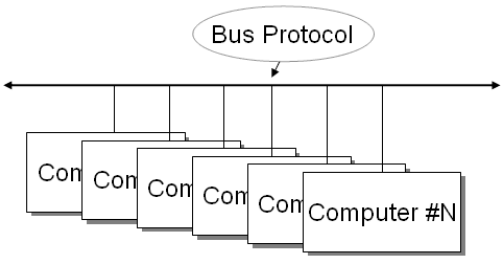


Figure 25.2: A protocol is needed when several computers are going to exchange information. The protocols must define the structure of the data messages sent between the computers, but also how the computers should cooperate to use the bus as a common communication link.

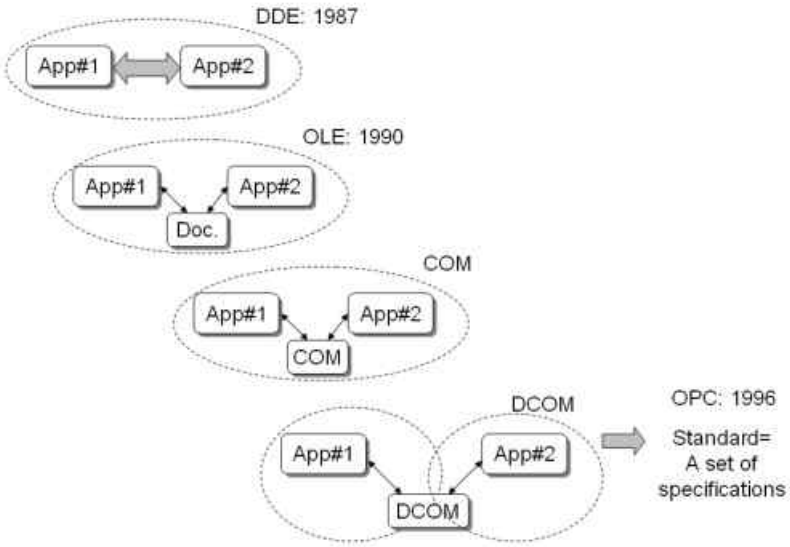


Figure 25.3: The background of the OPC standard on the Windows platform, from DDE exchanging data directly to COM exchanging information in objects.

tomation industry task force. The standard specifies the communication of real-time plant data between control devices from different manufacturers. The background for the OPC standard is shown in Figure 25.3.

After the initial release, the OPC Foundation[1] was created to maintain the standard. Today the OPC standard is a series of standard specifications and the standard is often called Open Process Control (OPC). The OPC is only software modules and supporting the application layer (layer 7) in the OSI model.

25.2 Operating systems

The OPC Specification was based on the OLE, COM, and DCOM technologies developed by Microsoft for the Microsoft Windows operating system family. The specification defined a standard set of objects, interfaces and methods for use in process control and manufacturing automation applications to facilitate interoperability.

OPC was designed to bridge Windows based applications and process control hardware and software applications. It is an open standard that permits a consistent method of accessing field data from plant floor devices. This method remains the same regardless of the type and source of data.

COM and DCOM was a proprietor protocol from Microsoft meaning that OPC first could only be used on systems based on the Windows operating system. Several other operating systems are now supporting the DCOM protocol like Solaris (Sun), Unix, VMS (Digital), Linux and AIX (IBM) giving

[1] The WEB site: http://www.opcfoundation.org (4-FEB-07).

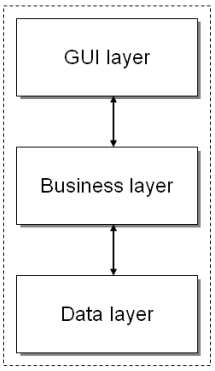


Figure 25.4: A 3 layers software application, divided the functions of an application into three different layers (or tiers) with minimum interactions.

that OPC can be used on different operating systems.

25.3 Software application

The term SCADA (Supervisory Control And Data Acquisition) refers to a large-scale, distributed measurement (and control) system, see Figure 25.10. A SCADA system includes input/output signal hardware, controllers, HMI, networks, communication, database and software. A more general description of such a system is a Process Control System (PCS).

Programmable automation controller (PAC) is a compact controller that combines the features and capabilities of a PC-based control system with that of a typical programmable logic controller (PLC), see Figure 25.9. PACs are often used in industrial settings for process control, data acquisition, remote equipment monitoring, machine vision, and motion control. Additionally, because they function and communicate over popular network interface protocols like TCP/IP, OLE for process control (OPC) and SMTP, PACs are able to transfer data from the machines they control to other machines and components in a networked control system or to application software and databases (www.wikipedia.org 2006). A PCS can be a SCADA system or a PAC.

The PCS (SCADA and PAC) software often consists of a 3 layer application model with the following layers (see Figure 25.4) :

	Type of layer	**Description**
1	User Interface	Interface between the user and the application
2	Business	Logic of the application; monitoring, calculation and analysis
3	Data	Data handling and storage, and external data flow.

The PCS software is a distributed measurement system getting the information from different distributed computer equipment (DCE) so a 3 layered software application on a SCADA or PAC system can be as shown in Figure 25.5.

A complex software system and the exchange of data will depend on a lot of different protocols. What if a new function has to be integrated into the system? To solve the problems regarding the exchanging of data and connection between all the modules, one of the solutions can be OPC. One solution of the system in Figure 25.5 is shown in Figure 25.6.

As can be seen in Figure 25.6 the system has a much easier structure and any extensions or changes are easy as the information will be available at the OPC protocol. The drawback is that every system must have support for the OPC protocol. A more specific figure is shown in Figure 25.7 showing two applications as OPC clients using informations from three OPC servers.

25.4 Data models

The OPC data model is based on servers and clients, the server being the source of the information and the client being the source of "needing" the information, see Figure 25.7. The server will be the owner of the information, and one or several clients can connect to that server to copy the information.

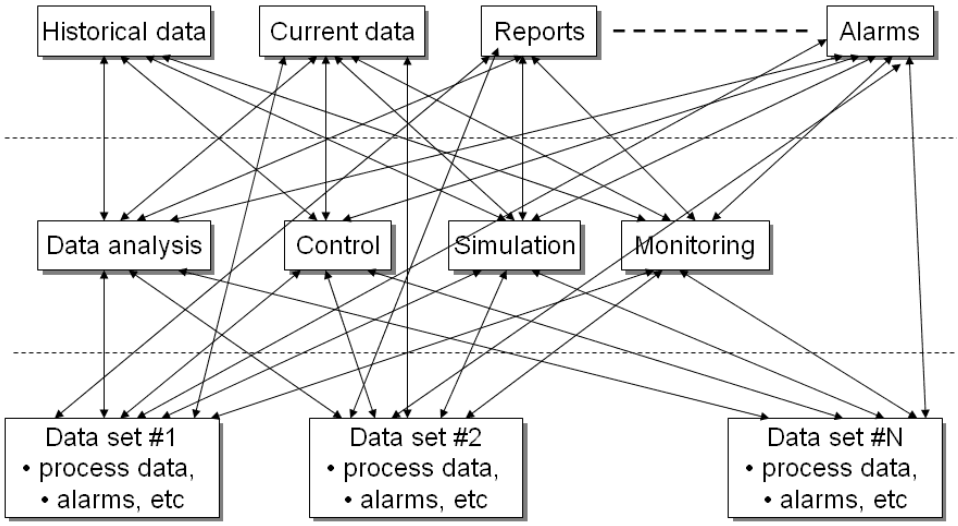


Figure 25.5: A general SCADA or PAC software application with a lot of different protocols between the software modules (Krogh 2005).

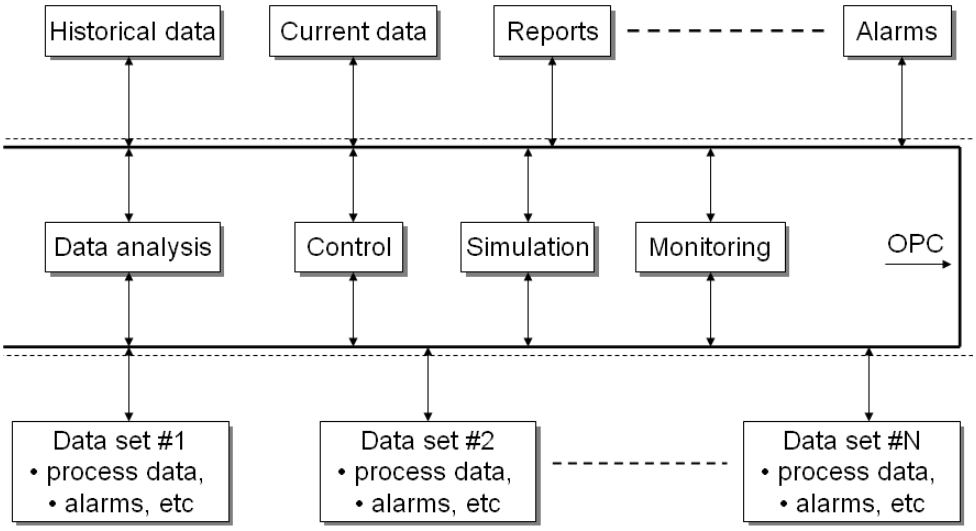


Figure 25.6: SCADA or PAC software using OPC as protocol for interconnection (Krogh 2005).

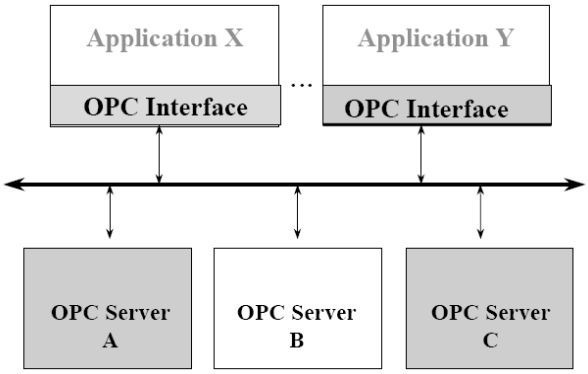


Figure 25.7: Applications in a system with OPC clients and OPC servers. The OPC servers communicate with the I/O devices (RTU) supporting the necessary protocols. The applications include an OPC client, communicating with the OPC servers only, not the I/O devices.

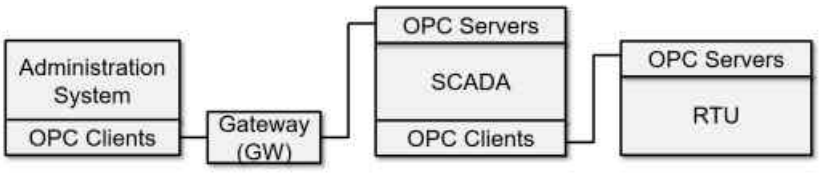


Figure 25.8: The usage of OPC servers and clients in a Process Control System. The Gateway may also include a firewall depending on the network structure.

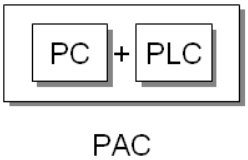


Figure 25.9: A PAC system as a combination of a PC and a PLC system.

The OPC server will be a software module with some sort of data access for the information, and include different protocols for interfacing the I/O hardware modules if such modules are connected. The OPC server will be a gateway in the system as an interface module between the information and the other software modules in the system. All OPC clients must also have the OPC protocols for communicating the OPC servers. The complexity of the software modules will be much higher for the server than the clients. Looking at Figure 25.6 all the data set modules need to contain OPC servers when using the OPC protocol. The modules in the UI layer need to contain OPC clients to be able to access the data sets, but what about the business layer? Should these modules contain the client, the server or both? Actually, they should contain both as they need to communicate with the data layer and serve the UI layer. Figure 25.8 shows the minimum usage of OPC clients and OPC servers in a PCS system, normally there will be several OPC servers for each layer.

The OPC server for the RTU in Figure 25.8 may run on the SCADA system as some RTUs do not have the computational power to run an OPC server. The gateway will often include a firewall as the administration system will run in the IT (Information Technology) section while the SCADA system will be part of the OT (Operational Technology) section

25.5 Communication models

PCS systems, using SCADA, DCS, or PAC as the main system module, will always be a distributed system and from a communication point of view a distributed system consists of a service provider and a service user. One part of the system wants or needs information, as the other part of the system has access to. A PAC system, as a combination of a PC and a PLC (Programmable Logic Controller), is shown in Figure 25.9. A PAC contains the cpu power of a PC and the robustness of a PLC. A DCS can be used as a solution for a small SCADA system. A SCADA system, with distributed I/O as a HMI (Human Machine Interface), PLC, DCS (Distributed Computer System), and PAC is shown in Figure 25.10. As shown can a DCS or PAC be part of a SCADA system indicating that a SCADA

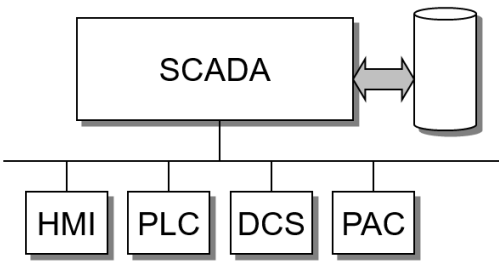


Figure 25.10: A SCADA system with distributed I/O as HMI, PLC, DCS, and PAC.

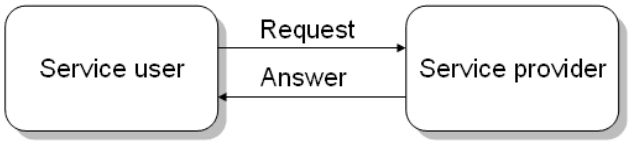


Figure 25.11: User and provider messages.

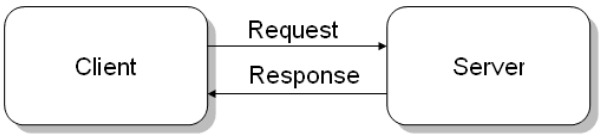


Figure 25.12: Client/server model with a request and a response.

system normally will be a more complex system than a DCS or PAC system, depending on the system requirements. The PLC, DCS and/or PAC is used as I/O devices for the plant, while the HMI is used as I/O devices for the user or operator, and the information should be distributed between these devices, and stored in the database as well.

In these system there will be a lot of information, and some subsystem has the information and other subsystems need this information. The service provider and service user must be logically connected for the user to get information from the provider. The service user will ask the service provider for information. See Figure 25.11.

This logical connection can be described using two different communication models:

1. Client / server model,
2. publisher / subscriber model.

OPC supports both communication models and distinguishes only between synchronous and asynchronous services. In asynchronous services can another request be answered before this answer, there is no relationship between the requests from the clients and the answers back to the clients. Note that all servers are supporting the client / server model, however the publisher /subscriber is optional. In a system with heavy network traffic the publisher / subscriber model is the best solution.

25.5.1 Client/server model

In a client/server model the server is the owner of the data (or resource) and the client, or clients, must poll the server to exchange data. The advantage of a client/server model is that several clients can access the data (or resource) at the server at the same time, using no server only one "client" can access the data (or resource) at a time. The server normally queue the requests and will answer as fast as possible. Remember that you should not have copies of your data in the process system, the data should be available in only one location and the client/server model is good way of dealing with the data resources.

The communication between the client and the server is determined by the OPC protocol. The sequence always starts with a client sending a service request to the server. The answer from the server will be a service response. Figure 25.12 shows the request and response in a client/server model.

Normally will several clients connect to a server as shown in Figure 25.13. A challenge with the client/server model is the polling (requests) from the client, the client should request a new set of data every time the data set is used. In real-time system this can be a lot of unnecessary requests, a better way can be the usage of the publisher/subscriber model.

The client /server model can be either synchronous or asynchronous. These types of read and write operations are shown in more detail in Figure 25.14 where the different read and write operations will depend on the implementation of the clients.

25.5.2 Publisher/subscriber model

The publisher/subscriber model assumes a cyclic data supply by the publisher, where the data transfer from the publisher depends on either an external request or an internal event (e.g. a timer). The client

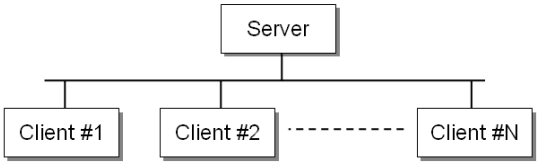


Figure 25.13: Several clients connected to a server.

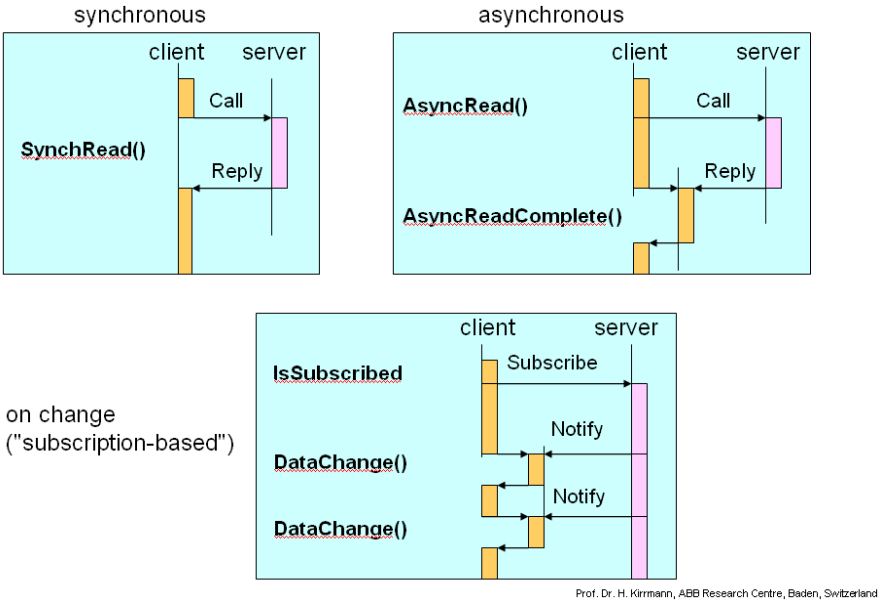


Figure 25.14: The synchronous, asynchronous, and subscription based read and write operations between a OPC server and a OPC client. Client/server types on the top and publisher/subscriber type at the bottom (Kirrmann 2007).

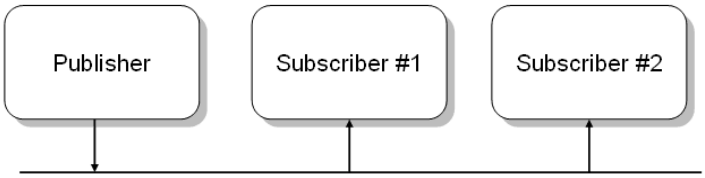


Figure 25.15: A publisher/subscriber model with 1 publisher and 2 subscribers.

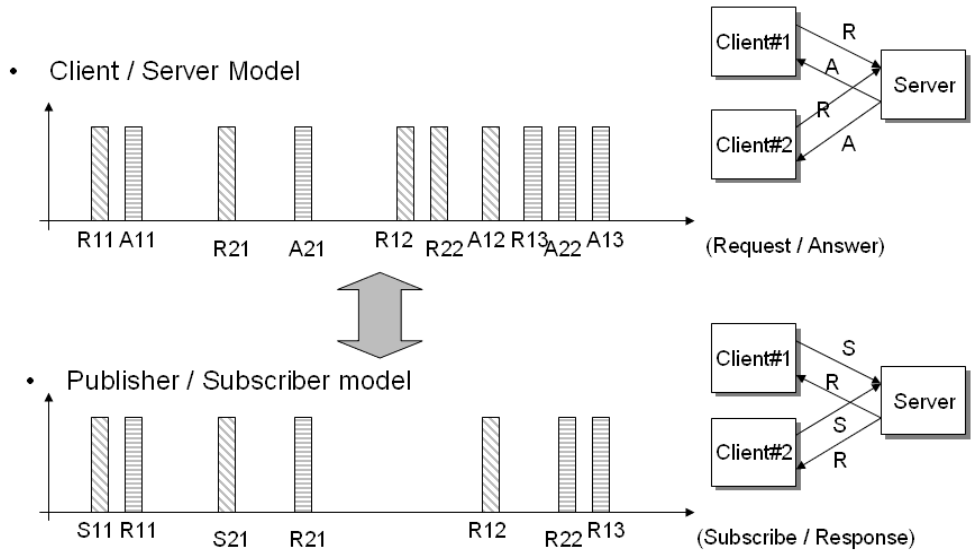


Figure 25.16: The client/server model shown on the top and the publisher/subscriber model shown at the bottom, both in the time domain. Notice the difference in trafic load between these models.

or the clients must first subscribe to data and define the type of subscription (request and/or events), and the server will give a response when the type of subscription is activated without the need of polling from the client. One or more clients can subscribe of the same type of data, as shown in Figure 25.15

Will the publisher be a client or a server? The server is the "owner" of data and the most logical solution will be to let the server also be the publisher. The reason for using the publisher/subscriber model, sometimes also named as subscribe/response, is to limit the amount of traffic on the communication media. Note that any communication media between devices in a distributed system will be an important resource in such a system.

The differences of these communication models are shown in Figure 25.16 showing the messages in the time domain. Notice the number of messages, the client/server model needs more network traffic to get the same number of information from the server.

Chapter 26

OPC specification

26.1 Introduction

The OPC protocol consists of a set of specifications, and most of these specifications are related to different server specifications. These specification is only software and for supporting the application layer (layer 7) in the OSI model. Most of these specification are:

	Name	Abbreviation	Classic
1	Common		X
2	Data Access	DA	X
3	Alarm and Events	AE	X
4	Data eXchange	DX	X
5	Historical Data Access	HDA	X
6	Batch		X
7	Complex Data	CD	X
8	XML-DA		X
9	Security		X
10	Commands		X
11	Unified Architecture	UA	
12	Express Interface	Xi	

The specifications are developed in Working Groups in the OPC Foundation[1] and only members will have access drafts, pre-releases and info from the working group. The released specifications are however available for everybody[2]. The specifications are often dealing with the data on the server side, giving a set of different servers for different specifications. The specifications numbered from 1 to 10 are refered to as the classic OPC standards.

The interconnection between the different specifications is shown in Figure 26.1.

Each specification is a description of a server, a software module that can be running on a node in the system. Note that several servers can be running on the same node in the system. In small systems will most probably all the servers be running on the same node. Figure 26.2 shows the usage of some of the OPC servers, the connection to the OPC protocol and the plant. Note that the OPC protocol is the connections between the systems and also the connection point for the OPC clients.

26.2 Communication

COM and DCOM is the basis for the OPC communication and the COM/DCOM connection consists of: 1) Objects, and 2) Interfaces.

The interface is connection point between the client and server object, the client is only able to "see" the contents of the DCOM server by the interface connection, see Figure 26.3. The DCOM server will be like a "black box" for the client as the client do not know anything about the functonality of the server.

[1] The WEB site: http://www.opcfoundation.org (4-FEB-07)

[2] The specification is avaliable at: http://www.opcfoundation.org/ → Downloads → Specifications.

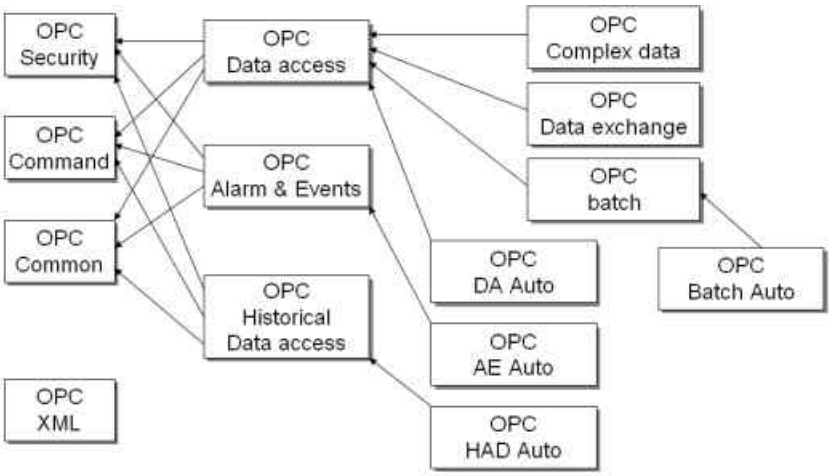


Figure 26.1: The interconnection of the OPC specifications.

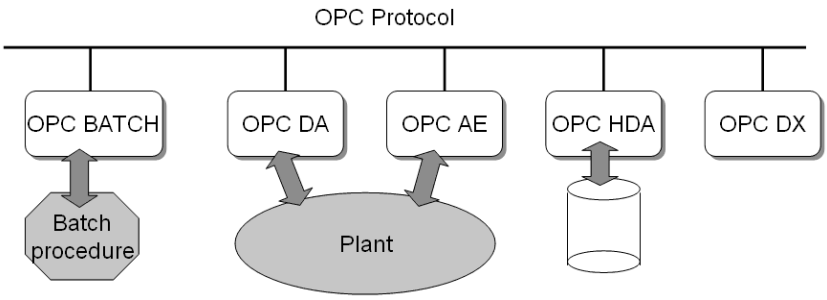


Figure 26.2: The connections between some of the OPC servers in a process system.

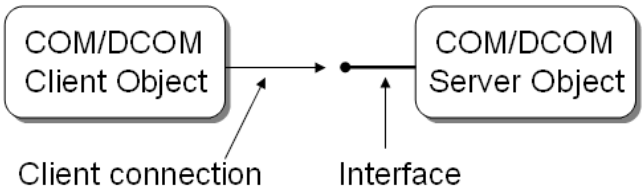


Figure 26.3: A DCOM client and server with a interface connection.

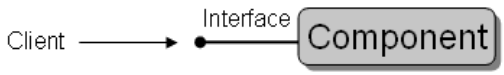


Figure 26.4: Local communication in the same process (application).

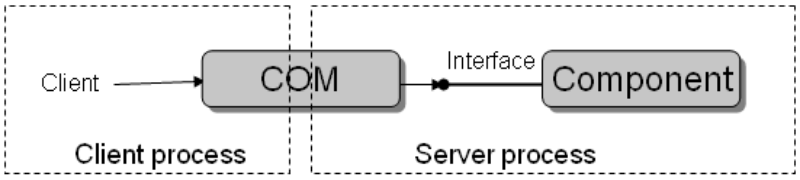


Figure 26.5: The interprocess communication between the OPC client and the OPC server using COM.

A set of functions are available at the main interface, like: 1) Addref(); add the reference of the client, 2) Release(); get the release number of the server object, 3) QueryInterface(); request information about the functionality of the server, and 4) The COM/DCOM solution gives the same functionality both on local and remote systems.

The communication between the client and the server will be the responsibility of the client to find the interface of the server component and connect. If the client and the server is in the same software application, which is not very likely, but possible, the client will connect directly to the server, as shown in Figure 26.4. All the code for the application will be in the same process giving a fast and direct connection. This solution is only possible for very special systems, and will not utilize OPC. A more usual way of doing the communication is between two different processes on the same computer. This communication is often called InterProcess Communication (IPC) and a lot of different IPC exists, also called Middleware. OPC is using COM and DCOM as the IPC, and is shown in Figure 26.5. The COM module is used as the IPC "channel" for the OPC on a local machine.

IPC can also be used between processes on different computers but the IPC must then support network connections as well. COM is only for IPC on the same computer while DCOM can be used communication between two processes on two different computers. The usage of DCOM is shown in Figure 26.6.

The network support can be different protocols as well and a set of protocols that DCOM can use is shown in Figure 26.7.

The most used classic OPC standards in the process industry are:

OPC DA	Data Access
OPC AE	Alarm & Events
OPC HDA	Historical Access

26.3 OPC Common

The OPC Common is a common definition used for several of the OPC specifications and contains mainly of instruction and interfaces for registration of OPC software modules. Some of the most important OPC Common interfaces are:

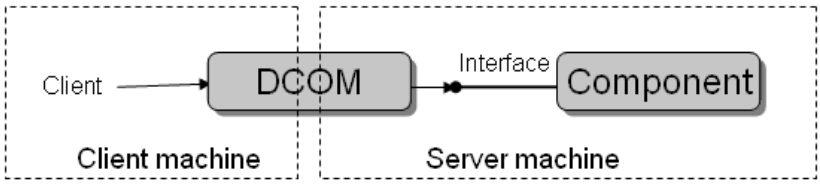


Figure 26.6: Usage of DCOM for COM communication between two computers.

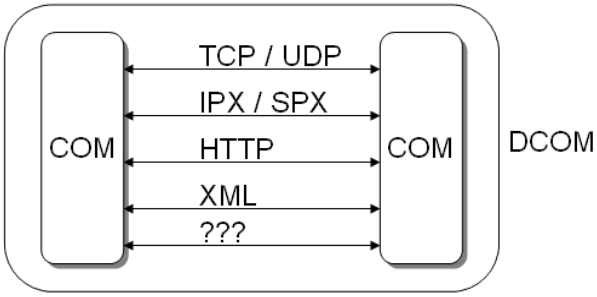


Figure 26.7: Network connection in a DCOM.

IOPCServerList	Find the OPC servers on a computer,
IOPCCommon	Several LocaleID functions like let the client define the language,
IOPCDataCallback	Callback functions for the client,
IOPCShutdown	Callback to all clients from a server for disconnection.

26.4 OPC Data Access (OPC DA)

The OPC Data Access (DA) specification version 3.0 has 19 interfaces and 69 methods (functions). Specification version 1.0*A* is from 1997, specification 2.0 from late 1998. The functions will differ from different specification so it will be important to know the specification number of the DA server. The usage of the OPC DA is:

1. reading of measurement values,
2. calculation and estimation of values,
3. writing of values,

OPC DA server has implemented a set of services, and the clients are using these services. Figure 26.8 shows an example of a system using an OPC DA server.

Tags are used a lot in the process industry and is normally assigned to a piece of information. A tag consists of a name describing a single point of information meaning that a process system (plant) consists of hundreds or even thousands of tags. The figure shows that the DA server contains one tag for each measurement point and controller point in the plant, and it is the responsibility of the DA server to get (or set) the information from the controllers. This is one of the reasons for the complexity of the servers, they need to have drivers for a lot of controllers and/or measurement systems.

The OPC servers has different rooms for grouping of items and adding access and names, and the client can use group index instead of an item index. The group concept is important and one or more items can be added to the same group. The group information is stored on the server, but the server lets the client maintains the group information, and the client can also browse in the name space of the server. See Figure 26.9 where the name space contains information about the items on the server. The *name space* is the area in the server where all the groups and tags information are stored.

One or several groups can be defined in the name space, and the groups can be used for connection the several clients, or the same client. The same input and/or output signals can also be connected to several groups, giving that any client can connect to tags or groups. These choices give a flexible way of configuration, and with the group configuration on the server it will be independent of the clients. Figure 26.10 shows a DA server with several OPC clients connected, and with a name space with different groups.

A more detailed figure is shown in Figure 26.11 showing the tree structure of the groups and tags information in a OPC DA server. The information can be stored in a layer structure with a root, several levels of branches containing the groups, and a leave level containing the tags. Each tag indicates a specific point of measurement or controller for a process.

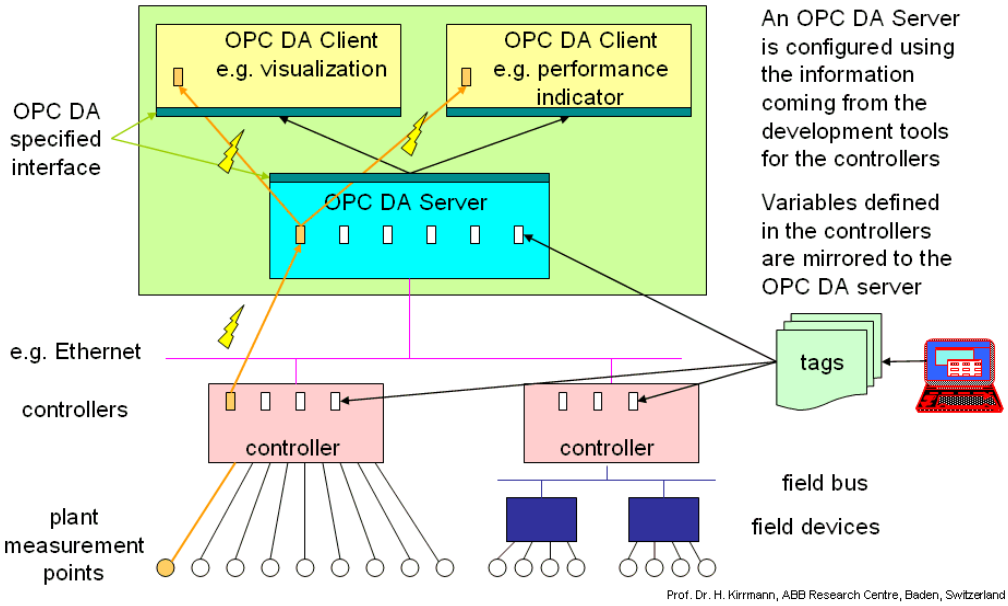


Figure 26.8: An example of a system with an OPC DA server and OPC clients using tags for the I/O values. Note that the OPC DA server will not use OPC for communication with the I/O devices. (Kirrmann 2007).

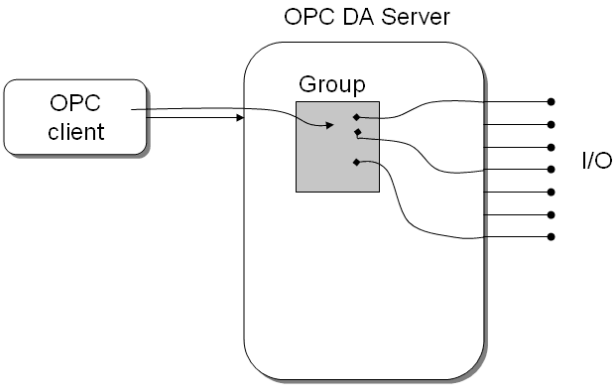


Figure 26.9: A client connection with a DA server group with a set of I/O items.

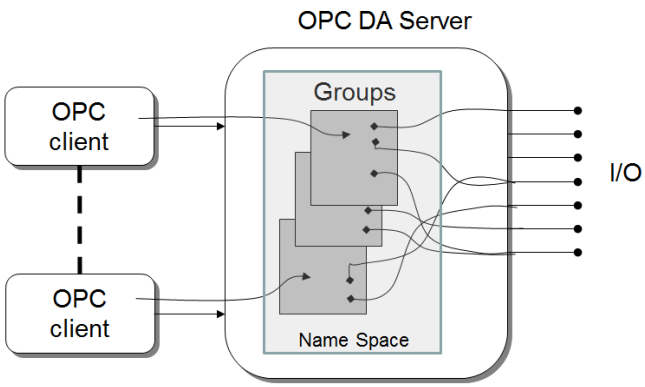


Figure 26.10: Several client connections with a DA server groups using the name spave area of the server.

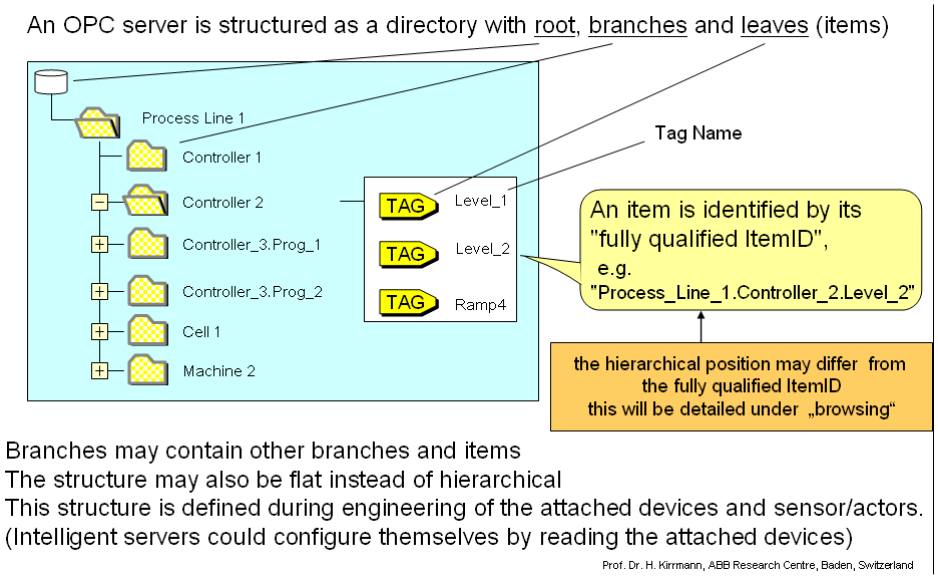


Figure 26.11: The name space information of a DA server (Kirrmann 2007).

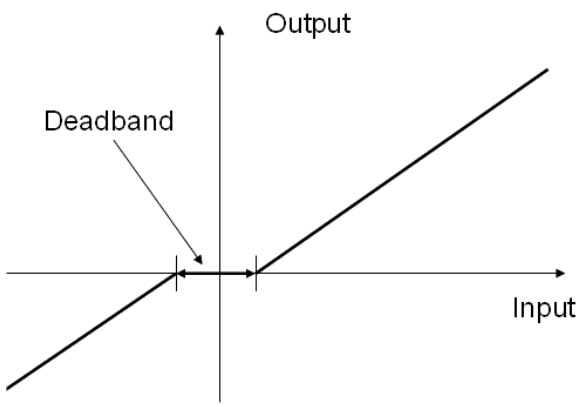


Figure 26.12: A deadband meaning that the output will not change even if the input is changing (system is "dead").

Exchange of measurement values can be done by groups, or items belonging to a group. The read/write operations between the server and the client to access the values can be: 1) reading or writing synchronous, 2) reading or writing asynchronous and 3) reading as a subscription.

The subscription based operation can depend on settings like 1) dead band variations (in %); The dead band is an area of a signal range (or band) where no action occurs (the system is dead), see Figure 26.12. Dead band is a way of data compression, remove some of the data but keep the information, 2) minimum time interval (in seconds) and 3) for each item in the group, or the whole group.

A sample of a value saved in the OPC DA server has the following descriptions:

	Type	Description
1	Value	Only the current value, no history
2	Quality	Signal codes (0 – 255) like GOOD, BAD, CONFIG_ERR, DEVICE_ERROR, etc
3	Timestamp	Coordinated Universal Time (UTC) / Greenwich Mean Time (GMT)
4	Access right	Read, write, delete, etc.
5	Propertis	SI unit, scaling, description, etc.

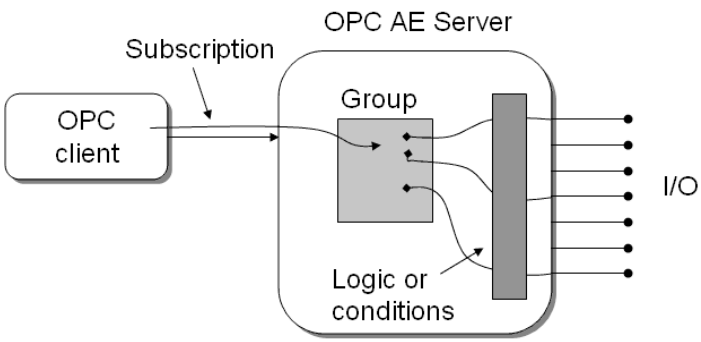


Figure 26.13: A OPC client and a OPC AE server with a group for condition events.

26.5 OPC Alarms & Events (OPC AE)

The first server specification implemented was the OPC Data Access, but to be able to use OPC in a PCS such a system must have support for alarms as well. As these OPC servers are just software modules, a new server specification was developed for the alarms as alarms and events (AE). The DA server will handle analog values, while the AE server will handle events only. The usage of an alarm and event server is monitoring and reporting events. Alarm is defined as an abnormal condition that requires special attention, either a single state or multi state condition. An example of a single state condition is a sensor device error, while a multi state condition can be a level alarm in a vessel with both high and high-high states. An event is a condition change that may be an alarm condition. Operator action or change of configuration data are events that are not alarm conditions.

An alarm is an event condition in the process control system, and the type of events in an alarm and event server are (Emerson 2016):

1. simple events; an event with an informal messages that do not require any action (no alarm) like warnings or just information to the operator,

2. tracking events; an event with an informal messages and an actor id that do not require any action (no alarm). Tracking means an actor id showing the change of any set points, configuration data or other types of access by an operator,

3. condition events; an event with a state information that requires some sort of response or acknowledgement by the user or operator. Type of condition events:

   (a) single state alarms; on/off alarms like a defect device, one single alarm limits, or time-out alarms to mention some,

   (b) multi state alarms; multi state alarms with both low and high alarms limits, or several high limits like high and high-high.

Usage of the OPC AE server will be detections of alarms and/or events from one or more sources. These sources can be: 1) alarms on sensor devices, 2) alarms on sensor values/data, 3) alarms on control parameters, 4) status on hardware connections, and 5) status on systems and subsystems. The best communication methods for an AE server is the publisher/subscription method to avoid polling meaning that the AE server will publish to one or more clients using subscription (including a filter) whenever an event. The type of clients can be GUI systems, reporting systems and any separate alarm systems to mention some.

An example of a OPC client having an event subscription on an OPC AE server is shown in Figure 26.13.

The structure for the connection between a client and an OPC AE server is shown in Figure 26.14 showing the connection sequence. The sequence for connecting and subscription, as shown in Figure 26.14, will be 1) the client connects to the AE server, 2) the client sets up a subscription request to the AE server, getting a connection point (CP), 3) the client will configured the connection point (CPC), and 4) the connection point (CP) will send an event whenever the condition of the configuration of the connection point is changing.

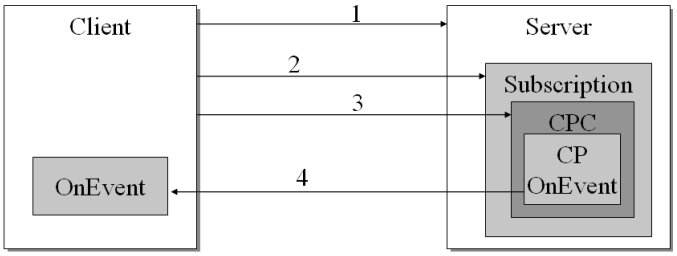


Figure 26.14: The connection sequence between a client and a AE server in an OPC system.

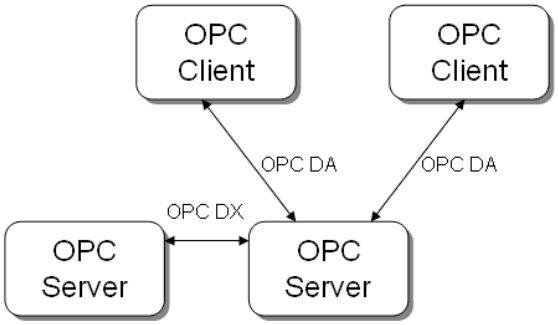


Figure 26.15: The normal differences of OPC Data Access (DA) and OPC Data Exchange (DX). Note that a DX server is an extended DA server and can also be conected by the OPC clients.

26.6 OPC Data eXchange (OPC DX)

An OPC DA server will only contain data from one or a few specific RTU devices. If an advanced controller function is wanted, requiring information from several DA servers, the DA servers should be able to exchange information. How to get information from several DA servers? The normal way will be to make an OPC client reading the values from the specific OPC DA servers and write these values to the DA server performing the advanced controller function. The disadvantage of this solution is that the advanced controller will be depended on the client which is not wanted. The functions and services of the servers should not be dependent on any clients. The normal way for the OPC Foundation is to define a new OPC server, and this is the OPC Data Exchange (DX) server.

OPC Data Access servers are used for *vertical* information exchanging (server to client), while OPC Data eXchange servers are used for *horizontal* information exchange. This means that OPC DA is used between servers and clients and OPC DX is used between servers, shown in Figure 26.15.

Note that a OPC DX is an extended OPC DA server, and can be used as an OPC DA server as well. The usage of a data exchange server is configuration and data exchange between different systems (OPC servers) based on standard OPC specification and standards. A OPC DX server is an OPC DA server with DX extensions. This extension is:

- readable; a data source that can be read from an OPC DA server,
- connectable; a data destination, the available source at the OPC DX server,

A connection is made between the readable and the connectable, the connection is saved on the destination side, and it is the responsibility of the OPC DX server to read the value of the readable on the OPC DA and update the connectable value. See Figure 26.16.

26.7 OPC Historical Data Access (OPC HDA)

A PCS also need historical values, an OPC DA server is only handling the current process values. This means that an OPC server for data storage is needed and the solution is the OPC Historical Data Access (HDA) server. This server is a type of a DX server connected to the other DA servers in the system and storing the values. The HDA is using a integrated time series database for storing the process values.

Figure 26.16: The readbale / connectable connection between OPC DA and OPC DX servers.

The reasons/usages for a OPC HDA server are 1) Reading of historical values, 2) Tools for historical values and 3) Tools for database clients.

A HDA server is using an integrated database based on a time serie based database, and not a relational type of database (SQL). The functions of the HDA server are 1) reading and writing data for process and time series database, 2) access of the name space of the DA server (given that this server also contains DX support) (correct name should be HDX?), 3) historical values with attributes, timestamps and quality, 4) support for annotation and aggregation and 5) support for replay (playback) of historical values.

The HDA specification can give a range of extra server functionality from a simple server for reading of trend data only, to a complex server with a lot extra functions and services. Saving the process values includes several attributes like:

Type	Description
maximum time interval	a new value has to be saved after this time interval
minimum time interval	a new value shall NOT be saved during this time interval
exception deviation	minimum change for saving a new value
exception deviation type	absolute value, percent of new value, or percent value of value span (span = high entry limit - low entry limit)
high entry limit	the upper limit for a valid value
low entry limit	the lower limit for a valid value

The process values include the timestamps and either absolute time (reference is UTC;Universal Time Coordinated Time, same time as the old GMT) or relative time can be used when accessing the database. Time keywords are; NOW, YEAR, MONTH, WEEK, DAY, HOUR, MINUTE, SECOND and the syntax is "Keyword ± Offset" where the offset can be; Y, MO, W, D, H, M, S. A example of the timing in a daily report can be; Start=DAY-1D and Stop=DAY

Several services for additional information for the database information:

Annotation (Comments): 1) text, 2) username, 3) timestamp.

Aggregation: 1) Calculation only on request, 2) no saving of the calculated values, - Types: Average, Minimum, Maximum, Start, Stopp, Count, etc.

Playback: Playback of the historical data from the OPC HDA server to a client, 1) define the speed and duration, 2) define values and aggregation. Such a playback can be useful for testing, simulation, and teaching.

26.8 OPC Batch

Batch is the execution of a series of programs (jobs) on a computer without human interaction. A batch process is something between a continuous process and a discrete process, a discrete process running for only a fixed time. A batch consists of a set of operations as shown in Figure 26.17.

The definition of batch process (Furenes 2009): "Processes that lead to the production of finite quantities of material by subjecting quantities of input materials to a defined order of processing actions using one or more pieces of equipment"

Some characteristics of a batch process (Furenes 2009) as 1) Run intermittently to produce low-volume and high-value products, 2) Have dynamic nature of operation, no steady state, 3) Have finite time of operation, and 4) Frequent repetition of the same process. The reasons for the OPC batch server is that this server is based on the S88 standard for batch control (IEC 61512-1), and it is easy to configure and operate batch processes.

An overview of a batch process is shown in Figure 26.18. The process will have some input variables, output variables, measured variables, and manipulated variables. The run time of one batch and the

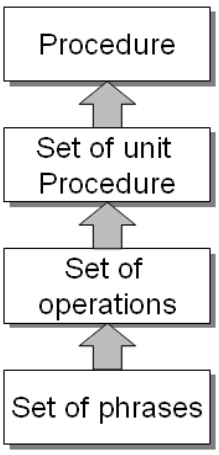


Figure 26.17: The contents of a batch procure.

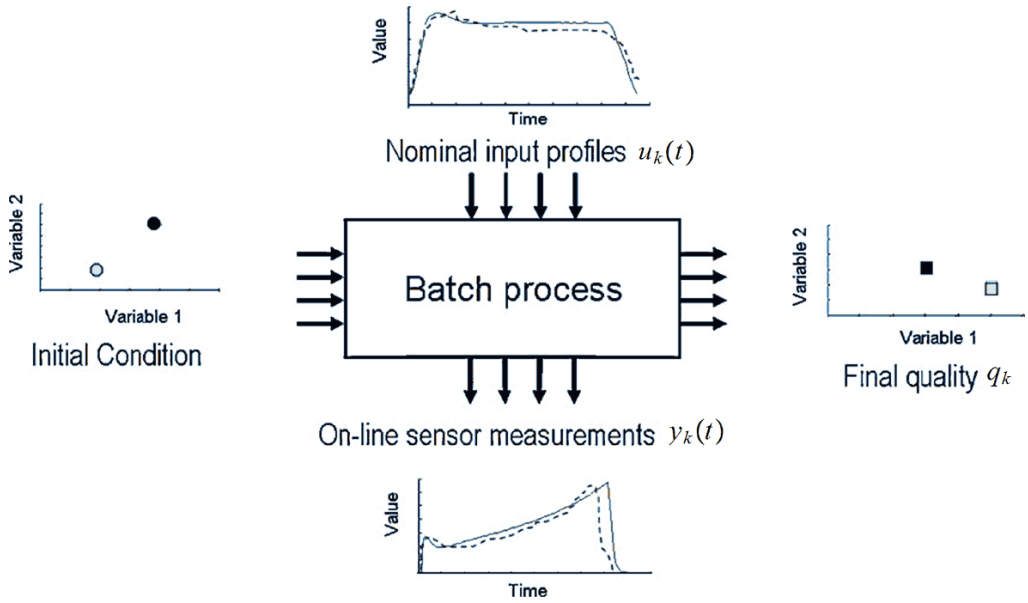


Figure 26.18: A batch process with input variables, output variables, measured variables, and manipulated variable (Furenes 2009).

batch run index will be two time variables.

The S88 standard contains: 1) Prescription handling, 2) Production planning, 3) Process control, and 4) Monitoring.

Examples of batch processes:

1. making a report; the process must collect a set of data, organize the data, make the report pages, and send the pages to the printing system,
2. baking a cake; the process is shown in Figure 26.19.

26.9 OPC Complex Data (OPC CD)

An OPC server handles as set of data, this can be any plant or process information, but seldom this information is a type of "stand-alone" information. From the DA server the information can be a tag og a group of tag, but in larger systems the need for more grouping of the data is necessary. The reason/usage for a OPC Complex Data server:

- Able to use more complex data types then OPC DA,

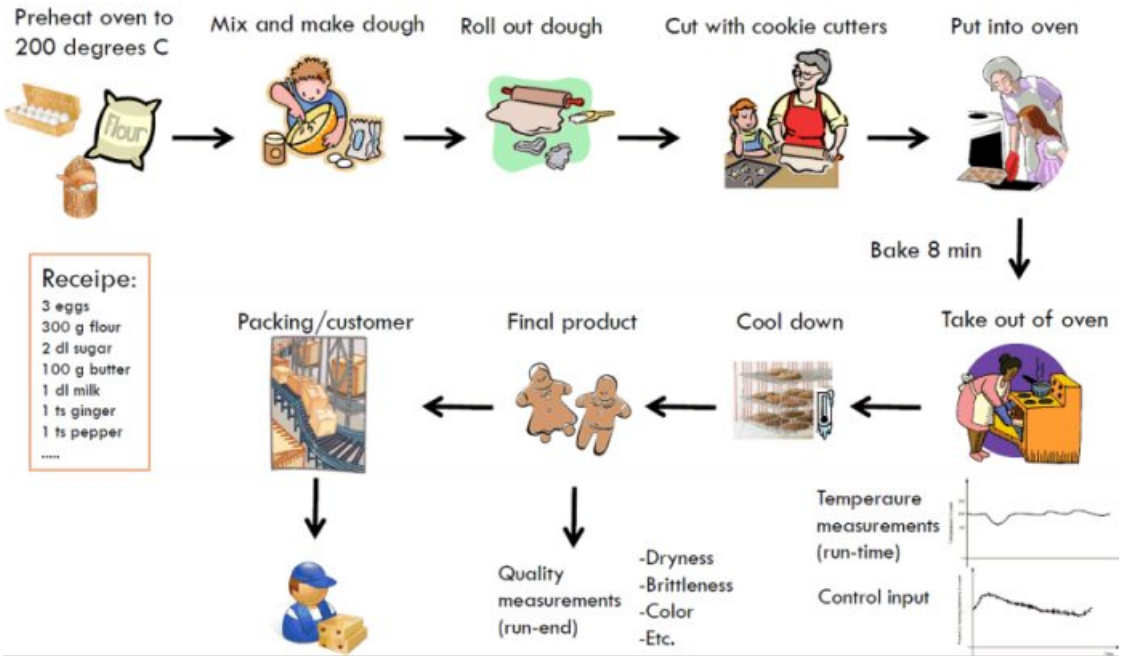


Figure 26.19: Baking a cake; an example of a batch process (Furenes 2009).

- Structure of simple items or complex items,
- The client should read both the structure and the values,
- Only extensions to the OPC Data Access.

The type of the complex data can consists of simple or complex items, and an unlimited number of nested levels. The structures of these items can be:

- arrays,
- structures (database records),
- arrays of structures,
- arrays and structures.

Figure 26.20 shows the usage of a OPC CD server to extend the OPC DA server with complex data structures for the OPC clients.

26.10 OPC Security

OPC is using COM and DCOM communication protocols and will depend on the services in these protocols. In addition has OPC access control of the tag information, group tag information and the configuration data. Each tag has both read and write access that can be controlled by the OPC security settings. The reasons for the security control are 1) control of the access of data in the system, 2) requirement for physical security of data, and 3) requirement for confidence of data. As the OPC is an open standard and that it is very easy to make an OPC client it is easy to access the data in an OPC system. Using a wireless network no physical connection is necessary when accessing the OPC protocol.

OPC is based on security in Windows:

1. User access in the system (principals)
2. User has to be member of a group (principals)
3. Access certificates,

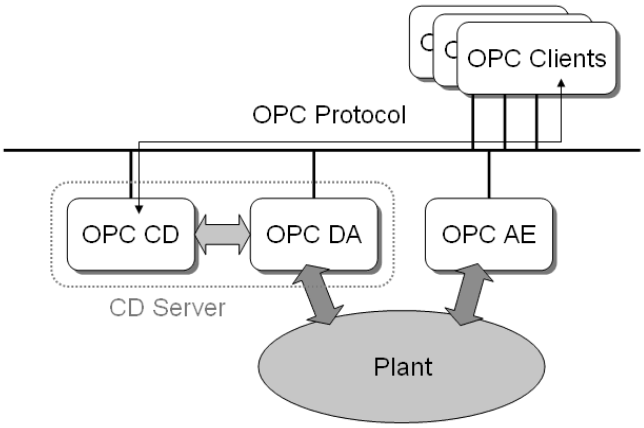


Figure 26.20: The OPC clients are using complex data structures when communication with the DA server. The DA server is extended with a CD server to support these complex data structures.

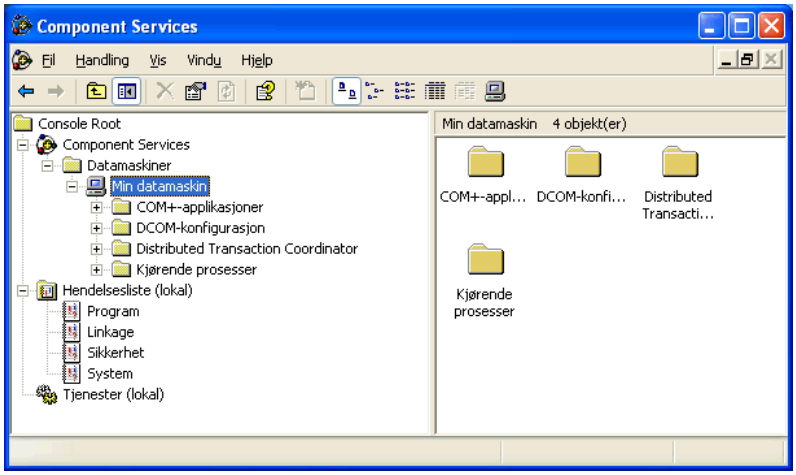


Figure 26.21: The DCOMcnfg.exe application.

4. Security objects,
5. Access control lists,
6. Reference monitor,
7. Communication channels,
8. Autorization,
9. Impersonation (being another principals).

COM/DCOM security objects:

1. the principal of the client must have access to the COM/DCOM objects,
2. using subscriptions the principal of the server must have access to the client,
3. using the application DCOMcnfg.exe for access configuration,
4. DCOM is using a dynamic range of ports making a challange for the firewalls,

The DCOMcnfg.exe application is shown in Figure 26.21.
Recommended security settings for OPC servers (Krogh 2005):

1. Authentication Level: Connect,

2. Impersonation Level: Identify,
3. OPC servers should be running on one specified account,
4. Use DCOMcnfg to allow users to access the servers,
5. Use DCOMcnfg to allow users to start the servers.

An OPC server can use 3 different levels of security:

1. No security,
2. DCOM security: security on users, no security access on objects, DCOM is using port 135 as a default starting point for the dynamic port range,
3. OPC security: No security or DCOM security, and security access on objects.

Using DCOM very often ends up with communication challanges as COM is an old technology and not suitable for todays solutions with firewalls. The reason is that DCOM is using a range of dynamic ports ending up with a ports to be open in the firewall. Another can be to use a "tunnel". A "tunnel" is a point to point connection between two computers are using a standard protocol like TCP/IP for communication on a "private" line. The "tunnel" can also include data encryption and data compression. OPC vendors often have their own OPC Tunneller solutions. The OPC tunneller will communicate using the local COM protocol on each computer.

COM/DCOM is based on binary documents, using an OPC XML-DA server can be an option in stead of the OPC DA server.

26.11 OPC XML-DA

An OPC DA server is using a binary document with the DCOM protocol ending up with a range of dynamic ports in the network. This can be a big challenge in a system with connection to intranet or internett (by the usage of a firewall), in stead a text document can be used. XML is a type text document where the binary information has been converted to text document with a special type of coding. The reasons/usages for a OPC XML Data Access server are 1) better integration of system not tight connected (different operating systems and/or different application domains), and 2) better usage of OPC over intranet or internet.

Extensible Markup Language (XML) is text based and has a set of rules for structured information. XML will focus on information, not presentation.

XML example:

```
<STUDENT>
        <NAME>
                <FIRST>     ABC         </FIRST>
                <MIDDLE>    DEF         </MIDDLE>
                <LAST>      GHI         </LAST>
        </NAME>
        <UNIVERSITY>
                <CODE>      TUC         </CODE>
                <WEB>       www.hit.no  </WEB>
        </UNIVERSITY>
</STUDENT>
```

SOAP (Simple Object Access Protocol) is a communication protocol which is based on XML, and OPC XML-DA is using SOAP.

The message sequence is shown in Figure 26.22. The operation using XML will be, as shown in the figure:

1. The OPC client is making a XML document (of input from the user),
2. Sending the XML document to the server,

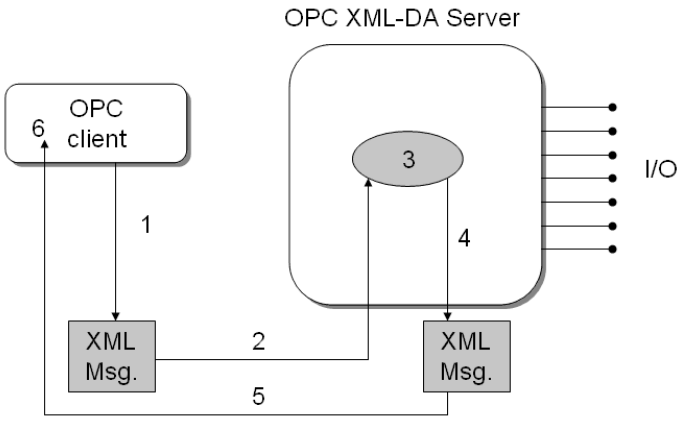


Figure 26.22: The message sequence in a XML-DA server.

3. The server extract the information from the document,
4. Making new information based on the client information,
5. Sending a new XML document back to the client,
6. The client will present the information for the user or extract the necessary information.

OPC XML can be used instead of DCOM based OPC Data Access as DCOM protocols may have problems with firewalls.

Will OPC XML replace DCOM based OPC?

1. SOAP/XML is basis for the new .NET communication technology,
2. SOAP/XML will be available on all new Microsoft operating systems,
3. Poor real-time support,
4. Overhead in the communication protocol.

26.12 OPC UA

The main challenges using OPC is the number of servers, the problem of DCOM deployment across different platforms, and the security.

In a process system the typical OPC servers needed are DA, HDA, and AE for all the hardware devices. In a large process system with a lot of different hardware devices several OPC servers of each type are needed. A new architecture with more functions in one server or a few servers are wanted. The platform used for communication in a distributed system is only DCOM so all the platforms need to support DCOM to let the OPC system communicate. The requirements for security is increasing being a problem for the OPC systems based on the DCOM technology.

The new architecture for OPC is Unified Architecture (UA), the first version was released in February 2009. The specification consists of 13 parts, and only the first part is free. The most important part is part 4. The total specification is of more then 1000 pages. These parts are:

- Kernel specifications; 1: Concepts; 2: Security Model; 3: Address Space Model; 4: Services; 5: Information Model; 6: Mappings; 7: Profiles.
- Access specifications; 8: Data Access; 9: Alarms and Conditions; 10: Programs; 11: Historical Access.
- Utility specifications; 12: Discovery; 13: Aggregates.

OPC Unified Architecture:

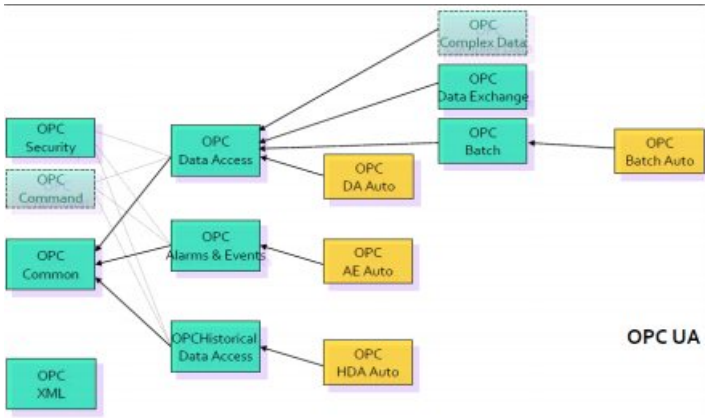


Figure 26.23: The relationship between the "classic" OPC and OPC UA (www.prediktor.no: oct−14).

1. using protocols based on XML and .NET technology from Microsoft. DCOM was based on DLL (Dynamic Link Library) given a lot of problems with different versions (DLL hell) and DCOM communications have problems through firewalls. DCOM is depending on a dynamic range of port addresses making it difficult to configure firewalls. Very often most of the security has to be switch off the let the OPC system work. OPC UA will be using only one port number that can be configured in the firewalls, as opposed to DCOM with need a dynmaic range of port numbers. Therefor DCOM is dufficult to use with security settings and impossible to use over internet.

2. The OPC UA server should be able to be implemented on any platform consisting of an ASCI C compiler and a TCP/IP stack,

3. with better integration between the different OPC specifications as the OPC-DA, the OPC-DX, the OPC-CD, the OPC-AE, and the OPC-HDA specifications. Normally a server will be installed for each of the specification, why not combing the specifications in fewer servers? UA wants to collect all the OPC server specifications into one UA server specification,

4. for focusing more on services, the OPC-UA will be more based on the *Service Oriented Architecture* (SOA) to focus more on services, not functions or data. The goal of SOA is to achieve loose couplings among interacting software agents.

5. for better integration on non-Microsoft systems, allow an easier integration of system not having a tight coupling. This includes systems like embedded systems, systems communicating over internet to mention some. The platforms can be among others Microsoft .NET, ANSI C, and Java implementations. The requirement for the platform is supporting an ANSI C compiler and a TCP/IP protocol stack.

6. based on information models using objects unlike the old OPC specifications with a flat data structure based on tags. OPC UA has defined four standard information models as 1) Data Access (DA), 2) Alarm and Conditions (AC), 3) Historical Data Access (HA), and 4) support for presentations from applications, start and stop of applications, execution status and execution results of applications. Figure 26.23 shows the relationship between the OPC "classic" solution and what will be covered by the OPC UA standard. Note however that OPC UA contains different type of profiles defining the type of functionality. Vendors can also use these profiles to make their own information models. Figure 26.23 shows that OPC UA contains the following OPC "classic" specifications: OPC XML, OPC Commin, OPC CD (Complex Data), OPC DX (Data Exchange) and OPC Batch in addition to the OPC DA, OPC AE and OPC HDA.

7. is more internet friendly than OPC "classic" specifications. OPC "classic" is based on the DCOM protocol using several dynamic port numbers in the TCP/IP protocol making it almost impossible to configure a firewall. OPC UA is using only one fixed port number in the TCP/IP protocol making firewall configuration much more easy, enabling internet communication.

Figure 26.24 indicate the layered model layer for OPC UA showing the base services, information models and vendor specific models.

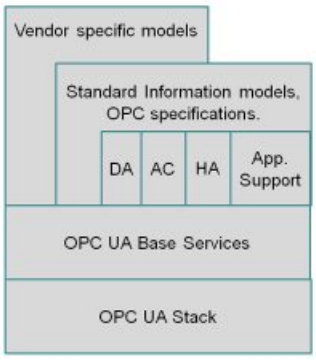


Figure 26.24: An overview of the layered model for OPC UA.

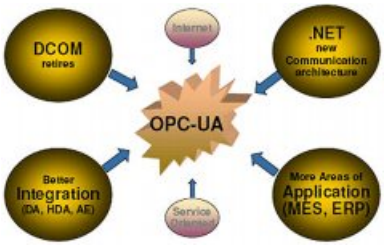


Figure 26.25: The contents of the OPC-UA specification. The specification contains a new comunication standard and has a better integration of the OPC-DA, the OPC-HDA, the OPC-AE, the OPC-CD, and the OPC-DX specifications (www.opcfoundation.com: jan-09).

Figure 26.25 shows an overview of the OPC-UA specification.

There are three primary factors that influenced the decision for moving forward with the OPC UA architecture (www.matrikon.com 2007):

1. The major OPC installation base, Microsoft is focusing their future efforts on Web Services and SOA applications. In addition, there is increasing pressure from end users looking for OPC support on Linux and other non-Windows platforms.

2. OPC is no longer a simple point-to-point solution, and is becoming the backbone of increasing complex OPC architectures, that involve multiple specifications. Vendors and users require a single interface that exposes the key functional areas of OPC.

3. The OPC Foundation would get more and more requests from clients and other institutions to leverage its standards to aid those who are defining other industry standards at a granularity below the interface level.

26.12.1 Service Oriented Architecture

SOA departs from object oriented programming where the data and the processing of the data is bind together in objects. In SOA the work is done by a service provider on a request from a service consumer, abstract actions for the servers based on requests from the clients.

The main guideline principles for UA (Murphy 2009):

- a relative small set of simple interfaces to all participating software applications,

- the messages are written in a format, structure, and vocabulary that are understood by all OPC UA applications,

- easy to make extensions to the services.

26.12.2 Security

DCOM cause a lot of challenges regarding the security of a process system. The OPC UA gives a infrastructure with multiple tiers of security implementation. The type of security required for a specific application will define which tier that need to be used. The tiers are (Murphy 2009):

- Small Ad-hoc Network Tier; define which OPC application to trust in the network (trust list),
- Normal Corporate Network Tier; define which applications in the corporative network that can be trusted by the OPC UA servers (trust list),
- High Security Tier; use a combination of local trust lists and certificates administrated by the administrator of the system,
- Anonymous Web Client Tier; using usernames and passwords.

OPC UA will not simply replace existing OPC standards, but rather complements them by providing a common interoperability layer for exchanging information and a richer environment for orchestrating operations (Murphy 2009). OPC UA has however maintained backward compatibility with previous OPC standards to ensure quick adoption in the market (Murphy 2009). OPC UA will use several protocols like UA XML and UA Binary. UA XML will use port 80 while UA Binary will use port 4840 as the default port. Both protocols are using only one port in the firewall, and the port can be configured at a fixed address. The binary protocol is faster than the XML protocol, which is text based. Both protocol are based on TCP/IP instead of the DCOM technology. Both protocols are using sertificates for a secure communication.

26.12.3 MDIS network

The MCS-DCS Interface Standardization (MDIS) network selected OPC UA as the Master Control System (MCS) and Distributed Control System (DCS) protocol standard. The MDIS network consists of many important actors within the subsea controls area, topsides specialists and oil companies. A standard interface, based on the OPC UA standard, will be developed for communication between the Master Control System (MCS) and Distributed Control System (DCS) on topside systems. The selection of OPC UA will provide secure and reliable data transfer for offshore oil and gas production. The MDIS network selected OPC UA based on several requirements, some of these requirements were: 1) MDIS Vendor Support, 2) Communication Integrity, 3) Robust Test Tool, 4) Communication Integrity, 5) Reliable Implementation, 6) Redundancy and Robustness, and 7) Independent Organization Support.

26.13 IoT and IIoT

Internet of Things (IoT) and Industrial Internet of Things are devices connected to an intranet or internet, and using TCP/IP for communication. TCP/IP is only supporting up to layer five in the OSI model meaning that it is not defined how to get the information out of the device. Normally a cloud service must be used for getting the information from these devices, and/or a specific API must be used. Including a protocol supporting layer seven in the OSI model means that any application can communicate directly with the IoT or IIoT devices and some work has been done to define such a protocol. Some of these protocols are CoAP (Constrained Application Protocol), MQTT (Message Queue Telemetry Transport), and XMPP (Extensible Messaging and Presence Protocol). The OPC UA has also been suggested for such a protocol.

Communicating directly with the IoT device means that the IoT device needs to have more processing power and memory to handle communication with several (or many) nodes, instead of communication with only one cloud service and let the cloud service handle the communication with all other nodes. The system architecture will affect the type of IoT devices that can be used, and the memory and power needed for the IoT device.

This is shown in Figure 26.26 with a set of IoT or IIoT devices. Normally one cloud service is needed for each of the devices, and any information from the IoT or IIoT devices must be done by communication with the cloud services. These cloud services will have different communication protocols like CoAP, MQTT or XMPP, or even proprietary protocols for communication, and these protocols must be supported by the PCS.

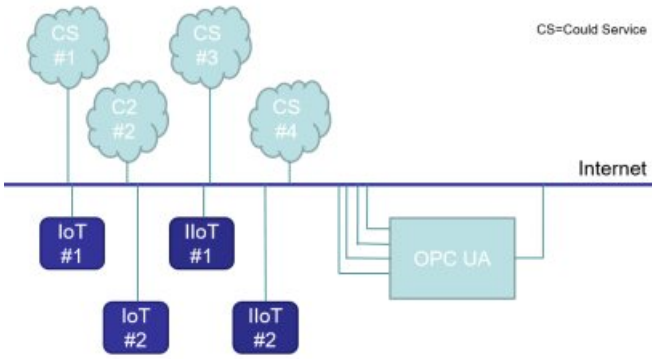


Figure 26.26: IoT (Internet of Things) and Industrial Internet of Things (IIoT) require some support in layer 7 of the OSI model, often with a cloud service. This is shown to the left of the figure, four IoT or IIoT devices and four cloud services that must be used to get information from the IOT or IIoT devices. These cload services will have different protocols or API for comminication. Another solution can be to use an OPC UA server communication with all the IoT and IIoT devices, and use only the OPC UA protocol for communication with the IoT or IIoT devices.

	MQTT	{REST:API}	OPC UA
Standardization	Low	Medium	High
Footprint	Lightweight	Medium	Extensive
Security	External	Partly Integrated	Integrated

Figure 26.27: Comparison of the MQTT, Rest API and OPC UA as the communication protocol for IoT devices (Pepperl&Fuchs - NOV-20).

Instead an OPC UA server can be used, and the OPC UA server will have the necessary protocols implemented, either to communicate with the could services, or even better, directly with the IoT or IIoT devices. This way will the PCS only need to use the OPC UA protocol for communicating with the IoT or IIoT devices. And any future extension will also work as only the OPC UA server must be extended to include any new protocol for communication with any new IoT or IIoT device.

The system architecture should decide for the implementation of communication for IoT devices. IoT devices are selected when collection information across different lokal networks (or a Fieldbus can be used), and some sort of communication protocol must be selected for communicating with the IoT devices. The footprint (speed and amount of information) is defining the MQTT is a light-weight protocol, Rest API is a medium protocol and OPC UA as an extensive protocol, as shown in Figure 26.27.

This figure shows also that the standardization of the same protocols is low for MQTT means that more configration and added software modules are needed for using this protocol, the Rest API is medium and OPC UA is high. High standarization means that most of the support is already included so adding new IoT deviced is just a matter of system configuration. Security is also an issue today, and the figure shows how security is included into these protocols.

26.14 OPC Xi

OPC UA is a large and complex specification and has shown to be too complex (or too time consuming) for many vendors. A light version of the OPC UA, OPC Xi (Express Interface), has been specified. OPC Xi consists of a simple data model with .NET interface supporting combination of the OPC DA, OPC AE, and OPC HDA specification.

Chapter 27

OPC systems

27.1 Introduction

OPC consists today of a lot of standards and each of the standards are implemented as a server. An OPC system will therefor consists of a set of servers having the data and a set of clients using the data. These servers can be installed on one or several computers depending on the structure of the SCADA system. The important aspect here is that every software module that should be integrated into this SCADA system must support the OPC protocol.

The implementation of the OPC protocol will depends on the functionality of the client or the server. Normally is the implementation of the OPC in the client less time consuming than the implementation on the servers.

27.2 Why use OPC

1. Using standard network technology; 1) proven, 2) good performance, 3) using existing networks (only layer 7 in the OSI model), 4) cost, 5) knowledge, and 6) wire and wireless.

2. Open communication standard; 1) independent, 2) many suppliers, and 3) good performance.

3. Disadvantages; 1) several server specifications (at least for classic OPC, better for OPC UA), 2) all devices and software modules MUST support OPC, 3) network communication using DCOM for classic OPC.

27.3 OPC development

Two main reasons for developing your own client or server is control (of protocol or hardware) and/or performance. It is easy to develop a client, from a couple of hours and up depending on the functionality of the client and the available tool kits. A server is much more demanding, almost a year (from 3 to 12 months) depending on the knowledge of the specifications. The servers should also be approved by the OPC foundation.

The type of knowledge (Krogh 2005) for making an OPC server or client:

1. OPC specifications,

2. Windows security,

3. OOP (Object Oriented Programming),

4. Developing tools,

5. COM/DCOM and XML for OPC Classic, TCP/IP and .NET for OPC UA,

6. OPC tool kits.

27.4 OPC test

The OPC system should be tested before installing into a real plant. The best way of testing such a system is to use any simulation modes of the server if the hardware is not available. Most OPC servers have a simulation mode. The appendix shows the usage of a freeware OPC server from Matrikon[1] in simulation mode. This server has a limited number of variables but can be used for smaller tests.

[1] Matrikon: See web page www.matrikon.com.

Part IX

Data Security

Chapter 28

Computer Security

28.1 Introduction

Computer security, data security and cyber security are terms used for methods protecting computer systems from unauthorized access, use, change, disclosure and destruction and includes network security, physical security, and file security. Computer security has more focus on the physical devices, data security has more focus on the information stored on the computer systems and cyber security has more focus on preventing, detecting, and responding to cyber attacks. Cyber attacks are attacks from other computers, either in the intranet or internet. Information security is closely aligned with the term data security. Computer security affect several areas and can be anything that can destroy your computer systems, either the computer system itself, access to the computer system, or information related to the computer system. This includes fire, explosion, power loss, theft, user error and cyber attack to mention the most normal reasons. So security threats affect both the location where the computer system is used, the environment of this location, the connection to external devices and network, and the computer system itself. Note that external devices can also be USB disks (memory disks), not only connections to other computer systems. Figure 28.1 shows many of these security threats.

People often think of attacks only when talking about threats, and by attacks access from a remote computer system into the local computer system. These attacks are only a part of all possible threats, the most normal attacks are 1) physical attacks by internal or external personnel, 2) software attacks by installing some type of malicious code, 3) communication attacks by access from a remote system, or 4) upgrade attacks by installing malware or modifying the boot sequence. The attacks towards an information system can be of many different kinds and the reasons can be several, however they can be grouped in two different types. These types are 1) access to a computer system to change, copy or steal data or 2) access to a computer system to use the system in criminal activity.

An overview of the security protection of a computer system can be done by the security triangle and security map, based on the concepts for security protections. A security map gives an overview of the information security in the system and the security triangle gives an indication of the current security

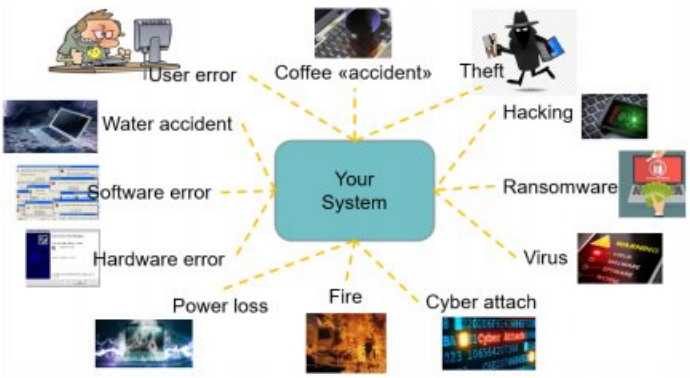


Figure 28.1: Many of the security threats that can affect a computer system, from physical threats, user threats and to cyber threats.

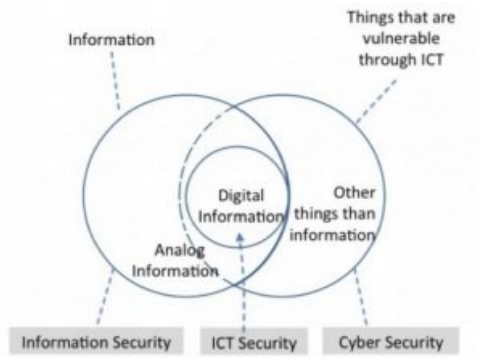


Figure 28.2: The relationship between information security and cyber security (www.ccis.no: Nov-19).

level for the system.

28.2 Information security

Information security is part of computer security with focus on the information in the computer. Information security involves computer security and cyber security and Figure 28.2 shows how information security affects both analog and digital information. Analog data is represented in a continuous form while digital data is represented as discrete values. Digital information is information available in the network and will be affected by both cyber security and information and communications technology (ICT) security.

The concepts for security protections are 1) confidentiality, 2) integrity or data contamination and 3) accessibility, and information security is protection against violation of these concepts (Daler, Gulbrandsen, Høie & Sjølstad 2019). It is important to define who is the owner of the information, the access rights for each part of the information, and the type of protection that is necessary for each part of the information. In systems today the information represents large value or asset and should be treated as a valuable resource for any business. The security protection concepts are often combined in the security triangle (Assev & Mikalsen 2012).

28.2.1 Security triangle (triad)

In a network will a connected computer system not be 100% secure as the computer must be able to communicate with other resources in the network. The security triangle is one way to define the security level of the computers connected in a network, based on the concepts for security protections. The information available in a network must be classified to be able to protect this information with the right methods and levels. These levels can be 1) open, 2) internal (or protected), 3) confidential, and 4) strictly confidential. These levels of information protection will affect the security system. The security triangle consists of the three concepts confidentiality, integrity, and accessibility, also known as the CIA triad: 1) confidentiality; hide "secret" information in the network, 2) integrity; be sure that the information is "correct", be sure of the source for the information, and 3) accessibility; define which department/employee that can access which information in the network (create, read, write and delete access).

These elements of the CIA triangle are considered the three most important components of security, making a security model. The security triangle is shown in Figure 28.3 with an indication that the accessibility is good for the company, the integrity is not that good, while some more efforts should be put into the confidentiality component.

The accessibility is divided into groups and users where the resources is basis for the groups, and the users needing this type of information are linked to the groups. A user can be linked to as many groups as necessary. Comments regarding groups and user; 1) The resources in a computer system or network system are allocated to users or groups, 2) The user account is personal, and linked to specific person, 3) The user belongs to a group with group access to the resources of the system like file, directories, printers, other computers to mention some, 4) Most computer systems give restriction for the user, not the data (information), and 5) the access is often grouped in create, read, write and delete.

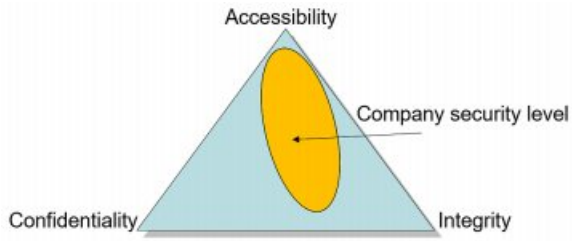


Figure 28.3: Use the security triangle to define the security level (Assev & Mikalsen 2012), based on the three most important components of security. The figure will indicate the security level for a specific company or business, in this case with less focus on the confidentiality component.

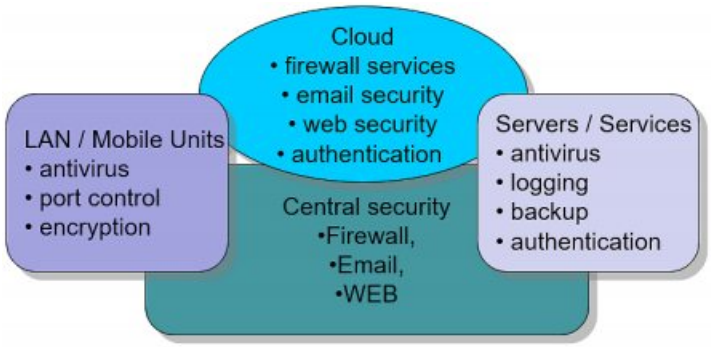


Figure 28.4: The areas for security solutions for a business, including mobile devices and cloud computer.

A security map is one way to get an overview of the information system, with focus on security, for making a security risk analysis and evaluation of the business.

28.2.2 Security map

A security map gives an overview of the areas where security solutions are necessary. The areas will be to secure the information and/or information systems against both internal and external attackers. Figure 28.4 shows an overview of the systems and connection points for a business.

Security can also be about physical and logical security like:

1. Physical Security; 1) prevent physical access like door lock on the rooms with the computers (servers), alarm systems, detectors in the computer rooms, and access restriction in different sections of the buildings, 2) maintenance of the computer equipment, and 3) uninterrupted power supply (UPS), battery backup of several minutes for the main computer systems,

2. Logical security; 1) password strategies; regular change of the password, number of letters in the password, number of old password to remember, 2) limitation in the number concurrent users of the systems, 3) limitation of which workstations an employee can use, 4) time restriction, how long can the employee be logged into the system, 5) automatic locking or logging out when no activity on the system, 6) file security; setting the read, write, and create privileges of a file, 7) backup strategies; how often to do the backup and which data to backup. Backup strategies can be *a*) full backup; backup of all files, *b*) incremental backup; backup of only the files changed since last backup, *c*) differentials backup; backup of only the files changed since last full backup, and 8) firewalls; filter the traffic, filter the services, hide computers in the intranet.

3. Security applications and services like 1) antivirus; avoid harmful software to get control or be installed on the information systems, 2) anti spam; avoid spam to enter the email system, about 70% of the emails are spam and will cost the business time, cpu time, and disk space, 3) firewall; a security level between internet and intranet, 4) authentication; add an extra level to get access to the information, simple passwords are no longer good enough, and 5) encryption; email and disk, requires some control of the encryption passwords.

Note that firewalls are software services, and that these services can either run on an external device or on your computer.

28.2.3 Risk management

All businesses should have some sort of active safety management with regular risk and impact assessment and risk management. The following steps should be performed to increase the security level of a computer or network system:

1. perform risk analysis; list unwanted events that can arise, the probability for these events, the scope of these events, and the type actions to reduce the impact of these events,
2. security education; educate the employees to think security in every action. Define a security policy for the company; a document describing the security levels in the company,
3. make a security culture including everybody in the company,
4. perform frequent security tests of the system; a) inside test to check the security functions and settings for the intranet, b) outside test by a person outside the company trying to make harm to the system, c) application testing, d) physical security check, e) security check of the wireless network.
5. update the systems with patches; this is important as attackers will search networks for servers and other devices with old versions of operating systems and applications with known errors that can be used for access.

Some more practical security issues for protecting your computer systems:

- Login access; normally the user name and password model. The user name is often the same code as the email and can easily be found given that the password is the important access code.
  - Passwords; select "strong" password meaning that it should be difficult to guess the password. Any personal information can be retrieved from information on sites like Facebook and Instagram to help the attacker to guess your password. The list of "worst" passwords the last years were 1) 123456, 2) password, 3) 12345, 4) 12345678, and 5) qwerty. Let the password be a combination of letters, numbers and special characters of at least eight characters so any automatic password decrypting system will use too long time to test all possible combinations,
  - Changing password; change the password on regular basis especially if you are using the same password on different systems and web sites,
  - Multi-factor authentication (MFA) or multi-step verification; adds another layer of security, supplementing the user name and password model with a code that only a specific user has access to (typically sending a code to the mobile phone),
- Antivirus protection; Install and use anti virus protection software, and keep the software up to date,
- Firewall; A firewall consists of a set of rules to define the communication traffic in and out of your system. Prefer a hardware firewall over a software firewall. Both a firewall and anti virus protection is needed in a system,
- Updates; Use automatic updates to always get the latest patches,
- Encryption of stored data; evaluate the use of encryption on stored data on important information,
- Permissions; use a clear permission policy in the company, and update this policy on regular basis,
- Wireless network; use WPA as a minimum on wireless connections, prefer WPA2 as the encryption standard. WPA2 can also be cracked, but takes much longer time then other encryption standards,
- Mobile devices; use a password requirement for mobile devices to connect to the company system,
- Backups; make a backup procedure and run regular testing of the backups.

An example of a risk reduction action plan for computer security is shown in Figure 28.5 showing threats, actions and measures for internal and external intruders.

Risk Reduction Action Plan

Threat	Actions	Measures
Insider	? Background Checks ? Training – Everyone, IT, HR, Leadership ? Remove Access Promptly	Regular Exception Reports
External Hacker	? Patches to Keep Software Updated ? Anti-Virus for Known Malware ? Limited Administrative Rights ? Two-factor Authentication	Regular Time Delay Reports and Rights Reviews
Successful Intrusion	? Certified IT Professionals ? Access Log Reviews ? Intrusion Detection Software ? Exfiltration Software ? "White-listing" for Control Systems	Frequent (Daily?) Results Reports
Successful Attack	? "Loss of IT" Business Continuity Exercises ? Engage/Develop Forensic Capability	Exercise Frequency and Results

Figure 28.5: An example of a risk reduction action plan for computer security (www.wpi.edu: oct-19).

28.2.4 Backup

The information is stored in data files on some sort of data storage that are easy to access. Backup of the information, data backup, refers to saving additional copies of the information in separate physical or virtual locations. It is important to backup the information on regular basis and store the backup in a separate location. Keeping reliable backups is an important part of data management. Regular backups protect against the risk of damage or loss due to hardware failure, software or media faults, viruses or hacking, power failure, or even human errors. Part of the data management is backup procedures making regular backups and regular checking of the backup copies by restoring on an unused storage device.

It is important to protect the backup system in an intranet system as malware software affecting the computer software may also affect the backup, especially online backup system. In such cases will a firewall setup separating the backup system and the backup user in the intranet system. For an infected computer system the system must be reinstalled and the backup for restoring the information. Any infected backup can not be restored if the computer system is hacked. In special cases may also the UEFI (Unified Extensible Firmware Interface) of the computer be changed and must be restored to an original version. The UEFI is the BIOS (Basic Input and Output System) of a computer system handling the startup of the computer, from turning on the power until an operating system (OS) is loaded.

28.3 Grading of information

Companies should have a good overview of the information needed and the value of this information. Based on this overview the company knows which part of the information that represents a high value and which part that is less valuable. The company should focus more on protection for the information that is valuable. It is not possible to protect the information 100% but the company should try to grade the importance of the information based on value. There exists different types of systems (Daler et al. 2019) for grading of information, an often used method is to divided the information into three categories like 1) unclassified, 2) protection level#1 and 2) protection level#2. Unclassified information is information open to public. Protection level#1 has partly protection of information and apply to information that do not need to be available for all employers in a company. This protection level is also known as confidential. Protection level #2 is for the information that is of a high value for the company and should be available only for a limit number of employers. This protection level is also known as secret.

A company must have knowledge about the type of information used and where this information is stored. It is also important that the information is not duplicated into several locations and that the company has backup procedures for making copies of the information. Duplication must be avoided because any changes or updates of the information must also be done on all duplicates. Information security, or data security, is often thought about as mainly preventing hackers from accessing the information, however information security involves 1) network errors/downtime, 2) loose of data and handling backups, 3) misuse of data, 4) operations inside the company by employees, and 5) operations outside the company by hackers.

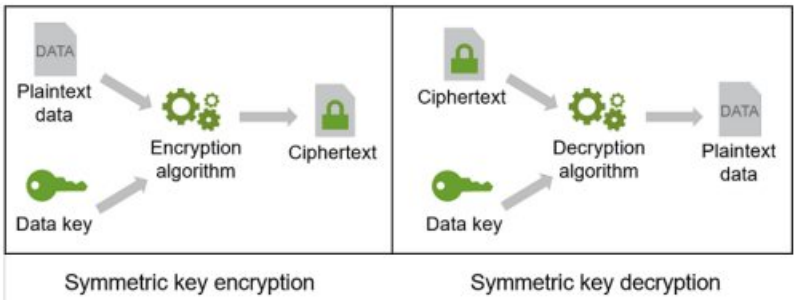


Figure 28.6: The processes of encryption and decryption using an algorithm to convert between plaintext (cleartext) and ciphertext (docs.aws.amazon.com - Nov-19).

Securing the information can be done in several ways, especially the most important information. Some of these methods are: 1) data encryption; converting the data into a code that cannot be easily read without a key that unlocks it, 2) data masking; masking certain areas of data so personnel without the required authorization cannot look at it, 3) data erasure; ensuring that no longer used data is completely removed and cannot be recovered by unauthorized people and 4) data backup; creating copies of data so it can be recovered if the original copy is lost. Deleting a file in an operating system will normally only remove the link in the directory structure of the disk, the data will still be present in sectors on the disk. A proper delete file tool will also delete all the sectors, overwriting these sectors with random information.

Note that the General Data Protection Regulation (GDPR) is very strict on that a company must have control of the information used and that any non-conformity or data leakage can be punished with large penalties.

28.3.1 General Data Protection Regulation (GDPR)

The General Data Protection Regulation (GDPR) is a regulation protecting the data of citizens within the European Union and other countries in Europe including Norway. It will affect every company making business within these countries. This regulation is protecting the information about the citizens that every computer system is storing, saying that the owner of the computer system should protect the data, know the type of data stored, and be able to delete all the data whenever a specific citizen request to be removed from a specific system. The type of information that is protected by GDPR is name, photo, email addresses, social media posts, personal medical information, IP addresses, bank details to mention some (Daler et al. 2019). The GDPR covers any information that can be classified as personal details or that can be used to determine the identity of a citizen. Parental consent will be required to process any data relating to children younger than 16.

28.3.2 Encryption

Encryption is the process of translating plain text data into something that appears to be random and/or meaningless type of information, also called ciphertext. This process can be performed on information that is going to stored on a storage medium or on information that is going to be transmitted in the network, both in the intranet and in Internet. Some sort of rules must be used in the encryption process to be able to convert the ciphertext back into the same plain text. Decryption is this process, converting ciphertext back into the plaintext. Figure 28.6 shows the processes of encryption and decryption. The quality of the encrypted message depends on the strength of the encryption algorithm and the secrecy of the encryption key.

There exist two main types of encryption algorithms, symmetric and asymmetric. In the symmetric algorithm will the same key be used for both the encryption process and the decryption process. One simple symmetric algorithm is the substitution algorithm where every character in a text is replaced with another character. The key is a character substitution table. The first used symmetric encryption algorithm for public use was the Data Encryption Standard (DES) in 1977 with a key length of 64 bits. 64 bits are too short and the algorithm was replaced by the Advanced Encryption Standard (AES) with key length of 128, 192 or 256 bits. Both DES and AES is symmetric encryption algorithms and is based on only one key. The goal of every encryption algorithm is to make it as difficult as possible to decrypt

the generated ciphertext without knowing the key.

Asymmetric algorithms are based on two keys, one public key and one private key. The encryption process is using the public key meaning that everybody can perform the encryption process. The decryption process can only be performed with the private key, so only the owner of the private key can decrypt the text. The RSA crypto system (developed by Rivest, Shamir and Adleman in the 80's) is one of the most used asymmetric algorithm and is widely used for secure data transmission.

In the beginning of the 1990s Philip Zimmermann (at Florida Atlantic University) developed an encryption and decryption algorithm for communicate and store files in open networks more securely. The method is called *Pretty Good Privacy* (PGP) and can be used without involving any official authorities, using a type of symmetric algorithm. PGP can be used for encrypting and decrypting texts, emails, files, directories, and even whole disk partitions. PGP is independent of the operating system and is available in many different software modules and libraries like OpenPGP[1], Gpg4win[2] and PGPCore. The PGP algorithm is using a random generated one-off key for encryption of the data, known as the session key. This type of encryption is very efficient, but the challenge is the session key. PGP solves this challenge with public key cryptography, also known as asymmetric cryptography. Public keys can be found on key servers or by asking the person directly. Public keys are normally used by the sender to encrypt data, but they can not decrypt the data. Normally only the private key can decrypt the data. In PGP the public key is used to encrypt the session key only, not the data. Encrypt the data with the public key uses a larger amount of computational resources and takes longer time than using the session key. The receiver needs to receive both the encrypted data, encrypted with the session key, and the encrypted session key, encrypted with the public key. Only the private key of the receiver can decrypt the session key, and only the decrypted session key can be used for decrypt the data.

Note that there has been several versions and features of the PGP algorithm over the years, meaning that older versions of PGP algorithms will not be able to decrypt messages that has been encrypted by newer versions. Both the sender and the receiver must be aware of which version and features that is used for decrypting the message. The PGP algorithm is not perfect but is known to be one of the safer ways to communicate online.

28.3.3 Block chains and Bitcoins

Block chains are a special type of distributed databases containing a set of blocks (posts) connected in a chain with a crypted connection between each block. Each block contains a cryptically hash value of the previous block in the chain, a time stamp and transaction information (Mazonka 2016). Block chains can be used for digital descriptions of a long set of transaction where none of the blocks can be changed without changing all subsequent blocks, as these blocks contains a cryptically hash value of the previous block. The original block chain was the database for the electronic currency Bitcoin back in 2009, but are today also used for many other type of applications. The encryption process is based on an asymmetric algorithm, using two keys, one private key and one public key. A block chain can exists over several computers.

The electronic currency Bitcoin was developed by Satoshi Nakamoto in 2008 and 2009 (released $3 - JAN - 09$) and consists of digital information that can not be copied. Normally digital information is easy to copy, while Bitcoin is impossible to copy. So a specific Bitcoin can not be copied, the specific Bitcoin exists only as information in one specific location. The owner of this information is also the owner of this specific Bitcoin. The Bitcoin information is a number consisting of 64 numbers and the maximum number of Bitcoins is 21 millions. The number of Bitcoins is 17 millions (MAR-18) and a number of Bitcoins is released at fixed time intervals for Bitcoin systems that have enough cpu power to "mine" this currency, often named Bitcoin mining.

28.4 Network

Internet is based on an interconnection of a huge amount of intranets from companies, private organizations, public and private persons. The computer systems in these intranet must be protected from illegal access and this is the purpose of the firewall. Figure 28.7 shows an overview of internet with several intranets connected using a network device. This network device will act as the gateway, router and firewall for the intranets.

[1] www.openpgp.org: Nov-2019
[2] www.gpg4win.de: Nov-2019

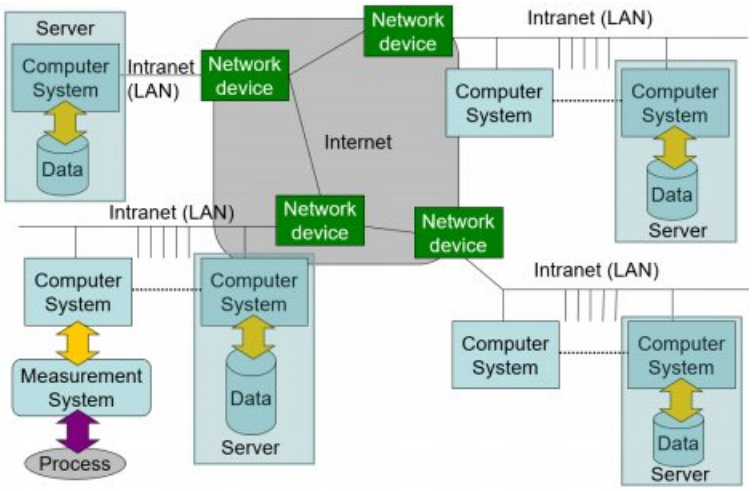


Figure 28.7: The intranet and internet interconnections with the network devices as connection points between the intranets and internet.

The gateway is the device that joins together two different networks, in this case the intranet and Internet. The router is the device that decides where the packets are sent based on the IP address. The firewall is a module (or filter) that examines the packets against a set of defined rules in order to decide whether to allow the packets to pass through between the intranet and Internet.

28.4.1 Firewall

Firewall is either a computer device with firewall specific software or a software module installed on a standard computer in the intranet. A separate unit is more secure than a software module as the separate unit has been designed with security in mind for both the operating system and the software running on the unit. A software unit installed on a computer is only an extra software module in addition to the network system and the operating system. A company needs several layers of protection, a low level of protection for the web servers, one higher level for most of the users and a high level of protection for the servers. The web server should be available for all users on internet but only for read access, only a few persons should have access to change the contents of the web site. The intranet should only be available for the persons related to the company but both local and remote access should be possible. The grading of the information should decide who should have access to the servers. The lowest level of protection, for web server, is known as a demilitary zone (DMZ). An example with three firewalls is shown in Figure 28.8.

Note that the production network can be part of the intranet, however in a production or process control system (PCS) or in an industrial computer system (ICS) the production network is often the operation technology (OT) devices and the intranet is the information technology (IT) devices.

The firewall is a security system protection a computer system from an external network and the main tasks are:

- Access control for users and customers,
- Access control for the wanted service,
- Authentication if possible,
- Logging of activities.

More advanced firewalls also have functions for 1) virtual private network (VPN); a more secure point to point connection on internet, 2) checking of "buffer overflow"; checking the maximum size of input requests or messages, 3) outbound protection; checking for illegal output messages, and other functions.

28.4.2 Internet

Internet is based on the TCP/IP protocol and this protocol is based on either the TCP or UDP protocol. Each computer on Internet has a specific Internet address (IPv4 and/or IPv6) meaning that each

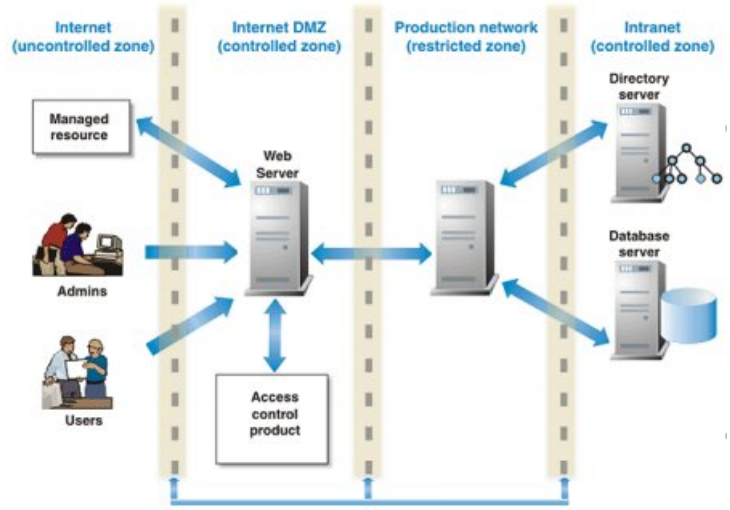


Figure 28.8: An overview of an intranet with three different protection levels of security, by firewalls. One DMZ area for web servers, one production network (which can also be aprt of the intranet) and the information servers in the intranet.

computer on Internet can communicate directly with any computer connected to Internet. Computers on Internet can not have the same IP address, each computer needs to have a specific address given by the organization handling Internet, and handled by the Internet service providers. There exists a set of private IP ranges that can be used in the intranets, the gateway will convert these private addresses to a global address given by the Internet service provider. These connections consist of the protocol type (TCP or UDP) and a specific protocol port. The maximum number of ports are 65535 for each protocol type. This means that every computer connected to internet has at least 65535 x 2 ports (or doors) that any other computer can use to connect to that computer. How many of these ports (doors) are locked or controlled on your computer system? The control of these ports is the responsibility of the network module in the operating system, however the firewall can also check for port access. Normally the operating system will lock most of these ports. Several protocols (often in layer seven of the OSI model) are using specific port numbers and these port numbers have to be open. A few examples of port numbers and protocol types are:

Port number	**Protocol type**	**TCP/UDP Type**	**Note**
20	FTP	TCP and UDP	File Transfer Protocol
23	TelNet	TCP and UDP	
57	Mail	TCP	
80	HTTP	TCP	WEB pages
118	SQL	TCP or UDP	
123	NTP	UDP	Network Time Protocol

The firewall can add extra rules to open ports and do some checks of the inbound and outbound messages on these ports.

The TCP/IP messages, both TCP and UDP contains the logical address of the receiving node and the transmitting node. This means that every node along the path of the message can check the message; the contents, the receiving node address and the transmitting node address. It is possible to encrypt the contents, but not the message header.

28.4.3 Internet of Things (IoT)

Internet of things (IoT) are devices connected to a TCP/IP network, either the intranet or Internet. One challenge with IoT is when connecting the IoT device on Internet meaning that the IoT device will be outside the network device between the intranet and Internet, and then outside the firewall. A IoT device consists of a small computer with network support meaning that anybody can try to connect to these devices. These devices support the TCP/IP protocol, in order to be an IoT device, meaning that it is possible to use a rlogin (remote login) tool to log into such a device. If an attacker succeed in login

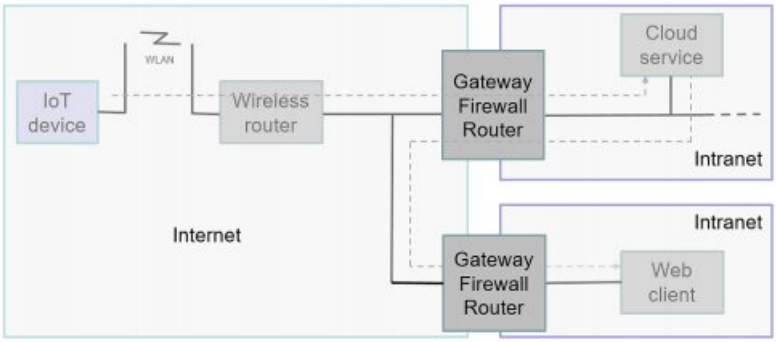


Figure 28.9: An IoT device connected on Internet, outside the network devices securing the intranet.

into the IoT device, then it is possible to login into the computer running the cloud service. The intranet with this computer will have rules in the firewall to allow packets from this device through the firewall and into the intranet. This is one reason why data security for IoT devices is important. Please do not let the administrator of the IoT OS be open (without a password). Figure 28.9 shows an IoT device using a wireless network on Internet.

An advice will be to include the IoT devices in an intranet where the device is protected by the firewall but remember that an intruder can still try to log into the wireless network in these intranet in an attempt to access the IoT devices. An industrial IoT (IIot) device is a more robust IoT device with extended functionality on among others security and interoperability.

Some advices regarding the security of IoT devices:

- change the standard password of the device and avoid devices that have a fixed password (stored in the firmware),
- use separate network segments for the IoT devices. Avoid connecting the IoT devices to the administration network or the production network. An IoT device should not have access to several network segments as the IoT device can be used as a bridge between these networks.
- turn off unused functions. An IoT device often has more functions then what is needed, turn off all unused functions.
- keep the IoT updated with the latest versions of the software for the device (operating system and/or control system).
- perform penetration tests of the Industrial IoT device by a security expert with knowledge of PLCs and SCADA systems. Make a risk-based assessment based on these tests to decide what is important to do (before anyone with hostile intentions decide to give it a try).
- understand main the functions even if you have bought a complete IoT system. Type of sensors, type of information, information flow and where is the information stored.
- make a plan for what to do when something goes wrong. It may be more lucrative to hack an IIoT device to stop the production then to work with ransomware.

Some examples of how IoT connectivity will be regulated and expected to converge on a minimum level of security to protect people and property:

- the European Union Agency for Cybersecurity, ENISA, which released the "Good Practices for Security of IoT" report to promote security by design for IoT,
- the Australian government released the "Code of Practice: Securing the Internet of Things for Consumers" to ensure that the expectations are met regarding cyber security,
- in California the $SB - 327$ legislation will come into effect from $JAN - 20$.

28.4.4 Mobile Phone

Newer mobile phones like smart phones are actually small computer systems and should be protected just like your other computer systems. Small applications (apps) can be downloaded and installed on these mobile phones. How secure are these applications? How are you protecting the information stored on your mobile phone? Connecting to wireless network all over the world, how secure are these wireless networks? Can you trust these networks?

28.4.5 Ping application

Ping is a computer network administration utility application used to test the reachability of a host on an Internet Protocol (IP) network, based on the lowest layers of the OSI model. It is available for virtually all operating systems that have networking capability. The ping application can be used by an attacker to check if a computer system is present and responding to TCP/IP messages and can also be used to slow down the response of the computer system by sending ping requests as fast as possible. The ping support can be turned off in many firewall systems.

28.4.6 Wireless network

Wireless Ethernet is often referred to as WLAN and gives a freedom of location both inside and outside an organization. The range of these WLAN's is at least 30 m and more than 100 m on newer standards. Using WLAN in an organization it will be important to have some sort of security policy to control the access into the network. An open WLAN will distribute the name of the WLAN and allow everybody to connect to the network, especially when using DHCP. Open means that everybody can see the name of the network and that the network is not using any form of encryption. The name of the WLAN is the SSID, Service Set Identifier, and is broadcasted to be able to connect to the network. The list of WLANs shown in the setup section of the computer will be the SSID. The SSID can be turned off but is still transmitted and can be displayed with additional software.

Two important parameters for the security of the WLAN will be the SSID and the encryption: 1) Select a SSID so that it is not so easy to see the connection between the organization and the SSID. 2) The encryption is the coding of the wireless traffic between the computers and the access point. The Wireless Equivalent Privacy (WEP) encryption will be one option, but is a weak and standalone encryption method with different standards. Using WEP will not guarantee that the computer will connect to the access point due to different standards and if the computer connects. The Wi-Fi Protected Access (WPA) protocol was created in response to several serious weaknesses in the WEP. WPA is administered by the Wi-Fi Alliance as a security protocol created to secure wireless computer networks. WPA is using a password key for access to the network. Pre-shared key (PSK) mode (also known as Personal mode) is designed for home and small office networks that don't require the complexity of an 802.1X authentication server. Each user must enter a password to access the network. The password may be from 8 to 63 printable ASCII characters or 64 hexadecimal digits (256 bits). WPA2 is replacing WPA. The IEEE 802.11i encryption, also known as WPA2, is defined as a standard and using the Advanced Encryption Standard (AES) for encryption. The AES is the current highest standard for encryption giving a much better encryption than WEP. The predecessor to WPA2, WPA, is using Temporal Key Integrity Protocol (TKIP) which is an significant improvement over WEP. Both WPA and WPA2 can be used for WLANs. The WPA2 and WPA encryption standards require encryption keys, either enter manually when setting up the computer, or using a server for distributing of the encryption keys.

28.5 Cyber threats

Cyber threads in this setting is an attack that are using a digital system via any of the I/O ports on the computer systems, mainly the communication ports. Most people think on cyberspace, which is a virtual space that doesn't exist, it is just a computer system in another intranet connected to Internet. So cyber threats are a computer system, in an intranet, that are used for trying to come in contact with your computer system. Cyber threats can be divided into three broad categories as 1) financial gain; somebody wants to earn money, 2) disruption; somebody wants to make some problems for or destroy something and 3) espionage; somebody wants access to valuable information.

Cyber security is the protection against cyber threads, often attempting to access your system using the network. This access can be 1) login into your system, 2) install software for some sort of action

on the computer, 3) install software that open a new way for login, 4) install some software for using your computer for some sort of action towards other computers or 5) try to shut down your computer system. These accesses will be either try to log into your system (remote login over the network), sending messages or sending an attachment to a message.

28.5.1 Passwords

There exists several applications like *rlogin, TelNet, ssh (secure shell)* and *scp* that can be used for connecting and log into remote computer systems. As long as you know the username and the password, you can log in remotely and do the same operations as you where in front of the computer system. The username is easy to find, based on the emails addresses. The only barrier will be the password, how easy will it be to guess your password? If the attacker knows your name, find your Facebook or Instagram or other social accounts, will he be able to guess your password? Remember that passwords are for granting access to your computer system, not for making it more difficult for you to use the system. The most used (popular) passwords are: 1) 123456, 2) password, 3) 12345678, 4) qwerty, 5) abc123, 6) admin, 7) letmein, 8) login, 9) name of pet, 10) name of girl friend, and 11) name of children(s). A two pass approach can also be a good way of logging into a computer system, using two different devices. The first pass will be to login with a password to the computer system and the computer system will send a code to your other device, like the mobile phone, the second pass will be a new password step with the code from other device. Your computer system needs an additional service to send the code to your mobile phone but in this way an intruder must have access to your computer user name, your user password and your mobile phone to be able to log into your computer system. Common sense and a good password routine can go a long way to protect consumers from cyber threats. A statement indicates that more than 70% of threats are due to weak or stolen passwords[3].

The most used passwords in 2020 and 2021, according to the "*Top 200 most common passwords*" list from Nordpass[4], are:

	Globally 2020	**Globally** 2021	**Norway** 2021
1	123456	123456	123456
2	123456789	123456789	123456789
3	picture1	12345	12345
4	password	qwerty	12345678
5	12345678	password	passord
6	111111	12345678	1234
7	123123	111111	password
8	12345	123123	1234567
9	1234567890	1234567890	lol123
10	senha	1234567	abc123
11		qwerty123	123123
12		000000	qwerty
13		1q2w3e	111111
14		aa12345678	1234567890
15		abc123	liverpool

Be aware that any hacker will launch an attack by testing these passwords, and this is usually done automatically. So choose your passwords carefully. Some tips for choosing good passwords, according to Nordpass, are:

- Use a phrase in stead of a word,
- Use a sentence in stead of a word,
- Do not use the same "password", better to use a fixed parts that can be combined in different ways,
- Combine the "password" with numbers and special characters.

[3] www.ictsecuritymagazine.com: 2017
[4] www.nordpass.com (27-DEC-21)

28.5.2 Emails

A good password routine will protect your system against remote login, but there exists other methods for getting access to a computer system. One methods is the usage of emails, either with an attachment or with a link to a specific web site. Opening an attachment will try to start a new application for presenting the attachment. The attachment can be a pdf file for opening in pdf reader, a word file for opening in MS-Word or an excel file for opening in MS-Excel. These attachments may also contain executable files that may install on your computer system, or they may include macros that can be executed. Therefore, always be aware when open an email attachment. In this case it is the user that is largest threat for the computer system. Another challenge with emails is spam email which are not a threats, but takes time and use computer system resources. Any links in emails may also be a way to trick the user to log into a fake web site with your username and password. Most of the attacks today starts with an email trying to manipulate the recipient of the email and therefore will user training be important.

Spam

Spam emails is not a threat but they will still affect the number of messages in an email system and use storage space. Spam is the abuse of electronic messaging systems to indiscriminately send unsolicited bulk messages. While the most widely recognized form of spam is email spam, the term is applied to similar abuses in other media like instant messaging spam, web search engine spam and mobile phone messaging spam to mention a few. More than 75 to 80% of the emails received by a company email system can be spam emails. Note that your system can be used for sending spam emails.

Business email fraud

Some of the business email frauds:

- A fraudster sending an email poses as a boss in a company instructing staff to make a wire transfer into the fraudsters account, where the boss seems to be in a hurry,
- Fraudsters pose as the IT services department of a bank saying they want to make a test transfer,
- Fraudsters claim to be a supplier and ask for outstanding invoices to be paid into a new bank account,
- Employees click on links within phishing emails containing malware which authorizes many small payments to the fraudsters account.

28.5.3 Malware

Malware software is any software that is designed to cause any damage to a computer system or computer network, and consists of different types like 1) virus, 2) worms, 3) trojans, 4) phishing and 5) ransomware. Antivirus software is a good way to avoid malware software however the antivirus software must be both updated automatically and scan the computer based on a scheduled plan.

Virus

A virus software is a small program written to alter the way a computer operates, without the permission or knowledge of the user. A virus must meet some criteria: 1) It must execute itself. It will often place its own code in the path of execution of another program, replace the code of an existing program or add the code to the end of an application, 2) It must replicate itself, 3) It can not be controlled by remote messages. Some viruses are programmed to damage the computer by damaging programs, deleting files, or reformatting the hard disk. Others are not designed to do any damage, but simply to replicate themselves and make their presence known by presenting text, video, and audio messages. As a result, they often cause erratic behavior and can result in system crashes. Computer viruses are often spread by attachments in email messages, instant messaging messages, attached to files that are downloaded on the Internet, including MS-Word and MS-Excel files (as macros).

There are some recognized types of viruses:

File infector viruses File infector viruses infect program files, normally executable code, such as .com or .exe files on a Windows system. Many of these viruses are memory resident and will infect other files when running.

Boot sector viruses Boot sector viruses infect the system area of a disk, the boot record on hard disks and USB disks. These viruses are always memory resident in nature and normally designed for Windows systems.

Multipartite viruses Multipartite (also known as polypartite) viruses infect both boot records and program files. These are particularly difficult to repair. If the boot area is cleaned, but the files are not, the boot area will be reinfected. The same holds true for cleaning infected files. If the virus is not removed from the boot area, any files that you have cleaned will be reinfected.

Macro viruses Data files infected by viruses and are the most common viruses affecting MS-Office data files like MS-Word, MS-Excel and MS-Powerpoint files.

Worm

Computer worms are the most harmful of the malware types. The reason is that worms do not require computer users to take any type of action to spread them. A computer worm is similar to a virus in that it has the ability to reproduce itself. While viruses require a host program to spread to other computers, the worm does not. Worms can do as much or more damage than computer viruses. They can corrupt programs and systems, delete files, and generally make a system run poorly. While viruses are more typically found on single computers, and spread to other single computers, worms prefer networks, where they can simply infect and move to the next computer in the network. Worms travel through the routers and servers that make up networks.

Firewalls are meant to make your computer less visible to worms and such, especially on networks. However, firewalls of today will only stop specific commands or access to specific communication channels. Firewalls will not check the contents of valid commands or information on valid communication channels. Many worms are delivered via email attachments, just the way viruses are. Worms can be controlled by remote messages.

Trojan

The term is derived from the classical story of the Trojan Horse. A computer trojan is normally part of another software or a software that seems OK. Once the trojan is installed it will try to steel information from your system and open some sort of communication channel to let other system connect to get the information. Trojan is a malware like worm and virus, but will not replicate itself. Trojans are normally delivered via email attachments.

Phishing

Phishing is a computer attack to get sensitive information from the owner of the computer or the computer network (company). This is the analog attack into a digital world, attacking the user, not the digital system. The attack exploits human qualities such as confidence and curiosity. The attack is normally based on emails or instant messages that seem to be sent from trustworthy source. These messages contain either malware code as an appendix or a link to malware web page. The goal is to install malware software on the computer to get private information about the user of the computer or the about the information on the intranet, or even access to the computer or the intranet. Protecting your systems from this type of data attack will be a balance between technology, routines and human factors.

Ransomware

In older days the threats were stealing information from the computer systems, now both stealing information and holding the information as hostage have been done. Holding the information or the storage device as hostage is named ransomware. The information on the storage device is encrypted and the owner is prompted to pay for the decryption key to get access to the information on the device. Ransomware has been growing rapidly since late 2013. Ransomware is a subclass of malware that is

characterized by encrypting all information on a device. Sophisticated ransomware attacks disk-level or file-level encryption, making it impossible to recover files without the decryption key. Ransomware is often spread as email attachments or malicious links on web sites. Some advises to avoid ransomware[5]:

- keep the operating system, antivirus and internet protection updated,
- backup the information regularly, and keep the backup in an offsite location. Best solution is to have two sets, one set in the cloud and one set on an external physical device (normally disconnected from the system),
- segment the network, distribute the information so not all the information is accessed by everybody in the company,
- do not trust information from anybody, neither friends nor colleges. Never open an attachment from a sender that you do not know,
- use a robust antivirus application,
- activate the file extension option in the operating system and avoid attachment files with the extensions ".exe", ".vbs" or ".scr",
- if you discover any runaway or unknown processes on your computer, disconnect from the Internet immediately,
- train and test for any attack.

If you have been attacked always report the matter to the police, avoid paying a ransom if possible, and check with the website nomoreransom.org if this type of attack is known. The website may even have a solution.

28.5.4 Denial of Service (DoS)

Denial of service (DoS)is a cyber attack where an attacker seeks to make a computer system or computer network resource unavailable to its intended users by temporarily or indefinitely disrupting services connected to Internet. Denial of service is typically accomplished by flooding the targeted system or resource with superfluous network requests in an attempt to overload systems and prevent legitimate requests from being fulfilled. In a distributed denial-of-service cyber attack (DDoS) the flooding of network requests originates from many different sources making it impossible to stop the attack by blocking a single source.

28.5.5 Botnet

Botnet is a term for a collection of software robots (bots), that run autonomously and automatically. The term is often associated with malicious software but it can also refer to the network of computers using distributed computing software. Bots can be used for either good or malicious intent. A malicious bot is self-propagating malware designed to infect a host and connect back to a central server or servers that act as a command and control center for an entire network of compromised devices (botnet). With a botnet, attackers can launch broad-based, "remote-control," flood-type attacks against their target(s).

In addition to the worm-like ability to self-propagate, bots can include the ability to log keystrokes, gather passwords, capture and analyze packets, gather financial information, launch Denial of Service (DoS) attacks, relay spam, and open backdoors on the infected hosts. Bots have all the advantages of worms, but are generally much more versatile in their infection vector and are often modified within hours of publication of a new exploit. They have been known to exploit backdoors opened by worms and viruses, which allows them to access networks that have good perimeter control. Bots rarely announce their presence with high scan rates that damage network infrastructure; instead, they infect networks in a way that escapes immediate notice. Botnets may also take advantage of internet of things (IoT) devices such as home electronics or appliances to increase automated attacks.

[5] Computerworld - 5-oct-20

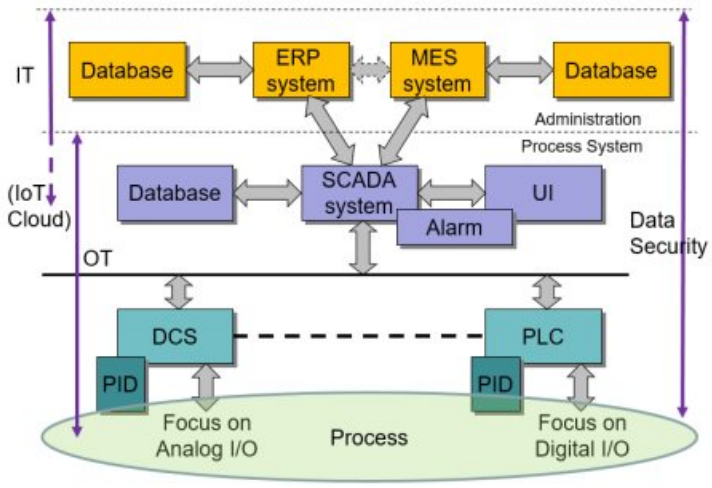


Figure 28.10: A process control system (or industrial control system) with the administration part (information technology) and the process system (operational technology).

28.5.6 Cyber attack

Hackers will look for the easiest way into computer systems, which will be the employees. So cyber security should be a company priority and not only a challange for the IT department. The company management should provide regular security awareness training for the employees to be aware of possible threats. More than 90% of all cyber attachs begin with an email and many people can not identify a sophisticated phishing email[6].

28.6 Process control systems (PCS)

An industrial control system is the heart of any production. It is in this OT environment that values are created. A stop in production often leads to direct losses in the form of e.g. reduced production time, wrecks and poorer quality, which in turn can lead to a reduced bottom line. The production also consists of machines, equipment and materials that can be harmful to people and the environment. Unwanted activity in an OT environment can therefore have major consequences. Risk analyzes often reveal that the greatest risk lies in production. The reason for this is both that an incident in this environment can have major consequences, but also that the combination of many vulnerabilities in an OT environment and increased activity among cyber criminals increases the probability of being affected.

The security aspect is going to more an more important for monitoring and control systems. Some of the reasons are that the technology is continuously dropping in price, the connections to internet are increasing, the use of internet of things (IoT) and industrial IoT (IIoT) devices, tighter connections between the IT (information technology) and OT (operational technology) and digitalization (sharing of information). Integration is the heart of any automation architecture. However, industrial security is a complicated, multifaced challenge that cannot be solved by simply purchasing the latest technology. Managing the security requires improved processes, tools and balancing the risks. A process control system (PCS) or industrial control system (ICS) is shown in Figure 28.10 with the IT and OT parts. One of the most important measures to reduce risk in production is to segment the plant so that a possible attacker does not have access to everything if he first comes within the company's digital systems. The first priority for many is therefore to separate the IT systems from the OT systems.

OT is used for monitoring and controlling industrial systems and other critical systems, and is still more often connected to Internet. These systems are often designed without the low level security and are therefore vulnerable. OT systems have a long life cycle and therefor it takes time to include the low level security into these systems. Many organizations also lack an overview of their OT networks, what is connected and where the vulnerabilities are.

Information technology (IT) focus on computer technology with regular patching and a short life cycle of the equipment. The focus for the IT systems will be to 1) manage data security, 2) handle information

[6] www.vipre.com (Dec-21).

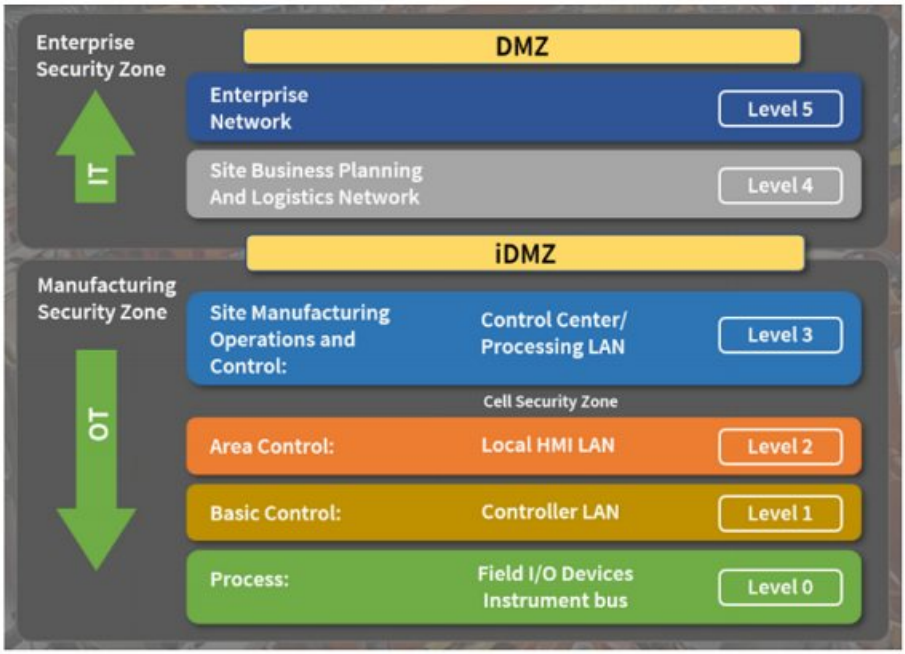


Figure 28.11: The PCS divided into the IT and OT security zones and levels, including the demilitary zones of the firewall system.

and 3) avoid downtime in that order. Operation technology (OT) focus on process technology with irregular patching and a long life cycle of the equipment. The focus for the OT system will be to 1) avoid downtime, 2) handle information and 3) manage data security, a different focus than the IT system. OT is all the equipment that the system will depend on to keep production going.

Most control system suppliers refer to the so-called Purdue model for segmenting systems in production. The Purdue model is also used in the IEC 62443 standard, a standard for data communication and cyber security in industrial control systems. The model describes a layered division, from equipment placed in the field such as sensors and actuators, via controllers and SCADA, and up to internal demilitary zones (iDMZ), the interface to the IT systems. The data security zones of a PCS or ICS can be divided into different levels as shown in Figure 28.11 where the IT and OT parts have be divided into different levels.

The figure also shows the demilitary zones (DMZ) of the firewalls having one zone between internet and the enterprise network (administration network), and another firewall zone (internal DMZ) between the enterprise network and the manufacturing network (process network). The manufacturing security zone is divided into four levels of security, and the three lowest levels integrated into cell security zones, from level 0 as the lowest security level to level 3. The enterprise network has security level 4 and 5. IoT and IIoT devices can be connected to the manufacturing network, enterprise network or outside the enterprise network and may need extra protection by additional firewall modules or devices.

If an attack is discovered, one solution will be to disconnect the OT environment from IT. If the system has an established iDMZ, this is easy, but the system must be designed for it. Looking into the details, there will be several challenges associated with this. The main challenge will be the integrations with the business systems (ERP). The ERP systems are located in the IT environment, but provide the production with important production data, and the status that is sent back. How long can production be maintained without access to important data from ERP depends on the business model. Companies that produce for stock will find it easier to continue producing even if the ERP system is down, than those companies that have an optimized value chain that is based on order production.

Attackers will exploit any weak section of any system, and in a PCS can these sections be:

- integration between the PCS and the administration system (office system), the integration between the IT and OT systems,
- weak or bad configuration of the firewalls,
- the security of any remote connections into the system,

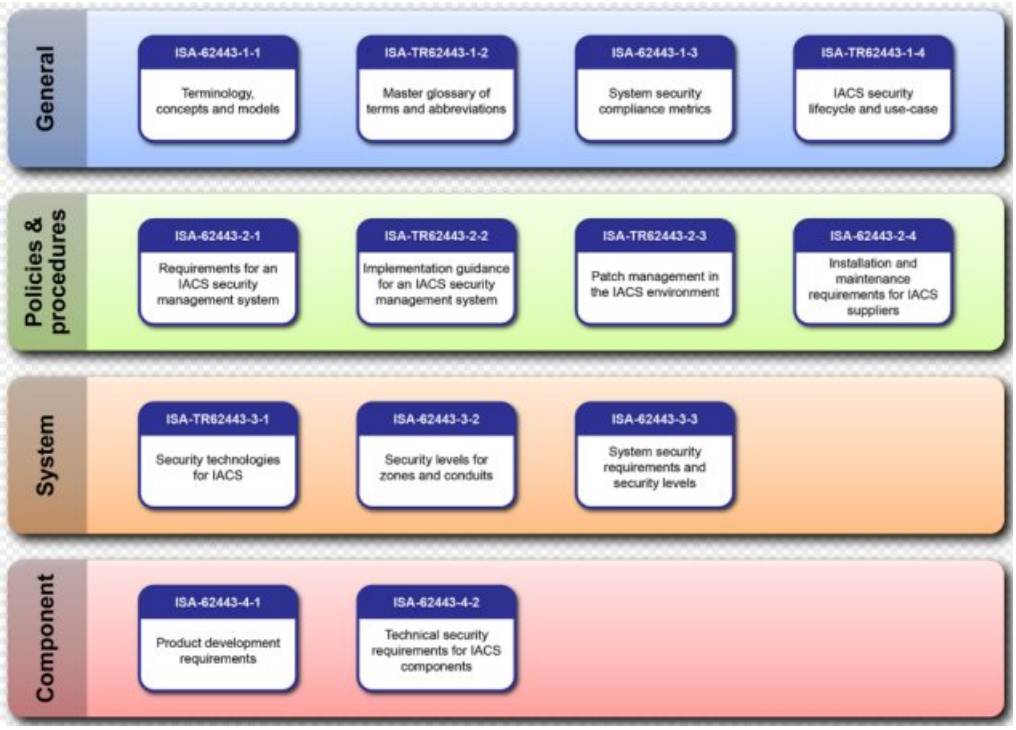


Figure 28.12: The organization of the ISA-62443 security standard for control systems into the four categories General, Policies and Procedures, System and Component (www.isa.org/isa99: nov-19).

- use of IoT and IIoT devices, especially outside the intranet protected by firewalls,
- relaxed authentication of users and wireless equipment,
- any cloud services in use,
- old operating systems like Windows NT, Windows XP and Windows 7.

The advice to have an ongoing process about the security issues with continuous maintaining and training. Try to avoid any type strict and static security system, involve all the users.

28.6.1 Control system security standards

Control system security is the prevention of unwanted interference with the operation of industrial automation and control systems.

ISA-62443

One important security standard is the ISA−62443 (ISA-62443 2018) standard, which is a series of standards, technical reports, and related information that define procedures for implementing electronically secure in industrial automation and control systems. This standard is more like a guidance that applies to end-users, system integrators, security practitioners, and control systems manufacturers responsible for manufacturing, designing, implementing, or managing such systems. This standard started as the ISA−99 standard, and was renumbered to the ISA-62443 in 2010. The standard is still maintained by the ISA−99 committee and involve hardware and software systems such as DCS, PLC, SCADA, networked sensing, monitoring and diagnostic systems associated with human, network or machine interfaces used to provide control, safety and manufacturing operations.

The ISA−62443 standards and technical reports are organized into four general categories called General, Policies and Procedures, System and Component as shown in Figure 28.12.

This figure shows these four general categories, or levels. The first, on the top, category includes foundational information such as concepts, models and terminology. The second category of work products targets the Asset Owner. These address various aspects of creating and maintaining an effective Industrial Automation and Control Systems (IACS) security program. The third category includes work

products that describe system design guidance and requirements for the secure integration of control systems. Core in this is the zone and conduit design model. The fourth category includes work products that describe the specific product development and technical requirements of control system products.

The standards describe cyber security as an ongoing process and not as a goal that has to be reached. Some important standards in the ISA−62443 series are the following:

- ISA−62443−2−4, "*Cyber Security Capabilities*" which covers the policies and practices for system integration,
- ISA−62443 − 4 − 1, "*Product Security Development Life-Cycle Requirements*" which covers the secure development life cycle requirements,
- ISA−62443 − 4 − 2, "*Security for Industrial Automation and Control Systems: Technical Security Requirements for IACS Components*" which covers the IACS components security specifications,
- ISA−62443−3−3, "*System Security Requirements and Security Levels*" which covers the security requirements and the security levels.

Regarding risk management will the standard ISA−62443−3−2, "*Security Risk Assessment, System Partitioning and Security Levels*", based on the understanding that IACS security is a matter of risk management.

The scope of the 62443 standard for IACS is:

Categories	**Apply to**		**Information**
General		1.1	Terminology and concepts
		1.2	Master glossary
		1.3	System security compliance metrics
		1.4	IACS security life cycle and use-cases
Policy	Asset	2.1	Requirements for an IACS security management system
Procedure	Owner	2.2	Implementation guidance
		2.3	Patch management
		2.4	Installation and maintenance
System	System	3.1	Security technologies for IACS
	Integrator	3.2	Security levels for zones and conduits
		3.3	System security requirements and levels
Component	Component	4.1	Product development requirements
	Supplier	4.2	Technology security requirements for IACS components

NIST 800-82

National Institute of Standards and Technology (NIST) [7] has published the document "Guide to Industrial Control Systems (ICS) Security" (Stouffer, Pillitteri, Lightman, Abrams & Hahn 2015) to understand and implement cyber security approaches to protect computer-based industrial systems.

28.6.2 Protocols

Industry based protocols like Transport Layer Security (TLS), Web Sockets Secure (WSS) and Hyper Text Transfer Protocol Secure (HTTPS) can be used for multilayered security in PCS. These protocols support encrypted communication and role based user security. TLS is a cryptographic protocol designed for secure communication over a computer network, both intranet and Internet. The basic functionalities of TLS are 1) message encryption, 2) detection of message alteration, and 3) authentication between server and client providing a fully encrypted communication.

Virtual Private Network (VPN) establishes a secure connection path between two connection points in a system and ensure that all communication is encrypted on this tunnel.

[7] NIST: www.nist.gov

28.7 Are the computer system hacked?

Any computer connected to a network or using external memory sticks can be hacked, meaning that some software can start controlling part of your computer system. People normally install anti virus software, but this application alone is not enough anymore. Some sign that you may have unwanted software in your system:

- "false" messages from anti virus software, telling you to install new software or visit a specific web site to get new software,
- new or unwanted tool bars in the web browser,
- random pup-up windows,
- unwanted installation of software applications,
- the mouse cursor starts moving without you doing anything.

In all cases, turn of the computer, disconnect any external devices and the network, and "clean" your system. Hacking is normally due to 1) not updated software, 2) running of a Trojan type of software and/or 3) responding to email of the phishing type (respond with any private information like credit card number or password).

28.8 Cyber security skills

The skills required to work in cyber security vary depending on the position and company, but will include knowledge about:

- functions and operations of operating systems (OS) with focus of security levels,
- data communication, protocols, OSI communication model and the integration with OS,
- software engineering; developing and deploying software,
- knowledge about malware software,
- risk analysis and management,
- security assessment,
- security standard,
- penetration testing.

Important to start the security work before a data attack occurs to escape many of the problems. Invest in technology and manpower before the attack, and start logging and response exercises. Important factors to understand the extend of a data attack is to: 1) have a good overview of the where the information is stored in the intranet, who has access and who the information is used in the organization, 2) have enough logs to explore the information burglary and 3) detect the intrusion in time.

Part X

Instrumentation

Chapter 29

Introduction

Instrumentation is a common description of the physical devices used for monitoring and controlling a process, sensor devices for monitoring purposes and actuator devices for controlling purposes. Figure 29.1 shows a section of a plant with a number of valves controlling the flow in a set of pipes. The valves normally contain sensor devices indicating the positions of the valve disc like fully open and fully closed. These signals are feedback signals to the control system for monitoring the actuators in addition to all the sensor devices monitoring the process.

The control system normally provides signals used to operate actuators like solenoids, valves, regulators, circuit breakers, or relays. These actuator devices control a desired output variable, and provide either remote or automated control capabilities. The figure shows an industrial process plant but a process can also other types like a car or a house. A car today contains several controllers like the engine controller, the brake controller and the air condition controller and all these controllers are communicating using a local area network inside the car. Each of these controllers have their own set of sensors and actuators.

Figure 29.1: A small plant with a set of valves (orange) together with postiotion sensors (green).

Chapter 30

Sensor and Actuator devices

The instrumentation of a plant includes both sensors and actuators. The sensors are used for getting information from the plant into the monitoring or controlling system. The actuators are used for the computer system to control/change/adjust any output parameter to the plant. The actuators are the output interface between the computer system and the process, and is converting a signal to a physical quantity (Olsson & Piani 1998). The signal is normally an electrical signal, but can also be pneumatic using air or fluid.

The type of common actuators are switches, solenoids, and motors (Bishop 2008). Switches are devices that can control an activity by turning a signal on or off, transistor types for low current flow (less than 1 A) and relays for high current flow. Solenoids are devices with a movable iron core that can be adjusted by current flow, controlling some type of hydraulic or pneumatic flow (Bishop 2008). Motors are divided into three different type; alternating current (AC) motors, direct current (DC) motors, and stepper motors. AC motors are generally cheaper than DC motors, but require a frequency controller for controlling the speed. The speed of DC motors can be controlled by a voltage signal. Stepper motors rotate a certain number of degrees in response to an input pulse signal (Bishop 2008). Actuators will be either the ON/OFF types or the continuous types. Actuator can be used to control (among others) flow, heat, position, rotation, and indications.

Actuators are a subdivision of transducers, devices which transform an input signal into some sort of motion. The motion actors can be divided into the following groups: 1) Multi-turn; an actuator which transmits to the valve a torque for at least one full revolution, 2) Part-turn; an actuator which transmits a torque to the valve for less than one full revolution, and 3) Linear; an actuator which transmits a torque to the valve as a linear operation.

The electrical signal used for controlling the actuators often requires more power than the output signals of a computer can deliver, so different kind of power amplifiers and switches like relays are used. The interface from a computer can be: 1) Analog current signal like $4 - 20$ mA, 2) Analog and digital by using $4 - 20$ mA and HART, 3) Analog voltage signal like $0 - 5$ V, 4) Digital on/off signals using a digital bit, 5) Digital by using some sort of fieldbus, 6) Frequency converters for controlling the speed of motors, and 7) Pulse Width Modulation (PWM).

These interfaces must be powered using a relay or an amplifier. A relay is just an on/off switch where a low current signal can be used to control a high current and/or high voltage signal. An amplifier is used for amplification of the computer signal being a voltage or current signal. The computer interfaces (or switches) for controlling the actuators uses methods like: 1) Power operational amplifiers, 2) Pulse Width Modulation (PWM), 3) Frequency converter, and 4) Relays.

Some of the actuator types are: 1) Electrical motors; An electric motor converts electrical energy into mechanical energy, 1a) stepping motors; controlled in fixed angular increments (steps), 1b) direct current (DC) motors; the speed is dependent of the current (or voltage), 1c) alternating current (AC) motors; better then DC motors, but require a more complex control circuitry, 2) pneumatic or hydraulic actuators, and 3) relays.

Actuators can be used to control devices like: 1) Valves: A valve is a device that regulates the flow of substances (gases, fluidized solids, slurries, or liquids) by opening, closing, or partially obstructing various passageways, 2) Pumps: A pump is a device used to move liquids, or slurries. A pump moves liquids from lower pressure to higher pressure, and overcomes this difference in pressure by adding energy to the system (such as a water system), and 3) Compressor: A gas pump. In very low pressure-rise applications, such as in heating, ventilating, and air-conditioning, the equipment is known as fans or blowers.

Figure 30.1: An actuator consistion of electronic and mechanical modules to convert an electrical signal into some sort of movement (www.rotork.com: $NOV - 2013$)

An actuator is shown in Figure 30.1 consisting of both mechanical and electronic modules, a motor, a gear and electronics for communication and control of the device.

Power Amplifiers

The power amplifiers are used for amplifying the signals from the computer interface before the signals can be feed to the actuators. These power amplifiers can be separate units or integrated with the actuator. These units are also known as power operational amplifiers, servo amplifiers or programmable power supplies (Olsson & Piani 1998).

Pulse Width Modulation (PWM) is a common technique to control power levels. The output voltage is switched between two constant levels at a frequency in the Hz or kHz range. The modulation of the width of the pulses is controlling the average voltage level to the actuator. PWM signals are used to drive continuous actuators where the frequency is higher than the time constant of the actuator. The oscillations in the behavior of the actuator may become negligible (Olsson & Piani 1998).

Frequency converters for controlling the speed and torque of motors, based on the PWM technique. Frequency converters are used together with AC motors (Brown 2010).

30.1 ON/OFF actuators

The ON/OFF types can be either a mono stable or a bistable type (Olsson & Piani 1998). The mono stable actuator has only one stable position and is controlled by a digital signal. The stable position is when the actuator has no signal, the digital "0" signal. The actuator is controlled by the digital signal, and will be in the stable position with the "0" signal and the unstable position with the "1" signal. These types of actuators are also often used in Fail-safe system where all ON/OFF actuator will return to a fail-safe position, the stable position, in case of any errors like power failure etc.

The bistable actuator remains in the latest position until receiving a ON signal. The ON signal will shift between the ON and OFF position of the actuator.

The output signal from a computer has a very low power and an amplifier is needed. The most commonly used ON/OFF actuator is the electromechanical relay. A current through the relay coil will generate a magnetic force moving one or several mechanical switches from one position to another. Some sort of cheap amplifier like a transistor or driver circuit is needed to control the current of the relay.

The disadvantages of electromechanical relays are:

- mechanical devices,
- slow operation.

Solid-state semiconductors can also be used for switching high power, like MOSFETs (Metal Oxide Field Effect Transistor), Thyristors and Triacs. Figure 31.11 shows why a diode should be used when switching an inductive ON/OFF actuator.

Lamps and horns are typical devices that can be controlled by on/off actuators.

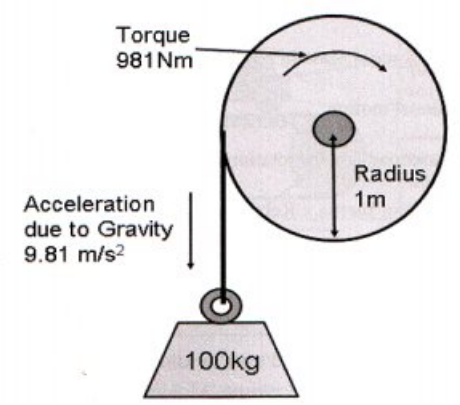


Figure 30.2: Torque for a pulley of radius 1 m, weight of 100 kg, and acceleration of gravity (Brown 2010).

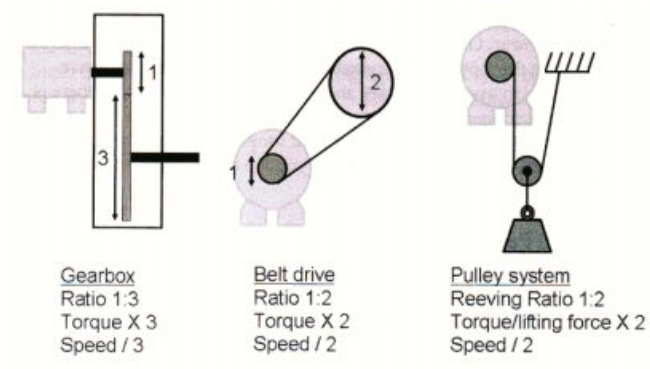


Figure 30.3: The relationship between speed and torque when using a gearbox, belt drive, and pulley (Brown 2010).

30.2 Continuous actuators

Electrical drive system indicates the combination of a motor and control electronics (Olsson & Piani 1998). The control electronics is used for the possibility of regulating the speed and torque within a wide range and with high precision.

30.2.1 Mechanics

Inertia and Moment of Inertia

A force is needed to accelerate a mass, or to overcome the acceleration due to gravity. A rotational mass has a rotational inertia, known as the moment of inertia.

Torque

Torque is the twisting force that turns things. The torque is proportional with the product of force, weight and distance. Figure 30.2 shows an example of torque.

Gear

The relationship between torque and speed when using any type of gearing is shown in Figure 30.3.

30.2.2 Actuator types

Most continuous actors used in control systems are D.C. (direct current) motors, asynchronous (induction) motors, synchronous motors, and stepping motors. Actuators are a subdivision of transducers and are devices which transform an input signal into motion. Actuators can be used to control devices like:

1. Valves: A valve is a device that controls the flow of substances (gases, fluidized solids, slurries, or liquids) by opening, closing, or partially obstructing various passageways.
2. Pumps: A pump is a device used to move liquids, or slurries. A pump moves liquids from lower pressure to higher pressure, and overcomes this difference in pressure by adding energy to the system (such as a water system).
3. Compressor: A gas pump. In very low pressure-rise applications, such as in heating, ventilating, and air-conditioning, the equipment is known as fans or blowers.

The motion actors can be divided into the groups:

1. Multi-turn; an actuator which transmits to the valve a torque for at least one full revolution,
2. Part-turn; an actuator which transmits a torque to the valve for less than one full revolution,
3. Linear; an actuator which transmits a torque to the valve as a linear operation.

Some of the actuator types are:

1. Electrical motors; An electric motor converts electrical energy into mechanical energy,
   (a) stepping motors; controlled in fixed angular increments (steps),
   (b) direct current (DC) motors; the speed is dependent of the current (or voltage),
   (c) alternating current (AC) motors; better then DC motors, but require a more complex control circuitry as the speed is depend of the AC frequency.
2. Pneumatic or hydraulic actuators,
3. Light; LEDs can be used for communication mediums like optic fibres.

30.2.3 Motors

An electric motor is an electromechanical device which converts electrical energy into mechanical energy. The basic principle of a motor is that a conductor carrying a current will be subject to a force when placed in a magnetic field. The opposite, a generator, is working from the principle that a conductor moved through a magnetic field will have a voltage induced, leading to a current if connected to a circuit.

All motors consist of a fixed or stationary part (stator) and a rotating part (rotor).

DC motors

In a DC motor will the rotating part carries the current and the stationary part will be a fixed magnet. A electromagnetically force will turn the conductor as shown in Figure 30.4. The important part of the DC motor is the commutator supplying and switching the current to the conductors. The commutator consists of sliding contacts, known as brushes, in contact with the conductors in the rotor.

AC motors

In most AC motors will the rotor be a fixed magnet and the stator will create the dynamic magnetic fields to rotate the rotor. The stator windings set up a rotating magnetic field when the stator windings are energized. The field rotates at a speed that is fixed by the frequency of the supply used to establish the field and the number of 'pole-pairs'. The frequency is controlled by a frequency converter. The speed the field rotates at what is known as the synchronous speed.

Stepper motors

Stepper motors convert electrical pulses into discrete mechanical movements. The shaft or spindle rotates in discrete step and increments when electrical pulses are applied to the device. The motors rotation has direct relationships to the applied input pulses. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied.

The advantages are:

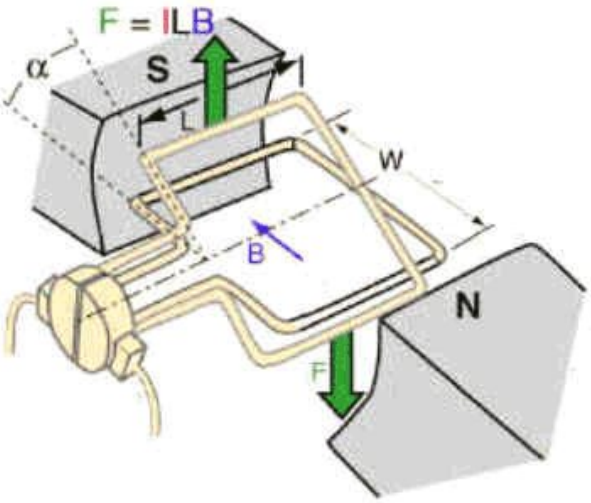


Figure 30.4: The principle of a d.c. motor (www.ncert.nic.in: AUG-2010).

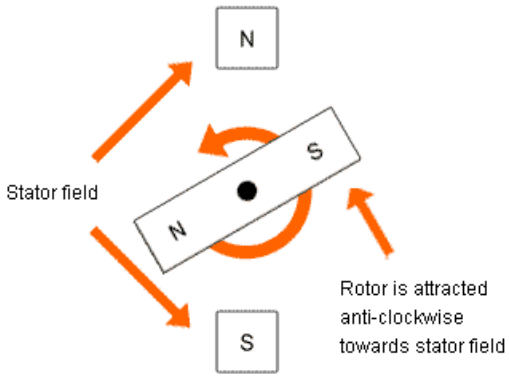


Figure 30.5: An a.c. motor with a fixed rotor magnet and a rotating stator magnetic field (www.lemac.com: aug-2010).

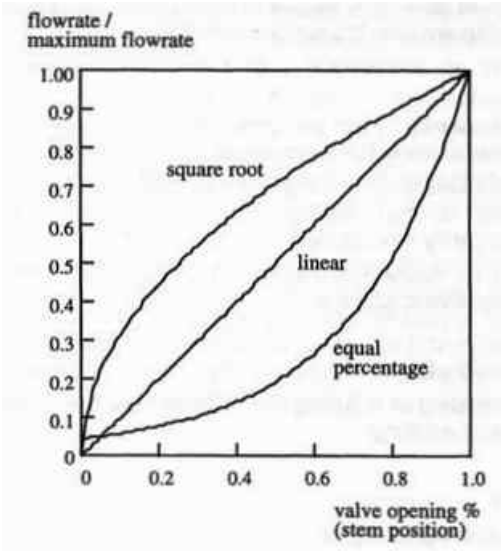


Figure 30.6: Control valve characteristics, linear, square root, and equal percentage (Olsson & Piani 1998).

- The rotation angle of the motor is proportional to the input pulse,
- Precise positioning and repeatability of movement,

The disadvantage is:

- The input pulses provides open-loop control,

Vibrations

Low frequency vibrations, like the one in mobile phones, can be made using a motor with a mass not in centre.

30.2.4 Control valves

Different types of valves exist and the type must be selected to fit the usage in the process. Valve factors taken into consideration: 1) pressure drop, 2) maximum flow rate, 3) range ability, 4) sensitivity, 5) linearity, 6) hysteresis, and 7) sensors.

Typical characteristics for commercially available valves are, see Figure 30.6.

The horizontal axis, x, is the value opening in $\frac{\%}{100}$ while the vertical axis, $f(x)$, is the ratio:

$$f(x) = \frac{\text{flow rate}}{\text{maximum flow rate}}.$$

These characteristics can be:

1. Linear: $f(x) = x$,
2. Square root: $f(x) = \sqrt{x}$,
3. Equal percentage: $f(x) = A^{(x-1)}$ where A is a design constant with a typical value equal to $20-50$.

An important property of a valve is the C_v value defined as:

- The number of U.S. gallons of water passing through a restriction per. minute with a pressure drop of 1 psi.

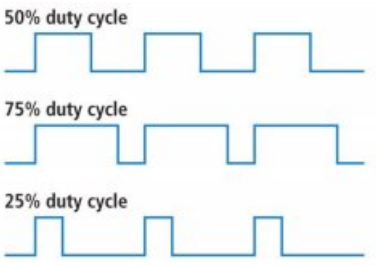


Figure 30.7: A PWM signal used for controlling the spees of a DC motor or the light of a LED. The duty cycle controls the average voltage to the actuator.

Using metric units this property is called K_v and the relationship between C_v and K_v is:

$$C_v = 1.16 \cdot K_v$$

A rule of thumb is that there will be problems with a valve if the pressure drop in the valve is more than 50% of the input pressure of the valve.

Sensors are used to indicate the position of the valve when feedback are used or needed. Two main types exists:

1. End switches, one on/off switch at each end of the valve position. The switches are often either mechanical or inductive, old valves having mechanical switches while newer valves normally will have either mechanical or inductive switches.

2. A linear indication like a potentiometer indicating the position of the valve.

30.2.5 Pumps

A pump is a device used to move fluids, such as gases, liquids or slurries. A pump displaces a volume by physical or mechanical action. Pumps can be divided into different types, where the type describe the method of moving a fluid: 1) direct lift, 2) displacement, 3) velocity, 4) buoyancy, and 5) gravity.

30.2.6 Motor control

Frequency converters or PWM signals are used for control of the rotational speed of electrical motors. Optimization, efficiency and energy savings of electrical motors can be solved by an intelligent and wise combination of usage of engine, transmission, frequency converter and control system.

Frequency converter

Frequency converters are used for controlling the rotational speed of electrical motors, both DC and AC motors.

PWM control

Pulse Width Modulation (PWM) can be used when the “maximum voltage” is needed together with the control of a signal by adjusting the modulation of the voltage. The concept is to divide the signal into time slots and use the maximum voltage into part of the time slots to adjust the speed of a DC motor or the light of a LED. The PWM signal is shown in Figure 30.7.

DC motor control

PWM signal can be used to control the speed of a DC motor either directly or by using a half-bridge controller or a full bridge (H-bridge) controller. A half-bridge controller is shown in Figure 30.8 controlling only the speed of the DC motor while a full bridge controller is shown in Figure 30.9 controlling both the speed and direction of the DC motor. The full bridge controller is also named an H-bridge controller and is used among others for Arduino and Raspberry Pi projects.

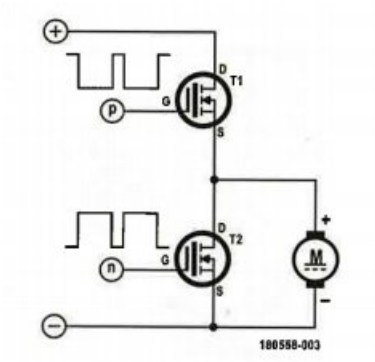


Figure 30.8: A half-bridge controller using PWM signals for controlling the speed of a DC motor (www.elektormagazine.com: Jan-19)

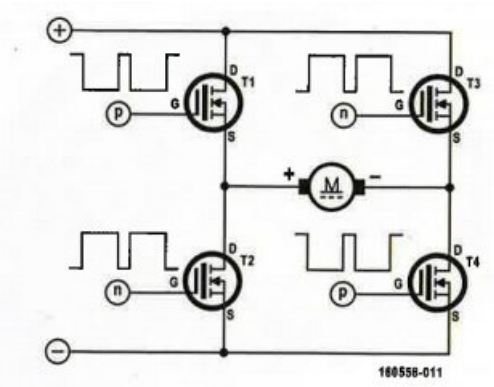


Figure 30.9: A full bridge (H bridge) controller using PWM signals for controlling the speed and direction of a DC motor (www.elektormagazine.com: Jan-19)

30.2.7 Relay

A relay is a very useful actuator for on/off control of output devices where a low power signal from the computer can control the relay, and the relay switch can control a high power signal for any control purposes. This way will the low power computer signals be isolated from the high power section of the process.

Lamp

Lamp is normally an on/off actuator that can be used for indication of an abnormal condition. A lamp is also useful for attract attention by blinking. A low power lamp like a LED can be powered directly from an output of a computer while a set of LEDs or normal lamp must be controlled by a relay or an amplifier controlled. The intensity of the light from the LED or lamp may also be controlled by a PWM signal.

Horn

Horn is an on/off actuator that can be useful for indication of an abnormal condition. A horn is a device converting an electrical signal into an audible signal that can be heard by humans. A horn is useful for attract attention by making a loud sound signal. Horn can be delivered in many sizes, small horns are also called buzzers. The horn will normally be controlled by a relay.

30.3 Sensors

Sensor devices are used for monitoring the process, giving feedback from the process. Sensors can also be used for feedback from the actuators, as an indication of the actuators working. Three categories of sensor devices exist (Olsson & Piani 1998):

1. analog sensors generating a continuous analog signal like; *a*) current signals like $4 - 20$ mA, *b*) resistance temperature detectors (RTDs) like PT-100, PT-1000, *c*) voltage signals like $0 - 5$ V
2. digital sensors giving a pulse train or binary word like; *a*) Hart; combine analog and digital information, and *b*) Field buses; uses only digital information (serial lines or networks),
3. binary sensors giving an on/off signal.

30.3.1 Smart sensors

The shifting towards digitalization, driven the Industry 4.0 and the Industrial Internet of Things (IIoT) will also affect the sensor devices. The wishes is to get as much information as possible out of the sensor devices and there is a change from analog sensor devices in to digital sensor devices. A smart sensor device is a digital sensor device with processing capabilities included, and some sort of bus protocol.

There exists a lot of different sensor types and measurement technologies, but this will not be part of this document. This document will focus more on the system and the sensors need to be connected to a measurement system.

Chapter 31

Data Acquisition (DAQ) Systems

31.1 Introduction

A measurement system is needed to get an analog value from a sensor device, and convert this analog value into a digital representation that can be handled by a computer system. A Data Acquisition (DAQ) System is such a system and used as the connection between the sensor devices, the actuator devices, and the computer system. A DAQ system will be able to convert analog signals from the real world (analog sensor devices) into digital representations for computer systems, and convert a digital representation from the computer system into an analog value as a control signal for an actuator. Figure 31.1 shows a DAQ system with N analog sensor devices connected.

The common sub systems of a DAQ system are: 1) Analog input; signals from analog sensors, 2) Analog output; signals to analog actuators, 3) Digital input; signals from on/off sensors, 4) Digital output; signals to on/off actuators, 5) Counters; counting the frequency, period, or the number of events, and 6) Timers; output event controls or pulse train generation.

It can be different types of modules connected to the computer's I/O ports (parallel, serial, PCMCIA, USB, FireWire, SCSI, network, wireless...) or cards inserted into the slots (PCI, ISA) on the motherboard of the computer. An overview of the connections are shown in Figure 31.2. The connection to the computer can be an internal card or an external device, while the sensor connections most often will be an external box.

Important factors of a DAQ system:

1. Interface: The connection between the DAQ system and the computer system. An internal or external system, connected to the intranet or internet? Cable or wireless?
2. Signal conditioning: The analog to digital conversion of the input signals and the digital to analog conversion of the output signals,
3. Number of analog channels: The number and range of the analog input and output signals,
4. Sampling rate: The time to convert an analog signal to a digital signal,
5. Resolution: The smallest value change the system can detect.

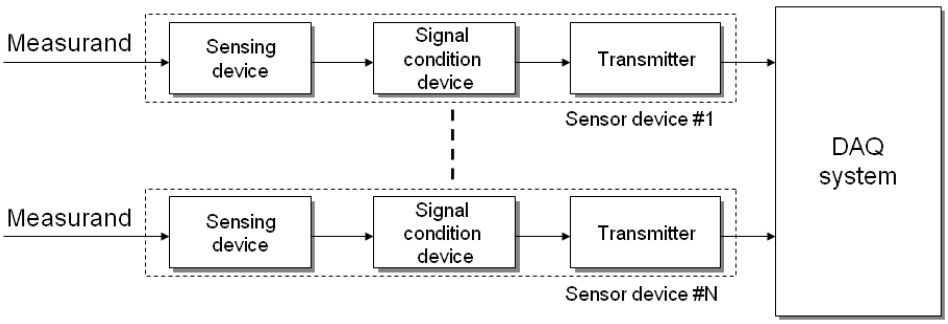


Figure 31.1: A DAQ system with a set of sensors connected.

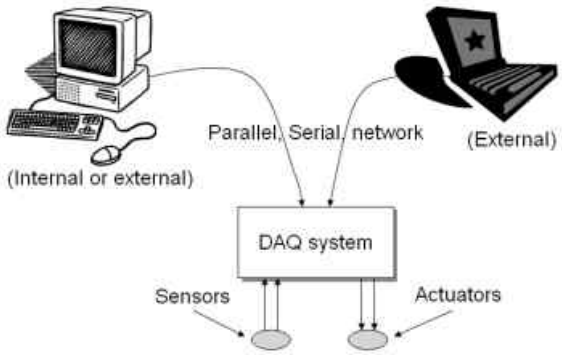


Figure 31.2: The connections of the sensors and the DAQ system to a computer.

6. Accuracy: This is a function of many variables in the system, including A/D nonlinearity, amplifier nonlinearity, gain and offset errors, drift, and noise.

7. Digital I/O: The number of digital input and output signals.

An example of a complete system is shown in Figure 31.3. This system contains a set of sensors, the measurement system, the motor control system, and the MMI (Man Machine Interface). This is a distributed system containing a set of microcontroller modules and serial communication between the modules. The type of serial communication will decide the maximum distances between the different modules.

Figure 31.4 shows the general structure of a measurement system containing the following devices: 1) the sensing device; part of the sensor device, 2) the signal conditioning device; normally part of the sensor device, 3) the signal processing device; normally part of the DAQ system, and 4) the data presentation device; normally part of the computer system.

The signal processing device will be the DAQ system collecting the values from the sensors, most often analog signals, converting these signals to digital representation of these analog signals, and may be some preprocessing of the digital signals as well. An overview of the main components for the analog input section of a DAQ system, where the analog to digital conversion (ADC) is part of the DAQ, is shown in Figure 31.5.

The DAQ system, with an ADC, will normally consists of the following components:

1. ADC; an important part of the DAQ system converting the analog signal to a digital representation of the analog signal. The ADC is often an expensive component so the DAQ system normally only have one ADC.

2. Mux; the multiplexer is used to be able to connect more sensors to only one analog to digital converter ADC, often $4, 8, 16$, or 32 inputs connected to one ADC.

3. μC, microcontroller used for controlling the multiplexer and the ADC.

   (a) The multiplexer must be connected to the right sensor,

   (b) the ADC must be told to start the conversion. The conversion from analog to digital will always take some time (μs or ms),

   (c) the ADC will inform the microcontroller when the conversion is finished,

   (d) the converted digital value will be read from the ADC,

   (e) select the next channel of the multiplexer.

4. Signal condition; some additional conversion of the digital value.

The conversion from analog to digital is called sampling, reduction of a continuous signal to a discrete signal. Sampling means to get a value at specific time in the time domain. Figure 31.6 shows the conversion from a analog continuous signal to a discrete signal using a MUX and a ADC.

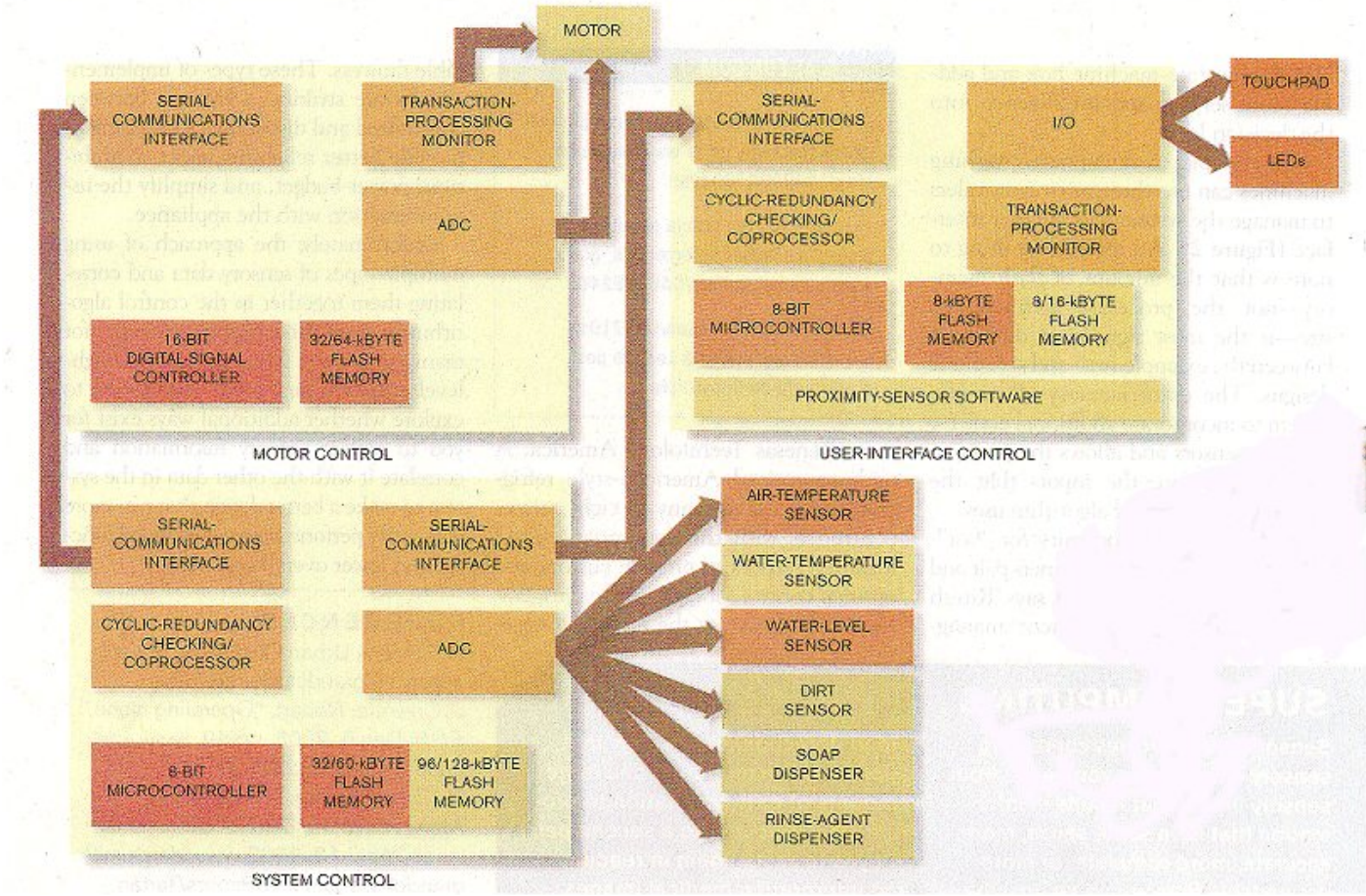


Figure 31.3: The system blocks of a distrubuted system containing sensors, measurement system, control system, and MMI (Man Machine Interface) (Cravotta 2008).

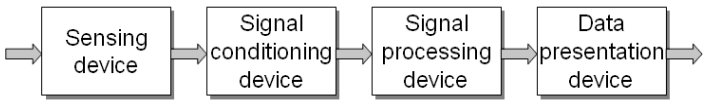


Figure 31.4: General structure of a measurement system for a single sensor device (Bentley 2005).

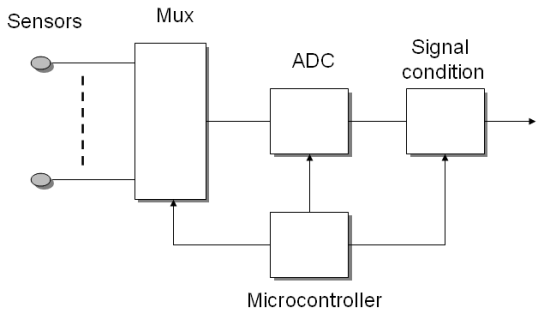


Figure 31.5: An overview of the main components for the analog input section for a DAQ system for reading sensor device values.

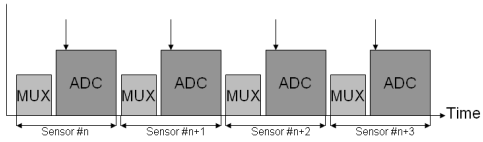


Figure 31.6: The conversion of the continuous analog signal to a discrete signal at a specific time. The arrows on the top indicate the specific times for each sensor.

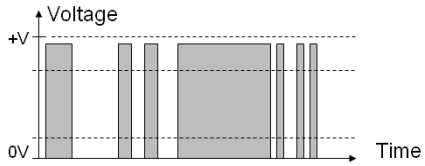


Figure 31.7: The electrical representation of digital information in a computer.

31.2 Digital representation of numbers

Numbers used by human are normally represented in base 10 (decimal), but is not practical for a computer. A more practical base for computers is base 2 as this base can be represented as a signal as voltage or not. A voltage will indicate a "1" and no voltage (0 V) will indicate a "0". See Figure 31.7 for the electrical levels where $+V$ indicate a "1" and $0V$ indicate a "0". The dotted lines are the limit voltages for detecting a "1" or a "0".

31.2.1 Integers

A number can then be represented by a set of voltages and the conversion between the base 10 and base 2 will be (in the range $[0, 256]$):

$$N_{10} = b_7\left(2^7\right) + b_6\left(2^6\right) + b_5\left(2^5\right) + b_4\left(2^4\right) + b_3\left(2^3\right) + b_2\left(2^2\right) + b_1\left(2^1\right) + b_0\left(2^0\right)$$

where N is the number in base 10 and b_n is the different bit numbers for base 2. The highest bit (b_7) is called the Most Significant Bit (MSB) and the lowest bit (b_7) is called the Least Significant Bit (LSB). The highest bit will be the same as the word size of the computer, most used is 8 bits, 16 bits, 32 bits, or 64 bits. It is common to break long binary numbers up into segments of 8 bits.

Example 1 *Convert the binary number* 00100101_2 *to a digital number. The result is* 37_{10}

The conversion from a number of base 10 to a number of base 2 is dividing the number of 2 and using the remainder as the base 2 numbers.

Example 2 *The* 8 *bit representation of* 101_{10} *will be:*

Number	**Remainder**	**Bit value**	**Value**	**Bit**	
101/2					
50/2	1	1	1	0	LSB
25/2	0	2		1	
12/2	1	4	4	2	
6/2	0	8		3	
3/2	0	16		4	
1/2	1	32	32	5	
0	1	64	64	6	
0	0	128		7	MSB
			101		

The result is the remainders starting with LSB given that $101_{10} = 01100101_2$. These numbers can represent only positive decimals integers, how to represent negative decimal integers?

Negative numbers are normally represented by 2's compliment meaning that a 8 bits integer will represent the range $[-128, 127]$ instead of the range $[0, 256]$. The way of converting a negative decimal number using 2's compliment is:

1. convert the integer to binary as if it were a positive integer,
2. invert all the bits, change all "0" to "1" and all "1" to "0",
3. add 1 LSB to the final result.

Example 3 *The* 8 *bit binary representation of* -101_{10} *and* -6_{10} *will then be:*

	-101_{10}	-6_{10}
1	$101_{10} = 01100101_2$	0110_2
2	10011010	1001
3	10011011	1010

The MSB is normally the indication of positive or negative decimal number, the number is negative if MSB="1", and positive if MSB="0".

An integer of 8 bits is limited to the range of $[-128, 127]$ or $[0, 255]$ depening on signed or unsigned integers. Larger ranges require more bytes, and short integers and long integers can be used. Short integers are often limited to 16 bits and long integers are limited to 32 bits, but may depend on the CPU architectures. A 64 bits CPU architecture may have different limitation than a 32 bits CPU architecture. The CPU architectures also differ in the order of the bytes in mulitbyte integers. The orders can be either "Little Endian" or "Big Endian".

Little Endian means that the low order byte is stored in the first (lowest) address and the high order byte in the last (highest) address. Big Endian means that the low order byte is stored in the last address and the high order byte in the first address. Intel CPU architectures (used in most PCs) are using Little Endian architecture, while Motorola CPU architectures (used in many MACs) are using Big Endian architecture.

A long integer may consist of four bytes, byte 0, byte 1, byte 2 and byte 3. Byte 0 is the low order byte and byte 3 is the high order byte. The long integer value will be:

$$value_{long} = Byte_0 + 256 * (Byte_1 + 256 * (Byte_2 + 256 * (Byte_3)))$$

but these bytes will be stores differently depending on the Endian architecture. The address positions in memory will be:

Endian	Adr#0	Adr#1	Adr#2	Adr#3
Little	Byte 0	Byte 1	Byte 2	Byte 3
Big	Byte 3	Byte 2	Byte 1	Byte 0

Exchanging information between different architectures are taken care of in protocols, but what about binary files? A short number consisting of 1 in MSB and 0 in LSB gives 256 on a Little Endian architecture and 1 on a Big Endian architecture.

Using 2's compliment we can represent positive and negative integers, but what about floating points?

31.2.2 Floating numbers

Floating points are represented by separating 3 parts of the number, the sign, the mantissa, and the exponent. The number

$$-4.28^{3.2}$$

will have a negative sign, 4.28 will be the mantissa (fraction), and 3.2 will be the exponent. See Figure 31.8 for the separation of the different bits in a floating point number.

Floating points can be represented by single or double precision and both representations are standardized from IEEE (IEEE 754).

The data for the single and double precision are listed below:

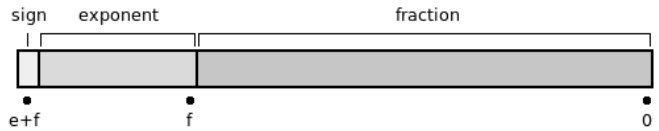


Figure 31.8: The separation of the sign, mantissa, and the exponent in floating point number.

	Single		**Double**	
Size in bits	32		64	
Sign bit	1	31	1	63
Exponent bits	8	$23-30$	11	$52-62$
Mantissa bits	23	$0-22$	52	$0-51$
Value	$\approx \pm 2.5 \times 10^{38}$		$\approx \pm 1.8 \times 10^{308}$	

The exponent is biased by $(2^{N-1}) - 1$, where N is the number of bits for the exponent. This means that for single precision will the exponent be adjusted by 127 . If the exponent is 6, will the biased exponent be $6 + 127 = 133$ and the mantissa will be adjusted according to the exponent.

31.3 ASCII codes

The computer is using binary signals, but base 2 numbers will be large numbers. Instead base 8, octal, or base 16, hex, are used. In octal groups of 3 and 3 bits are used and in hex groups of 4 and 4 bits are used as a number.

The conversion between base 2, 8, 10, and 16 for the range $[0_{10}, 15_{10}]$:

Base	**2**	**8**	**10**	**16**
	0000	00	00	00
	0001	01	01	01
	..	..	..	..
	0111	07	07	07
	1000	10	08	08
	1010	12	10	A
	..	..	..	..
	1111	17	15	F

When computers are communicating with the outside world there must some sort of protocol[1] for defining the conversion between binary numbers and numbers for the outside world. These numbers will be different from the binary numbers and the most common code is ASCII, American Standard Code for Information Interchange. In this code 8 bits represent 256 characters. Today Unicode is used to extend the ASCII code for different types of languages and type of media. An example of the ASCII codes are [Character to Base 10 Code] :

Character	Base 10	Character	Base 10	Character	Base 10
<Space>	32	0	48	A	65
!	33	1	49	..	..
(	40	..	..	Z	90
)	41	8	56	..	..
+	43	9	57	a	97
-	45	..	..	..	..

The codes 0-31 are special control characters.

[1] Protocol: A set of rules (hardware and/or software) defining how to exchange the information between two systems.

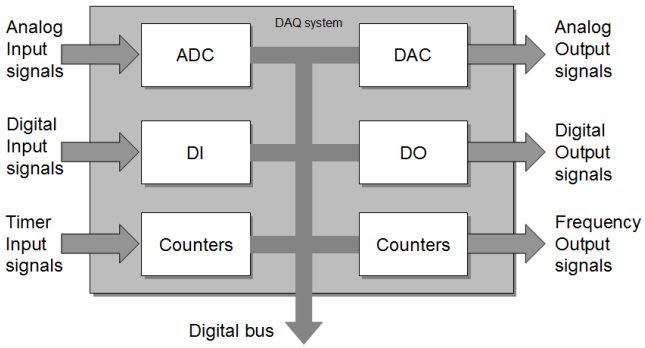


Figure 31.9: The input and output sections of a DAQ system. The left side is the input section (analog and digital), the right part is the output section (analog and digital), and the lower section in the middle is the computer I/O (digital bus).

31.4 DAQ parts

The main parts of a DAQ system will be the the analog to digital converter (ADC) as shown in Figure 31.5. A DAQ system may also contain support for digital inputs and outputs and counters as well, see Figure 31.9 for a complete DAQ system.

31.4.1 Counters

Counters are input and output signals that can be used for timing purposes. The input counters can be used to count a number pulses or measuring the time between any changes of the input signals. These signals are often connected to internal counters or the interrupt system. The output counters can be used for generating frequency signals.

31.4.2 Digital inputs

Digital input signals are only ON/OFF signals and often will the number of digital input signal be a multiply of the data width for the system. This means that the number of digital inputs can for example be $8, 16, 24$, or 32. Digital inputs can be used if the process change is a change between two states. This should be converted to an electrical signal being on or off.

Another type of digital input useful in data acquisition applications is the hardware trigger. This allows an external event, often as an interrupt signal to the system.

Important aspects of digital inputs are the:

1. input range of the digital inputs,
2. input current,
3. noise protection.

The 0 V (or ground) signal is a good references and by using pull-up resistors and a diode, only the 0 V signal can be used as the only reference. A capacitor can also be used for noise reduction and/or for bouncing. An example of a digital input is shown in Figure 31.10.

31.4.3 Digital outputs

Digital output signals are just like the digital input signals, only ON/OFF signals. Digital outputs are used to control equipment that are going to turned only ON and OFF. Often these outputs will control relays that can control any type of signal. A relay can be used both as Normally Open (NO) and Normally Closed (NC) devices. Important aspects of digital outputs are the:

1. output range of the digital outputs,
2. output current.

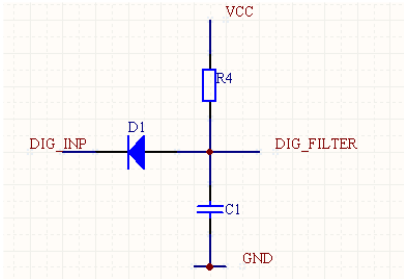


Figure 31.10: A digital input using only ground (GND) as reference, the external input will be independent of the voltage in your measurement system. The input also contains a high frequency filter.

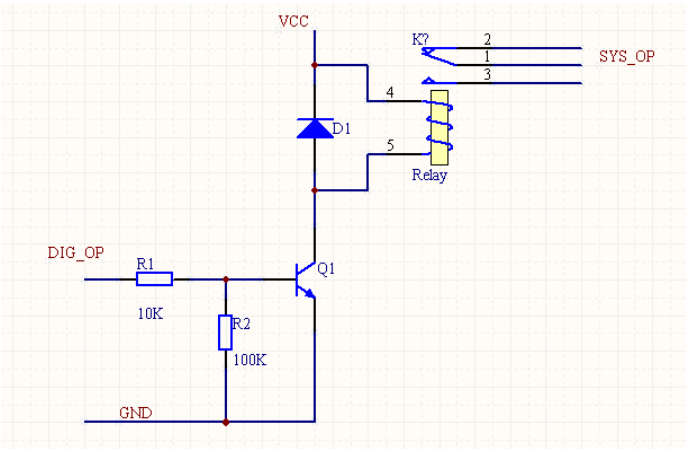


Figure 31.11: A digital output using a relay, a diode for protection, and a transistor for amplification of the digital output signal from your measurement system.

An amplifier can be used to amplify the output current for controlling the relay. This amplifier can be a digital inverter or a single transistor. However, always remember to add a diode with the relay. An example of a digital output with an amplifier, a diode, and a relay is shown in Figure 31.11.

This Figure also shows the usage of a transistor for amplification of the digital output signal from your measurement system as these signals often have low current driving capacity. The digital output is often $5 - 25\,\text{mA}$, while a relay often need $100 - 500\,\text{mA}$ to operate.

The relay is an inductor and may create problems when the transistor (or any other switch) is turning the current off. The voltage across the inductor in the relay is:

$$v = L\left(\frac{di}{dt}\right)$$

where L is the inductance of the relay and i is the current flowing in the inductor. When the current is switched off, the voltage across the actuator can become very high during the switching phase (Olsson & Piani 1998). The diode in Figure 31.11 is used to reduce the voltage spikes. Figure 31.12 shows the current in the actuator to the left and the voltage across the actuator to the right.

31.4.4 Digital to Analog Converters

In a process system the output of control values can be just as important as input for sensor signals and the DAQ system can be used for both input of analog values and output of analog values.

A Resistor Ladder, or R-2R Ladder is the most simple and inexpensive way to perform digital-to-analog conversion, using repetitive arrangements of precision resistor networks in a ladder-like configuration (www.wikipedia.org 2006). Other types of DAC exists as well, but normally more simple than a ADC. A resistor ladder DAC is shown in Figure 31.13

The digital inputs or bits (Bit 0 to Bit 4) range from the most significant bit (MSB) to the least significant bit (LSB). The bits are switched between either 0V or VREF and depending on the state and

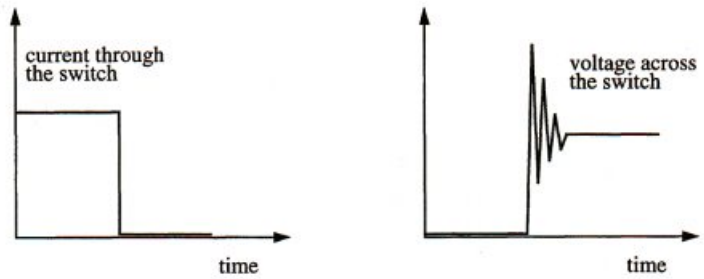


Figure 31.12: The current and voltage in an inductive actuator when turned off (Olsson & Piani 1998).

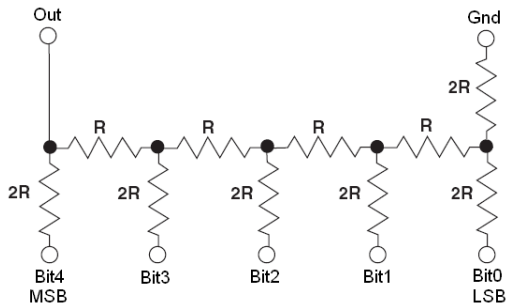


Figure 31.13: A resistor ladder for a digital to analog converter.

location of the bits OUT will vary between 0V and VREF. VREF will be the same voltage as for a logic 1. See Figure 31.14.

A detailed output signal of the DAC is shown in Figure 31.15. The output will consists of small steps and the number of steps will depends on the resolution given from the number of bits.

The output voltage from the DAC will be:

$$V_o = \frac{V_{MAX} - V_{MIN}}{2^N} \cdot D_v$$

where $[V_{MIN}, V_{MAX}]$ is the output range of the DAC, D_v is the digital output value and N is the number oif bits for the DAC. An example; $D_V = 128, N = 8, V_{MAX} = 5\,\mathrm{V}, V_{MIN} = 0\,\mathrm{V}$ gives $V_O = 2.5\,\mathrm{V}$. The voltage range defines if it is a unsigned value or signed value. A signed value may have a negative V_{MIN} value.

DAC specifications:

1. Settling time: Period required for a D/A converter to respond to a full-scale set point change.
2. Linearity: This refers to the device's ability to accurately divide the reference voltage into evenly sized increments.
3. Range: The reference voltage sets the limit on the output voltage achievable.
4. Output control: amplifiers and signal conditioners often are needed to drive a final control element.
5. Output filter: A low-pass filter may also be used to smooth out the discrete steps in output.

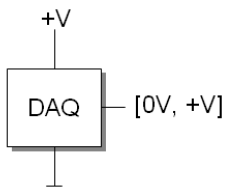


Figure 31.14: The output range of a DAC in a DAQ system.

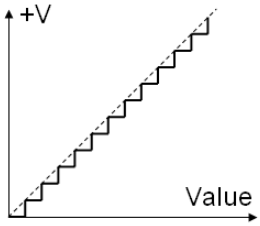


Figure 31.15: The output of a DAC device. The steps are due to the bit levels and can be smoothed by using a RC filter.

31.4.5 Analog to Digital Converter

The analog to digital converter is used for converting the analog signal from the output of the MUX to a digital representation of the analog signal. A set of challanges arise when doing the analog to digital conversion like:

1. the digital conversion will not be an exact representation of the analog signal,
2. the conversion will take some time so the analog signal must be stored while converting.

The representation of the analog signal will seldom be exact due to the resolution. Assuming an analog signal in the range of $[0V, 10V]$ and only 2 bits resolution of the system. 2 bits is $2^2 = 4$ numbers $(0, 1, 2, 3)$ so the analog voltage range must be divided into 4 steps giving $\frac{10V-0V}{4} = 2.5V$ for each step. The conversion will the be:

Value	**LSB**	**MSB**	**Number**
$[0V, 2.5V]$	0	0	0
$[2.5V, 5V]$	1	0	1
$[5V, 7.5V]$	0	1	2
$[7.5V, 10V]$	1	1	3

In general will the output of an ADC have 2^N possible values where N is the number of bits used for conversion in the ADC. An ADC will have 8, 12, 14, 16, 18, 20, 22 or 24 bits. The resolution for a converter will be:

$$R = \frac{High - Low}{2^N}$$

ADC can vary widely, but there is 4 important properties for the converter:

1. the number of bits used for conversion. The greater the number of bits, a more accurate presentation of the analog input (8/10/12/14/16/18/20/24 bits),
2. the input range. The input range can be unipolar ($0V$ to $+10V$ or $-5V$ to $0V$) or bipolar ($-5V$ to $+5V$),
3. the reference voltage of the converter. This voltage will be part of the accuracy of the converter (often given as ppm/ °C),
4. the conversion speed, the time for converting the analog input to a digital representation (μs or ms).

The Figure 31.16 shows the details of an A/D converter consisting of an analog section and a digital section. The analog signal is feed to the converter, the converter is started by a *Start Conversion* signal. The converter will report the *Conversion Finished* signal when the digital representation of the analog value is available in the digital section of the converter. The *Read Value* signal is used for reading the digital value from the converter. The reference voltage is an important property for the accuracy of the A/D converter. On some converters this will be controlled by internal logic, on other converts will this be external logic.

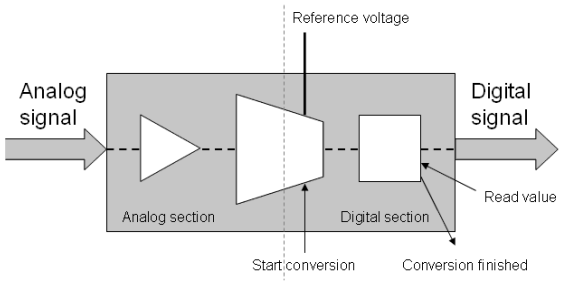


Figure 31.16: The details of an Analog to Digital Converter.

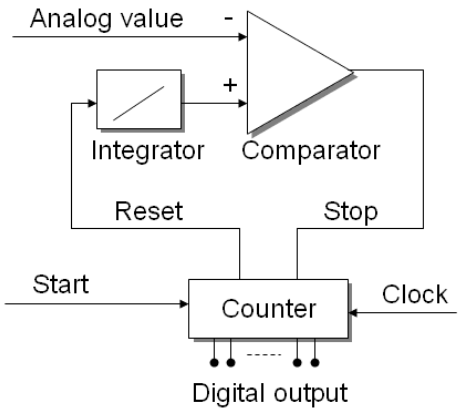


Figure 31.17: The principles for a unipolar single-slop analog to digital converter (Wheeler & Ganji 2004).

Different types of ADC exist, lets use a unipolar single-slope integrating converter to demonstrate the analog to digital conversion process. A block diagram of the converter is shown in Figure 31.17 with the analog input signal on the top left side.

A Sample and Hold (S/H) part is needed for the Analog value to keep the analog value constant while doing the conversion. A start signal, lower left side, will start the conversion. The start signal will:

1. set the "lock" of the input signal as the input signal must be constant during the conversion,
2. reset the digital output of the counter,
3. reset the integrator,
4. reset and start the counter using the clock input.

The clock signal is used for the counter to let the digital output be a presentation of the analog value. In parallel with the counter will the integrator start rising a voltage and the output from the integrator is compared with the analog value. When the integrator has a higher voltage than the analog voltage, the counter will stop and the digital representation of the analog value can be read from the digital output of the counter.

The comparator will normally contain a "sample and hold" on the analog value input to keep the analog value at a constant value while conversion is performed. The conversion time will depend on the speed of the integrator and often there is a relation between the speed and resolution as shown:

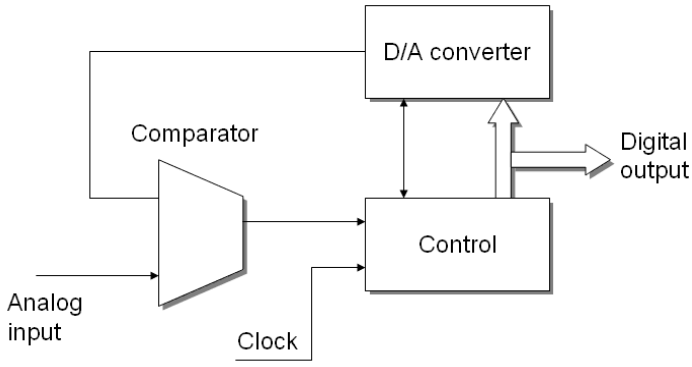


Figure 31.18: The principle of a successive approximation A/D converter.

Speed	**Low**	**Medium**	**High**
Precision			
Low			Interpolation Folding
Medium		Successive approximation Algorithm	
High	Integration Oversampling Sigma Delta		

The most used types of ADC are:

1. Sigma-Delta ADC,
2. successive-approximation ADC.

ADC converter types

Successive-approximation A successive-approximation ADC is the most common used type of analog to digital converters. A DAC is used for converting a digital representation to a analog value. A DAC will be very simple compared to a ADC. The principle of successive approximation is shown in Figure 31.18, using a comparator, DAC, and a control module including a binary counter. The main different from the single-slop integrator is the usage of the DAC instead. The comparator also includes a sample and hold circuit (S&H) to hold the input value while searching. The binary counter starts with the most significant bit (MSB) and works towards the least significant bit (LSB) using the clock input, giving a fast conversion. The DAC converts the output of the binary counter to an analog value which is compared with the analog input value in the comparator. When these analog values are approximately equal, the counter stops and the binary counter value is available as the digital output value.

This design offers an effective compromise among resolution, speed, and cost. In this type of design, an internal DAC converter and a single comparator are used to narrow in on the unknown voltage by turning the bits in the DAC until the voltages match within the least significant bit. Raw sampling speed for successive approximation converters is in the range of 50 kHz to 1 MHz.

Sigma-delta A sigma-delta ADC uses a 1-bit DAC, filtering, and oversampling to achieve very accurate conversions. The conversion accuracy is controlled by the input reference and the input clock rate.

The primary advantage of a sigma-delta converter is high resolution. The flash and successive approximation ADCs use a resistor ladder or resistor string. The problem with these is that the accuracy of the resistors directly affects the accuracy of the conversion result. Although modern ADCs use very precise, laser-trimmed resistor networks, some inaccuracies still persist in the resistor ladders. The sigma-delta converter does not have a resistor ladder but instead takes a number of samples to converge on a result.

The primary disadvantage of the sigma-delta converter is speed. Because the converter works by oversampling the input, the conversion takes many clock cycles. For a given clock rate, the sigma-delta

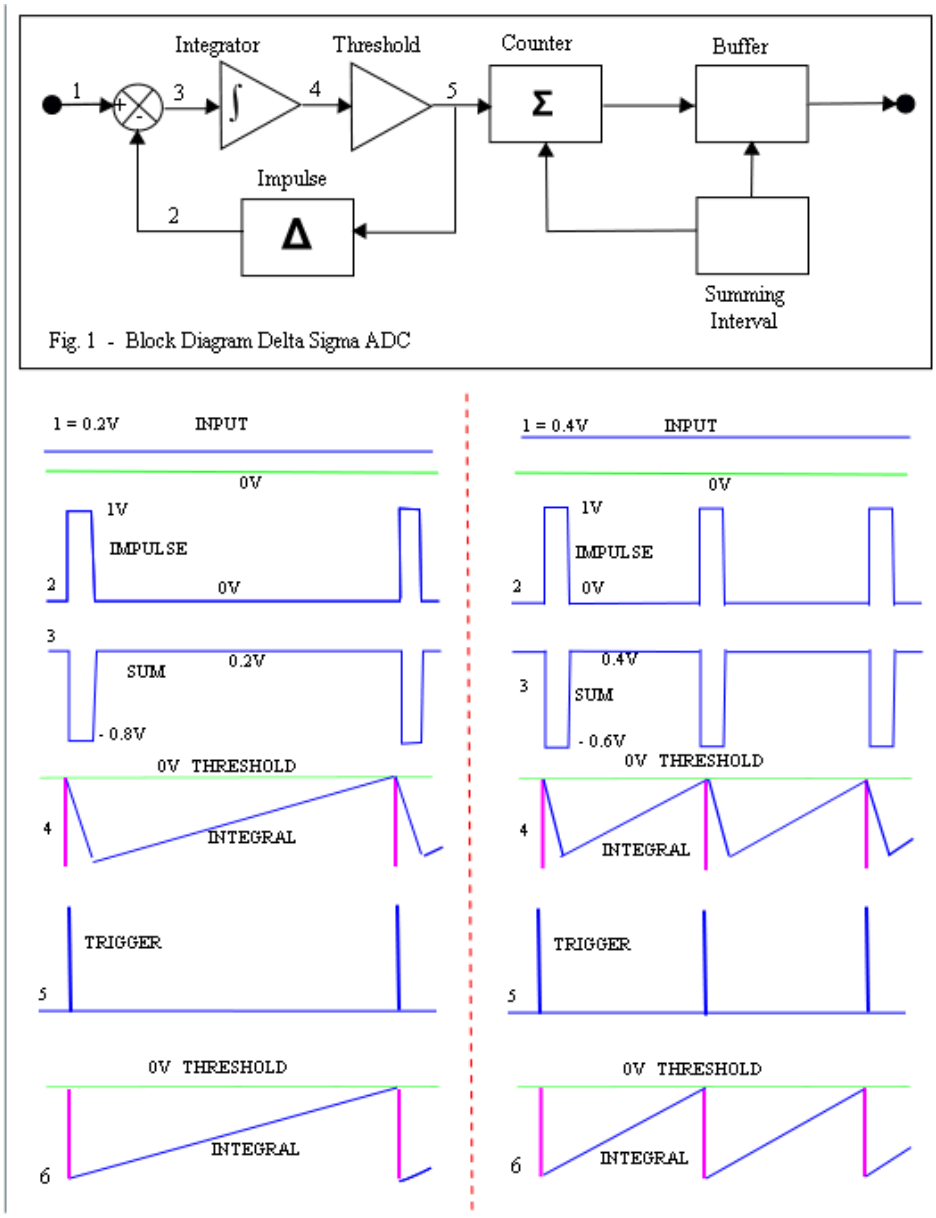


Figure 31.19: A simplified Sigma delta ADC with examples of the signal levels.

converter is slower than other converter types. Or, to put it another way, for a given conversion rate, the sigma-delta converter requires a faster clock. Another disadvantage of the sigma-delta converter is the complexity of the digital filter that converts the duty cycle information to a digital output word.

Figure 31.19 shows a simplified Sigma Delta ADC.

Figure 31.20 shows a sigma delta converter from Analog Devices. The A/D converter, AD7190, consists of 4 analog inputs, a multiplexer, a sigma delta A/D converter and a signal condition unit converting the digital representation for serial communication of the data. The device also contains a temperature sensor that can be used as an extra input for temperature compensation.

Integrating This type of A/D converter integrates an unknown input voltage for a specific period of time, then integrates it back down to zero. This time is compared to the amount of time taken to perform a similar integration on a known reference voltage. The relative times required and the known reference voltage then yield the unknown input voltage. Integrating converters with 12 to 18-bit resolution are available, at raw sampling rates of 10-500 kHz.

Because this type of design effectively averages the input voltage over time, it also smooths out signal noise. And, if an integration period is chosen that is a multiple of the AC line frequency, excellent

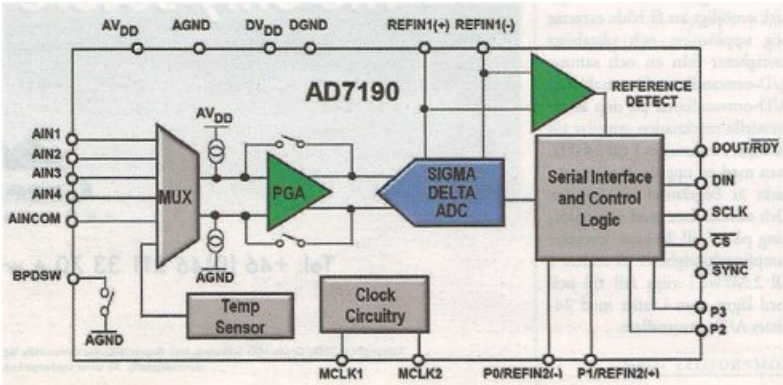


Figure 31.20: The block diagram of a Sigma Delta analog to digital converter, the AD7190 from Analog Devices (www.analog.com; FEB-09).

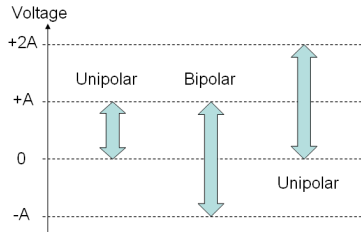


Figure 31.21: The difference of unipolar and bipolar signals

common mode noise rejection is achieved. More accurate and more linear than successive approximation converters, integrating converters are a good choice for low-level voltage signals.

The solution with the integrator is using more time of the analog signal is closer to the high range, while the successive approximation is using a kind of binary search and the time will not depend that much of the analog signal.

ADC converter properties

Polarity A DAQ system can convert either unipolar or bipolar signals, or both. A unipolar signal contains only zero and positive values. A bipolar cotains both zero, negative, and positive values. The input devices will decide the type of signals and the DAQ system should adopt to the type of signals to be able to exploit the resolution of the A/D converter. Figure 31.21 shows the difference of unipolar and bipolar signals.

Range In bipolar converters 2's complement is used, starts with a binary number of $\frac{2^N}{2}$ at the lower end, 0 in the middle, and $\frac{2^N}{2} - 1$ as the top range. The output of a 2's complement A/D converter will then be:

$$D_O = int\left\{\frac{V_i - V_{rl}}{V_{ru} - V_{rl}} \cdot 2^N\right\} - \frac{2^N}{2}$$

where V_i is the analog input voltage, V_{ru} is the upper value of input range, V_{rl} is the lower value of input range, N is the number of bits, and D_O is the digital output.

A polar converter will have the output:

$$D_O = int\left\{\frac{V_i - V_{rl}}{V_{ru} - V_{rl}} \cdot 2^N\right\}$$

Example 4 *If the input voltage is $2V$, the range is $[-5V, +5V]$, and the number of bits are 8, will the digital output be:*

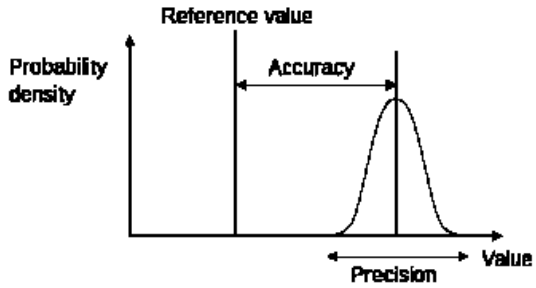


Figure 31.22: The relationship between accuracy and precision ((www.wikipedia.org 2010)).

$$D_O = int(\frac{2-(-5)}{5-(-5)} \cdot 2^8) - \frac{2^8}{2} = int\left(\frac{7 \cdot 256}{10}\right) - 128 = 179 - 128 = 51$$

Resolution Central to the performance of an A/D converter is its resolution, often expressed in bits. An A/D converter essentially divides the analog input range into 2^N bins, where N is the number of bits.

Since the output of an ADC changes in discrete steps (one LSB) will there be a resolution error, also known as the quantizing error:

$$\pm 0.5 LSB$$

The input resolution error will then be:

$$\pm 0.5 \frac{V_{ru} - V_{rl}}{2^N} V$$

The input resolution error for the example above will be (Range $[-5, 5]$ and 8 bits):

$$\pm 0.5 \left(\frac{10}{256}\right) V = \pm 19.5\,\mathrm{mV}$$

The "standard" resolution[2] of AD converters exists from about 12 bits to about 22 bits depending on the price and the conversion time.

Resolution, precision, and accuracy are often mixed. The difference between resolution and precision; resolution is the fineness to which an instrument can be read and precision is the fineness to which an instrument can be read repeatably and reliably. This means that the difference between resolution and precision is repeatability. The difference between precision and accuracy is correctness. See the Figures 31.22, 31.23 and 31.24.

Reference Voltage A voltage reference is an electronic device (circuit or component) that produces a fixed (constant) voltage to the ADC irrespective of the loading on the device, power supply variation and temperature. It is also known as a voltage source, but in the strict sense of the term, a voltage reference often sits at the heart of a voltage source (www.wikipedia.org 2006). Voltage references are used in ADCs and DACs to specify the input or output voltage ranges. The accuracy of the referance voltage will have an impact of the conversion accuracy of the ADC or DAC.

Single-Ended and Differential Inputs Another important consideration when specifying analog data acquisition hardware is whether to use single-ended or differential inputs. In short, single-ended inputs are less expensive but can be problematic if differences in ground potential exist (www.wikipedia.org 2006).

In a single-ended configuration, the signal sources and the input to the amplifier are referenced to ground. This is adequate for high level signals when the difference in ground potential is relatively small. A difference in ground potentials, however, will create an error-causing current flow through the ground conductor otherwise known as a ground loop (www.wikipedia.org 2006). The input voltage is measured with reference to ground and compared against the reference voltage.

[2] as of 2008

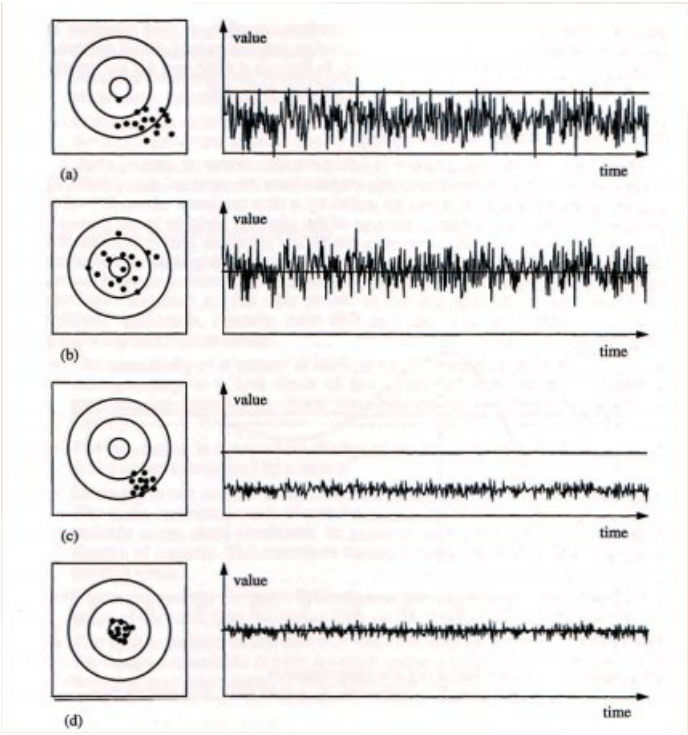


Figure 31.23: The relationship between precision, accuracy, and the signal in the time domain (Olsson & Piani 1998).

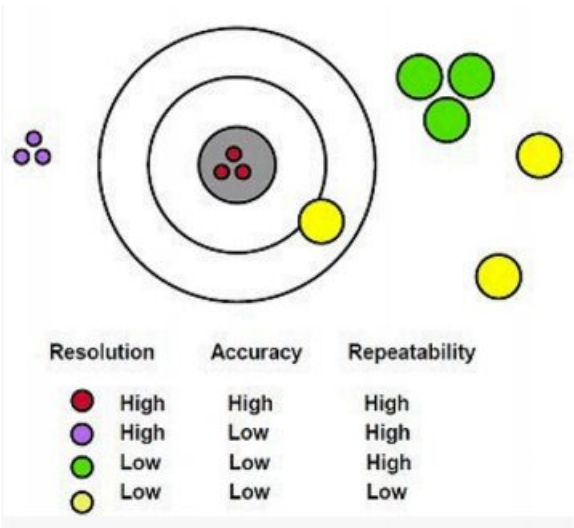


Figure 31.24: The relationship between the resolution, accuracy and repeatability. The repeatability can be high even if the resolution or the accuracy is low (www.keithley.com: Oct-10).

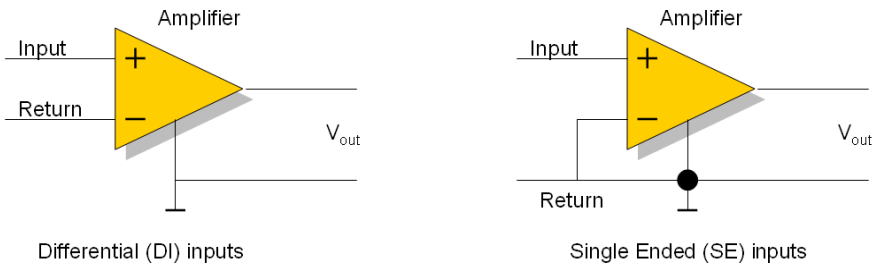


Figure 31.25: The difference between differential inputs and single ended inputs using an amplifier.

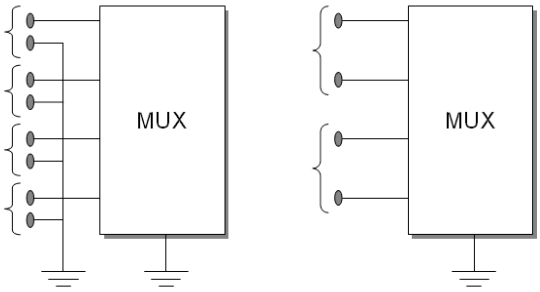


Figure 31.26: The usage of SE or DI inputs in a DAQ system. Single Ended (SE) to the left and Differential Inputs (DI) to the right.

Differential inputs, in contrast, connect both the positive and negative inputs of the amplifier to both ends of the actual signal source. Any ground-loop induced voltage appears in both ends and is rejected as a common-mode noise. The downside of differential connections is that they are essentially twice as expensive as single-ended inputs; an eight-channel analog input board can handle only four differential inputs (www.wikipedia.org 2006). The input voltage is measured as the difference between the input lines and compared against the reference voltage.

Figure 31.25 shows the difference of these inputs for an amplifier showing that DI inputs is not using the ground as reference for the input signals. DI inputs will also require more input connections as each input connection has a separate input and return connection. The input signals to the DAQ system is first connected to the multiplexer. The number of connection will depend if DI or SE inputs are used, normally the number of SE inputs will be twice the number DI inputs. The inputs can either be DI or SE inputs, it is not possible to mix a number of SE and DI inputs on the same DAQ device. If one of the inputs need to be connected as DI input, all the inputs must be DI inputs. See Figure 31.26.

The advice is to use DI inputs if:

- the input signal has a low level, normally less than 1 volt,
- the wires connecting the signal is longer than 3 meters,
- one of the input signals is using a reference different from the ground reference.

Number of channels It is important to acknowledge that a multiplexer does reduce the frequency with which data points are acquired, and that the Nyquist sample-rate criterion still must be observed. During a typical data acquisition process, individual channels are read in turn sequentially. This is called standard, or distributed, sampling. A reading of all channels is called a scan. Because each channel is acquired and converted at a slightly different time, however, a skew in sample time is created between data points.

Scaling Because A/D converters work best on signals in the 1-10 V range, low voltage signals may need to be amplified before conversion-either individually or after multiplexing on a shared circuit. Conversely, high voltage signals may need to be attenuated.

Amplifiers can also boost an A/D converter's resolution of low-level signals. For example, a 12-bit A/D converter with a gain of 4 can digitize a signal with the same resolution as a 14-bit converter with a

gain of 1. It's important to note, however, that fixed-gain amplifiers, which essentially multiply all signals proportionately, increase sensitivity to low voltage signals but do not extend the converter's dynamic range.

Programmable gain amplifiers (PGAs), on the other hand, can be configured to automatically increase the gain as the signal level drops, effectively increasing the system's dynamic range. A PGA with three gain levels set three orders of magnitude apart can make a 12-bit converter behave more like an 18-bit converter. This function does, however, slow down the sample rate.

Range, Gain and Measured Precision The range of the DAQ system can often be configured to different gains like $[-10\,\mathrm{V}, +10\,\mathrm{V}]$, $[-5\,\mathrm{V}, +5\,\mathrm{V}]$, $[-2\,\mathrm{V}, +2\,\mathrm{V}]$, and $[-1\,\mathrm{V}, +1\,\mathrm{V}]$. The precision will depend on the output signal of the sensor device and the corresponding input range of the DAQ system. The input range should be as close as possible to sensor signal to have as good precision as possible.

Software calibration Sometimes you need an accurate reference, more accurate than the product cost will support. When manual adjustment is out of the question, the software can compensate for reference voltage variations. This is typically done by providing a known, precise input, which is used to calibrate the ADC. This reference can be very precise (and very expensive) because only a few are needed for the production line.

31.4.6 Transfer of A/D conversion to system memory

The A/D converter will use some time to convert the analog signal to a digital representation, and normally the A/D converter will inform the controller when the conversion has finished. The controller will read the converted value from the A/D converter and write the value to a specific location in the system memory. A First In First Out (FIFO) buffer can be part of the signal condition unit reading the converted values from the A/D converter.

The transfer from the A/D converter or the FIFO buffer to the system memory can be done in different ways. These are:

1. polling; The controller is waiting for the A/D controller to finish the conversion. This is seldom used as it is a waste of the controller usage,

2. interrupt; The controller will get a signal from the A/D converter every time the operation has finished. The controller will enter an interrupt function only when informed from the A/D converter,

3. Direct Memory Access (DMA) transfer; The A/D converter needs a memory interface (DMA controller) and will transfer the converted data to system memory without interference of the controller. The A/D converter (DMA controller) and the controller can not use the memory at the same time, normally the A/D controller (DMA controller) will have the priority and the controller must wait for any memory access when in use by the A/D controller. The controller can however continue with any other tasks, without accessing the system memory.

31.5 Range check of signal values

Reading values from sensors can give good, bad, wrong, or illegal values. It is important to have some sort of value checks of the sensor device signals. Some of these value checks are (Pettersen 1984):

1. limit checks; checking that the sensor value is within the valid range of the sensor,

2. validation checks; checking that the sensor value is within a "window" of the last value,

3. redundancy checks; checking with other sensors.

Limit checks and validation checks can always be used, as they can be included in the software. The usage of limit checks and validation checks is shown in Figure 31.27.

These checks are important for time series data as the fact that time is unidirectional can be used as quality constrains. This can be for time collision of time samples, samples that are out of range, the rate of change is too high or missing samples as shown in Figure 31.28.

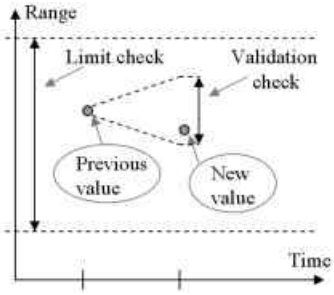


Figure 31.27: The usage of limit checks and validation checks for a value read from a sensor (Skeie 2008).

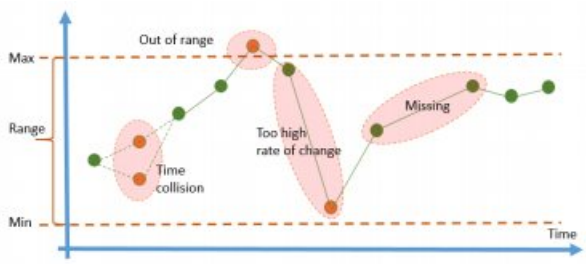


Figure 31.28: Qualtiy checks of time series data (Kadal 2018).

Chapter 32

Discrete Sampling

Digital systems record signals at discrete times and record no information about the signal between these times, see Figure 31.6. So sampling means to take snapshots of analog signals at discrete times. The time interval between the samples is kept constant in most application and is known as the sampling rate. The designer or user of such digital systems, known as DAQ system, must be aware of problems of recording at discrete times and take the right actions to get the correct information from the analog signals. The conversion from an analog value to a binary number is called quantization. The binary number consists of a number of bits used for the conversion, normally in the range of 8 to 24 bits. An analog signal will always vary giving a frequency respons. The type of measurement and type of process will influence of the frequency of signal, the frequency of a temperature signal should have a lower frequency that a pressure signal.

Figure 32.1 shows how a DAQ system will separate the analog and digital section of a measurement system, and the digital section will only contain digital representation of the analog signal at discrete time intervals. This is shown in the right part of the Figure.

The sampling time decide how often the analog signal will be read and the sampling thorem (Wheeler & Ganji 2004) states that the sampling rate f_s must be equal or greater than twice the highest frequency f_m:

$$f_s \geq 2 \cdot f_m$$

f_m is the highest frequency component of the original signal.

32.1 A/D conversion

A/D systems is called analog input subsystems as they convert analog signals to digtal signals for a computer. These analog input subsystems typically consists of eight or sixteen input channels, but only the number of channels used will influence on the sampling rate for the system. Sampling will always be a time dependent operation and the analog signal should be kept constant during the conversion period. A sample and hold (S/H) is normally used, consisting of a signal buffer, an electronic switch, and a capacitor.

The operation of the S/H section will be:

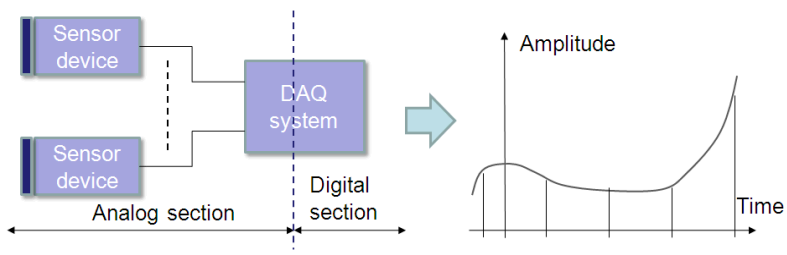


Figure 32.1: The DAQ system consists of an analog section and a digital section. The sensing part will always be analog, and somewhere in the system will the DAQ convert an analog value to a digital representation. This is shown in the right part of the figure.

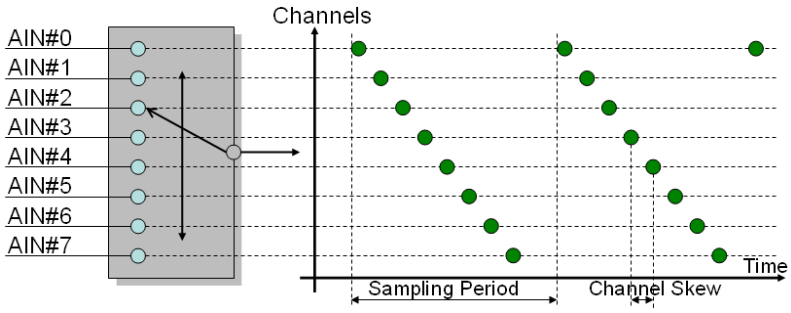


Figure 32.2: The sampling of eight consecutive analog input channels, the sampling period, and the channel skew.

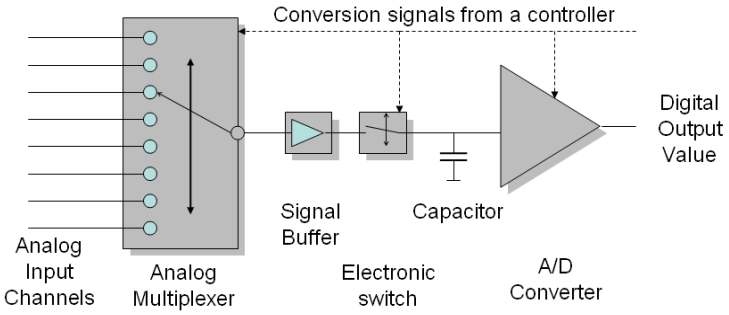


Figure 32.3: A DAQ system consisting of an analog multiplexer, a sample and hold section, and an analog to digital converter.

1. The electronic switch will connect the capacitor to the input signal through the signal buffer, the capacitor will be charged to the input voltage,
2. The electronic switch will disconnect the input signal, and the capacitor will have a "constant" voltage while the A/D converter is converting the analog voltage to a digital representation of the input signal,
3. The S/H section will be common for analog input systems with several input channels,
4. The charging is repeated for every conversion cycle of the A/D converter.

As the DAQ system contains an analog multiplexer to be able to sample signals from several channels. The A/D converter is an expensive component, while the analog multiplexer is a cheaper component. Using several channels for an A/D converter, the highest frequency of each channels will be:

$$\text{Maximum sampling rate per channel} = \frac{\text{Maximum sampling rate for AD converter}}{\text{Number of channels used}}$$

Using a multiplexer means that every input channel, or analog input signal, will be scanned after each other. This means that the sampling period will be the time from one specific input channel is converted until the same input channel is converted again. This is the reason why the maximum sampling rate will depend on the number of analog channels are used. This means that the designer of the DAQ system should only scan the analog inputs that are used, not all available input channels. The maximum sampling rate for the DAQ system will be when only one input channel is used. When several channels are used, those channels cannot be sampled simultaneous and a time gap will exists between consecutive samples channels. This time gap is called channel skew. The timing for sampling the analog channels, the sampling period, and the channel skew are shown in Figure 32.2.

An analog input subsystem will, as a minimum, consists of a multiplexer for several analog input signals, a sample and hold section, and the analog to digital (A/D) converter. These devices are interconnected as a DAQ system in Figure 32.3.

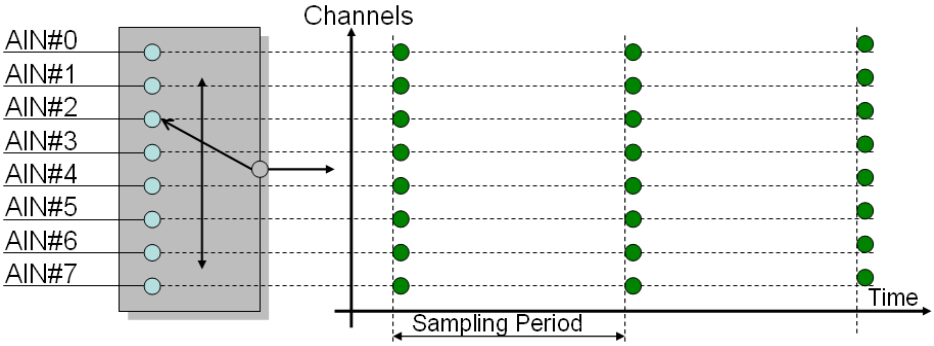


Figure 32.4: The sampling of eight consecutive analog input channels with Simultaneous Sample and Hold section.

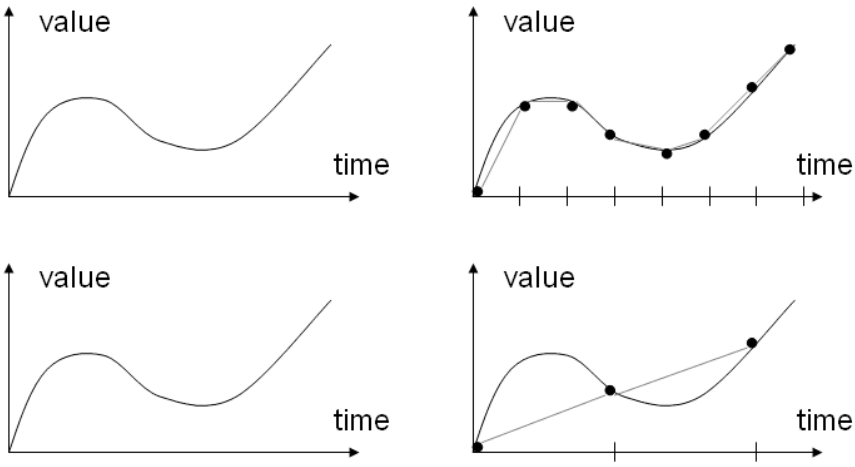


Figure 32.5: The aliases problem if the sampling frequency is too low. The upper part of the Figure shows a sampling frequency high enough, while the lower part shows a too low sampling frequency ending up with a strain line signal as the digital representation of the signal.

32.2 Simultaneous Sample and Hold

In some system the channel skew can not be tolerated, all the channels must be read at exactly the same time. Then Simultaneous Sample and Hold are used to sample all input signals at the same time, and holds the value until the A/D converter has converted all the values. Either a set of electronic switches and capacitors can be used or separate A/D converters for each input channel. Figure 32.4 shows the usage of Simultaneous Sample and Hold logic.

32.3 Aliasing

If the sampling frequency is too low, the discrete time values will not be a correct conversion of the continuous time values as shown in Figure 32.5.

This is known as aliasing. Aliasing causes frequency components that are higher than half of the sampling frequency to "fold over" and overlap with the lower frequency components. Figure **??** is showing aliasing as the sampling frequency is too low to the correct information. One way of avoiding aliasing can be by adding a hardware filter before the A/D converter for removing frequency components above the sampling frequency, see Figure 32.6.

The LP filter can be located before or after the mux. If special filtering of the different input signals is wanted, use a LP filter before the mux, but remember that every input signal needs a filter. Using a filter after the mux, only one LP filter is needed. The ADC section of a DAQ should always contain a LP filter as the ADC always has a maximum input frequency.

The design of a RC LP filter will be to define the cut off frequency for the filter, this frequency will

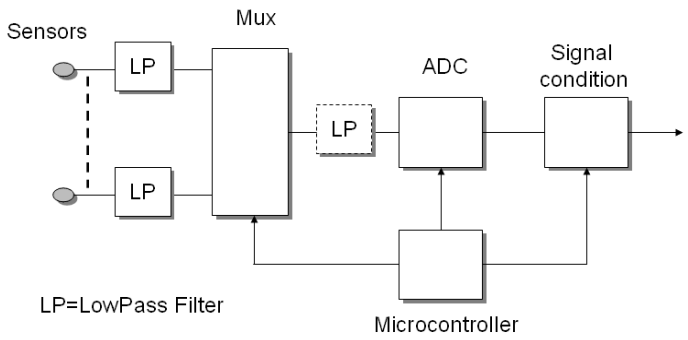


Figure 32.6: A lowpass filter used for avoiding aliasis in the measurement system.

be defined from the frequency components in the input signals and the sampling frequency of the ADC. The cut off frequency will be:

$$f_{cut-off} = \frac{1}{2 \cdot \pi \cdot \tau} = \frac{1}{2 \cdot \pi \cdot R \cdot C}$$

The filter should have a cut-off frequency and an order to reduce the amplitude of the signal to less than resolution error, this is the only way to remove these frequency components.

32.4 Oversampling

In some cases, it is desirable to have a sampling frequency considerably more than twice the desired system bandwidth so that a digital filter can be used in exchange for a weaker analog anti-aliasing filter. This process is known as oversampling and the filter will now be part of the signal condition section in Figure 32.6. Different types of software filters exist and is used for data cleaning purposes. An effective data-cleaning filter should satisfies two important properties (Pearson 2005):

1. outliers[1] should be replaced with data values that are more consistent with the local variation of the nominal sequence,
2. the filter should cause no or little change in the nominal data sequence.

Digital filter techniques include exponential, box-car averaging, moving average, and weighted moving average[2] filtering, Fourier transformation, correlation analysis, and others (Meier & Zünd 2000).

32.5 Folding diagram

It is possible to estimate the lowest alias frequency using a folding diagram. The folding diagram can be used to predict the alias frequency based on the signal frequency and the sampling rate, like:

1. Compute the folding frequency: $f_N = \frac{f_s}{2}$ where f_s is the sampling frequency,
2. Compute the folding diagram index: $\frac{f_i}{f_N}$ where f_i is the sampled signal frequency (frequency of interest),
3. Find the index $\frac{f_i}{f_N}$ in the folding diagram, draw a vertical line to the lowest line and read the folding diagram index on the lowest line (base line),
4. The alias frequency will be: folding diagram index $\cdot$ f_N.

The folding diagram is shown in Figure 32.7.

Note that the folding diagram will not make any solution to the problem, only indicate how a frequency above the sampling frequency will affect the digital representation of the analog signal.

[1] outlier: an entry in a dataset that is anomalous to the other entries in the dataset, with respect to the behavior (Pearson 2005)

[2] also called Savitzky-Golay filter or Hampel filter.

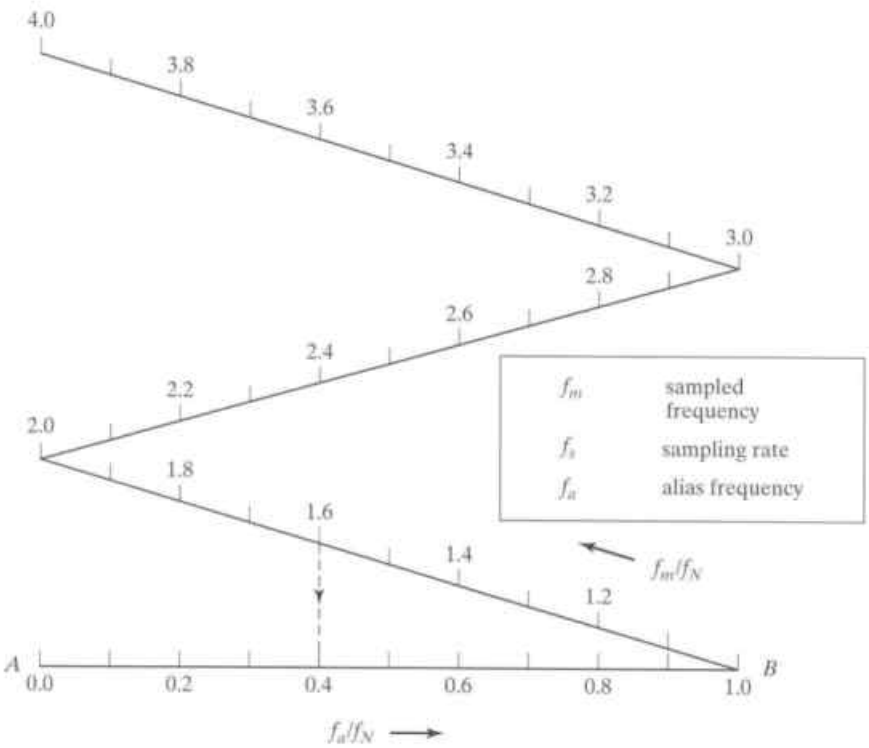


Figure 32.7: A folding diagram used for estimation of the alias frequency (Wheeler & Ganji 2004).

Example 5 *A system has a sampling frequency of* 100 Hz *and a signal frequency of interest is* 80 Hz. *As the sampling frequency is not twice the frequency of interest, the will be an aliases frequency, but where?*

1. The folding frequency: $f_N = \frac{100}{2} = 50$
2. Folding diagram index: $id = \frac{80}{50} = 1.6$
3. Index in the folding diagram $= 0.4$
4. The lowest aliases frequency will be: $f_A = 0.4 \cdot 50 = 20\,\text{Hz}$

Your measurement system will read a new frequency signal because the sampling frequency is wrong. What is the correct sampling frequency?

Example 6 *A system has a sampling frequency of* 100 Hz *and a maximum frequency of interest of* 100 Hz. *As the sampling frequency is not twice the frequency of interest, the will be an aliases frequency, but where?*

1. The folding frequency: $f_N = \frac{100}{2} = 50$
2. Folding diagram index: $id = \frac{100}{50} = 2$
3. Index in the folding diagram $= 0.0$
4. The lowest aliases frequency will be: $f_A = 0.0 \cdot 50 = 0\,\text{Hz}$

Note that the signal amplitude is not zero, only the signal frequency, a steady signal.

32.6 Selecting the sampling rate and filtering

A signal consists of a set of sine and cosine waves with different frequencies, how to sample and filter these frequency components?

1. Find the maximum frequency of the signal by analyzing the signal and/or the system. If any doubt, use FFT (Fast Fourier Transform) or DFT (Discrete Fourier Transform) on the signal if a recorded part is available,
2. Define the maximum frequency of interest,
3. Check the sampling rate of the measurement system,
   (a) the sampling rate will be equal for all input channels of a specific measurement system,
   (b) a low-pass filter is necessary if the sampling rate is too high,
   (c) remember that a filter will not remove frequency components, only reduce them!,
4. Consider a filter, either a hardware filter or a software filter,
   (a) to avoid aliasing, use a hardware filter,
   (b) if oversampling is possible, use a software filter,
5. Many sensing devices eliminate unwanted frequency at the sensing stage,
6. Try to avoid system noise at all stages in the measurement system.

32.7 Filtering

An input signal will often be complex, a sum of many different frequencies and amplitudes. To be able to remove some of these frequencies, filtering is used. Two very common situations where filtering is used are noise and aliasing. Aliasing is regarding sampling and will be discussed later in the course. Noise is unused frequency components of the signal and a filter can remove these unwanted components.

Filters can either be hardware devices or implemented in software. Hardware filters are used if filtering is necessary before the measurement (DAQ) system, software filters can only be implemented on the signal after the digital conversion. Hardware filters are made by combinations of resistors, capacitors and/or inductances using the physical principles that the impedance of inductors increases with frequency and the impedance of capacitors decreases with frequency.

There are four different type of filters:

1. low-pass filter; see Figure 32.8 (a), pass band will be low frequency, stop band in high frequency,
2. high-pass filter; see Figure 32.8 (b), pass band will be high frequency, stop band in low frequency
3. band-pass filter; see Figure 32.8 (c),
4. band-stop filter; see Figure 32.8 (d),

As shown in Figure 32.8 the filters have a corner frequency, f_c, indicating the frequency for the signal attenuation. A large number of hardware filters exist, divided into different classes of filters. Each filter class has unique characteristics that make them suitable for a particular application. The most used classes are:

1. Butterworth; maximally flat in the pass band but not a crisp cut-off in the stop band , see Figure 32.9,
2. Chebyshev; crisp cut-off in the stop band but ripples in the pass band , see Figure 32.10,
3. elliptic; a very crisp transition between pass band and stop band, but has ripples in both pass band and stop band ,
4. Bessel; a good linear variation of the phase angel with frequency in the pass band , but have a lower roll-of rate than Butterworth filters. See Figure 32.11.

A common characteristic is the filter order. The higher the order, the greater will the attenuation outside the corner frequencies be (in the stop band).

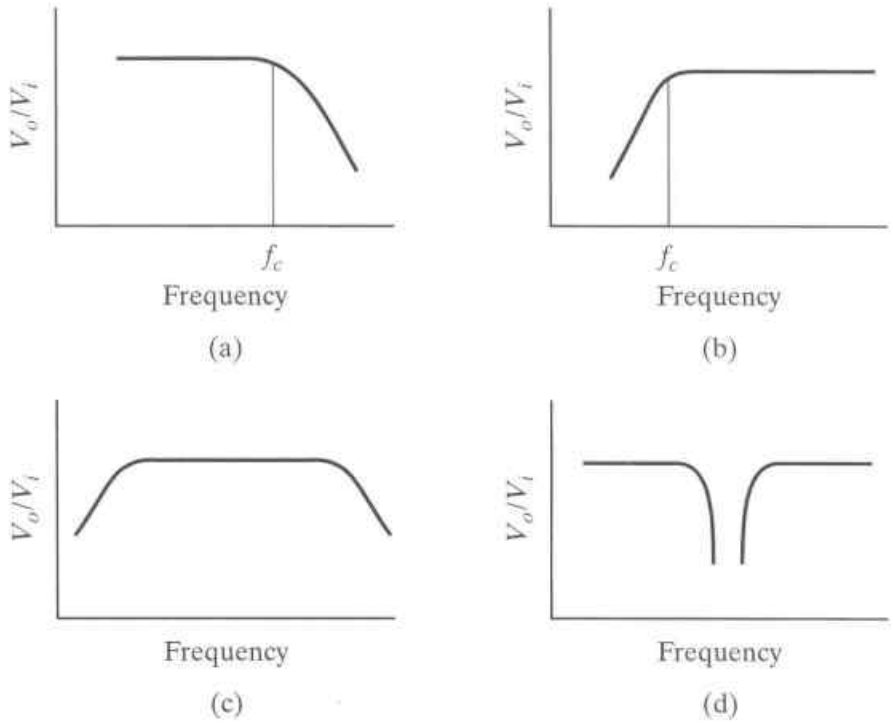


Figure 32.8: Four different type of filters: (a) lowpass filter; (b) highpass filter; (c) bandpass filter; (d) bandstop filter (Wheeler & Ganji 2004).

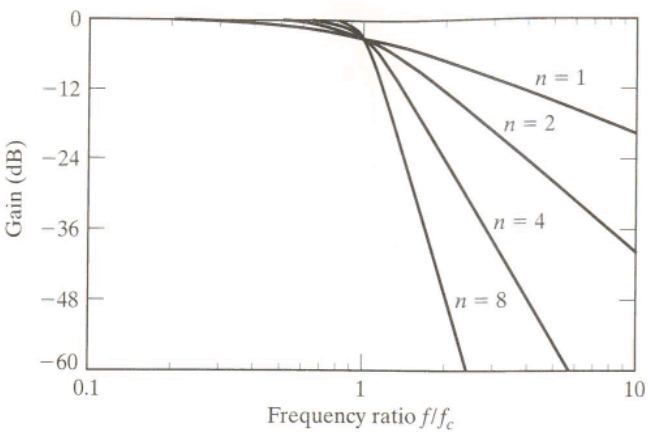


Figure 32.9: Gain of a lowpass Butterworth filter as a function of filter orders and frequency (Wheeler & Ganji 2004).

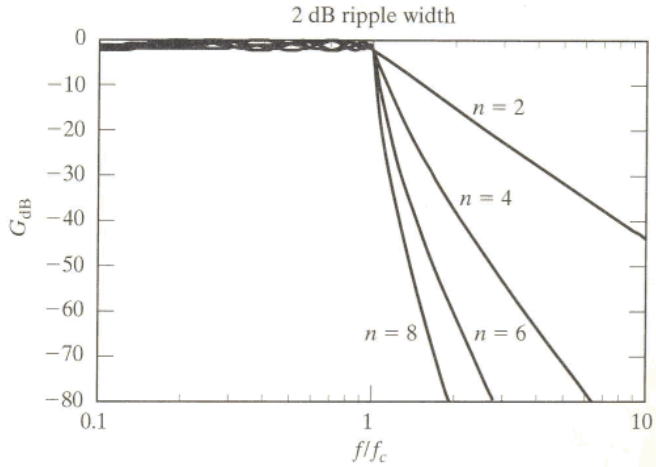


Figure 32.10: Gain of a lowpass Chebyshev filter as a function of filter orders and frequency (Wheeler & Ganji 2004).

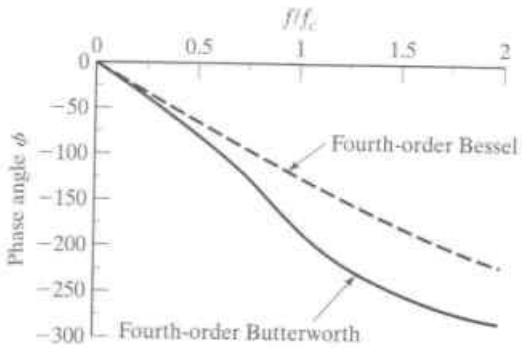


Figure 32.11: Phase angle variation with frequency for Bessel and Butterworth classes of filters (Wheeler & Ganji 2004).

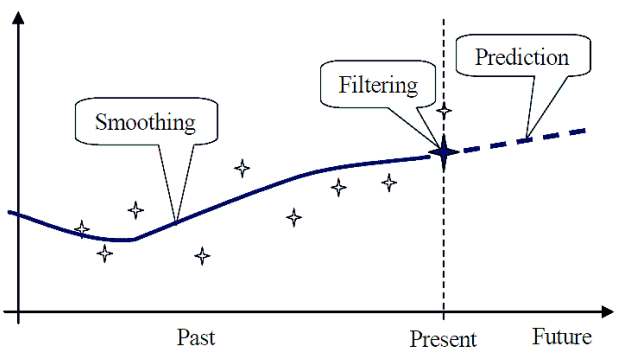


Figure 32.12: Filter values are delayed values of the present value, smooting several of the last present values. Past values are used for trending and predictiable values are for estimation in the future.

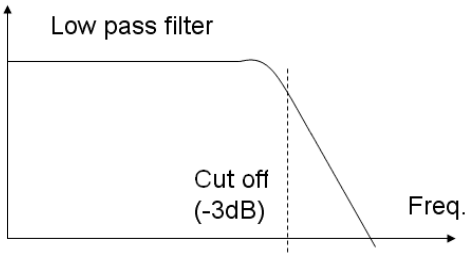


Figure 32.13: The response of a low pass filter.

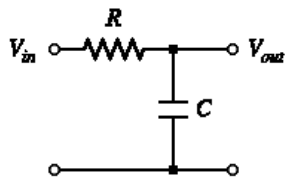


Figure 32.14: A first order LP filter using a resistor and a capacitor.

32.7.1 Low pass filter

A low pass filter is used for smoothing a set of the last input values (present values) and will give some delay of the actual input signal. The differences of the present value, a filtered value, a predicted value, and a trend value is shown in Figure 32.12.

The low pass filter is shown in Figure 32.13 where the cut off frequency, f_c, is where the amplitude is $-3dB$ or $\frac{1}{\sqrt{2}}$.

Hardware low pass filter

The simplest and cheapest filter available is a single pole RC filter, first order filter, a resistor (R) in serial with the signal and a capacitor (C) between the signal and ground. The filter rolls off at 6 dB per [3] (20 dB per decade[4]) above the corner frequency at:

$$f_c = \frac{1}{2\pi RC}$$

A simple low pass filter can be designed using only a resistor and a capacitor as shown in Figure 32.14. The time constant for the filter will be:

$$\tau = \frac{1}{\omega_{cutt-off}} = \frac{1}{2\pi f_c} = R \cdot C$$

In discrete-time will a low pass filter be:

$$y_k = (1-\alpha)\, y_{k-1} + \alpha u_k$$

where the filter constant α is a number between 0 and 1, y is the output value, and u is the input value. The filter is based combining the the new input value and the old output to get the new output value. The filter constant is:

$$\alpha = \frac{dT}{RC + dT}$$

where dT is the sampling time. Knowing the sampling time and the filter constant, the RC factor is:

$$RC = dT\left(\frac{1-\alpha}{\alpha}\right)$$

[3] Octave: Double the frequency (range from f to $2 \cdot f$).
[4] Decade: Ten times the frequency (range from f to $10 \cdot f$).

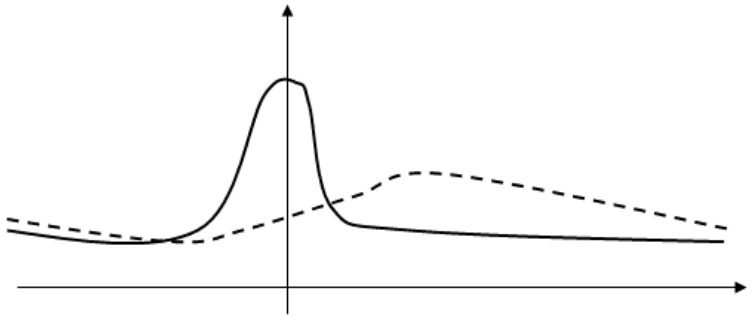


Figure 32.15: The input (solid draw) signal and the output (dotted line) signal for an IIR low pass filter.

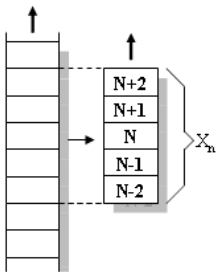


Figure 32.16: A moving average filter.

Software low pass filter

A low pass filter is often implemented as either an exponential filter or as a moving average filter. The exponential filter is using a fraction of the old filtered value and a fraction of the new unfiltered value, hence the name. The moving average filter will only use a set of the unfiltered values, not any of the filtered values.

Infinite Impulse Response (IIR) filter The exponential filter will be (Haugen 2008):

$$y_k = \theta x_k + (1 - \theta)\, y_{k-1}$$

where y_k is the new value, x_k is the new sensor value, and θ is the filter weight. The filter is very easy to implement in software as only the previous value is needed for each calculation. The transfer function for such a filter is given in Appendix **??**. Figure 32.15 shows the input signal (solid line) and the output signal (dotted line) of such a filter.

Another type of exponential filter is the error summing filter, described as:

$$y_k = y_{k-1} + \frac{(x_k - y_{k-1})}{\theta}$$

where y_k is the new value, x_k is the new sensor value, and θ is the filter weight. The filter uses the deviation between the input and output values to correct the new output value. The filter weight decide how much of the deviation should be used to adjust the new output value, the value of the weight is $[\theta \leq 1]$.

The exponential filter and the error summing filter are infinite impulse response (IIR) (Ifeachor & Jervis 2002) filters meaning that there is feedback in the filter. Also note that the exponential filter is also denoted as Exponential Weighted Moving Average (EWMA) filter .

Finite Impulse Response (FIR) filter The box-car average, moving average and weighted moving average filters are finite impulse response (FIR) (Ifeachor & Jervis 2002) filters meaning that there is no feedback in these filters. A moving average filter is shown in Figure 32.16.

The filter requires a ring buffer in software to save the last n values from the sensor device and the average calculation for each reading of a new sensor value. The output value y_k will be:

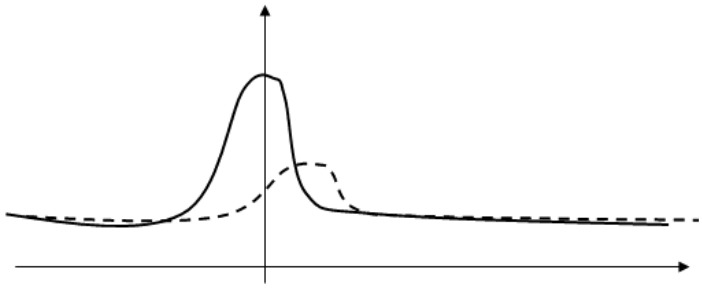


Figure 32.17: The input (solid draw) signal and the output (dotted line) signal for an FIR low pass filter.

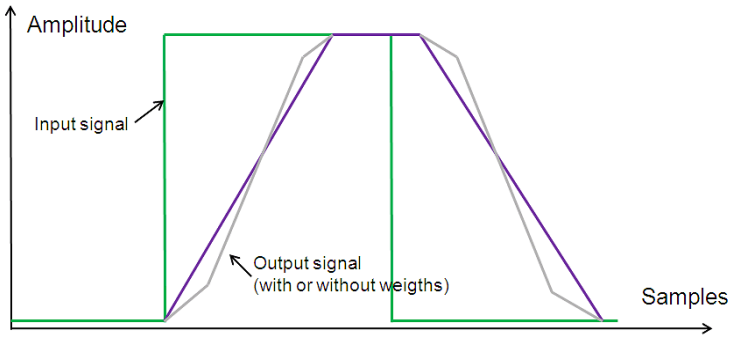


Figure 32.18: A moving average filter for a step response, with and without index weights.

$$y_k = \frac{\sum_{i=1}^{N} x_i}{N}$$

where y_k will be the filter output value at step k estimated from the N sensor device values. Figure 32.17 shows the input signal (solid line) and the output signal (dotted line) of such a filter.

Each index in the ring buffer can add a weight to change the behavior of the filter. Figure 32.18 shows such a filter on a step response signal. The step is delayed, and the highest frequencies are removed. Using weights the sharp corners can be removed as well. The sum of the weights must be equal to 1.Using a moving average filter of 6 values gives the following sums. The weights are only an example, normally all the weights are one $(C_i = 1)$.

Signal	Weight	Value
N_1	$C_1 = 1,5$	$N_1 * C_1$
N_2	$C_2 = 1,5$	$N_2 * C_2$
N_3	$C_3 = 1$	$N_3 * C_3$
N_4	$C_4 = 1$	$N_4 * C_4$
N_5	$C_5 = 0,5$	$N_5 * C_5$
N_6	$C_6 = 0,5$	$N_6 * C_6$
Sum=6	Sum=6	

32.7.2 High pass filter

The simplest and cheapest filter available is a single pole RC filter, a resistor (R) and a capacitor (C) as shown in Figure 32.19.

The filter rolls off at 6 dB per octave (20 dB per decade) below the corner frequency at:

$$f_c = \frac{1}{2\pi RC}$$

The time constant for the filter will be:

$$\tau = \frac{1}{\omega_{cutt-off}} = R \cdot C$$

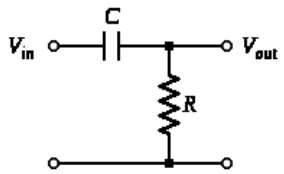


Figure 32.19: A simple high pass filter using a capacitor (C) and a resistor (R).

In discrete-time will a high pass filter be:

$$y_k = \alpha y_{k-1} + \alpha(u_k - u_{k-1})$$

where the filter constant α is a number between 0 and 1. The filter constant is:

$$\alpha = \frac{RC}{RC + dT}$$

where dT is the sampling time. Knowing the sampling time and the filter constant, the RC factor is:

$$RC = dT\left(\frac{\alpha}{1-\alpha}\right)$$

32.7.3 FIR or IIR filter

The Finite Impulse Response (FIR) filter is based on feed forward while the Infinite Impulse Response (IIR) is based on both feed forward and feedback. A suggestions for using these filters:

- IIR (exponential, feedback):
  - usage if a sharp cut-off is wanted and having a large data rate,
  - used for applications where linear characteristics are not of concern,
  - IIR is better for lower-order tapping, whereas the FIR filter is used for higher-order tapping,
  - IIR filters have much better frequency response than FIR filters,
- FIR (moving average, no feedback):
  - usage if a linear phase is wanted and having small data rate,
  - required for linear-phase characteristics,
  - FIR filters are preferred over IIR because they are more stable since feedback is not involved,
  - FIR filters are more complicated than IIR filters, but also more flexible.

The results of a low pass filters of a step response and a rectangular waveform is shown in Figure 32.20 and Figure 32.21. Figure 32.20 shows the exponential filter, the error summing filter and the moving average filter for a step response. Figure 32.21 shows the exponential filter, the error summing filter and the moving average filter for a rectangular waveform.

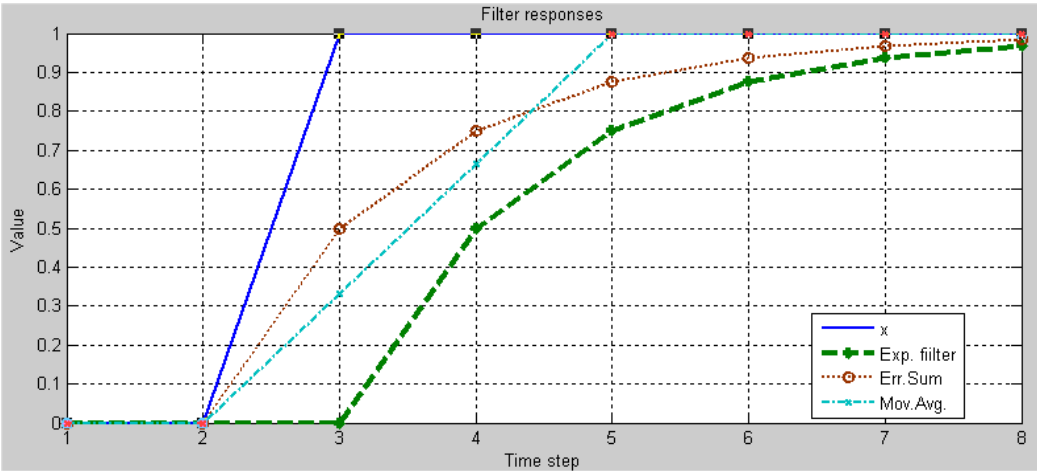


Figure 32.20: The responses of the filters from a step response.

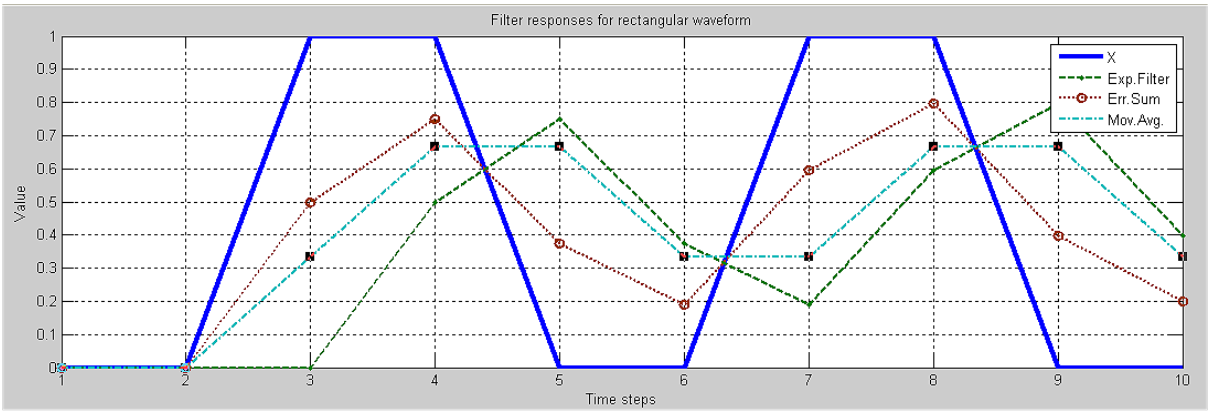


Figure 32.21: The responses of filter values from a rectuangular waveform.

Part XI

Process database

Chapter 33

Introduction

Databases are more and more used in computer systems due to a growing amount of data and a more easy exchange of data. Databases can also be used as an interface between different computer systems or applications, as an application programming interface (API). This section will focus on the usage of databases in industrial IT systems, meaning the:

1. configuration data for all items in the physical process,
2. measured values from the physical process,
3. derived values,
4. logging of historical data,
5. different type of database types and where to use these databases.

The motivation will be to understand why and where to use a database in an industrial system, know the difference of database models, know the difference of a relational database and a time series database, and know briefly about a SQL database.

The database will often be connected to the SCADA system as shown in Figure 33.1. The database will contain the information about the configuration of the system (plant), the current values for the system (plant), and optional the historical values.

The focus in this part will be the OT (operational technology) part of the system, but database modules are also important for the IT (information technology) part of the system.

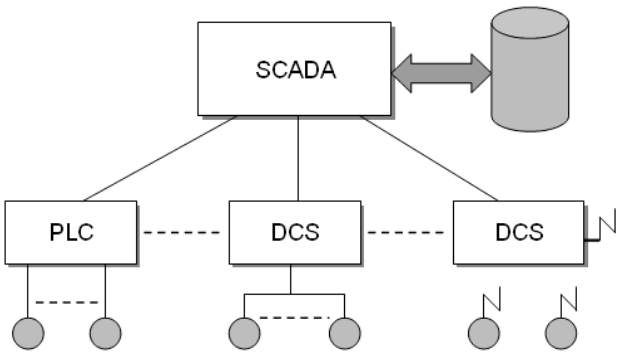


Figure 33.1: A SCADA system with a database.

Chapter 34

Process data

A control system often has hundreds or thousands of interface points to the physical process. It will be almost impossible to manage the information for each point in the software modules without some kind of database. Instead of designing the software modules for each point in each physical process, some sort of configuration should be added to the software modules to make them more flexible connecting to a point in the process. This configuration can be a set of data for each point in the physical process, describing for the software module how to deal with the information for this specific point. The software modules can be standard modules, while the specific physical process data will be the configuration data. Figure 34.1 shows two different physical processes with different type and number of sensors connected to the measurement system.

34.1 Configuration data

The software system will consist of standard software modules, configuration tools for collecting the specific information for each point in the physical process, and the configuration data. The configuration data will be (Olsson & Piani 1998):

1. parameters for all sensors and actuators (all I/O),
2. parameters for the computation of derived variables,
3. event definition and connection with control actions (if required),
4. parameters for the digital controllers.

The configuration data will be specific for each physical process and should be stored "outside" the software modules to make the software modules independent of the layout and size of the physical processes. The configuration data must be available for both the configuration tools and the software modules meaning that the configuration data should be saved in some sort of database. The database can be any type of device being able to save this kind of configuration data, either proprietor or standard database system. See Figure 34.2.

The interface between the database system and the software modules/applications using the database will include among others:

1. support for data input to the database system,

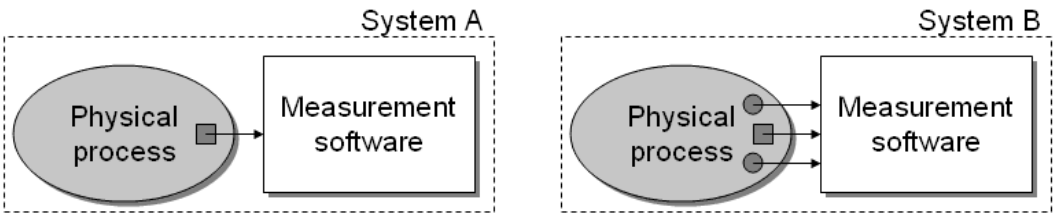


Figure 34.1: Physical processes with different type and number of sensors.

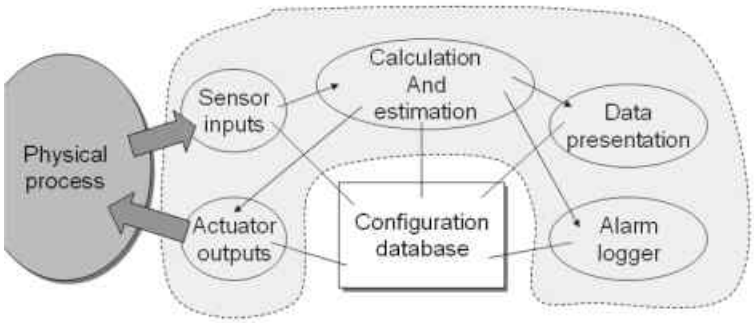


Figure 34.2: A process system using a "standard" software measurement and control module together with a configuration database.

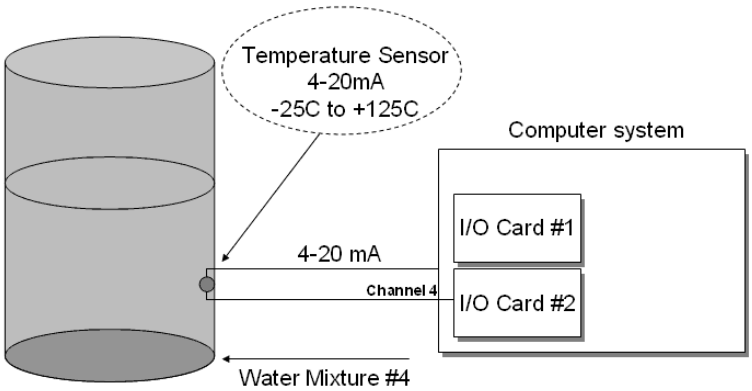


Figure 34.3: A $4-20\,\mathrm{mA}$ sensor connected to a measurement system.

2. support for data output from the database system,
3. data representation,
4. support for command input to the database system.

The information for each Input/Output (I/O) point in a process system is often referred to as a tag. The type of configuration data and/or information saved in the database for a tag will be (among others): 1) code, 2) name / description, 3) type, 4) ranges; range and unit conversions, error levels, 5) address or physical reference, 6) event class, 7) alarm class, 8) sampling time and 9) filter parameter(s).

Additional information for analog tags saved in the database will be (among others): 1) scaling or range coefficients, 2) measuring units, 3) minimum and maximum measuring unit limits and 4) limits for generation of alarm(s).

Additional information for digital tags saved in the database will be (among others): 1) description of the states in clear text and 2) state for generation of an alarm.

Tag information: A temperature sensor device with a $4-20\,\mathrm{mA}$ interface is used for measuring the temperature at the bottom of a vessel called WaterMixture #4 and the temperature range for the sensor is $-25\,^{\circ}\mathrm{C}$ to $+125\,^{\circ}\mathrm{C}$. The sensor is connected to channel 4 on I/O card address 2. See Figure 34.3 for sensor connection in this example.

The configuration of this I/O point will be (minimum):

Code	WM4	
Description	WaterMixture #4	
Type	Sensor	
Range minimum	4 mA	$Range_{\min}$
Range maximum	20 mA	$Range_{\max}$
Error low	3.5 mA	
Error high	22.5 mA	
Address	2	
Channel	4	
Unit	Temperature	
Minimum unit	−25 °C	$Unit_{\min}$
Maximum unit	+125 °C	$Unit_{\max}$
Sampling time	30 s	
Accuracy (±)	0.5 °C	
Filter	Auto	

In many process systems tag is used as the item identifier instead of the code identifier. Tag will be defined according to company specific codes and is often based on some sort of standard. The International Society of Automation (ISA) has the standard "Instrumentation Symbols and Identification" (ISA-S5.1) (ISA 1992) defining standard tag number convention for valves, transmitters, other field devices, and calculation and control functions.

The converted value from a sensor device can be calculated using the configuration data as:

$$Val\,(2:4)_{converted} = Unit_{\min} + (\frac{Val_{sensor} - Range_{\min}}{Range_{\max} - Range_{\min}} \cdot (Unit_{\max} - Unit_{\min}))$$

where Val_{sensor} is the raw value from the sensor, $Range_{\min}$ and $Range_{\max}$ are the range minimum and maximum values, $Unit_{\min}$ and $Unit_{\max}$ are the unit minimum and maximum values for the sensor, and $Val_{converted}$ is the converted value for the sensor. The value for this sensor device is received from channel #4 of I/O device #2, so the correct unit must be part of the configuration data.

Reading a value of 13.5 mA as the raw value from the sensor gives the temperature (with an accuracy of ±0.5 °C), as the tag and configuration states that this channel is a temperature sensor:

$$Temp = -25 + (\frac{13.5 - 4}{20 - 4} \cdot (125 - (-25))) = 64.1\,^{\circ}\mathrm{C} \pm 0.5\,^{\circ}\mathrm{C}$$

34.2 Runtime and alarms

The configuration database is normally combined with the current data (values) of the process, called the process database. The process database contains the configuration data for the system and the current state of the system. The additional data for the current state of the system will be the:

1. raw value,
2. converted value,
3. displayed value,
4. sampling time (log time),
5. alarm states,
6. any derived values of interest.

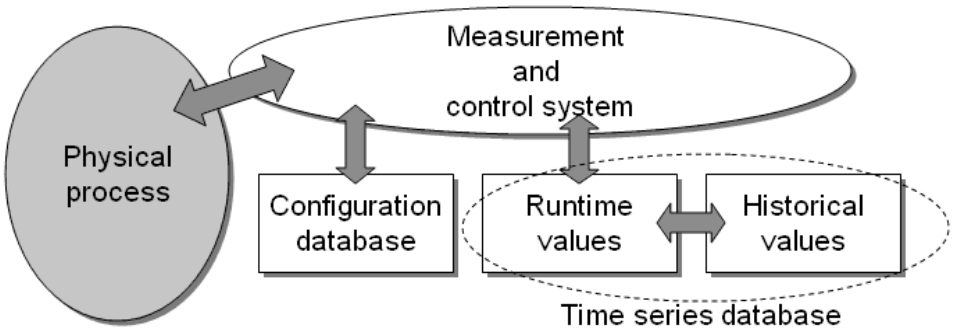


Figure 34.4: The usage of a configuration database, a runtime database, and a historical database.

34.3 Historical data

An important task in the process industry is to keep records of production rates and produce statistics. The data contained in the database will be used as input to the statistical functions, but the basic statistical operation is integration of data over time (Olsson & Piani 1998). The time can be the last hours, days, weeks, or even years and the historical values should then be saved in a database as well. The values should be the same as listed in the runtime section.

A measurement and control system with a combination of databases for configuration, runtime, and historical data is shown in Figure 34.4.

One main reason for storing data is so that it is available to retrieve later. This makes the retrieval performance important, this performance will depend on the type of database and the structure of the database.

34.4 Data compressions

Time history information of plant can be large and compression of data is one way of reducing the amount of data. The reasons for compressions is to reduce storing space and the amount of information sent between the modules in a distributed system. Each tag in a process control system is a time and value pair, every value should be stored together with a time stamp. Several compression methods exist as compression will always loose some information, different methods deals with this loss in different ways. Some of the compression methods are Boxcar, Backslope, SLIM (Straight Line Interpolative Methods), and Swinging Door to mention some. The methods and algorithms will not be presented, but the reader should be aware of some of the possibilities.

Chapter 35

Database structure

A database (DB) is a collection of data stored in a structured collection of records (or data) in such a way that a database management system can use it to answer queries. The records retrieved in answer to queries become information that can be used to make decisions. The records are a collection of several data types with the same type of information. A database should normally have an unlimited number of records. The application used to manage and query a database is known as a database management system (DBMS).

The central concept of a database is a collection of records containing some specific pieces of knowledge. Typically, for a given database, there is a structural description of the type of facts held in that database. This description is known as a schema. The schema describes the objects that are represented in the database, and the relationships among them. This schema will be a structural description of type of data held in the database.

The records are collection of data, a group of data, and used for saving a group of data, not a single byte. Figure 35.1 shows the structure of a database.

The most used database models are a) relational DB models, b) hierarchical DB models, and c) network DB models. The relationships are a) one to many, b) many to many, and c) one to one.

35.1 Database models

The database contains normally a lot of data organized as a set of records. The structure of these records is defined in a schema and is the modelling of the database structure (www.wikipedia.org 2006). This database structure is known as database model or the data model. The model most used today is the relational model, where all the information is in the form of multiple related tables each consisting of rows and columns. This model represents relationships by the use of values common to more than one table.

Various techniques are used to model data structure. Most database systems are built around one particular data model, although it is increasingly common for products to offer support for more than one

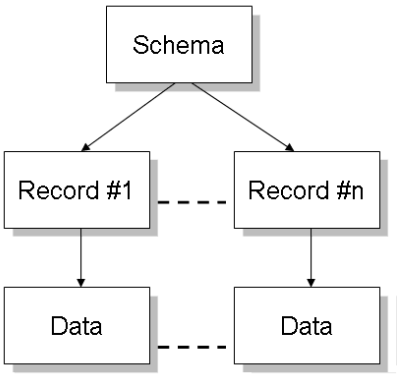


Figure 35.1: The schema, records and data in a database.

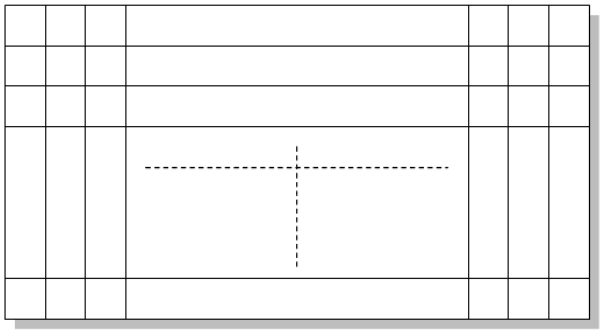


Figure 35.2: A flat database model as row and columns where a member of a given column is given the values in the rows.

model. For any one logical model various physical implementations may be possible, and most products will offer the user some level of control in tuning the physical implementation, since the choices that are made have a significant effect on performance.

$$\left\{ \begin{array}{l} \text{Database model} \\ \text{Data model} \end{array} \right\} \quad \left\{ \begin{array}{l} \text{Database structure} \\ \text{Type of operations} \end{array} \right\} \quad \{ \text{ Schema } \}$$

A data model is not just a way of structuring data: it also defines a set of operations that can be performed on the data. The database structure is defined in the schema, the schema is defining the records, and the records is defining the data and/or information in the database.

$$\text{Schema} \iff \text{Records} \iff \text{Data}$$

Different database models (www.wikipedia.org 2006) are flat model, hierarchical model, network model, relational model, dimensional model, and object database models among others.

35.1.1 Flat model

This may not strictly qualify as a data model, as defined above. The flat (or table) model consists of a single, two-dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another. Columns of the table often have a type associated with them, defining them as character data, date or time information, integers, or floating point numbers (www.wikipedia.org 2006). Figure 35.2 shows a flat database model.

Example:

Student	**Address**	**Course**	**Grade**	**Teacher**	**Department**
Student #1	Address #1	Course #1	Grade	Teacher #1	Department #1
Student #1	Address #1	Course #2	Grade	Teacher #1	Department #1
..	..	..	..	..	..
Student #1	Address #1	Course #M	Grade	Teacher #R	Department #1
Student #2	Address #2	Course #1	Grade	Teacher #2	Department #1
..	..	..	..	..	..
Student #N	Address #N	Course #1	Grade	Teacher #3	Department #1

Redundancy of data would occur because flat databases do not handle one-to-many or many-to-many relationships well. The example has multiple instances of student names, student addresses, course names, teacher names and department names. If changing any of these instances, the same instance must be updated in all locations.

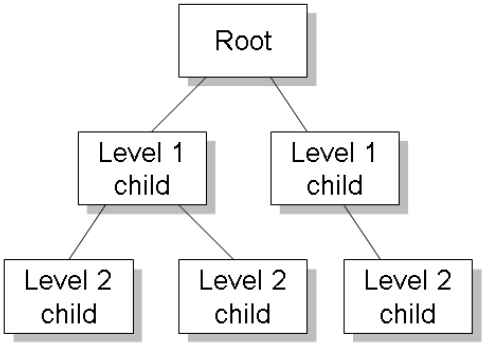


Figure 35.3: A hierarchical model with a tree of data starting from the root.

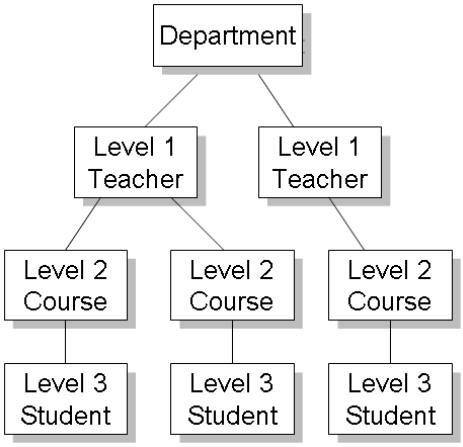


Figure 35.4: An example of the hierachical model.

35.1.2 Hierarchical model

In a hierarchical model, data is organized into a tree-like structure, implying a single upward link in each record to describe the nesting, and a sort field to keep the records in a particular order in each same-level list. Hierarchical structures were widely used in the early mainframe database management systems, such as the Information Management System (IMS) by IBM, and now describe the structure of XML documents. This structure allows one 1:N relationship between two types of data. This structure is very efficient to describe many relationships in the real world; recipes, table of contents, ordering of paragraphs/verses, any nested and sorted information. However, the hierarchical structure is inefficient for certain database operations when a full path (as opposed to upward link and sort field) is not also included for each record (www.wikipedia.org 2006). Figure 35.3 shows a hierarchical model.

Redundancy of data would occur because hierarchical databases handle one-to-many relationships well but do not handle many-to-many relationships well. This is because a child may only have one parent. However, in many cases you will want to have the child be related to more than one parent.

Example shown in Figure 35.4, but the database still has redundant data since a child can not have several parents.

35.1.3 Network model

The network model (defined by the CODASYL[1] specification) organizes data using two fundamental constructs, called records and sets. Records contain fields (which may be organized hierarchically). Sets (not to be confused with mathematical sets) define one-to-many relationships between records: one owner, many members. A record may be an owner in any number of sets, and a member in any number of sets. The operations of the network model are navigational in style: a program maintains a current position, and navigates from one record to another by following the relationships in which the record

[1] "Conference on Data Systems Languages" created by the Data Base Task group.

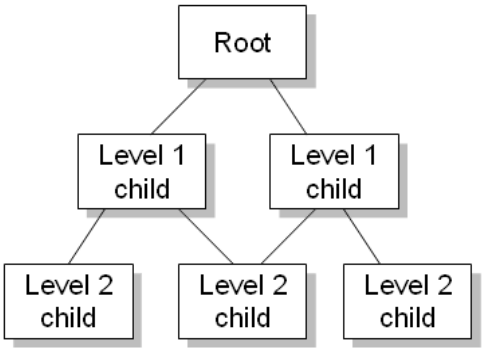


Figure 35.5: A network data model with records and sets.

participates. Records can also be located by supplying key values. A record can be linked to several owners, or multiple records can be linked to the same owner. In Figure 35.5 is the *Level 2 child* record linked to two different owners.

35.1.4 Relational model

The relational model was introduced in an academic paper by E. F. Codd in 1970 as a way to make database management systems (DBMS) more independent of any particular application. It is a mathematical model defined in terms of predicate logic and set theory. The products that are generally referred to as relational databases in fact implement a model that is only an approximation to the mathematical model defined by Codd. Three key terms are used extensively in relational database models:

1. relations,
2. attributes,
3. domains.

A relation is a table with columns and rows. The named columns of the relation are called attributes, and the domain is the set of values the attributes are allowed to take. The basic data structure of the relational model is the table, where information about a particular entity (say, an employee) is represented in columns and rows (also called tuples). Thus, the “relation” in “relational database” refers to the various tables in the database; a relation is a set of tuples. The columns enumerate the various attributes of the entity (the employee’s name, address or phone number, for example), and a row is an actual instance of the entity (a specific employee) that is represented by the relation. As a result, each tuple of the employee table represents various attributes of a single employee. All relations (and, thus, tables) in a relational database have to adhere to some basic rules to qualify as relations.

A relational database contains multiple tables, each similar to the one in the "flat" database model. One of the strengths of the relational model is that, in principle, any value occurring in two different records (belonging to the same table or to different tables), implies a relationship among those two records. A key can be defined for describing the relation between data in different tables and both primary and secondary keys can be used for describing these relations.

A database with two tables is shown in Figure 35.6, with different numbers of rows and columns. The columns (or fields) represent the specific data while the rows (or records) represent the data sets. A column can be defined as a primary key and can be defined like:

1. AUTO_INCREMENT: A primary key that will be incremented every time a new row is inserted,
2. UNIQUE: This field must be unique for every row,
3. NOT NULL: This field can not be empty.Keys are used for making the combinations of tables and is describing the relationship between data in different tables.

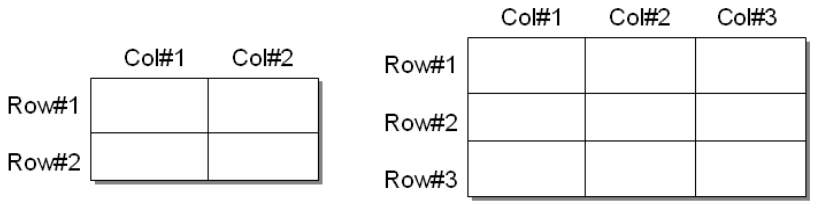


Figure 35.6: A relation database with two tables.

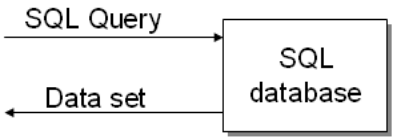


Figure 35.7: SQL query and return of information for a SQL database.

Tables

Tables can also have a designated single attribute or a set of attributes that can act as a "key", which can be used to uniquely identify each tuple in the table. A key that can be used to uniquely identify a row in a table is called a primary key. Keys are commonly used to join or combine data from two or more tables. For example, an Employee table may contain a column named Location which contains a value that matches the key of a Location table. Keys are also critical in the creation of indexes, which facilitate fast retrieval of data from large tables. Any column can be a key, or multiple columns can be grouped together into a compound key. It is not necessary to define all the keys in advance; a column can be used as a key even if it was not originally intended to be one. A key that has an external, real-world meaning (such as a person's name, a book's ISBN, or a car's serial number) is sometimes called a "natural" key. If no natural key is suitable (think of the many people named Brown), an arbitrary or surrogate key can be assigned (such as by giving employees ID numbers). In practice, most databases have both generated and natural keys, because generated keys can be used internally to create links between rows that cannot break, while natural keys can be used, less reliably, for searches and for integration with other databases (www.wikipedia.org 2006).

Operations

Users (or programs) request data from a relational database by sending it a query that is written in a special language, usually a dialect of Structured Query Language (SQL). Although SQL was originally intended for end-users, it is much more common for SQL queries to be embedded into software that provides an easier user interface. Many web sites, such as Wikipedia, perform SQL queries when generating pages. In response to a query, the database returns a result set, which is just a list of rows containing the answers. The simplest query is just to return all the rows from a table, but more often, the rows are filtered in some way to return just the answer wanted. Often, data from multiple tables are combined into one, by doing a join. Conceptually, this is done by taking all possible combinations of rows (the Cartesian product), and then filtering out everything except the answer. In practice, relational database management systems rewrite ("optimize") queries to perform faster, using a variety of techniques. There are a number of relational operations in addition to join. (www.wikipedia.org 2006).

See Figure 35.7 for the functions of a SQL database.

Background

Edgar Codd worked at IBM in San Jose, California, in one of their offshoot offices that was primarily involved in the development of hard disk systems. He was unhappy with the navigational model of the Codasyl approach, notably the lack of a "search" facility which was becoming increasingly useful when the database was stored on disk instead of tape. In 1970, he wrote a number of papers that outlined a new approach to database construction that eventually culminated in the ground breaking "*A Relational*

Model of Data for Large Shared Data Banks"[2].

In this paper, he described a new system for storing and working with large databases. Instead of records being stored in some sort of linked list of free-form records as in Codasyl, Codd's idea was to use a "table" of fixed-length records. A linked-list system would be very inefficient when storing "sparse" databases where some of the data for any one record could be left empty. The relational model solved this by splitting the data into a series of normalized tables, with optional elements being moved out of the main table to where they would take up room only if needed.

Just as the navigational approach would require programs to loop in order to collect records, the relational approach would require loops to collect information about any one record. Codd's solution to the necessary looping was a set-oriented language, a suggestion that would later spawn the ubiquitous SQL. Using a branch of mathematics known as tuple calculus, he demonstrated that such a system could support all the operations of normal databases (inserting, updating etc.) as well as providing a simple system for finding and returning sets of data in a single operation (www.wikipedia.org 2006).

35.1.5 Dimensional model

The dimensional model is a specialized adaptation of the relational model used to represent data in data warehouses in a way that data can be easily summarized using OLAP[3] queries. In the dimensional model, a database consists of a single large table of facts that are described using dimensions and measures. A dimension provides the context of a fact (such as who participated, when and where it happened, and its type) and is used in queries to group related facts together. Dimensions tend to be discrete and are often hierarchical; for example, the location might include the building, state, and country. A measure is a quantity describing the fact, such as revenue. It's important that measures can be meaningfully aggregated - for example, the revenue from different locations can be added together. In an OLAP query, dimensions are chosen and the facts are grouped and added together to create a summary. The dimensional model is often implemented on top of the relational model using a star schema, consisting of one table containing the facts and surrounding tables containing the dimensions (www.wikipedia.org 2006).

35.1.6 Object database model

In recent years, the object-oriented paradigm has been applied to database technology, creating a new programming model known as object databases. These databases attempt to bring the database world and the application programming world closer together, in particular by ensuring that the database uses the same type system as the application program. This aims to avoid the overhead (sometimes referred to as the impedance mismatch) of converting information between its representation in the database (for example as rows in tables) and its representation in the application program (typically as objects). At the same time object databases attempt to introduce the key ideas of object programming, such as encapsulation and polymorphism, into the world of databases. A variety of these ways have been tried for storing objects in a database. Some products have approached the problem from the application programming end, by making the objects manipulated by the program persistent. This also typically requires the addition of some kind of query language, since conventional programming languages do not have the ability to find objects based on their information content. Others have attacked the problem from the database end, by defining an object-oriented data model for the database, and defining a database programming language that allows full programming capabilities as well as traditional query facilities. Object databases suffered because of a lack of standardization: although standards were defined by ODMG[4], they were never implemented well enough to ensure interoperability between products. Nevertheless, object databases have been used successfully in many applications: usually specialized applications such as engineering databases or molecular biology databases rather than mainstream commercial data processing. However, object database ideas were picked up by the relational vendors and influenced extensions made to these products and indeed to the SQL language (www.wikipedia.org 2006).

[2] Codd, E.F. (1970). "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387.

[3] On Line Analytical Processing

[4] Object Data Management Group, www.odmg.org (feb-07).

35.2 Time series database

A time series database server (TSDS) is a software system that is optimized for handling a time series. In this context, a time series is an associative array of numbers indexed by a datetime or a datetime range. These time series are often called profiles or curves, depending upon the market.

TSDSs simplify the development of software with complex business rules in a wide variety of sectors. Queries for historical data, replete with time ranges and roll ups and arbitrary time zone conversions are difficult in a relational database. Compositions of those rules are even more difficult. This is a problem compounded by the free nature of relational systems themselves. Many relational systems are often not modelled correctly with respect to time series data. TSDS on the other hand impose a model and this allows them to provide more features for doing so.

The TSDS allows users to create, enumerate, update and destroy various time series and organize them in some fashion. These series may be organized hierarchically and optionally have companion meta data available with them. The server often supports a number of basic calculations that work on a series as a whole, such as multiplying, adding, or otherwise combining various time series into a new time series. They can also filter on arbitrary patterns defined by the day of the week, low value filters, high value filters, or even have the values of one series filter another. Some TSDSs also build in a wealth of statistical functions.

The drawback of a time series database is that there is no standard for a server language, how to get the data out of the database.

Some of the TSDS are (Fuglset 2003):

Name	Firm	Note
PI System	OSI Software	Plant Information System
InfoPlus	AspenTech	
iHistorian	Intellution	
Hostorian	Wonderware	Extension of Microsoft SQL Server

35.3 Database interface (API)

The interface between the process control system and the database need to be according to a protocol. The protocol is often part of an Application Programming Interface (API) for access of database data from a software application.

35.3.1 ODBC

Open DataBase Connectivity, a standard database access method developed by the SQL Access group in 1992. The goal of ODBC is to make it possible to access any data from any application, regardless of which database management system (DBMS) is handling the data. ODBC manages this by inserting a middle layer, called a database driver , between an application and the DBMS. The purpose of this layer is to translate the application's data queries into commands that the DBMS understands. For this to work, both the application and the DBMS must be ODBC-compliant – that is, the application must be capable of issuing ODBC commands and the DBMS must be capable of responding to them.

35.3.2 SQL

Structured Query Language, is a database computer language designed for the retrieval and management of data in relational database management systems (RDBMS), database schema creation and modification, and database object access control management. SQL is an extension to the ODBC API.

35.3.3 WDB

WDB is a software toolset that simplifies the integration of SQL databases into the World Wide Web. WDB lets you provide WWW access to the contents of SQL databases without writing a single line of code, yet provides the flexibility to easily add new capabilities as needed.

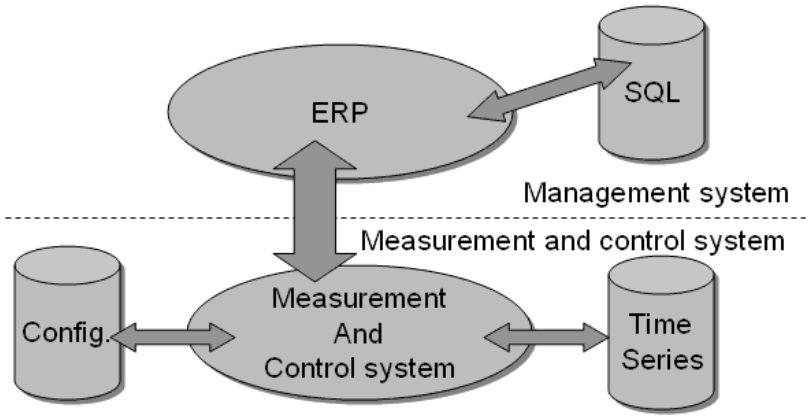


Figure 35.8: A process system with an ERP and a measurement/control modules with a set of databases.

35.3.4 DBI

The DBI is the standard database interface module for Perl. It defines a set of methods, variables and conventions that provide a consistent database interface independent of the actual database being used.

35.4 Process database system

A process system with communication to a management system, an Enterprise Resource Planning (ERP) is shown in Figure 35.8.

The figure shows a system with 3 different types of databases;

1. a configuration database that can be any type of database model,
2. a time series database for runtime data,
3. a SQL database for the management system.

Chapter 36

SQL Database

Configuration data, runtime data and historical data should be saved in the process database. Normally a relation database will be used, but historical data is not that suited for a relation database. A time series database is often used for historical data but depending on the amount of data like APIS[1] and IMATIS[2] among others. Configuration data and runtime data can be saved in a relation database using SQL like MySQL[3], PostgreSQL[4], Oracle[5], Sybase[6], and Microsoft SQL[7] among others. Structured Query Language (SQL) is the most popular computer language used to create, retrieve, update and delete data from relational database management systems. The language has evolved beyond its original purpose, and now supports object-relational database management systems. SQL has been standardized by both ANSI and ISO.

36.1 Standardization

SQL was adopted as a standard by ANSI (American National Standards Institute) in 1992 and ISO (International Organization for Standardization) in 1987.

The SQL standard has gone through a number of revisions:

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	
1989	SQL-89		Minor revision.
1992	SQL-92	SQL2	Major revision (ISO 9075).
1999	SQL:1999	SQL3	
2003	SQL:2003		Introduced XML-related features,++
2006	SQL:2006		

SQL is designed for a specific purpose: to query data contained in a relational database. SQL is a set-based, declarative computer language, not an imperative language such as C or BASIC.

The complexity and size of the SQL standard means that most databases do not implement the entire standard giving some differences between the different SQL database systems.

36.2 SQL data types

Some of the data types in a SQL database are:

1. TEXT; a string of maximum 65535 characters,
2. CHAR(N); where N is the number of characters, from 1 to 255,

[1] APIS using OPC: www.prediktor.no (feb-07).
[2] IMATIS: www.cardiac.no (feb-07).
[3] MySQL: www.mysql.com (feb-07).
[4] PostgreSQL: linuxfinances.info/info/postgresql.html (feb-07).
[5] Oracle: ww.oracle.com (feb-07).
[6] Sybase: www.sybase.com (feb-07).
[7] Microsoft SQL: www.micrososft.com (feb-07).

3. VARCHAR(N); where N is the maximum number of characters, from 1 to 255,
4. INT; a positive or negative integer,
5. FLOAT(N,D); where N is the number of characters[8] and D is the number of decimals,
6. DECIMAL(N,D); where is the number of characters and D is the number of decimals,
7. DATE; the date on the format YYYY-MM-DD,
8. TIME; the time on the format HH:MM:SS,
9. BLOB[9]; binary data like picture and sound, maximum 65535 bytes.

36.3 SQL keywords

SQL keywords fall into several groups.

36.3.1 Data retrieval

The most frequently used operation in transactional databases is the data retrieval operation. When restricted to data retrieval commands, SQL acts as a declarative language.

SELECT is used to retrieve zero or more rows from one or more tables in a database. In most applications, SELECT is the most commonly used Data Manipulation Language command. In specifying a SELECT query, the user specifies a description of the desired result set, but they do not specify what physical operations must be executed to produce that result set. Translating the query into an efficient query plan is left to the database system, more specifically to the query optimizer.

Commonly available keywords related to SELECT include:

FROM is used to indicate from which tables the data is to be taken, as well as how the tables JOIN to each other.

WHERE is used to identify which rows to be retrieved, or applied to GROUP BY. WHERE is evaluated before the GROUP BY.

GROUP BY is used to combine rows with related values into elements of a smaller set of rows.

HAVING is used to identify which of the "combined rows" (combined rows are produced when the query has a GROUP BY keyword or when the SELECT part contains aggregates), are to be retrieved. HAVING acts much like a WHERE, but it operates on the results of the GROUP BY and hence can use aggregate functions.

ORDER BY is used to identify which columns are used to sort the resulting data.

Data retrieval is very often combined with data projection; usually it isn't the verbatim data stored in primitive data types that a user is looking for or a query is written to serve. Often the data needs to be expressed differently from how it's stored. SQL allows a wide variety of formulas included in the select list to project data.

Example 7 *#1*

```
SELECT * FROM books
WHERE price > 100.00
ORDER BY title
```

This is an example that could be used to get a list of expensive books. It retrieves the records from the books table that have a price field which is greater than 100.00. The result is sorted alphabetically by book title. The asterisk (*) means to show all columns of the books table. Alternatively, specific columns could be named.

Example 8 *#2*

[8] decimalpoint and the minus sign is not included.
[9] BLOB: Binary Large Object.

```
SELECT books.title, count(*) AS Authors
FROM books
JOIN book_authors
ON books.book_number = book_authors.book_number
GROUP BY books.title
```

Example 2 shows both the use of multiple tables in a join, and aggregation (grouping). This example shows how many authors there are per book.

36.3.2 Data manipulation

First, there are the standard Data Manipulation Language (DML) elements. DML is the subset of the language used to add, update and delete data.

INSERT is used to add zero or more rows (formally tuples) to an existing table.

UPDATE is used to modify the values of a set of existing table rows.

MERGE is used to combine the data of multiple tables. It is something of a combination of the INSERT and UPDATE elements. It is defined in the SQL:2003 standard; prior to that, some databases provided similar functionality via different syntax, sometimes called an "upsert".

DELETE removes zero or more existing rows from a table.

Example 9 ***INSERT***

INSERT INTO my_table (field1, field2, field3) VALUES ('test', 'N', NULL);

Example 10 ***UPDATE***

UPDATE my_table SET field1 = 'updated value' WHERE field2 = 'N';

Example 11 ***DELETE***

DELETE FROM my_table WHERE field2 = 'N';

36.3.3 Transaction Controls

Transaction, if available, can be used to wrap around the DML operations.

BEGIN WORK (or START TRANSACTION, depending on SQL dialect) can be used to mark the start of a database transaction, which either completes completely or not at all.

COMMIT causes all data changes in a transaction to be made permanent.

ROLLBACK causes all data changes since the last COMMIT or ROLLBACK to be discarded, so that the state of the data is "rolled back" to the way it was prior to those changes being requested.

COMMIT and ROLLBACK interact with areas such as transaction control and locking. Strictly, both terminate any open transaction and release any locks held on data. In the absence of a BEGIN WORK or similar statement, the semantics of SQL are implementation-dependent.

```
BEGIN WORK;
UPDATE inventory SET quantity = quantity - 3 WHERE item = 'pants';
COMMIT;
```

36.3.4 Data definition

The second group of keywords is the Data Definition Language (DDL). DDL allows the user to define new tables and associated elements. Most commercial SQL databases have proprietary extensions in their DDL, which allow control over nonstandard features of the database system. The most basic items of DDL are the CREATE, ALTER, RENAME, TRUNCATE, and DROP commands.

CREATE causes an object (a table, for example) to be created within the database.

DROP causes an existing object within the database to be deleted, usually irretrievably.

TRUNCATE deletes all data from a table (non-standard, but common SQL command).

ALTER command permits the user to modify an existing object in various ways – for example, adding a column to an existing table.

```
CREATE TABLE my_table
(
  my_field1 INT,
  my_field2 VARCHAR (50),
  my_field3 DATE NOT NULL,
  PRIMARY KEY (my_field1, my_field2)
);
```

36.3.5 Data control

The third group of SQL keywords is the Data Control Language (DCL). DCL handles the authorization aspects of data and permits the user to control who has access to see or manipulate data within the database. Its two main keywords are:

GRANT, authorizes one or more users to perform an operation or a set of operations on an object.

REVOKE, removes or restricts the capability of a user to perform an operation or a set of operations.

GRANT SELECT, UPDATE ON my_table TO some_user, another_user.

36.4 SQL commands

Some of the SQL commands are:

1. SHOW DATABASES; List the databases in the system,
2. USE db_name; Use the database with the name db_name (open the database),
3. SHOW TABLES; Shows the tables in the database (must be opened),
4. DESCRIBE db_table; Shows the table definition,
5. CREATE DATABASE db_name; Create a new database with the name db_name,
6.
```
1 CREATE TABLE db_table
2 (
3   Col1_name datatype,
4   Col2_name datatype,
5 );
```
7.
```
1 INSERT INTO db_table
2 (col1_name, col2_name)
3 VALUES
4 (col1_val1, col2_val1),
5 (col1_val2, col2_val2);
```

An example using MySQL as a SQL server is shown in the appendix. The example shows how to install, create, and use the database.

Chapter 37

Database design

The structure of a database system is defined when designing the database. The design means to decide:

1. the type of database,
2. the type and number of tables,
3. the relationship between the tables,
4. the structure of each table.

The main purpose of the design is to decide the type of data to be stored in the database, described by a model. The model will describe the structure of the data in the database. The development can be done in several phases, from a conceptual stage to a physical stage as shown in Figure 37.1.

The model can be described in different ways, one way is the use the Entity / Relationship (E/R) diagram, a graphical language where entities (objects), the attributes (properties) of these objects, and relationship is shown (Kristoffersen 2009). An E/R diagram will not be dependent on the type of database, and can be used different types of databases.

The logical design means to convert the E/R diagram into a logical form describing the logical tables with the keys (Kristoffersen 2009). In this phase the check of redundancy (of data) should be done. In larger systems the user access should be included by defining the user access to each table. Views are used to define different access of tables and external forms are used to define the different users (or groups of users) and the access.

The physical design means to convert the logical design into a specific database system with access and data structures. This means the physical tables that should be created in the database, the primary key columns, the foreign key columns, non-key columns, relationships, and the data types.

37.1 Data modeling

Data modeling is a method used to define and analyze data requirements needed to support the business processes of an organization. The process of data modeling is to make the schemes as shown in Figure

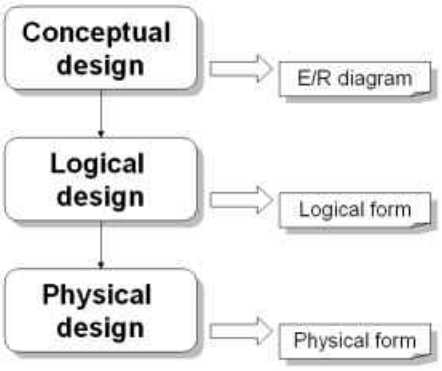


Figure 37.1: The phases of a database design (Kristoffersen 2009).

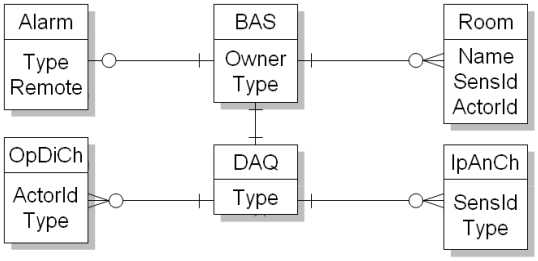


Figure 37.2: The ERM for a BAS.

37.1. The data modeling starts with the conceptual design making a conceptual schema. The conceptual schema describes the semantics of a domain, the entity classes and relationships between entity classes. The entity classes representing kinds of things of significance in the domain. One notations for data modeling is the entity relationship model (ERM) giving the E/R diagram as indicated in Figure 37.1. The EMR is a graphical diagram of the database structure. The next step is to make a logical schema describing the structure of the information in the domain. This schema consists of descriptions of tables and columns, object oriented classes, and XML tags, among other things. The last step is to make the physical schema describing the storage of the data.

37.1.1 Entity-relationship model

The building blocks of a entity relationship model (ERM) are entities, relationships, and attributes.

- An entity is a "thing" having an independent existence and can be uniquely identified, can be thought of as nouns, drawn as rectangles,
- A relationship indicates how two or more entities are related to one another, can be thought of as verbs, drawn as diamonds, connected to entities with lines,
- Every entity must have a minimal set of uniquely identifying attributes, which is called the entity's primary key, are drawn as ovals.

One way of making the ERM is using Crow's Foot diagrams where entities are represented as boxes, and relationships as lines between the boxes. The ends of these lines are shaped to represent the cardinality of the relationship (www.wikipedia.org 2010).

One goal of the ERM is to minimize the number of columns and maximize the number of the rows in the tables. Focus on tables having one to one or one to many relationship, avoid tables with many to many relationships. Use a new table in a many to many relationship to open into several one to many relationships.

Figure 37.2 shows the ERM of a Build Automation System (BAS) consisting of a set of rooms with sensors and actuators. This system has only one sensor and one actuator in each room. The sensors are connected to the analog input channels of the DAQ and the actuators are connected to the digital output channels of the DAQ.

The primary key is normally the first attribute and each table can have only one primary key. A table can several foreign keys, describing some relationship between tables. The Room table in Figure 37.2 contains the primary key *Name* and the foreign keys *SensId* and *ActorId*. The foreign keys can be used to find the DAQ channel and type in the *IpAnCh* and *OpDiCh* tables.

The Unified Modeling Language (UML) can also be used for designing and modeling relational databases (Gornik 2003).

Chapter 38

Data Privacy

38.1 GDPR

The General Data Protection Regulation (GDPR) is a regulation for protection of personal privacy that will be valid no later than May 2018. This regulation will control how the industry and the authorities will handle the information about the citizens. The regulation will control the type of information that is saved for each citizen, give the citizen the right to know what type of information that are saved, why it is saved, and be able to get the information deleted. The main focus of the GDPR is that each citizen should be able to control this information, and only information that is needed should be saved. Note that this regulation is valid for all systems world wide handling information about EU citizens.

Personal information is information that can be used to track a specific citizens and are divided into general personal information and sensitive personal information. General personal information is data or evaluation of a specific citizen like name, address, phone number, car registration number, the shops you are using, the roads you are driving on, how you are driving, and the web sites you are visiting to mention a few. Sensitive personal information is more related to your ethnic origin, political affiliation, convicted, health status and sexual orientation to mention a few.

This regulation will be a large challenge for most of the industry and the authorities because today IT systems try to save as much information as possible about the users, with this regulation as little as possible should be saved. To make it even more complicated, every citizens should be able to have an overview of this information, whenever they want. If this regulation is not fulfilled, a large penalty will be the result. This regulation should be remembered whenever you are designing a database system saving information about the users.

GDPR defines that your system must have documentation and policies on how personal data is handled. This means where the data is stored, how the information is used in your systems, and who can access this type of information. Then it is even more important not to duplicate this type of information in different systems, the information should be stored in only one location. In the GDPR, consider;

- Personal data might be in free-text fields, in unstructured data, and in data streams,
- A continuous process of maintaining and updating the systems,
- An organization wide focus as everyone in the organization handling personal data should know about the GDPR applies to them.

Some steps in managing the GDPR:

1. An organization is responsible for physical access to all stored data; investigate all data access and workflow in the systems regarding personal data,
2. Identify and analyze the handling of personal data in multiple sources and systems in the organization,
3. Define clear roles and responsibilities when the personal data has been identified, making a governance model,

4. Protect the personal data based on the governance model. The data can be protected based on the following de-identification techniques: 1) anonymization (remove identification information), 2) pseudonymization (replace identification information), and 3) encryption (encode identification information).

In SMART systems more data and information are needed about the usage of these systems, how to handle the storage and processing of this information. In a SMART building, how to handle the information about guests entering and leaving the building?

Part XII

Alarm systems

Chapter 39

Introduction

An alarm system will be an integrated part of any monitoring or controller system, and an important module of such a system. Often the alarm system will be integrated with several of the monitoring and controller systems present, but still it is desire to have only one common alarm system. Any alarm system should be based on some sort of standard, and the focus will be the YA-711[1] and IEC62682 specifications (IEC62682 2014) for alarm system design (YA-711 2001)(YA-710 2001). IEC62682 is a newer specification then YA-711, extending the functionality of the YA-711 standard.

Some historical information regarding alarm specifications is shown in Figure 39.1 showing the EEMUA 191, NAMUR 102, ISA 18.2 and IEC 62682.

An Abnormal Situation Management Consortium (ASM) was established in 1994 and helped Engineering Equipment and Materials Users' Association (EEMUA) to publish the first guidelines about alarm management, publication 191 (Sun 2016). User Association of Process Control Technology in Chemical and Pharmaceutical Industries (NAMUR) published their "Alarm Management" as publication NA-102 in 2003 (Sun 2016). International Society of Automation (ISA) approved the publication "Management of Alarm Systems for the Process Industries" in 2009 as ISA-18.2, more as a guideline rather than a standard (Sun 2016). The Norwegian Petroleum Directorate released the YA−710 and YA−711 standards in 2001. Several other organizations and institutes have released the same type of standards and guidelines showing the need for alarm standards and guidelines (Sun 2016).

An alarm system monitors the states of the system (process) and gives warnings or alarms at an abnormal or irregular condition or situation in the system. The operator has to deal with this warning or alarm. All alarms will be generated by the computer systems, the alarms will **not** be generated by the process. The alarms will be generated based on signals or events from the process or the computer system itself. The alarm system will be a module in the SCADA system as shown in Figure 39.2. The main function of an alarm system is to inform the operator about a situation that is not normal.

In addition to process alarm system other alarm systems exist as well like fire alarm system, intruder alarm system, and car alarm system to mention some.

An alarm system should also have an alarm management. Alarm management is a description of the usage of the system, the instrumentation engineering and systems thinking to manage the design of an alarm system to increase its usability.

An alarm system is a system that will monitors the process and give a warning or indication when an abnormal or specific condition is met. An alarm system can be a stand-alone system, an integrated part of a system, and/or a distributed part of a total system.

A stand-alone alarm system consists of the input devices, the alarm system, and the output devices for giving the warnings and alarms. An example can be an intruder alarm for a building and such a system is shown in Figure 39.3.

A stand-alone system contains the sensors, the computer, and the actuator as a complete system.

An integrated alarm system consists of a complete monitoring and control system where the alarm system is an integrated part of this monitoring and control system. This is shown in Figure 39.4.

A distributed alarm system consists of several monitoring and control systems where the alarm system is integrated to all of these monitoring and control systems. This is shown in Figure 39.5.

The reason for an alarm system is to inform an operator that some sort of action has to be performed. One or several values from the process have exceeded a limit and the system needs "help" getting control of the values. No need for an alarm system if the system can control the values itself. The alarm limits

[1] YA-710 is the Norwegian specification and YA-711 is the English specification.

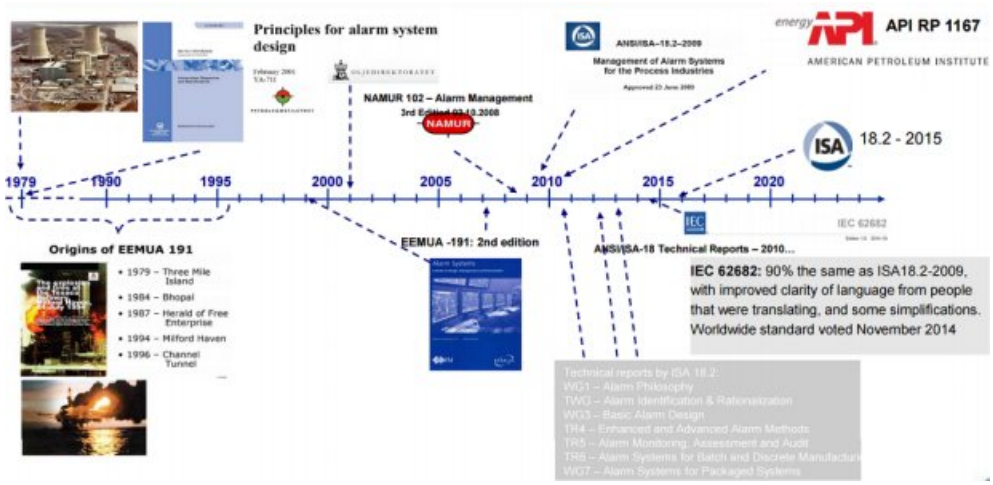


Figure 39.1: Some history of alarm specification, from a presentation on the "ARC Industry Forum 2015".

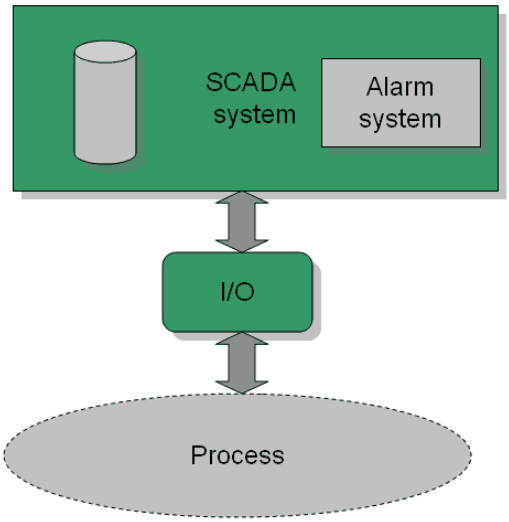


Figure 39.2: The alarm system will often be a part of the SCADA system. The alarms will only be generated based on signals and events from the plan, and internal events from the SCADA system.

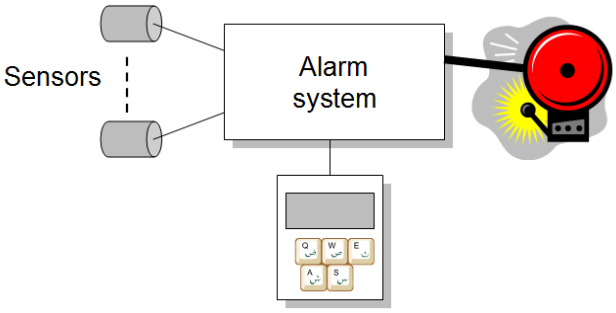


Figure 39.3: A stand-alone alarm system with the necessary I/O devices conneded directly to the system.

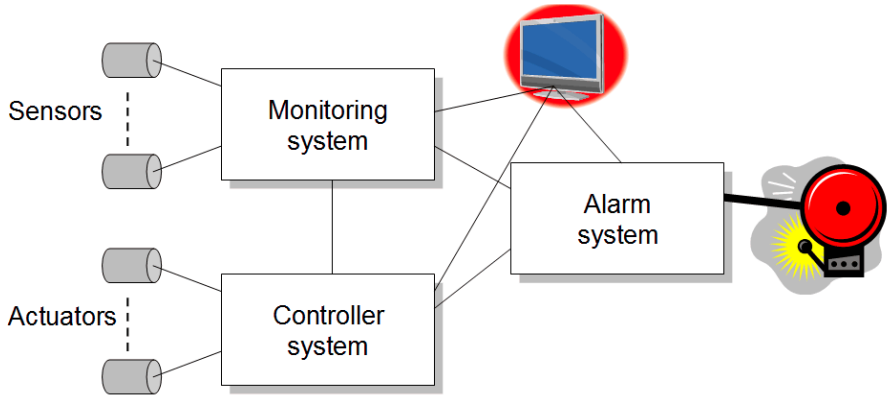


Figure 39.4: The alarm system is an integrated part of the monitoring and controller system.

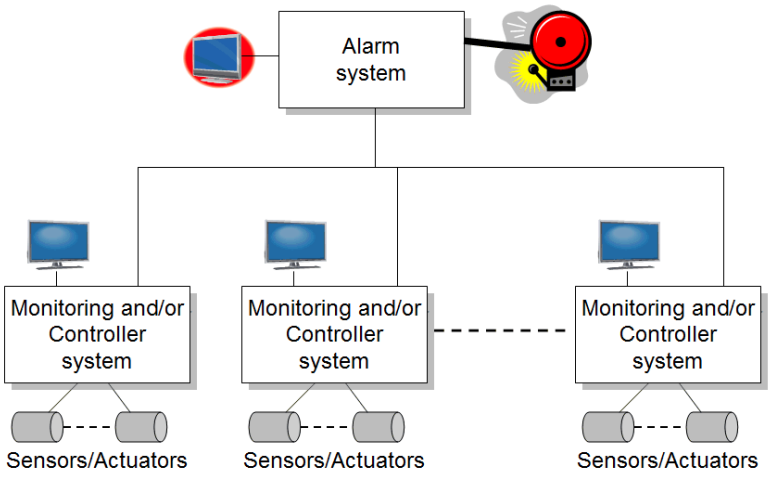


Figure 39.5: Several monitoring and/or controller system is sharing a common alarm system.

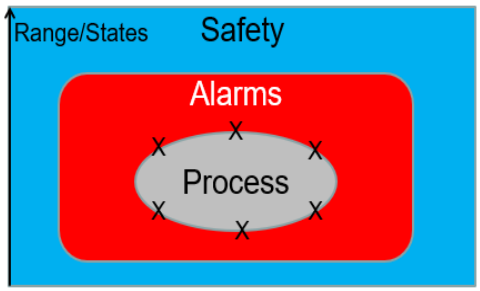


Figure 39.6: The purpose of an alarm system. If any of the process states or values are crossing the border from Process to Alarm, an alarm should be activated. If a process value is crossing the border between Alarm and Safety, a safety action should be activated. Note that these borders are only defined in the control system.

Figure 39.7: The alarm is generated in the PCS and any alarm state change is also handled by the PCS, not the process.

are managed by the system itself and an alarm is activated if a value is outside any of these limits. Figure 39.6 shows the purpose of an alarm and security system. If any value of any parameter is crossing the border between the process section and the alarm section, an alarm is activated. The responsible of the operator is to figure out why the alarm is activated and correct the operation so the value will again be within the range of the process. If the correction fails and the value is crossing the border between the alarm section and the security section, some sort of shutdown action will be activated.

An alarm can be defined as an audible and/or visible means of indicating to the operator an equipment malfunction, process deviation, or abnormal condition requiring user response. An alarm is generated in the process control system (PCS), not in the process. The PCS receives a process value and the alarm is generated by the PCS based on a set of predefined state limits. An overview of the alarm generation is shown in Figure 39.7.

Chapter 40

Management

40.1 Introduction

Alarm management is the combination of:

1. human factors or ergonomics,
2. instrumentation engineering,
3. systems thinking,

to design an alarm system for its usability. Alarm management is the process by which alarms are engineered, monitored, and managed to ensure safe and reliable operations. A normal challenge for an alarm system is either too few alarms or too many alarms. It is important to design the alarm system so it is clear how to use the system, clearly alarm messages, the setting of the alarm limits, how to deal with a lot of alarms raising at the same time. Several standards exist for Alarm Management like the EEMUA 191 (EEMUA 1999) and the ISA 18.2 (ISA 2016). EEMUA 191 (IEC62682 2014) is written as a text book and provides some examples in the appendixes, while the ISA 18.2 has been written more like a standard.

Some of the steps of Alarm Management:

1. Create and Adopt an Alarm Philosophy; A comprehensive design and guideline document that makes it clear "exactly how to do alarms right.". This document focus on the standards to follow, the type of alarms, alarm texts and the interaction between the alarm system and any operators,
2. Alarm Performance Benchmarking; Analyze the alarm system to determine its strengths and deficiencies, and effectively map out a practical solution to improve it,
3. "Bad Actor" Alarm Resolution; From experience, it is known that around half of the entire alarm load usually comes from a relatively few alarms. The methods for making them work properly are documented, and can be applied with minimum effort and maximum performance improvement,
4. Alarm Documentation and Rationalization (D&R); A full overhaul of the alarm system to ensure that each alarm complies with the alarm philosophy and the principles of good alarm management.
5. Alarm System Audit and Enforcement; DCS alarm systems are notoriously easy to change and generally lack proper security. Methods are needed to insure that the alarm system does not drift from its rationalized state.
6. Real-Time Alarm Management; More advanced alarm management techniques are often needed to ensure that the alarm system properly supports, rather than hinders, the operator in all operating scenarios. These include Alarm Shelving, State-Based Alarming, and Alarm Flood Suppression technologies,
7. Control and Maintain Alarm System Performance; Proper management of change and longer term analysis and KPI monitoring are needed, to ensure that the gains that have been achieved from performing the steps above do not dwindle away over time. Otherwise they will; the principle of "entropy" definitely applies to an alarm system.

Figure 40.1: The alarm philosophy is part of the alarm management process.

An alarm system should consists of 1) alarm levels (define a set of basic alarms), 2) alarm lists (next alarm on top or bottom, sort on active alarms?, ..) 3) alarm colors, 4) alarm actions, and *5*) alarm logging. A system should contain **only one alarm system module** and the design of the alarm UI is important. The design of an alarm UI is described in several media including the book (Bullemer et al. 2008).

The alarm management process should result in an alarm management document describing the alarm system for a specific company and plant. Often the alarm philosophy is also described in a document being a part of the alarm management document as shown in Figure 40.1.

40.2 Alarm Philosophy

Designing a new alarm system or modifying an existing alarm system require some basic planning. The first step in this planning process is developing an alarm philosophy (if not already existing), documenting the objectives of the alarm system and the processes to fulfill these objectives (IEC62682 2014). An alarm philosophy starts with some basic definitions of your alarm system and extends them to the operation definitions for the process (IEC62682 2014). New alarm systems should have the alarm philosophy as the basis for alarm system requirements.

According to the alarm management standard IEC 62682 the alarm philosophy is defined as a document that establishes the basic definitions, principles, and processes to design, implement, and maintain an alarm system. There exists no generic alarm philosophy document as these documents must be adopted to the details that are specific and relevant for each company, and processes of these companies. The alarm philosophy should at least include specifications of the GUI for the alarm system, alarm priorities, definition of alarm classes, alarm colors, alarm symbols, any alarm standard used, description of any alarm lists, the type of alarm lists, and any alarm reports. The alarm philosophy should be maintained throughout the lifetime of the alarm system, and should also be consistent for all alarm systems for the plant and/or company. Note that the alarm philosophy should not be general overview, but very specific to your alarm system.

The alarm standards ISA 18.2 and IEC 62682 are general descriptions of all type of alarm systems, covering all type of combinations. An alarm philosophy is a detailed description of your specific alarm system, how the system are fulfilling these alarm standards, but with focus on practical implementation. Typical questions that should be answered by the alarm philosophy are 1) how to configure the alarms, 2) how to acknowledge an alarm, 3) how to see the differentiate between an active and an acknowledge alarm, 4) how many alarms in your system, 5) how many and type of alarms lists, 6) how to handle alarm priorities, and 6) how to make a report to mention some. The alarm standards are also focusing on the key performance index (KPI) indicating the maximum number of alarms per time unit, like maximum one alarm every 10 minutes. How can you control the number of alarms activated per time unit in your system to avoid stressing the operator?

An alarm philosophy document should contain the following sections:

- Introduction; purpose of the alarm system, definitions, references (standards), roles and responsibilities,
- System; alarm classes and priorities, configuration and documentation,
- Design; User Interface, alarm list overview, type of alarm lists and special considerations,
- Operation; implementations, alarm response, training, testing, and system maintenance,
- Reporting; performance monitoring, history logging, and reports.

Note that the most important part for an alarm philosophy is that it MOST define a specific alarm system, your alarm system, not a general alarm system. Based on your alarm

philosophy the reader should understand how your alarm system is designed, and how to use your alarm system (how to see the difference between an active and an acknowledge alarm, how to list the alarms, how to acknowledge an alarm, and so on.).

Chapter 41

Specifications

41.1 Introduction

An alarm system contains a set of specifications defining the alarm types, alarm priority, alarm colors to mention some. The specification will often be part of a company alarm management or alarm philosophy system, a document describing the alarm handling and procedures. The specification should contain alarm definitions, alarm selections, alarm design principles and configuration, alarm priority determination and alarm handling methods to mention some.

The three major criteria for an alarm system should be (Hollifield & Habibi 2010):

1. any alarm event should require an operator action,
2. a single "process" event should only produce one single alarm event,
3. an alarm should only be generated from an abnormal situation.

All of these criteria are often violated in alarm systems, giving an alarm system that may be more "stressed" and appear more "complex" then necessary. This section will handle the specifications for a general alarm system with focus on the YA-711 specification.

41.2 Alarm System Functions

41.2.1 YA-711 specifications

The main functions of an alarm system is shown in Figure 41.1. The alarms are generated inside the SCADA system due to a set of system limits, process behavior and/or process status.

The main functions of an alarm system as shown in Figure 41.1 are:

- Signal filtering; processing of the raw input signals to the alarm system in order to remove signal noise and other information that is unimportant for the purpose of an alarm system, such as small, rapid oscillations.

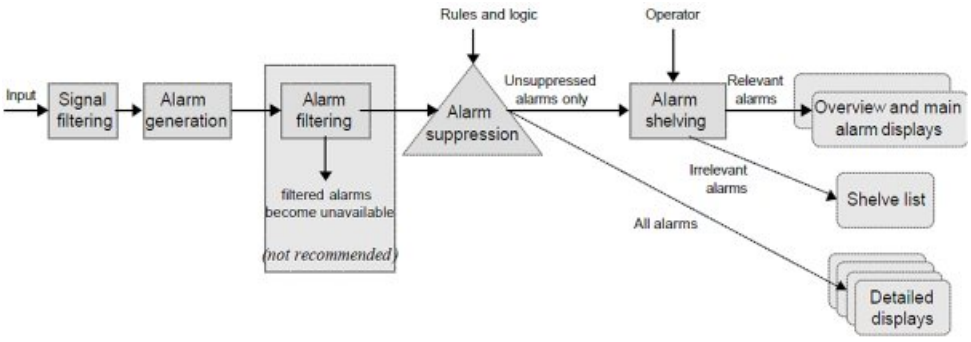


Figure 41.1: The main functions of an alarm system (YA-711 2001).

- Alarm generation; comparing the input signal with signal limits and checking the process and system states.
- Alarm filtering; preventing an alarm signal so that it is not available for the operator in any part of the system. Disable an alarm.
- Alarm suppression; preventing an alarm from being presented in main alarm displays, e.g. overview displays, but the alarm is still available in the system at a more detailed level. A set of rules or logic defines which alarms to present in which displays. These rules and logic will typically deals with any combination of inputs signals where each signal will generate an alarm, a sequence of input signals where each signal generates an alarm and redundant alarms. An important goal with this standard is to use alarm suppression in stead of alarm filtering. If a set of alarms generated in a short amount of time pointing to the same root cause, and the situation can be returned to normal with a limited set of actions, there is no need for multiple alarms. The alarm suppression module should then contain the logic to generate only one alarm.
- Alarm shelving[1] is a facility for manually removing an alarm from the main list and placing it on a shelve list, temporarily preventing the alarm from re-occurring on the main list until it is removed form the shelf. Shelving will normally be controlled by the operator, and is intended as a "last resort" for handling irrelevant nuisance alarms that have not been caught by signal filtering or alarm suppression mechanisms. A reason can be for maintenance of part of the system, for testing, or part of the system is currently not in use.
- Alarm lists with the important (main) alarms, the shelve alarm list and the detailed alarm list with all the alarms. Other alarm list can also be useful to integrate in the alarm system. Also note that any alarm should be present in the process view of the PCS, not only in the alarm list.

The difference between the Alarm suppression and Alarm shelving is that the operator can control alarm shelving and system designers or the supervisor can control the alarm suppression.

The IEC 62682 is an alarm standard defined in 2015, extending current alarm standards like YA-711 and ISA 18.2 − 2009 and will become the new worldwide standard for alarm systems.

41.2.2 IEC 62682 specifications

Some of the extensions in the IEC 62682 specifications includes timing, 16 alarm types and the definition of alarm attributes. An alarm in IEC 62682 is defined as an audible and/or visible signal, indicating to the operator an equipment malfunction, process deviation, or abnormal condition requiring a timely response. IEC 62682 is defining the alarm system as shown in Figure 41.2 where both the process control systems and safety systems are included. Alarms can be handled by panels, by basic process control systems (BPCS) and by safety instrumented system (SIS).

The word “timely” has been added to include some timing definition and recommendation, for response and performance. This will be referred to in a later section in this chapter. The specification also include the development of an alarm philosophy as part of the system and tracking of the “performance” of the alarm system. The specification also defines when an alarm can be suppressed like:

- shelving; time limited manual suppression of an alarm by the operator,
- suppression by design; automatic suppression based on operating conditions, process condition and predefined logic,
- out of service; let maintenance be responsible for the suppression, remove the responsibility from the operators.

The operator should be able to see the suppressed alarms in detailed displays or dedicated lists. The “performance” of the alarm system includes the average number of alarms for a 30 days period should be about 145 per day for each operator, maximum 288, should be about 6 for every hour, maximum 12, and the time the 10 most common alarms contribute to the total alarm should be about 1%, maximum 5% of the total alarm load.

[1] shelving: abandon; Norwegian=henlegge

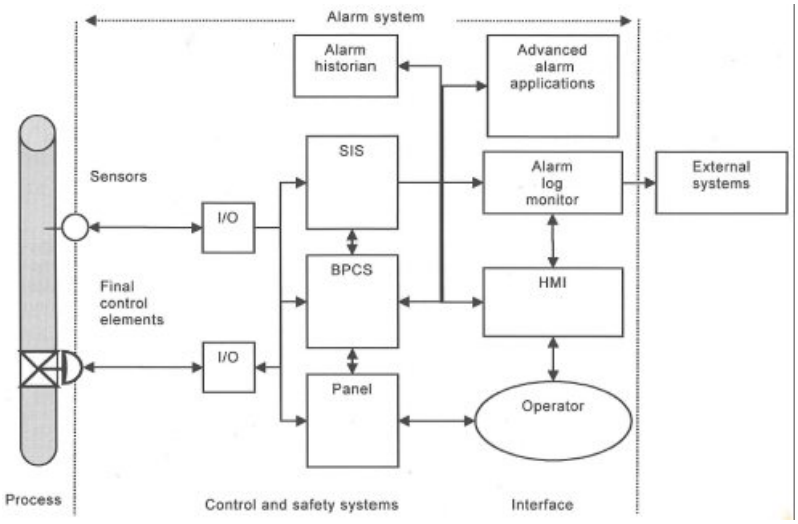


Figure 41.2: The alarm system dataflow between the process and the operator (IEC62682 2014).

41.3 Barriers

Barriers are functions and actions planned to stop or violate an unwanted course of events. Several different types of barriers exist for an automation system like a control system, alarms, operators, security systems and security values to mention some. A barrier gives control of different types of errors, and different barriers gives a higher security in the system. A type of error can damage one type of barrier, but has less probability to damage different types of barrier. An alarm can be the last barrier where the operator is responsible to avoid an accident.

41.4 Alarm Types

Alarms are always generated by the monitoring (or SCADA) system based on signals and events. The most used alarm types are (YA-711 2001):

1. Basic alarms; generated by detecting deviations on single process measurement, either analog or digital, or single pieces of equipment,
2. Aggregated alarms; generated by combining the state of a number of basic alarms. This combination should describe the state of the process system or equipment/sub-system more precisely than a single alarm,
3. Model based alarms; generated based on online simulations by mathematical models of the process or part of the process,
4. Key alarms; selection of important alarms presented in a way that makes them available and usable even during alarm overloads. All important safety-related alarms must be defined as key alarms, other alarms could also be included if appropriate.

41.5 Alarm Levels

An alarm system will be responsible for any alarms, warnings and informations in a monitoring and control system. An alarm system should be able to prioritize between alarms, warnings and information, but also prioritize between different types of alarms. A system contains different alarms like:

1. Critical level alarms like High High (HH) or Low Low (LL) alarms. In some systems critical alarms can be the High High High (HHH) or the Low Low Low (LLL) alarms. These alarms can generated from analog or binary signals,
2. “Normal” alarm condition like High or Low alarms. These alarms can generated from analog or binary signals,

3. Difference or change alarms, analog signals changing too much or too fast/slow. These alarms can be either a critical alarm or a normal alarm condition.

4. I/O device error,

5. System device error.

A system contains different alarms and/or warnings like:

1. Time-out; no input from a sensor or another computer system within a specific amount of time. Warning for a short amount of time and an alarm condition if a longer amount of time period. Time-out alarm can be tricky to detect as there will be no event when the alarm condition should be activated.

2. Out of range; the measured signal is outside a valid range for the process but still inside the range of the sensor device, or outside the range of the sensor device as well. Using a sensor with $4-20$ mA interface there should be an out of range alarm condition if the signal from the sensor is for example 0, 3.5 or 20.5 mA.

Note that the combination of normal and critical alarms are used for the normal alarm to be a sort of an early warning. The normal alarm will be activated first, and if the normal alarm is not handled within a requested time period a critical alarm will be activated. This also means that not all normal alarms will have critical alarms.

41.6 Alarm generation

All sensor signals contain noise giving that the measurement system may indicate a wrong value from the process system. If the value is outside the normal condition should an alarm event be generated? The signal is normal filtered but the alarm generation should be based on the raw value, the unfiltered value. One solution can be to require at least three continuous values outside the normal condition, but depends on the sampling time of the systems and time requirements in the system. Note that the alarm generation is inside the process control system, no alarm information is received from the process. A value or a state change is received from the process and only the alarm configuration of the process control system decides if an alarm should be generated or not. The accuracy of the sensor devices and the measurement system (DAQ devices) may also affect the alarm generation. The signal filtering module in Figure 41.1 is the filtering of the alarm signal, not the low pass filtering of the process signal.

The YA−711 standard (YA-710 2001)(YA-711 2001) is require a short response time from alarm generation until the alarm is present in the overview and main alarm display, maximum two seconds after activation.

41.7 Alarm Priority

An alarm is always important but there may be alarms that are more important then other alarms for an operator to handle first. Therefore should a process control system allow for multiple alarm priorities to be able to distinguish between alarms. The best practice is to use maximum three levels of priority (Hollifield & Habibi 2010). The distribution of these alarm priorities should be about 3% to 7% of all alarms for the highest priority ($\approx 5\%$), about 15% to 25% for the middle priority ($\approx 15\%$), and the rest, 70% to 80% for the lowest priority ($\approx 80\%$). However, there is no requirements for using alarm priorities, and quite normal to use only the lowest priority.

Note that alarm priority is not the same as alarm state, each alarm has several states, but a specific alarm can only have one type of priority. The priority is often defined based on the type of process, plant and how important that specific alarm type is.

41.8 Alarm colors

The alarm color should be red, and the red color should be used only for alarm indication. Yellow and Orange (or purple) can also be used for sub functions in an alarm system like warnings etc. Note that alarm color(s) should ONLY be used for alarm indications. Lamps can have different colors as well in

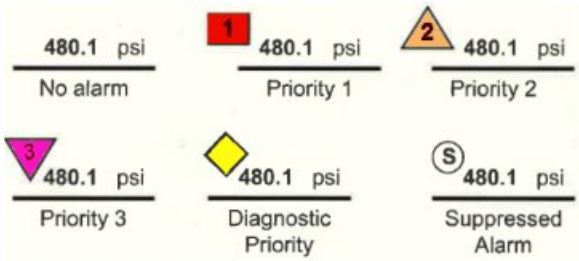


Figure 41.3: A suggestion for multiply coded alarm indication elements, with priority, and without blinking (Hollifield & Habibi 2010).

cooperation with a horn. The alarm management system or the alarm philosophy will define the colors to be used in addition to the red color.

When designing an alarm system take into consideration that the operator can be color blind, meaning that they can mix the information with green and red color.

Blinking text should be avoided as it is annoying and it is possible to see an empty display if looking when the text is off. It is better to include a symbol close to the process value indication the alarm state and also the alarm priority. However the symbol can blink to indicate an unacknowledged alarm (Hollifield & Habibi 2010). Different symbols with different colors should be used for different priorities. An example can be a red square for high priority, a red triangle for medium priority and a red circle for low priority. Different colors can also be used on different priorities, these properties should be defined in the alarm philosophy document. Suggested color choices can be red for the highest priority, orange or yellow for middle priority and orange or magenta for the lowest priority.

Figure 41.3 shows a suggestion for combining alarm elements with different priorities, and avoiding blinking.

41.9 Alarm States

Warnings are not critical states and will just be on and off states. An alarm is treated as a critical state and must be acknowledged by an operator. It is the responsibility of the operator to do the right actions for this type of alarm. An alarm will then have three different states like:

1. Passive alarms; no alarm, the condition is within the normal operating range. This condition will not be present in an alarm system.

2. Active alarms; the condition has crossed an alarm boundary and the operator is noticed by some sort of output device. The alarm will be present until the alarm is acknowledged by an operator. This state of the condition will be present in the alarm system, often together with some blinking indicators and/or horns,

3. Acknowledge alarms; an active alarm that is acknowledged by an operator. The acknowledged alarm will be passive when the condition is within the normal operating range. This state of the condition will be present in the alarm system, often together with some sort of steady indicators. The alarm condition will be removed from the alarm system when returning to the normal operating range or when the alarm is turned off.

Different combinations of these alarm states are shown in the Figures 41.4 and 41.5.

Figure 41.4 shows a situation where the value is entering an alarm state at time t_1, the user is acknowledging the alarm at time t_2, and the value is no longer in alarm state at time t_3. Figure 41.5 shows a situation where the value is entering an alarm state at time t_1, the user is delaying the acknowledging of the alarm so the value is no longer in alarm state before the alarm is acknowledged. The alarm condition is still an active alarm until the alarm state is acknowledged by the user.

The transitions between the alarm states are:

- a passive alarm can only change to the active state; the state change will occur when the process value is passing an alarm limit,

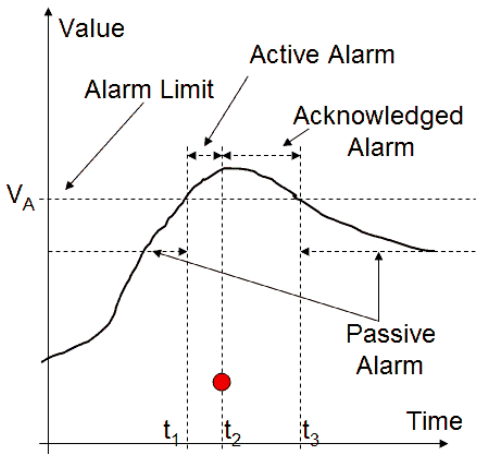


Figure 41.4: A value is making an alarm condition at time t_1, the alarm is acknowledged at time t_2, and the alarm condition is passive at time t_3. The alarm is created in the alarm system at time t_1 and is removed from the alarm system at time t_3.

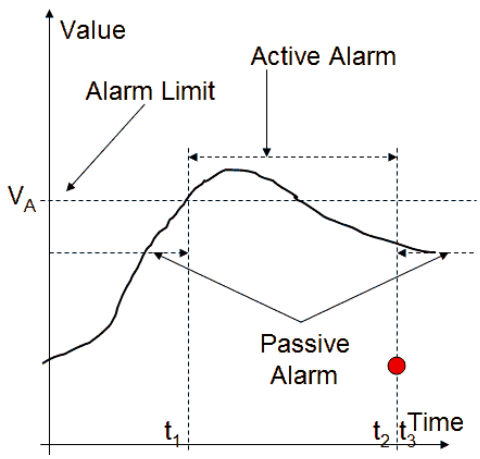


Figure 41.5: A value is making an alarm condition at time t_1, but the alarm condition is passive before the alarm is acknowledged at time t_2. The alarm is created in the alarm system at time t_1, but is not removed from the alarm system unntil the alarm is acknowledged at time t_2. The acknowledged time t_2 is equal to the removal time t_3.

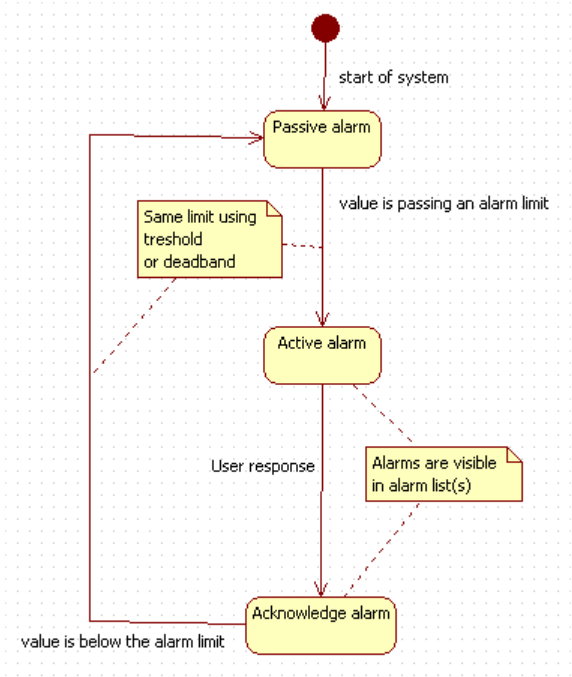


Figure 41.6: The state diagram of the alarm states, shown with an UML state diagram.

- an active alarm can only change to the acknowledge state; the state change will occur when the operator is responding to the alarm by some sort of action including an acknowledge of the alarm,
- an acknowledge alarm can only change the state to the passive state; the state change will occur when the process value is below the alarm limit.

The alarm states are shown in a state diagram in Figure 41.6. There can only be one alarm of each type in the alarm list shown in the Figure as the user has to respond to each alarm and the alarm has to be removed from the list before the same alarm can be activated.

An alarm system with support for shelving and/or suppression must have additional alarm states to handle these states. The IEC 62682 has included three additional states to handle these states; 1) shelved, 2) suppressed by design, and 3) out of service (IEC62682 2014). The shelved state is controlled by the operator, the suppressed by design state is controlled by the plant operating states, and the out of service state is controlled by any maintenance states of the plant.

41.10 Alarm Shelving

Alarm shelving provides a way for the operator to manage nuisance alarms that are popping up from time to time due to malfunction or abnormal conditions (exida 2017). These malfunction or abnormal conditions can be repeatedly transitions between active and passive states (chattering and fleeting alarms), condition that remains in active state for an extended period of time (like more than 24 hours) (stale alarms) or the most frequently occurring alarms (exida 2017). A shelving option in an alarm system is the best way to reduce an operator error as this affect the impact of a nuisance alarm in the system. Alarm shelving is a safe and secure way of suppressing a nuisance alarm and is fully controlled by the operator. The nuisance alarm should be temporarily moved from the alarm main list to the shelved alarm list, and stay in this list until it is cleared by the operator or a maximum time period has elapsed. Note that shelving is not disabling nor deactivating an alarm as the alarm is still active in the system, but not in the alarm main list (overview list). Disabling or deactivating the alarm means to remove the alarm information from the alarm system. Two types of shelving exists (exida 2017):

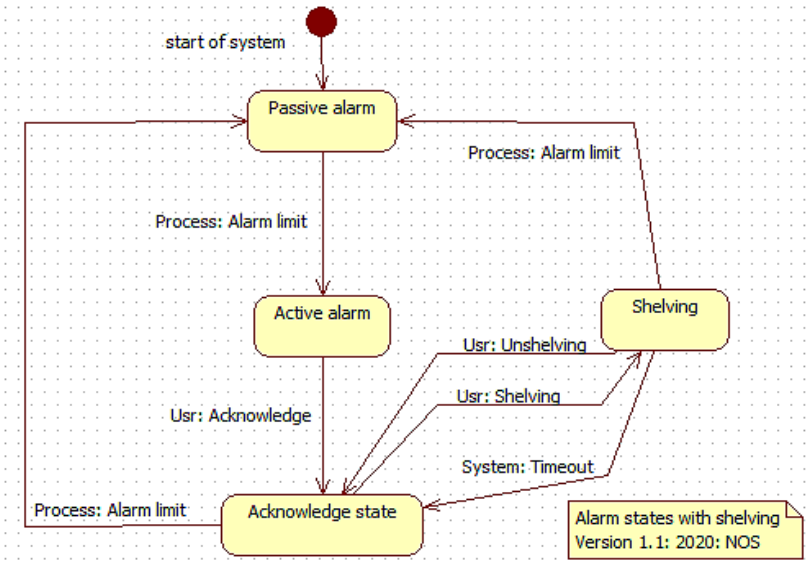


Figure 41.7: Alarm state diagram with shelving. Several transistions for changing the alarm state between active and shelving.

Type	Behavior	Alarms
Continuos	Automatically re-shelved	chattering
	Shelved for a maximum time only	fleeting
One-shot	New alarm will be active	stale
	must be re-shelved very time	

Most alarm systems with the shelving option support just the one-shot shelving while some alarm systems with shelving support both types. Alarm shelving should be part of the alarm philosophy including information about which alarms that can be shelved, the maximum time for an alarm to be shelved, the maximum number that can be shelved at the same time and describe the shelving procedures.

One solution of alarm states for a system with shelving is shown in Figure 41.7 where several states have been added for the shelving state.

The state diagram is based on the alarm state diagram in Figure 41.6 and adding the new shelving state. Note that in this case only acknowledge alarms can be shelved. The new transitions for the shelving state are:

Transition	Description
Usr: shelving	User selects an alarm for shelving, move from main to shelve list
Process: alarm limit	In one shot type will the alarm state change from shelving to active when re-activated.
Usr: unshelving	User unshelves an alarm, move from shelve to main list
System: timeout	An alarm is automatically moved from shelve to main list

The number of alarms in the shelve list should not increase over time, and the alarm philosophy should define the maximum number of alarms allowed at the same time to be in the shelve list.

41.11 Response Time

The response time of an alarm system should be maximum two seconds, preferably less than one second. The requirement in the YA−711 standard is maximum two seconds(YA-710 2001)(YA-711 2001). The IEC 62682 has no recommendation for a specific response time, but have some recommendation for the performance of the alarm system, see the "*Performance*" section. The operator may redo the operation if no response within the maximum response time. The response time for information in industrial systems can be divided into the following categories and classes:

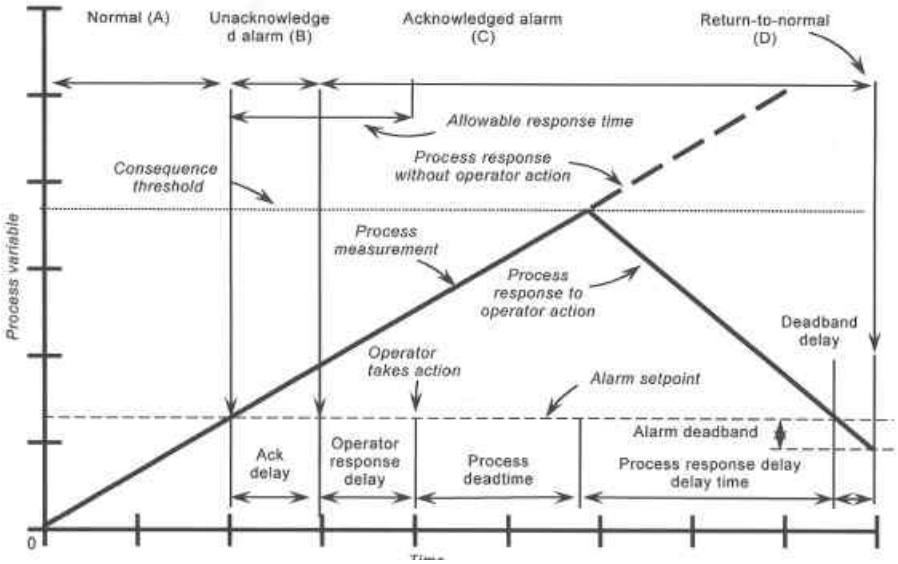


Figure 41.8: The alarm respons timeline for a process variable increasing above the alarm limit, showing the delays and responses (IEC62682 2014).

Category	Class	Application	Description
Safety	0	Emergency action	Always critical
Control	1	Closed loop regulatory control	Often critical
	2	Closed loop supervisory control	Usually non-critical
	3	Open loop control	Human in the loop
Monitoring	4	Alerting	Short term operational consequence
	5	Logging and downloading/uploading	No immediate operational consequence

The YA−711 standard (YA-710 2001)(YA-711 2001) is also defining a maximum time resolution in the alarm lists and alarm displays of one second. The same standard is also defining that the resolution in the alarms- and events log should be less than 100 ms. The alarm suppression module should also have a time resolution of less than 100 ms.

The IEC 62682 standard is defining a set of time periods or time intervals, but is not defining any specific time. The time periods is defined as shown in Figure 41.8 for a process variable increasing above an alarm limit. The Figure shows the different responses, delay periods, and dead time periods.

41.12 Performance

The performance of an alarm system is an indication of the number of alarms that the system can handle. As a human is included in the alarm "loop", the number of alarms must be adjusted according to the humans. The IEC 62862 alarm standard has included a recommended alarm performance metrics (IEC62682 2014) as shown as a table in Figure 41.9. This Figure shows an acceptable limit as an average and the maximum for at least the last 30 days of data.

The alarm system key performance indicator (KPI) was concluded in the Brandsby & Jenkinson survey from 1988 that one alarm per ten minutes was consider as acceptable. One alarm per ten minutes was very fast converted to six alarms per hour, but this is wrong. Why? Because these six alarms can still be activated within a time frame of ten minutes. Two main situations are to be consider when evaluating the KPI of the alarm system, 1) steady state operation and 2) abnormal operation (startup, testing, configuration, ..). Remember that an alarm system detects and annunciates an abnormal condition, malfunction or process deviation which requires human intervention. Alarms should have some documented action or actions and time or workload should be estimated for each alarm, as part of the

Alarm performance metrics based upon at least 30 days of data		
Metric	**Target value**	
Annunciated alarms per time	Target value: very likely to be acceptable	Target value: maximum manageable
Annunciated alarms per day per operator console	~144 alarms per day	~288 alarms per day
Annunciated alarms per hour per operator console	~6 (average)	~12 (average)
Annunciated alarms per 10 minutes per operator console	~1 (average)	~2 (average)
Metric	**Target value**	
Percentage of hours containing more than 30 alarms	~< 1 %	
Percentage of 10-minute periods containing more than 10 alarms	~< 1 %	
Maximum number of alarms in a 10-minute period	≤ 10	
Percentage of time the alarm system is in a flood condition	~< 1 %	
Percentage contribution of the top 10 most frequent alarms to the overall alarm load	~< 1 % to 5 % maximum, with action plans to address deficiencies.	
Quantity of chattering and fleeting alarms	Zero, action plans to correct any that occur.	
Stale alarms	Less than 5 present on any day, with action plans to address.	
Annunciated priority distribution	3 priorities: ~80 % low, ~15 % medium, ~5 % high or 4 priorities: ~80 % low, ~15 % medium, ~5 % high, ~< 1 % highest Other special-purpose priorities (e.g. system diagnostic alarms) excluded from the calculation.	
Unauthorized alarm suppression	Zero alarms suppressed outside of controlled or approved methodologies.	
Unauthorized alarm attribute changes	Zero alarm attribute changes outside of approved methodologies or MOC.	

Figure 41.9: A summary of the recommended alarm perfomance metrics in the IEC 62682 alarm standard (IEC62682 2014).

alarm philosophy. The alarm philosophy should also state the alarm rates otherwise will the alarm system KPI be compared with the standards and guidelines, and specify a time interval instead of a fixed time period.

Chapter 42

Configuration

An alarm needs an active band with different on and off limits otherwise will the system generate a lot of on and off events. This will be part of the alarm generation section.

42.1 Dead band

A dead band (sometimes called a neutral zone) is an area of a signal range or band where no action occurs (the system is dead). Dead band is used in voltage regulators and other controllers. The purpose is common, to prevent oscillation or repeated activation-deactivation cycles (called 'hunting' in proportional control systems). Dead band is different from hysteresis. With hysteresis there is no dead zone, the output is always in one direction or another. Figure 42.1 shows the "conversion" between an input and output value in a system with a dead band configuration.

42.2 Hysteresis

Use different values to turn the alarm on and off. The reason is to use different limits for turning the alarm on and off, otherwise can if the process value is stable at the alarm limit trigger the alarm on and off at the same. Hysteresis will see to that the alarm will be triggered at different values for on and off state of the alarm. See Figure 42.2 for how the alarm can be turned on at the *unloading* graph and turned off at the *loading* graph.

42.3 Alarm limits

The alarm limits will be the value for activating the alarm. A high alarm will be activating when the process value is equal or higher, a low alarm will be activated when the process value is equal or lower. Typical alarm limits:

1. High; activated when equal or higher,
2. High High; security or critical "normal" alarm, activated when equal or higher,

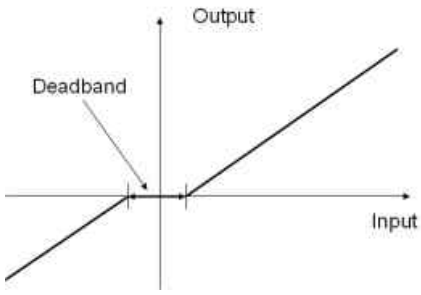


Figure 42.1: The deaband used to make the output "dead" while the input is still changing.

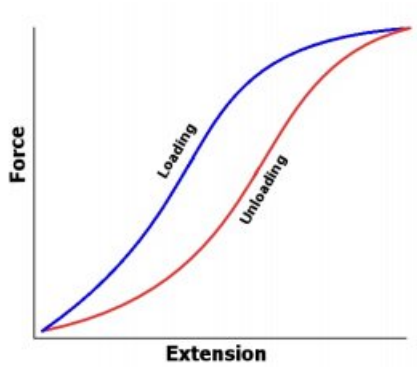


Figure 42.2: Hysteresis, using different different values for turning the alarm in on and off states (www.wikipedia.org 2006).

3. Low; activated when equal or lower,
4. Low Low; security or critical "normal" alarm, activated when equal or lower,
5. Difference; activated when the difference between two process values is equal or larger than this limit, or one process limit is larger than this limit. Difference can also be that one process value should not change more then within this range (+/- difference) based on the process value when setting the alarm.

The alarm limits, including hysteresis/dead band, must be configurable for all alarm types in a specific system. Some of the alarm limits can only be changed as an administration option, other alarm limits can be changed by the operator. Dead band is shown in Figure 42.1 and hysteresis is shown in Figure 42.2. The hysteresis or dead band may be a system parameter depending on the type of parameter or the type of I/O device, or the parameter specific settings can often be configured.

42.4 Fail/Safe

Fail-safe or fail-secure describes a device or feature which, in the event of failure, responds in a way that will cause no harm or at least a minimum of harm to other devices or danger to personnel. Fail-safe devices are distinguished from fail-secure devices in that failure disables any security features. Thus, a fail-safe lock will unlock during a failure whereas a fail-secure lock will remain locked during a failure, but cannot be unlocked.

42.5 Alarm list

Any alarm display, reports and/or list of the current alarms in the system should minimum contain : 1) alarm id, 2) tag name, 3) text description, 4) type of alarm, 5) alarm limit, 6) priority (if any), 7) activation time and 8) acknowledge time, if any.

An example of an alarm report is shown in Figure 42.3 indicating some of these properties.

42.6 Alarm devices

I/O devices available for alarm information and/or handling:

- Screen; display the alarms, the display should be able to sort on alarm priority, differentiate between active and acknowledge alarms, and focus on the most important alarms,
- Keyboard; alarm operations like silence, acknowledge, shelving and sorting,
- Horn; indicate a new active alarm, or security alarm, can be turned off by the silence key (until the next active alarm),

Date : 10/01-2009 (GMT) 09:33:36 Code: log-test:net-tes Page 1

MT Skarpefjord
ALARM REPORT

Condition: CM Online Cond

Num	Location	Message	Info	Alarm Time	Ack Time
1	C3	High Volume	10,0 %	10-JAN 09:32:38	
2	C1	High Volume	10,0 %	10-JAN 09:31:50	10-JAN 09:32:48
3	C2	High Volume	10,0 %	10-JAN 09:32:59	10-JAN 09:33:07

Figure 42.3: A printed alarm report with information about each alarm in a system.

Figure 42.4: A selection of different type of lamps and horn for alarm indications (www.tormatic.no: nov-08).

- Lamps; indicate an active alarm, or an active alarm by blinking and an acknowledge alarm by a steady light. Different lamp colors can be used in cooperation with horn to indicate different type of alarms. Different types of alarm lamps are shown in Figure 42.4.
- Printer; logging of changes of the alarm states, changes of alarm limits or other changes of configuration.

42.7 Alarm logging

Often an alarm system will have requirements for making different types of alarm reports. The conditions for logging for the reports can be: 1) The change of an alarm state (Active/Acknowledge/Passive/Shelve), 2) The change of the alarm limits for any process condition, 3) The change of the alarm activation (on/off), 4) Any change of the alarm configuration, 5) Who changed the alarm setting(s) and 6) Time of change.

Chapter 43

Alarm system advises

Some advice[1] for alarm reduction:

- always start with an alarm philosophy ,
- important that specific persons have ownership (responsibility) for specific alarms,
- take care of both the operator and the alarms in the control room,
- create an alarm (project) group with regularly meetings,
- use individual dead band on all alarms, normally 1%, but may be increased on "unstable" measurements,
- if you want to add an alarm in the PCS, try to remove another alarm,
- remove warnings for automatic actions,
- be critical to new project or modifications, be sure that the alarm philosophy is followed,
- use alarm statistics,
- remove any radio or newspaper from the control room.

[1] Given by an alarm guru in Norway, Jan Grønlid, working at Ineos, in the Norwegian magazine "Automatisering" in March 2010.

Part XIII

Multitasking systems and Real-Time requirements

Chapter 44

Introduction

A multitasking system is a computer system able to execute multiple tasks concurrently for a certain period of time (Tanenbaum & Woodhull 1997), (Bach 1986). Normally the operating system (OS) handles the concurrent execution of these tasks, controlled by an OS service named scheduler. Most operating systems today are multitasking, like Windows, Linux, Android OS and iOS to mention a few. However, these tasks are software application designed and developed by software developers so each software developer should be responsible for making applications that can execute together with other applications (tasks) in a multitasking environment. Larger software systems should/can be divided into several tasks running in parallel by the OS, and the structure of the software system can often be simplified by exploring a multitasking OS.

A Process Control System (PCS) consists of among other things the SCADA system, a software system normally running on a multitasking system. A SCADA system will often be divided into several concurrent tasks (or processes or applications) controlled by a multitasking operating system monitoring and controlling a physical process. The structures and requirements for the PCS will often define how to divide the SCADA system in several tasks. This will be the responsibility of the software designers and/or software developers to design the software for the PCS in such a way that it will run in a multitasking environment together with other tasks.

A real-time system means a computer based system where one or more of the applications must be able to "synchronize" with a physical process. "Real-time" means that the computer system is monitoring the states of the physical process and must respond to changes of one or more of these states within a **maximum time**. A real-time system can be used for monitoring of different parameters in the physical process for presentation, warnings, alarm situations and for control. The control is possible by changing any of the input variables to the physical process. A typical system is shown in Figure 44.1 where a real-time system is influenced by events from in the physical process and the real-time system is using the information from these events for control of input variables to the physical process.

A real-time system does not mean as fast a possible, a good design of a real-time system just means as fast as necessary to satisfy the requirements of the system. Events are measured using sensor devices. Signals from sensor devices will always be delayed due to analog to digital (A/D) conversion and low pass filtering meaning that there will always be some delay in measurement and control systems.

In a distributed system messages must be sent between different modules. One part of such a system is the user interface where the operator can make a change and the system can respond with an indication when the change is executed. The user interface must get the change from the operator, send the information to a remote module using a communication protocol, the remote module must execute the

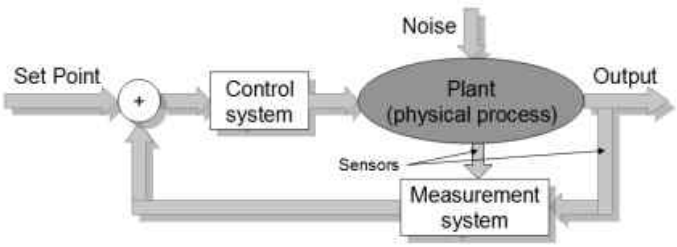


Figure 44.1: A real-time system for monitoring and control of a physical process.

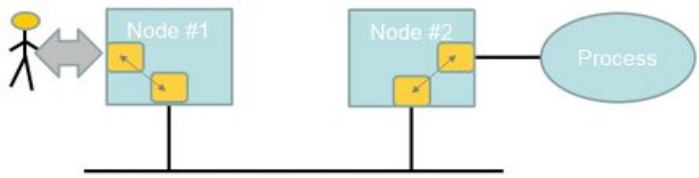


Figure 44.2: A distributed system where part of the system is running on different nodes. The communication between the nodes must be based on a set of protocols.

change and respond with information back to the user interface. To have this update in real-time a maximum time must be defined for the system. A maximum time of one second is normally accepted for an user interface. Figure 44.2 shows one possible configuration of such a system.

44.1 Maximum time

The important part of a real-time system is to react within a maximum time. This means that a calculated value or an output signal must be available within a maximum time. Normally a real-time system will have several different time requirements depending on plant sections or plant domains and devices used within these sections or domains. Some of the sections or domains can also be without any time requirements meaning that this part is not a real-time system.

A normal way of developing such a system is to divide it into several tasks as software applications where each application will have the responsibility for one or a few time requirements. This means that several applications will be running in a real-time system cooperating in solving time requirements for the whole system.

Sensor signals should also be time stamped in a real-time system to have better control of any delay and traceability of these signals.

44.2 Real-Time Examples

Some examples where time is important. Note that it is the maximum that is important, not as fast as possible.

44.2.1 Example #1: Buffertank

In Figure 44.3 a real-time system is used for monitoring and controlling the liquid level in a buffer tank with a capacity of 20 liter. The buffer tank should never be empty nor full, and the liquid level is monitored using a high level switch at 19.5 liter and a low level switch at 0.5 liter. The output from the switches is low when not covered by liquid and high when covered by liquid, with a delay of 0.08 to 0.1 second. The pump is controlled by a simple ON/OFF signal and the pumping rate is 1 liter per. second. The delay for the ON/OFF control of the pump is maximum 0.2 second.

What will be the real-time requirements for this system? Will it be a real time system?

The real-time requirements are to start or stop the pump based on the sensor signals without the buffer tank being overfilled or empty. The pump capacity is 1 liter per. second giving 0.5 seconds to start or stop the pump from the low or high level sensors. In a real-time system the worst case timing should always be used so the high level sensor, having the largest delay time, will be used. The real-time requirements for the system will be:

$$t_{RT-req} = t_{available_{MAX}} - t_{pump_{MAX}} - t_{sensor_{MAX}} = 0,5 - 0,2 - 0,1 = 0,2\,\mathrm{s}$$

meaning that the system will work if the RT system can deliver the pump signal within $0,2$ second. If the system can not react within $0,2$ second, the system will fail.

Can you use a PLC with a cycle time of $0,150$ second as the RT system?

What will be the solution if the capacity of the pump is increased to 2 liters per. second?

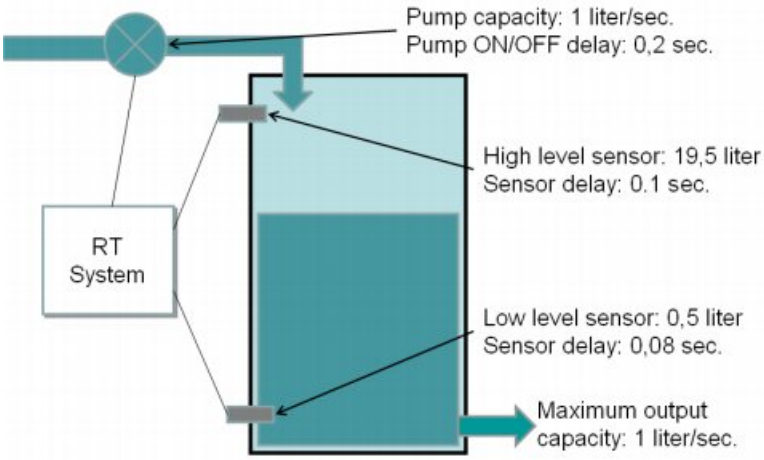


Figure 44.3: Buffer tank with low and high control of the level, a pump and a real-time system controlling the pump based on the low and high level sensors.

44.2.2 Example #2: Data Communication

Computer systems can communicate by using serial ports based on asynchronous communication. The sending is OK, what about the receiving of bytes? The receiving node do not know when receiving the next byte. The baud rate can be any settings like 9600, 19200, and up to 115000 bits per second. Since the receiving node do not know when the next byte is coming, interrupts and interrupt functions should be used. The communication will in this case be "simplified" to that each character contains 10 bits, 8 data bits and two control bits, and one interrupt for each character. The simplification is based on the number of control bits used in the protocol and the number of data bits. A baud rate of 115000 bits per second means in this case 11500 interrupts each second and the interrupt functions must be executed the same number of times. Assuming that the interrupt function takes between 10 ns to 25 ns, the timing (in seconds) will be:

Baud rate	Interrupts	Minimum	Maximum
		μs	μs
9600	960	$9,6$	24
19200	1920	$19,2$	48
38400	3840	$38,4$	96
115000	11500	$115,2$	288

All the interrupt functions MUST finish within the maximum time, otherwise information will be lost. The table contains the timing for one serial port only, what will be the situation for two serial ports? And four serial ports? Any real-time requirement?

44.3 Operating System

An operating system (OS) is a software module that manages the computer hardware, and provides services for any applications. The operating system will load when the computer is starting, before any application, and be an interface between any application and the computer system. An operating system consists of modules controlling I/O devices like disk, network, display, keyboard and mice to mention some, and resources like memory, disk and network. The operating system consists of device drivers at the lowest level to handle the interface with all the hardware devices, some sort of memory management system and a scheduler for handling the applications.

44.4 Synchronization and multitasking

Applications in process/industrial control systems must run together with the physical process, so the control system must be able to manage simultaneity. Several or even all the events may occur at the same

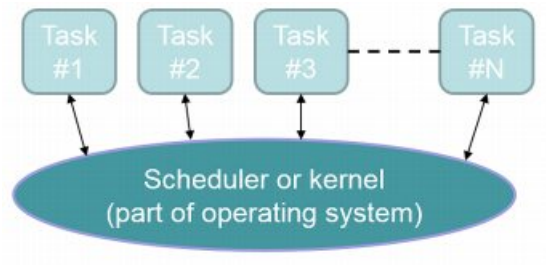


Figure 44.4: Using services (scheduler) in the operating system to handle multitasking.

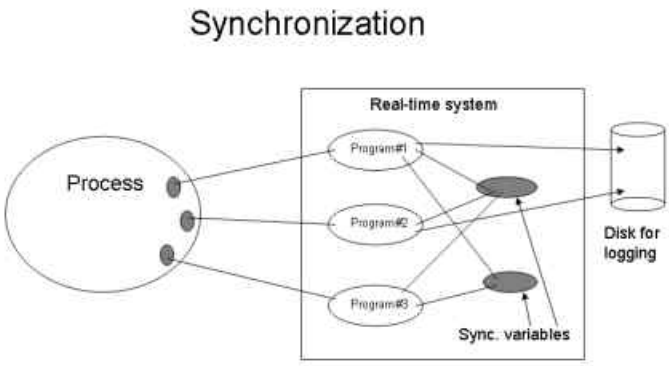


Figure 44.5: Syncronization in a real-time system.

time. The solution is to run several applications on the computer system or the different applications on different computers in a distributed system. Each application can then handle only a few of the events. This requires a multitasking solution and is often solved by using a scheduler. The scheduler must be part of the kernel or operating system for the computer. Figure 44.4 shows that the scheduler, a service in all operating systems, can handle a set of tasks as a multitasking system.

These solutions, by using several application on one computer or in a network, requires some sort of synchronization between the applications and between the applications and the physical process. Probably the most used way of synchronization between the applications and the physical process is the usage of sensors, while the synchronization between the applications is the usage of global variables or messages. This is shown in Figure 44.5.

44.5 Resources

When several applications are running "simultaneous" on a computer system, there must also be some control of the usage of the resources in the computer system. Resources can be both hardware and software, like I/O devices[1], global variables in the software, the CPU, memory, disk etc. One example is the printer device, only one application can use the printer at a time. If several applications are printing at the same time will the text be mixed based on how the scheduler are running the different tasks. Another example is the usage of a global array (table), one task is writing to the table and another task is reading from the table. It is important that the "writing" task is finished with the whole table before the "reading" task starts reading information from the table. Figure 44.5 shows the usage of the disk as a common resource for logging for both program #1 and program #2. The usage of synchronization services in the operating system (or kernel) is important to get a correct logging.

Regarding the usage of resources in a real-time system, the most important resource will be the CPU.

44.6 Programming

Programming of office systems will be based on a sequential data flow, meaning that the application will be started, will execute a set of operations in a fixed sequence. The system will use the data already

[1] I/O devices: Input and/or output devices.

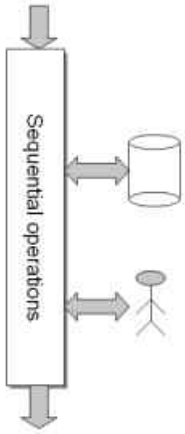


Figure 44.6: The sequential dataflow of an office software application.

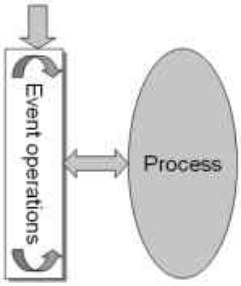


Figure 44.7: The event based operations of a real time software application.

available in the system or the user, and will end when the operations are finished. The application consists of a set of functions called in a sequence decided by the input data and the user responses. This type of behavior is shown in Figure 44.6.

A real-time system will be running in parallel with a physical process and must be running as long as the physical process is running. Many processes will be running all the time, for such systems the real-time system must be a 24×7 system[2]. A real-time system is executing the operations depending on events from the physical process and must react within a specific time on these events. The system will normally consists of several applications with a set of functions called when an external event occur. Data will not be available in the system, the real-time system must read the data from the physical process when needed. This type of behavior is shown in Figure 44.7.

A real-time system is divided into several software applications often called a process, a task, an application, or a program. These difference of these names will be explained later. A real-time system will be solved by the software, by dividing each part into different background processes that can handle specific events, independent of other background processes running in the system. Large PCS, partly or without any real-time requirements, may also have a better structure for making any extensions and/or maintenance, by using background processes. The only way of making background processes is by making the application as a multitasking application by exploiting the possibilities of threads and time events in the operating system. This means that software designers and software developers must have some knowledge about multitasking strategies and multitasking services in a multitasking operating system.

44.7 Embedded system

An embedded system is a special-purpose system in which the computer is completely encapsulated by the device it controls, a system dedicated to a specific purpose. An embedded system may not have any real-time requirements, but the concept "embedded system" is often used for "real-time system" and vice versa. The differences are:

[2] 24×7: 24 hours $\times$ 7 days: The system must be running all the time, can not be stopped or restarted!

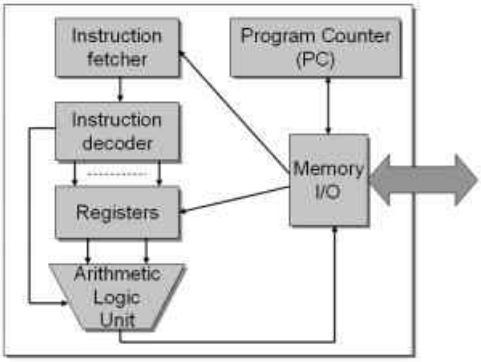


Figure 44.8: The main parts of a Central Processing Unit (CPU), the important part for a real time system is the registers and the PC.

1. An embedded system is often a system that should work without interaction with a user, will often be a "black box".
2. A real-time system is running in parallel with a physical process, requirements for simultaneousness and the reaction of external events have to be within a specific time.

Remark 12 *A real-time system will very often be an embedded system, while an embedded system do not need to be a real-time system.*

44.8 CPU and micro controller

The CPU, Central Processing Unit, is the main unit in a computer system. The CPU consists of a set of registers, a program counter (PC) having the address for the next CPU instruction, an Arithmetic Logical Unit (ALU) for the operations and logic for the control of the memory address. In real-time systems often a lot of I/O is needed so a micro controller can be used instead. The main contents of a CPU is shown in Figure 44.8. A micro controller is a CPU with different type of I/O units integrated. The CPU or the micro controller will normally be the most important resource in a real-time system and a scheduler is used to handle this resource. A real-time system must be able to run several software processes using a scheduler, a service in the operating system. The scheduler will handle multitasking, switching between the software processes as the cpu can only run one or a few software processes at the same time. The number of software process at the same time is depending on the number of cores for the CPU. The state of the CPU is based on the registers and the program counter (PC) register, and the contents of these register are stored by the scheduler. The changing of the CPU state is called a context switch, and the context switch time is the time to change the CPU from running one software process to run another software process. Note that the scheduler is just software, but an important part of the kernel or the operating system for the system. Any CPU or micro controller will contain either a kernel or operating system to handle the usage of the CPU, the memory, and any other resources.

44.9 Sequence Diagram

A sequence diagram is used for structuring the time sequence of the messages between modules, like objects, in a system. A sequence diagram can be used to indicate any time requirements when analyzing and designing the software. Figure 44.9 shows the main structure of an application running on a RTU using a sequence diagram, using Unified Modeling Language (UML) (Fowler & Scott 1997), (Douglass 1999). An UML sequence diagram shows any objects, with a specific responsibilities, on the horizontal axis, and the time on the vertical axis. The messages (methods and attributes) between the objects are shown as lines, one line at a time. Any time requirements can be added as comments, or indicated in the structure of the coding. In this case a sampling time of less than 100 ms is used for the DAQ devices.

The application will first setup any DAQ devices and the corresponding sensor and actuator devices. Then a continuos loop is used for reading the sensor values, performing the estimation of new values based on an algorithm, and writing to new values to the actuators. A delay of 100 ms is used to adjust the sampling time and the running requirements of the application.

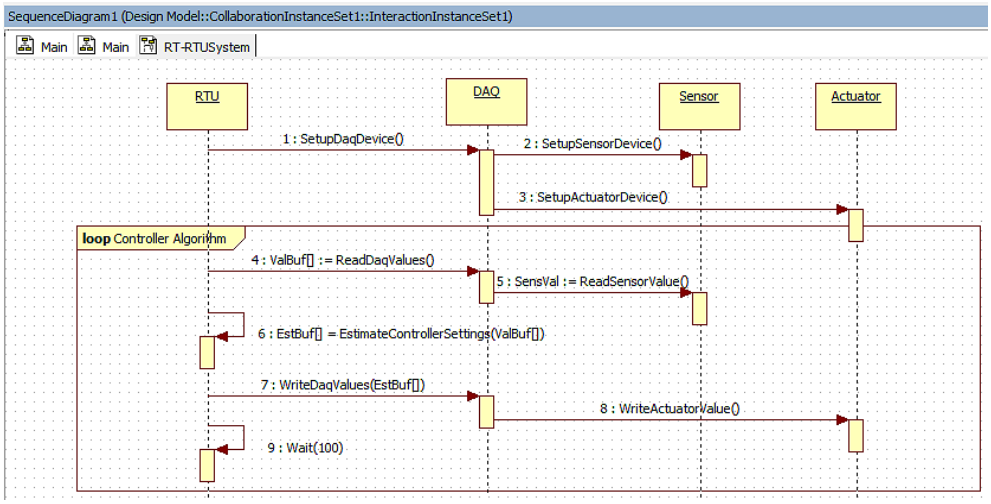


Figure 44.9: An UML sequence diagram showing the data handling in a RTU, based on a controller algorithm.

Any time requirements will always be regarding the process, so the sensor will be used to detect any events in the process and the actuators will be used to influence the process. In this case polling of the input signals is used (mainly analog signals) so the time will be for the whole loop. This indicate how a sequence diagram will indicate the number and type of messages that has to be executed within any time requirement period and can be used in the analysis and design steps of real-time systems.

Chapter 45

Multitasking

The architecture of a system depends on the size of the physical process to be monitored and controlled, any real-time requirements of the physical process, and security. The size of the physical process will define the number of input signals and output signals (sensors and actuators), and the real-time requirements to these signals. Security means among others the requirement for Mean Time To Failure (MTTF), Mean Time To Repair (MTTR), and what if the deadline can not be fulfilled.

The architecture can be a single system, duplication of a single system for redundancy, or distributed systems. Distributed systems can be distribution of the micro controllers (CPU cards), distribution of I/O units, and distribution of resources among others. Network and data buses are important elements in a distributed system as these elements will be the communication elements between the distributed units. A real-time system is often a distributed system in one way or another, especially distributed I/O.

A single system and a distributed system is shown in Figure 45.1.

In distributed system will the real-time functions/services in the operating system be important since tasks and processes can not use common areas in the memory for communication or synchronization.

45.1 Introduction

A real time system normally consists of several functions where each function will have a maximum time requirement. Few systems are that simple that one software application can take care of all these time requirements. The normal way is to divide the system into tasks where each task will be responsible for one or a few of these time requirements. A multitasking system is needed to be controlling these tasks. Each task must be designed in such a way that they will be responsible for fulfilling the requirements only for the time critical events assigned to this task. A service is needed for controlling the execution of these tasks as all tasks should be running simultaneously. However, only one task can be executed by the CPU, or a few tasks if using a multicore CPU. The service for controlling the usage of the CPU is the scheduler, an important part of an operating system.

Depending on the system two types of scheduler exists:

1. Long term scheduler (batch job scheduler, not useful for real-time systems),
2. short term scheduler (CPU scheduler, useful for real-time systems).

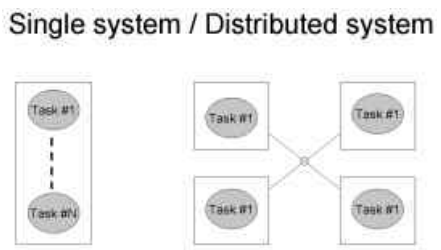


Figure 45.1: A “single” system and a distributed system.

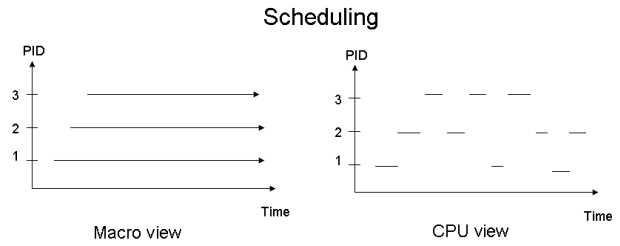


Figure 45.2: The principles of multitasking, macro view and CPU view.

The CPU is an important resource in a real-time system due to the requirements of dead lines and simultaneousness. The simultaneousness is solved by letting the different software contexts be active in short time lags.

Figure 45.2 shows the principle of multitasking, the macro view to the left and the CPU view to the left. The macro view shows that the contexts are running simultaneousness for instance the last 10 minutes, while the CPU view shows the details for instance the last second.

The task, always a software module, will be controlled by the scheduler. Without a scheduler only one task can run, called a single task system. A single task system is an application that run in an endless loop. Any real-time part of such an application must be solved by interrupt service routines (ISR). This solution is used in small systems, uncomplicated systems, or if the real-time behavior is not critical.

The CPU view in Figure 45.2 shows how the scheduler is working. Each task is running for short period of time before the next task is started. A scheduler can be either non-preemptive or preemptive (Bach 1986),(Krishna & Shin 1997):

1. non-preemptive scheduler: a task will use the CPU until it release the CPU to the next task.

   (a) Every context is performing a number of operations before it is calling a function in the scheduler. A context is the current state of a software application running on the computer, the memory range, areas, and the contents of the CPU registers. Every context must consists of a main function, a state machine and require good knowledge when developing the software. The reason for this solution is that every context has the responsibility to run for only a short time and then call the scheduler function. See Figure 45.3 for this type of a scheduler.

2. preemptive scheduler: the CPU can be taken away from the task during execution.

   (a) The normal way of making the scheduler is to use a scheduler task controlled by an interrupt. This interrupt is controlled by a hardware signal from a timer, and the timer is then giving a hardware interrupt to activate the scheduler a fixed time intervals. The interrupt with the highest priority is used so the scheduler task can abort all other contexts. Both Windows and Linux is using 20 ms, but can be adjusted according to the maximum number of tasks that can run simultaneous. Figure 45.4 shows the principle of scheduler controlled by an interrupt. At fixed time intervals will the timer issue an interrupt to the CPU and the current context will be aborted and the scheduler context will be started. The scheduler will save the context of the aborted context (CPU registers, PC and stack), check which task to be started and restore the context of this task. It is important that the running time of the scheduler should be as short as possible as this is overhead in the real-time system.

   (b) An extended way of the interrupt controlled scheduler is using a shorter time intervals only for checking if the running context should be aborted. Normally the running context will run just as long as in number $2a$, but the system can react faster for instance on external events. This solution gives a faster reaction to events, but also more overhead since the scheduler will be active more often.

Remember that the interrupt and/or the scheduler can abort a task at any time, at any instruction in the software task. This is important to have in mind when analyzing and designed the system. Any task can be aborted at any time, and any of the other tasks can be the next active task.

A dispatcher is a software module (often a part of the scheduler) that gives control of the CPU to the selected task by:

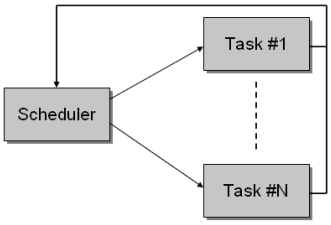


Figure 45.3: A non preemptive scheduler; each task will finish the operations before the scheduler gets the control.

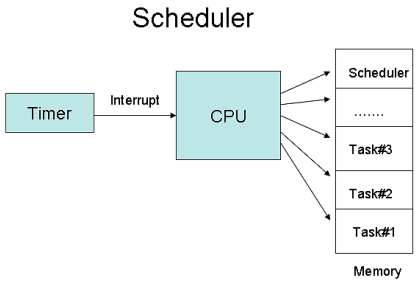


Figure 45.4: A scheduler controlled by interrupt.

1. switching context (load the CPU registers, set the PC and the stack segment),
2. start the task (jump to the code of the Program Counter (PC)).

The scheduling policies are clock (cyclic) or priority (events) driven. Clock driven means that a time interval is used for starting the task (sleep()) and is used when the task or system knows the next time the task can run. Priority driven means that the task is controlled by external asynchronous events, and the priority is used as an indication of important the execution of the task is for the system.

There are generally several different ways of implementing scheduling in a real-time system (Walls 2006):

- a simple processing loop,
- a background processing loop, based on service routines controlled by interrupts,
- a multitasking system using a scheduler.

45.2 States

Using a scheduler gives the tasks different states. The scheduler will decide which task will be running next and must then know which of the tasks that are ready run. Remember that a task can also be waiting for instance an external event and it is waste of time starting a task that is just waiting. Therefore will the context of the tasks have a state, being:

1. Offline; The task is not loaded into memory, no context exist of this task yet,
2. Waiting; The task is waiting for a resource. The scheduler will check the waiting condition every time it is active to check if the state can be changed from waiting to ready,
3. Ready; the task is ready to be running,
4. Running; the task that are running, only one task can be running at a time (on a single core CPU).

The relation between the different states is shown in Figure 45.5. The scheduler has tables or lists containing information of every task in the system, including the state.

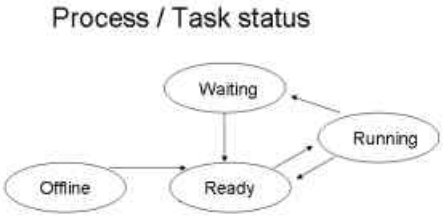


Figure 45.5: The states and relationships of a task or process.

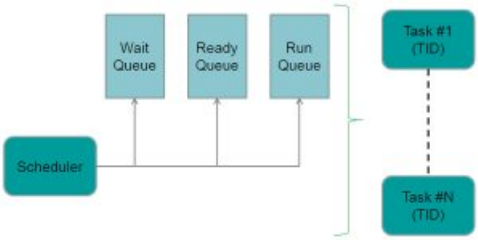


Figure 45.6: The scheduler is maintaining a set of queues (or tables) to keep track of the state for each task, using the task ID (TID).

45.3 Scheduler queues

The scheduler is using a set of queues (or tables) to maintain the states of the tasks, one queue for each of the task states. One queue handles the tasks ready for using the CPU (ready queue), one queue handles the task or tasks using the CPU (running queue), and one queue handles the tasks waiting for a resource (wait queue). The tasks will be assigned an ID number when loading into memory, a task ID (TID or PID). These ID numbers are used as references in the queues. Figure 45.6 shows an overview of the usage of the task state queues. The running queue will be equal to the number of CPU cores, only one task can be in the running state at a specific time for each CPU core.

When the scheduler is activated, the operations will be:

- save the program counter (PC) and registers for the running task,
- loop through the wait queue to check if any of the tasks can be moved to the ready queue. The wait events can be the usage of an resource, software or hardware. Software can be for synchronization or delay to mention some. Any delays will be decremented, and the task will be moved to ready queue if delay is zero,
- loop through the ready queue to find the task with the highest priority to use the CPU,
- load the program counter (PC) and registers for the new task.

45.4 Strategies

When the scheduler is activated, the running context is aborted and saved. The scheduler will check the waiting list (table/queue) and check if any of the waiting tasks can be moved to the ready list or table. Then the scheduler will check the ready list to check which one is running next. The scheduler must use a set of criteria to select the next task for running. The reason for using a set of criteria:

1. CPU utilization: minimize overhead, by keeping the CPU as busy as possible,
2. Throughput: number of processes or tasks completed per unit time,
3. Turnaround time: from creation time to termination time. Time=startTime+Waiting in the ready queue+Executing on the CPU+Doing I/O
4. Response time: from creation time to first output,
5. Fairness: each process or task should have a fair share of the CPU.

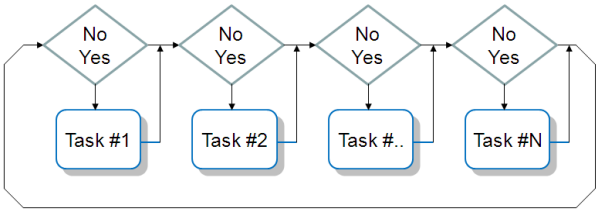


Figure 45.7: The Round Robin scheduling algorithm.

Some of these criteria can be:

1. cooperation; is used when the scheduler is not controlled by the interrupt, but with cooperation between the tasks. Used for non-preemptive schedulers,

2. first-come, first-served (FCFS); each task runs until it blocks or terminates. Used for non-preemptive schedulers,

3. shortest job first (SJF); the task in the ready queue with the shortest running CPU time first. Used for non-preemptive schedulers. The criteria earliest deadline first (EDF), a type of SJF, can be used on preemptive schedulers and both SJF and EDF criterias are suitable for system with a load less than 100%,

4. round robin; using the sequence of the ready list (queue) to decide which task to start next. Used for preemptive schedulers (preemptive version of FCFS), works if all the tasks have the same priority. This is shown in Figure 45.7,

5. priority; the task with the highest priority is stated first. May give problems for tasks with low priority, but can be solved by rising the priority while waiting in the ready list (queue). The priority driven is most used for preemptive schedulers,

6. event driven; priority based, each task is assigned a fixed priority and the task with the highest priority will run. The scheduler is based on an event queue picking the event with the highest (or lowest) number,

7. adaptive; newer operating systems have adaptive partition schedulers that may combine several of these criterias to assist in managing large systems, often with a mixture of hard real-time and non real-time applications.

Priority inversion; If a task with a high priority must wait for a resource used by a task with a lower priority, will the priority of the task be increased until the resource is released. This is the responsibility of the scheduler and used for preemptive schedulers.

Other criteria can also be used, the criteria can be important when selecting a RTOS. In some RTOS it is also possible to select a set of criteria to be used when building or even starting the system.

45.5 Performance

Reliable performance is an important consideration of a real-time system. One important requirement is the interrupt latency. Interrupt is used mainly for event control of the software from the plant. The software can either poll a signal from the plant to check for an event or just wait for the event to happen. Polling means that the software must check the signal regularly wasting CPU power, while interrupt can be used for informing the software when an event is active.

The interrupt latency is the time from the event is activated until the correct software module is running handling the event. The timing is divided into several sections like:

1. interrupt service routine (ISR) latency; time from the event signal is activated until the interrupt service routine is running,

2. ISR; running time of the ISR, this routine is part of the operating system and should be as short as possible,

3. interrupt service function (ISF) latency; time from the ISR has finished until the software module handling the event is running.

Chapter 46

Synchronization

Normally a task has to be synchronized (Bach 1986), (Labrosse 1999), (Krishna & Shin 1997) with another task in some way, and any multitasking or real-time system has to offer a set of services for synchronization like semaphores, mutex, events, interprocess communication and shared memory to mention the most used synchronization services. These synchronization services are used for tasks synchronization and sharing resources to be sure that only one task can change or update a specific resource at a specific time. This document will focus on mutex and semaphores (Barr 2016), as synchronization primitives. Mutex are used for locking shared resources, while semaphores are a signaling mechanism between tasks for sharing resources. Note that in some documentation will mutex be referred to as a special semaphore used for mutual exclusion.

46.1 Mutex

A mutex is the simplest form of synchronization and is used for locking shared resources. A mutex is a binary flag, controlled by the operating system, used for indicating the access to a common resource. The owner of the mutex is the only task that has the right to access the common resource. However, it is important that all tasks are using the mutex service for accessing the common resources. The mutex service has two basic functions and these functions are:

1. request; the scheduler will move the task to the wait queue if the mutex is already "occupied" by another task. The function name can be either request() or wait()[1],
2. release; releases a mutex, a non-blocking behavior. The next time the scheduler is running it will move the "blocked" tasks from the wait queue to the ready queue. Note that the next task that will get access to the mutex is not based on FIFO (First In First Out), but will be based on the scheduler strategies [2].

A mutex is a binary flag and will only have the values 0 and 1. Any attempt to allocate the resource using the mutex will decrement the value of the mutex, and release of the resource will increment the value of the mutex. A mutex value of zero means that any task trying to allocate the mutex has to wait for the release of the resource, while a positive value of the mutex means that the task can allocate the resource.

The real-time system must support a set of mutex services, minimum two operations as indicated, request and release. The name of these operations can be *request()* or *wait(),* and *release()*, but these names may vary from real-time system to real-time system.

Release() will always increment the value, while *request()* or *wait()* will decrement the value, but only if the value is 1. The *wait()* function will contain both checking of the value and decrement the value if 1. The mutex service will guarantee that the scheduler will not abort these operating service operations.

The create() function makes a mutex, store the reference as part of the operation system services, that can be available for all tasks in the system. Note that there must be at least one mutex for each of the common resources that can be used by any task. Normally one of the tasks creates the mutex and then this task and other tasks can use this mutex. All common resources should normally be protected

[1] In C# (.NET): use *WaitOne()* for requesting access to a mutex.
[2] In C# (.NET): use *ReleaseMutex()* for releasing a mutex.

by mutex. A task that is doing a wait() for a resource already in use, will be put in the wait queue until the resource is ready for usage. Therefor it is important for the programmer to always use release whenever the resource is not in use.

The of the mutex (note that this **must** be part of the operating system services, why? - Hint: preemptive scheduler):

```
// Define a mutex for this resource
Mutex  mutRes1 = 1 ;

// Release the semaphore after using the resource
release(Mutex mutRes1)
{
   mutRes1 = 1 ;
}

// Request the mutex, before using the resource
wait(Mutex mutRes1)
{
   if (mutRes1 != 1)
   {
       system_oswait();
       // Call to the scheduler for moving the task to the wait queue
       // and start a new process. Will rerun from this state after
       // the wait state, and the resource will be available.
       // Do NOT use a wait loop !!!
   }
   mutRes1 = 0 ;
   return OK ;
}
```

The mutex is used for protecting shared resources so that only one task can change or update the shared resource at a specific time, for control of data consistency. Note that a mutex must be an operating system service as several tasks must use/share the same mutex. Mutex is used only for a specific task, as a signaling service between two or several tasks a semaphore should be used. If only one instance of an application can run at one time, a mutex can be used to verify that only one instance is running. The application can check with the operating system if the mutex exist. If the mutex is not existing, the application will create the mutex, request the mutex and continue. Just remember to release the mutex when ending the application. If the mutex exists, the application must try to request the mutex, or check if the mutex is already requested to make a message to the user and then end.

46.2 Semaphore

A semaphore is the simplest form of synchronization and is used for signaling between tasks. The main function for a semaphore is for signaling from one task to another task, and one task can either send a signal to another task or wait for a signal from another task. The mutex is used for reserving a resource by a specific task, and used by the same task, and released by the same task when finished. The semaphore is used for signaling to other tasks, not for the same task. A semaphore can be a binary variable being only 0 or 1 or an unsigned integer (char/short/integer) being 0 or a positive number (like 256 or 65535). The task signaling will increase the value, while the task waiting for the signaling will decrease the value. The task waiting for signaling will be put in the wait queue if the semaphore value is zero (0), or below a maximum value defined as the signaling value. A semaphore is also a good way for communication between tasks and interrupt service routine (ISR) as signaling as a sending service is a non-blocking behavior and safe to be used in a ISR. However, an ISR should never wait for signaling as this is a blocking behavior.

Semaphores can be used to control access to shared resources, signal the occurrence of an event to other tasks and allow two or several tasks to synchronize their activities (Labrosse 1999). Just like the mutex will the semaphore service have two basic functions, these functions are:

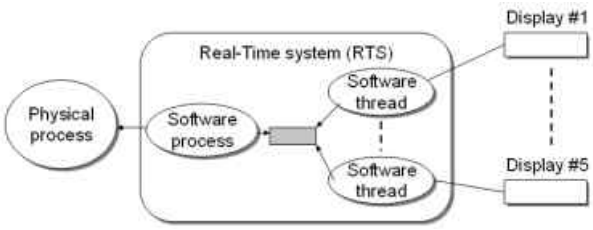


Figure 46.1: The RTS system with the software process, software threads and the displays.

1. post; the tasks will update the semaphore (increment) whenever a specific event occur. This is a non-blocking behavior and will not involve the scheduler. Operating systems have different names on this service [3],

2. pending; a task is waiting for a semaphore to be updated (incremented). The scheduler will move the tasks to the wait queue if the value of semaphore is zero (0) [4].

46.3 Syncronization example

An example of synchronization for a plant can be:

A real-time system has several displays, located at five different locations, but shall be updated based on the same parameter in the physical process. These displays are updated from a text buffer of 6 characters showing the parameter as a text string. The system contains a process for reading the values from the process, calculating the parameter, and converting the parameter to the text string in the text buffer. The process will then use five threads for displaying the value, one thread for each display. At this stage just look at processes and threads as two different type of tasks (software), the difference will be explained later in this document. All the threads will be the same software application, but each thread will handle only one display. This way it will be easy to scale the system if more displays are wanted, just start one thread for each display. See Figure 46.1.

The main process will run with the same priority as the threads, and using the text buffer as a shared resource. Lets assume the current value of the text buffer is 0.9876 and the new calculated value from the process is 1.0123. The operation for converting the floating value (of binary form) to the text string requires some CPU time updating one and one character in the text buffer. Let us assume that this floating point to text conversion and updating the text buffer is time consuming and only one character will be updated in each time slot of the system. The "simultaneous" requirement of the system gives that the process only have time to update the first character in the buffer before the threads will run. The value in the text buffer will now be 1.9876 and the threads will now write this value on the display and end the execution. The process will then run again and update the next character in the text buffer, now containing 1.0876. Then the threads run again and update the displays. The process will have time to update only one character in the text buffer between each display process. This way the operators will see the value as unstable.

This kind of multitasking is using the scheduler to switch between the context for the process and the threads, the problem is the synchronization of the text buffer. This challenge will exists for all type of multitasking operating systems like Windows and Linux, not only real-time operating systems. The results of displays will be as shown in Table 46.1, showing the values at different update stages and the correct value.

How to solve this example by using synchronization?

The first challenge is the text buffer, the text buffer is a shared resource. A shared resource should be protected by a mutex. The simplest solution will be to use just one mutex and let the task or thread that should use the text buffer request the mutex first. The challenge with this solution will be that each process or thread has to wait for the mutex to be available, and since the process is the only tasks updating or changing the text buffer, this should be the only task using the mutex.

The second solution can be to combine mutex and semaphores, use the mutex to request the text buffer when changing the information, and the semaphore when signaling to the threads that the displays can be updated. In this solution will the mutex actually not be necessary, but it is a good programming

[3] In C# (.NET): use *Release()* for signaling using a semaphore.
[4] In C# (.NET): use *WaitOne()* for wating for signaling of a semaphore.

Table 46.1: The displayed values on the different values when the parameter value is updated from 0.9876 to 1.0123

Update	Display value	Correct value
1	1.9876	1.0123
2	1.0876	1.0123
3	1.0176	1.0123
4	1.0126	1.0123
5	1.0123	1.0123

practice always to use a mutex when updating a shared resource. When the text buffer is updated, semaphores will be used to signaling to the threads for updating the displays. This solution will require one mutex for the text buffer, and one semaphore for each of the threads.

The third solution can be to use one thread and one semaphore, but let the semaphore be a number equal to the number of displays. The process must then set both the mutex and semaphore to zero when updating the text buffer, and set the semaphore to the number of threads when releasing both the mutex and the semaphores. This way will all the threads be signalled that they can read the text buffer and update the displays.

State machines are an useful architecture for real-time tasks as these tasks often have a set of predefined states. Our process has three defined states, read the values from the process, calculate the value to be displayed, and convert the value to a text string and update the text buffer. A wait state is also very useful, in this case for the sampling time. Note that by using the wait state, the priority can also be increased as the wait state will release the cpu to tasks with a lower priority. The pseudo coding for such a process can be:

```
Method Main()
{
    while (state != QUIT_STATE)
    {
        switch (state)
        {
            case INIT:
                SetupDaqDevice();
                LoadConfigurationData();
                state = READ_VALUES;
                break;
            case READ_VALUES:
                dVal = ReadDaqDevice();
                state = CALC_VALUES;
                break;
            case CALC_VALUES:
                dispDVal = CalcDisplay(dVal);
                state = TEXT_BUF;
                break;
            case TEXT_BUF:
                dispText = EstimateDisplayText(dsipDVal);
                // Request the semaphore as well to stop any thread from updating
                RequestThreadSemaphore(textBufSemaphore);
                // Request the mutex before updating the text buffer
                RequestMutex(textBufMutex);
                UpdateTextBuffer(dispText);
                ReleaseMutex(textBufMutex);
                ReleaseThreadSemaphore(textBufSemaphore);
                state = WAIT_DISPLAY;
            case WAIT_DISPLAY:
                // release some CPU time to update the display faster
                sleep(100);
                // check for any special system states, default is READ_VALUES
```

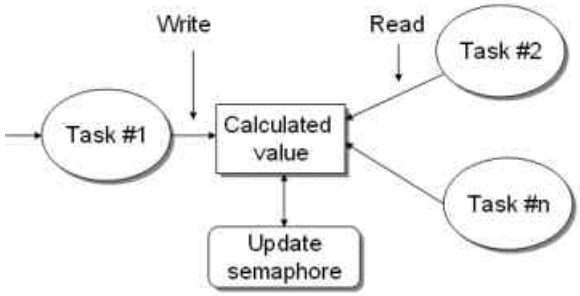


Figure 46.2: Using syncronization events to read a updated value. Task#1 is updating the value, and task #2 to task#n is only reading the value.

```
                state = CheckSystemOperation();
                break;
            case QUIT_STATE:
                // Any operations before quiting
                break;
            default:
                state = INIT;
                break;
        }
    }
}
```

Note that the software can be solved in many different ways depending on the services of the operating system and cpu power. In real-time system, always focus on optimizing the usage of the cpu.

Another example is writing some text on the printer. Normally the operating system has a printer service handling the printer devices. If no printer services is available, this service must be included. Having only one printer device, only a mutex can be used. Having several printer devices, a combination of one semaphore (equals to the number of printer devices) and one mutex for each printer device must be used. This way any task will be put in the wait if no printer device is available, and the correct task is using the correct printer device if several printer devices are available in the system.

46.4 Events

An event is used if several tasks(Krishna & Shin 1997) are waiting for a semaphore and all of these tasks is going to do an operation where a semaphore is not needed.

One example can be reading of a value that have been calculated or estimated by a task. The task writing the new value will use a semaphore to update the value, but after the value has been updated several tasks can read the value without the usage of a semaphore. All tasks wanting to read the value will use a wait() function and be put in the wait queue of the scheduler. When the writing task has changed the value and updated the synchronization, all waiting tasks will get an event and can read the new value. See Figure 46.2 where task #1 is updating the value using an event semaphore and task #2 to task #n is only receiving an event when they can read the value. This means that the tasks are moved from the wait queue to the ready queue of the scheduler.

Real-time systems supporting events normally have the operations set(), clear() and wait() for events.

46.5 Interprocess communication (IPC)

In a multitasking system it may be a challenge when communication of data or messages is necessary between tasks. Normally there will be a requirement that the tasks must be able to communicate with each other. Mutex, semaphores and events can be used, but they are only the primitive operations on a single computer (node). However, mutex, semaphores and events are used as building blocks by the more complex Interprocess communication (IPC) operations, as synchronization, protection and sharing

are the goals of IPC. IPC can be locally on one node or globally between several nodes. Note that mutex and semaphores are only working on a single node.

46.5.1 Events

Events a “small" message that can be sent between the tasks to “signal” a change of state.

46.5.2 Pipes

A pipe is a FIFO[5] buffer created in a common data area of the real-time system for used as read/write buffer for the task (Bach 1986). A pipe can then be used as a simple communication channel between two tasks as one task can write data to the buffer and the other task can read data from the buffer. If a task is reading from a empty queue, the task will be moved to the wait queue until the other task is writing data into the buffer.

46.5.3 Shared memory

Shared memory is a common data area where the tasks can read and write to random locations in the buffer (Bach 1986). Using pipes, a FIFO buffer is used, using shared memory a random location in the buffer can be read and write. Pipes are often implemented by using shared memory and a set of services to handle the FIFO buffer. Remember to use some sort of synchronization if several tasks can change the same information in shared memory; i.e. writing to the same locations.

46.5.4 Message queue

Message queues are set of pipes used as post boxes, all created in the common data area of the real-time system (Bach 1986). These message queues can then be used for sending messages and data between the tasks in the system. Synchronization can be done by sending commands to other tasks and waiting for the answers. If a message queue is empty and a task is trying to read, the task will be moved to the wait queue of the scheduler. If the write queue of another task is full, the task is also put in the wait queue until the message queue can be written to. Message queues are combining shared memory, mutex and semaphores.

Synchronization can also be done by a master task sending commands to other (sub) tasks with information about what each (sub) task should do. These (sub) tasks will perform each command and then go back to the message queue to wait for the next command.

46.6 Communication Protocols

As time is an important parameter in RTOS, knowledge about the maximum time of receiving or transmitting data is important to be able to calculate the deadlines for the functions in the RTOS. Protocols are used for receiving and transmitting data between two or more devices in a system, often distributed systems, and protocols will be important for the RTOS. The real time requirement is normally higher on the control level and field level. Today will protocols that are using the Ethernet standard and following the IEEE 802.3 be important to be able to use standard infrastructure building blocks.

In distributed systems there will be two important properties regard communication and these properties are: a) direct cross traffic between the nodes in the network, b) network topology. Direct cross traffic means that no master is necessary, the messages should be sent as broadcast mode messages and the information will be available for all the nodes in the network at the same time. This will give a more effective communication, less traffic load in the network, and more capacity available on the master. Mixture of network topologies gives a system that easily can be upgraded and/or extended.

46.6.1 Token Ring

The token ring protocol is using a token which define the owner of the communication media. The node having the token will be the master of the network. The master node must send the token to the next node when finishing the network operation.

[5] FIFO: First In First Out

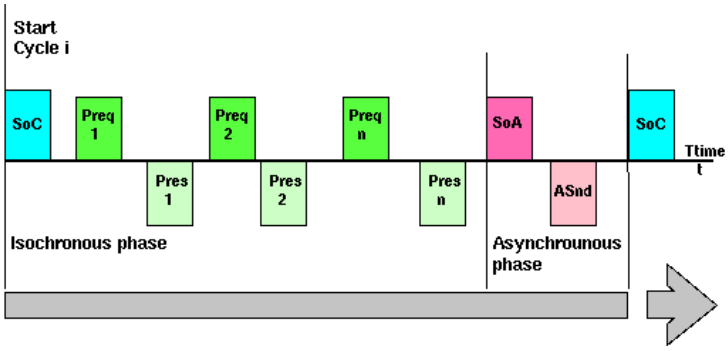


Figure 46.3: The basic cycle of the POWERLINK communication (www.wikipedia.org 2010).

46.6.2 CSMA/CD

The Carrier Sense Multiple Access / Collision Detect (CSMA/CD) protocol is defining that every node can be the master whenever the communication media is needed. If several nodes are trying to use the media at the same time, a collision will occur, and both nodes will stay of the media for a random time and retry to use the media.

This protocol may not be used in real-time systems as the protocol delay is not defined. The protocol is not deterministic. However, the maximum time and hard/soft type will decide if this protocol can be used. Different industrial Ethernet protocols have been developed, some based on hardware extensions and other on only software extensions. Hardware extensions mean that standard Ethernet devices for CSMA/CD can not be used, like the Profinet IRT (Isochronous Real-Time). Other protocols like Ethernet POWERLINK is using only software and is compatible with existing network devices like routers, gateways, etc. These protocols are using a combination of polling and time slices where only one node can use the communication media in specific time slices.

Ethernet POWERLINK

Ethernet POWERLINK is a deterministic real-time protocol for standard Ethernet communication systems. It is an open protocol managed by the Ethernet POWERLINK Standardization Group (EPSG) (www.wikipedia.org 2010). It is based on the standard Ethernet, but extent the protocol with a mixed Polling- and Time slicing mechanism. This gives:

- transfer of time-critical data in short isochronic cycles with configurable response time,
- time-synchronisation of all nodes in the network with high precision (μs),
- transmission of less time critical data in a reserved asynchronous channel.

The communication media is controlled by a Managing Node (MN) and the overall cycle time depends on the amount of isochronous data, asynchronous data, and the number of nodes to be polled during each cycle.

The basic cycle consists of the following phases (See also Figure 46.3):

1. Start Phase: The MN is sending out a synchronization message to all nodes, called “Start of Cycle” (SoC),
2. Isochronous Phase: The MN calls each node to transfer time-critical data (PollReq. and PollRes. messages). Since all nodes are listening to all data during this phase, the communication system provides a producer-consumer relationship. Time slots are used for each addressed node,
3. Asynchronous Phase: The MN grants the usage of the communication media to one particular node for sending non real-time data (Soa and AsyncData messages). Standard IP-based protocols and addressing can be used during this phase.

The quality of the real-time behavior depends on the precision of the basic cycle time. The duration of the isochronous and the asynchronous phase can be configured. POWERLINK is extended the Data Link Layer with time slots for each node in the isochronous phase giving that only a single has access

to the communication media. The number of slaves (CN) can vary from cycle to cycle as all the CN do not need to be polled in every cycle. CN with lower priority can share the same time slot giving a multiplexing of the lower priority slaves.

Chapter 47

Resources

A system with requirements to simultaneousness must also have some requirements to access the resources in the system. A resource can be hardware or software devices, a device that two or more tasks must be able to share. A resource can not be used by several tasks at the same time, so the resource has to be reserved by the task before being able to use the resource. The task must also release the resource when finished using the resource.

The reservation and releasing of resources is often difficult, especially to debug when it is not working. Normal problems are:

1. Forget to reserve a resource before using it,
2. Forget to release the resource after using it.

A good advice is to reserve only one resource at the time and release the resource before reserving another resource. Deadlock is a situation that can arise if several tasks are trying to reserve a set of resources.

47.1 Deadlock

Deadlock arise if task #1 has reserved resource A and need resource B to finished the operation, and task #2 has reserved resource B and need resource A to finish the operation. These tasks will then never be able to finished their operations.

```
// Task #1
resourceA.wait() ;  // wait for resource A
resourceB.wait() ;  // wait for resource B

// Task #2
resourceB.wait() ;  // wait for resource B
resourceA.wait() ;  // wait for resource A
```

Example 13 *In an alarm system should all important alarms be saved on both disk and paper. Several tasks are used with alarm checking and these tasks are then going to log the alarm on the disk as will be a resource and write on the printer as will be another resource. The scheduler will run task #1 who detects a alarm situation and is reserving the disk. Then the scheduler will run task#2 who also detects a alarm situation, task#2 is written by another programmer, and is reserving the printer first. Task#1 is then activating again writing the alarm message to the disk and tries to reserve the printer. As the printer is already reserved, task#1 will be put onto the wait queue and task#2 will be started. Task#2 will write the message on the printer and try to reserve the disk for logging the alarm message. Task#2 will also be put in the wait queue because the disk resource is already reserved. And both tasks will be in the wait queue for ever ...*

Remark 14 *Also note that the watchdog task will not detect this type of errors, and will not reset the system!*

Table 47.1: The task for the alarm system using wait() and release() functions.

Step	Task#1	Task#2
N	..	..
N+1	wait(disk)	wait(printer)
N+2	wait(printer)	wait(disk)
N+3	log alarm	print alarm
N+4	print alarm	log alarm
N+5	release(printer)	release(disk)
N+6	release(disk)	release(printer)
N+7	..	..

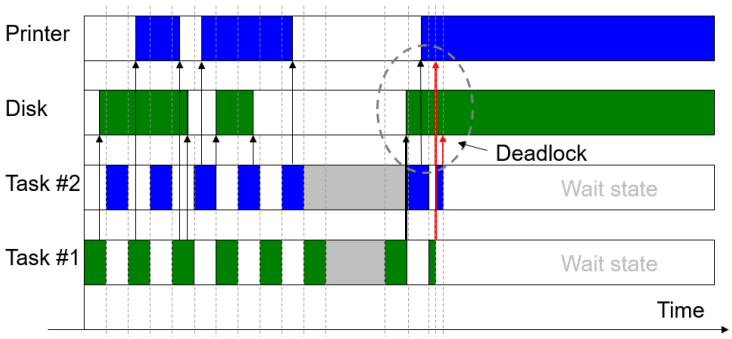


Figure 47.1: A scenario without deadlock is shown to the left, and a scenario with deadlock is shown to the right. The tasks are reserving the disk and printer for logging the alarm states.

Table 47.1 shows the usage of wait() and release() functions for the example.

Two scenarios for the tasks in table 47.1 are shown in Figure 47.1. The scenario to the left is without the deadlock situation where each task is reserving the disk and printer resources for logging the alarm states. In this scenario will the tasks not request the disk and printer resources at exactly the same time and a task is releasing the resources before the next task is requesting the same resource. The scenario to the right shows a deadlock situation because both tasks are requesting the resources at the same time, and will not release any resource before both resources are reserved. This scenario is possible since the scheduler can abort any task at any time, and in this case is task #1 aborted at instruction $N + 1$ and task #2 will run the same instruction $(N + 1)$ before task #1 will continue.

How to avoid the deadlock situation? Reserve and release only one resource at the time? This can the solution in this case, but in some cases both resources are needed at the same time. In such cases should the resources be assigned priority numbers, and the reserve sequence should be in the priority order. This assignment is normally the responsibility of the project leader.

Deadlock can also arise in systems using messages for synchronization between the tasks. The system has 2 tasks that should be synchronized by the x and y messages, shown with the code below. The *receive()* method will wait until a message is received before returning the control back to the task.

```
// Task #1
receive() ;  // Wait for next message
send(x) ;    // Send message x

// Task #2
receive() ;  // Wait for the next message
send(y) ;    // send the message y
```

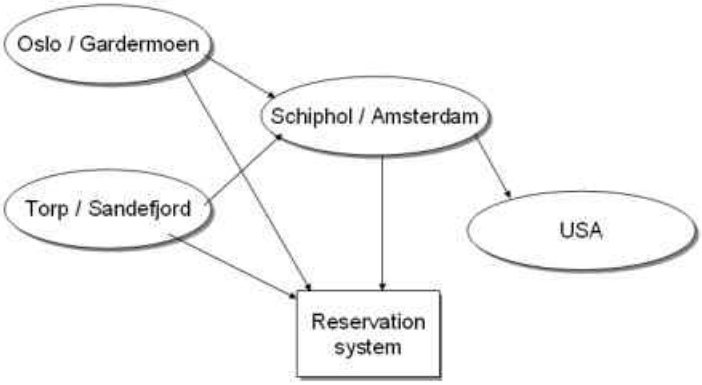


Figure 47.2: Reservation of plane seats from different airports.

This example will give a deadlock because both systems will wait for a new message. How can deadlock be avoided in this example[1]?

Note that a deadlock scenario can only arise if minimum two resources are reserved at the same time, not when reserving only one resource at a time. The normal software "bug" when dealing with resources is to forget to release a resource, looking just like a deadlock situation.

47.2 Critical region

Often is it necessary to protect a sequence of commands, in some sort of high level programming language, to avoid that the scheduler is aborting this sequence. This sequence is called a critical region and will never be aborted by the scheduler or another interrupt function, and must be a service of the real-time operating system.

Example 15 *An example of a critical region is a reservation system for an airplane. Two passengers are checking in, one at Oslo/Gardermoen and one at Torp/Sandefjord, both at exactly the same time. Both passengers are going to USA using the same plane from Schiphol/Amsterdam. See Figure 47.2 for the airports and reservation system.*

Both passengers wants seat 9A and without a critical region the sequence may be:

1. The system in Oslo/Gardermoen is calling the reservation system for checking of seat 9A,
2. The system in Torp/Sandefjord is calling the reservation system for checking of seat 9A,
3. The scheduler on the reservation system starts the task from Oslo/Gardermoen,
4. The reservation system checks that seat 9A is free,
5. The scheduler on the reservation system is switching to the task for Torp/Sandefjord,
6. The reservation system checks that seat 9A is free,
7. The scheduler on the reservation system is switching to the task for Oslo/Gardermoen,
8. The reservation system informs the system in Oslo/Gardermoen that seat 9A is taken,
9. The scheduler on the reservation system is switching to the task for Torp/Sandefjord,
10. The reservation system informs the system in Torp/Sandefjord that seat 9A is taken.

Using a critical region on the free/taken sequence, the reservation will now be:

1. The system in Oslo/Gardermoen is calling the reservation system for checking of seat 9A,

[1] HINT: Change the sequence of the rx/tx functions in the tasks, have different sequences in the two tasks. One task should be the master of the communication, the other tasks should be slaves.

2. The system in Torp/Sandefjord is calling the reservation system for checking of seat 9A,
3. The scheduler on the reservation system starts the task from Oslo/Gardermoen,
4. The task for Oslo/Gardermoen is entering a critical region,
5. The reservation system checks that seat 9A is free,
6. The reservation system informs the system in Oslo/Gardermoen that seat 9A is taken,
7. The task for Oslo/Gardermoen is leaving the critical region,
8. The scheduler on the reservation system is switching to the task for Torp/Sandefjord,
9. The reservation system informs the system in Torp/Sandefjord that seat 9A is busy.

As the critical region will disable the scheduler and other interrupts, it must be as short as possible. Some questions:

1. Never ever wait or try to reserve a resource inside a critical region. Why?
2. Can you implement a critical region using one semaphore?

Chapter 48

Software modules

When analyzing and designing the real-time system, the system will be divided into tasks. Designing applications for a real-time systems are always challenging. One way to decrease the complexity of these application is to use a task-oriented design and divide a project into different modules (or tasks). Each module is then responsible for a specific sub parts of the application. With such a system it is important to be able to specify that some modules (or tasks) are more important than others. The applications can then be divided into several tasks, and each task can be developed as a software application or program, but need to intercommunicate with each other.

After compiling the software, an image of the systems memory will be made and this image must be transferred to the targets disk, EPROM[1] or FLASH[2] memory depending on the type of real-time system. Figure 48.1 shows one way of developing software for a real-time system, the software is developed using a "standard" development system and must be uploaded to the real-time system for finale testing.

The software is uploaded to the real-time system and started using some type of boot loader[3]. The software will be located in memory with a code area (read-only), data area (both read-only and read-write), a working area[4], and a stack area. The stack is located from the top of the memory and is used for temporary storage. The contents of the memory is shown in Figure 48.2, loaded from a storage device to the left and loaded from a EPROM/FLASH system to the right.

The running software is called a context being:

1. location of the application code, application data and stack in the address range,
2. the contents of all the CPU registers,
3. the content of the program counter (PC) (pointing to the next program instruction to be performed).

Every program task is developed as a standalone application that will be implemented in the real-time system as a task or a process.

A task or a process consists of the following segments in memory:

- code (text); the memory used for the compiled code of the application,
- stack; the memory used for automatic variables within functions. Data is stored using the Last In First Out (LIFO) method. This means that memory is allocated and deallocated only at the top of stack memory. Stack is mainly used for temporary storage of information.

[1] EPROM: Erasable Programmable read Only Memory; Can also be ROM, PROM and EEPROM (Electrical EPROM).
[2] FLASH: Similar to EEPROM, but erasing can only be done in blocks or the entire chip.
[3] Boot-loader: A software device copying the software from the storage device into the working memory of the system.
[4] The working area is often called the heap.

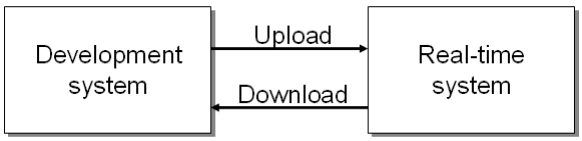


Figure 48.1: Developing software for real-time systems.

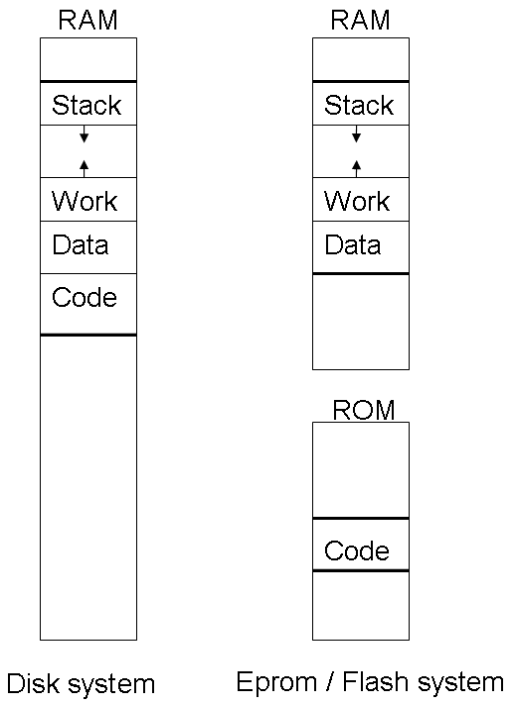


Figure 48.2: The context of a running software task.

- data segments;
  - initialized data; data area included in the task or process at startup time (included in the code),
  - uninitialized data; data area included in the task or process at startup time, but not used (normally different type of buffers/tables),
  - heap; the memory used for dynamic memory allocation. Blocks of memory can be allocated and freed in any order. Heavy use of the heap will give a fragmented memory area.

48.1 Instruction time

An instruction cycle is the time period during which a computer processes a machine language instruction from its memory or the sequence of actions that the central processing unit (CPU) performs to execute each machine code instruction in a program.

The name fetch-and-execute cycle is commonly used for the instruction cycle. The instruction cycle will vary from instruction to instruction and a CPU normally has large set of instruction. The operation for an instruction cycle will be:

1. Fetching the instruction; read the contents of the memory location of the program counter (PC),
2. Decode the instruction; the instruction register will hold the instruction,
3. Execute the instruction; a control unit is instructed by the instruction register how this instruction shall be executed, often using the arithmetic logic unit (ALU),
4. Store the results; any results will be stored in memory.

In some system it can be necessary to disassemble the application code to get a list of the CPU instructions. The sequence of the CPU instructions can be used to calculate the exact execution time of the application or part of the application.

An example of the instruction set for a CPU is shown in Figure 48.3. This Figure shows a few instructions for the 80386 CPU giving the instruction codes and number of clock cycles.

intel

386™ MICROPROCESSOR

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary

INSTRUCTION	FORMAT				CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
GENERAL DATA TRANSFER								
MOV = Move:								
Register to Register/Memory	1000100w	mod reg r/m			2/2	2/2	b	h
Register/Memory to Register	1000101w	mod reg r/m			2/4	2/4	b	h
Immediate to Register/Memory	1100011w	mod000 r/m	immediate data		2/2	2/2	b	h
Immediate to Register (short form)	1011 w reg	immediate data			2	2		
Memory to Accumulator (short form)	1010000w	full displacement			4	4	b	h
Accumulator to Memory (short form)	1010001w	full displacement			2	2	b	h
Register Memory to Segment Register	10001110	mod sreg3 r/m			2/5	18/19	b	h, i, j
Segment Register to Register/Memory	10001100	mod sreg3 r/m			2/2	2/2	b	h
MOVSX = Move With Sign Extension								
Register From Register/Memory	00001111	1011111w	mod reg r/m		3/6	3/6	b	h
MOVZX = Move With Zero Extension								
Register From Register/Memory	00001111	1011011w	mod reg r/m		3/6	3/6	b	h
PUSH = Push:								
Register/Memory	11111111	mod110 r/m			5	5	b	h
Register (short form)	01010 reg				2	2	b	h
Segment Register (ES, CS, SS or DS)	000sreg2110				2	2	b	h
Segment Register (FS or GS)	00001111	10sreg3000			2	2	b	h
Immediate	011010s0	immediate data			2	2	b	h
PUSHA = Push All	01100000				18	18	b	h
POP = Pop								
Register/Memory	10001111	mod000 r/m			5	5	b	h
Register (short form)	01011 reg				4	4	b	h
Segment Register (ES, SS or DS)	000sreg2111				7	21	b	h, i, j
Segment Register (FS or GS)	00001111	10sreg3001			7	21	b	h, i, j
POPA = Pop All	01100001				24	24	b	h
XCHG = Exchange								
Register/Memory With Register	1000011w	mod reg r/m			3/5	3/5	b, f	f, h
Register With Accumulator (short form)	10010 reg				3	3		
IN = Input from:				**Clk Count Virtual 8086 Mode**				
Fixed Port	1110010w	port number		†26	12	6*/26**		m
Variable Port	1110110w			†27	13	7*/27**		m
OUT = Output to:								
Fixed Port	1110011w	port number		†24	10	4*/24**		m
Variable Port	1110111w			†25	11	5*/25**		m
LEA = Load EA to Register	10001101	mod reg r/m			2	2		

* If CPL ≤ IOPL ** If CPL > IOPL

115

Figure 48.3: The first page of the instruction for 80386 CPU, the "father" of the CPU used in Windows computer today (Int 1988).

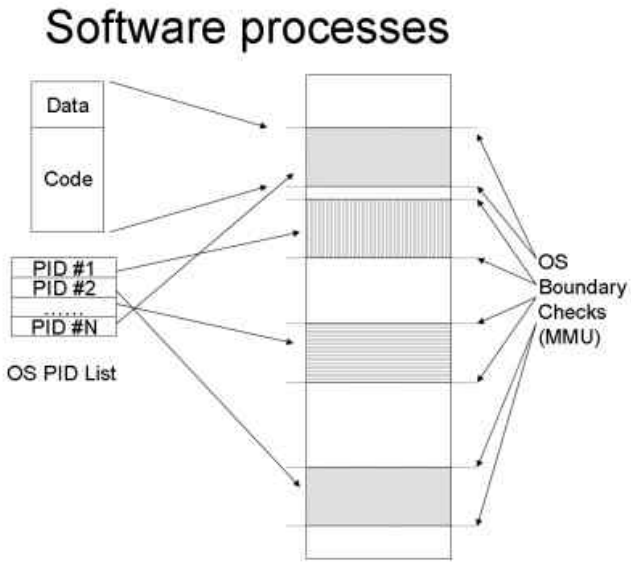


Figure 48.4: The memory layout and the PID list in a real-time operating system.

48.2 Software application

A running application is a context, and can be either a process, a task or a thread.

48.2.1 Process

Processes are used for program tasks in bigger real-time systems and require more memory and hardware devices. The most important hardware unit is the MMU[5], a unit converting the addresses from a virtual (logical) memory area to the physical memory area for the system. Normally real-time systems using 32 and 64 bits microprocessors (CPUs) are using MMU, and software applications are implemented as processes. The reason for the MMU is a more effective utilization of the memory and ensure that no processes can access memory area outside it's own memory area. The advantage is when a software application contains an error; the application will not be able to influence other applications in the system. The disadvantage is that a software application will not be able to access data from other applications. How shall these processes be able to exchange data or be synchronized? The solution is to use operating system services for data exchange and synchronization.

The operating system must maintain the MMU table and the process ID (PID) lists, shown in Figure 48.4. The operating system also contains services for synchronizations between the processes, and in modern operating system threads can also be used.

48.2.2 Thread

A thread, also called a "light weighted" process, must be started by a process, will have a new code memory area (with it's own program counter), but will have access to the same data memory area as the process. A process can start a number of threads, all of them will have separate code memory area but all will share the same data memory area.

Figure 48.5 shows a process containing a code memory area and a data memory area. The process has also started a number of threads, having separate code memory area but all of them having a common data memory area.

When the process and the threads have the same data area will synchronization be important. Normally will a thread be started to perform a small part of the program task and often will this small part of the program task take some time.

Threads can be very useful for I/O functions, because:

1. error checks can be done in the thread and will not influence the execution of the process,

[5] MMU: Memory Management Unit.

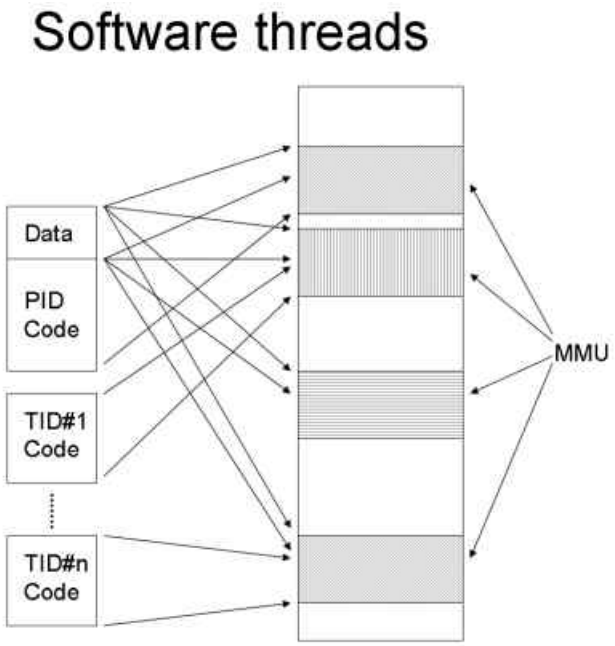


Figure 48.5: A single process with a number of threads.

2. scaling is easy, just start a new thread if more I/O is necessary,
3. waiting for an event, only the thread will wait for the event, the rest of the application will run as normal.

The drawback when using threads is synchronization, in a real-time system the process and the threads must be executed "simultaneous" and Example 46.3 shows the problem of synchronization.

48.2.3 Task

Smaller real-time systems using $4, 8$, or 16 bits microprocessors (CPU) do not have a MMU and there will not be any mapping between virtual (logical) and physical memory. Using no MMU, there is not possible to have the boundary checks between the program tasks, and these program tasks are called tasks. A task can always access the whole memory area as shown in Figure 48.6, showing the task ID list and the memory where each task can access the whole memory area.

The advantages of using tasks (and no MMU):

1. faster real-time operating system as no conversion between logical and physical address is necessary,
2. all tasks can access the whole memory area, easy and fast to exchange data between the tasks,
3. easy to synchronize the tasks..

The disadvantages of using tasks (and no MMU):

1. Difficult to trace a software problem as a task can destroy for the other tasks (which one is first?),
2. All access to common data has to be synchronized.

From time to time there is a mixing of the term *process* and *task* in the literature, the basis is that a process is a software application on a mainframe computer while a task is a software application on a microcomputer. A common description of a job in a real-time system is a task, so this can be the reason for this mixture. It is however important to understand the technical differences of a process and a task.

A process:

1. is using a logical or virtual memory area (needs a MMU),

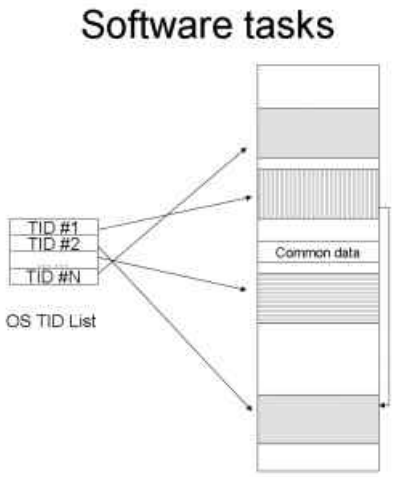


Figure 48.6: A real-time system with 4 tasks with common data area and no check of memory addresses.

2. must use operating system services for communication and synchronization with other processes,
3. with a software error will not destroy the data for other processes in the system.

A task:

1. is using a physical memory area,
2. can communicating directly with other tasks using memory buffers, but should of course use OS functions for synchronization,
3. can destroy the data for other tasks in the system if an error.

48.3 Core and Multicore

A CPU can contain one core or several cores. A core is a single CPU with the registers, program counter (PC), memory and I/O controller, and the arithmetic logic unit. The core is the CPU resource and a multicore means that the system has several CPU resources. The cores will use the same memory and I/O devices and range, meaning that the task/processes can share resource the same way as a single core system. A multicore system will have several run queues meaning that the scheduler has to decide the next running task for all the cores.

Figure 48.7 shows how two independent loops can utilize the two cores in a multicore processor.

48.4 Input monitoring

A real-time system must be preemptive, meaning that a software context can be aborted by another software context. This is solved by interrupt, either hardware interrupt or software interrupt. Hardware interrupt is also very useful when monitoring an event in the physical process. This monitoring can be done in two different ways:

1. interrupt,
2. polling.

Using interrupt, a real-time system is utilizing the preemptive support and switch the software context to read the event and then switch back to the interrupted context. See the top section of Figure 48.8.

Using polling, the real-time system must have a process/task checking the event all the time wasting a lot of unnecessary CPU power. See the bottom section of Figure 48.8.

Interrupt should be used for digital signals (state changes), while polling should be used for analog values. The analog values vary all the time so a new value should be read at every polling time. The polling time can be the sampling rate of the monitoring system.

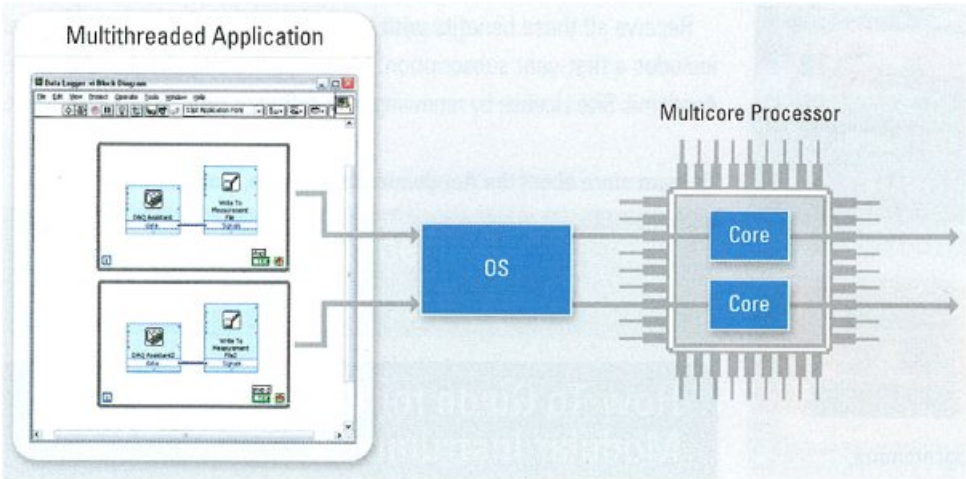


Figure 48.7: A Lab VIEW application with two independent loops running on two diferent cores on the multicore processor (www.ni.com: jan-10).

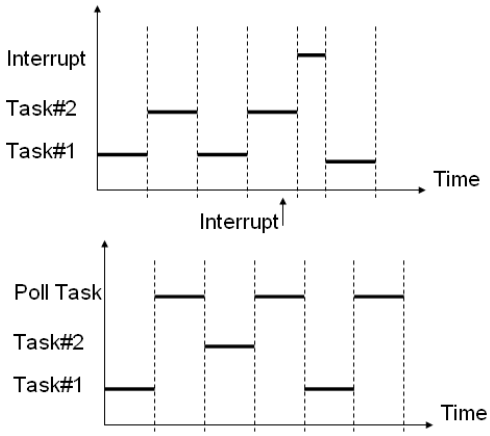


Figure 48.8: Interrupt control in the upper part and polling in the lower part. Notice the vast of CPU time in the lower part.

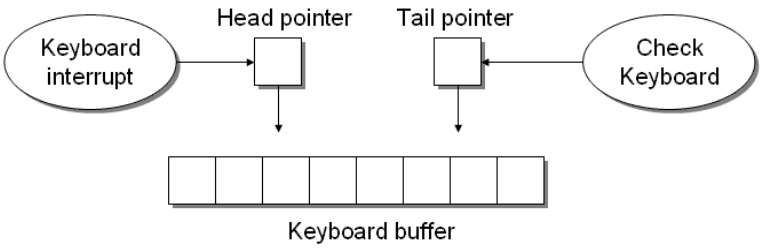


Figure 48.9: Using an interrupt task for reading the keyboard characters.

48.4.1 Interrupt operation

An interrupt will always abort the running task, and start the interrupt task. A system has several interrupts and the interrupt with the highest priority is served first. The scheduler is also connected to an interrupt, if a preemptive scheduler. This interrupt is controlled by a timer and will have the highest priority. An interrupt task should be as short as possible because it will change the timing requirements of the running (and aborted) tasks. Any task can be aborted at any time!

The steps for the interrupt operation:

1. An interrupt is triggered, hardware or software,
2. Depending on the interrupt priority and the current states of the other interrupts, the system will respond to this interrupt,
3. A context switch is performed, saving the program counter and registers of the running task, and loading the program counter and registers of the interrupt tasks. This is the interrupt latency timing,
4. The interrupt tasks is performing a set of operation, very important that the interrupt task should be as short as possible. Normally the interrupt task is just turning a flag on or off or releasing a semaphore,
5. A context switch is performed, saving the program counter and registers of the interrupt task, and restoring the program counter and register of the aborted task.

48.4.2 Interrupt example

A keyboard is a good example of device using interrupt. When a key is pressed, the current software context is interrupted, the software context for the keyboard is started, the key pressed is read and saved in memory, and the interrupted software context is started. See Figure 48.9.

Using polling, the keyboard must be polled fast enough not to loose any keys, meaning that the keyboard has to polled at least every 500 ms all the time.

48.5 Priority Inversion

A microprocessor often has a lot of interrupts, both hardware and software. These interrupts have different priorities and an interrupted task with lower priority will be aborted by an interrupted task with higher priority. The right priority of the interrupts must then be assigned during the analysis and design of the real-time system. A system with multiple tasks running in parallel may in some cases give a low priority task that can cause a high priority task to be halted. In systems where different tasks need exclusive access to a resource, priority inversion can occur. A task with a high priority will be given a low priority for a specific period of time. A low priority task gets to run and starts to use a resource with exclusive access. If a higher priority task needs the same resource it halts until the low priority task has released the resource. This would effectively give the high priority task a priority level beneath the low priority task.

This can be a problem for the task with the high priority because the time requirements is "harder" than the tasks with lower priority. This is shown in Figure 48.10 where the high priority task, *TaskH*, is given a priority lower than the low priority task, *TaskL*, in the period marked with priority inversion.

There are different solutions to this problem, the most common ones are:

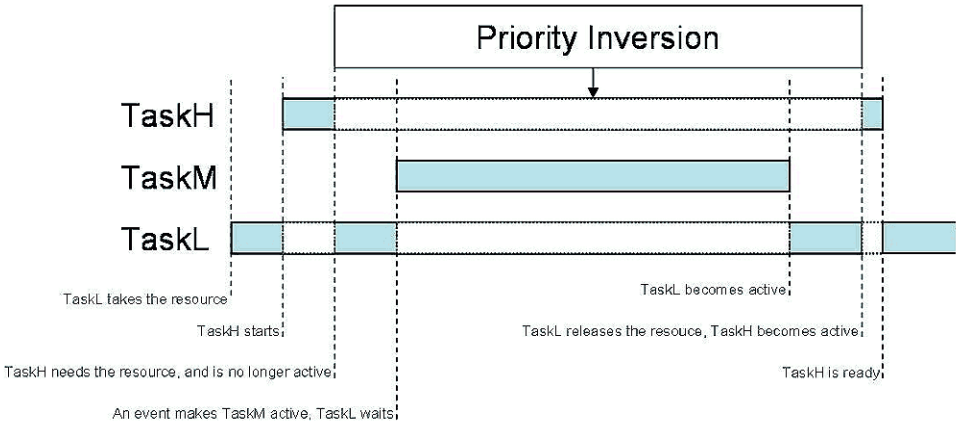


Figure 48.10: Priority inversion where a low priority task *TaskL* has exclusive access to a resource and prevent the high priority *TaskH* from running (www.iar.com: Nov-08).

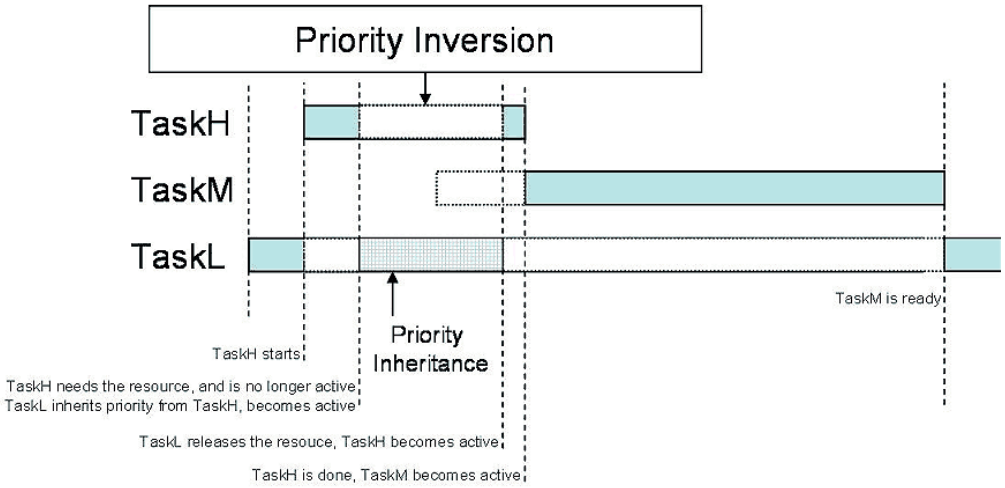


Figure 48.11: Priority inversion with inheritance where a low priority task *TaskL* has exclusive access to a resource, but inherit the high priority from the *TaskH* (www.iar.com: Nov-08).

1. Priority inheritance; let the low priority task inherit the priority from the tasks waiting for the resource,
2. Disabling task switching to protect critical sections; turn off the switching when using the resources.

The priority inheritance will let a low priority task inherit the highest priority from any tasks waiting for the resource. This is shown in Figure 48.11 where the low priority task TaskL inherit the priority from the high priority task TaskH. The low priority task TaskL will finished using the resource and let the high priority task TaskH finish before the medium priority task TaskM is started.

48.6 Watchdog

A monitoring function of the RTOS system. A hardware device, often a counter that will reset the CPU after a specific amount of time. A task with the lowest priority is used for resetting the hardware device and the system have to be designed that the watchdog task will not reset the system when everything is working OK. If a software error occurs (or system error), the watchdog will not be reset and the system will be restarted. it is however important to understand that the watchdog will only work if your software is designed wrong with endless loops or crashing, but will not with problems like deadlock. Why?

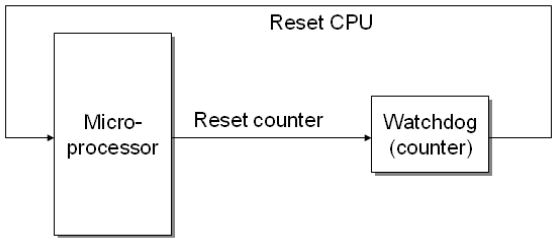


Figure 48.12: The watchdog is a hardware device, a counter, using the hardware reset signal for reset of the CPU.

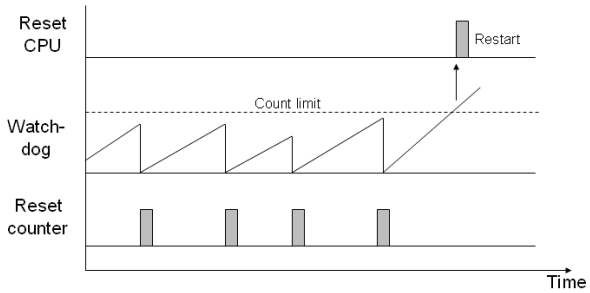


Figure 48.13: The watchdog is counting the number of pulses and a limit of pulses when the output will be activated. The reset signal will force the counter to start count from 0.

Figure 48.12 shows the principle for the watchdog. The watchdog is a hardware device, a counter, connected to an oscillator or a crystal. The counter will count a number of pulses and the output will be high when this number of pulses is reach. The counter output is connected to the hardware reset input of the CPU resetting the CPU, meaning restarting the whole real-time system.

The reset task in the system has the responsibility to reset the counter, to start from 0, so it will never reset the CPU. Resetting the CPU is an error condition and should normally not happen. Figure 48.13 shows the functions of the watchdog. The watchdog will count the number of pulses and the output will be activated when a specific number pulses are counted. The reset task should reset the watchdog counter before reaching this specific number of pulses, forcing the counter to start from 0.

Chapter 49

Real-time specifications

Process control system (PCS) are often controlled by multitasking system, and may have real-time requirements. Real-time is about deadlines, and missing a deadline.

49.1 Descriptions

Definition 16 *A real-time system is a system that react at the right time, in a predictable way, on an unexpected external event*[1].

Definition 17 *A real-time system is a system where the calculations not only depend on a logical correct execution, but also of the time when the result is available*[2].

The requirements for a system to be a real-time system:

1. deadline; the real-time system must detect any changes of the states for the physical process within a specific time. The system is defined as failing if these deadlines are not kept,
2. simultaneousness; the system must be able to handle several changes in the physical process at the same time and all of these changes must be detected within the deadline. This requirement demands a parallelism in the system, the solution may be using multitasking and/or a distributed system,
3. resources; the real-time system will have a limited set of resources like CPU, memory, disc, I/O devices so these resources must be shared, synchronization variables in the software and so on. Important that the requirement for simultaneousness is kept even if there is a limited set of resource.

A real-time system can be classified as a "hard" or "soft" system(Gallmeister 1995) having the following consequences of missing a deadline:

Type	**"Hard" real-time system**	**"Soft" real-time system**
Deadline	the system will fail if not kept	lower performance can be accepted if not kept
Delay	will NEVER be accepted	can be accepted, but at a higher "cost"
Data	is useless if given at a wrong time	
Rule	"shall not miss a deadline"	"should not miss a deadline",
Examples	Control systems for fighters	IP-phone
	Antilock Braking System (ABS) for cars	Youtube videos

Note that some may classified real-time systems as "hard", "firm" and "soft", however 'firm" is still a "soft" type of classification.

[1] Martin Timmerman, Belgium.
[2] Hans Christian Lønstad, Data Respons ASA, Norway.

49.2 Properties

Important properties for a multitasking system, and involving the real-time requirements:

- Event; changes in the physical process involving an event into the real-time system. An event is something that comes from outside the system or part of the system. An event can be a message sent to one or more of the tasks in the real-time system,
- Task / Process / Thread: to be able to have simultaneousness in the real-time system should the problem be divided into smaller parts depending on the deadline for these parts. Each of these parts can then be designed as a software module like a task, process or thread,
- Multitasking; to be able to have simultaneousness in the real-time system as all the tasks, processes, or threads must run in parallel. Multitasking is the process of scheduling and switching the CPU between several tasks that will share the CPU,
- Scheduler; The service in the real-time system responsible for the multitasking, from a set of rules to select the next task to run[3],
- Preemption; Important events in the physical process may require a fast decision, faster then the scheduler service can give. A real-time system must therefore be able to be preemptive; to interrupt the running task and start a new task. But note that the deadline must still be held for the interrupted task,
- Interrupt; An event in the real-time system that is used for temporary stopping the running task and start a new task,
- Interrupt latency; The time from an interrupt occurs until the new task (interrupt service routine)(ISR) is running,
- Interrupt service routine (ISR); A specific task, named ISR, started by an interrupt. The ISR must be as short as possible and is normally used only as a synchronization point between the hardware and software, or between different software tasks.
- Priority; The tasks will be assigned different priority levels when analyzing and designing the system, depending on the importance of the task. The scheduler will always try to run the tasks with the highest priorities first,
- Context; the CPU state of the software application loaded into primary memory with code, data, and stack areas, and the contents of CPU registers,
- Watchdog; A hardware device monitoring the whole system, and restarts the system if the software fails.

[3] Both Windows and Linux are multitasking operating systems containing a scheduler.

Chapter 50

Programming

50.1 Introduction

When programming of a real-time system it is important to have focus on the dead lines, simultaneousness, and the usage of resources. The software system has to be as fast as possible, never waste any CPU time, and minimize the usage of the resources.

If the program is going to wait for an event, **never ever use a waiting loop**, the solution is to tell the scheduler that the task has finished and will wait for a period of time. The normal operation is *sleep(time)* where the waiting time will be time in milliseconds (VentureCom 1993). Whenever a task is performing this *sleep()* function, the scheduler is told to place the task in the waiting queue and start the next task in the ready queue. Every time the scheduler is activated, it will check the tasks in the waiting queue and the time will be decremented until 0. At 0 will the task be moved to the ready queue again. A trick is often to use the sleep(0) which will start the next task in the ready queue and place the aborted task at the back of the ready queue.

The program flow in a real-time application will be depended on external events as distinct from office applications using a sequential program flow controlled by user and data. A real-time application responds to external events from sensors among others, while office applications responds to user interactions and available data.

Some advises when designing real-time systems:

1. Always think synchronization when using resources,
2. Always think synchronization when changing global data,
3. Always use a sleep() function if the application has to wait (or can wait). Can even use $sleep(0)$ for a short waiting period,
4. State machines are useful in real-time systems.

50.2 Memory allocation

The memory allocation is a service of the operation system and is also treated as a resource in a real-time system. The normal operation is to allocate the memory when needed and release the memory after the usage. This should be avoided in a real-time system because the memory will be fragmented and the software will not be able to allocate more memory. This is special for $24x7$ systems as the memory allocation service (table) will never be reset. The solution for the real-time system can be to allocate all the memory at startup time, and just reuse this memory while running.

The memory allocation will be used more in an object-oriented based system as every *new()* method is allocating memory for the class. Some programming languages like C++ are using delete() method to release memory. For programming languages like C# and Java is a garbage collector running in the background releasing memory for classes with no references. The garbage collector will use CPU time making problems with the deadlines, and the memory will be fragmented giving problems for the allocation function.

50.3 Posix.4

Portable Operating System Interface (Posix) is a standard from IEEE and the meaning with Posix is to have portable source code for the applications (Gallmeister 1995). The Posix standard defines a set of system functions that can be used to make a more portable source code, and Posix.4 defines a set of real-time functions. Using a Posix.4 compatible real-time system can be useful when developing real-time systems.

50.4 Multitasking examples

50.4.1 Threading

An application developed in C# shows the usage of threads, wait and simultaneousness. The main program starts 3 threads and all 4 tasks (process and threads) are running simultaneously (multitasking), waiting some time (sleep() method, entering the wait state) and printing a message on the screen for each loop. The application consists of 2 classes, one Thread class (*ThreadClass*) and the main program (*ThreadSys*).

Both the main program and the threads are using the sleep() function to let the scheduler abort the context, move the context to the waiting queue and start the next context in the run queue. The application is using different waiting times for the main program and the threads and every context is writing the status on the screen. The output of the application is shown in Figure 50.1, showing the state of the main program and the threads.

The application starts in the ThreadSys class, in the main function. The function of the main application is:

1. Write the text string " Start of main program " on the screen,
2. Create the first thread class, set the name Thread#1 and the wait time to 200 ms,
3. Create the second thread class, set the name Thread#2 and the wait time to 300 ms,
4. Create the third thread class, set the name Thread#3 and the wait time to 400 ms,
5. Then entering a loop counting to 30. In each loop, print a dot on the screen and wait 100 ms,
6. End the main program by writing " End of main program ".

What will be the duration of the main program?

Duration=_____

The functions of the thread applications will be:

1. Set the name and wait time of the thread,
2. Create the threading function of the thread class (the run function),
3. Start the threading function (the run function),
4. Write the text string "Starting Thread#n ",
5. Loop for 5 times with waiting the wait time for the thread and writing the thread name and the loop counter.
6. End the thread by writing "Ending Thread#n ".

What will be the duration of each thread?

Thread#1=_____; Thread#2=_____; Thread#3=_____

```
using System;
using System.Threading;
//Example in using sleep() and Threads in C#
namespace ThreadSys
{
    /// <summary>
    /// Threadclass
    /// </summary>
    class ThreadClass
    {
        int         loopCnt, loopDelay;
        Thread      cThread;

        public ThreadClass(string name, int delay)
        {
            loopCnt = 0 ;
            loopDelay = delay ;
            cThread = new Thread(new ThreadStart(this.run)) ;
            cThread.Name = name ;
            cThread.Start() ;
        }
        // The main function in the ThreadClass
        void run()
        {
             Console.WriteLine(" Starting " + cThread.Name) ;
             do
             {
                  loopCnt++ ;
                  Thread.Sleep(loopDelay) ;
                  Console.Write(" ") ;
                  Console.Write(cThread.Name) ;
                  Console.Write(": ") ;
                  Console.WriteLine("Loop=" + loopCnt) ;
             } while (loopCnt < 5) ;
             // Ending of the thread
             Console.WriteLine(" Ending " + cThread.Name) ;
         }
     }
     // The application
     class ThreadSys
     {
        /// <summary>
        /// Start of the main program
        /// </summary>
        static void Main(string[] args)
        {
           Console.WriteLine(" Start of main program ") ;
           // Making 3 threads ..
           ThreadClass ct1 = new ThreadClass("Thread#1", 200) ;
           ThreadClass ct2 = new ThreadClass("Thread#2", 300) ;
           ThreadClass ct3 = new ThreadClass("Thread#3", 400) ;
           // Wait while the threads are running ...
           for (int cnt = 0; cnt < 30; cnt++)
           {
              Console.Write(".") ;
              Thread.Sleep(100) ;
           }
```

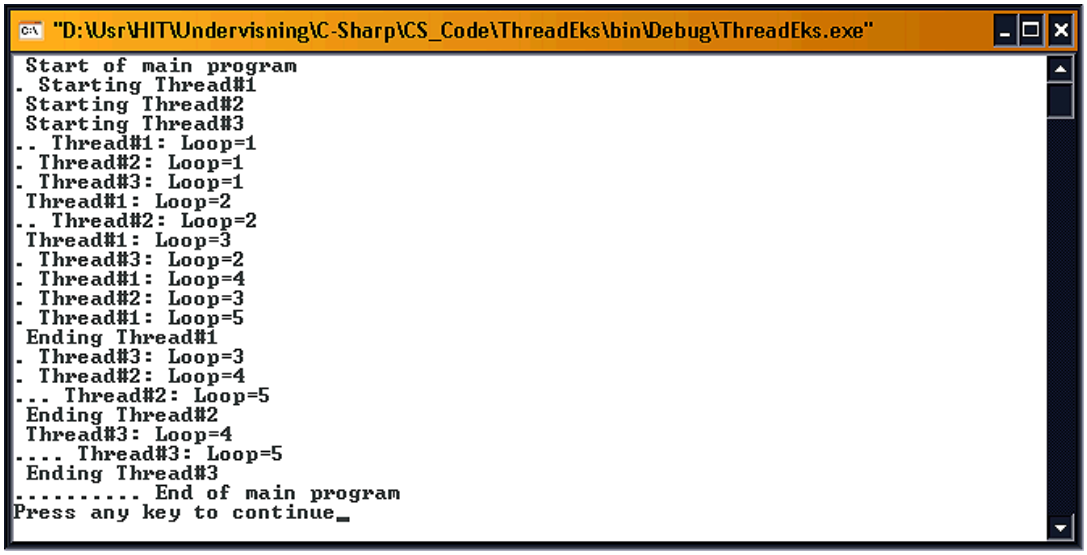


Figure 50.1: The output from the C# application consisting of the main program and 3 threads.

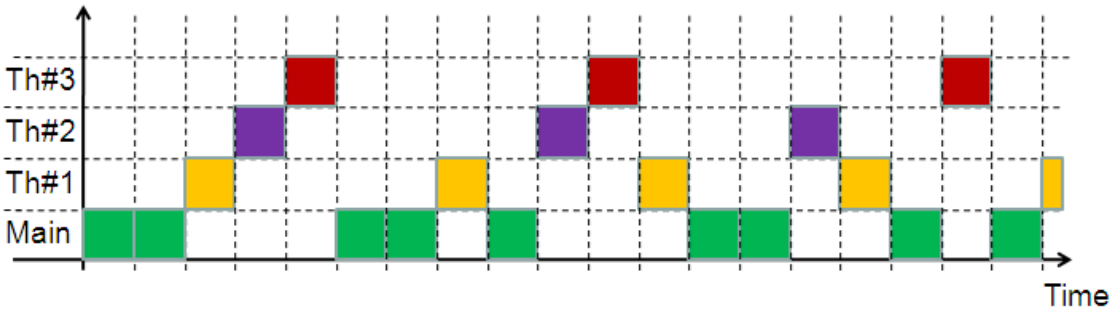


Figure 50.2: The execution of the main process and the 3 threads.

```
            // End of main program
            Console.WriteLine(" End of main program ") ;
        }
    }
}
//
//////////////////////////// EOC ////////////////////////////////////
```

The timing of the main process and the threads shown in Figure 50.1 is shown in Figure 50.2.

50.4.2 Traffic light controller

A traffic light controller can be programmed as a singletask application but is also a good example on how to use a multitasking system. Assume a junction as an intersection of two roads with traffic lights at each end of the junction, one road in the north-south (NS) direction and one road in the east-west (EW) direction. In addition there will be pedestrian crossings of the roads so extra buttons and lights will be included for the pedestrians. An overview of the junction is shown in Figure 50.3.

Assume you want to make a single software application for this junction which means that you need to handle the timing of the red, yellow and green lights for the roads, the timing of the lights for the pedestrians, and a continuos checking of the buttons in the same application. This will end up in a complex structure most probably using several state machines to keep track of the different states in the application. In stead a multitasking system can be used by separating the functions into four separate processes or tasks, one process handling the road lights, two processes handling the pedestrian lights (and possibly buzzers), and one process handling the pedestrian buttons. The reason for two processes for

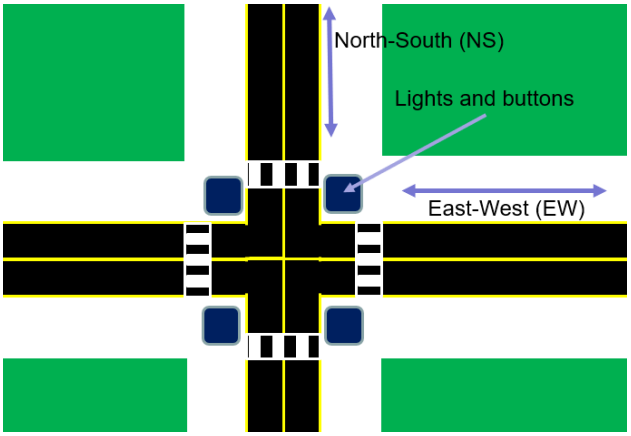


Figure 50.3: The intersection with the two roads, the four pedestrian crossings and the four light poles with the road lights, pedestrian lights and buttons.

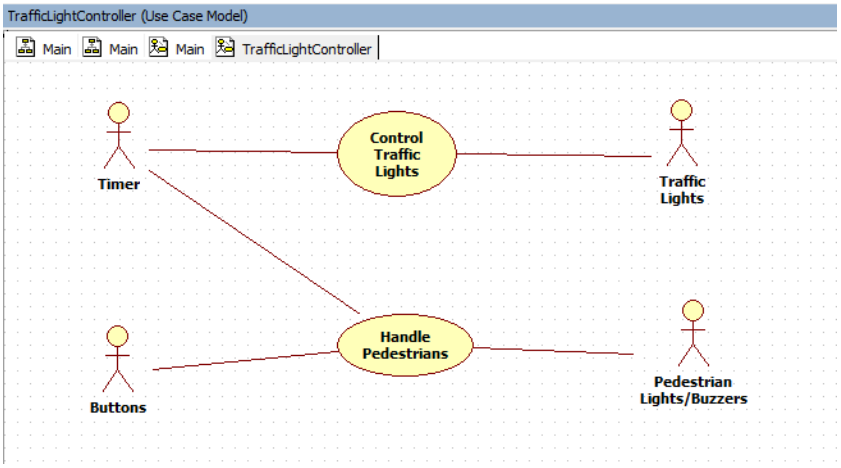


Figure 50.4: The UML use case diagram for the traffic light controller, showing the two main functions for the software.

the pedestrian lights is one process to handle the north-south (NS) direction, and one process to handle the east-west (EW) direction. The communication between the processes (tasks) will be solved by using a set of semaphores. The first task when developing software is making a use case diagram, an UML use case diagram is shown in Figure 50.4. The use case diagram shows the two main function of this software, to control the traffic lights for the crossing roads, and handle any request from the pedestrians for crossing the roads.

The multitasking solution will be four processes, one process handling the traffic lights for the roads, two processes for handling the pedestrian lights, and one process for handling the pedestrian buttons. The process solution can be modelled in a deployment diagram, and an UML deployment diagram for this software solution is shown in Figure 50.5. The deployment model shows the running processes of the application, the main process ProgramStart and the four processes handling the lights and buttons. Semaphores are used for communication between the processes as the processes states will depend on each others.

Two semaphores, one for each of the NS and EW direction buttons are used as communication between the button process and the crossing lights process to indicate that pedestrian is waiting for crossing any road, and the crossing lights process will use semaphores for communicating with the pedestrian lights processes to let any pedestrian to cross at the correct traffic light states. The application consists of five classes, where three classes are implemented as stand alone processes. The class controlling the pedestrian lights are instantiated as two objects, one object for each of the NS and EW directions. The class diagram is shown as an UML class diagram in Figure 50.6, showing the relationship between the five classes in the application.

Note that the object diagram will consists of six objects, what is the difference between the class

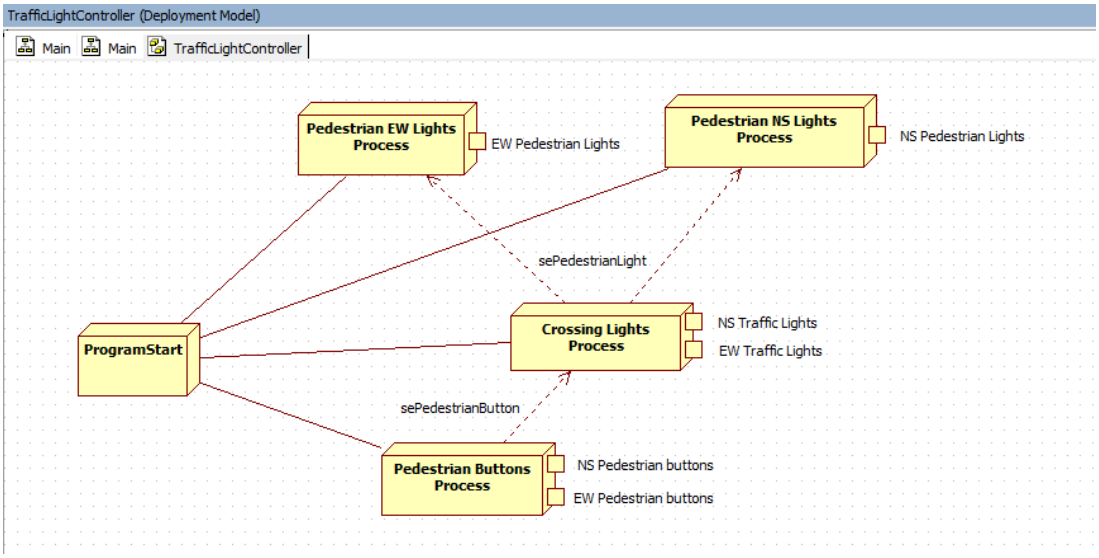


Figure 50.5: The UML deployment diagram for the multitasking traffic light controller with one process for the road trffic lights, two prosesses for the pedestrian lights, and one process for the pedestrian buttons. Semaphores are used for communication between these processes.

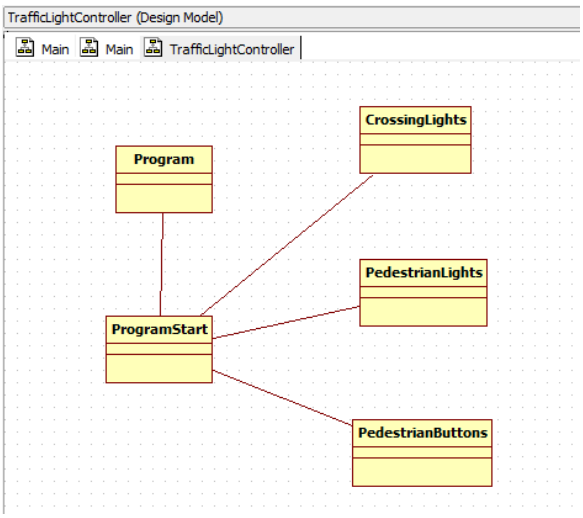


Figure 50.6: The class diagram for the traffic light controller.

diagram and the object diagram for this application? As explained will the PedestrianLights class be instantiated as two objects making six objects from the five classes.

The traffic light controller is developed as a console application is C# where the IDE is creating a default version with the Program class and the Main method. Since the Main method must be a static method a new object, ProgramStart, is made to start the application. The ProgramStart object will make four new object where each object will start a thread running as a separate process in the operating system environment (Windows or Linux). The classes and objects are:

Class name	Object name	Description
CrossingLights	rCrossingLights	Handle the traffic lights in both NS and EW directions
PedestrianLights	rPedestrianNSLights	Handle the pedestrian lights in the NS direction
PedestrianLights	rPedestrianEWLights	Handle the pedestrian lights in the EW direction
PedestrianButton	rPedestrianButton	Handle the pedestrian buttons, checking every 100 ms

The coding standard used in developing this software states that an object should use the class name with a "r" first to indicate an object reference to the class. Snippets of the source code is shown below. All comments in the source code are removed to save space. First the source code for Program and ProgramStart classes (The Main() method is always the starting point):

```
using System;
using System.Threading;

namespace MultitaskingTraficLights
{
    class Program
    {
        static void Main(string[] args)
        {
            ProgramStart rProgramStart;

            rProgramStart = new ProgramStart();
        }
    }
    class ProgramStart
    {
        CrossingLights rCrossingLights;
        PedestrianLights rPedestrianNSLights, rPedestrianEWLights;
        PedestrianButton rPedestrianButton;
        bool bRunFlag;

        public ProgramStart()
        {
            rCrossingLights = new CrossingLights();
            rPedestrianNSLights = new PedestrianLights("NS");
            rPedestrianEWLights = new PedestrianLights("EW");
            rPedestrianButton = new PedestrianButton();

            bRunFlag = true;
            while (bRunFlag == true)
            {
                if ((bRunFlag = rPedestrianButton.getUserStatus()) == false)
                {
                    rCrossingLights.stopProcess();
                    rPedestrianNSLights.stopProcess();
                    rPedestrianEWLights.stopProcess();
                }
                Thread.Sleep(100);
            }
```

```
    }
}
```

The constructor will be called when creating an object so the statement "new ProgramStart()" will run the method ProgramStart() when creating the ProgramStart object in the Main() method. The ProgramStart() constructor will create four new objects and starting the processes in these objects by the constructors of these objects. The source code for CrossingLights class, creating the four semaphores and the CrossingLigthsProcess thread that will run as a separate process in the system. Creating the semaphores are adding a name since the same semaphores are used in the other classes so the operating system will handle the semaphores with the same name as the same semaphore. This class also contain a state machine handling the control of the traffic lights, and the state machine will check the semaphores to delay the red light states in case a pedestrian is waiting for crossing the road.

```
class CrossingLights
{
    Thread tCrossingLights;
    bool bRunFlag;
    Semaphore sePedestrianNSLight = new Semaphore(0, 1, "sePedestrianNSLight");
    Semaphore sePedestrianEWLight = new Semaphore(0, 1, "sePedestrianEWLight");
    Semaphore sePedestrianNSButton = new Semaphore(0, 1, "sePedestrianNSButton");
    Semaphore sePedestrianEWButton = new Semaphore(0, 1, "sePedestrianEWButton");

    public CrossingLights()
    {
        bRunFlag = true;
        tCrossingLights = new Thread(new ThreadStart(this.CrossingLightsProcess));
        tCrossingLights.Start();
    }
    void CrossingLightsProcess()
    {
        int iCrossState, iSleep, iLoop;

        iCrossState = 0;
        iLoop = 0;
        while (bRunFlag == true)
        {
            switch (iCrossState)
            {
                case 0:
                    Console.WriteLine("  Cross#NS:Red - Cross#EW:Green");
                    if (sePedestrianEWButton.WaitOne(TimeSpan.FromSeconds(0)) == false)
                        iSleep = 3000;
                    else
                    {
                        sePedestrianEWLight.Release();
                        iSleep = 6000;
                    }
                    break;
                case 1:
                    Console.WriteLine("  Cross#NS:Yellow - Cross#EW:Yellow");
                    iSleep = 1000;
                    break;
                case 2:
                    Console.WriteLine("  Cross#NS:Green - Cross#EW:Red");
                    if (sePedestrianNSButton.WaitOne(TimeSpan.FromSeconds(0)) == false)
                        iSleep = 3000;
```

```
                    else
                    {
                        sePedestrianNSLight.Release();
                        iSleep = 6000;
                    }
                    break;
                case 3:
                    Console.WriteLine("  Cross#NS:Yellow - Cross#EW:Yellow");
                    iSleep = 1000;
                    iCrossState = -1;
                    iLoop++;
                    break;
                default:
                    iCrossState = -1;
                    iSleep = 1000;
                    break;
            }
            iCrossState++;
            Thread.Sleep(iSleep);
            if (iCrossState == 0 && iLoop > 3)
            {
                Console.Clear();
                iLoop = 0;
            }
        }
    }
    public void stopProcess()
    {
        bRunFlag = false;
        sePedestrianNSLight.Release();
        sePedestrianEWLight.Release();
    }
}
```

The class handling the lights for the pedestrian. This class will be instantiated as two objects, see the ProgramStart class. The PedestrianLightsProcess() method is the thread of this object and will be controlled by the semaphore, and will only run when the semaphore is released. The WaitOne() method will wait until the semaphore is released by the ControllingLIghtsProcess() method.

```
class PedestrianLights
{
    Thread tPedestrianLights;
    Semaphore sePedestrianLight;
    bool bRunFlag;
    string sLightDir;

    public PedestrianLights(string sDir)
    {
        string seName;

        seName = "sePedestrian" + sDir + "Light";
        bRunFlag = true;
        sLightDir = sDir;
        sePedestrianLight = new Semaphore(0, 1, seName);
        tPedestrianLights = new Thread(new ThreadStart(this.PedestrianLightsProcess));
        tPedestrianLights.Start();
    }
```

```
        void PedestrianLightsProcess()
        {

            while (bRunFlag == true)
            {
                sePedestrianLight.WaitOne();
                Console.WriteLine("  ** Pedestrian " + sLightDir + " Green");
                Thread.Sleep(5000);
                Console.WriteLine("  ** Pedestrian " + sLightDir +  " Red");
            }
        }
        public void stopProcess()
        {
            bRunFlag = false;
        }
    }
```

The class handling the buttons for the pedestrians, with the process checking any button response every 100 ms. The process will release the button semaphore for sending a signal to the ContollerLightsProcess that a button has been used by a pedestrian. Since the application is a C# console application, the ConsoleKey() methods are used to read any key used on the keyboard. The 'N' and 'S' keys are used for the NS direction and the 'E' and 'W' keys are used for the EW direction, and the 'Q' key is used quitting the application. The 'Q' is used to set the RunFlag to false that will stop all the running processes.

```
    class PedestrianButton
    {
        Thread tPedestrianButton;
        bool bRunFlag;

        public PedestrianButton()
        {
            bRunFlag = true;
            tPedestrianButton = new Thread(new ThreadStart(this.PedestrianButtonProcess));
            tPedestrianButton.Start();
        }
        void PedestrianButtonProcess()
        {
            Semaphore sePedestrianNSButton = new Semaphore(0, 1, "sePedestrianNSButton");
            Semaphore sePedestrianEWButton = new Semaphore(0, 1, "sePedestrianEWButton");
            ConsoleKey cKey = ConsoleKey.S;
            ConsoleKeyInfo cKeyInfo;

            /// Check button
            while (cKey != ConsoleKey.Q)
            {
                cKeyInfo = Console.ReadKey();
                cKey = cKeyInfo.Key;
                switch (cKey)
                {
                    case ConsoleKey.N:
                    case ConsoleKey.S:
                        Console.WriteLine("  >> Wait: Pedestrian NS button ...");
                        sePedestrianNSButton.Release();
                        break;
                    case ConsoleKey.E:
                    case ConsoleKey.W:
```

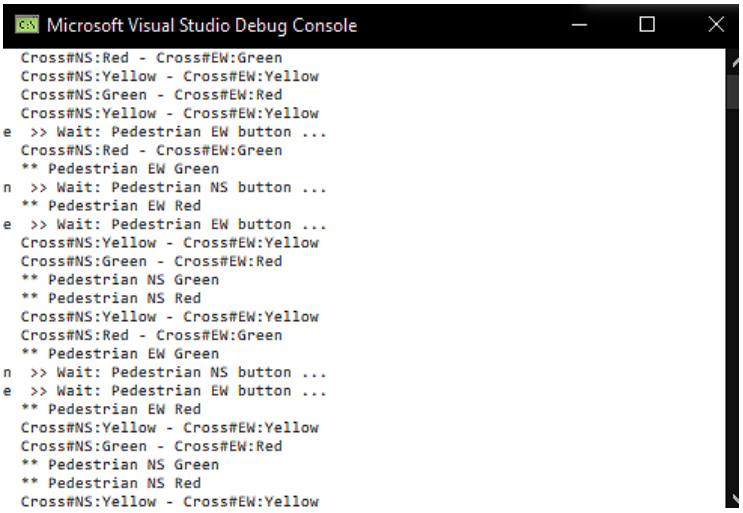


Figure 50.7: The output from the four processes for the traffic light controller. The first 11 lines is output from the states of the traffic lights, line 12 is a pedestrian pushing the EW button, and line 13 is a pedestrian pushing the SN button. Line 14 and 15 is continuing the traffic light states unntil a red light state for NS direction. Line 16 and 17 is the extension for the pedestrian crossing in the NS direction. Line 18 and 19 is continuing the traffic light states unntil a red light for the EW direction. The last two lines are the extension for pedestrian crossing in the EW direction.

```
                    Console.WriteLine("  >> Wait: Pedestrian EW button ...");
                    sePedestrianEWButton.Release();
                    break;
                case ConsoleKey.Q:
                    bRunFlag = false;
                    Console.WriteLine("  Press any key to quit!");
                    sePedestrianNSButton.Release();
                    sePedestrianEWButton.Release();
                    Console.ReadKey();
                    break;
                default:
                    break;
            }
            Thread.Sleep(100);
        }
    }
    public bool getUserStatus()
    {
        return bRunFlag;
    }
}
```

The objects are using the Console.WriteLine() statement to write information in the console window, as shown in Figure 50.7.

This figure shows first the normal sequences for the traffic light controller sequence in handling the NS and EW directions. In the middle of the list a pedestrian in EW direction is pushing the button, and the next line is a pedestrian in NS direction is pushing the button. The traffic light state machine continues the controlling the traffic lights until the red light is turned on for the NS direction. Now the pedestrian waiting for crossing the NS direction will have a green light and can cross the road, and the traffic light controller will delay the red period to let the pedestrian cross the road. Then the traffic light controller state machine will continue until the EW direction is in red light state, now the pedestrian waiting for crossing the EW direction will have the green light. So three structured processes with few statements will combine the multitasking possibilities with a set of semaphores for signaling between the

process to solve this application.

Any real-time requirement? Maximum waiting time for any pedestrians pushing a button?

Chapter 51

Operating systems

Windows, macOS and Linux are operating systems designed as General Purpose Operating Systems (GPOS) meaning that they will be a general operating system, they are not very suitable as a Real-Time Operating System (RTOS). Today versions of Windows, macOS and Linux is probable more a Network Operating System (NOS) than a GPOS. All modern operating systems like GPOS, NOS and RTOS are multitasking systems including solutions for Arduino (Grey 2020) and Raspberry Pi (Ibrahim 2020). The Raspberry Pi system is normally using a version of Linux or Windows.

Time is an important parameter for a RTOS as the system needs to fulfill the requirements of deadlines. The RTOS operates in an environment in which computer memory and processing power is limited, and need to provide the services in a definite amount of time. This means that every function in the OS must have an indication of the maximum time used before the control is given back to the RTOS application. Without this information it is not possible to use the OS, or the functions, in a RTOS. A RTOS is sometimes called real-time multitasking kernel. A RTOS is a software component that ensures an efficient processing of time critical events by dividing the application into multiple independent modules called tasks.

There are two main technologies of real-time operating systems. Shared Memory RTOS where the developer is responsible for protecting memory from mutual access by using specific RTOS tools like semaphores, and Direct-Message Passing RTOS where data is encapsulated in messages used for both inter-process communication and synchronization.

Several operating systems are designed as RTOS, both in complexity and price. The most popular RTOS[1] is shown in Figure 51.1, showing VxWorks, XP Embedded, Windows CE, DSP/BIOS, Red Hat Linux and QNX on the top.

Different factors are important when selecting a RTOS, especially factors like price of the development system, price for each license and the development tools. Technical factors will be the number of interrupt levels, the number of priority levels and the strategy of the scheduler.

The POSIX (Portable Operating System Interface) standard (by ISO, IEC and IEEE) has included four levels for real-time support. These levels are. 1) Minimum (embedded without the file system), 2) Controller (minimum with file system and asynchronous I/O), 3) Dedicated (controller with MMU), and 4) Multipurpose (dedicated with network and user interface).

51.1 RTOS requirement

The requirements for a RTOS are 1) multitasking, 2) preemptive, 3) the number of priority levels should be at least 16, preferably 64 or 256. The number of levels depends on the number of tasks in the system, 4) predictable synchronization mechanisms, 5) interrupt latency (depending on hardware), 6) several interrupt levels (depending on hardware), 7) deterministic response, by knowing the current states of a system, any states at any time in the future can be predicted, and 8) a set of services like time management, synchronization and message passing.

Important specifications for some of OS systems are listed in Table 51.1.

A RTOS often consists of a small kernel and a set of services to fulfill the RTOS requirements. The main service of a kernel is the task management together with memory allocation, syncronization,

[1] From july-2006.

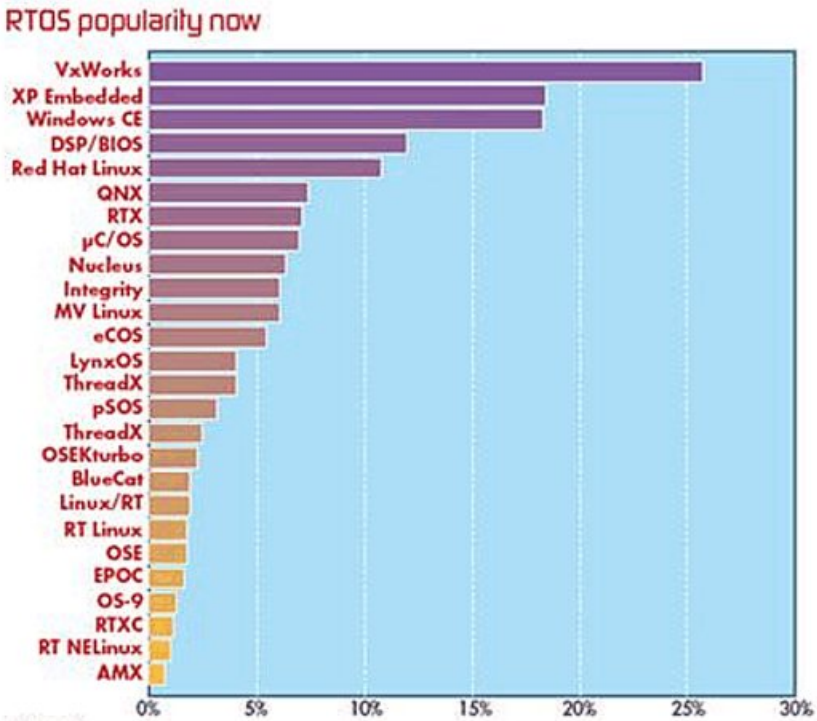


Figure 51.1: The RTOS popular in july-06, from EmbeddedSystems Europa June/July 2006, page 18.

Table 51.1: Some of the important OS specifications for real-time systems.

Requirements	WxWorks	XP Emb.	Win CE	Linux	QNX
Multitasking	Yes	Yes	Yes	Yes	Yes
Preemtible	Yes	Yes	Yes	Yes	Yes
Priority levels	256	6	256	32	32

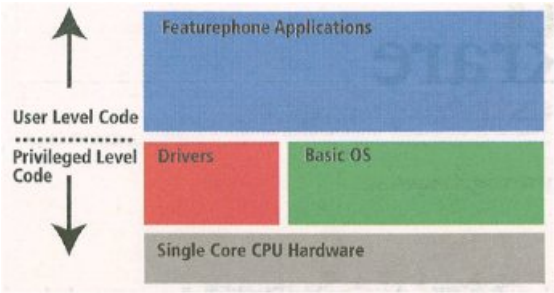


Figure 51.2: A simple OS with a set of drivers and basic OS functions. The applications will use the drivers and the basic OS functions.

intertask communication and device I/O. An example of such a real-time kernel as a RTOS is (Labrosse 1999).

51.2 Driver

The driver is software modules used as communication between the RTOS and the hardware. Any operating system is using a Hardware Abstraction Layer (HAL) for communication (interface) with a virtual driver. The driver will be a software module for communication (interface) between the virtual driver and the hardware. The responsibilities of drivers are to control, manange and configure I/O devices connected to the system. These modules will be a layered structure like:

Operating system Services	OS software
Hardware Abstraction Layer	OS software
Driver	Software
Hardware	

The driver can be a problem for Windows and Linux system as the driver often is developed for a GPOS and not RTOS. In RTOS the response time is important and the driver must be developed for a fast response from the hardware to the RTOS. The driver must also be robust (reliable) as a real-time system very often will be a $24x7$ system.

Figure 51.2 shows a simple OS or RTOS with drivers and a set of basic OS functions. The application will use only the drivers and the OS functions to communicate with the hardware, the application should NOT contain any specific coding for the hardware.

51.3 Windows / Linux

Windows and Linux are not RTOS even if the Linux kernel 2.6 and later has much better real-time performance then before. One solution that can be used if you still want to use Windows or Linux is the solution shown in Figure 51.3. The solution is to use a real-time kernel (a simple real-time operating system) where the standard operating system like Windows and Linux is running as process on the lowest priority. The real-time kernel will be the scheduler for the real-time processes and the operating system process. The scheduler in the standard operating system will be the scheduler for the standard operating system processes. This system will then have both real-time support and the standard operating system support, the standard operating system must even be installed from the standard CD (or DVD).

Another solution (www.ardence.com) is a real-time solution for Windows where the real-time system is running in parallel with Windows, see Figure 51.4. The communication between Windows and the real-time sub system is solved using IPC, a driver and a DLL[2]. This way can real-time processes and Windows processes communicate and be synchronized.

Figure 51.5 shows a solution for Linux or Symbian OS where the OS is part of the User Level Code, only a RT Executive is part of the Privileged Level Code.

[2] DLL: Dynamic Link Library.

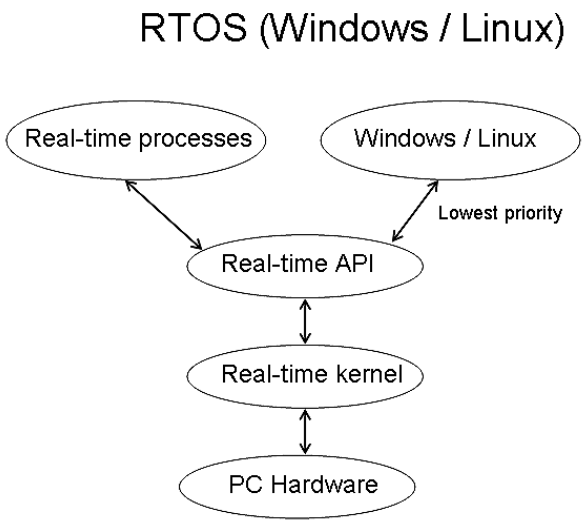


Figure 51.3: One solution to have real-time support together with a standard OS like Windows and Linux.

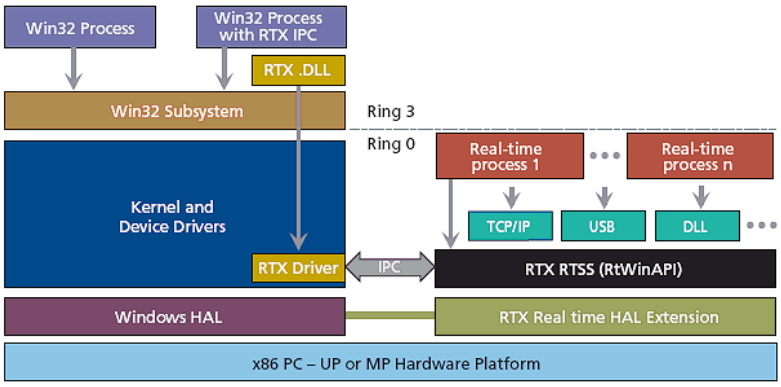


Figure 51.4: A real-time solution for Windows (www.ardence.com).

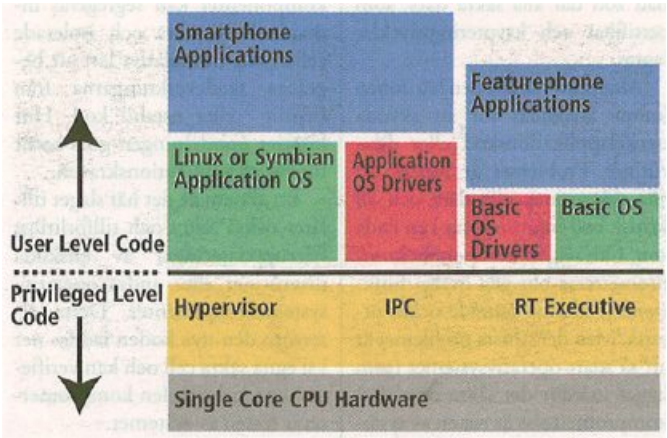


Figure 51.5: A real-time solution for Linux or Symbian with a high degree of system security as most of the code is in *User Level*. This solution is for different types of phones.

```
Command Prompt
22.08.2006  18:12    <DIR>          Documents and Settings
10.10.2006  09:34    <DIR>          Garmin
22.08.2006  18:13    <DIR>          Links
06.09.2006  16:07               871 msildbg.log
09.02.2009  12:19    <DIR>          My Documents
23.04.2009  10:54    <DIR>          NDPS
10.06.2005  13:38    <DIR>          NOVELL
06.09.2007  11:15           165 136 npnipp.dll
22.04.2009  14:49    <DIR>          Paint Shop Pro 7
23.03.2009  09:49    <DIR>          Program Files
23.08.2006  09:29    <DIR>          Programfiler
30.04.2008  09:43    <DIR>          Programfiles
25.08.2006  14:33             7 021 PWlang.log
16.03.2009  09:36    <DIR>          TEMP
11.03.2009  14:39    <DIR>          Usr
26.08.2005  13:55        25 165 824 VIRTPART.DAT
23.04.2009  12:08    <DIR>          WINDOWS
01.04.2009  15:52               600 winscp.RND
24.05.2005  15:26                 0 WSREMOTE.ID
22.04.2009  23:25    <DIR>          Zenworks
               9 fil(er)       25 369 452 byte
              15 mappe(r)   5 220 143 104 byte ledig

C:\>cd usr

C:\Usr>dir
 Volumet i stasjon C er NOS_HIT_C
 Volumserienummeret er 809A-27B1

 Innhold i C:\Usr

11.03.2009  14:39    <DIR>          .
11.03.2009  14:39    <DIR>          ..
06.02.2009  13:45    <DIR>          bin
06.02.2009  13:40           155 120 bin.zip
11.03.2009  14:39    <DIR>          Images
11.05.2007  17:03    <DIR>          Pub
               1 fil(er)          155 120 byte
               5 mappe(r)   5 220 143 104 byte ledig

C:\Usr>cd pub

C:\Usr\Pub>_
```

Figure 51.6: A "DOS" terminal showing some commands and the results from this commands. The terminal has only a text based user interface, it is not a graphical user interface.

Why do we want to use Windows or Linux? The cost of a developer is often higher than the system cost and a developer has some knowledge how to use these systems, 2) good support of development tools, and 3) good debugging tools.

51.3.1 Windows history

Windows started with the Microsoft DOS (MS-DOS) in 1982, the standard operating system for the first PCs. The MS-DOS was a single user, single task, operating system with a "DOS" terminal for typing the commands. Figure 51.6 shows such a text based user interface, it is also known as a "DOS" window.

The history of the Windows operating system is:

Windows	**Major changes**
"DOS"	Text based user interface and only one "DOS" window.
95	Greatly simplified interface (graphical); much more friendly to the average user
98	Improved multimedia capabilities and built-in Internet functionality
2000	Industrial-strength Windows NT code base, but in a much more polished package
XP	Unified the Win9x and WinNT/2K code bases; allowed businesses to standardize on one OS
Vista	Added to much security, avoid a lot of WinXP application to run.
7	Merging the good solutions from XP and Vista
8	Adding more and "harder" security to Windows 7.
10	Adding more features to Windows 8, and include removed "useful" options from Windows 7.

51.3.2 Windows Embedded

Windows XP or 7 Embedded is an OS build of modules and component. The number of modules and components can be configured to fit the requirement of a system, function, CPU and memory. A Windows XP Embedded will be a standard Windows XP system if all the modules and components are included.

51.3.3 Windows CE, Windows Embedded or Linux

When selecting a RTOS should you select Windows CE, Linux, or a specific RTOS?

1. Windows XP/7 Embedded is an embedded OS, not a RTOS, but has an option for a full Windows API,

2. Lisence fee for WCE, Linux is “free”,
3. WCE has a subset of the Windows API,
4. WCE from v3.0 (2000) is a RTOS, from v4.2 it is a stable RTOS, from v6.0 (2006) Visual Studio can be used as a tool,
5. The startup “cost” is higher for Linux than Windows CE,
6. Windows CE has only one distribution, Linux has several,
7. Windows CE has a footprint from 300 KB, but between 8-12 MB is usual with display support etc.. With explorer will the footprint be about 20 MB. A normal Linux footprint is about 1-4 MB, minimum from 400 KB. A normal Linux system is about 8-16 MB,
8. Documentation of Linux is much better than Windows CE.

51.4 QNX

QNX Neutrino is a popular RTOS that is both time tested and field proven[3]. The QNX Neutrino RTOS is said to be a micro kernel operating system, meaning that every driver, application, protocol stack, and file system runs outside the kernel, in the safety of memory-protected user space. Any component can fail, and will be automatically restarted without affecting other components or the kernel.

[3] QNX has delivered RTOS since the beginning of the 1980, see www.qnx.com for more information.

Chapter 52

Design of RTOS tasks

A real-time system consists of a set of time requirements and these time requirements are the key focus for the design. The functional requirements, normally controlled by the time requirements, require a distribution of the functions. This distribution is solved by dividing the system into a set of sub tasks, and in real-time system will the distribution be structured with one time requirement, or time requirements belonging to each other, in the same sub tasks. In practice will most of the requirements be soft real-time, very few of the time requirements are hard real-time.

A real-time system consists of both hardware and software, while the real-time requirments are mainly solved in software. A software application should also be divided into different tasks, or modules, regardless if it is a real-time system or not. This mean that the design of a real-time system should be handle like a normal system, but with more focus on the time requirements. In a real-time systen care should be taken that the main system or any task should not be able to delay the whole system to destroy the real-time responses. Often extra hardware can be added to handle any hard real-time requirement(s) (distributed systems).

Normally, communication between tasks are easy and fast in most operating system, and espesially for real-time systems. Let all I/O devices have their own tasks, like any DAQ system (Dickinson 2013), keyboard, screen, sensors and actuators.

52.1 Benefits of any RTOS

The benefit of using a RTOS (Schultz 1999):

1. It is more easy to get all the details when breaking a set of jobs into tasks,
2. Multitasking can make it easy to respond to real-time demands,
3. Intertask communication provides a solid method of controlling execution order and timing,
4. Tasks can be independent,
5. Tasks will be small and easier to manage,
6. Commercial or build-your-own?
   (a) commercial RTOS will be fully debugged and tested,
   (b) commercial RTOS will have a license fee,
   (c) commercial RTOS will be more general (efficient and memory size),

52.2 Selection of a real-time system

1. performance; focus on context switching and interrupt latency,
2. software development tools; quality and speed of developing the software and debugging,
3. algorithm integration; included libraries, include open source modules, access to high level functions and modules,

4. communication protocols,
5. hardware and I/O devices support, development of drivers,
6. code footprint; memory requirements,
7. supporting modules like task management, scheduler strategies, semaphore services, and memory management,
8. learning time of the tools.

52.3 Design of a real-time system

The sub tasks in a real-time system will often run concurrently giving a stronger restraints of the design (Ripps 1989). The rules for real-time tasking (Ripps 1989):

1. Use structured design (make the application more easy to design, implement, test, and maintain);
   (a) The main activities should be assigned to at least one separate task for each of these activities,
   (b) Closely related functions should be kept in the same task,
   (c) Functions that deal with the same inputs, data, and/or output should be kept in the same task,
   (d) Activity with I/O devices should be kept in a separate task, gives better error handling and scaling,
   (e) Try to make the tasks manageble, divide a “large” task into ”smaller” tasks if difficult to manage.
2. Try to keep the CPU(s) always busy with productive work;
   (a) Use the operating system wait function in stead of a wait loop.
3. Functions with different attributes must be assign to different tasks;
   (a) Functions with different requirements should be assigned to separate tasks,
   (b) Functions with significantly different levels of urgency must be assigned to separate tasks,
   (c) Functions proceeded at different time scales be assigned to separate tasks.

Part XIV

Embedded systems

Chapter 53

Introduction

This section contains embedded systems, with focus on the purpose of embedded systems and the properties of such systems. An embedded system is a:

1. special-purpose system in which the computer is completely encapsulated by the device it controls,
2. computer system hidden inside a product other than a computer (Simon 1999).

Unlike a general-purpose computer, such as a personal computer, an embedded system performs one or a few pre-defined tasks, usually with very specific requirements. Since the system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product. Embedded systems are often mass-produced, so the cost savings may be multiplied by the number of items sold (www.wikipedia.org 2006).

Embedded systems have at least a microprocessor (processor and I/O), memory and serial port(s). See Figure 53.1.

Today a network connection is more and more common, but the embedded systems usually do not have keyboard, screens, or disk drives. These systems are used as WEB serves among others. See Figure 53.2.

Examples of embedded system can be sensors, actuators, PID regulators and PLC in the industrial segments, and mobile phones, calculators, mp3 player, dish washers in the private segment to mention some. Industrial embedded systems are many, some of them are shown in some figures. Figure 53.3 shows a PID regulator as an embedded system as the PID regulator contains the microprocessor, the memory, the I/O, and the GUI is included in the same device. The computer will only be used for the operation of the PID controller. Figure 53.4 shows a PLC and a network switch, the PLC to the left, and the network switch to the right. These devices are also embedded systems as they contain a computer, but the computer can only be used for special device operations. Embedded systems for private area exist as well and some of these are shown in Figure 53.5. These devices are: (a) a SATA hard disk for a portable computer, (b) a digital pocket camera, (c) a MP3 music player, and (d) a mobile phone.

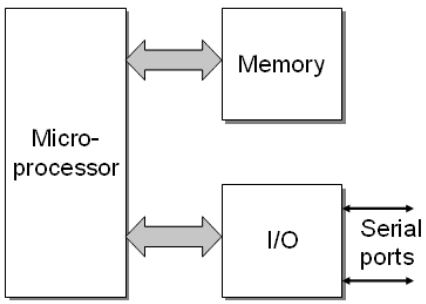


Figure 53.1: An embedded system with serial ports for communication.

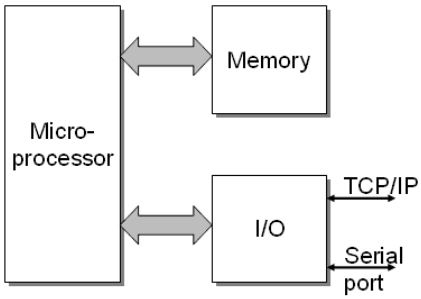


Figure 53.2: An embedded system with network connection and serial ports for communication.

Figure 53.3: A CD101 PID regulator as an embedded system.

Figure 53.4: A PLC to the left and a network switch to the right as embedded systems.

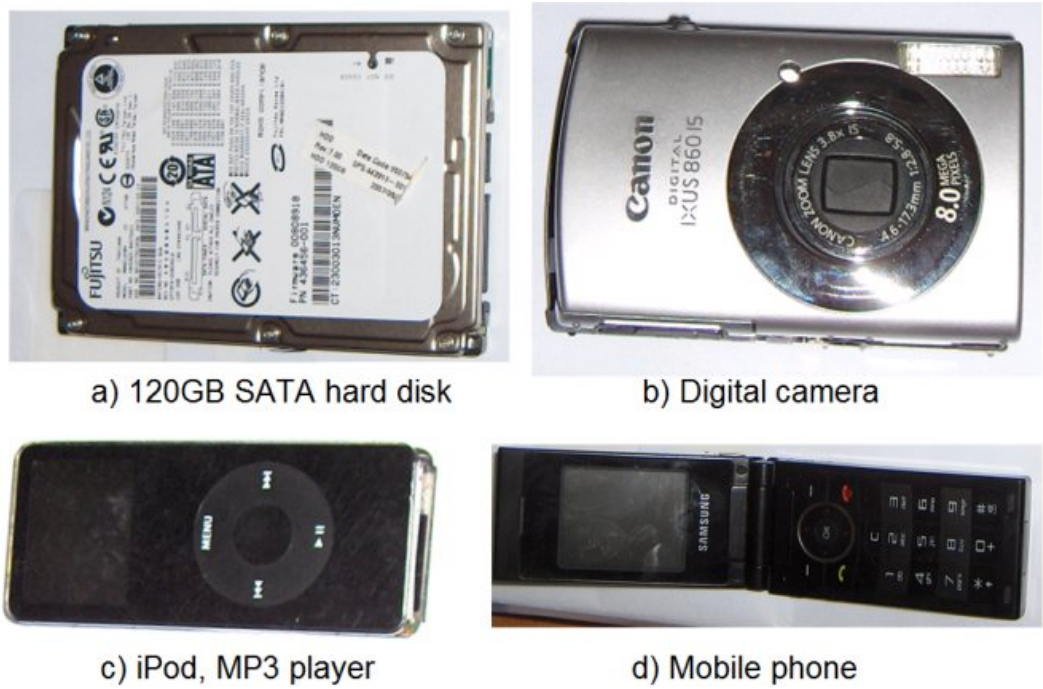


Figure 53.5: Some examples of embedded systems, the system is important not the computer. These examples are: (a) a 120GBytes hard disk, (b) a digital pocket camera, (c) a MP3 music player, and (d) a mobile phone.

53.1 Properties

Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some embedded systems may also have real-time performance constraints that must be met, for reason such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

For high volume systems such as portable music players or mobile phones, minimizing cost is usually the primary design consideration. Engineers typically select hardware that is just "good enough" to implement the necessary functions.

The software developed for embedded systems is normally named firmware, and is stored in ROM or Flash memory chips rather than a disk drive. It often runs with limited hardware resources: small or no keyboard, screen, and little RAM memory.

Embedded systems reside in machines that are expected to run continuously for years without errors and in some cases recover by them selves if an error occurs. These systems are also called $24x7$ systems, meaning 24 hour's 7 days a week. Therefore software is usually developed and tested more carefully than that for personal computers, and unreliable mechanical moving parts such as disk drives, fans, switches or buttons are avoided. Recovery from errors may be achieved with techniques such as a watchdog timer that resets the computer unless the software periodically notifies the watchdog.

A list of parameters that can be important for an embedded system:

1. Throughput; the throughput of data in the system,
2. Response; response to external events,
3. Reliability, embedded system must be able to handle any situation without human intervention,
4. Memory space; a limited resource and the application and data must always fit,
5. Program installation; how to install a new version on the embedded system,
6. Power consumption; embedded system should conserve power,
7. Process hogs; computing require extra CPU time can complicate response problem,

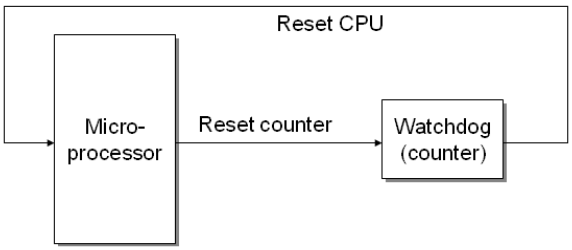


Figure 53.6: A watchdog control of the computer.

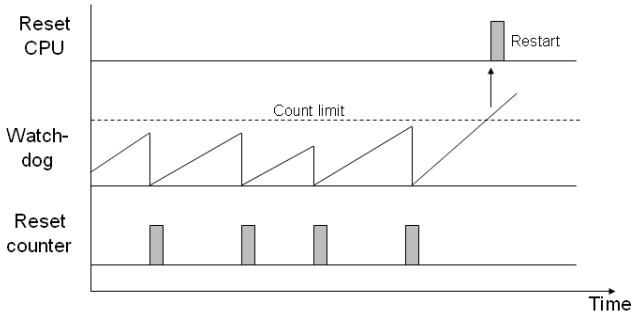


Figure 53.7: The signals and timing of the watchdog counter.

8. Testability; Important to test embedded system,
9. Debug ability; without screen and keyboard, how to debug the embedded system?
10. Cost; reducing the cost of hardware and software is important when developing embedded systems.

53.2 Real-Time Systems

A real-time system is normally an embedded system, but an embedded do not need to be a real-time system. Why? In a real-time system time requirements are important meaning that the dead-lines should be fulfilled. In order to fulfill the dead-lines the system should have an limited number of activities being an embedded system. The response of an embedded system do not need to be within any dead-lines and therefor not need to be a real-time system.

53.3 Watchdog

A watchdog is a hardware device, normally a counter, giving an output signal after a specific time to reset the system. A software task, with a low priority, has to reset the watchdog counter periodically to avoid the watchdog counter to output the signal for resetting the system. See Figure 53.6.

The watchdog counter will than normally be reset by the software and no reset of the system occurs. If some kind of error occurs, the watchdog timer will not be reset, and after the specific time, the system will be restarted. The timing and signals are shown in Figure 53.7.

53.4 WEB user interfaces

Embedded systems range from no user interface at all - dedicated only to one task - to full user interfaces similar to desktop operating systems in devices such as mobile phones and PDAs. In between are devices with small character- or digit-only displays and a few buttons (www.wikipedia.org 2006).

The rise of the World Wide Web has given embedded designers another quite different option, by providing a web page interface over a network connection. This is successful for remote, permanently installed equipment. This avoids the cost of a sophisticated display, yet provides complex input and

Figure 53.8: Using the web client as a GUI of the embedded systems.

display capabilities when needed, on another computer. These devices can be WEB servers, switches, routers, firewalls, and gateways among others.

Almost all computer systems today has a web client like Firefox, Opera, or Internet Explorer and by using the internet address of the embedded system, the web client can be connected to the web server. A user name and a password should be mandatory when logging into a web server. The web client will be the GUI of the embedded system for configuration, logging, monitoring, debugging etc. See Figure 53.8.

53.5 CPU platform

There are many different CPU architectures used in embedded designs, but important factors often are:

1. power,
2. heat,
3. number of I/O ports (pins),
4. integrated communication protocols,
5. using micro controller instead of a microprocessor.

A micro controller (or MCU) is a "computer-on-a-chip". In contrast to a general-purpose microprocessor, a micro controller can contain the memory and interfaces needed for a simple application, whereas a general purpose microprocessor requires additional chips to provide these functions.

CPU architectures like "system on a chip" (SoC) or application-specific integrated circuit (ASIC) can also be used for "high-volume" embedded systems.

53.6 Debugging

Embedded Debugging may be performed at different levels, depending on the facilities available, ranging from assembly- or source-level debugging with an in-circuit emulator, to output from serial debug ports, to an emulated environment running on a personal computer. Devices:

1. Light-emitting diodes (LED) on the print displaying a binary pattern (or seven-segment display),
2. Serial port for output of system status and debug status,
3. Serial port for output of system status and debug status, and input of user commands,
4. Network port for output of system status and debug status, and input of user commands,
5. In-circuit emulators for debugging the software,
6. Hardware emulators for running the software on a PC platform,

As the complexity of embedded systems grows, higher level tools and operating systems are migrating into embedded systems as well.

53.7 Start-up

All embedded systems must have start-up code. Usually it sets up the electronics, runs a self-test, and then starts the application code. The start-up process is commonly designed to be short, such as less than a tenth of a second, though this may depend on the application and type of operating system. The system also needs a boot loader loading the software from some kind of storage into the memory, and then starts the software.

Start-up sequence:

1. self test,
2. boot loader loads the application from storage memory to working memory,
3. run the application.

53.7.1 Self Test

Most embedded systems have some degree or amount of built-in self-test, running at start-up time. The self test is part of start-up sequence, included in part #1 and #2. In safety-critical systems, they are also run periodically or continuously. There are several basic types:

1. Testing the computer: CPU, RAM, and program memory. These normally run once at power-up. Memory check; very often using the patterns $0x55$ (0101 0101) and $0xAA$ (1010 1010) to check the memory,
2. Tests of peripherals: Verify the I/O devices installed, simulate inputs and read-back or measure outputs.
3. Tests of power supply, including batteries or other backup.
4. Safety tests: These run within a 'safety interval', and assure that the system is still reliable.

Part #1 and #2 of the start-up sequence is testing of hardware devices, and is known as the self test of the system.

53.7.2 Boot Loader

Any computer systems need a boot sequence from the power is turned on until the application is running. When a computer system is turned on, or is reset, the CPU will start executing code from a specific address in memory. A BIOS (Basic Input/Output System) is located at these addresses containing the startup code for the system. Note that this BIOS has to be developed for a specific hardware design as it will be the interface between the used hardware and the software. The BIOS code is often called firmware as it is very hardware specific and is the lowest level interface between the hardware devices and the software. The BIOS has to be located on the mainboard, often as a ROM (Read Only Memory), an EPROM (Erasable Programmable ROM) or a FLASH (type of electrically erasable programmable read-only memory (EEPROM)) device. Computers today normally use a FLASH device which can be updated by a software application. Remember however the usage of the BIOS, if the code is destroyed during the update process, the system will not be able to boot.

Newer computers have extended the BIOS with UEFI (Unified Extensible Firmware Interface) and on some computers has UEFI replaced the BIOS. The contents of the UEFI can be replaced by the operating system (OS) or specific update software, and can be a security challenge as the UEFI code can be extended with a virus or ransom code.

A normal booting sequence are:

1. The CPU is reset and starts executing the code in the BIOS or UEFI (located in a FLASH memory on the motherboard),
2. The BIOS or UEFI will do a setup and self-test of the hardware devices,
3. The BIOS or UEFI will load the master boot record (MBR) from a external storage device into the RAM,
4. The BIOS or UEFI will give control the MBR,

5. On a diskless system the MBR will
   (a) initialize the OS,
   (b) startup the software tasks,
6. On a disk system, MBR is the main sector on the disk containing the layout of the disk (partition table),
   (a) MBR will check the active partition, or display a menu to let the user select a partition ,
   (b) MBR will use the partition table to locate the main sector for the selected OS and load this sector,
   (c) the OS boot sector will load the rest of the OS and start the system.

Remember that a disk is a slow block device because all data are stored in several blocks. These blocks can be anywhere on the disk so the OS will need a table to be able to locate the correct blocks. The searching and locating these blocks takes some time, especially on a disk with a rotating magnetic media.

53.8 Reliability

Embedded systems should have a large uptime and the availability of a system is:

$$Availability = \frac{MTTF}{MTTF + MTTR}$$

where $MTTF$ is Mean Time To Failure and $MTTR$ is Mean Time To Repair. High availability (HA) means systems with 99.999% uptime ("five nines"). The availability can be improved by either decrease $MTTR$ or increasing $MTTF$. Difficult to measure or calculate $MTTF$ and this is often depending on the software, so increasing the $MTTF$ is a matter of cost. $MTTR$ can easily be computed, and improvement can often be made at a lower cost. Most HA recovery techniques focus on decreasing $MTTR$.

Chapter 54

Power

Embedded systems can be powered by different type of power sources like main power, battery, and solar cells among others. Many embedded systems are battery operated and power consumption is an important issue in analysis and design phase. Embedded systems always contain both hardware and software, and the software controls the hardware. This means that the software is controlling the power consumption of the system as the hardware devices is the devices using the power. Power debugging is a methodology providing system and software developers with information how the software implementation is affecting the power consumption of embedded systems.

Software development of embedded systems requires efficient code and minimizing the usage of memory. In low power system the power consumption needs to be taken into account. Power debugging needs to be supported in the development tools of the systems as different software solutions will affect the power usage of the systems. The usage of the whole system needs to be consider when reducing the power consumption, not only the CPU processor. The usage of the processor, any sleep period, adjusting the CPU clock, any sampling time, and the usage of the peripheral devices will all affect the power consumption, and can be controlled by the software.

Figure 54.1 shows the power consumption of an embedded system. The system is in sleep mode from t_0 to t_1 waiting for any events. The first event activates the system at t_1 giving a rise in the current from i_0 to $i_1$being the current consumption in active mode. A set of peripheral devices are also needed for this event so the total current consumption will be i_2. At time t_2 another event is at a higher priority will suspend the code for the previous event, but the peripheral devices is not turned off. The current consumption will rise to i_3 when the code for the higher priority evert is running.

The time period from t_2 to t_3 peripheral devices are power but not used, and the power consumption can be reduced by turning the devices off in this period. By debugging the current consumption together with the execution of the system, power debugging can be used to analyze and minimize the power consumption of such systems.

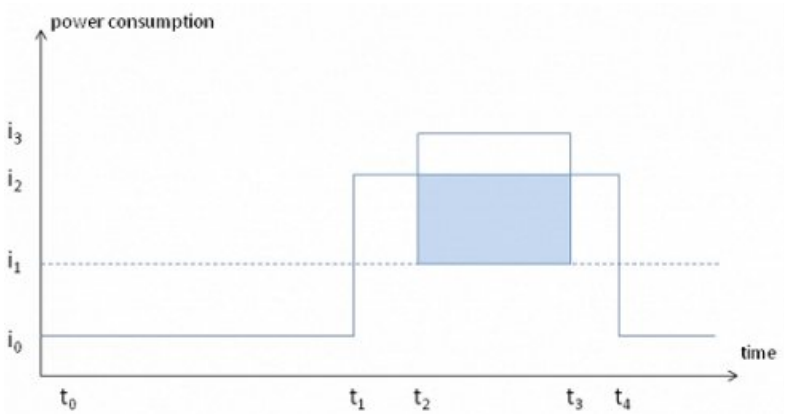


Figure 54.1: The power consuption of an embedded system. Unused hardware devices increase the power consumption from t_2 to t_3 (www.iar.com: JAN-2011).

Chapter 55

Embedded software architectures

The software will have a set of tasks to control the embedded system. Designing applications for an embedded system is always challenging. One way to decrease the complexity of such a system is to use a task-oriented design and divide the system into different modules (or tasks). Each module will be responsible for a sub part of the system. In such a system it is important to be able to specify that some modules (or tasks) are more important than others, assigning priorities to the modules (or tasks).

The system may have a scheduler, a software module controlling the execution of these tasks from a set of requirements. These requirements can be any of the following:

- priority; the importance of a task,
- time; how long the task has waited, or how long time the task will use for execution,
- order; the order waiting for execution,
- events or resources; waiting for an event or resource.

A scheduler is also named a task manager in many systems.

The system will only have a limit number of modules (or tasks) and several different types of software architectures are used to control the execution of the modules (or tasks):

55.1 Simple control loop

The software simply has one loop. This loop calls subroutines, each of which manages a part of the hardware or software. This system's strength is its simplicity, and on small pieces of software the loop is usually so fast that nobody cares that its timing is not predictable. It is common on small devices with a stand-alone micro controller dedicated to a simple task.

Weaknesses of a simple control loop are that it does not guarantee time to respond to any particular hardware event (although careful design may work around this), and that it can become difficult to maintain or add new features. This can be a solution if input of data is based on polling instead of interrupt driven input. Pseudo coding can be used for showing the functionality of such a simple control loop. Here the pseudocoding will be based on the C# programming language.

Example:

```
while (TRUE)
{
    task_#1() ;
    task_#2() ;
    task_#3() ;
}
```

The control loop can be implemented into an embedded system without the need of an OS. A common model for this kind of design is a state machine, which identifies a set of states that the system can be in and how it changes between them, with the goal of providing tightly defined system behavior. For a

simple scheduler as this the applications should be based on a state machine. Pseudo coding of a simple control loop based on a state machine.

Example:

```
task_#1()
{
   // If task is part of an object, state can be a locale variable in the object as long
   // as a new object is no created for every time. Otherwise a static must be used.
   switch (state)
   {
       case STATE_INIT:
           (void) initTask();
           state = STATE_READ ;
           break;
       case READ:
           state = readPort(state);   // Next state depends on read operation
           break ;
       case WRITE:
           state = writePort(state);  // Next state depends on write operation
           break;
       case WAIT:
           sleep(100);                // Wait 100 milleSec.
           state = STATE_READ ;
           break ;
       default:                       // Restart if an illegal state
           state = STATE_INIT;
           break;
    }
}
```

This example shows the usage of a state machine in some sort of communication protocol functions which first initializate the task and go into the read state waiting for the next message. The task will stay in this state until it has received some information from the port. Normally the readPort() function will just return READ. When a valid message has been received the function will return WRITE so the next time this function (task_#1()) is called it will write a message to the port. The writePort() will return WAIT meaning that the next time this function is called the task will be but in the wait queue of the scheduler to let the task wait for a new message. How can you extend this state machine to handle read errors, write errors and timeouts from the readPort() function?

55.2 Round Robin

Round Robin is the same as the control loop, but the loop checks each of the I/O devices in turn and services any I/O device that need service(Simon 1999). Round Robin uses a predefined time for each task to run (time slices), but is still handling the task without any priority. The Round Robin will check if the task is needing the services, otherwise will the next task be running in the time slice. The checks will normally be based on information from polling of data ports. Round Robin can be implemented in an OS kernel or without the OS. Pseudocoding of a Round Robin controller.

Example:

```
while (TRUE)
{
    if (task_1_flag == TRUE)
        task_#1() ;
    if (task_2_flag == TRUE)
        task_#2() ;
    if (task_3_flag == TRUE)
```

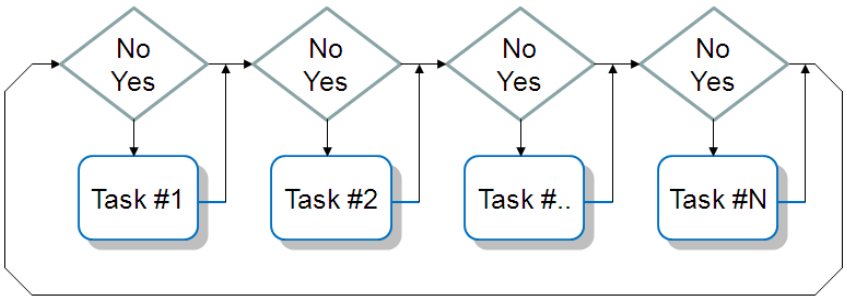


Figure 55.1: A control loop based on Round Robin with N tasks.

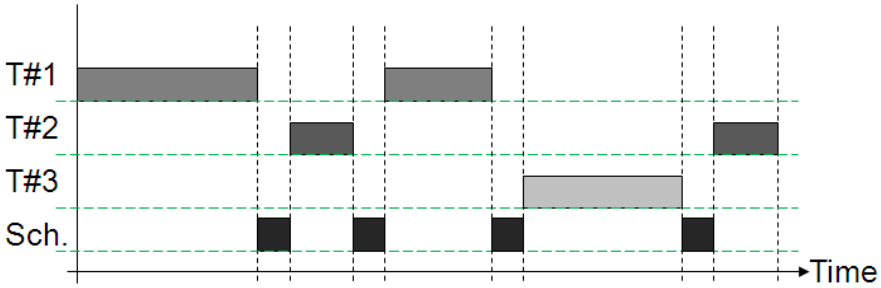


Figure 55.2: Non preemptive multitasking with three tasks and a scheduler. Each task will finished the current operation before giving the control back to scheduler.

```
        task_#3() ;
}
```

Figure 55.1 shows a graphical presentation of a control loop based on Round Robin.

55.3 Round Robin with Interrupt

An embedded system often depends on external events. Events can be detected by polling or interrupt, and interrupt is the most efficient way of doing it. One or several interrupt functions can be used to control the flags for running the tasks in the main loop of the Round Robin control loop.

55.4 Non preemptive multitasking

A non preemptive multitasking system is very similar to the simple control loop and Round Robin, except that the "loop" is hiding in the OS or the scheduler. A simple scheduler can be developed in stead of a OS depending on the number of services requested from the scheduler/OS. A set of software tasks are defined by the programmer, calling a system function when the task is finished. The system function will check the next task that can be run, and will start this task. The responsibility is still the programmers to make manageable sizes and logic of the tasks. The next task will NOT start before the previous task has ended. Some sort of scheduler will decide the next task to run.

Easy to add new tasks, but important to use "small" tasks. See Figure 55.2.

55.5 Preemptive multitasking

A timer is used to allow a task to run for only a specified period of time before the next task is running, called a time slice This is the level at which the system is generally considered to have an OS, and introduces all the complexities of managing multiple tasks running seemingly at the same time. See Figure 55.3.

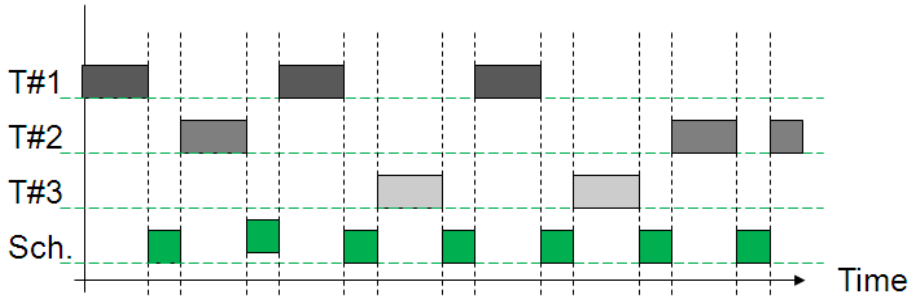


Figure 55.3: Preemptive multitasking with three tasks and a scheduler. The scheduler will take controll at a specific time interval, aborting the running task, and decide which task to run next.

Chapter 56

Operating system

An embedded operating system are designed to be very compact and efficient, removed many functionalities that non-embedded computer operating systems (OS) provide and which may not be used by the specialized applications they run. The reasons for having an OS is to a set of services to maintain the operation of the computer system. These services can be implemented in the application, but normally will a OS be used instead. An OS will also give more flexibility for the application, the number of hardware platforms to support, and a manageable complexity of the application.

56.1 Device driver

A device driver or a software driver is a specific type of computer software, typically developed to allow interaction with hardware devices. Often the operating system has specific interface, called Hardware Abstraction Layer (HAL), used for communication with the device driver. See Figure 56.1.

Device driver:

1. must be developed for self made hardware (or "rare" hardware devices),
2. no user interactions,
3. handle all type of errors,
4. never use any type of endless loops.

Advises:

1. select an Embedded OS with good device driver support,
2. select hardware devices with device driver for the OS,
3. if buying a device driver, try to get the source code as well,

56.2 Linux

Embedded Linux is a Linux based embedded operating system used in cell phones, personal digital assistants, media player handsets and other consumer electronics devices.

A Linux system consists of the Linux kernel and a set of applications. An embedded Linux consists normally of the Linux kernel and a minimum set of applications for monitoring and controlling the system. One of these applications is the Busybox[1] containing the commands needed for a minimum Linux system. Busybox is a single application where all the standard commands are implemented optimized on memory and performance for an embedded system.

Embedded Linux systems are:

1. UCLINUX; used on system with MMU,
2. Open Embedded; supporting different hardware platforms,

[1] Busybox, see www.busybox.net (dec-08).

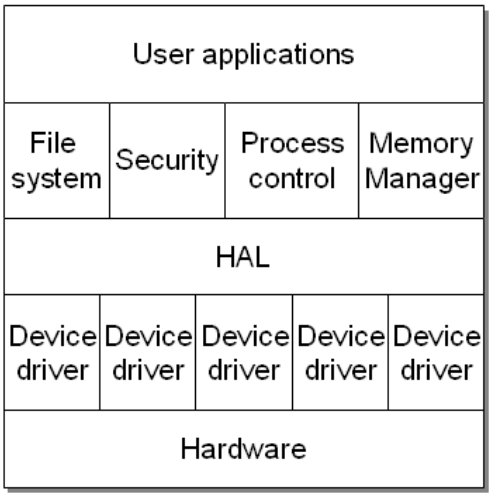


Figure 56.1: The structure layers of a operating system.

3. Buildroot; supporting different hardware platforms.

Summary:

1. Open source,
2. Footprint[2] of minimum 2MB,
3. No royalty costs,
4. Mature and stable (over ten years of age and used in many devices),
5. Well supported,
6. Real time support on newer versions.

56.3 Windows CE

Windows CE (sometimes abbreviated WinCE) is a variation of Microsoft's Windows operating system for minimalistic computers and embedded systems. Windows CE is a distinctly different kernel, rather than a "trimmed down" version of desktop Windows. This OS is also known as the Windows Mobile.

Summary:

1. Support more CPU types then standard Windows,
2. real-time support,
3. subset of Windows API (application programming interface),
4. footprint of minimum 350 KB,
5. Royalty cost.

56.4 Windows Embedded

Windows XP Embedded, like XPe, is the componentized version of Microsoft Windows XP Professional. XPe is based on the same binaries as XP Pro, but XPe is marketed towards developers that want the full Win32 API support of Windows but without the overhead of the full Professional version. It runs existing Windows applications and device drivers on devices with 32MB Compact Flash and 32MB RAM. An Embedded version of Windows 7 is also available, 7e. In the Embedded versions different modules

[2] Footprint: The memorysize used for the OS alone without any applications.

can be configured to adapt to the requirements of the system, and minimize the amout of RAM and CPU power needed. However, if network support is needed, all modules must be installed giving a full Windows version.

XPe is not related to Windows CE. They target different devices and they each have their pros and cons which make them attractive to different type of systems. For instance, XPe will never get down to the small footprint that CE works in. However, CE does not have the Win32 APIs XPe has (although CE has an API that is similar to the Win32 API), nor can it run the tens of thousands of drivers and applications that already exist.

Summary:

1. requirements for x86 (Intel or AMD) cpu,
2. real-time support (using extensions),
3. full Windows API (all Windows applications and device drivers),
4. footprint of minimum 8 MB,
5. Royalty cost,
6. Component based, with network support almost all components must be included being almost a full Windows XP!

56.5 eCOS

eCos (embedded Configurable operating system) is an open source, royalty-free, real-time operating system intended for embedded systems and applications which need only one process with multiple threads. The OS is configurable, and can be customized to precise application requirements, with hundreds of options, delivering the best possible run-time performance and minimized hardware needs. A thriving net community has grown up around eCos ensuring ongoing technical innovation and wide platform support.

Summary:

1. Open source,
2. Royalty free,
3. Real-time,
4. footprint of minimum 100 KB.

56.6 Mobile Phones and PDA

Operating systems for Mobile phones and PDA:

1. Symbian OS is an operating system, designed for mobile devices, with associated libraries, user interface frameworks and reference implementations of common tools, produced by Symbian Ltd. It is a descendant of Psion's EPOC and runs exclusively on ARM processors,
2. Palm OS is a compact operating system developed and licensed by PalmSource, Inc. for personal digital assistants (PDAs) manufactured by various licensees. It is designed to be easy-to-use and similar to desktop operating systems such as Microsoft Windows. Palm OS is combined with a suite of basic applications including an address book, clock, note pad, sync, memo viewer and security software. Palm OS was originally released in 1996,
3. Windows CE (sometimes abbreviated WinCE) is a variation of Microsoft's Windows operating system for minimalistic computers and embedded systems. Windows CE is a distinctly different kernel, rather than a "trimmed down" version of desktop Windows,
4. Embedded Linux,
5. IPhone; MAC OS for the iPhone, also called iPhone OS or OS X iPhone,

Figure 56.2: Many of the available systems for Mobile Phones (www.tu.no: APR-10).

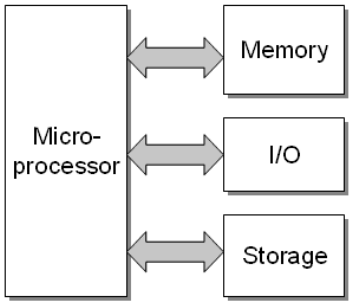


Figure 56.3: Storage memory for application in an embedded system.

6. Windows Mobile; Windows version for mobile phones and PDA,
7. Android; using the Linux kernel and allows development in the Java programming language,
8. RIM (Research In Motion); a proprietary multi-tasking operating system (OS) for the BlackBerry devices, supports the Java programming language,
9. Bada; an "open" platform on Samsung mobile devices.

Figure 56.2 shows most of available operating systems for mobile phones in the beginning of 2010, divided from open to proprietor.

56.7 Storage

An embedded system is normally not using standard disk, a non-rotating medium (no moving parts) is used for storage. See Figure 56.3.

Why is the embedded system avoiding disk drivers? Mainly for the wear and tear of the drive, for portable devices also the problem with strokes that can destroy the disk drive.

The most used storage types are:

1. ROM: Read-only memory (often referred to as its acronym ROM) is a class of storage media used in computers and other electronic devices. Because it cannot (easily) be written to, its main uses lie in the distribution of firmware (software that is very closely related to hardware, and not likely to need frequent upgrading). ROM, EPROM, EEPROM.
2. FLASH: Flash memory is a form of non-volatile computer memory that can be electrically erased and reprogrammed. It is a technology that is primarily used in memory cards. Unlike EEPROM, it is erased and programmed in blocks consisting of multiple locations (in early flash the entire chip had to be erased at once). Flash memory costs far less than EEPROM and therefore has become the dominant technology wherever a significant amount of non-volatile, solid-state storage is needed. Examples of applications include digital audio players, digital cameras and mobile phones. Flash memory is also used in USB flash drives (thumb drives, handy drive), which are used for general storage and transfer of data between computers. It has also gained some popularity in the game console market, where it is often used instead of EEPROMs or battery-powered SRAM for game save data.

3. NVRAM: Non-volatile random access memory (NVRAM) is the general name used to describe any type of random access memory which does not lose its information when power is turned off. This is in contrast to the most common forms of random access memory today, DRAM and SRAM, which both require continual power in order to maintain their data. NVRAM is a subgroup of the more general class of non-volatile memory types, the difference being that NVRAM devices offer random access, as opposed to sequential access like hard disks.

4. PCMCIA: PCMCIA originally stood for Peripheral Component MicroChannel Interconnect Architecture. It was then renammed to name the standards organization, the Personal Computer Memory Card International Association. PC card (originally PCMCIA card) is the form factor of a peripheral interface designed for laptop computers. It was originally for memory expansion, but the existence of a usable general standard for notebook peripherals led to all manner of devices being made available in this form. Typical devices include network cards, modems and hard disks.

5. USB sticks: Memory sticks that can emulate disk drives. Easy to upgrade the application software.

6. CD-ROM: A CD-ROM is a kind of compact disc. A CD-ROM is Read Only, hence the name, while CD-R is Write-Once-Read-Many (analogous to a PROM), and CD-RW is Read/Write (analogous to an EEPROM).

56.8 Memory

The memory allocation is a service of the operation system and is also treated as a resource in an embedded system. The normal operation is to allocate the memory when needed and release the memory after the usage. This should be avoided in an embedded system because the memory will be fragmented and the software will not be able to allocate more memory. This is special for $24x7$ systems as the memory allocation service (table) will never be reset. The solution for the embedded system can be to allocate all the memory at startup time, and just reuse this memory while running.

An application developed using Object Oriented methods will normally use more memory then a function based application.

56.9 Misc

Avoid using fan(s) as well since these devices contain moving parts and will stop working sooner and later. And what will happen with the system when the fans are not working?

An advice is to have regular maintenance of embedded systems using disk drivers or/and fans, and change these devices every second year.

Chapter 57

Development

An embedded device is a device with limit resources, both I/O devices, memory devices, cpu capacity and storage devices. Development tools like editors, compilers, linkers, debugger and source code control systems that are essential software development can not run on the embedded device. The software must be developed on a development system, often a computer running Windows or Linux, and then transfered to the embedded system. The development system is named as the host platform or host system and has the capability to manage the development cycle. The embedded system is named the target platform or target system and the developed software has to be transfered from the host system to the target system, uploaded to embedded system. The developed software has to be able to run on the target platform so if the target platform contains a different cpu than the host platform, a cross compiler most be used. A cross compiler will compile the developed software for the correct target platform. Using a different cpu platform will be a challenge for testing as the testing on the host platform will be done on a different cpu platform and with a software compiled and linked with another compiler. An emulator can be useful for testing on the host platform.

Challenges like cpu architectures with big endian and little endian data formats must be handle when developing the software. Big endian and little endian are terms that describe the order in which a sequence of bytes are stored in memory. Little endian is an order in which the least significant value in the sequence is stored first. Big endian is an order in which the most significant value in the sequence is stored first. A 16 bits value like 2048 is stored in memory as the hexidecimal number $0x0800$. This value stored as 00 at low address and 08 at high address for little endian cpu arcitectures, while big endian cpu architectures will store 08 at low address and 00 at high address. Developing software indexing data at byte level must be able to handle these cpu architectues.

Part XV

Ex / ATEX

Chapter 58

ATEX and Ex

58.1 Introduction

The usage of electricity in explosive atmosphere may cause an explosion. Ex regulations are used to prevent explosions when explosive atmosphere such as gas or dust accumulates, breaks out from a process or during handling to/from the process. The regulations try to confine and/or reduce the size and impact of explosions, if it is realized. The regulations are controlled globally, regulated by both governments and marked. The governments has control over WTO (World Trade Organization), IMO (International Maritime Organization), EU (The European union), EFTA (European Free Trade Association), while the marked is controlling ISO (International Organization for Standardization), IEC (International Electrotechnical Commission), CEN (Comité Européen de Normalisation/European Committe for Standardization), CENELEC (Comité Européen de Normalisation Electrotechnique/European Committee for Electrotechnical Standardization) and ETSI (European Telecommunications Standards Institute) regions.

The CEN/CENELEC/ETSI is standardization body, and they are given authorization from EU commission to develop harmonized norms/standards that fulfils the fundamental safety regulations in ATEX equipment directive. Even though norms in principle is voluntary, would the harmonized norms become the most important tools to fulfill the demand from security regulations in the directive.

ATEX (ATmosphères EXplosibles) is an EU Directive and equipment sold in other parts of the world may need to meet different local legislative requirements. In North America, documentation is submitted to an appropriate approvals body and testing is also carried out. In North America a set of classes and divisions are used in stead of zones. The classes are based on the type of danger, while the division is based on the danger of explosion for the material.

In the rest of the world, most countries use IEC Standards as national standards. The IECEx is rules is given by IEC. There are different directives and international standards related to explosive atmospheres. The rules related to Norway and Europe is the ATEX directive. However, at the bottom line, both the IECEx rules and the ATEX directives are almost the same standards.

Figure 58.1 shows an overview of the relationships of these regulations, this Figure has focus on Norway with some Norwegian descriptions.

The legislation is split, controlled by a set of ATEX directives. The important ATEX directives are:

- ATEX equipment directive 94/9/EC; also named ATEX 100a, Equipment and protective systems intended for use in potentially explosive atmospheres; In includes 16 Articles and XI annex,
- ATEX workplace directive 99/92/EC; also named ATEX 118a or ATEX 137, minimum requirements for improving the safety and health protection of workers potentially at risk from explosive atmospheres. This directive is a work environment directive that gives risk evaluation, area classification and organized actions in a workplace.

The connection between these directives is shown in Figure 58.2 where the ATEX equipment directive defines any requirement for the products to be located in explosive areas, while the ATEX workplace directive defines that the end-user is responsible for defining the type of explosive areas (zones), and the installation and usage of the equipement.

The ATEX directives have been valid in Europa since 2006. These directives are a framework for controlling explosive atmospheres and the standards of equipment and protective systems used in them.

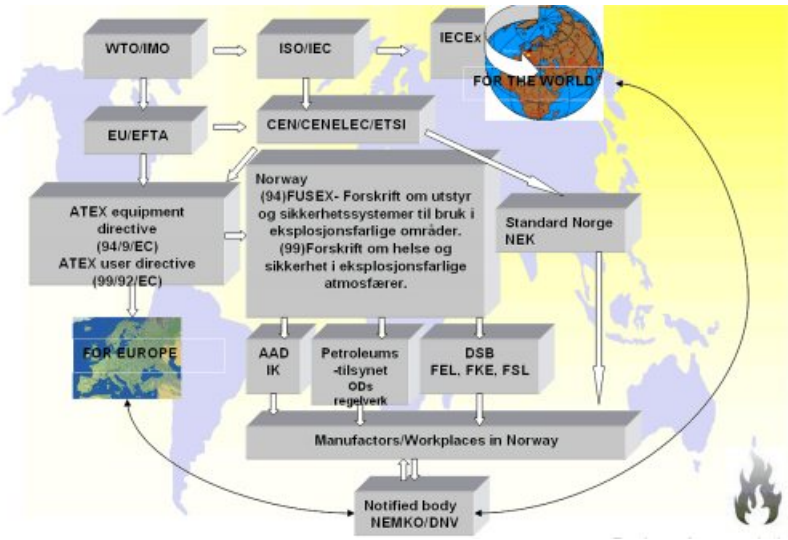


Figure 58.1: An overview of the Ex regulations world wide (copy from a presentation of T. Øyvang and K. Rai at TUC in 2006).

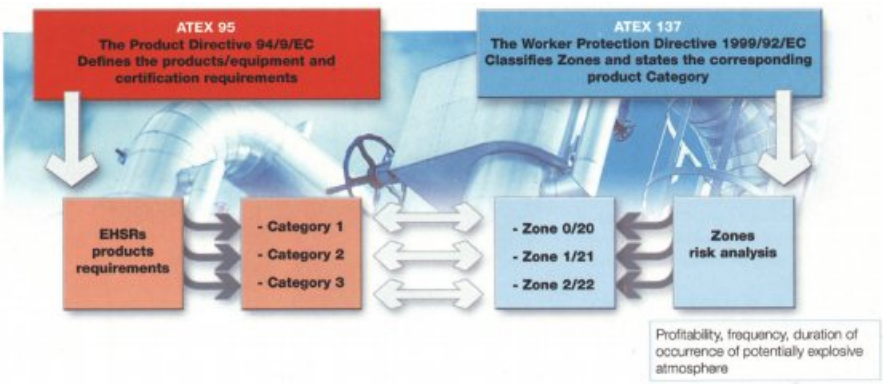


Figure 58.2: The connection between the ATEX equipement directive and the ATEX workplace directive (www.abb.com).

Figure 58.3: The Ex sign used to indicate an Ex approved device (Company 2010)

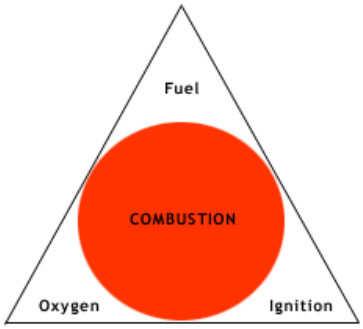


Figure 58.4: The combustion triangle showing the parts that must be available to get a combustion.

Devices to be installed into explosive atmosphere should be Ex approved and should contain a sign/stamp as shown in Figure 58.3.

The directives are using for avoiding explosions to prevent personal injury, damages on plant and devices, and down time of the plant and production resulting in loss of production. Explosive atmospheres or hazardous areas means areas where there can be a combustion. A hazardous atmosphere is an area that contains elements that may cause an explosion. Any combustion depends on a triangle as shown in Figures 58.4 and 58.5, showing what must be available to be able to have an combustion. This figure shows that a combustion needs the properties fuel, oxygen, and ignition.

Protection systems are used to eliminate or minimize at least one of these properties. An explosion can be catastrophic both for human, equipment and environment, and rules and/or directives are created to protect these properties. An example can be the explosion and fire in the Macondo prosject oil field 20-APR-2010, shown in Figure 58.6.

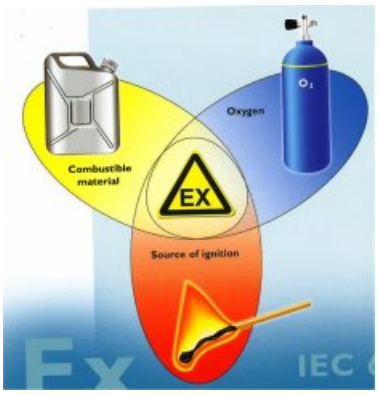


Figure 58.5: The combustion triangle showing the parts that must be available to get a combustion (Company 2010).

Figure 58.6: The Macondo accedent in April 2010 referes to the explosion and fire of the Deepwater Horizon drilling rig. The explosion killed 11 workers and injured 16 others. The drilling rig ended up sinking, resulting in a massive offshore oil spill in the Gulf of Mexico. This is considered the largest accidental marine oil spill in the world.

58.2 Definitions

58.2.1 Explosive atmosphere

Explosive atmosphere consists of a mixture of flammable substances with air in the form of gas, vapor or mist in such proportions that it can be exploded by excessive temperature, arcs or sparks. The gases, vapors or mists will only explode when mixed with air between specific percentage mixtures, these are called: Lower Explosive Limit (LEL) and Upper Explosive Limit (UEL)

58.2.2 Hazardous area

Hazardous areas are defined as places where concentrations of flammable gases, vapors, and/or dust may occur. Electronic equipment that is installed in such locations must be specifically designed, tested and certified to ensure that it does not ignite an explosion.

58.2.3 Explosion

Explosion is any uncontrolled combustion wave. In order to create an explosion the has to be fuel (for example and explosive gas such as hydrogen), oxidizer (oxygen in air) and a source of ignition (for example, a hot surface or an electrical spark. These three items are commonly referred to as 'fire triangle' and represented as below in figure 1. In addition to this, two additional facets are required; something to mix the fuel and the oxidizer (such as the turbulence created in a gas leak under pressure) and containment. It is however common industrial practice to use the term 'explosion' for both confined and unconfined combustion.

58.2.4 Potentially explosive atmosphere

Equipment is only considered to be within the scope of the directive 94/9/EC (will be defined later) if it is intended (either in whole or in part) to be used in a potentially explosive atmosphere. If a product containing an intended potentially explosive atmosphere, for example a vessel, itself contains equipment

as defined in the directive, then the latter equipment is in effect in a potentially explosive atmosphere, albeit one which is contained by the vessel, and is therefore subject to the directive. If equipment containing a potentially explosive atmosphere can, due to its construction, operation etc. create a potentially explosive atmosphere itself, which wholly or partially surrounds it, then such equipment is in effect in a potentially explosive atmosphere, and is therefore subject to the directive 94/9/EC.

58.2.5 Equipment

Equipment, as defined in directive 94/9/EC, means machines, apparatus, fixed or mobile devices, control components and instrumentation thereof and detection or prevention systems which, separately or jointly, are intended for the generation, transfer, storage, measurement, control and conversion of energy and/or the processing of material and which are capable of causing an explosion through their own potential sources of ignition.

58.2.6 Gas explosion

A gas explosion is defined as a process where combustion of a premixed gas-air cloud causes a rapid increase of pressure. Gas explosions may occur inside process equipment or pipes, in buildings or off-shore modules, in open process areas or in unconfined areas. A gas explosion is part of a chain of events as illustrated in the diagram below. When gas explosion risk is analyzed, it is necessary to analyze this chain of events from the onset of a leak, via dispersion and mixing (including the effect of ventilation conditions), ignition, the explosion itself to the structural response following the explosion.

58.2.7 Risk

Function of Severity (elements: possible harm for the considered explosion hazard) and Probability of occurrence of that harm (elements: frequency and duration of exposure, probability of occurrence of hazardous event, possibility to avoid or limit the harm).

58.2.8 Risk Assessment

A series of logical steps to enable, in a systematic way, the examination of the hazards associated with unit operations and equipment.

58.2.9 Hazard Identification

Hazard Identification is a systematic procedure for finding all of the hazards which are associated with the unit operations and equipment. It is the process of determining what, why and how things can happen.

58.2.10 Risk Estimation

Determination of the frequency at which the identified hazards could be realized and give rise to specified levels of severity.

58.2.11 Risk Evaluation

Comparison of the risk estimated with criteria in order to decide whether the risk is acceptable or whether the unit operations and/or equipment design must be modified in order to reduce the risk.

58.2.12 Risk Reduction Option Analysis

The final step of risk assessment is the process of identifying, selecting and modifying design changes which might reduce the overall risk from unit operations and equipment.

58.2.13 Residual Risk

The remaining level of risk after all actions has been taken to reduce the probability and consequence of risk.

58.2.14 Critical Ignition Energy

For any mixture of a combustible gas or vapor with an oxidizer there is a Critical Ignition Energy. If one releases less than that critical amount of energy into the mixture, there will not be a self-propagating explosion. Some combustion may occur transiently, but the combustion wave will not grow and become self–propagating. If one releases at least the critical amount of energy, the combustion wave will pass through the incipient stages of growth and become self-propagating as a plane wave, resulting in an explosion.

58.2.15 Most Easily Ignited Concentration (MEIC)

Most Easily Ignited Concentration (MEIC) is the amount of energy required to cause ignition is minimal. If the ignition experiment is conducted under conditions where it is assumed that all the energy injected into the gas/vapor cloud is used in the combustion process, the critical energy at the MEIC is called minimum ignition energy (MIE). As the concentration is varied from the most easily ignited concentration the amount of energy required to cause ignition in-creases, until at certain points, the mixture is no longer explosive.

58.2.16 Ignition Sources

The type of ignition sources can be: **a)** Hot Surfaces, **b)** Flames and Hot Gases, **c)** Mechanically Generated Sparks, **d)** Electrically Generated Sparks, **e)** Electrostatically Generated Sparks, **f)** Adiabatic Compression, **g)** Electromagnetic Radiation, **h)** Ionising Radiation, **i)** Chemical Reactions, **j)** Ultrasound, **k)** Flashes.

58.3 Standards and Certifications

58.3.1 IEC

The International Electrotechnical Committee (IEC) is creating international standards for electrical equipment. The specific committee TC31 is responsible for covering the hazardous areas. The standards for hazardous areas are numbered $IEC\ 60079 - xx : yyyy$ where xx is a number for a specific section and $yyyy$ is the year of publication. Almosts all national standards organisations are members of IEC. IECEx is an affiliated organisation whicj\h issues certificates of compliance with the IEC 60079 series of standards. Anyone can use the IEC standards, but can not create an IECEx certificate.

58.3.2 CENELEC

The European Committee for Electrotechnical Standardisation (CENELEC) is creating the European standards for electrical equipement. Currently IEC and CENELEC are using the same numbers for standards in the hazardous areas, except starts with EN in stead of IEC. CENELEC is also working to make the standards more acceptable interpretation of the ATEC directive. The ATEX directive is an European Union (EU) document rather than a CENELEC standard, and hence introduces some minor differences.

58.3.3 BSI

The British Standards Institute (BSI) is the UK member of both IEC and CENELEC. BSI is publishing an English language version of the CENELEC standards. The naming of the documents are $BS\ EN\ 60079 - xx : yyyy$.The technical content of a IEC, CENELEC and BSI standards are the same.

58.3.4 ATEX

The are two ATmosphere Explosive (ATEX) directives currently in use. One directive covers the marketing and manufacture of equipment for use in hazardous atmospheres (94/9../EC and 2014/34/EU), also named the "Apparatus Directive". The other directive covers the minimu level of protection for workers in the industries using hazardous materials (1999/92../EC), also named the "User Directive". This directive requires a detailed and well documented risk analysis of an installation in hazardous area.

The ATEX equipment directive covers the design, manufacture and marketing of equipment for use in hazardous areas.

58.4 ATEX equipment directive

There are three requirements for a product to be comprised by this directive. These requirements are that the product:

1. must have an ignition source like a hot surface and/or spark,
2. is intended for explosive aresa,
3. is going to be used under normal atmospherical conditions.

Equipment categories based on the 100a equipment directive:

- Based on Annex I - Classification of Equipment-groups into Categories,
- decision based on the intended use, as to which Group and Category the product belongs,
- The directive divides equipment into two groups:
  - Group I is for equipment used in underground mines and surface installations, and is divided into two groups:
    - Category M1; Products of this Category are required to remain functional for safety reasons when an explosive atmosphere is present.
    - Category M2; These products are intended to be de-energised in the event of an explosive atmosphere.
  - Group II covers other equipment and this is divided into three categories:
    - Category 1; Equipment that requires the highest level of protection as it will be used for long periods in potentially explosive atmospheres, and must be able to provide protection even if one means of protection fails or if two faults occur,
    - Category 2; Equipment needs a high level of protection for use where potentially explosive atmospheres are likely to occur,
    - Category 3; Equipment in this category will have a normal level of protection and is intended for use where potentially explosive atmospheres are infrequent or exist for short periods only.

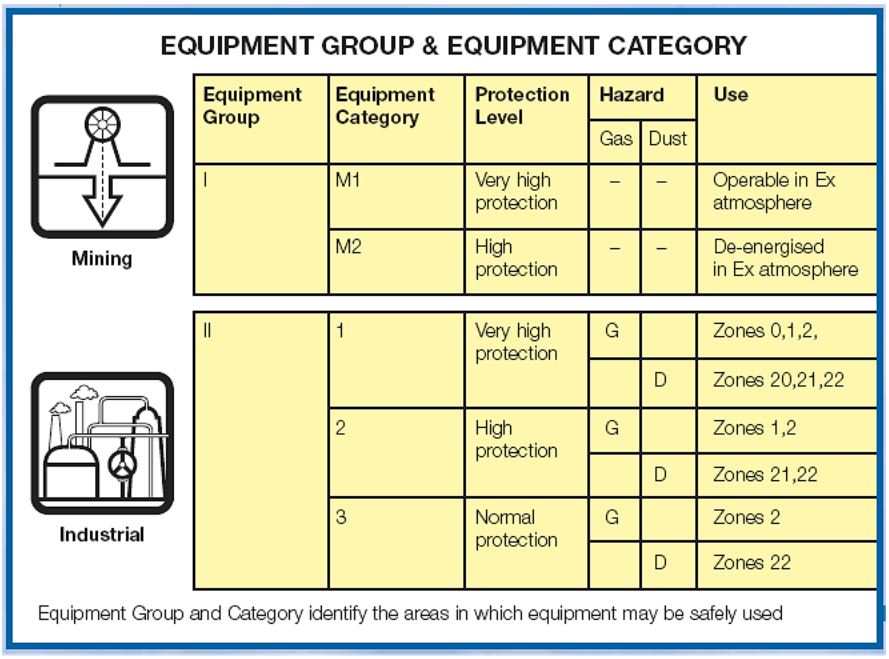
The equipment is divided into two groups, one group for mining and one group for industrial usage. The groups are listed in a table in Figure 58.7.

Gas or dust environment for equipment:

- Explosive atmosphere consists of a mixture of flammable substances with air in the form of gas, vapor or mist,
- Gases can be divided in two groups. Only Group II is subdivided into categories using the letters A, B and C. This is to classify the differences in the ignitability and the likelihood of flashover of potentially explosive mixtures.
- The gas groups are:

System	Gas group group	Representative test gas
Mining	**I**	Methane
	IIA	Propane
Industrial	**IIB**	Ethylene
	IIC	Hydrogen

Hazardous areas for the equipment:



EQUIPMENT GROUP & EQUIPMENT CATEGORY

Equipment Group	Equipment Category	Protection Level	Hazard		Use
			Gas	Dust	
I (Mining)	M1	Very high protection	–	–	Operable in Ex atmosphere
	M2	High protection	–	–	De-energised in Ex atmosphere
II (Industrial)	1	Very high protection	G		Zones 0,1,2,
				D	Zones 20,21,22
	2	High protection	G		Zones 1,2
				D	Zones 21,22
	3	Normal protection	G		Zones 2
				D	Zones 22

Equipment Group and Category identify the areas in which equipment may be safely used

Figure 58.7: The equipment categories for group I and group II (www.wolf-safety.co.uk 10-DEC-09).

The hazardous areas are divided in zones for gas and dusts. These zones are (see Figure 58.7):

Gas	Dust	Zone criteria
Zone 0	Zone 20	explosive atmosphere is continuously present, or present for long periods. (more than 1000 hours per annum, more than 1.5 month a year).
Zone 1	Zone 21	explosive atmosphere is likely to occur in normal operation (between 10 hours and 1000 hours per annum).
Zone 2	Zone 22	explosive atmosphere is not likely to occur in normal operation. If it does it will only be for a short time (less than 10 hours per annum).

Figure 58.8 shows examples of hazardous area zones for a gas or a petrol station. When refilling petrol or gas on a vehicle the area will be zone 1. Notice that any basement area in your garage is also zone 1, and zone 1 regarding hazardous areas.

58.5 Equipment protection

There exists different solutions to protect equipment to cause an combustion. The method is to remove one of the property in the combustion triangle, see Figure 58.4. The most used protect methods are:

- oil immersion: Ex o; the equipment is immersed in oil or protective liquid, excluding the explosive gas,
- pressurized: Ex p; the equipment is filled with an inert gas with a higher pressure than the environment gas pressure, excluding the explosive gas. Often this solution will require a flow of inert gas to maintain the overpressure of inert gas.
- powder filled: Ex q; the equipment is filled with quartz or glass powder particles to prevent the possibility of an ignition.
- flameproof: Ex d; the equipment is placed within an enclosure that can withstand the force from an explosion. The enclosure also prevents the hazardous atmosphere to come in contact with the equipment.
- encapsulation: Ex m; the equipment is covered by a compound or resin to prevent contact with the hazardous atmosphere. The encapsulation also limits the surface temperature of the equipment during operation.

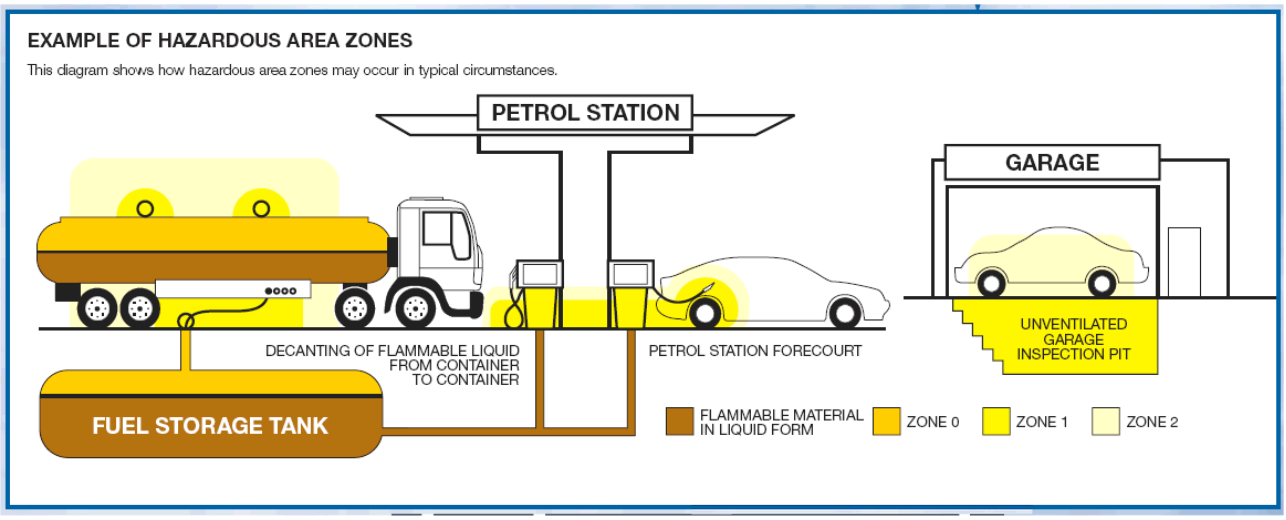


Figure 58.8: Examples of hazardous area zones (www.wolf-safety.co.uk 12-DEC-09).

- intrinsic safe: Ex ia for zone 0 and Ex ib for zone 1; limit the amount of energy stored on the device to limit the surface temperature and/or sparks from the electrical equipment. Any electrical equipment must be designed for the Ex ia or ib protection. Type "ia" protection allows for the occurence of two faults at a time during operation, while type "ib" protection only allow the occurence of one fault at a time during operation.

58.6 Temperature classes

Equipment installed within the Ex area is divided into six temperature classes (T1 to T6), where T1 is the highest temperature. These temperature classes are not the operating temperature range of the equipment, but the maximum permissible surface temperature of the equipment. The maximum surface temperature of the equipment must remain below the ignition temperature of the surrounding medium at all times. The temperature classes are:

Class	°C	°F
T1	450	842
T2	300	572
T3	200	392
T4	135	275
T5	100	212
T6	85	185

Note that the temperature should be as low as possible, meaning that T6 is the best temperature class.

58.7 Intrinsic safety

Intrinsic safety (IS) is a low energy signaling technique that prevents explosions from occurring by ensuring that the energy transferred to a hazardous area is well below the energy required to initiate an explosion. This concept was developed for safe operation of process control instrumentation in hazardous areas. Appropriate intrinsically safe apparatus can be used in all zones and the only solution for zone 0 instrumentation.

The theory behind intrinsic safety is to ensure that the available electrical and thermal energy in the system is always low enough that ignition of the hazardous atmosphere cannot occur. The two mechanisms being consider that can initiate an explosion are sparks or hot surfaces. The reduction of energy is achieved by ensuring that only low energy enter the hazardous area, and that all electrical supply and signal wires are protected by safety barriers. The types of safety barrier are zener diode

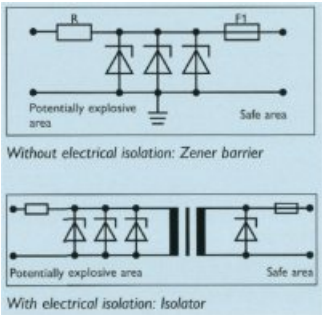


Figure 58.9: The principles for safety barriers without electrical isolation at top, and with electrical isolation at buttom (Pho 2010).

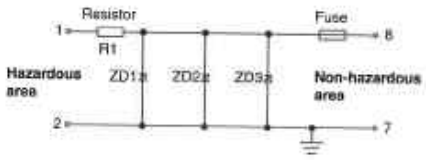


Figure 58.10: An example of a zener diode type of safety barrier (Klatt 2006).

based or galvanic isolation based, the working principles for these types are shown in Figure 58.9.

An example of a zener diode barrier type of safety barrier is shown in Figure 58.10 consisting of a resistor, fuse and at least three zener diodes. Note that a zener based barrier must be grounded to earth.

The fuse is protecting the zener diodes. A zener diode will conduct if the voltage exceed the zener diode voltage. If the input voltage increases above zener diode voltage, the zener diode conducts and the fuse may blow depending on the voltage. If the fuse is blown, the zener diode barrier needs to be replaced. The maximum output voltage can not exceed the voltage limit of the zener diodes (V_Z). The maximum output current is restricted by the resistor and can not exceed $\frac{V_Z}{R}$.

An example of a galvanic isolation barrier type of safety barrier is shown in Figure 58.11. A galvanic isolator is an active device, as the device requires external powering, that limits the energy into hazardous area without the dependence on the eath ground of the barrier. Galvanic isolators also has the advantage of supplying higher voltage at the hazardous area terminals and allowing longer cable lengths than zener diode based barriers.

The available electrical and thermal energy in the system must always be low enough so that no ignition can occur no matter of the type on error in the equipment. Note that these equipments have to be approved by a certificate like the device to be located into the hazardous area.

The usage of safety barrier is shown in Figure 58.12. The safety barrier is located in the non-hazardous area and is protecting the total amount of energy avaialble in the hazardous area.

Figure 58.13 illustrates a typical intrinsically safe system where the safety is depending on all the equipment in the system. In most process controll applications each equipment in the system is individ-

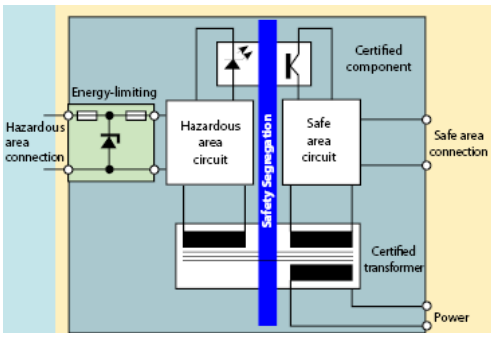


Figure 58.11: An example of a optical galvanic isolation type of safety barrier (www.iceweb.com.au: oct-16).

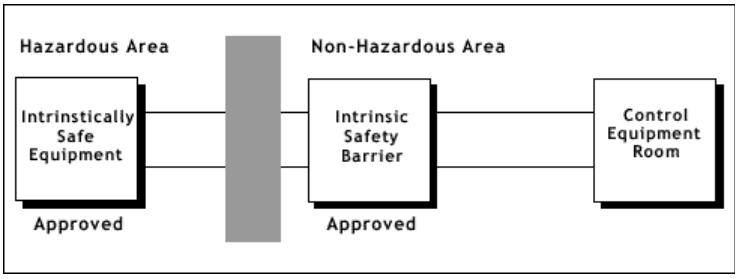


Figure 58.12: The usage of safety barrier to reduce or limit the amount of energy to the IS equipment.

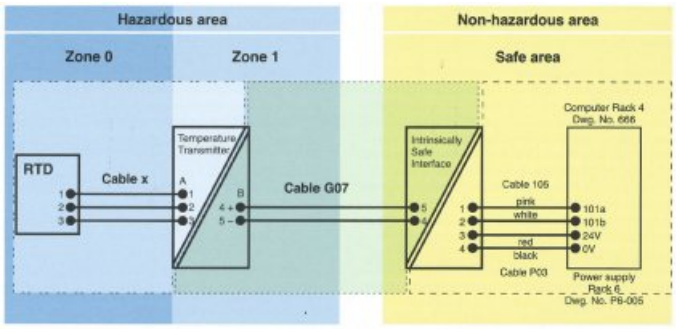


Figure 58.13: An example of an intrinsic safety system (Crouse-Hinds 2014).

ually certified. A document confirming the safety of the whole system is then based on the certification of each equipment.

IS equipment is composed of IS circuits designed in such a way that they neither give sparks energetic enough to ignite an explosion, nor do they consume power high enough to heat surfaces to a dangerous level. As an IS circuit is safe by design, in principle no additional protection is needed. The barriers will mainly be for protection of the signals between the equipment in hazardous area and connections in safe area (non-hazardous area).

IS circuits must also be provided with clearly deifined input and output parameters, allowing the user to calculate which combinations of other circuits are acceptable, and which are not. All inputs and outputs will affect the interconnecting systems with capacitance and inductance.

The power consumption must be kept under control when designing IS circuits, even with the usage of barriers. Components like capacitors and inductors may store energy sufficient to give sparks. All nodes in the schematics must be analyzed using the worst case scenario, and any components storing too much energy must be changed, either reduce the value or include a resistor. IS circuits of level *ia* (zone 0) must be explosion proof with up to two failures at the same time, while IS circuits of level *ib* (zone 1) will be explosion proof with one one failure at the same time. Level *ic* (zone 2) must be explosion proof during normal operation, meaning that the circuit will not be heated or store too much energy during normal operation.

In a cabinet will the cable trays containing cables for the hazardous area very often be blue. It is not allow to mix cabling from the hazardous area and non-hazardous area (safe area).

58.7.1 Design of simple IS systems

Most of the intrinsically safe systems designed are simple systems, containing a single source of power and a single piece of IS equipement in the hazardous area. The first step is to obtain the safety data of the two pieces of equipement. This safety data must be avialable on the certificate which must be available when designing such systems. A design example from (Crouse-Hinds 2014) is used as an example of a simple IS system. The example is shown in Figure 58.14.

The design will be based on the compatibility of the equipments and should be performed by comparing the data of each equipments (with connection to the hazardous area). The design steps are normally (Crouse-Hinds 2014):

a) compare the levels of protection. If they differ, the system takes the least sensitive level,

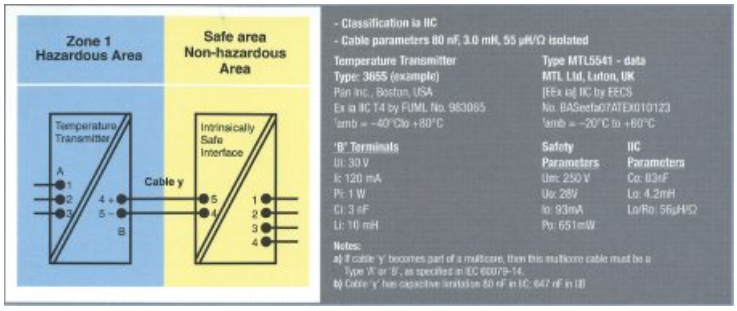


Figure 58.14: A simple IS system ofinterface and transmitter.

Sequence step	Parameter	Interface	Temperature transmitter	System
a)	Level of protection	ia	ia	ia
b)	Gas group	IIC	IIC	IIC
c)	Temperature classification	T4		
d)	Ambient temperature	- 20°C to +60°C	- 40°C to +80°C	
e)	Parameter comparison			
	Voltage	Uo: 28V	Ui: 30V	
	Current	Io: 93mA	Ii: 120mA	
	Power	Po: 650mW	Pi: 1W	
f)	Cable parameters			
	Capacitance	Co: 83nF	Ci: 3nF	Cc: 80nF
	Inductance	Lo: 3.05mH	Li: 10µH	Lc: 3mH
	L/R ratio	Lo/Ro: 55µH/Ω	Lo/Rc: 55µH/Ω	
g)	Isolation	isolated	isolated	isolated

Figure 58.15: The analysis for the design of a simple IS system for Figure 58.14.

b) compare gas classification. If they differ, the system takes the keast sensitive classification,

c) Determine the temperature classification of the field mounted equipment (hazardous area),

d) The permissible ambient temperature range of each equipment should be noted,

e) The output parameters of the source power (voltage U_o, current I_o, and power P_o) should be compared with the input parameters (U_i, I_i, P_i) for the field device. The output parameters should NOT exceed the relevant input parameters.

f) Determine the permitted cable parameters. 1) The permitted cable capacitance (C_c) is derived by subtracting the input capacitance of the field device (C_i) from the permittet output capacitance of the source power (C_o), giving:

$$C_c = C_o - C_i$$

2) The permitted cable inductance (L_c) is derived by subtracting the input inductance of the field device (L_i) from the permittet output inductance of the source power (L_o), giving:

$$L_c = L_o - L_i$$

3) Determining the the permitted $\frac{L}{R}$ ratio of the cable ($\frac{L_c}{R_c}$).

g) Check for an acceptable level of isulation from earth.

A good way to make this design analysis is to create a table with the analysis information. The design analysis of the simple IS system in Figure 58.14 is shown as a table in Figure 58.15

58.7.2 Maintenance, repair and test of IS systems

IS equipment is approved equipment with a document. This is safety equipment and should only be repaired or tested by trained personell.

58.7.3 IS systems standards

The Intrinsic safety standards world wide are:

- International; IECEx; IEC60079 − 0, IEC60079 − 11, IEC61241 − 11,

Physikalisch-Technische Bundesanstalt

Braunschweig und Berlin

PTB

(1) EC-TYPE-EXAMINATION CERTIFICATE

(Translation)

(2) Equipment and Protective Systems Intended for Use in Potentially Explosive Atmospheres - Directive 94/9/EC

(3) EC-type-examination Certificate Number:

PTB 98 ATEX 2210

(4) Equipment: Valve control unit type FDO-VC-Ext.PA***

(5) Manufacturer: Pepperl + Fuchs GmbH

(6) Address: Königsberger Allee 87, D-68307 Mannheim

(7) This equipment and any acceptable variation thereto are specified in the schedule to this certificate and the documents therein referred to.

(8) The Physikalisch-Technische Bundesanstalt, notified body No. 0102 in accordance with Article 9 of the Council Directive 94/9/EC of 23 March 1994, certifies that this equipment has been found to comply with the Essential Health and Safety Requirements relating to the design and construction of equipment and protective systems intended for use in potentially explosive atmospheres, given in Annex II to the Directive.

The examination and test results are recorded in the confidential report PTB Ex 98-28034.

(9) Compliance with the Essential Health and Safety Requirements has been assured by compliance with:

EN 50014:1997 EN 50020:1994

(10) If the sign "X" is placed after the certificate number, it indicates that the equipment is subject to special conditions for safe use specified in the schedule to this certificate.

(11) This EC-type-examination Certificate relates only to the design and construction of the specified equipment in accordance with Directive 94/9/EC. Further requirements of this Directive apply to the manufacture and supply of this equipment.

(12) The marking of the equipment shall include the following:

Ex II (1) 2 G EEx ia IIC T4

Zertifizierungsstelle Explosionsschutz

By order:

Braunschweig, December 18, 1998

(signature)

Dr.-Ing. U. Johannsmeyer

Regierungsdirektor

sheet 1/5

EC-type-examination Certificates without signature and official stamp shall not be valid. The certificates may be circulated only without alteration. Extracts or alterations are subject to approval by the Physikalisch-Technische Bundesanstalt.

In case of dispute, the German text shall prevail.

Physikalisch-Technische Bundesanstalt • Bundesallee 100 • D-38116 Braunschweig

Figure 58.16: A standard certificate for an intrinsic safe device, marked with the Ex code EEx ia IIC T4 (Klatt 2006).

- Europe; ATEX; EN60079 − 0, EN60079 − 11, EN61241 − 11,
- North America; FM; FM3600, FM3610, FM3810,
- North America; CSA; C22.2 No. 157

58.8 Cetificate

All devices that will be located into explosive atmoshere must be approved and will have a certificate stating the Ex code. This Ex code must also be availabe on the device. An example of an Ex certificate is shown in Figure 58.16. Note that these certificates are very specific and will be valid for only one type of device, with a very specific device number og device description.

The same Ex code from the certificate has also the be marked on the devices. The structure of these Ex codes (labels) is shown in Figure 58.17.

58.9 Process

An example of a plant is shown in Figure 58.18 combining communication in both safe area and zone 1. Different types of protections are used for the communication lines between safe area and zone 1, like safety barrier, surge protectors and isolators.

Products to be mounted in the hazardous area can be approved either under the LOOP or ENTITY approval concept (www.omega.com: AUG-08).

1. The LOOP concept specifies the exact part number and products that can be used in the loop. No deviation from the specified units is allowed.
2. The ENTITY concept specifies parameters which any approved intrinsic safety barrier must meet. This allows the user to select barriers from different approved manufacturers. Under entity approval two items may be interconnected if the following conditions are met, shown in the table below (www.omega.com: AUG-08).

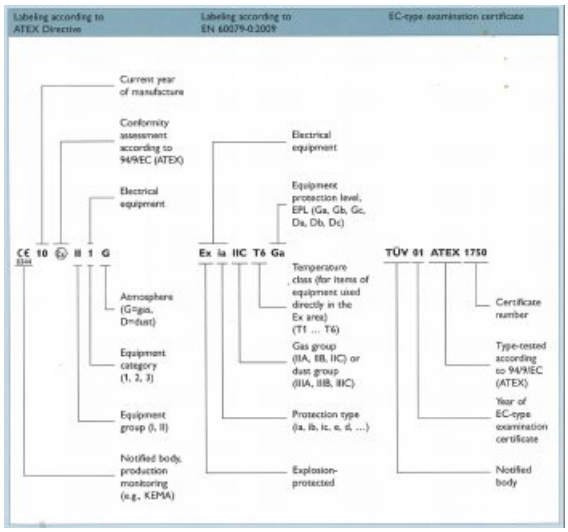


Figure 58.17: The structure of the Ex code or Ex label (Pho 2010).

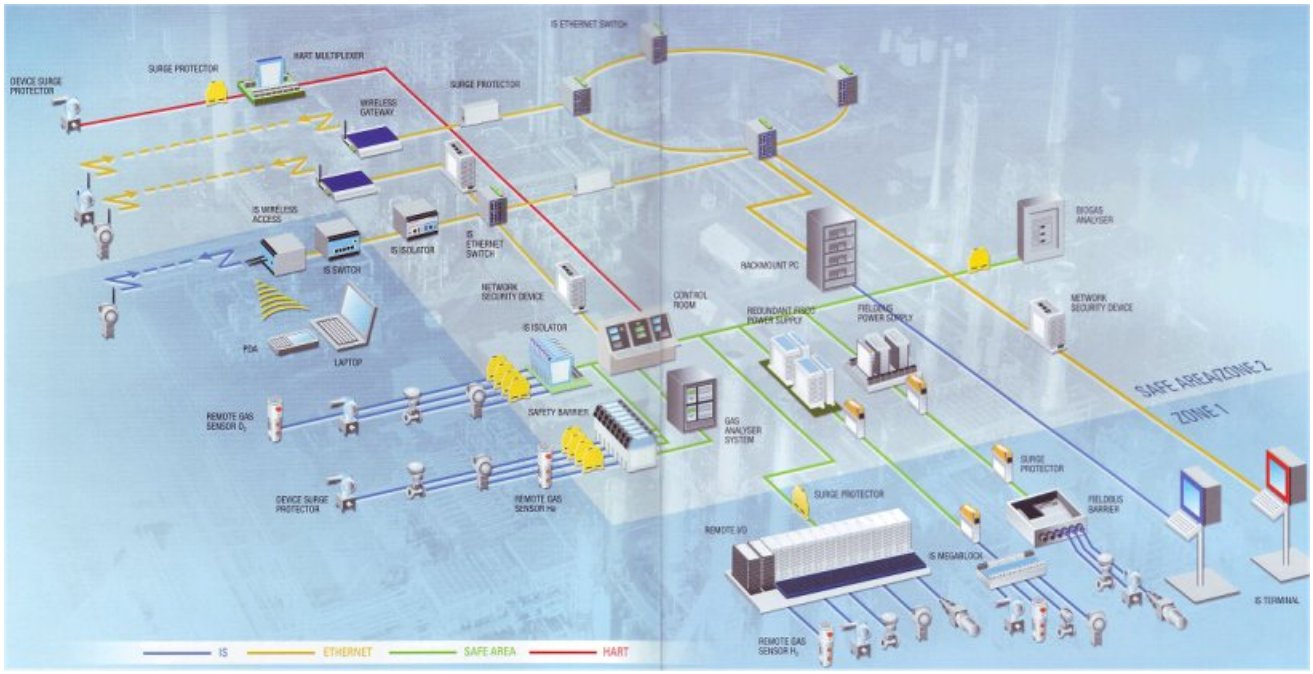


Figure 58.18: A process system combining safe area and zone 1, and using Intrinsically Safe (IS) in Zone1 (www.mtl-inst.com: FEB-10).

HAZARDOUS AREA		**NON-HAZARDOUS (SAFE) AREA**
Intrinsically Safe Approved Apparatus	Must be	Intrinsically Safe Barrier
$V_{\max}$	less than	V_{OC}
$I_{\max}$	less than	I_{SC}
L_a	greater than	$L_i + L_w$
C_a	greater than	$C_i + C_w$

Where:

$V_{\max}$	=	Maximum Open Circuit Voltage
$I_{\max}$	=	Maximum Short Circuit Current
L_i	=	Maximum Unprotected Inductance
C_i	=	Maximum Unprotected Capacitance
V_{OC}	=	Maximum Open Circuit Voltage (barrier)
I_{SC}	=	Maximum Short Circuit Current (barrier)
L_a	=	Maximum Allowed Inductance (barrier)
C_a	=	Maximum Allowed Capacitance (barrier)
L_w	=	Inductance of interconnecting wiring
C_w	=	Capacitance of interconnecting wiring

In all cases the intrinsically safe barriers and equipment MUST be wired per an approved drawing. Capacitance and inductance of the wiring and cables must be included in the loop evaluation.

The regulation of the equipment in hazardous areas is in Europe controlled by the ATEX[1] directives and guidelines.

58.10 Ingress Protection (IP) Ratings

IP rating is used to specify the environmental protection for electrical enclosure in electrical equipment. Ingress Protection (IP) ratings are developed by the European Committee for Electro Technical Standardization (CENELEC) (NEMA IEC 60529 Degrees of Protection Provided by Enclosures - IP Code), specifying the environmental protection the enclosure provides. IP is another way of protecting equipment in hazardous areas. The main concern for the IP protection is ingress protection of dust and liquid (water).

58.11 IP coding

The IP rating normally has two numbers (IP xy):

- First number: Protection from solid objects or materials,
- Second number: Protection from liquids (water).

The first number of the IP rating: Protection against solid objects:

[1] ATEX: ATmosphères Explosibles

IP First code	Degree of protection - solid object
0	No special protection
1	Protected against solid objects up to 50 mm, e.g. accidental touch by persons hands.
2	Protected against solid objects up to 12 mm, e.g. persons fingers.
3	Protected against solid objects over 2.5 mm (tools and wires).
4	Protected against solid objects over 1 mm (tools, wires, and small wires).
5	Protected against dust limited ingress (no harmful deposit).
6	Totally protected against dust.

The second number of the IP rating: Protection against liquid:

IP Second code	Degree of protection - liquid
0	No protection
1	Protection against vertically falling drops of water e.g. condensation.
2	Protection against direct sprays of water up to an angle of 15 degree.
3	Protected against direct sprays of water up to an angle of 60 degree.
4	Protection against water sprayed from all directions - limited ingress permitted.
5	Protected against low pressure jets of water from all directions - limited ingress.
6	Protected against temporary flooding of water - limited ingress permitted
7	Protected against the effect of immersion between 15 cm and 1 m.
8	Protects against long periods of immersion under pressure.

Example: a protection of IP 61 means totally protected against dust and direct water sprays of an angle up to 15 degree.

Part XVI

Testing

Chapter 59

Testing

59.1 Introduction

A test case in hardware engineering, software engineering or system engineering is a set of conditions or variables under which a tester will determine whether an application or software system meets the requirements. The requirements will be based on a set of specifications, however the requirements must be testable. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle. In some settings an oracle could be only one requirement or one use case, but it normally takes many test cases to determine whether a hardware device, a software application or a system is functioning correctly. Test cases are often referred to as test scripts, particularly when written, in the literature. The written test cases are usually collected into test suites.

Testing means often validation of several combination of hardware and software functions, defined by a set of requirements. Testing of the hardware and the software can be a challenge, and will consist of several different type of tests. One example can be as shown in Figure 59.1 where different tests will validate the functions depending on the design. The model shown in the figure is also referred to as the V model in some literature.

Figure 59.1 focus on software projects, but the same live cycle, or model, is also valid for hardware and system development. In such a case will the "coding" block be implementation. Testing a system means to check that the requirements are fulfilled by adding a set of inputs and checking the corresponding outputs of the system. Testing must be done in a structured way by defining the combinations of input signals, and checking the related output signals. This structure must be defined in a document, named a test plan.

Testing can be manually, partly automated or fully automated. Manually testing and partly automated means that there must be a human in the test loop doing the testing. The testing operation is to check if a system is fulfilling the requirements, but can also be to check under which condition will the system stop working, finding the working boundaries for the system. The system is affected by nature and nature can often give a set of random conditions in different conditions so knowing the system boundaries should be part of any testing. A fully automated testing will often use tools like hardware in the loop (HIL) or digital twins. The type of process will also affect the testing, the focus in this document will be continuos processes.

59.2 Test Plan

A test plan is a systematic approach to test a system such as a machine or software, and must include the test coverage, the test methods, and the test responsibilities. The plan typically contains a detailed understanding of what the workflow will be. The test coverage defines what will be tested, test methods describes how each sections of the test coverage will be tested, and test responsibility assign the tasks and responsibilities. The test plan should contains at least three parts. Part one is describes the system to test, part two includes a detailed list of the test to be performed on this system, and part three as section to be signed by any testers. The test plan should also contain a PASS column to be marked by the tester, for every test. The test plan should also contain a comment field for any comments from the tester(s).

A test plan will contain:

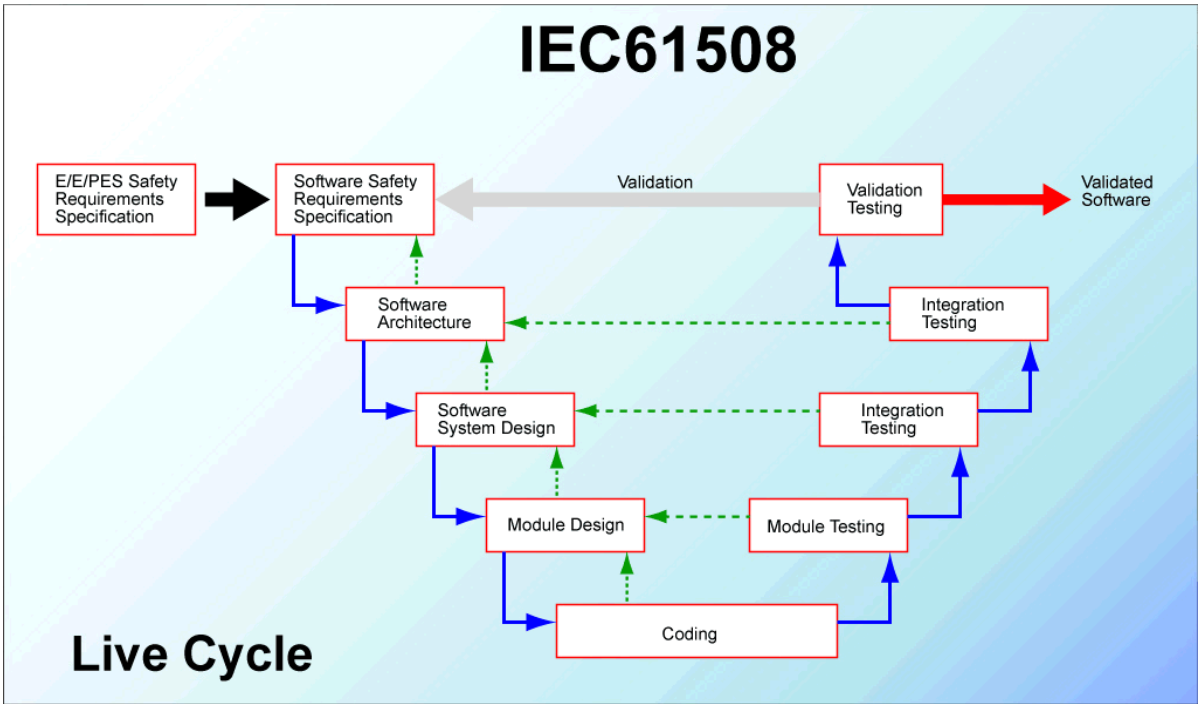


Figure 59.1: The life cycle of a software product according to the IEC61508 safety standard.

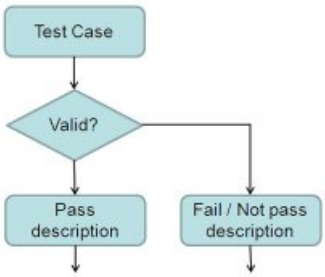


Figure 59.2: The process flow for a test case.

- system description and diagram; devices used and how the devices were interconnected. This part should describe in detailed how the system is used with all inputs and outputs defined. Any simulators/emulators should also be included, including any test databases used.
- test lists; test cases with pass and comment sections.
- responsibility section, stating who is doing the testing with a field to sign.

The list of the tests should contain a set of test cases for testing each function and/or module of the system. The flow of a test case is shown in Figure 59.2.

A test plan should be a technical document and should contain at least the following chapters:

- introduction; the reason for the test plan,
- system description; description of the system and/or devices to test,
- results; the test plan consisting of a set of test cases. The test cases can be in a table with four columns; index, description, pass, and no pass or comment.
- conclusion; the test conclusion with the name and signatures of the testers.

The table of the test cases can be:

Id	Description	Pass	Comments
..			
5	Conversion from meter (m) to feet (ft)		
5.1	Convert 0.75 m to 2.461 ft	Fail	2.460, no round up.
5.2	Convert 1.55 m to 5.085 ft	OK	
5.3	Convert 9.86 m to 32.349 ft	OK	
..			

The test plan should contain detailed information in such a way that the tests can be redone with exact same results, meaning have the same input signals, combinations and results. Any results, OK or wrong, should be able to reconstruct based on the test plan. Why? One reason is to let the developer be able to have the same error(s) when trying to fix the error. Another reason is to be able to test the same error(s) after it has been corrected, how to be sure that the error(s) have been corrected?

Test cases are dealing with an input, an action and an output. The action acts upon the input to derive a specific output. A test case must be specific by defining the inputs, action and any outputs, and the outcome should be OK or an error.

How can the following software method be tested?

```
public int CheckTypeStep(int val)
{
  if (val > 10)
  {
    ....
    val = 1 ;
  }
  else if (val < -10)
  {
    ....
    val = -1 ;
  }
  else
  {
    ....
    val = 0 ;
  }
  return val ;
}
```

Will this test case be a valid test of the function?

Id	Description	Pass	Comments
7	Testing of CheckTypeStep() function	____	
7.1	Input value = 23, Output value = 1	OK	

Will these test cases be a valid test of the function?

Id	Description	Pass	Comments
7	Testing of CheckTypeStep() function	____	
7.1	Input value = 18, Output value = 1	OK	
7.2	Input value = 5, Output value = 0	OK	
7.3	Input value = -18, Output value = -1	OK	

What will be the minimum number of tests or test cases?

The software method contains three *if* - *else* blocks, the minimum will be to have one test of each block. Sometimes the limits are important meaning that the minimum number of test cases should be

increased to five.

What should the test cases be?

Good values can be −11, 11 and 0, however depending on the data type.

An example of a good test case:

Id	Description	Pass	Comments
7	Testing of CheckTypeStep() function	____	
7.1	Input value = 11, Output value = 1	OK	
7.2	Input value = 10, Output value = 0	OK	
7.3	Input value = 0, Output value = 0	OK	
7.4	Input value = 10, Output value = 0	OK	
7.5	Input value = −11, Output value = −1	OK	
7.6	Input value = −23456, Output value = −1	OK	

All three blocks have been tested, both limitations of the if tests have been tested, the special case with 0 has been tested, and a large number has been tested. The large number can be either positive or negative, or both. A good advice is to test at least one value "for out" related to the "normal data" range.

An example of a small test plan is shown in the appendix. This is a small overall system test with very few test cases. A test plan can be from a couple of pages, as shown in the appendix, up to several hundred of pages very depending of the complexity of the system. The time duration can be from a couple of days, as shown in the appendix, and up to several weeks.

A rule is to try to avoid that the testing is done by the developers. The developers should of course test when developing the system. When releasing a new software version or a new product or system, a test plan should document the testing and preferential be done by other people than the developers. Larger companies have their own test department and their own testers, however smaller companies may use the developers as testers as well. So in smaller companies a test plan is even more important than in larger companies with their own test department.

59.3 Test Levels

Testing of systems should always start with testing each sub system of the lowest level. Testing each sub device with a manageable size. When all sub devices are tested, sub systems will be interconnected new combinations must be tested. The number of test cases will increase when interconnecting or combining several sub devices.

59.3.1 Unit or Device Testing

In computer programming, unit testing is a method of testing that verifies the individual units of source code are working properly. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual program, function, procedure, etc., while in object-oriented programming, the smallest unit is a method, which may belong to a base/super class, abstract class or derived/child class.

Unit testing is also valid for devices, testing the hardware functions of each sub device. Unit testing is a type of white box testing as we need to know the working details inside the unit or device (box) to be able to test the box.

59.3.2 Module testing

Module testing (sometimes called Integration and Testing, I&T) is the phase of software testing in which individual software modules are combined and tested as a group. It follows unit testing and precedes system testing.

Module testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. Module testing is based on integration and is a type of grey box testing as we need to have some information about both the working details inside and the I/O information of the modules.

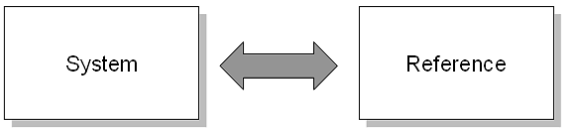


Figure 59.3: A system is tested using a reference system.

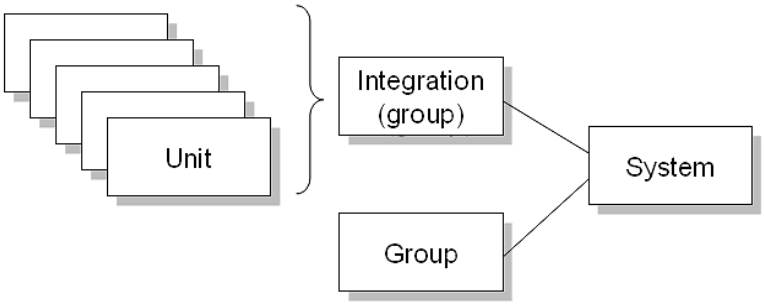


Figure 59.4: A system is tested by testing each unit in the system. A unit can either be a either a software unit or a hardware device.

59.3.3 System Testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. Figure 59.3 shows how a system can be tested using a reference system.

As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limiting type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole. Figure 59.4 shows how a system can be tested by testing each unit in the system.

System testing is also called black box testing as the focus is interconnections and combinations, not the "functions" of each module.

59.4 Factory Acceptance Testing (FAT)

In engineering and its various subdisciplines, acceptance testing is black-box testing performed on a system (e.g. software, lots of manufactured mechanical parts, or batches of chemical products) prior to its delivery. In some engineering subdisciplines, it is known as Functional testing, black-box testing, release acceptance, QA testing, application testing, confidence testing, final testing, validation testing, usability testing, or factory acceptance testing.

In most environments, acceptance testing by the system provider is distinguished from acceptance testing by the customer (the user or client) prior to accepting transfer of ownership. In such environments, acceptance testing performed by the customer is known as beta testing, user acceptance testing (UAT), end user testing, site (acceptance) testing, or field (acceptance) testing.

Figure 59.5 shows an overview of the testing process, from unit testing to FAT. FAT is often a system testing with a specific system configuration.

Be aware that a FAT is only testing a system regarding a specific set of specification, it is not a system test. Normally a system test is more comprehensive than a FAT. The system testing is more a more general test, while the FAT is more specific test.

59.4.1 Types of acceptance testing

Typical types of acceptance testing include the following:

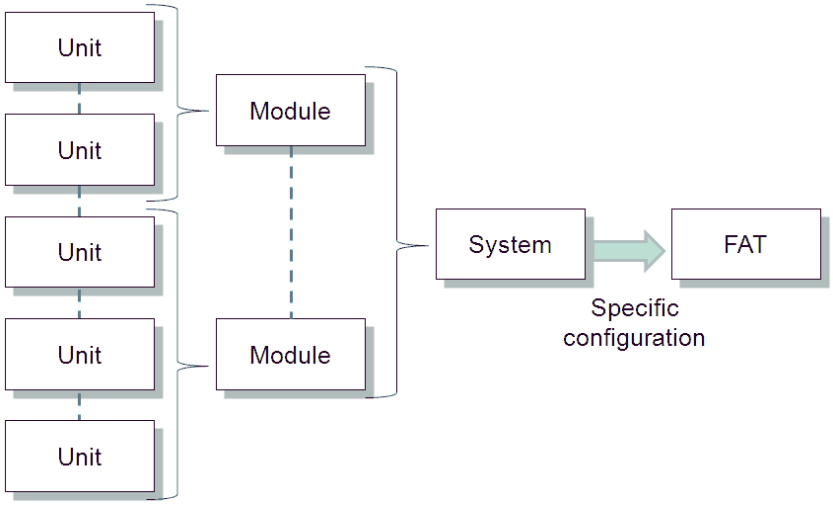


Figure 59.5: An overview of the total testing process, from unit testing to FAT. Note that the units can be either a software unit or a hardware device.

1. User acceptance testing; This may include factory acceptance testing, i.e. the testing done by factory users before the factory is moved to its own site, after which site acceptance testing may be performed by the users at the site.

2. Operational acceptance testing; Also known as operational readiness testing, this refers to the checking done to a system to ensure that processes and procedures are in place to allow the system to be used and maintained. This may include checks done to back-up facilities, procedures for disaster recovery, training for end users, maintenance procedures, and security procedures.

3. Contract and regulation acceptance testing; In contract acceptance testing, a system is tested against acceptance criteria as documented in a contract, before the system is accepted. In regulation acceptance testing, a system is tested to ensure it meets governmental, legal and safety standards.

4. Alpha and beta testing; Alpha testing takes place at developers' sites, and involves testing of the operational system by internal staff, before it is released to external customers. Beta testing takes place at customers' sites, and involves testing by a group of customers who use the system at their own locations and provide feedback, before the system is released to other customers. The latter is often called "field testing".

59.5 Positive or Negative Testing

Positive testing attempts to show that a given unit, module or system is fulfilling the requirements for the system. Negative testing attempts to show that a given unit, module or system is NOT fulfilling the requirements for the system. Negative testing may often consider to be boundary analysis, checking that the system is handling any boundary challenges.

Most syntax and input validation tests will be positive testing as the testing is looking for positive results. Even with an invalid input the test case should indicate a positive results like an error message or a warning. Negative testing means to try to let the module do something differently than what the module was designed to do. It is the thinking behind the test that should be positive or negative.

The developer very often end up with positive testing while a separate test department often end up with negative testing.

59.6 Accelerated testing

In order to verify the MTBF (Mean Time Between Failures) numbers may take a long time, often years, based on the product type and volume. Because this process is so slow and costly, the accelerated testing has been established. There are two type of accelerated testing:

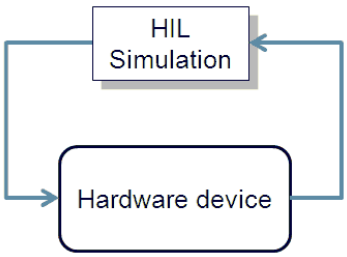


Figure 59.6: HIL simulation in testing a hardware device. The HIL simulation is simulating the process input signals to the hardware device based on a process model and the output(s) from the hardware device.

1. simulation; also known as accelerated life testing (ALT), the life cycle is accelerated. Use time compression with higher stress.
2. stimulation; also known as accelerated stress testing (AST), high amount of "stress" for a short period of time should exhibit the same failure mechanisms as it would at lower stress levels for a longer period of time.

59.7 Testing tools

Testing is often time consuming tasks and computer based tools should be used for automation of the process. Simulation or emulation of the part of the system or the external system can be a good solution.

59.7.1 Simulation

Simulation is the imitation of some real thing, state of affairs, or process. Simulation will be to make a model (software) of part of the system or a part of the external "world" in software. The software model will describe the important behavior of the "input" process for the system to be tested, and should cover most of the ranges for the input parameters, and any combination of these parameters. The model should be able to represent certain key characteristics or behaviors of a selected physical or abstract system.

59.7.2 Emulation

Emulation is the process of imitating the behavior of one or more pieces of hardware (typically a system under design) with another piece of hardware, typically a special purpose emulation system. The emulation will be hardware and will be an complete and/or exact representation of a selected physical or abstract system.

59.7.3 HIL

Hardware-in-the-loop (HIL) simulation is a technique that is used in the development and test of complex real-time and/or embedded systems. HIL simulation provides an effective platform by adding the complexity of the plant under control to the test platform. The complexity of the plant under control is included in test and development by adding a mathematical representation of all related dynamic systems. These mathematical representations are referred to as the "plant simulation".

The principle is based on mechanistic simulation model of the system that should be monitored and/or controlled. The simulation model will be connected to the device or system to be tested. The simulation model must be developed in such a way that the system or device to be tested is not able to make any difference between the real-world system and the simulated system. This way can the test be done at an early state of the development and without the need of the real-world system. Figure 59.6 shows a HIL simulation used for testing a hardware device. The HIL simulator will test the hardware device by making a set of input signals depending on the process model and the hardware device output signals, and validate the output signals from the hardware device as well.

HIL can be used to first test every sub devices separately and then test the complete system.

The advantages of HIL are:

1. reduce the down time of the system when installed,
2. reduce the time for commissioning and periodic tests,
3. less destructive testing,
4. better testing of a complete system.

Note that the HIL Simulation module also must contain any documentation module(s) as well for documenting the testing. In a real-time system any timing parameters must be included as well.

59.7.4 HALT

Highly accelerated life testing (HALT) is a testing methodology for provoking failures commonly seen after long term use. HALT cannot indicate how long the lifetime of a product will be, but the results can be used to increase the lifetime of the product. HALT is performed by combining vibration, high and low temperatures, and rapid temperature changes.

59.8 Software testing

Systems consists of hardware and software. The tendency of hardware modules is more standard modules, and add more functions into software. The quality of software will be important, and testing of system with focus on the software. Software testing:

- documentation;
  - analyses and design; for instance UML with use cases, use case documents, and class diagrams,
  - coding standards; define the way of coding, more easy to read the code,
  - coding; add comments for important parts of the code.
- compiler; errors and warnings,
- system errors (runtime error),
  - Try/Catch blocks,
  - Testing.
- Logical errors,
  - combinations,
  - logic,
  - testing.
- programming errors;
  - system errors,
  - logical errors.
- code inspection;
  - explain or walk trough of the code.

The challenge in software testing is to synchronize the code and any documentation.

One way of designing software is using test-driven design; based on the analysis phase the test cased can be defined first and then the software can be designed to solve these test cases. Another way of testing object oriented software is to include a test method in each class testing all other methods in the class. Using the same name of the test method in every class it is very easy to include a test class that can be used to test all other classes. The test class will only used during the test operations, not in normal operation. This way, the classes can very easy be tested when any changes has been performed.

Chapter 60

Maintenance

Maintenance in a system is important. The old way of doing this is to have fixed periods for the maintenance, today there will be a requirement for extended time periods for these fixed maintenance periods. Different types of equipment may require maintenance at different time intervals, and measurements should be included to check/estimate/verify when maintenance is necessary. Especially moving or rotating machinery may require a higher degree of maintenance than stationary systems.

Predictive maintenance, also called "online monitoring", "condition based monitoring", or "risk based maintenance"(Hashemian & Bean 2011) has a long history in the process industry. IoT (Internet of Things) devices can be used to get more information to make a better predictive maintenance.

60.1 Condition Monitoring

Condition monitoring (CM) is due to the increasing demands on productivity and machinery. Condition monitoring strategy is taking over from preventive or reactive strategies, to optimize time and cost for maintenance. Conditional monitoring is using information from sensors, often critical information, to estimate the usage of the system and the need for maintenance. Information from the sensors, often on the machinery or other systems, is transferred to a maintenance planning software. In the event of a problem, or regular maintenance, can then be solved during a planned maintenance period. Any problem like wear of mechanical parts or exceeding temperature limits can be solved by making an order of the software, and an action can be taken before the breakdown of a machine.

Figure 60.1 shows the ideal case for optimum cost on maintenance. The Figure also show the reactive strategy period and the preventive strategy period.

Rotating machinery will always require some maintenance and the maintenance should be planned. The old way of doing it is according to the calender, at fixed interval. Older rotating machinery will require maintenance more often, but how much? Condition monitoring can be based on manual measurements, but online measurements is best. These measurements should be included in the process control system to give a warning before the system is stopped because of an error with any of the rotating machinery. A sensor like a microphone or an accelerometer can be used to record a signal that must be analyzed using methods like fast Fourier transform (FFT). A FFT will take signal with a mixture of frequencies and split it into a separate set of the basic frequency signals. Figure 60.2 shows how a measurement signal can be separated into its basic frequency components.

These frequency components has to be related to different parts of the rotating machinery. An analysis of each frequency component can be performed to monitor the state of each part of the rotating machinery. If any of the signals has an average value over or below a limit, an warning can be given for requesting maintenance If a signal starts increasing or decreasing too fast or too slow, that may also indicate the requirement for maintenance The rate of change, derivative, may also be an indication of the need for maintenance. Figure 60.3 shows an example of the relation between the frequency components and parts of a rotating machinery. The mass and/or rotating speed of each component often decide these frequency components.

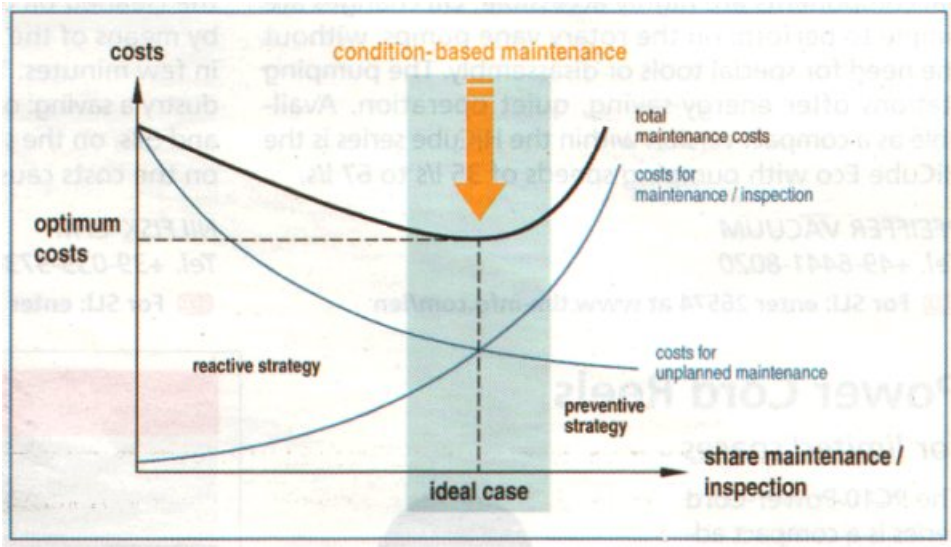


Figure 60.1: Condition based maintenance for minimizing the cost (www.ifm-consulting.com: apr-09).

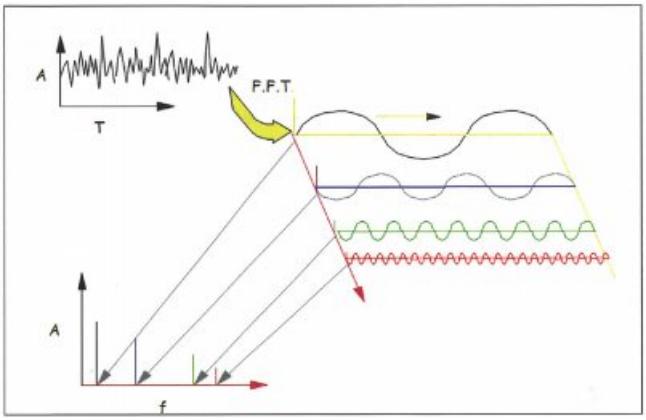


Figure 60.2: The usage of FFT to split a measured signal into set set of separate basic frequency signals.

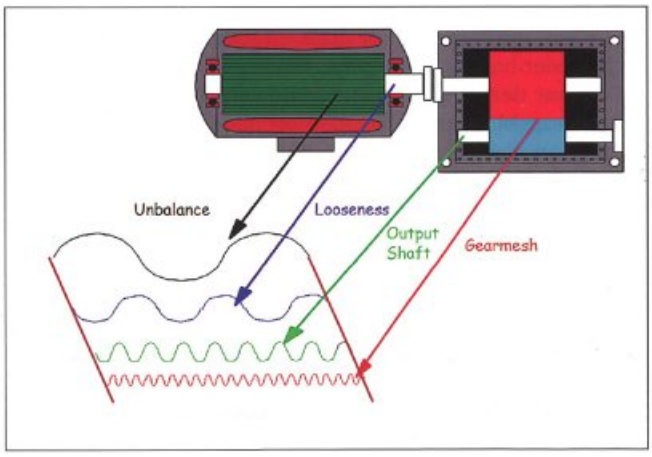


Figure 60.3: An example of the relation between the frequency components and the rotating machinery.

60.2 Predictive maintenance

"Maintenance is the combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function" (Standards 2010) and exists of several maintenance management group strategies. The most used are (Carvalho, Soares, Vita, da P. Francisco, Basto & Alcala 2019):

- Corrective maintenance (CM) or Run to failure (R2F); maintenance when the equipment stops working,
- Preventive maintenance (PvM); time based or scheduled maintenance performed periodically with a planned schedule,
- Predictive maintenance (PdM); use of predictive tools to determine when maintenance actions are necessary.

Companies have to understand exactly what they want to achieve first before introducing predictive maintenance or condition monitoring. This is often the first step to an effective asset management and a strategy is necessary. The strategy will make it possible to identify any potential asset failures and plans for repairs or replacement to minimize the impact of production. This strategy should include 1) make a general view of the process and the needs in the process, 2) be realistic with focus on costs and downtime for the process and focus on small and achievable goals. Expand gradually. 3) flexibility is the key as businesses have different needs, evaluate and implement a number of different solutions without focusing too much on the cost. The cheapest solution is rarely the most effective. 4) focus on known failures within the last five or ten years and try to identify patterns and/or common factors across these failures. 5) effective data handling by a data management system to get as much information as possible. Machine learning is one way of solving the predictive maintenance in a more effective way (Carvalho et al. 2019).

Equipment degradation can be detected before faults occur, reducing downtime, cutting costs and improving safety. With relevant data held by repositories, either locally or in the cloud, equipment can be analyzed, based on the data, helping to map failure patterns, failure modes and equipment performance. Failure mode and effects analysis (FMEA) is a structured way of approaching equipment failures and their possible causes (IEC60812 2006). Once the structured FMEA model is in place, the expected time from the first indication of a fault to actual failure can be calculated. Knowing the time, the maintenance activities can be planned. The process can be shifted to redundant equipment or the load can be reduced until the maintenance can be carried out.

60.2.1 Handling the data

The amount of raw data increases quickly with the number of monitored assets and will generate a high message traffic in the communication infrastructure. The high communication traffic will require a large amount of processing power and data storage. Some sort of edge computing should be consider to lower the message traffic. A sensor may generate more than 250 megabytes of raw data every day and much of this data may be irrelevant for understanding the trend of the machine health. Edge computer will process the data locally and reduce the message traffic by sending only processed information. This way the data sent may be reduced from hundreds of megabytes to a few kilobytes.

60.2.2 Machine Learning

The concept of machine learning (ML) is to develop a model that can use the data patterns to predict a future event. In predictive maintenance (PdM) a model is wanted that based on the current data can predict the remaining useful life of the components. The patterns will be any trends in the data set that can be used to estimate the degradation of a component. The process of using ML in PdM is shown in Figure 60.4; 1) gather data from all type of useful sources. The knowledge that you want out of the model must be in this data set. 2) preprocessing of the data from these sources.

You should check for outliers, missing samples and do some sort of filtering. Outliers should only be marked, not removed, as they may contain useful information for your PdM system. Missing samples should be adjusted to include information from all sources in each time interval. Do no try to include some default values as this may destroy the knowledge in the data. Always do some sort of filtering of external data, like low pass filtering. At the end, the data from the different data set should be merged

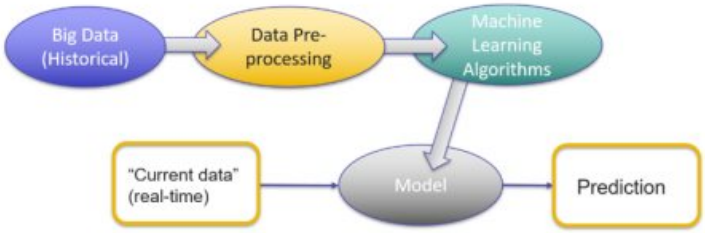


Figure 60.4: The process of using machine learning for making a model: 1) Data gathering (historical), 2) preprocessing of the data removing anomalies, combining and scaling, 3) choose a classification method and train a model based on the method and the data, 4) test the model with a test set, 5) use model for prediction based on current data.

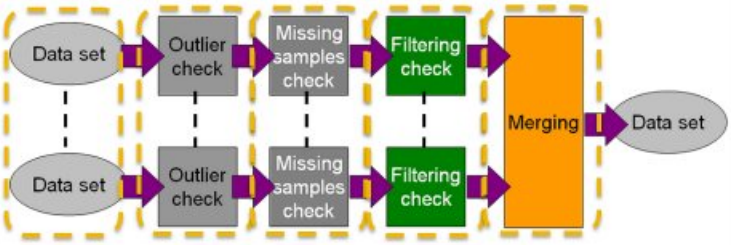


Figure 60.5: The pre-processing of the data sets; start with an outlier check, then a missing samples check and finish with a filtering action before merging all the different data into a new data set.

into a new common data set, as shown in Figure 60.5. The merging operation is alignment, scaling and centering of the data.

The data set should be divided into a training set, a calibration set if you need to adjust any hyper parameters in the model and a test set for testing the accuracy of the model. The data set is used to train, calibrate and test the model. The model is based on a classification algorithm and will use the data to train any classification parameters. There is three methods of training (or learning) a ML model, all based on the type of data that are available, are:

1. Supervised learning; All data sets contain responses so the algorithm knows the inputs and outputs for every sample,
2. Unsupervised learning; No responses in the data sets, only input data. The algorithm needs to find similarities between the inputs and group them,
3. Reinforcement learning; No responses in the data sets, only input data. However, the algorithm is informed when the answer is wrong, but not how to correct the answer.

In ML the challenge will be to select a good algorithm and have enough data, and it may be time consuming to develop a good model. However, using the model for PdM should be straight forward.

60.2.3 Challenges

The challenges implementing PdM:

- not enough data to make a predictive maintenance system; many PdM systems rely on machine learning algorithms so there need to be enough data to create an accurate model. Check your list of data sources, are your system collecting enough data? It is important to collect data also when the system is working ok, not only when a fault occur. The data includes both events, event codes, sensor values and time stamps. A model, like a digital twin, can also be used to generate synthetical data to combine both real data and synthetical data for predictive maintenance algorithms. Also make an analysis to understand which features that are important and which may be redundant.
- lack of failure data need for accurate results; failure data is a crucial part of teaching algorithms to recognize the warning signs for any just in time maintenance. These data may not exists if

maintenance is performed too often or the system can not be left to fail. One way to overcome this obstacle is to simulate the failure data. This means that a deep knowledge of the system (domain) and the components in the system is needed to include the behaviors into a model. Even if failure data might not be present, may the operations data show how a machine degrades over time. Principal component analysis is a statistical technique that can provide valuable insight into how the equipment operates over time.

- understand the failures, but not able to predict them; There is a significant difference between identifying what went wrong and knowing how to predict it. Start by defining the goals, and then think about how the predictive maintenance algorithm will affect these goals. Start with a minimum solution first, use the domain knowledge and choose a system that is deeply understood to practice on. Build a predictive maintenance algorithm based on the understanding of the features and factors affecting the performance of the system. Be sure to understand the solution and use the knowledge to dig into more complex systems. To further validate any models, add generated failure data similar to known historical conditions and test the system to build the confidence that the process is working.

- do not know how to do predictive maintenance; machine learning is a method that can be evaluated, but requires lot of data. Start by gathering data from multiple sources like databases, spreadsheets or web services. Make sure that the data is in the right format, including time stamps. If data comes from different sources, the data sets should be combined. Instead of feeding raw sensor data into machine learning models, it is common to extract features from the sensor data first. The data must be classified as healthy or faulty before the model can be trained, and use thresholds and estimate the remaining useful life of the components. A list of failure scenarios to predict will be useful to test the model.

Part XVII

Data Recovery

Chapter 61

Data Recovery Introduction

This part will focus on how to secure your data. The data of the SCADA system is of great value especially the historical data. This part contains information about RAID (Redundant Array of Independent Disks), UPS (Uninterruptedly Power Supply), and backup strategies. RAID is a technology in using several physical disk drives as one logical disk drive in order to achieve a higher level of performance and/or reliability. RAID used to achieve a higher level of reliability can be used for data recovery. Another solution is data backup where a copy of the data is duplicated on another physical device in order to have a copy of the data in case the disk drive is broken. Backup must be used on a regular basis in order to have the latest version of the data available on the backup device. UPS is used to have a more stable power supply to the computer systems. Modern computer systems will store part of the OS disk information in RAM for fast access toe the disk drives, and need to store this information on the disk drives before shutting the computer systems down.

61.1 Disk drives

Floppy drives and disk drives contain mechanical parts; a motor with rotating magnetic plates and a magnetic head for reading/writing the data on the magnetic plates. The magnetic heads are mounted on actuator arms moving the heads perpendicular on the magnetic plates. The motor will be rotating all the time and the actuator arms will be moving every time data is read or written to the disk. These operations will tear the disk drive. The rotating speed of disks are normally 3600 rotation per minute (rpm), 5400 rpm, 7200 rpm or 10000 rpm make the most of the disk noise. These mechanical parts are the main reasons for disk failures and the main reason for backup/recovery operations. The RAID technology can increase the reliability of disk drives by duplicating the disk drives. These parts are shown in Figure 61.1.

Floppy and disk drives are block based meaning that the smallest amount data that can be read or written is a block of data. These blocks of data are often 512, 1024 or 2048 bytes of data. The disk drive is divided into a set of logical blocks addressed by the logical block address (LBA). The physical disk drive stores data onto cylinders, heads and sectors. The sectors unit is the smallest size of data to be stored and often the same size as the block. The OS of the computer can however define a block size as

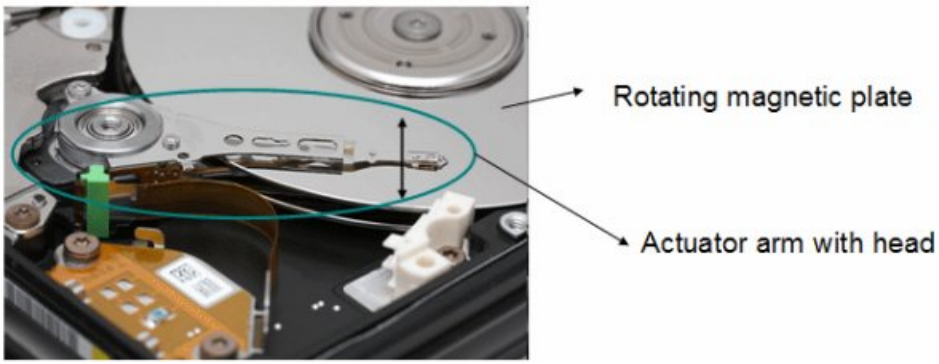


Figure 61.1: The inner of a disk drive with the rotating magnetic plate, the actuator arms with the head.

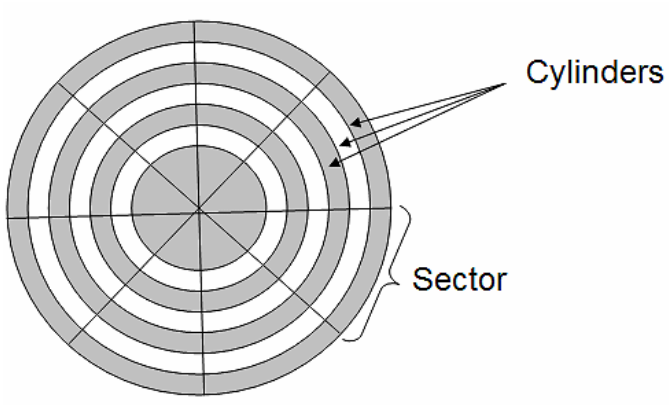


Figure 61.2: The cylinders and sectors of a disk plate. The size of the sector is fixed, giving a fixed number of sector for each cylinder.

a multiple of sectors, for example can a sector be of 512 bytes while the block size can be of 2048 bytes. The number of cylinders is the number of tracks on one magnetic plate, each magnetic plate will have tracks on both sides. This means that the disk must have an actuator arm with a head on both sides as well, shown in Figure 61.1. Each cylinder will be divided into a number of sectors, and given the number of cylinders, heads, sectors per cylinder, and bytes per sector, the number of bytes on the disk will be (shown in Figure 61.2):

$$Bytes = Cylinders \cdot Heads \cdot Sectors_per_Cylinder \cdot Bytes_per_Sector$$

A disk with 1024 cylinders, four heads (two plates), 64 sectors per cylinder and a sector size of 512 bytes has:

$$Bytes = 1024 \cdot 4 \cdot 64 \cdot 512 = 128MBytes$$

Reading or writing data, the disk must first position the head and then wait for the right sector. If the number of bytes is more than a block, the next sector can be at any location on the disk, very seldom it will be the next sector on the disk. The disk will use some time for the positions of the head and plate for every sector and the OS has to wait for the disk. The disk drives will also get fragmented over the time. This means that the data blocks belonging to each other (same file) will be randomly located on the disk and the time will increase when reading or writing the file. The defragmentation process is a process of moving the blocks to increase the time reading or writing the file.

61.2 File systems

The disks are block based meaning that all information will be divided into blocks before saving on the disks. The data system, the operating systems, need some sort of structure to find the corresponding blocks for a specific file. This structure is the file system and different operating systems have different structures or file systems. This will complicate the backup and recovery process. Windows based system may have the FAT, the FAT-32, or the NTFS file systems. MAC OS may have the HFS or the HFS+ file systems. Linux and Unix systems may have the UFS or the NFS to mention a few. Even USB flash drives and memory cards for cameras (SD-cards) are using file systems for storing the information. In a network with work stations with different file systems, which file system to select for any server?

The data recovery process needs to know the file system to be able to do a backup and especially data recovery. The file system must also be known to be able to retrieve data from corrupted disk drives, also known as volumes.

Chapter 62

RAID

62.1 Introduction

Redundant Array of Independent Disks (RAID) is a technology that employs the simultaneous use of two or more hard disk drives to achieve greater levels of performance, reliability, and/or larger data volume sizes (Watson 2002). RAID was first describe as Redundant Array of Inexpensive Disks because it allows computer users to achieve high levels of storage reliability from low-cost and less reliable PC-class disk-drive components, via the technique of arranging the devices into arrays for redundancy.

"RAID" is now used as an umbrella term for computer data storage schemes that can divide and replicate data among multiple hard disk drives. Hard disk drives are normally the slowest device in a computer system. RAID's various designs all involve two key design goals:

1. increased data reliability,
2. increased input/output performance.

When several physical disks are set up to use RAID technology, they are said to be in a RAID array. This array distributes data across several disks, but the array is seen by the computer user and operating system as one single disk drive. This is shown in Figure 62.1.

Redundancy is a way that extra data is written across the array, which are organized so that the failure of one (sometimes more) disks in the array will not result in loss of data. A failed disk drive may be replaced by a new disk drive, and the data on the new disk drive will be reconstructed from the remaining data and the extra data. A redundant array allows less data to be stored. The RAID technology can increase the performance by saving sectors belonging to the same block on different disks. RAID technology is used in NAS (Network Attached Storage) systems. NAS is an embedded device in the network used mainly for storage.

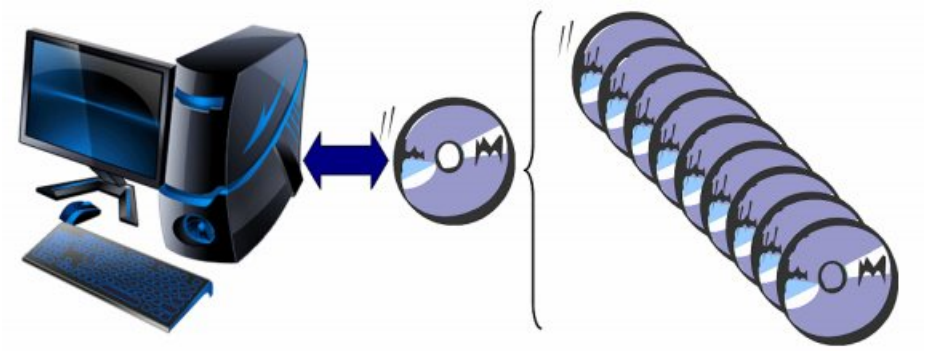


Figure 62.1: A RAID system is used as a single disk device by the computer, but consists of two or more physical disk drives. The computer will use the physical disk drives as one logical disk drive.

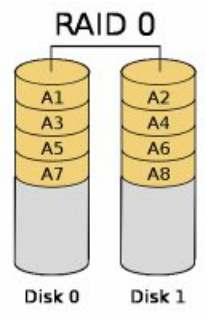


Figure 62.2: A RAID 0 solution with two disks. A block consists of two sectors, the first block consists of sector $A1$ and $A2$, the next block consists of sector $A3$ and $A4$, and so on (www.wikipedia.org 2006).

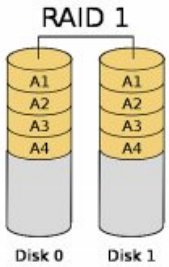


Figure 62.3: A RAID 1 solution with two disks. A block can consist of one sectors and the same sector will be written to both disks. Sector $A1$ contains the same data on both disks (www.wikipedia.org 2006).

62.2 RAID levels

RAID levels 0, 1, and 5 are the most commonly found, and cover most requirements (www.wikipedia.org 2006). The different RAID levels are:

0 "Striped set without parity" or "Striping". Provides improved performance and additional storage but no fault tolerance. Any disk failure destroys the array, which becomes more likely with more disks in the array. A single disk failure destroys the entire array because when data is written to a RAID 0 drive, the data is broken into fragments. The number of fragments is dictated by the number of disks in the array. The fragments are written to their respective disks simultaneously on the same sector. This allows smaller sections of the entire chunk of data to be read off the drive in parallel, giving this type of arrangement huge bandwidth. RAID 0 does not implement error checking so any error is unrecoverable. More disks in the array means higher bandwidth, but greater risk of data loss. RAID 0 (striped disks) distributes data across several disks in a way that gives improved speed and full capacity, but all data on all disks will be lost if any one disk fails. Figure 62.2 shows a solution with RAID 0 with four blocks, block #1 consists of sector $A1$ and $A2$, block #2 consists of sector $A3$ and $A4$, and so on. The system will now read any block almost twice as fast as not using a RAID technology since both disks are reading the sector in parallel.

1 'Mirrored set without parity' or 'Mirroring'. Provides fault tolerance from disk errors and failure of all but one of the drives. Increased read performance occurs when using a multi-threaded operating system that supports split seeks, very small performance reduction when writing. Array continues to operate so long as at least one drive is functioning. Using RAID 1 with a separate controller for each disk is sometimes called duplexing. RAID 1 (mirrored settings/disks) could be described as a real-time backup solution. Two (or more) disks each store exactly the same data, at the same time, and at all times. Data is not lost as long as one disk survives. Total capacity of the array is simply the capacity of one disk. At any given instant, each disk in the array is simply identical to every other disk in the array. Figure 62.3 shows a solution with RAID 1 with two disks, both disks having the same copy of each sector, here shown as $A1$ to $A4$.

2 Hamming code parity. Disks are synchronized and striped in very small stripes, often in single

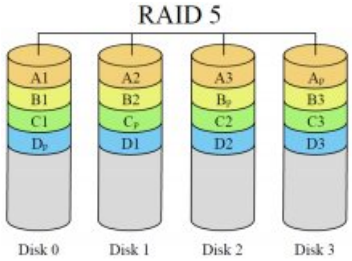


Figure 62.4: A RAID 5 solution with four disks. A set of blocks are dsitributes over the disks with a parity block. (www.wikipedia.org 2006).

bytes/words. Hamming codes error correction is calculated across corresponding bits on disks, and is stored on multiple parity disks.

3 Striped set with dedicated parity or bit interleaved parity or byte level parity. This mechanism provides an improved performance and fault tolerance similar to RAID 5, but with a dedicated parity disk rather than rotated parity stripes. The single parity disk is a bottle-neck for writing since every write requires updating the parity data. One minor benefit is the dedicated parity disk allows the parity drive to fail and operation will continue without parity or performance penalty.

4 Block level parity. Identical to RAID 3, but does block-level striping instead of byte-level striping. In this setup, files can be distributed between multiple disks. Each disk operates independently which allows I/O requests to be performed in parallel, though data transfer speeds can suffer due to the type of parity. The error detection is achieved through dedicated parity and is stored in a separate, single disk unit.

5 Striped set with distributed parity or interleave parity. Distributed parity requires all drives but one to be present to operate; drive failure requires replacement, but the array is not destroyed by a single drive failure. Upon drive failure, any subsequent reads can be calculated from the distributed parity such that the drive failure is masked from the end user. The array will have data loss in the event of a second drive failure and is vulnerable until the data that was on the failed drive is rebuilt onto a replacement drive. RAID 5 (striped disks with parity) combines three or more disks in a way that protects data against loss of any one disk; the storage capacity of the array is reduced by one disk. Figure 62.4 shows a solution with RAID 5 using five disks.

The downside of using RAID 5 is when combining disks of differing sizes, the total storage space will be calculated based on the size of the smallest disk.

$$Total\ available\ storage\ space = (size\ of\ smallest\ disk) * (total\ number\ of\ disks\ -\ 1)$$

6 Striped set with dual distributed parity. Provides fault tolerance from two drive failures; array continues to operate with up to two failed drives. This makes larger RAID groups more practical, especially for high availability systems. This becomes increasingly important because large-capacity drives lengthen the time needed to recover from the failure of a single drive. Single parity RAID levels are vulnerable to data loss until the failed drive is rebuilt: the larger the drive, the longer the rebuild will take. Dual parity gives time to rebuild the array without the data being at risk if one drive, but no more, fails before the rebuild is complete. RAID 6 is sometimes referred to as Advanced Data Guarding (ADG). RAID 6 (striped disks with dual parity) (less common) can recover from the loss of two disks.

The downside of using RAID 6 is when combining disks of differing sizes, the total storage space will be calculated based on the size of the smallest disk.

$$Total\ available\ storage\ space = (size\ of\ smallest\ disk) * (total\ number\ of\ disks\ -\ 2)$$

10 RAID 10 (or 1+0) uses both striping and mirroring. "01" or "0+1" is sometimes distinguished from "10" or "1+0": a striped set of mirrored subsets and a mirrored set of striped subsets are both valid, but distinct, configurations. Figure 62.5 shows a solution where RAID 1 and RAID 0 is combined.

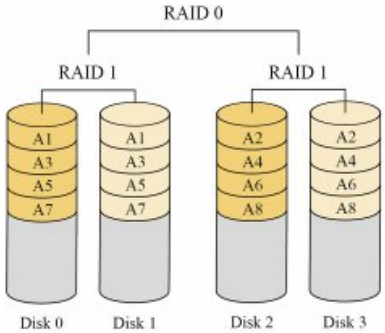


Figure 62.5: A RAID 10 solution with four disks. (www.wikipedia.org 2006).

Level	RAID Minimum # of disks	Space Efficiency	Figure
0	2	93%	62.2
1	2	47%	62.3
2	3		
3	3	62%	
4	3	62%	
5	3	62%	62.4
6	4	47%	

Chapter 63

Backup and Restore

63.1 Introduction

In information technology, backup refers to making copies of data so that these additional copies may be used to restore the original after a data loss event. These additional copies are typically called "backups". Backups are useful primarily for two purposes:

1. to restore a state following a disaster (called disaster recovery).
2. to restore small numbers of files after they have been accidentally deleted or corrupted.

Any backup strategy starts with a concept of a data repository. The backup data needs to be stored somehow and probably should be organized to some sort of level. It can be as simple as a sheet of paper with a list of all backup tapes and the dates they were written or a more sophisticated setup with a computerized index, catalog, or relational database. Different repository models have different advantages. This is closely related to choosing a backup rotation scheme.

Backup can also be a type of storage, and the speed of storage is also of interest. The speed is decided from the solution and the type of device to be used. The flexibility of storage may also be important, giving several factors to evaluate when defining a solution for backup. "Big Data" is also a concept requiring a large and fast storage space, since this data is used for multivariate data analysis.

63.1.1 Unstructured

An unstructured repository may simply be a stack of floppy disks or CD-R/DVD-R media with minimal information about what was backed up and when. This is the easiest to implement, but probably the least likely to achieve a high level of recoverability.

63.1.2 Full + Incremental

A Full + Incremental repository aims to make storing several copies of the source data more feasible. At first, a full backup (of all files) is taken. After that, any number of incremental backups can be taken. There are many different types of incremental backups, but they all attempt to only backup a small amount data relative to the full backup. Restoring a whole system to a certain point in time would require locating the full backup taken previous to that time and the incremental backups that cover the period of time between the full backup and the particular point in time to which the system is supposed to be restored. The scope of an incremental backup is typically defined as a range of time relative to other full or incremental backups. Different implementations of backup systems frequently use specialized or conflicting definitions of these terms.

63.1.3 Continuous data protection

This model takes it a step further and instead of scheduling periodic backups, the system immediately logs every change on the host system. This is generally done by saving byte or block-level differences rather than file-level differences. It differs from simple disk mirroring in that it enables a roll-back of the log and thus restore of old image of data.

63.2 Devices

Regardless of the repository model that is used, the data has to be stored on some data storage medium somewhere.

63.2.1 Magnetic tape

Magnetic tape has long been the most commonly used medium for bulk data storage, backup, archiving, and interchange. Tape has typically had an order of magnitude better capacity/price ratio when compared to hard disk, but recently the ratios for tape and hard disk have become a lot closer. There are myriad formats, many of which are proprietary or specific to certain markets like mainframes or a particular brand of personal computer. Tape is a sequential access medium, so even though access times may be poor, the rate of continuously writing or reading data can actually be very fast. Some new tape drives are even faster than modern hard disks.

63.2.2 Hard disk

The capacity/price ratio of hard disk has been rapidly improving for many years. This is making it more competitive with magnetic tape as a bulk storage medium. The main advantages of hard disk storage are low access times, availability, capacity and ease of use. External disks can be connected via local interfaces like SCSI, USB, FireWire, or eSATA, or via longer distance technologies like Ethernet, iSCSI, or Fibre Channel. Some disk-based backup systems, such as Virtual Tape Libraries, support data de-duplication which can dramatically reduce the amount of disk storage capacity consumed by daily and weekly backup data.

63.2.3 Optical disc

A recordable CD can be used as a backup device. One advantage of CDs is that they can be restored on any machine with a CD-ROM drive. In addition, recordable CD's are relatively cheap. Another common format is recordable DVD. Many optical disk formats are WORM type, which makes them useful for archival purposes since the data can't be changed. Other rewritable formats can also be utilized such as CD-RW or DVD-RAM. The newer HD-DVDs and Blu-ray Discs dramatically increase the amount of data possible on a single optical storage disk, though, as yet, the hardware may be cost prohibitive for many people. Additionally the physical lifetime of the optical disk has become a concern as it is possible for some optical disks to degrade and lose data within a couple of years.

63.2.4 Floppy disk

During the 1980s and early 1990s, many personal/home computer users associated backup mostly with copying floppy disks. The low data capacity of a floppy disk makes it an unpopular and obsolete choice today.

63.2.5 Solid state storage

Also known as flash memory, thumb drives, USB flash drives, CompactFlash, SmartMedia, Memory Stick, Secure Digital cards, etc., these devices are relatively costly for their low capacity, but offer excellent portability and ease-of-use.

63.2.6 Remote backup service

As broadband internet access becomes more widespread, remote backup services are gaining in popularity. Backing up via the internet to a remote location (in the cloud) can protect against some worst-case scenarios such as fires, floods, or earthquakes which could destroy the backups. These systems can also be online all the time so it is possible to have almost instant backup service. However, this may be a disadvantage with the ransomware virus as they will crypt both the storage devices and the backup storage devices as these are online. Regarding ransomware virus an offline backup device will be a good solution in addition to the online backup service. Cloud Recovery is another name of this remote backup system.

The advantages of remote backup services are (cloud based) : 1) someone else is responsible for the backup system, 2) available from several locations (as long as internet is present), and 3) just a remote device.

The drawbacks of remote backup services are: 1) the internet connections (particularly domestic broadband connections) are generally substantially slower than the speed of local data storage devices, which can be a problem for people who generate or modify large amounts of data, 2) the users need to trust a third party service provider with both privacy and integrity of backed up data. The risk associated with putting control of personal or sensitive data in the hands of a third party can be managed by encrypting sensitive data so that its contents cannot be viewed without access to the secret key, 3) the uptime of the remote systems and 4) the challenges with any ransomware virus that will crypt both the local and cloud based storage devices.

63.3 Strategies

63.3.1 Advises

1. The more important the data that is stored on the computer the greater the need is for backing up this data.
2. A backup is only as useful as its associated restore strategy.
3. Storing the copy near the original is unwise, since many disasters such as fire, flood and electrical surges are likely to cause damage to the backup at the same time.
4. Automated backup and scheduling should be considered, as manual backups can be affected by human error.
5. Backups will fail for a wide variety of reasons. A verification or monitoring strategy is an important part of a successful backup plan.
6. Store backup archives in open/standard formats. This helps with recovery in the future when the software used to make the backup is obsolete. It also allows different software to be used.
7. Make a backup of the restore utility or restore application as well.
8. If an online backup service (cloud service) is the main backup service, add an additional offline backup system in case of a ransomware virus attach.

63.3.2 Reasons for backup

1. Hardware errors/problems,
2. System errors/problems,
3. Software errors/problems,
4. Virus,
5. Human errors/problems,
6. Site disasters.

63.3.3 Proposal

1. Use automatic backup, daily, every night,
2. Use incremental backup on Monday, Tuesday, Wednesday and Thursday,
3. Use full backup on Friday,
4. Backup sets:
   (a) Use four sets for each day of the week (4),

   (b) Use three sets for each weekend (full backup) (3),
   (c) Use twelve sets for end of each month (full backup) (12).
   (d) 19 data sets with backup from the last year.

5. A person must be responsible for changing the backup set every day!,
6. Save the last month backup in a different physical location then the computer (office),
7. Save a copy of the restore application with the backup.

Chapter 64

UPS

64.1 Introduction

An UPS (uninterruptedly power supply or uninterruptedly power source) is an electrical apparatus that provides emergency power to a load when the input power source, typically the utility mains, fails (Mohan, Undeland & Robbins 1989). The UPS will monitor the utility mains, and if the mains fail, the UPS will switch to a battery source as long as the utility mains fail, or as long as the battery has a charge. If the battery is becoming out of charge, the UPS can send a signal to the computer system for a secure shutdown.

Studies of down time in computer networks indicates that about 30% of the down time is due to problems with the power supply, and about 50% of unaccountable errors on computer systems are due to the power supply. This should indicate that one reason to avoid the need of data recovery is to use an UPS. Modern operating systems like Windows and Linux also require some time for shut down procedures. The reason is that part of the file allocation table for the slow disk devices are stored in RAM to speed up the access of the disk drives. These tables must be synchronized between RAM and disk before the power is turned of to the computers.

64.2 Function

An UPS consists of a battery, a charger, an inverter, and a monitoring part. The monitoring part is checking the normal power supply, the utility mains, and will switch between the usage of the utility mains or the battery as the power source. The charger is used for rectification and charging the battery when running on utility mains. The inverter is used for inverting the DC voltage of the battery to a valid AC power when running on battery. A Figure with these modules are shown in Figure 64.1. The monitoring part is checking the quality of the utility mains and will switch to the best source depending on this quality.

Figure 64.1 shows the offline UPS topology with normal operation (upper part) and main failure (lower part) situations. The upper part will also be used if there is an UPS failure. This offline UPS topology is also shown in Figure 64.2.

The offline UPS topology is using the mains supply directly in normal operation and if any UPS failure. This offline UPS topology is also known as a passive UPS device in some literature as the active part of the UPS is only used when any power failure.

In some situations a monitoring and/or control of the mains supply is wanted. In these cases the inverter is always used to deliver a controlled mains supply. This online topology is based on a rectifier in stead of a charger as shown in Figure 64.3. The UPS will always use the rectifier and inverter when delivering to the output for some critical load. In case of an UPS failure a bypass supply is used. This online UPS technology is also known as an active UPS in some literature as the active part of the UPS is used at all times.

In an active UPS there will be a more stable power supply to a critical load as this is free of the noise and disturbance that may be present on the mains supply.

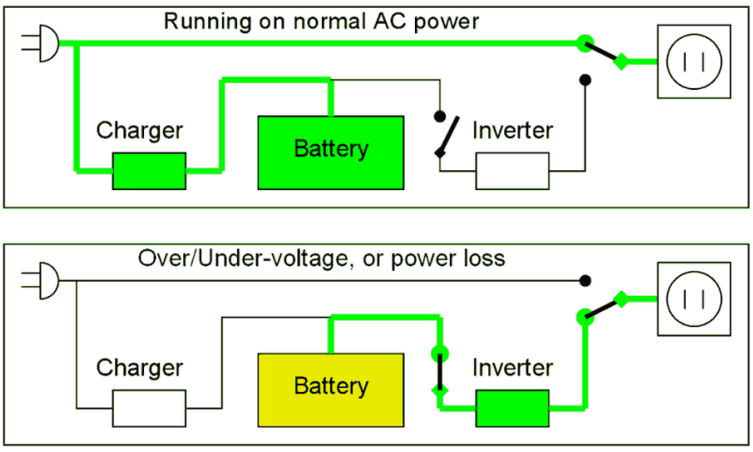


Figure 64.1: An UPS using the utility mains in the upper part, and the battery in the lower part (Mohan et al. 1989). This is also know as an offline or a passive UPS.

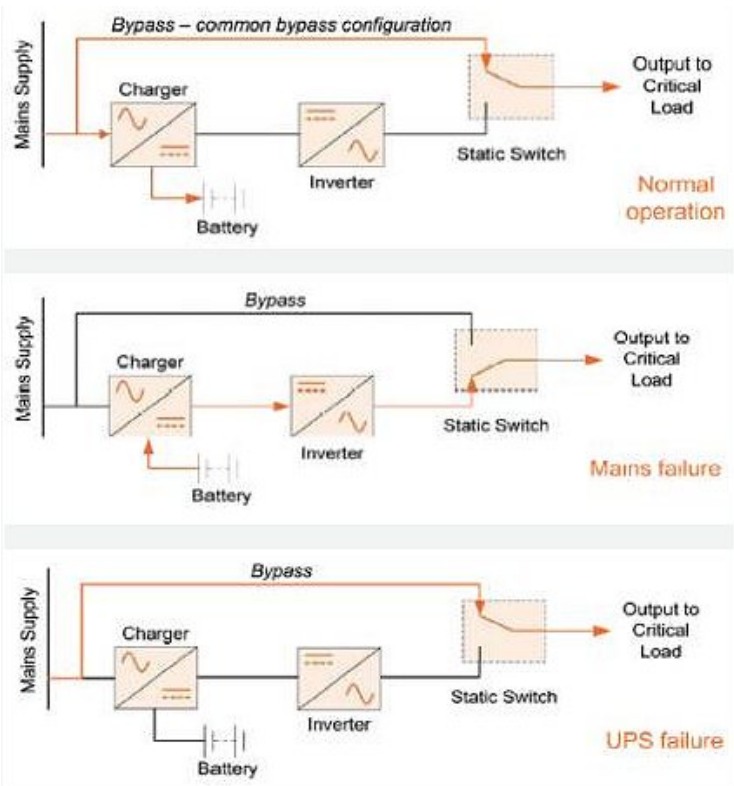


Figure 64.2: Offline UPS topology (Elms 2012).

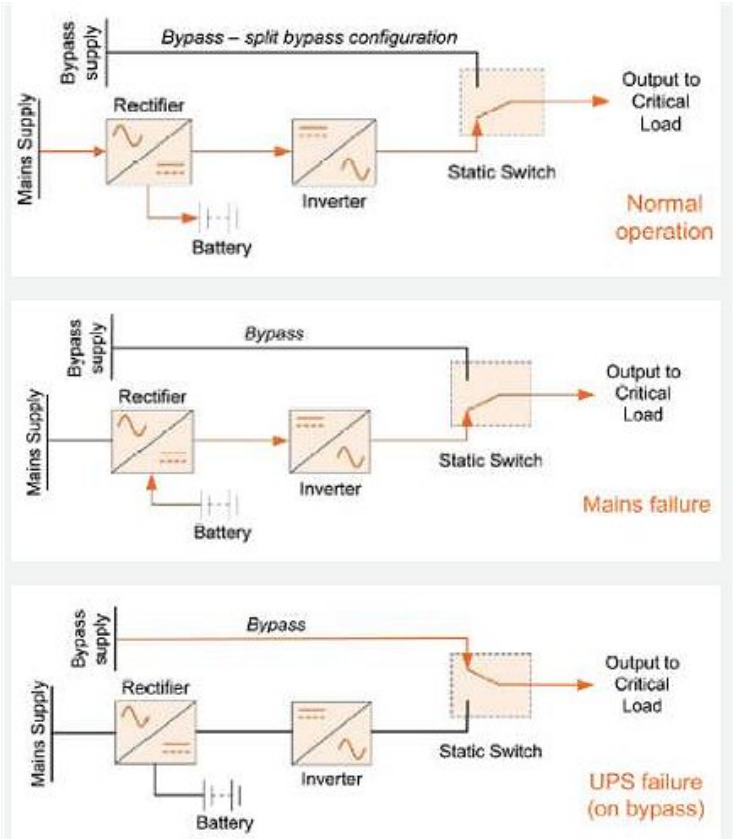


Figure 64.3: Online UPS topology (Elms 2012).

64.3 Rating

There will be two properties for deciding the rating of the UPS. These properties are:

- power consumption; how much power is needed for the connected equipment?
- battery time; depending on the power consumptions, type of UPS, and battery configuration.

The power consumption is the usage of the connected equipment. If your SCADA system consists of a server of 400 W and five workstations of 180 W each will the power consumption be 1300 W. A role of thumb is to add a safety margin of about 20% giving a power consumption of 1500 W. The selected UPS should be able to handle a power consumption of minimum 1500 W.

The battery time should be about 5 to 10 minutes to handle short failures of the mains, and to be able to make secure power down of the system for longer power loss. A larger battery or battery configuration can give a longer battery time of the UPS.

64.4 Location

Normally an UPS can operate in the temperature range of 0 °C to 40 °C and can be located anywhere within this temperature range. The life time of the battery however depend on the surrounding temperature and should not be above 25 °C. A rule of thumb is that an increase in the temperature of 10 °C will reduce the life time of the battery to the half. Also note that an UPS will also produce some heat, often about 10% of the power consumption, meaning that ventilation may be required. Ventilation means that the system may be affected by dust, so the dust should be minimized at the location of the UPS.

The UPS is normally located close to the computers as long as the computers are located in the same physical location. If the computers are located on a larger geographical area, a centralized or a decentralized solution has to be selected. A centralized solution consists of one UPS device with separate cables for power to the equipment. A decentralized solution means several UPS devices located at each computer.

64.5 Software

The UPS is using a software module (software package) on one of the computers in the SCADA system. The UPS is using a protocol based on serial port, USB or the network. The software module is monitoring the status of the UPS and is used for:

- shutdown of servers,
- reboot of servers,
- deal with alarms.

The alarms will the status of the UPS, the batteries, and the utility main. These alarms can also be transmitted using emails or SMS.

Chapter 65

Computer Room Design

A server room or a data center contains several computers and it is important that these computers are not interrupted in any way. Cloud systems are often also running on a data center and there are three different type of data centers. These data centers are 1) hyperscale; large data centers owned and operated by the companies themselves. 2) colocation; larger data centers used by many different users and owned and operated by a data center operator. 3) Edge; smaller data centers, size of a container and owned and operated by the owner.

Such a server room or data center should include several other devices or systems in addition to the computers. The design and maintenance of devices and systems in such a location includes:

- UPS; uninterruptedly power supply or uninterruptedly power source with battery backup,
- Power distribution; power switch boards and racks,
- Fire fighting; systems to avoid fire,
- Security; protection against entry, fire, dust, etc.,
- Air condition; solutions to get rid of the heat from the computer systems
- Cable racks and trays; get rid of all unused cables on the computer room floor,
- Server racks; locations of the computers,
- Network racks; location of network devices,
- Monitoring and control system; a remote location for monitoring and controlling the physical devices, the computer room, and the infrastructure.

Part XVIII

System Design

Chapter 66

Systems Design

66.1 Introduction

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. The main tasks of systems engineers are (Parnell et al. 2012):

- using an interdisciplinary systems thinking approach to consider the complete problem in every systems decision in every stage of the system life cycle,
- converting customer needs into system functions, requirements, and performance measures,
- lead the requirements analysis, design and system validation to achieve a successful system realization.

An overview is shown in Figure 66.1 where the system is fulfilling the requirement of converting the input information to the output information. System design is the process of transferring an idea or a need into an actual system, part of Systems Engineering. This can be a totally new system, an extension of an existing system, or a replacement of an existing system. Defining the architecture means to define a conceptual model of the system describing the structure, the behavior and the view. A view means a specific viewpoint allows the designer to examine one or several portions of a particular interest area. Components and modules will be the combinations of hardware and software. Interfaces are how to exchange information in the system. Data is the type of information needed in the system and where to store the information, duplication of data should always be avoided.

Systems related to industrial information technology (IT) consists of both hardware and software modules, interconnected in many different ways. For such systems will the design be divided into hardware design and software design, but will be analyzed, designed, developed and tested in parallel. System design starts with collection of requirements, requirements describing the hardware, the software, and how the system is going to interact with the "rest of the world". Hardware devices normally end up with a ready made modules, commercial, off-the-shelf (COTS) modules, while software design normally end up with some development. Normally hardware devices contain one or several microcontrollers, and will need software to fulfil the requirements. An overview of the development phases for hardware and software is shown in Figure 66.2.

Systems may be defined in several ways, one of these can be as "an integrated set of interoperable elements, each with explicitly specified capabilities, working together to solve a specific need with a specified outcome and probability of success".

Figure 66.2 shows the phases in system design, starting with analysis and collecting the requirements, designing a new system according to the requirements, construction, and finally testing according to the

Figure 66.1: Systems design is the proces of defining a system, based on hardware components and software modules, to fullfil the requirements for the output information, based on the input information.

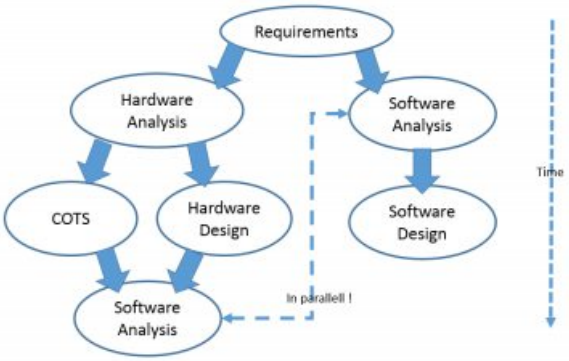


Figure 66.2: The development phases for integrating hardware and software. Due to time limitation will the hardware components and software modules normally be developed in parallell.

requirements. System development starts with the analysis and collection of requirements as a very important phase of the project (Whitten & Bentley 1998). The existing business system, the challenges and problems will be learned during the analysis phase, and this knowledge must be present before the objectives for the improvement can be defined and the business requirements that must be fulfilled by the new technical solution. Any design and implementation of a new system depends highly of the quality of the preceding systems analysis (Whitten & Bentley 1998).

Systems design is the process of:

- converting systems theory to product development. Important disciplines are systems analysis, systems architecture and systems engineering,
- defining the architecture, components, modules, interfaces, and data for a system to fulfil a set of specified requirements.

The specification or the requirements are important, without any requirements there is difficult to do the analysis part, the design part, and test the system.

System design is the process of bridging the gap between the user needs and the development of systems, products, and/or services to satisfy those needs (Wasson 2006), consisting of system analysis, systems architecture and system engineering.

The main focus in this course will be information systems consisting of hardware and software modules, and information going in and out of the system. Normally these systems also have the requirement for some sort of database. Systems design involves the same steps as software development, like some knowledge about the development process, how to collect requirements, how to make analysis, and convert information from analysis part to design and construction. Information from requirements and analysis part should be used for testing.

The design and development of a system should follow some sort of method describing a set of steps of phases. The most important steps are:

- prestudy; to decide if a new system is going to be developed,
- collection of requirements; the specifications and requirements for the new system,
- analysis; what is the system going to do, should be part of the collection of requirements, should include specification of the system and the requirements to fulfill the specification,
- design; how should the system be constructed to fulfil the requirements, should include logical and physical architecture, hardware components, software modules, and database model,
- implementation; construction or development of the system, starting with a prototype and sometimes testing out in a production 0 (zero) serie,
- testing; verify that the system is fulfilling the requirements.

These activities requires education, training, and experience. The goal of the process is to deliver a system, product, or service without any known defects on schedule, within budget, and be able to make a profit of it. These steps or phases will include the following sections.

66.2 Collection of requirements

Requirements are always difficult to collect as the problem domain may be new to the system engineers, different users have different requirements, and humans normally remember the last system challenges best. The requirements are divided into functional and non-functional requirements;

- functional requirements; defining what the system has to do (system behavior),
- non-functional requirements; specifying criteria not related to the behavior of the system.

The non -functional type of requirements may include (Parnell et al. 2012):

- customer requirements; describing what the client expect from the system, including the usage of the system. May also include functional requirements,
- performance requirements; specifying in engineering units how well the functions have to be performed,
- reliability requirements; specifying any safety aspects, environment aspects like temperature range, dust and so on, and the running time of the system,
- constraint requirements; describing the constraints under which the performance needs to be demonstrated,
- design requirements; specifying in detail what is to be build, coded, and/or bought,
- support requirements; describing how to add new functions and extending the system,
- derived requirements; collecting from higher-level requirements,
- allocated requirements; deriving from higher level requirements and assign to a subsystem,
- physical requirements; giving the form, fit and finish of the system.

Each requirement should be unique, specific and testable. Each requirement should be described in such a way that it is not duplicated, no misunderstanding and should be tested in a practical way. Description like a "*fast and easy user interface*" can not be a requirement.

66.3 Prototypes

Prototypes are very useful in the communication with any customers, are used a lot, and known by several names like simulate, model and prototype. In the analysis phase a prototype can be very useful in defining any shape and/or any basic functionality. In a system with a user interface, a prototype of the user interaction can be useful when working on the requirements. The customer can more easily figure out how the system/product can be operated and/or used when playing with the prototype. Using software development tools like MS-Visual Studio it is easy and fast to make a prototype of the Graphical User Interface (GUI) by including any buttons, fields, menus and tab controls to have a working GUI without any real functionality.

There exist several prototyping concepts that can be useful during a system development life cycle, like:

- Functional; implement some of the system functions and use real data, either I/O or a database, without implementing any model,
- Simulated; simulate some of the system functions without using any real data, neither I/O data or database,
- Illustrative; implement only the user interface, on screens and/or reports,
- Evolutionary; develop and implement model(s) than can be used in the final system.

The advantages of using a prototype is for the analyst to get an overview of the system faster and to be sure that the analyst and any customer agree on the functionality of a system. A prototype will be useful for both the analyst to better understand the system and the customer to see how a new system can be designed. Very often bad requirements result in systems being delivered too late, over budget, and with less functionality then desired by the customers. In systems more and more of the development time is spent on developing software, not hardware, giving that the analysis phase is important. Any prototyping concepts can be a good communication tool between the analysts, developers and customers.

The improved communication can have any advantages like:

- a better understanding of specifications and requirements for the system,
- more easy to get the customer involved in the analysis phase of the new system,
- customers and users more prepared for the final product,
- easy to change/extend/discard prototypes during a meeting.

However, any prototyping concepts also have some disadvantages and/or potential risks like:

- the iterative process can make it difficult to plan and schedule the development of the system,
- the customers and/or users may think that the product is almost finished in the analysis phase, resulting in an unrealistic schedule and/or budget expectations,
- confusing the customers and/or users to believe that the prototype is the solution,
- the performance (good or bad) of the prototype may mislead the customers and/or users,
- can be difficult to get the necessary feedback from the customers and/or users required for iterative prototyping.

Prototypes can also be used during construction phase, especially in software design, to let the users test prereleases of the system to get feedbacks early in the development process.

66.4 Pseudo coding

Pseudo coding is an informal high-level description of the operating principle of a computer program or other algorithm. Pseudo coding is a way of writing down the logic of the solution using plain English intended for a human to read it, not a computer. This way it can be a good way of documenting the behavior of your system in the analysis phase. Pseudo coding can be based on any programming language, but focus on human reading. Use functions, variables, loops and conditional statements, but no need of defining variables and use logical function names and names on conditional statements. Focus on the building blocks in the system and do not focus on using the right "words" and syntax for the programming language.

66.5 Architecture

The architecture defines how modules are interconnected to form a system that will fulfil the requirement. The architecture should fulfil the current requirement, but it should also be able to scale the system if growing, and maintaining by changing modules with new modules when any failures. An UML use case diagram can be a good starting point for making the system architecture, as this diagram gives a graphical overview model of the requirements for the actual system.

66.5.1 SysML and UML diagrams

Unified Modeling Language (UML) diagrams can be used in system design as well to describe both logical and physical architecture. The use case diagram can be used to define the boundaries for the system, define any inputs and/or outputs, and the main functions of the system. The deployment diagram can be used to define hardware modules in the system, the input and output modules, and which modules that will contain the software. The interaction diagram can be used to structure the sequence of input of output information in the system and a class diagram can model the data flow in such a system.

System Modeling Language (SysML) is an extension to the UML to support specification, collection of requirements, analysis, design and verification of complex systems.

66.5.2 Context diagram

A is used to establish the context and boundaries of the system to be analyzed or modelled. The diagram focus on the entities that are both on the inside and outside of the system being modelled, and the relationship of the system with these external entities. This diagram, also called a "level 0 data-flow" diagram, is made in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. The entire software system is shown as a single process.

As the diagram focus on entities and flow of information, the external entities and the data flows must be defined for making the context diagram. The steps for making such a diagram can be 1) identify the data flows by study the flow of information in the system, both structured and unstructured information, 2) based on the information flow, identify the external entities as the senders and receivers of information outside the system, 3) define a process representing the system and the entities needed for this process, 4) focus on the data flows between the process and the entities, no focus on the internal data flows. Use the context diagram to understand the system boundaries and scope, and the data flow between the system and any external entities.

66.5.3 Logical Architecture

The logical design of a system gives an abstract representation of the data or information flows of the system, and the main hardware and software blocks and modules in such a system. An UML deployment diagram gives a graphical overview model of the actual system. The logical architecture should give a structure of the hardware modules and interconnections, the software modules, and the database structure.

66.5.4 Physical Architecture

The physical design relates to the actual input and output processes of the system. Any system needs at least one input channel and one output channel. The physical architecture defines how the information is input into a system, how the information is verified/authenticated, how it is processed, and how it is output out of the system. The physical architecture should also describe the distribution of modules, the interconnection of these modules (protocol and/or interfaces) and the type data exchange between the modules.

The logical and physical architecture must also support any requirements for high availability, reliability and performance. Availability is defined as the probability that a system is operating successfully when needed, often expressed in percent. Reliability is defined as the likelihood that a system, or device, will perform the intended function during a specific period of time. Performance can be defined in several ways, in this context it will be to perform a set of operations within a specific period of time.

66.5.5 Big Data Architecture

Digitalization entails more data and new architecture should benefit from the potential of big data. In case of data analysis new architecture should take into account that it can handle a huge quantities of data, and should fit within the existing enterprise IT environment. design of new architecture should evaluate common structures that explains how Big Data complements and differs from existing analytics, databases and systems. This common structure is called a reference architecture. A reference architecture is a document or set of documents describing best practices for such things as hardware, software, processes, specifications and configurations, as well as logical components and interrelationships. One example of such a reference architecture is the NIST Big Data Reference Architecture (NBDRA)[1] from the National Institute of Standards and Technology (NIST) in US.

66.5.6 Hardware

A system design involves hardware modules, and very often there will be a distributed system design involving communications. Define the communication protocols for interconnection of the modules, but do not be too constrain. The usage of wired or wireless communication can be both unless it is stated in the requirement.

[1] See https://www.bigdataframework.org/big-data-architecture (Nov-20).

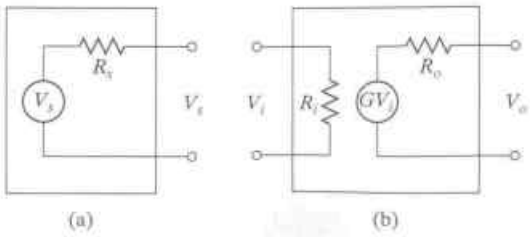


Figure 66.3: A simple model of the sensing device (a) and the amplifier (b) (Wheeler & Ganji 2004).

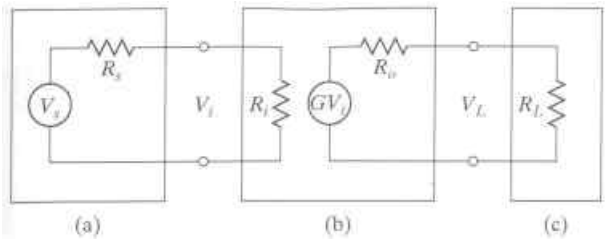


Figure 66.4: A connection of the sensing device model (a), the amplifier model (b), and the load (c) (Wheeler & Ganji 2004).

Interconnecting several or different hardware modules be aware of the input and output loading of the modules. Connecting sensing devices, signal condition devices, and measurement systems can give challenges for input and output loading for these devices and the systems. Simple models of the devices can be used to analyze these challenges(Bentley 2005). Any sensing device or input device can be modeled as a voltage generator $V_{s'}$ in series with a resistor R_s. This model is shown in Figure 66.3(a) . The output voltage is V_S. An equivalent model can describe an amplifier module or other type of modules, shown in Figure 66.3(b).

The amplifier model consists of an input resistor R_i, a voltage generator GV_i, and an output resistor R_o. A model of the complete system is shown in Figure 66.4 with the sensing device (a), the amplifier (b), and the load (c).

The current from the sensing device will be:

$$I_i = \frac{V_s}{R_s + R_i}$$

giving the input voltage of the amplifier to be:

$$V_i = R_i \left(\frac{V_s}{R_s + R_i} \right) = \frac{R_i V_s}{R_s + R_i}$$

Let $R_i >> R_s$, then $V_s \approx V_i$.

The output of the amplifier will be:

$$V_L = I_o \cdot R_L$$

and

$$V_L = G \cdot V_i + I_o \cdot R_o$$

Substitute the current gives:

$$V_L = G \cdot V_i + \frac{V_L \cdot R_o}{R_L}$$

Reorder gives:

$$V_L = \frac{R_L G V_i}{R_o - R_L} = \frac{R_L}{R_o - R_L} \cdot G \cdot \frac{R_i V_s}{R_s + R_i} \tag{66.1}$$

Let $R_i >> R_s$ and $R_L >> R_o$, then an approximate of the Equation 66.1 will be:

$$V_L = G \cdot V_S$$

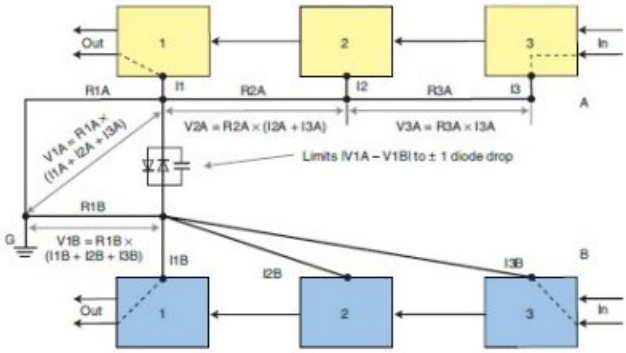


Figure 66.5: Figure is showing two sections, A and B, that both can be on the same Printed Circuit Board (PCB) like an analog and digital ground. The diodes and capacitor will limit the potential differences due to ground bounce. Section A is using a shared ground plane where the voltage drop will depend on the resistance and current, while section B will be less affected by the voltage drops (Novacek 2010).

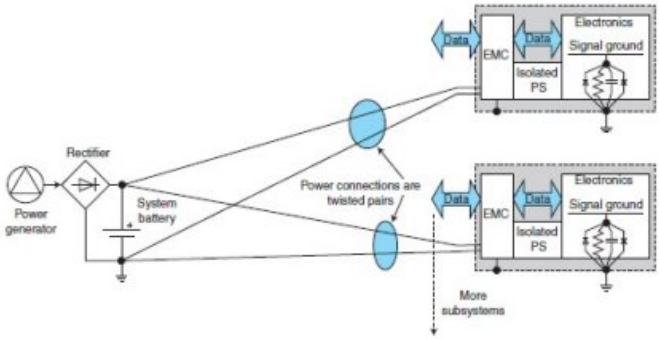


Figure 66.6: Connecting sub systems to single points for power and ground to avoid ground loops (Novacek 2010).fig

The output voltage of the sensing device, V_s, will depend on the load of the device. Generally should the input load be as high as possible (R_i and R_L) and the output load as low as possible (R_S and R_o) giving $R_i >> R_s$ and $R_L >> R_o$.

66.5.7 Grounding

Grounding means the common 0 V potential to which signals are referenced. The "chassis ground" is often referred to as grounding, but may or may not be correct depending on the grounding conductors. The fundamental rule for grounding is to use one single point for grounding(Novacek 2010). Figure 66.5 shows a system with a common ground. The system is divided into two sections, like an analog and digital section. The upper section, section A, has a common ground meaning that depending on the resistance and current for section $A3$ may affect the grounding voltage drop for stage 1 and 2. In such a configuration should the section part with the highest current be closest to the ground reference point G. The lower section, section B, is using separate grounding for each of the stages 1, 2 and 3. Also notice that the grounding of the input signal returns, indicated by the dotted lines, will have an additional resistance. Especially in section A, may the input signal be affected by noise from stage 1 and 2.

Section B shows the approach that is used for analog sections with no sharing of the grounds. In systems with a few inches of wire for grounding it is a good idea to insert fast signal diodes and a capacitor as shown in Figure 66.5 (Novacek 2010).

In system with separate sub systems enclosed in metal cabinets or in safety critical systems all power and ground signals should be connected to single points as shown in Figure 66.6. The enclosure of the sub systems is shown with dotted lines. Both analog and digital systems should follow these grounding rules.

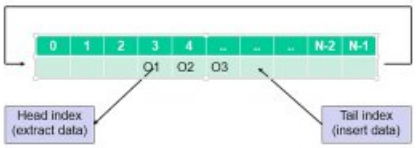


Figure 66.7: A ring buffer with a head and tail index.

66.5.8 Interface

A system consists of many devices and modules. The system can only fulfill the requirements if these devices and modules are exchanging data in a correct way, and the only way of exchanging data is by an interface. Most of the challenges in system design, and systems engineering, are the interfaces. Normally will each of the devices or modules work as expected, however the system is still not working as expected. Always put some extra effort into the interfaces, be sure that all the needed data is available by the interface, and that all the needed services or functions in each device or module, is available by the interface. The OSI model is an important tool when dealing with the interfaces. An important tool will be the N^2 diagram, also referred to as N^2 chart, in shape of a matrix indicating the functional or physical interfaces between the devices and modules in the system (Bonnema, Veenvliet & Broenink 2015). The matrix contains all the devices or modules in the diagonal, with all the inputs as columns and the outputs as rows.

66.5.9 Software

A system design also involves the analysis and design part of the software, any system design will involve software design. Normally ready made hardware modules will be used and software modules must be developed to fulfil the system requirements. Developing software is a demanding operation, both as a individual and as a group. It is important to have the requirements and make a proper analysis. Analysis means to study existing situations and/or solutions to gain an in-depth knowledge of the current "system". Then use this knowledge to create something new, design a new "system" with many possible "correct" answers. Designing and developing software will bring along bugs and errors, mostly based on all the combinations that must be handled by the applications, but also challenges with the programming language used. Developing software tools like a coding standard, test driven development and formal code inspection should be evaluated. Coding standards, like MISRA (McCall, Burden, Burnard, Hennell, Hills, Montgomery, Tapp & Whiting 2008), can be used to structure the usage of a specific programming language and the documentation of the code (both naming conventions and comments). Test driven development is used to include structured testing in the development process, like the Unified Process (UP)(Kruchten 2000). Formal code inspection is a systematic examination (or review) of the source code. This examination or review are done in various forms like informal walk-throughts (explain the function of the code for a colleague), pair programming (two and two programmers working together on the same code), or more formal inspections (external reviewers). These tools are intended to find mistakes overlooked in software development, improving the overall quality of software. Both confidence and safety should be important when designing systems.

Software also include evaluating data structures like arrays, lists, stacks, queues and ring buffers. Array buffers are the most common data structure used and is useful if the maximum number of items is constant, or will not pass a maximum number of items. An array buffer is also useful if the new item can always be added to the end of the array. A list buffer is an array with a more flexible structure as the maximum number do not need to be known and an item can be inserted at any location in the list buffer. Array and list buffers can access any items in the buffer as opposed to a stack and a queue buffer. In a stack or a queue buffer only the items at the end can be accesses, normally either by inserting or removing an item. A stack buffer is a LIFO (Last In - First Out) buffer that can be accessed only from one location, the last item into the stack will be the first item out of the stack buffer. The queue buffer is a FIFO (First In - First Out) buffer that must be accessed from two location only, the input location and the output location. The first item into the queue buffer will be the first item out of the queue buffer. A ring buffer is a queue buffer (FIFO) with a limited size (n items) where only the last n items will be stored. A ring buffer will also have a head pointer and a tail pointer indicating the buffer offset to the first and last item in the buffer. Figure 66.7 shows a ring buffer with a head and tail index.

Note that when the ring buffer is full must both the tail and head index be updated when a new item

is inserted giving that the "oldest" item in the buffer will be overwritten. In some cases, like low pass filter and keyboard input functions to mention some, will a ring buffer data structure be a good solution.

66.5.10 Filtering

All input signals to a system should be filtered in some way, do not trust the signal levels nor information from any system. Digital signals should use the ground as reference and be independent of the maximum voltage for any subsystems. A digital input signal can use a diode and a resistor to be independent of the voltage, while a digital output signal can use a relay. Analog signals are converted to a digital representation of the signal and should be low pass filtered, either by a finite impulse response filter (FIR) or an infinity impulse response filter (IIR). The Discrete Sampling section in the Instrumentation part contains more information about these filter types. Also remember to take care of any data format deviation due to the hardware platforms like Big and Little Endian format, affecting all data format larger then 8 bits.

66.5.11 Platform

A system design has to be connected to a specific platform or a set of platforms, defined in the requirements. The platform defines how to convert from a logical architecture to a physical architecture. The system design platform can also be divided into several abstraction levels like architecture platforms, application programming interface (API) platforms and application specific services platforms. The UML deployment diagram can represent the architecture platform.

Part of the platform decision is to define the type of operating system, should the systems support only one operating system or several operating systems. Do you actually need an operating system? Define the type of services that is needed from the operating system like network support, file support, user interface, embedded options, real-time support, and software development tools.

Also note that the data format may differ between different hardware platforms, known as the Big and Little Endian format. The endian format is handling the low and high end of a word (16 bits) in different ways, so this will affect all data format of more then 8 bits.

66.5.12 Service Oriented Architecture

Service Oriented Architecture (SOA) is an approach using an architecture where the use of services is more important than the information in the system. The services will carry out some soft specific function, such as producing data, analyzing a set of data, finding some specific information in a data set (Big Data), or validating a customer some sort information to mention some sort of service. In SOA is the information from the data more important than the data itself. In order to have SOA support in a system an layer must be added to support the services. This layer must be part of the business layer in the three layer software model as SOA is going to support a set of business based services, working on the information (data in the data layer). Figure 66.8 shows an example of the business slayer in a software application.

The business layer consists among other of the of service layer supporting SOA. The service layer will use the data service layer for handling the information needed for the services. Data services and messaging will use the data abstraction layer for communication with RTUs, databases and external system needed for collecting the necessary information.

66.6 Development

The development phase is the phases of collecting the requirements, do the analyses, design the system, making a prototype, testing and the first running version of the final system. The development phase consists of combining a set of tools and modules to convert the wishes into a running system.

66.6.1 Development stack

The development stack is the development modules used for development of the system. The main focus is software development, and the development stack is the software modules that will be used in the system design. These modules are from the operating system, from the communication modules, from the database support, and any programming languages.

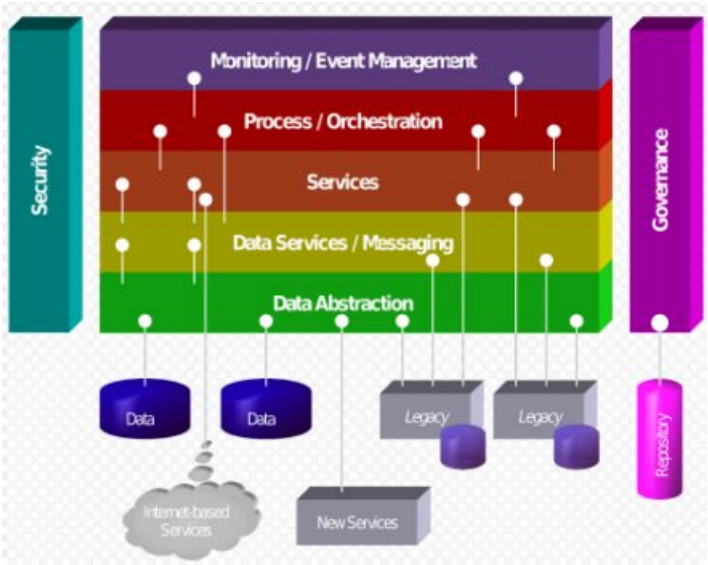


Figure 66.8: An extra service layer must be added to support SOA. The figure shows the UI and business layers of an application where the service sub layer is part of the business layer together with data abstraction, data services, and monitoring and event management. The figure also shows the importance of security (SOA meta-model, The Linthicum Group, 2007).

66.6.2 Communication Stack

The communication stack or communication suite is the modules containing the functionality and services of the OSI model layers. The stack defines how much functionality and services of the protocol may be solved by communication module and how much must be solved by the applications. Depending on type of information and communication, try to focus on having layer #1 (Physical), layer #2 (Data Link) and layer #7 (Application) for serial communications, for network communications include layer #3 (Network) and layer #4 (Transport). If security is important, look into layer #6 as well.

66.6.3 Database Modeling

All information systems requires some sort of a database module, for storage of information. This information has to be structure and this is the main purpose of the database modeling. The information is normally stored in a rational database consisting of tables. Systems involving logging also have to decide using a time series database or a rational database.

66.6.4 Hardware modules

The system should be divided into a set of hardware and software modules. The hardware modules can be ready made modules or modules that must be developed as part of the development phase. Measurement requirements and/or EEx requirements may end up with "in-house" developed hardware. The advice however is to try to use as much as possible of ready made hardware modules and use software modules to fulfill the requirements. The development price of developing and maintaining the hardware is higher than the price of development and maintaining the software, especially when software development is needed anyway. An UML deployment diagram can be used for an overview of the hardware modules in a system.

66.6.5 Software Modules

Software development is more and more demanding in systems using industrial IT. The systems are being more complex with more combinations and the users want a higher degree of integration. An UML deployment diagram can be used for an overview of the software modules in a system.

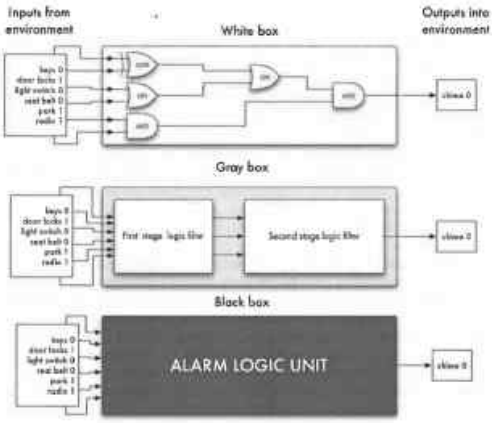


Figure 66.9: The three different "views" of understanding a system (Parnell et al. 2012).

66.6.6 Data Flow

All computer systems consists of some inputs, some sort of computation including some sort of algorithms, and some outputs. The heart of any process control systems will be a data engine. This module will be responsible for getting the data into the system, using some sort of communication protocol, be responsible for the computation based on calculation and/or logic, and getting the data out of the system. A system can consists of several data engines, each responsible for part of the computation, however a superior data engine should be used for monitoring the data flow between any of the data engines. The interconnections will often consists of several communication protocols, and many of the protocols will be a master/slave type of protocol also requiring a data engine.

An UML deployment diagram can be used for an overview of the engine modules in a system. An Use Case diagram can also be used to describe the type of data engines necessary in a system.

66.7 Models

Models can be very useful in the system design as they are flexible, save time and save money. A model is an abstract representation of a system with focus on the information wanted from the system. In system design there are three different ways of viewing, "digging" into, or understanding a system, 1) white box approach, 2) gray box approach, and 3) black box approach. Figure 66.9 shows a "simple" alarm logic unit in a system.

The models will be the same types where the white box type of models consist of mathematical equations describing the relations between the input and output signals. The black box type of models consist of a description of the outputs related to the inputs, based on historical data for both the inputs and outputs. The gray box type may be any combinations.

A conceptual model is a theoretical representation of a system including logic and algorithms, and explicitly recognizes assumptions and limitations (Parnell et al. 2012). The conceptual model has different purposes, 1) identify the purpose of the model, why making the model, and 2) identify the input and output variables. The conceptual model can be constructed as a mathematical, physical, or codified constructed model. Remember that any model should be validated.

There are several types of models including physical, graphical and mathematical. The characteristics of these models can be descriptive or prescriptive, static or dynamic, deterministic or probabilistic, and discrete or continuous (Parnell et al. 2012). The advantage of constructing a model is for simulation, testing out different scenarios while designing and testing the system. Simulation is based on any models operated over time.

66.8 Documentation

There exist different type of documentation for a process control system, from high level overview to detailed lists. Some of these documentation are:

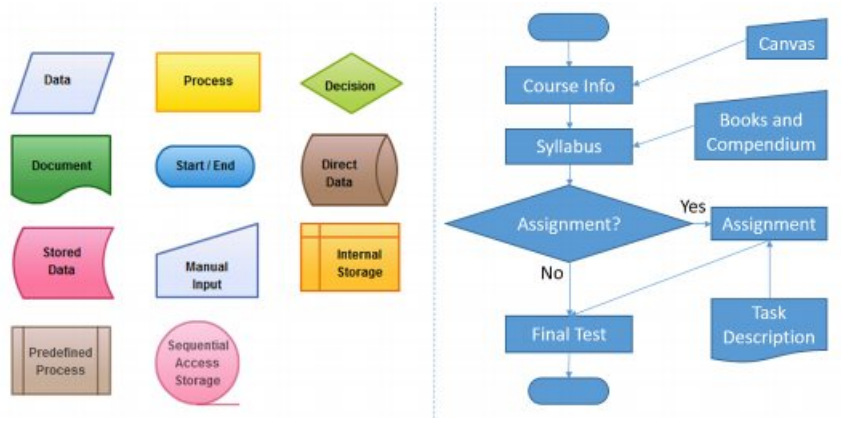


Figure 66.10: The most used flow chart symbols to the left, and an example of a flow chart to the right, showing the work out of our course.

- Process Flow Diagram (PFD); this diagram shows the quantities and type of raw materials needed for making the product, how much and what kind of product each plant will make, indicate critical process conditions and major piping and equipment (Meier & Meier 2011),
- Piping and Instrumentation Diagram (P&ID); this diagram is the master design document for a process, using symbols and descriptions for equipment, piping, instrumentation and control system (Meier & Meier 2011),
- Instrument List or Instrument index; this list contains the data related to the control system components and their functions (Meier & Meier 2011),
- Specification Forms; this form defines all the details for each tag in the process (Meier & Meier 2011),
- Instrumentation Data Sheets; the same as Specification Forms (Meier & Meier 2011),
- Loop Diagrams; this diagram contains a schematic representation of the control loop, sensing elements, control components and control elements (Meier & Meier 2011),
- N2 Diagrams; this diagram contains a matrix with the input and output for each device or module in the system. The column indicate the inputs for each module, and the row indicate the output (Bonnema et al. 2015),

66.8.1 Flow charts

A flow chart is a type of diagrams consisting of a set of boxes of various kinds, representing the data flow, work flow, information flow, algorithm or process. Each box shows the steps for the flow, algorithm or process, connected with arrows indicating the orders. Flow charts is mainly used for documentation of software applications and/or the data flow in a system. Flow charts are defined in the ISO 5807 standard[2].

The main types of flow charts are:

- Data flowcharts; Indicates how data are processed and handled when solving a specific challenge,
- Program flowcharts; Represents the sequence of operations in a software application,
- System flowcharts; Represents the control of operations and the data flow of the system.

Flow charts consists of a set of symbols, based on the boxes. The most used symbols are shown to the left in Figure 66.10. A flow chart of our course is shown to the right in the same Figure, indicating the work flow and the type documentation needed or used.

[2]ISO 5807: web site: https://www.iso.org/obp/ui/#iso:std:iso:5807:ed-1:v1:en (Sep:2017).

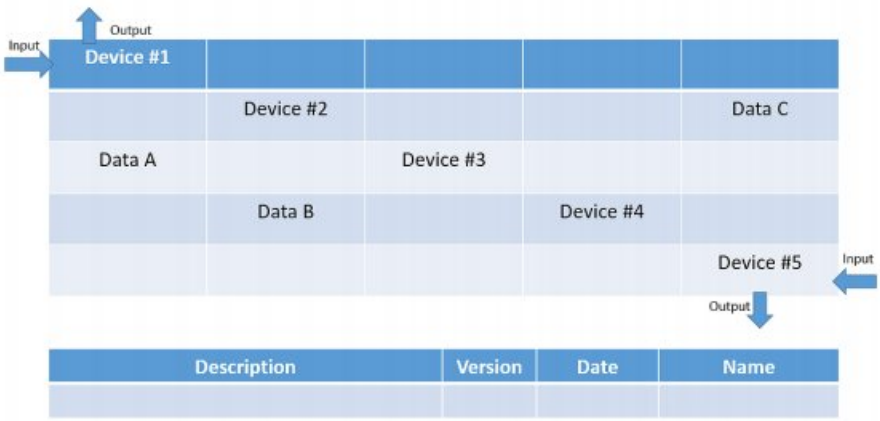


Figure 66.11: A N^2 diagram showing a system consisting of five devices or sub modules. Data A is output from device #1 and input to device #3. Data B is output from Device #2 and input to Device #4.

66.8.2 UML diagrams

UML diagrams (Fowler & Scott 1997) is a type of diagrams consisting of a set of symbols that can easily be drawn by hand, representing the data flow, work flow, information flow, algorithm, devices or process. Different type of UML diagrams gives different type of information like system functions, user functions, static and dynamic information. Some of these diagrams are:

- use case diagram; indicate the main user functions (use case) for the system or the software,
- class diagram; represent the static information in the system, or in the software,
- interaction diagram; represent the dynamic information, the data, work or information flow in the system, or in the software,
- deployment diagram; represent the hardware devices in the system.

66.8.3 N2 diagrams

The N^2 diagram is a matrix containing the input and output connections between all the devices and sub modules in a system. Figure 66.11 shows such a diagram of a system consisting of five devices or sub modules. Device #1 output data A as input to Device #3, Device #2 output data B as input to Device #4, and Device #2 output data C as input to Device #5.

This gives a good overview of the interconnections between the devices and sub systems, and the type of interfaces and data used between the devices and sub systems. In the system in Figure 66.11 there seems to be two independent system functions as Device #1 and Device #3 has no interconnections with Device #2, Device #4 and Device #5. Does this diagram indicate a challenge in the system design of this system?

66.9 Summary of Systems Design Steps

1. Collection of requirements,
2. Analysis,
3. System Architecture,
4. Logical Architecture,
5. Data Flow,
6. Physical Architecture,

7. Development platform,
8. Hardware and Software modules,
9. System Platform,
10. Interconnections (data communication),
11. Communication protocols,
12. Standards.

66.10 Project Management

Development of a system should contain a process and a plan. The design planning should contain a development schedule with the steps, resources and milestones. An example can be:

Index	Main Task	Sub Task	Milestone
1	Preliminary concept		
2			Project Kickoff
3	Startup development	Concept design	
4			Concept review
5		Preliminary design	
6			Prelim. review
7	Detail Design	Prototype	
8			Critical design review
9		System Integration	
10			Design review
11	Verification	System Integration	
12			Test readiness review
13		Formal testing	
14	Industrialization		
15			Production review
16	Initial production		
17			Production review
18	Production		

A Gantt diagram is a useful tool for making a project plan.

66.11 Advices

1. Simplify the solutions. Focus on solutions and results, do not try to make a "perfect" solution. Remember that no perfect system exists!
2. Find the "best" solutions, consider the advantages and disadvantages of all the sub systems.
3. Focus on the development tools.
4. Use a project plan, and be clear on the responsibilities for the sub systems.
5. Be patient, do not demand a solution for all challenges in the start of the project, focus on iterations! Only solve the necessary challenges for each iteration.

66.12 Design of a measurement system

A measurement system consists of both hardware and software. The hardware will be a computer device and some type of DAQ device. The DAQ device is the connection between the computer and any sensor devices and/or actuator devices, the connection between the computer "world" and the "real" world. The DAQ device consists of a combination of input and output ports, like Analog Inputs (AI), Analog Outputs (AO), Digital Inputs (DI) and Digital Outputs (DO). The software will be a combination of a

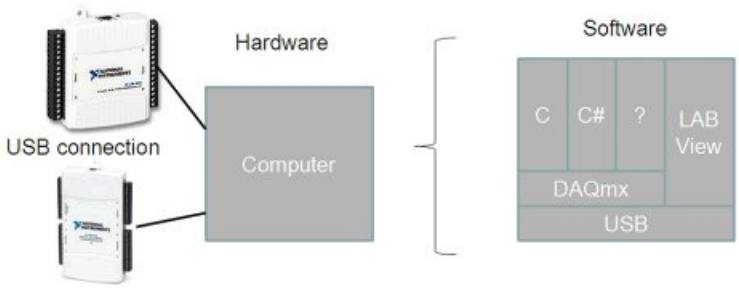


Figure 66.12: An overview of software and hardware using USB devices from NI.

driver or library for the DAQ device and an application communicating with this driver or library. DAQ devices are RTUs (Remote Terminal Units) and can have several shapes and forms. The DAQ device can be an Arduino card, a Raspberry PI card, a PLC (Programmable Logic Controller) or other devices using a specific protocol. Several useful protocol exists, however a protocol supporting layer 7 in the OSI model, like USB and Modbus TCP, is the best choice.

In this case study DAQ devices from NI (National Instruments) using the USB protocol will be used as an example. These DAQ devices can be controlled by the LABview software from NI. However, LABView has an integrated graphical design and configuration "tool" that may not be that useful in an industrial application. NI has a separate library named DaqMx handling the USB protocol and communication with the DAQ devices. The DaqMx library contains an API (Application Programming Interface) meaning that you can write your own software in an object orientated programming language like C#, instead of LABView, for getting the input values from any sensor devices, and write the output values to any actuators. The DaqMx library or DLL (Dynamic Link Library) can be downloaded from NI web site and configured at installation time with the API for the .NET platform (C# programming language).

Figure 66.12 shows the hardware and software needed for such a system. One solution can be to use LABView, another solution can be to make an application yourself to interface to the DAQmx library (DLL) shown in Figure 66.12. The same concept can be used to make interfaces to other type of DAQ devices like the Arduino platform or other type of platforms. More examples of DAQ using a PC are described in (Magda 2009) and (Magda 2011).

Making your own software in C, C# or other programming languages supported by the DAQmx library or other type of DAQ platforms, the following sections need to be addressed:

- configuration,
- calibration,
- sampling time(s),
- reading the sensor values,
- converting the sensor values from the voltage value to a valid unit value, display and log the values,
- writing the actuator values,
- converting the output unit value to a valid voltage output value,
- error handling,
- looping reading all sensor values and writing all actuator values.

These sections are shown in Figure 66.13, indicating the sequence of the section in the software application. The application also need a configuration section to be able to add and remove sensors and/or actuators. The configuration and calibration data should be stored on files or in a database so the application can use stored data and reload data when starting and stopping.

The section are normally solved by looping through all the sensor devices and actuator devices, these devices are connected to different input and output channels on a DAQ device. Some DAQ devices can be used to read or write all channels in one command, other DAQ devices only each channel can be

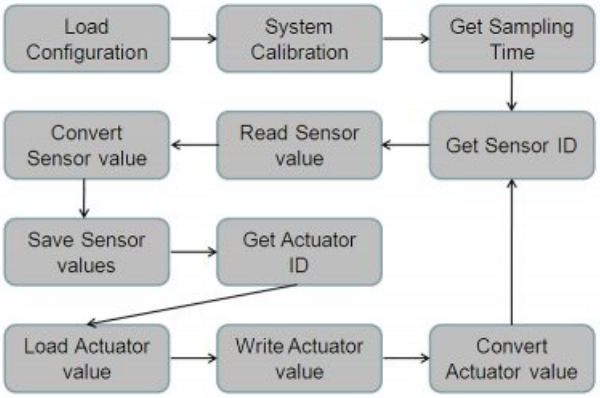


Figure 66.13: The main steps that should be addressed when designing a measurement system.

used at a specific time. The software can be divided into more details as shown in Figure 66.14. This Figure shows the startup sections and two main loops, one loop for handling all the sensor devices, and one loop handling all the actuator devices. Most of the systems today are sequential meaning that the system can handle only one type of input or output at a time. The software application must control the sequence. Also note that in this Figure the sampling time is checked for every sensor ID, some system can or must use a common sampling time, other system can use different sampling time for each channel. The sampling time can be controlled either by hardware or software, or both, depending on the design of the software.

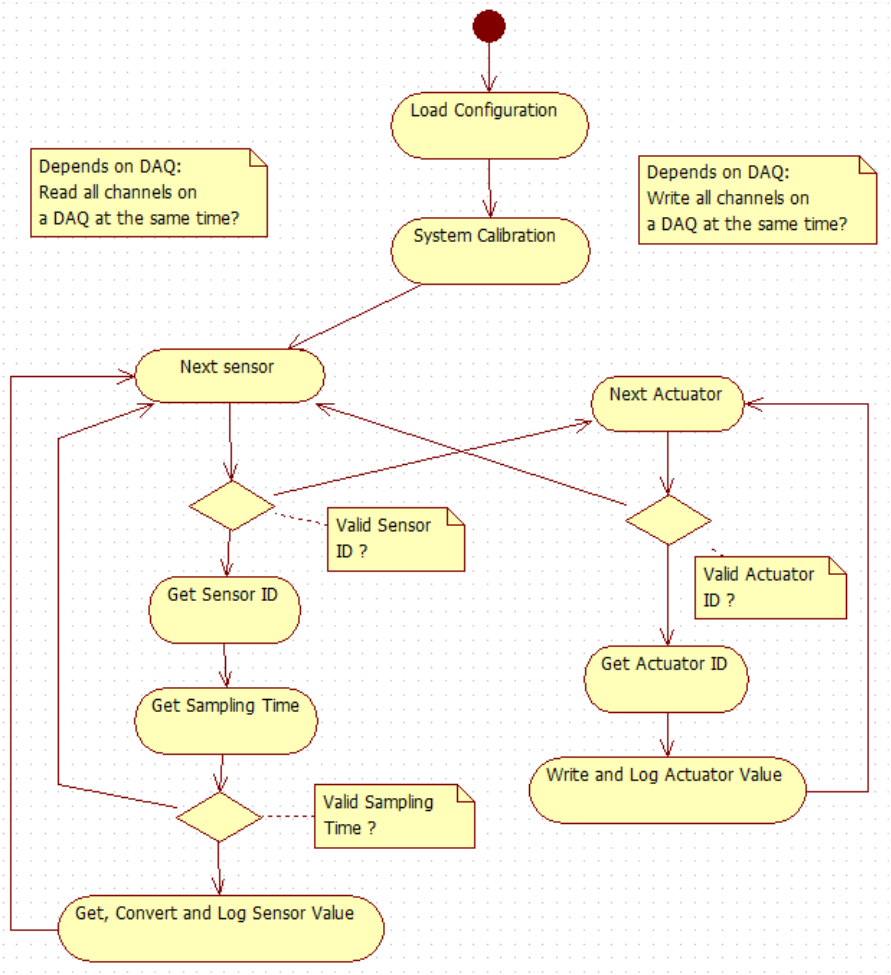


Figure 66.14: The main steps that should be handled in a DAQ measurement software, documented using an UML Activity diagram.

Part XIX

Documentation

Chapter 67

Guidelines for reports and projects

A report will be a document that communicates the work that has been carried out during some sort of experiments, group work, literature study, evaluations, and any combination. It is important to figure out who will be the reader of the report and the competence, experience and/or skills of these readers, and adjust the contents of the report to these readers.

67.1 Reports

67.1.1 Informal report

An informal report is mainly used for internal purposes, and should normally be as brief as possible. The main focus is to document an experiment, a meeting, or a discussion. This report should normally include an introduction, body, conclusion and recommendations. Always start the report with to whom the report is going, whom it is from, the date and the subject. After the conclusion include your contact information if needed, at least the company, email and phone number. The "Recommendations" section can be used to list any other people supporting the information of your report.

67.1.2 Formal report

A report shall communicate the work done and the results of the work. Remember that the grading or any decisions will normally only be based on the information in the report. Therefor focus on getting all the necessary information into the report. A formal report should contain:

1. Title page; title, author, and date,
2. Abstract (or summary); brief problem description, brief description of methods, and the main results,
3. Preface; what the report is about, any changes of tasks, and credits,
4. List of tables and figures (optional)
5. Table of Contents
6. Introduction; short about the background, previous work, and new work,
7. Problem description; process and equipment description, measurement setup, etc.,
8. Theory; model and/or method development,
9. Methods; Method development,
10. Results; simulation results, model fitting, optimization results,
11. Discussions; the results, as expected or what are causes, uncertainty, remaining work,
12. Conclusions; focus on the results and what you have learned,
13. Appendices; task description, details, listing, data sheets, etc,

14. References; necessary information for finding the references,
15. Index (optional).

67.1.3 References

Literature references can be done in several ways and it is up to writer to select the right style. The most used styles are the Harvard style and the Vancouver style. Do NOT mix these styles within the same document.

References to web pages should be avoided, but if you really need to use such a reference, you should include the author (if the name of the author is not available, you may use the name of the organization/institution) as well as the title of the web article, the URL and the date when you accessed the web page.

All Figures and Tables MUST also be referenced in the report text, including any sections in the Appendix. A Figure or a Table without a reference in the text do not need to be in the report.

Harvard style

In the Harvard style, the reference in the text body should be placed in parentheses after (or in some cases inside) the sentence. It should include the author's name and the year of publication (example (Flor, 2006)).

The references should be listed in detail in a separate list at the end of the report. The references should be listed alphabetically, using the authors last name.

Vancouver style

In the Vancouver style, the references in the text are given as numbers in brackets (example: [3]), and the references in the reference list should be listed using the numbers in brackets as well. When using this style, the references should be given according to the order of appearance in the body of the text.

67.1.4 Article or paper

An article or paper is used to document any research work and should contain the following sections:

1. Introduction; introduce the problem you will be discussing in your article or write a short story of your experience with the problem.
2. Methods; discuss the methods that you want to use on the problem or challenge you outlined in the introduction. Break up each point into separate paragraphs,
3. Results; discuss the results of any experiments or simulations used with the methods. Break up each point into separate paragraphs,
4. Discussion; this should include a brief summary of the article.

A mnemonic rule can be IMRAD (Introduction, Methods, Results, And Discussions).

67.2 Presentation

Documenting a project can often be combined with a presentation. Some advices keeping a presentation (Kolda & Torczon 2011):

- Make eye contact with the audience, do not look to much at the screen,
- Try to use some colors on your presentations, but not too much. Focus on the contents!
- Do not overcrowd the slides,
- Avoid acronyms and math symbols,
- Make good graphics (and animations),

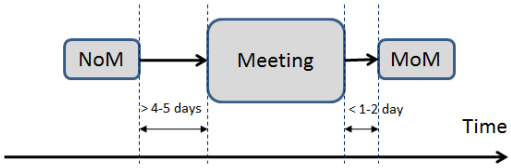


Figure 67.1: Suggestions for minimum timing of "Notice of Meeting" (NoM) and "Minutes of Meeting" for a formal meeting.

- Put your work into context, but limit the time used on background material,
- Use time preparing your slides,
- Limit your time!

67.3 Meetings

67.3.1 Non formal meetings

Small meetings, often short meetings once a day or week. Use these meetings for discussions about challenges in the project, the work done and/or the plans before the next meeting.

67.3.2 Formal meetings

Formal meetings include a "Notice of meeting" (NoM) and a "Minutes of meeting" (MoM) and these meetings are often used with milestones in the project. The milestones will be part of the planning and should be part of the schedule. Figure 67.1 indicates the timing of these documents with reference to the meeting. Note that the NoM timing is the absolute minimum, at least a couple of weeks is optimum in the industry. The timing for the MoM is maximum, try to make the MoM as fast as possible after the meeting while everybody is remembering the cases.

"*Notice of meeting*" should contain:

1. Name of the project; which project is the meeting for,
2. Place; the location, often a building and room number,
3. Date and time,
4. The name of the members of the meeting,
5. The agenda.

Remember to inform/schedule the meeting early,
"*Minutes of meeting*" should contain:

1. Name of the project,
2. The name of the members of the meeting (indicate who was present and who was not present),
3. The date and time,
4. A table of the subject discussed on the meeting. Very often will this table contains 3 columns, the ID, the subject, and the responsible. An example of such a table can be:

ID	**Subject**	**Responsible**
1	Subject #1 discussed ..	
2	Subject #2 discussed ..	Name and date
..	..	
N	Subject #N discussed ..	Name and date

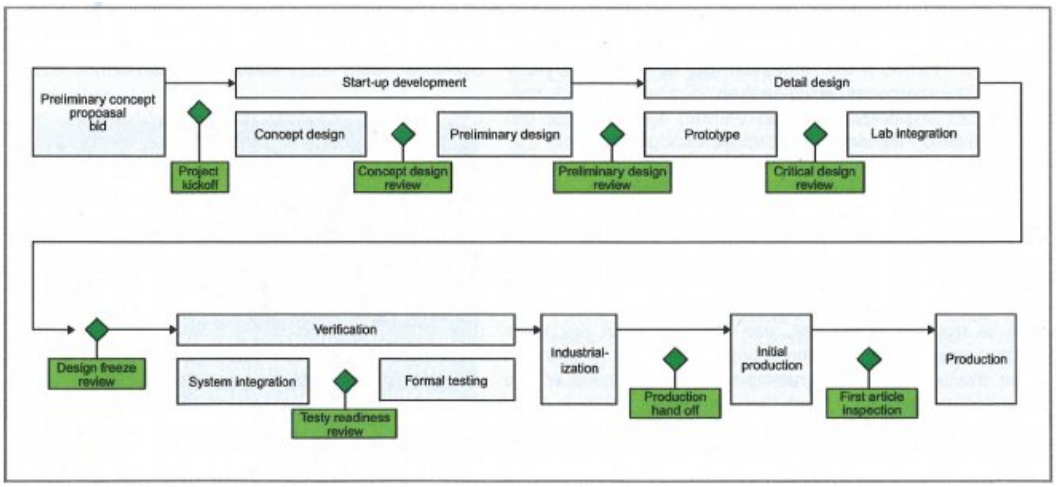


Figure 67.2: Product development steps with the main milestones.

67.3.3 Effective meetings

Some tips for both the attendees and the chair of the meeting.

1. Determine whether the meeting is necessary, determine the number of people and the location,
2. Be precise, do not let the late comers control the start of the meeting! Do NOT recapping the meetings for latecomers!
3. Be prepared, the background material should be received and reviewed prior to the meeting!
4. Have an objective, "what should be the result of the meeting",
5. Have an agenda and distribute the agenda prior to the meeting!
6. Address the most important items first. If you run out of time, the most important items have been addressed.
7. Be clear about the responsibilities,
8. Document the meeting, include deadlines and responsibilities.

67.4 Projects

Figure 67.2 indicates the fundamental stages in product design. It does not matter whether there is a big project or a small project, a large company or a small company, the stapes are the same. The milestones are marked as green blocks and each milestone must be measurably.

Even if the steps are the same will the extent of each step vary according to the type of product, system requirements and size of the project team. A competent engineer must conceptualize the project, make a proper development plan, and then try to follow this plan. Any deviations, update the plan to reflect the real situations. An important document in project management is the schedule, made as a Gantt diagram. An example of a Gantt diagram is shown in Figure 67.3 with the tasks, subtasks and milestones. This diagram also shows the timing of each task (horizontal), any dependencies between the tasks and resources used in each task. Based on this diagram the fastest delivery of the product is possible and the status of the project at any time as long as the project members update the status of each subtasks.

The tasks is based on Work Breakdown Structure (WBS) of the project. Start a project planning by defining a set of WBS. A WBS defines a set of tasks that can be completed independently of other WBSs (tasks). Each WBS must be braked down into manageable tasks. If a task is not manageable, break down the task into sub tasks. If a sub task is not manageable, break down the sub task into sub sub tasks, and so one. A task is manageable when it is possible to estimate the amount of work to do, and the time for doing this work. Use the WBS to facilitating resource allocation and assignment of

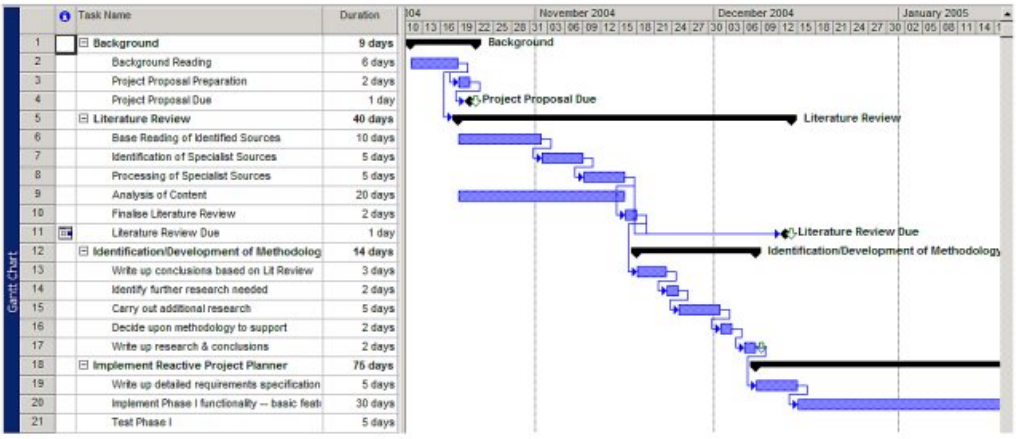


Figure 67.3: An example of a product development schedule (https://www.sitepoint.com/project-management-web-geeks: oct-16).

responsibilities for the tasks. The WBS will also be used for measurement and control of the project as the schedule indicates the start of each task and the responsibility of each task. Several software tools can be used to make the Gantt diagram like Microsoft Excel and Microsoft Project. Microsoft Project is a tool used a lot in the industry.

The development process consists of a set of milestones, shown as green diamonds in Figure 67.2 and black diamonds in Figure 67.3. The milestones are assessments of the project status and are important for formal reviews of the project and making decisions for further progress of the project. Reviews are to the engineer's advantage as the customer can confirm the project's compliance according to the requirements. These reviews prevents the customer from demanding (unpaid) changes or a redesign at a later time. The number and type of milestones may depend on the project, the milestones indicated in Figure 67.2 are:

- Project kickoff: just a startup of the project,
- Concept design review: defining the overall structure and architecture of the system,
- Preliminary design review (PDR): define the system architecture, partitioning of functions between hardware and software using block diagrams and pseudo coding, preliminary reliability analysis (if needed). An approval from the PDR indicates that the design of the system can start,
- Critical design review (CDR): the culmination of the detail design phase of the project, both hardware and software modules with bill of material (BOM) of hardware modules, and tested and documented software modules,
- Design freeze review: testing of the prototypes and normally this will consists of loops before the design is freezed,
- Testly readiness review, production hand off and first article inspection are milestones for system testing and production of the system or product.

Important guidelines for planning a project include:

- Plan the milestones, not the tasks: Focus on the milestones. A milestone is either OK or not, while it is very hard to measure how complete a task is.
- Include milestones into your WBS to manage the "deliverables" of the tasks.
- Each milestone should be assigned to only one person (resource): Clear responsibility makes it easier to get the work done and the project team will trust the results of the milestones as well,
- Let all members of the project be responsible for the planning of the tasks behind each milestone. If a person has been planning a task, he may also have the responsibility and ownership of the task to fulfill it within the schedule.

Part XX

Appendix

Appendix A

OPC Example

Install the *MatriconOPCSimulation.EXE* application on a Windows computer (Windows version only) and start the OPC server. Note that this installation will only install the Simulation server, not the DDE server. The DDE server must also be installed if using data from the Microsoft Excel application. Figure A.1 shows the startup window of the OPC simulation server.

Click on the *Alias Configuration* line, and a new window is shown on the right side of the main window. Right click on the line and select *insert new alias* item as shown in Figure A.2.

Insert a new value, call the value Temp and select the value as a Triangle Waves and Int4 as shown in Figure A.3. An item called Temp is now defined in the OPC server.

Lets use the *MatrikonOPCExplorer* as an OPC client to display the item Temp from the OPC server. The startup window of the OPC client and the Temp item defined in the OPC server is shown in Figure A.4.

First right click on the server name to connect to the server. Select connection to the OPC simulation server on local host. The client should now connect to the server and the icon in from of the server name should change to show that the client is connected. See Figure A.5 for the connect menu.

When the client is connected, a group must be added to be able to connect to the item8s) on the server. Right click on the server name again, and select the Add Group option as shown in Figure A.6.

Select the Add Group and call the new group for TempGroup as shown in Figure A.7. Other parameters as Update Rate and % Deadband can be set on the group as well.

Select OK to save the group and right click on the group name to add items to the group. Select Add Items in the menu, shown in Figure A.8.

Select New Items from the menu, and a new window showing the items for this group will be present. The window is shown in Figure A.9.

Click on the Configured Alias folder name and the available tags will shown in the lower window. Double click on the Temp tag and the tag name will show in the Tag ID line on top of the window. Use the button with the arrow to transfer the tag name to the added tag window shown in Figure A.10.

Then select File $\longrightarrow$ Validate Tags to validate the tag. Use File $\longrightarrow$ Update and return to save the tag in the group and the application will return to the main window of the OPC client. The OPC client will now display the value of the item as shown in Figure A.11.

More practice:

1. Try to add more items to the group, different type of values and so one.
2. Try to use another client, try for example MATLab® if you have the OPC toolbox.

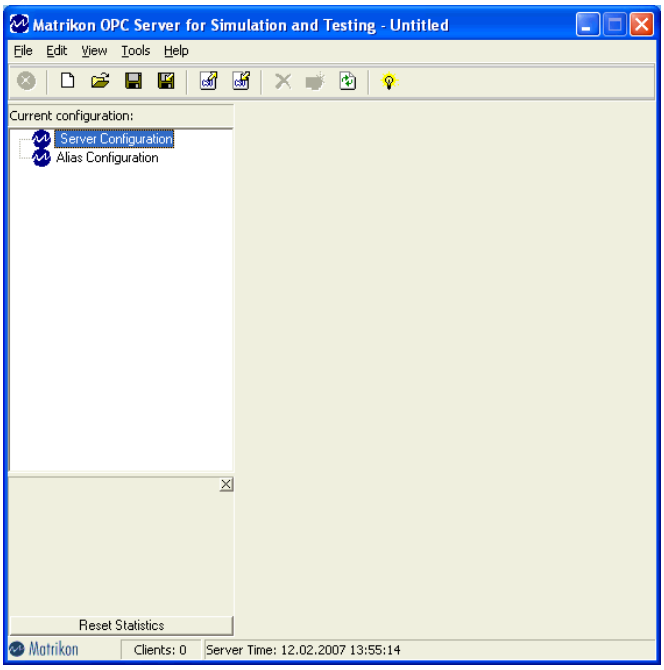


Figure A.1: The startup window of the OPC simulation server.

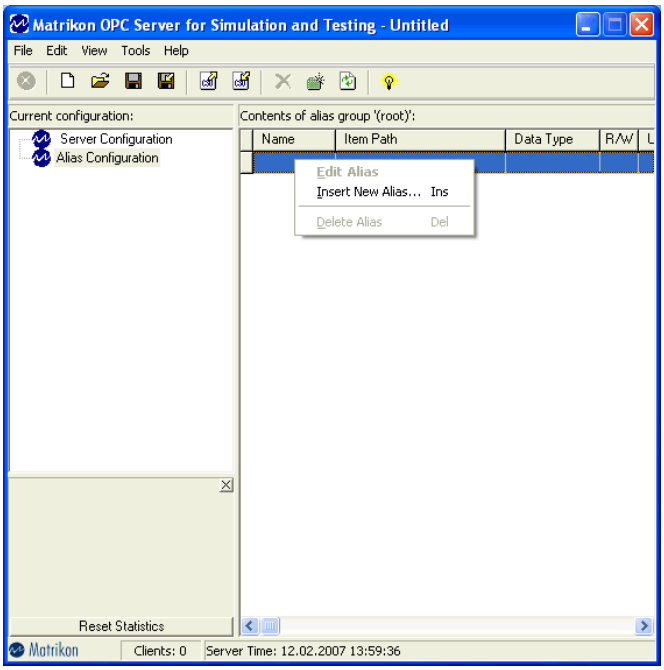


Figure A.2: Insert of new alias items in the OPC server.

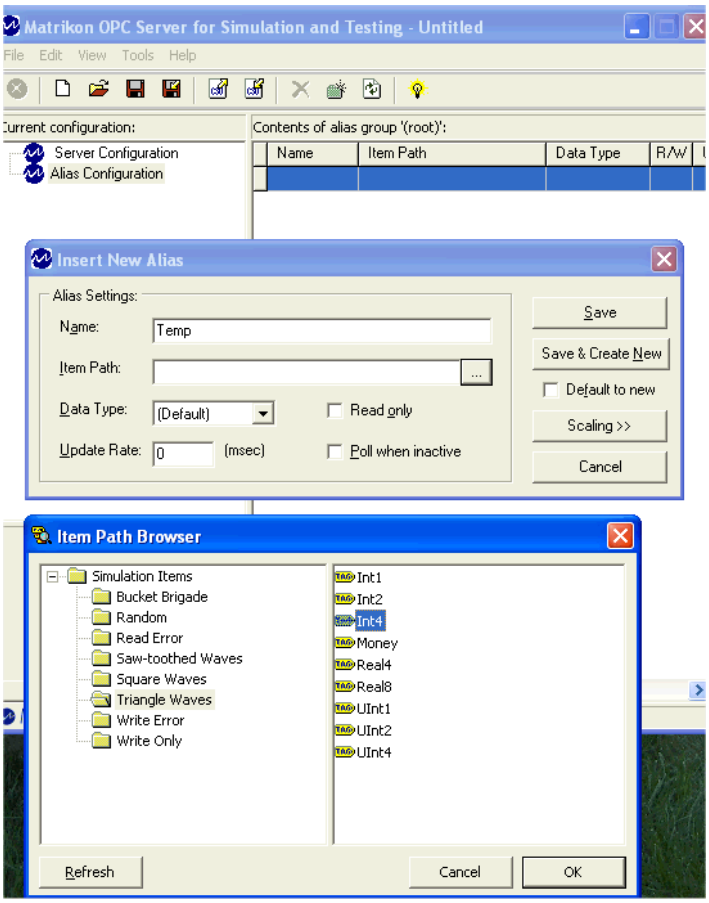


Figure A.3: Insert an item called Temp being a Triangle Waves type as Int4.

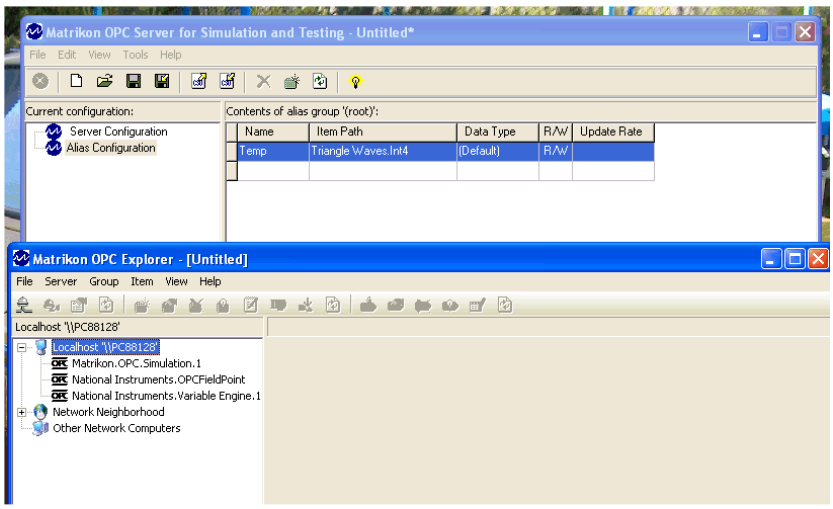


Figure A.4: Start of the OPC client on top of the OPC server.

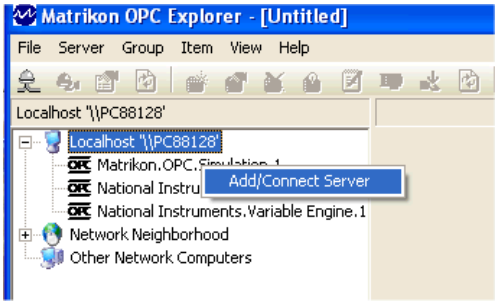


Figure A.5: Right clikc on the server name to get the connect menu option.

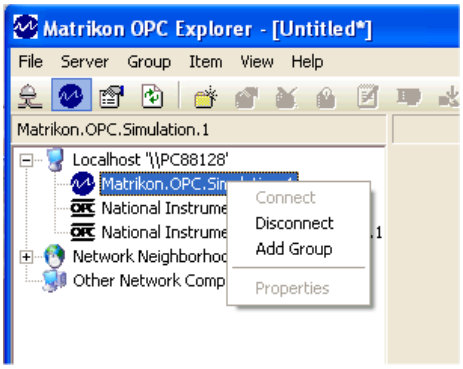


Figure A.6: Right click on a connected OPC server to get the Add Group menu option.

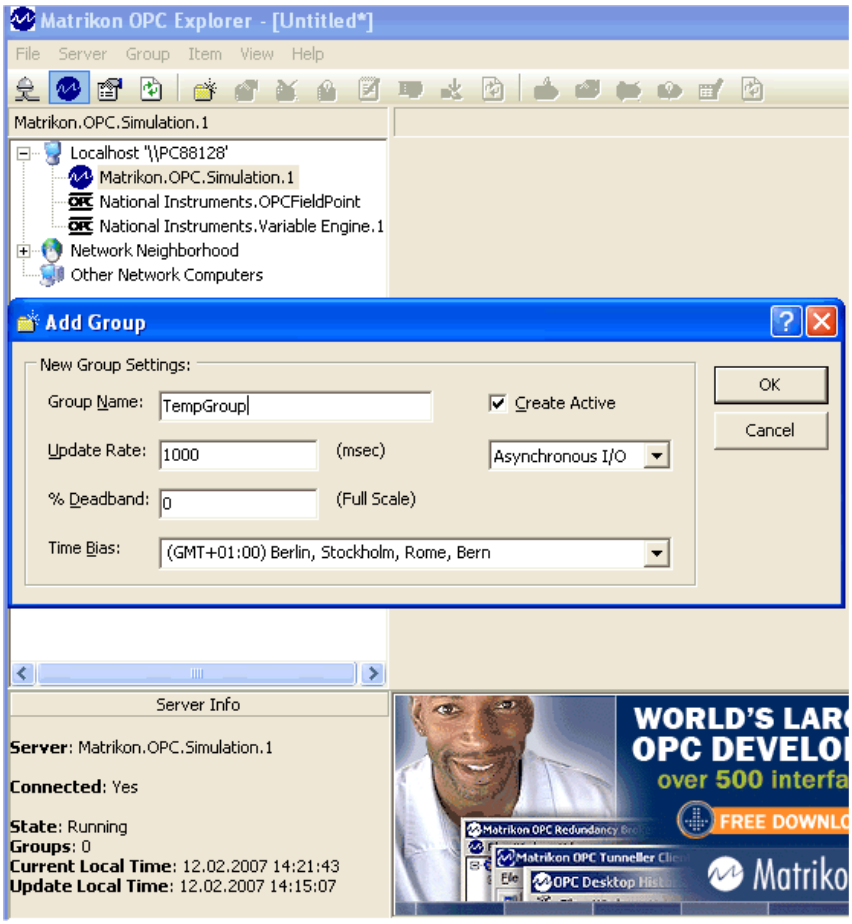


Figure A.7: Add a new group to the OPC server from the OPC client.

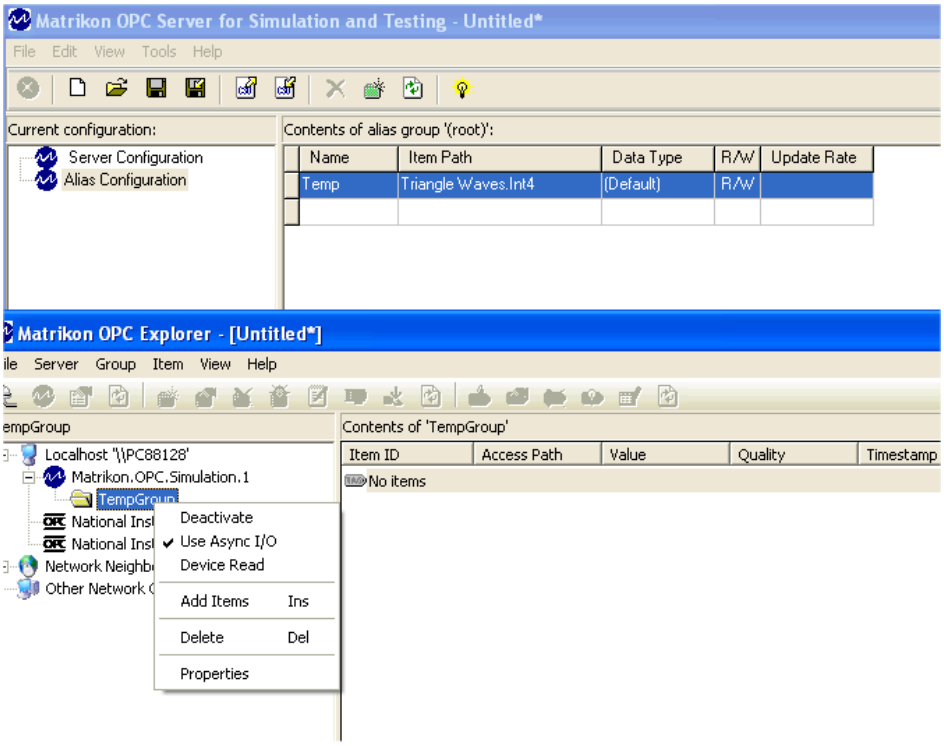


Figure A.8: Right click on the group name to get a menu for adding item(s) to the group.

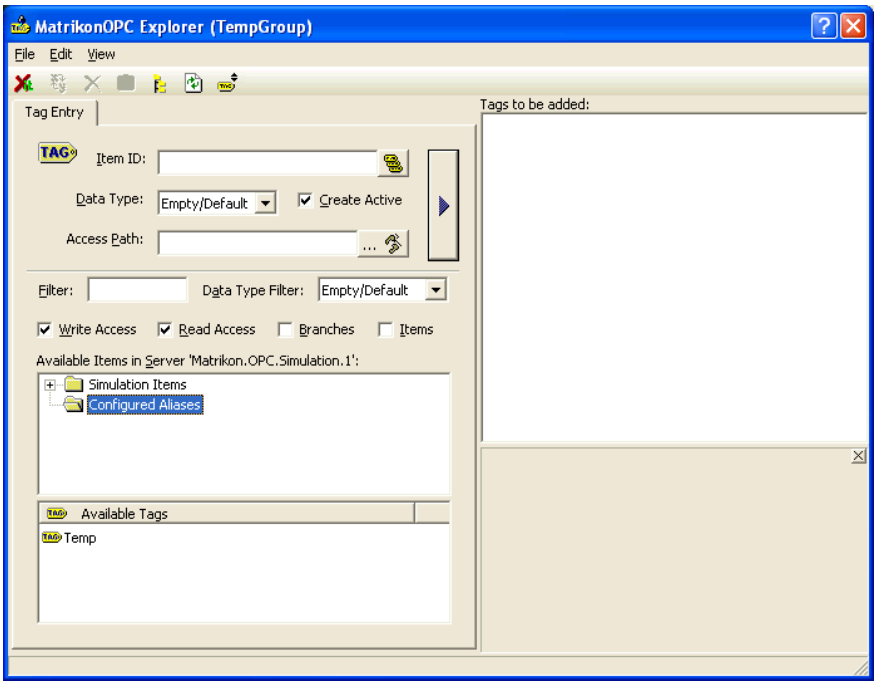


Figure A.9: The window for adding new items to a specific group on the OPC server.

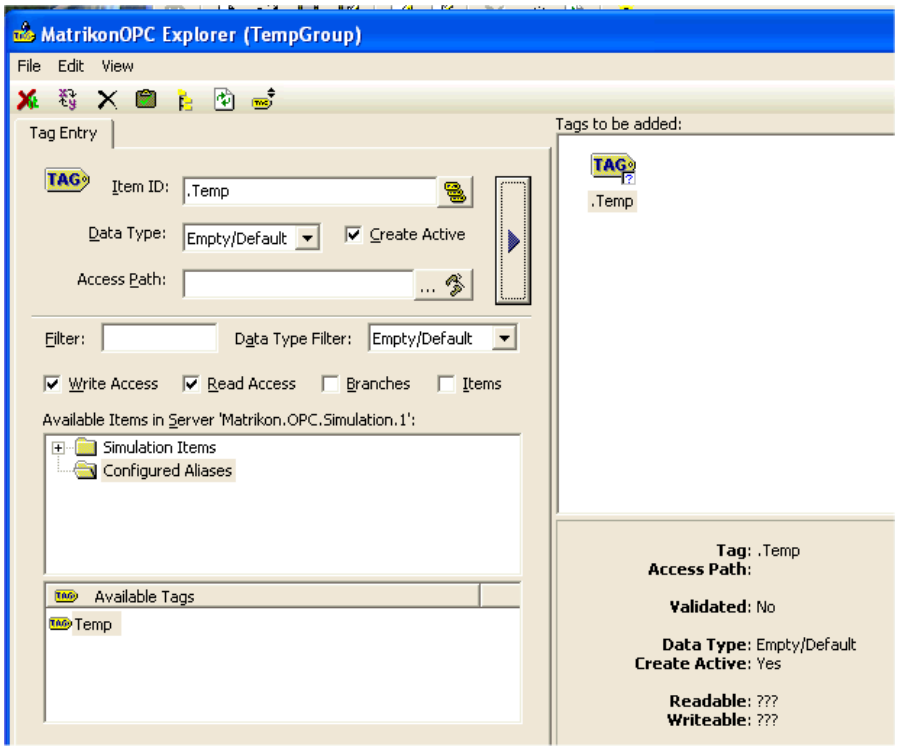


Figure A.10: The Temp item from the OPC server is added to our group.

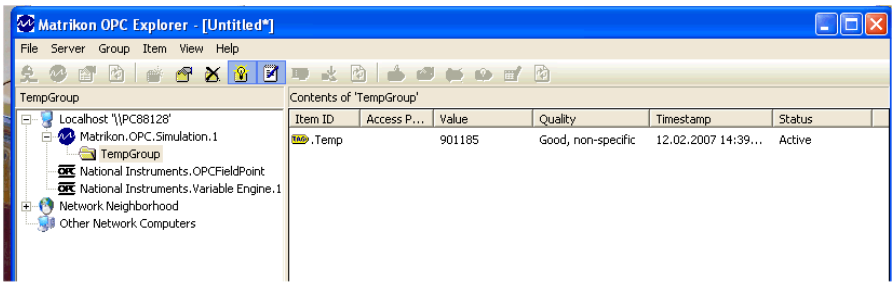


Figure A.11: The OPC client is displaying the value of the Temp item from the OPC server.

Appendix B

MySQL Example

MySQL[1] can be used for noncommercial usage of SQL databases. This example shows how a MySQL database can be installed, created and used.

B.1 MySQL installation

The database can be installed on the computer and the installation of MySQL server 5.0 on the Windows XP operating system is shown in the Figures B.1, B.2, B.3, B.4, B.5, and B.6.

Note that the MySQL server is started as a Windows service and will be running all the time. Use the ControlPanel -> Adminstration Tools -> Services to change the MySQL services to manual if you want to start and stop the MySQL database yourself.

B.2 MySQL example

Use the command line based SQL client to make query for the database. Figure B.7 shows the start of the SQL client, starting with the password. The password is the same password as given in Figure B.4. Figure B.7 shows the prompt *mysql>* where new commands can be given. A database called *nos_test* was created using the command:

```
CREATE DATABASE nos_test;
```

and Figure B.7 shows the databases using the command:

```
SHOW DATABASES;
```

The command:

```
USE nos_test;
```

is used for open the database, also shown in Figure B.7. The next command:

```
SHOW TABLES;
```

is used for listing the data tables in the *nos_test* database and as shown in Figure B.7 no tables exist in this database. A table can be created using the CREATE TABLE command as shown in Figure B.8. The table is created with 3 fields, 2 text fields and one character field, named name, address, and grade.

The command DESCRIBE table_name; can be used for listing the type of the fields in the table, shown in Figure B.9 for the student table.

Elements can be added to the table using the INSERT INTO command shown in Figures B.10 and B.11.

The contents of the table can be listed using the SELECT command, shown in Figure B.12.

[1] MySQL: Can be downloaded from www.mysql.com (feb-07).

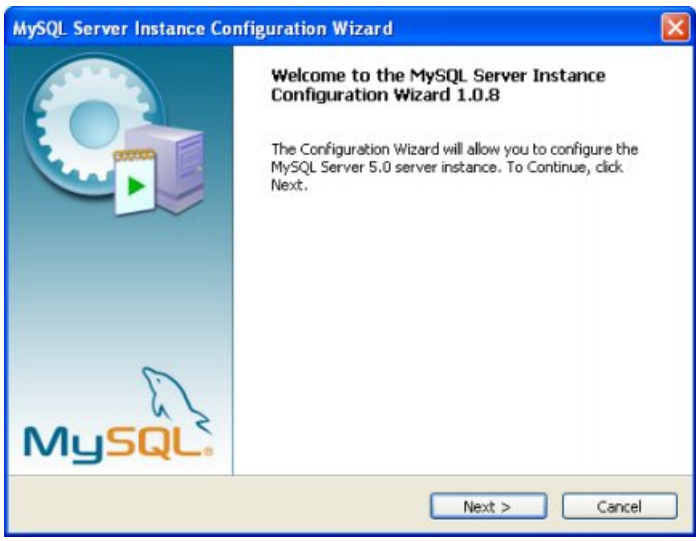


Figure B.1: Installation meny #1 in installation of the MySQL database.

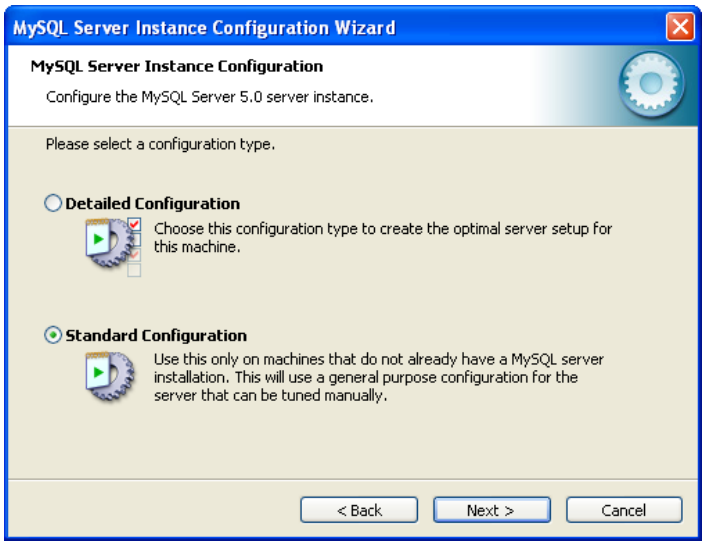


Figure B.2: Installation meny #2 in installation of the MySQL database.

Figure B.3: Installation meny #3 in installation of the MySQL database.

Figure B.4: Installation meny #4 in installation of the MySQL database.

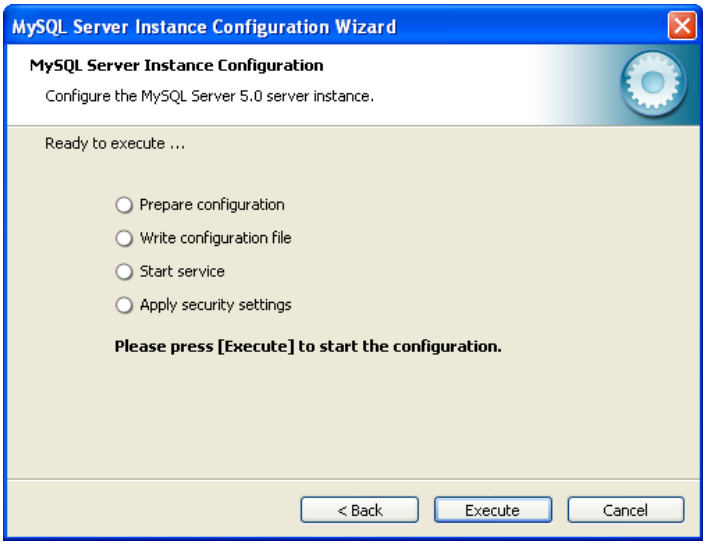


Figure B.5: Installation meny #5 in installation of the MySQL database.

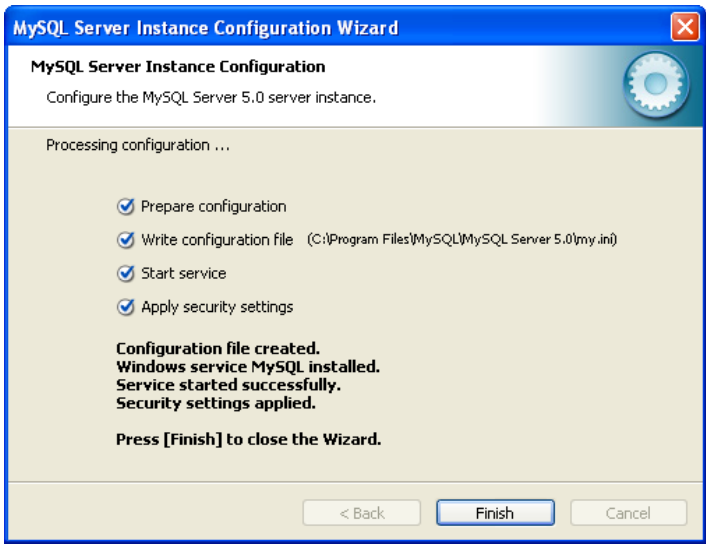


Figure B.6: Installation meny #6 in installation of the MySQL database.

```
MySQL Command Line Client
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.27-community-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| nos_test           |
| test               |
+--------------------+
4 rows in set (0.12 sec)

mysql> use nos_test
Database changed
mysql> show tables
    -> ;
Empty set (0.00 sec)
```

Figure B.7: Start of the command line based SQL client.

```
mysql> create table students(
    -> name text,
    -> address text,
    -> grade char(1));
Query OK, 0 rows affected (0.13 sec)

mysql> _
```

Figure B.8: Create the students table on the SQL server.

```
mysql> describe students;
+---------+---------+------+-----+---------+-------+
| Field   | Type    | Null | Key | Default | Extra |
+---------+---------+------+-----+---------+-------+
| name    | text    | YES  |     | NULL    |       |
| address | text    | YES  |     | NULL    |       |
| grade   | char(1) | YES  |     | NULL    |       |
+---------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Figure B.9: Use the DESCRIBE command to see the definitions of the table.

```
mysql> insert into students
    -> (name, address, grade)
    -> VALUES
    -> ('Nils Nilsen', 'Nilsen street Porsgrunn', 'C');
Query OK, 1 row affected (0.04 sec)
```

Figure B.10: Insert the first item into the student table.

```
mysql> insert into students
    -> (name, address, grade)
    -> VALUES
    -> ('Hans Hansen', 'Hansen street Porsgrunn', 'B'),
    -> ('Fname Lname', 'Main street Porsgrunn', 'A');
Query OK, 2 rows affected (0.05 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

Figure B.11: Insert the next two items into the student table.

```
mysql> select * from students;
+-------------+-------------------------+-------+
| name        | address                 | grade |
+-------------+-------------------------+-------+
| Nils Nilsen | Nilsen street Porsgrunn | C     |
| Hans Hansen | Hansen street Porsgrunn | B     |
| Fname Lname | Main street Porsgrunn   | A     |
+-------------+-------------------------+-------+
3 rows in set (0.01 sec)

mysql> _
```

Figure B.12: Use the SELECT command for listing the contents of the students table.

Appendix C

Alarm System Example

CARGOMASTER®, a level gauging system, is used as an example for an alarm system. This application, shown as version 5, is a monitoring system for tanks on board a ship and contains a tank monitoring section and an alarm section. The alarm section will be the focus in this chapter. Note that this is an older system and is not according to the YA-711 standard. The main screen of the application is shown in Figure C.1 showing the alarm section and the user section of the GUI.

The alarm section is green when there is no alarms and red when any alarms are present. This section also contains 3 buttons:

1. Silence; to silence the alarm horn when an alarm is activated,
2. Acknowledge; to acknowledge an alarm,
3. Overview; to present a list of the alarms and add some more alarm options.

The acknowledge button is shaded out since there is no alarm in this system. The overview button is used for the alarm list window, shown in Figure C.2.

This alarm window contains the following options:

1. A button for acknowledging the current alarm,
2. A button for acknowledging all alarms,
3. Sort options for showing all alarms, only active alarms, or only acknowledge alarms,
4. Test options for testing the alarm system,
5. A button for printing the current alarm list,
6. A OK button to remove the alarm overview list.

This level gauging system consists of a set of sensors, a measurement system, and a computer systems. The sensors consists of temperature sensors like PT-100 and PT-1000, pressure sensors of the $[0-10]$ mV and the $[4-20]$ mA types, FMCW radar sensor, and sensors to measure the angle. The temperature sensors, the pressure sensors, and the angle sensors are all analog sensors and the measurement system will use a ADC to convert the analog values to a digital representation of the values. The radar sensors have an internal ADC and is using a protocol to exchange the values with the measurement system. The measurement system is using double RS-485 serial communication lines for communication with the computer system. The computer system contains the level gauging application and the alarm application.

This system is an integrated system as shown in Figure 39.4. A more detailed block diagram of the system is shown in Figure C.3.

The alarm limits and the alarm enable/disable settings is for configuration, but can also be changed in the application. The possibility to change these properties in the application should also be part of the configuration. In the CARGOMASTER® application can these properties be changed by the user, or they can be fixed, depending on the requirements of the system. Figure C.4 shows how the alarm enable/disable settings can be changed in this application. The type of information is shown as a folder or tab to the right, this type of information is the *Alarm On/Off*. Each alarm type for each tag can then be enabled or disabled by setting the these properties.

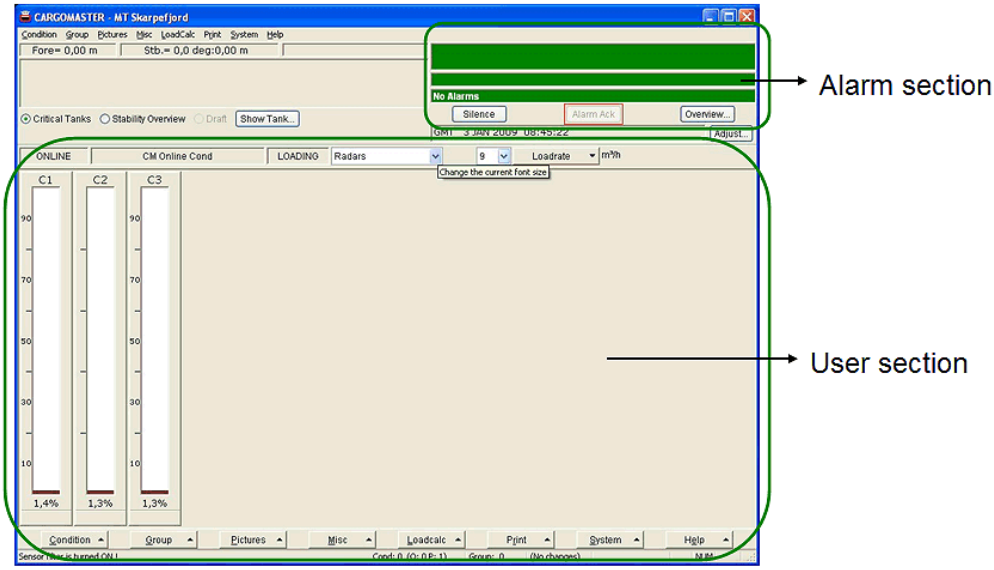


Figure C.1: The main screen of the CARGOMASTER® level gauging system.

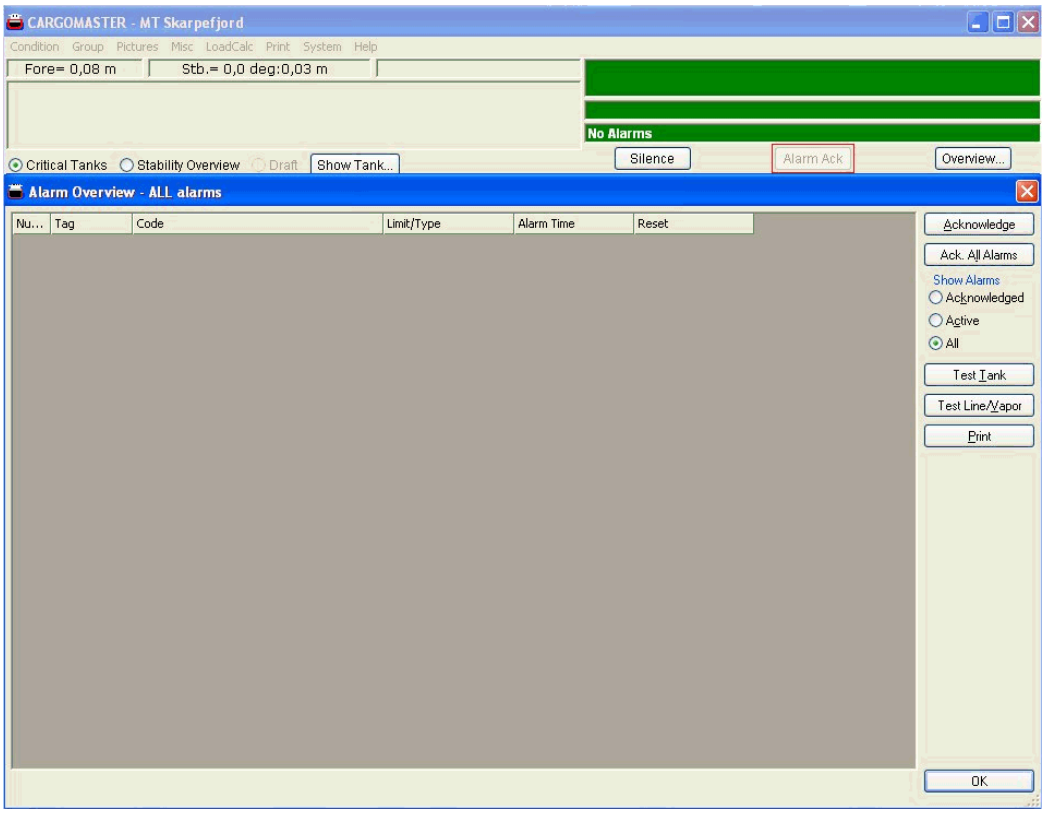


Figure C.2: The alarm list window in the user section. This alarm list window contains some actions, sort options, and test options on the righthande side.

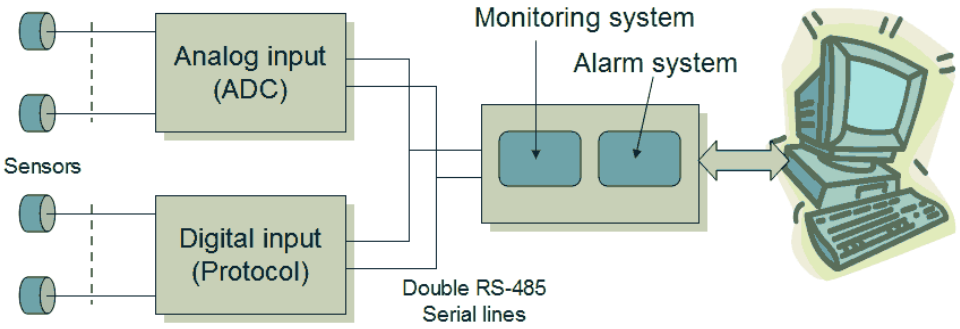


Figure C.3: A block diagram of the CARGOMASTER® level gauging system.

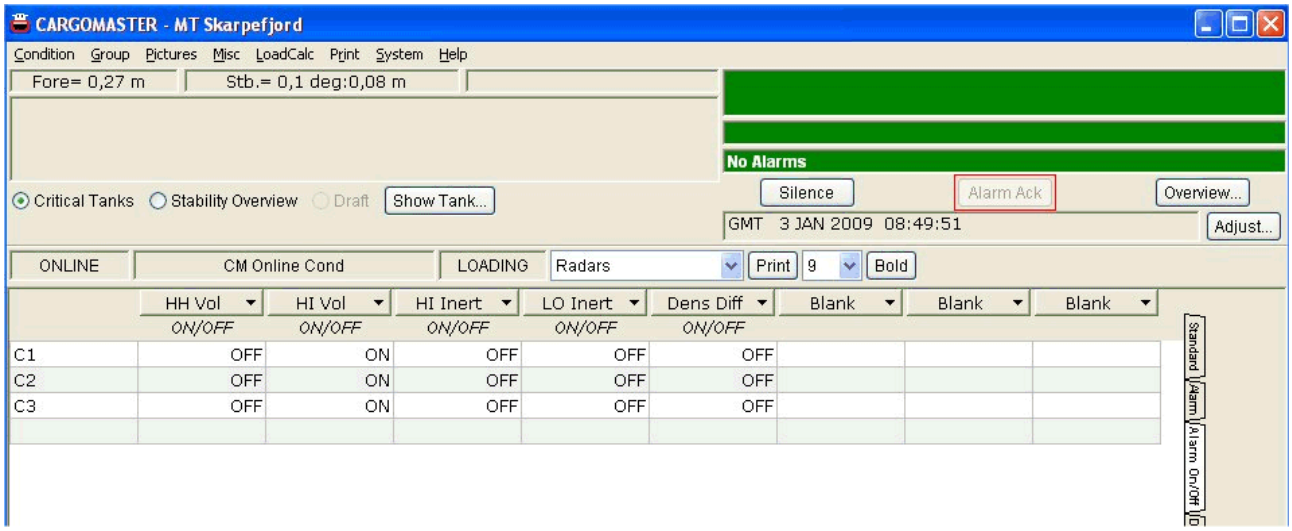


Figure C.4: The Alarm On/Off properties for a set of tags in the system.

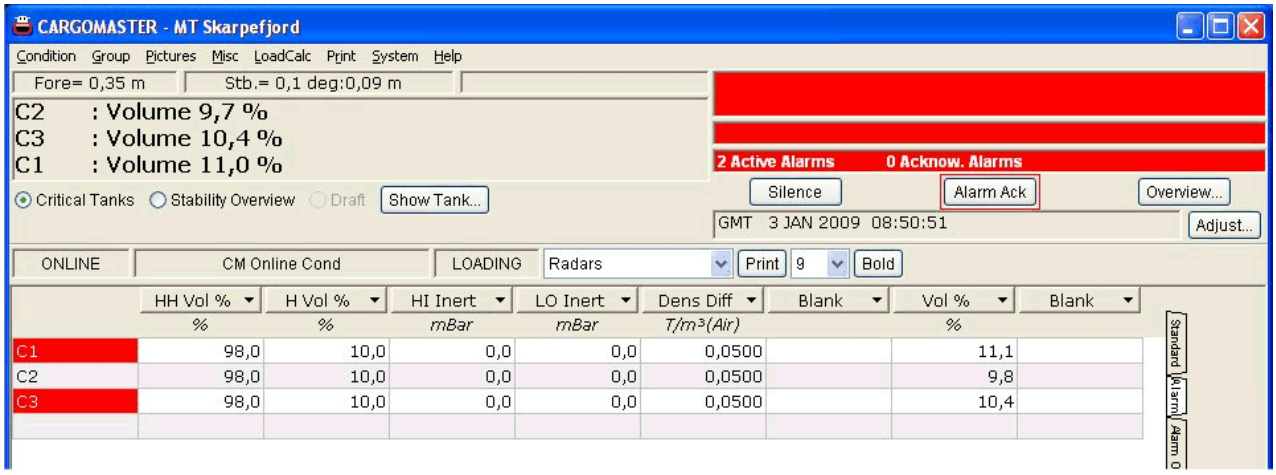


Figure C.5: The Alarm properties together with the current process values for a set of tags in the system. 2 of the tags have active alarms, the last tag does not have any alarms.

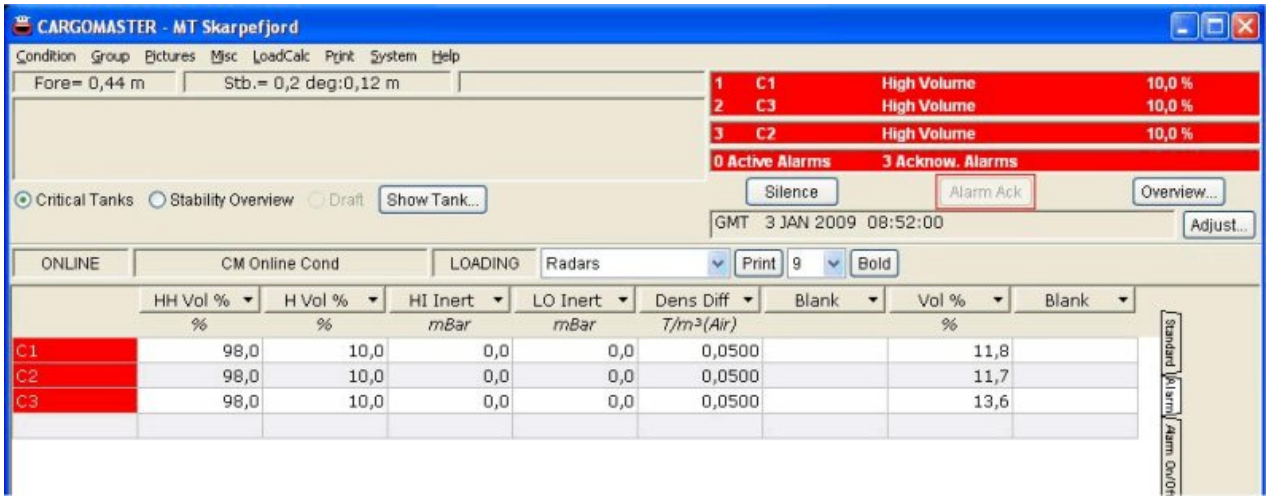


Figure C.6: The Alarm properties together with the current process values for a set of tags in the system. All the tags have acknowledged alarms.

Equivalent is the alarm limits available in the *Alarm* folder or tab as shown in Figure C.5. This figure shows the current alarm settings and the current process value to the right. The current process value is above the alarm limits, the alarm is enabled, so 2 of the process values have activated alarms. The alarm section has changed the color from green to red to indicate the alarm state. The alarm section is showing the alarm information with blinking text, with a duty cycle of 60% on and 40% off to separate the active alarms from the acknowledged alarms. In the figure is the blinking text information in the off duty cycle state. The figure shows 2 active alarms, see the alarm section, and these alarms are also shown with the red color in the user section.

These alarms should activate a beeper, horn and/or lamps to attract the information from the user or users of the system. They should do some action due to these alarms and then acknowledge the alarms. Figure C.6 shows this situation, all the alarms are acknowledged. The alarm section shows only steady alarm texts, no blinking, and the bottom text indicate 0 active alarms and 3 acknowledged alarms. The user sections shows the tags with the alarm limits and the current process values, all above the alarm limits.

The alarms will only be removed from the system when they are passive alarms, meaning:

1. the alarm state is acknowledged,
2. the current process value is within the passive alarm range.

An acknowledged alarm can be an passive alarm in 3 ways:

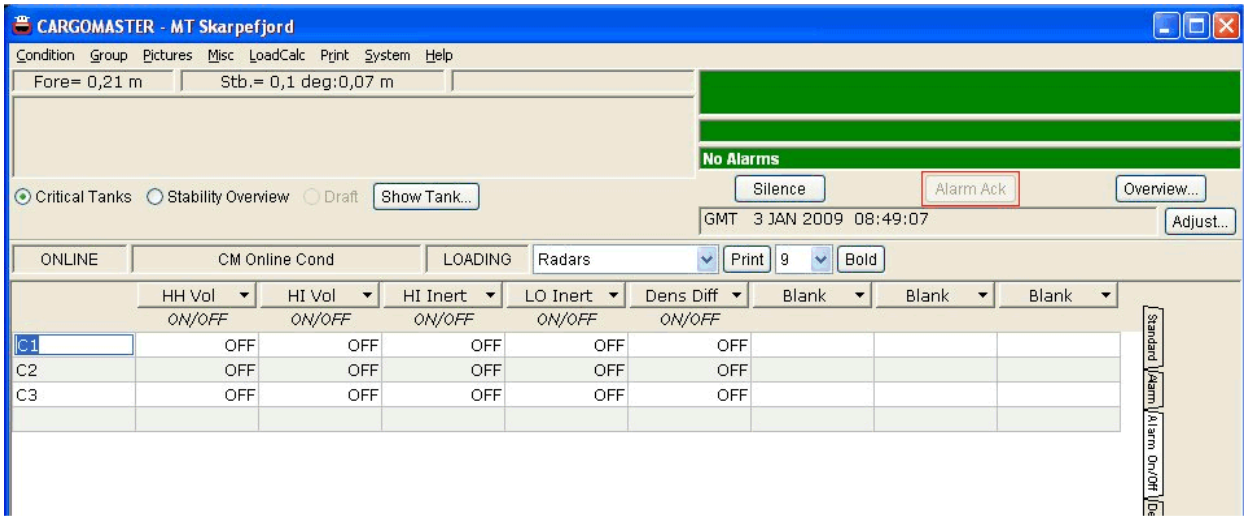


Figure C.7: All the alarms are disabled to turn of the alarm check of the process values.

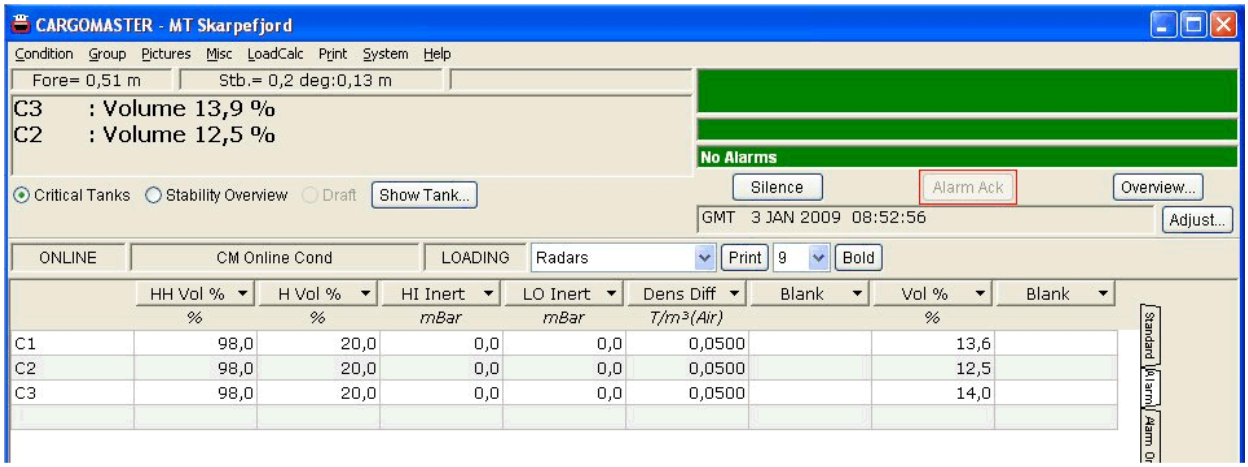


Figure C.8: All the high alarm limits are changed to a value higher than the current process values removing all the acknowledged alarms from the system.

1. the current process value is changing, and changed into the passive range of the alarm settings,
2. the alarm limit is changed so the current process value is within the passive range of the alarm settings,
3. the alarm is disabled.

Figure C.7 shows the situation where all the alarms are disabled, and acknowledged alarms will be removed as the alarm is disabled. In this situation was all the disabled alarms acknowledged so all the alarms are removed from the system and the alarm section is turned into a green color to indicate no alarms.

Figure C.8 shows the situation where the high alarm limits are changed to higher value. The new high alarm limits are all higher than the current process values and the corresponding acknowledged alarms are removed from the system. The alarm section indicate no alarms in the system and is changed to the green color for a more clearly usage of the alarm system.

Appendix D

Test Plan Example

A test plan or a test report must at least consist of three sections, one section describing the setup of the system, one section describing the test cases or check list, and one section documenting by whom the test was performed. This appendix contains copy of small test report for testing the CARGOMASTER® system, a level gauging system used on tank ships world wide, on a specific version of the Windows operating system. The system was approved for Windows 2000 (2K), but needed a new approval for running on Windows XP.

The first page is shown in Figure D.1, describing the type of document. This should always be according the document template of the company, do not make your own lookout or front pages of any reports, use the tempplate.

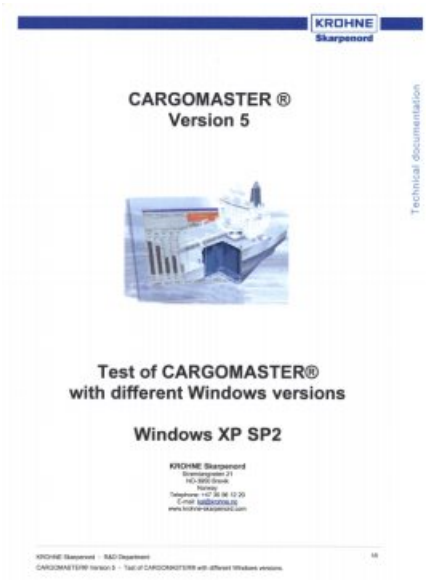
The second page is shown in Figure D.2, showing the preface of the document. The template states that the preface should also include an approval of the document with a version list.

The third page is shown in Figure D.3, containing the List of Contents as defined in the template.

The fourth page is shown in Figure D.4, describing the system requirements and interconnections. The page is divided into two parts, the first part list the requirements for the hardware components in use, and the second part describe the requirements for the test. The requirements are divided into two parts, one specification defining how the components should be interconnected and operation, the second part contains the evaluation and approval of the results.

The last page, page five, is shown in Figure D.5, describing the test cases. According to the test requirements there is a time requirement, the first part is documenting the test period. The second part of the page is the test cases, or check list as used in this document. The check list contains a set of test cased where each test case must be fulfilled to pass the test. The last part contains information by whom the test has been performed. The date, department and name of person doing the test, or being responsible for the test. In this case a note has been added that this has been a minor test of the system just checking the main functionality on a new version of the operating system. The main focus has been to check that the new version of the operating system still supports a defined reliability of the input and output channels and the main system functionality.

Note that this is just an example of a test plan or a test report, a very small and limited test, using the standard template of the company. The example is small enough to shown the typical contents of such a test report and has enough information to let anyone be able to do the same test and hopefully get the same results.



KROHNE
Skarpenord

CARGOMASTER ®
Version 5

Technical documentation

Test of CARGOMASTER®
with different Windows versions

Windows XP SP2

KROHNE Skarpenord
Strømsingveien 21
NO-9950 Grønvik
Norway
Telephone: +47 36 96 12 20
E-mail: kst@krohne.no
www.krohne-skarpenord.com

KROHNE Skarpenord - R&D Department
CARGOMASTER® Version 5 - Test of CARGOMASTER® with different Windows versions.

Figure D.1: The front page of the test plan, according to the document template of the company.

KROHNE
Skarpenord

CARGOMASTER ® version 5		Author date	Nils-Olav Skare (NOS) 21-MAR-06
Document	Test of CARGOMASTER® with different Windows versions	Approved Date	[redacted]
Doc. No.			
Version	1.1	Start date	12-NOV-05
File		Readers	

Technical documentation

Version List

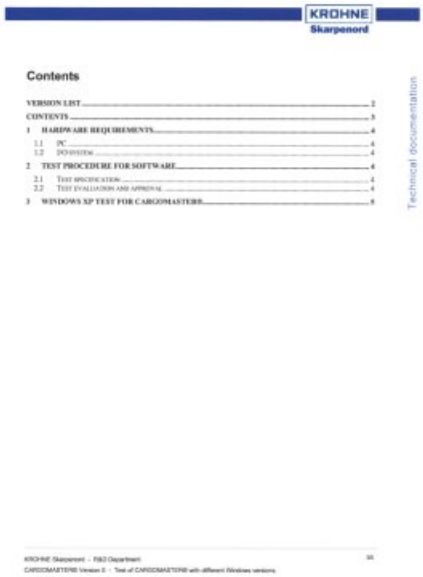
Version	Date	Changes	Changed	Approved
1.0	12-NOV-05	First version of the test document	NOS	
1.1	22-MAR-06	Test of Windows XP SP2 (Service pack 2)	NOS	OAS

Preface

This document specifies the testing of CARGOMASTER® v5 using a new Windows operating system.

KROHNE Skarpenord - R&D Department
CARGOMASTER® Version 5 - Test of CARGOMASTER® with different Windows versions.

Figure D.2: The second page showing the version of the document, according to the template.



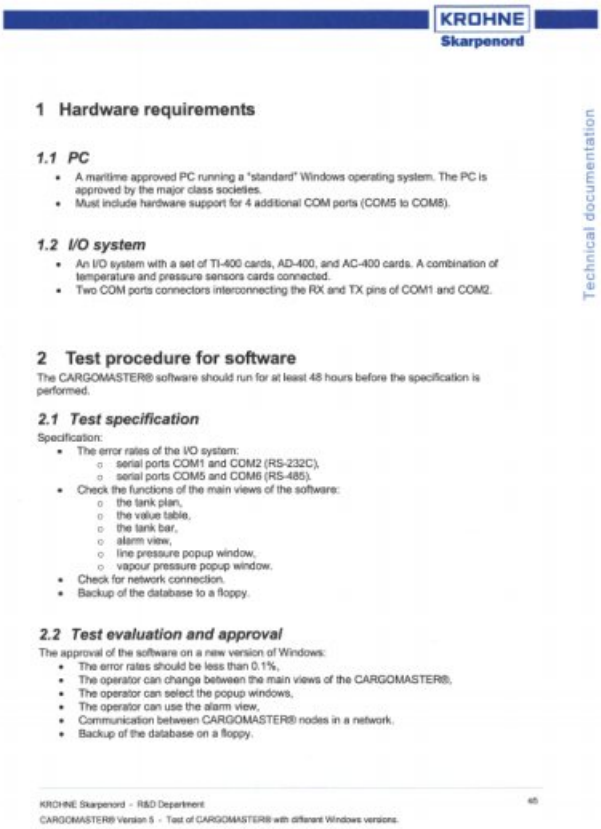
KROHNE Skarpenord

Contents

VERSION LIST 2
CONTENTS 3
1 HARDWARE REQUIREMENTS 4
1.1 PC 4
1.2 I/O SYSTEM 4
2 TEST PROCEDURE FOR SOFTWARE 4
2.1 TEST SPECIFICATION 4
2.2 TEST EVALUATION AND APPROVAL 4
3 WINDOWS XP TEST FOR CARGOMASTER® 5

Technical documentation

Figure D.3: The third page is showing the List of Contents, according to the template.



KROHNE Skarpenord

1 Hardware requirements

1.1 PC

- A maritime approved PC running a "standard" Windows operating system. The PC is approved by the major class societies.
- Must include hardware support for 4 additional COM ports (COM5 to COM8).

1.2 I/O system

- An I/O system with a set of TI-400 cards, AD-400, and AC-400 cards. A combination of temperature and pressure sensors cards connected.
- Two COM ports connectors interconnecting the RX and TX pins of COM1 and COM2.

2 Test procedure for software

The CARGOMASTER® software should run for at least 48 hours before the specification is performed.

2.1 Test specification

Specification:

- The error rates of the I/O system:
  - serial ports COM1 and COM2 (RS-232C),
  - serial ports COM5 and COM6 (RS-485).
- Check the functions of the main views of the software:
  - the tank plan,
  - the value table,
  - the tank bar,
  - alarm view,
  - line pressure popup window,
  - vapour pressure popup window.
- Check for network connection.
- Backup of the database to a floppy.

2.2 Test evaluation and approval

The approval of the software on a new version of Windows:

- The error rates should be less than 0.1%,
- The operator can change between the main views of the CARGOMASTER®,
- The operator can select the popup windows,
- The operator can use the alarm view,
- Communication between CARGOMASTER® nodes in a network,
- Backup of the database on a floppy.

Technical documentation

KROHNE Skarpenord - R&D Department 4/5
CARGOMASTER® Version 5 - Test of CARGOMASTER® with different Windows versions.

Figure D.4: Page four contains the system descriptions, containing the hardware requirements and the system eqiupment and interconnections.

KROHNE Skarpenord

Technical documentation

3 Windows XP test for CARGOMASTER®

1	The time period		
The test vas performed during : Nils-Olav Skeie			
Start date:	17-MAR-06	Start time:	13:30
End date:	20-MAR-06	End time:	10:15

4	Checklist	
Verify that the test is performed according to the specification.		
Windows version used in the test:	**Windows XP SP2 (Service Pack 2)**	
		Approved
The test was run for more than 48 hours before performing the check of the specifications.		OK / NOS
Communication status on COM1 and COM2 are aceptable?		Yes / NOS
Communication status on COM5 and COM6 are aceptable?		Yes / NOS
Network connection status OK?		Yes / NOS
Change between tank plan, tank value, and tank bar view is working?		Yes / NOS
Use of the popup windows		OK / NOS
Change of alarms in alarm view		OK / NOS
Check of the database backup, floppy content OK?		Yes / NOS

6	Tested by
Name and department of the person(s) performing the test.	
Comments: The functions and operations of the CARGOMASTER® system is the same using Windows XP as the operating system on a type approved computer as using the Windows 2000 operating system.	
21-MAR-06 : R&D department : Nils-Olav Skeie (NOS)	

KROHNE Skarpenord - R&D Department 55

CARGOMASTER® Version 5 - Test of CARGOMASTER® with different Windows versions.

Figure D.5: The last page, page five, contains the timing and check list (test cases), and the tested by section.

Bibliography

3COM (2001), 'Understanding ip addressing: Everything you ever wanted to know', White Paper.

Andersen, L. M. & Bakkeli, M. (2015), Big data - Hva er big data, og hva betyr big data for deg? (in Norwegian), Technical report, Pwc Consulting.

Assev, S. M. & Mikalsen, A. B. (2012), *Drift av lokalnettverk - Design og sikkerhet (in Norwegian)*, Tanum.

Bach, M. J. (1986), *The Design of the Unix Operating System*, Prentice-Hall.

Barr, M. (2016), Mutexes and semaphores demystified, Technical report, BARR group.

Bentham, J. (2002), *TCP/IP Lean, Web servers for embedded systems*, second edn, CMPBooks.

Bentley, J. P. (2005), *Principles of Measurement Systems*, 4th. edn, Pearson Education Limited, Essex, England.

Bishop, R. H. (2008), *Mechatronic systems, sensors, and actuators; Fundamentals and Modeling*, CRC Press. ISBN 0-8493-9258-6.

Bonnema, G. M., Veenvliet, K. T. & Broenink, J. F. (2015), *Systems Design and Engineering Facilitating Multidisciplinary Development Process*, Taylor & Francis Inc (CRC Press).

Bormann, A. & Hilgenkamp, I. (2006), *Industrial Networks; Ethernet Communication for Automation Applications*, Phoenix Contact GmbH, Blomberg, Germany. ISBN 3-7785-2950-1.

Brown, M. (2010), *Power Electronics in Motor Drives, Principles, Application and Design*, elektor, Elektor International Media BV, The Netherlands.

Bullemer, P., Reising, D. V., Burns, C., Hajdukiewicz, J. & Andrzejewski, J. (2008), *Effective Operator Display Design*, ASM Consortium. ISBN 978-1440431647.

Campbell, J. (1984), *The RS-232 Solution*, Sybex, USA. ISBN 0-89588-140-3.

Caro, R. H. (2004), *Automatiom Network Selection*, ISA-The Instrumentation, Systems, and Automation Society.

Carvalho, T. P., Soares, F. A. A. M. N., Vita, R., da P. Francisco, R., Basto, J. P. & Alcala, S. G. S. (2019), 'A systematic literature review of machine learning methods applied to predictive maintenance', *Computers and Industrial Engineering* **137**.

Cheswick, W. R., Bellovin, S. M. & Rubin, A. D. (2003), *Firewalls and internet security*, 2nd edn, Addison-Wesley.

Columbus (2019), 'Digitalisering i praksis - derfor bør du starte med erp-systemet', Whitepaper (in Norwegian). www.columbusglobal.no.

Comer, D. E. (1991), *Internetworking with TCP/IP; Volume I; Principles, Protocols and Architecture*, second edn, Prentice-Hall International Editions.

Company (2010), Explosion protection; theory and practice, Technical report, Phoenix Contact.

Cravotta, R. (2008), 'Sensor-rich designs', *EDN Europa* pp. 24–28. www.edn-europa.com.

Crouse-Hinds, C. (2014), Intrinsic safety solutions protecting investments in hazardous areas worldwide, Technical report, MTL Instruments.

Crowcroft, J. & Phillips, I. (2002), *TCP/IP and Linux Implementation; Systems code for Linux Internet*, Wiley.

Daler, T., Gulbrandsen, R., Høie, T. A. & Sjølstad, T. (2019), *Håndbok i datasikkerhet (in Norwegian)*, 4 edn, Fagbokforlaget. ISBN 978-82-450-2789-1.

Dickinson, M. (2013), *Process MeasureApplications C# Applications*, Elektor International Media B.V. ISBN 978-1-907920-24-0.

Douglass, B. P. (1999), *Doing Hard Time Developing Real-Time Systems with UML, OObject, Frameworks, and Patterns*, Addison Wesley.

EEMUA (1999), Eemua publication 191 alarm systems - a guide to design, management and procurement, Technical report, EEMUA.

Elm (2016), *ELM327 - OBD to RS232 Interpreter*. Accessed FEB-2019.

Elms, M. (2012), 'Comparing uninterruptible power supply topologies', *Electronic Engineering Times Europa* **July/August**, 20–22. Part I of III.

Emerson (2016), Opc alarms and events overview, Technical report, Emerson Process Management.

Esbensen, K. H. (2002), *Multivariate Data Analysis - in Practice*, 5th edn, Camo, Nedre Vollgt. 8, N-0158 Oslo, Norway.

exida (2017), The Alarm Shelving eBook, Technical report, exida.

Fortinet (2018), '2018 security implications of digital transformation report:'.

Fortuna, L., Graziani, S., Rizzo, A. & Xibilia, M. (2007), *Soft Sensors for Monitoring and Control of Industrial Processes*, Springer, London, UK. ISBN 1-84628-479-1.

Fowler, M. & Scott, K. (1997), *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley, USA.

Fuglset, S. (2003), 'Lagring av høyoppløselige produksjonsdata, hvilken database skal jegvelge?', *Industridata* **3**(3), 19–21. In Norwegian.

Furenes, B. (2009), 'Methods for batch-to-batch optimization of parallel production lines', PhD Trial lecture.

Gallmeister, B. O. (1995), *POSIX. 4: Programming for the Real World*, O'Reilly & Associates, Inc. ISBN 1-56592-074-0.

Gamer, T., Wallstadter, M., Kloepper, B., Bauer, R. & Isaksson, A. J. (2019), 'The autonomous industrial plant - Future of process engineering, operations and maintenance', *International Federation of Automatic Control (IFAC)* **52**, 6.

Gamey, J. (2000), Universal serial bus specification, Technical report, www.usb.org. Revision 2.0.

Gofton, P. W. (1986), *Mastering Serial Communications*, Sybex. ISBN 0-89588-180-2.

Gornik, D. (2003), Relational modeling with uml, Technical report, IBM (Rational Software).

Grey, W. (2020), *FreeRTOS for ESP32-Arduino; Practical multitasking fundamentals*, Elektor International Media B.V.

Hall, D. L. & Llinas, J. (2001), *Handbook of Multisensor Data Fusion*, CRC Press, Florida, USA.

Hall, D. L. & McMullen, S. A. (2004), *Mathematical Techniques in Multisensor Data Fusion*, second edn, Artech House, Inc., 685 Canton Street, Norwood, MA02062, USA.

Hashemian, H. M. & Bean, W. C. (2011), 'State-of-theart predictive maintenance techniques', *IEEE Transactions on Instrumentation and Measurement* **60**(10), 3480 – 3492.

Haugen, F. (2008), Derivation of a discrete-time lowpass filter, Technical report, TechTeach. techteach.no.

Haugen, F. A. (2014), *Reguleringsteknikk (in Norwegian)*, 2nd edn, Fagbokforlaget AS.

Hollifield, B., Oliver, D., Nimmo, I. & Habibi, E. (2008), *The High Performance HMI Handbook*, PAS, Houston, USA.

Hollifield, B. R. & Habibi, E. (2010), *The Alarm Management Handbook, Comprehensive Guide*, second edn, PAS, Houston, USA.

Houldsworth, J. (1990), *OSI Handbook*, International Computers Ltd. (ICL), UK.

Ibrahim, D. (2011), *Control Area Network Projects*, Elektor International Media B.V.

Ibrahim, D. (2020), *Multitasking with Raspberry Pi*, Elektor International Media B.V.

IEC60812 (2006), Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA), Technical report, International Electrotechnical Commission (IEC). Second edition.

IEC62682 (2014), Iec62682: Management of alarms systems for the process industries, Technical report, International Electrotechnical Commission (IEC).

Ifeachor, E. C. & Jervis, B. W. (2002), *Digital Signal Processing, A Practical Approach*, 2nd. edn, Pearson Education Limited, Essex, England.

Int (1988), *386 Microprocessor, High performance 32-bit CHMOS microprocessor with integrated memory management.*

ISA (1992), Instrumentation symbols and identification (ansi/isa-s5.1-1984 (r 1992)), Technical report, American National Standard (ANSI).

ISA (2016), Ansi/isa-18.2-2016, management of alarm systems for the process industries, Technical report, ISA.

ISA-62443 (2018), Standards for security capabilities for control system components, Technical report, International Society of Automation (ISA).

Isaksson, A. J., Harjunkoski, I. & Sand, G. (2018), 'The impact of digitalization on the future of control and operations', *Computers and Chemical Engineering* **114**, 122–129.

Kadal, J. (2018), Data science and machine learning in an industrial context, Technical report, DNV GL AS, Hovik, Norway.

Kapurch, S. J. (2007), Systems engineering handbook, Technical Report NASA/SP-2007-6105, rev.1, National Aeronautics and Space Administration, NASA Center for AeroSpace Information, 7115 Standard Drive, Honover, MD 21076-1320, USA.

Kirrmann, H. (2007), OLE for process control (OPC), Technical report, ABB research Centre, Baden, Switzerland. Industrial Automation, OPC, Data Access Specification.

Klatt, T. (2006), Explosion protection; intrinsic safety, Technical report, Pepperl+Fuchs.

Kolda, T. & Torczon, V. (2011), 'Top ten ways to lose an audience', *SIAM News* **44**(3).

Kossiakoff, A. (2003), *Systems Engineering Principles and Practice*, John Wiley & Sons.

Krishna, C. & Shin, K. G. (1997), *Real-Time Systems*, The Mc Graw Hill Companies, Inc. ISBN 0-07-114243-6.

Kristoffersen, B. (2009), *Databasesystemer*, 2. edn, Univeristetsforlaget, PoBox 508 Sentrum, 0105 Oslo, Norway. Norwegian, ISBN 978-82-15-01522-4.

Krogh, E. (2005), *OPC Seminar*, Prediktor AS, Fredrikstad, Norway. www.prediktor.no (feb-07).

Kruchten, P. (2000), *The Rational Unified Process (UP) An Introduction*, second edn, Addison-Wesley, USA.

Kuhn, M. & Johnson, K. (2016), *Applied Predictive Modeling*, Springer.

Labrosse, J. J. (1999), *MicroC/OS-II: The Real-Time Kernel*, Miller Freeman, Inc. ISBN: 0-87930-543-6.

Levermore, G. J. (2000), *Building Energy Management Systems Applications to Low-Energy HVAC and Natural Ventilation Control*, second edn, E and FN Spon.

Mackay, S., Wright, E., Park, J. & Reynders, D. (2004), *Practical Industrial Data Networks; Design, Installation and Troubleshooting*, Elsevier (Newnes), Oxford, UK.

Magda, Y. (2009), *Complete Practical Measurement Systems Using a PC-based*, elektor. ISBN 978-0-905705-79-8.

Magda, Y. (2011), *Linux PC-based Measurement Electronics, hardware and software*, elektor. ISBN 978-1-907920-03-5.

Mahmoud, Q. H. (2004), *Middleware for Communications*, Wiley, West Sussex PO19 8SQ, England.

Marsland, S. (2015), *Machine Learning, an algorithmic perspective*, second edn, CRC Press.

Mazonka, O. (2016), Blockchain: Simple explonation, Technical report, Defence Science and Technology Group (DST) Canberra, Australia Join this Institution.

McCall, G., Burden, P., Burnard, A., Hennell, M., Hills, C., Montgomery, S., Tapp, C. & Whiting, L. (2008), Misra-c: Guidelines for the use of the c language in critical systems, Technical report, MISRA (The Motor Industry Software Reliability Association). 2. Ed.

McNamara, J. E. (1988), *Technical Aspects of Data Communication*, Digital Equipment Corporation, USA.

Meier, F. A. & Meier, C. A. (2011), *Instrumentation and Control Systems Documentation*, second edn, International Society of Automation (ISA).

Meier, P. C. & Zünd, R. E. (2000), *Statistical Methods in Analytical Chemistry*, second edn, John Wiley and Sons, Inc., New York, NY 10158-0012, USA.

Miller, L. C. (2011), *Next-Generation Firewalls for Dummies*, Wiley Publishing, Inc.

Mohan, N., Undeland, T. M. & Robbins, W. P. (1989), *Power Electronics: Converts, Applications, and Design*, John Wiley & sons, Inc. ISBN 0-471-50537-4.

Muller, G. (2012), *Systems Architecting A business perspective*, CRC Press.

Murphy, E. (2009), Opc ua: 5 things everyone needs to know, Technical report, Matrikon, Alberta, Canada.

Murphy, R. R. (2019), *Introduction to AI robotics*, second edn, The MIT press.

Müller, M., Müller, T., Talkhestani, B. A., Marks, P., Jazdi, N. & Weyrich, M. (2021), 'Industrial autonomous systems: a survey on definitions, characteristics and abilities', *Automatisierungstechnik* **69**, 10.

Novacek, G. (2010), 'My analog world the significance of grounding', Circuit Cellar magazine, issue 244.

Næs, T., Isaksson, T., Rearn, T. & Davis, T. (2004), *Multivariate Calibration and Classification*, NIR Publications, Chichester, UK.

Olsson, G. & Piani, G. (1998), *Computer Systems for Automation and Control*, 2nd edn, Prentice Hall International (UK) Ltd., London, UK.

Olsson, G. & Rosen, C. (2003), *Industrial Automation - Application, Structures and Systems*, Lund University, Lund, Sweden.

Parnell, G. S., Driscoll, P. J. & Henderson, D. L. (2012), *Decision Making in Systems Engineering and Management*, 2nds edn, Wiley. ISBN 978-0-470-90042-0.

Pearson, R. K. (2005), *Mining Imperfect Data, Dealing with Contamination and Incomplete Records*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, USA.

Pettersen, O. (1984), *Sanntidsprogrammering for Prosess-Styring (in Norwegian)*, 4th edn, Tapir forlag, Trondheim, Norway. ISBN 82-519-0263-0.

Pho (2010), *Explosion protection Theory and Practice.*

PROFIsafe (2009), 'Pc-based, but safe!', *Control Engineering Europe* **November/December**, 28–30. www.controlengeurope.com.

Rausand, M. & Høyland, A. (2004), *System Reliability Theory, Models, Statistical Methods, and Applications*, 2nd edn, Wiley Interscience, John Wiley and Sons, Inc. Hoboken, New Jersey, Usa.

Regjeringen.no (n.d.), 'Digitalisering i offentlig sektor', Accessed 17-FEB-19.

Ripps, D. L. (1989), *An implementation guide to real-time programming*, Prentice-Hall, Inc.

Schultz, T. W. (1999), *C and the 8051, Building Efficient Applications*, Prentice Hall PTR, New Jersey, USA.

Simon, D. E. (1999), *An Embedded Sofware Primer*, Addison-Wesley, USA.

Simon, P. (2013), *too BIG to IGNORE - The business case for big data*, Wiley and SAS Business Series.

Skeie, N.-O. (2008), Soft Sensors for Level Estimation, PhD thesis, The Norwegian University of Science and Technology (NTNU).

Skrivervik, H. (2000), Nettverksarkitektur for the 21. århundre (norwegian), Technical report, Team Mellvik as.

Stallings, W. (2001), *Wireless communication and networks*, Prentice-Hall International Editions.

Standards (2010), BS EN 13306 Maintenance - Maintenance terminology, Technical report, British Standard (BS).

Stenerson, J. (2003), *Industrail Automation and Process Control*, Pearson Education. ISBN 0-13-033030-2.

Stevens, W. R. (1990), *Unix networking programming*, Prentice-Hall International Editions.

Stouffer, K., Pillitteri, V., Lightman, S., Abrams, M. & Hahn, A. (2015), Guide to Industrial Control Systems (ICS) Security, Technical report, National Institute of Standards and Technology (NIST).

Sun, R. (2016), Improved alarm system for cyberbase system from national oilwell varco, Master's thesis, University College of Southeast Norway.

Tanenbaum, A. S. & Woodhull, A. S. (1997), *Operating systems: design and implementation*, second edn, Prentice Hall, US.

Tektronix (2013), Wi-fi: Overview of the 802.11 physical layer and transmitter measurements, Technical report, Tektronix.

Tufféry, S. (2011), *Data mining and statistics for decision making*, John Wiley and Sons.

VentureCom (1993), *RealTime Programming Guide & References*, VetureCom, USA. RTOS for Venix SVR4.2.

Walls, C. (2006), *Embedded Software*, Elsevier.

Wasson, C. S. (2006), *System Analysis, Design, and Development Concepts, Principles, and Practice*, Wiley-Interscience. ISBN 978-0-471-39333-7.

Watson, R. T. (2002), *Data Management Databases and Organizations*, third edn, John Wiley & Sons, Inc. ISBN 0-471-41845-5.

Westerman, G., Bonnet, D. & McAfee, A. (2014), *Leading Digital - Turning technology into business transformation*, Harvard business review press, Boston, Massachusetts.

Wheeler, A. J. & Ganji, A. R. (2004), *Introduction to Enineering Experimentation*, 2. edn, Pearson, USA.

Whitten, J. L. & Bentley, L. D. (1998), *Systems Analysis and Design Methods*, fourth edn, McGrawHill.

Wiener, N. (1961), *Cybernetics or Control and Communication in the Animal and the Machine*, second edition edn, The MIT press. ISBN: 9780262230070.

www.matrikon.com (2007), 'Matrikon'. OPC information.

www.wikipedia.org (2006), 'Wikipedia'. Wikipedia, the free encyclopedia.

www.wikipedia.org (2010), 'Wikipedia'. Wikipedia, the free encyclopedia.

YA-710 (2001), Prinsipper for utforming av alarmsystemer (ya-710)(in Norwegian), Technical report, Oljedirektoratet.

YA-711 (2001), Principles for alarm system design (ya-711), Technical report, Norwegian Petroleum Directorate.

Yehia, S. (2016), 'Choosing between a plc and dcs for proces applications', *Control Engineering Europe* pp. 12–13.

Yokogawa (2018), 'Yokogawa's synaptic business automation oprex profit-driven operation digitizing plant data to generate sustained future value', White paper.

Index

6LoWPAN, 147

a.c. motor, 315
accelerated testing, 498
accuracy, 335
adapter
    USB to serial port adapter, 197
ADC
    integrating, 333
    sigma-delta, 332
    successive-approximation, 332
advanced encryption standard
    AES, 299
ageing, 10
alarm colors, 386
alarm levels, 385
alarm list, 395
alarm management, 376, 380
alarm philosophy, 381, 393
alarm priority, 386
alarm states, 387
alarm system, 376
alarm types, 385
antivirus, 292
antivirus software, 301
API, 95
application programming interface
    API, 193
    api, 55
architecture
    logical, 527
    physical, 527
    reference, 527
arduino, 448
array buffer, 530
artificial intelligence, 76
AS-Interface, 239
ASCII, 326
asymmetric algorithm
    cryptic, 295
asynchronous communication, 195
at commands
    attention commands, 200
ATA
    AT Attachment, 128
ATEX, 490
    100a, 476
    118a, 476
    137, 476
    ATmosphères EXplosibles, 476
    equipment directive, 476
    workplace directive, 476
ATEX directives, 481
augmented reality, 76
automatic electric meter system
    AMS, 60, 73
automation
    factory, 236
    factory automation, 14
    process, 236
    process automation, 14
    process control, 14
automation system, 71
autonomous system, 46, 78
    automation versus autonomous, 46
    building blocks, 47

BAS, 180
    Building Application Systems, 256
batch, 276
beacon
    iBeacon, 147
beidou
    bds
        gps, 178
BEMS, 180
big data, 64
big endian, 325, 474
BIOS
    basic input / output system, 462
bit coin mining, 295
bitcoin, 295
block chains
    cryptic, 295
bluetooth, 146, 151
    low energy, 147
    SMART, 147
botnet, 303
bridge, 101
broadcast, 93
    IP address, 114
broadcast address, 109
buffer
    array, 530
    list, 530
    queue, 530
    ring, 530
    stack, 530

building automation system, 180
building energy management system, 180
bytes, 324

CAN bus, 240
    Controller Area Network, 251
CAT cable, 104
channel skew, 341
check
    limit, 338
    redundancy, 338
    validation, 338
ciphertext, 294
classic OPC, 268
cloud, 57
cloud based services, 192
    IaaS, 192
    PaaS, 192
    SaaS, 192
cloud based types, 192
cloud computing stack, 192
cloud system, 191
coding standard, 530
COM
    Component Object Model, 260
combustion, 478
comma separated values
    CSV, 179
confidential
    protection level, 293
context, 407, 435
context diagram, 527
control loop
    scheduler, 465
control system
    ICS, 21
    PCS, 21
    SCADA, 21
controller area network
    CAN bus, 200
Controlnet, 248
coordinated universal time, 122
CRC
    Cyclic Redundancy Check, 137
critical region, 422
crypt, 217
cryptic
    advanced encryption standard (AES), 294
    asymmetric algorithm, 295
    block chains, 295
    data encryption standard (DES), 294
    pgp, 295
    public keys, 295
    rsa, 295
    symmetric algorithm, 294
CSMA/CD, 106
CSV
    comma separated values, 179
current loop, 140
cyber-physical system, 75
cybernetic, 2

d.c. motor, 315
daq, 536
data acquisition system, 321
data backup, 294
data encryption, 294
data engine, 28
data erasure, 294
data fusion, 65
data masking, 294
data structure, 530
database, 55, 354
    management system (DBMS), 359
    relation, 367
    relational, 55
    schema, 359
    SQL, 367
    time serie, 367
    time series, 55, 365
database manager, 55
database model, 359
DCOM, 260
DCS, 21, 22, 34, 37, 84, 284
    distributed control system, 3
    distributed controller system, 21
DDE
    Dynamic Data Exchange, 260
deadband, 273
deadlock, 420
debugging, 461
decision support system, 28
decrypt, 217
decryption
    encryption, 294
defragmentation, 508
demilitary zone
    DMZ, 296
denial of service
    DoS, 81, 303
design, 11
development, 11
device driver, 469
DeviceNet, 254
DHCP, 115
    169.254.0.x, 117
    dynamic bindings, 116
    network configuration, 115
    static bindings, 116
digital twin, 75
distributed control system
    DCS, 3, 33
distributed controller system
    DCS, 21
distributed denial of service
    DDoS, 303

DMA, 338
DMZ, 101
    demilitary zone, 296
dns
    Domain Name Server, 119
domain, 120
dynamic routing, 119

E/R diagram, 371
edge computing, 70
edge device, 70
EIB, 240
electronic mail
    email, 123
elm327 interface
    ELM327, 200
email, 123
    IMAP, 123
    POP3, 123
    SMTP, 123
email address, 123
email protocol, 123
Embedded operating system
    operating system, 469
Embedded system
    Control loop, 465
embedded systems, 457
encryption
    decryption, 294
endian
    big, 180, 474, 531
    little, 180, 474, 531
enterprise resource planning
    ERP, 3, 22, 38
entity relationship model, 372
EOBD
    obd, 200
ERP
    enterprise resource planning, 3
ethernet, 102, 151
Ethernet/ip, 255
event, 416, 435
Ex
    ATEX, 481
    BSI, 481
    CENELEC, 481
    IEC, 481
extensible markup language
    XML, 63

factory acceptance testing
    FAT, 497
factory automation, 14
failure mode and effects analysis
    FMEA, 503
FDDI, 254
FEC
    forward error correction, 135
Fiber Distributed Data Interface
    FDDI, 254
fibre, 143
field bus, 232
fieldbus
    Fieldbus Foundation, 246
    Interbus, 250
    Profibus, 242
    PROFINET, 247
Fieldbus Foundation, 246
FIFO, 530
fifth generation network
    5G, 151
file format
    csv, 54
    docx, 54
    pdf, 54
    xlsx, 54
    xml, 54
filter
    band-pass, 345
    band-stop, 345
    Bessel, 345
    Butterworth, 345
    Chebyshev, 345
    exponential weighted moving average, 349
    finite impulse response, 349
    high-pass, 345
    infinite impulse response, 349
    limit check, 338
    low-pass, 345
    moving average, 349
    redundancy check, 338
    validation check, 338
finite impulse response filter
    FIR filter
        filter, 531
firewall, 101, 121, 186, 292
FireWire, 160
fog computing, 70
fragmented, 508
FSK, Frequency Shift Keying, 162
full bridge controller
    H bridge controller, 318
full duplex, 89

galileo
    gps, 178
galvanic isolation, 141
    active, 142
    passive, 142
gateway, 101
GDPR, 373
gdpr, 64
    general data protection regulation, 294
general purpose operating system
    gpos, 448
global positioning system, 178
glonass

gps, 178
gps, 178
beidou
bds, 178
galileo, 178
global position system, 196
glonass, 178
NMEA-0183, 178
Graphical User Interface (GUI), 24
grounding, 529

HAL
Hardware Abstraction Layer, 450
half duplex, 89
hardware in the loop (HIL), 499
HART, 146
HART-IP, 164
Highway Addressable Remote Transducer, 161
WirelessHART, 164
hazardous areas, 478
highly accelerated life testing (HALT), 500
host platform, 474
hot plugging, 235
hub, 101
Human-Machine Interface (HMI), 24
hybrid cloud system
cloud - hybrid system, 192
hyper text transfer protocol secure
HTTPS, 307

IANA
The Internet Assigned Numbers Authority, 112
ICS
industrial control system, 21
ICT
information and communications technology, 290
IDE
Integrated Drive Electronics, 128
IEEE 802, 92, 93, 100
IEEE 802.3
ethernet, 102
IEEE 802.5, 108
IIoT, 284
IMEI
international mobile equipment identity, 173
imrad
article, 542
industrial control system
ICS, 296
ics, 4
Industrial Ethernet, 255
industrial internet of things
IIoT, 66, 193
infinity impulse response filter
IIR filter
filter, 531
information and communications technology
ICT, 290
information management system
IMS, 22
information sanagement system
IMS, 39
information technology
IT, 71, 81, 296, 304
infrastructure as a service
IaaS, 58, 192
ingress protection (IP), 490
input loading, 529
input signal
ADC, 330
analog, 330
differential (DI), 337
digital, 327
single ended (SE), 335
integrated operation, 42
integrated operations, 43
Interbus, 250
internet message access protocol
IMAP, 123
internet of things
IoT, 66, 170, 180, 193, 297
protocols, 170
interprocess communication
IPC, 416
interprocess communication (IPC)
events, 417
message queue, 417
pipe, 417
shared memory, 417
interrupt, 338, 429, 435
latency, 435
service routine, 435
interrupt service routine
ISR, 410
intrinsic safety, 484
IoT, 216, 284
Internet of Things, 501
internet of things, 117, 147, 148, 164, 180, 297
IP coding
IP rating, 490
IPv4, 117
IPv6, 117
isa
International Society of Automation, 28, 357

javascript object notation
JSON, 64, 179

kpi
key performance indicator, 391

LAN, 97
virtual, 124
learning
reinforcement learning, 504
supervised learning, 504
unsupervised learning, 504

lidar
light detection and ranging, 49
life extension, 10
LIFO, 530
list buffer, 530
little endian, 325, 474
LON works, 250
loopback testing, 197

MAC, 100
MAC address, 93, 100, 101, 108, 116
machine learning, 77, 503
machine to machine communication
M2M, 151
maintenance
corrective (CM), 503
predictive (PdM), 503
preventive (PvM), 503
run to failure (R2F), 503
malware
phishing, 302
trojan, 302
virus, 301
worm, 302
malware software, 301
Man-Machine Interface (MMI), 24
manufacture execution system
MES, 22
manufacturing execution systems
MES, 38
manufacturing operations management
MOM, 39
MCS
master control system, 37, 284
MDIS network, 36, 284
mean time between failures, 19
meeting
minutes of meeting, 543
notice of meeting, 543
middleware, 184
.NET, 185
CORBA, 185
DCOM, 185
file, 184
message queue, 185
named pipes, 185
RPC, 185
shared memory, 185
SOAP, 185
socket, 185
XML, 185
MMS, 177
mobile
authentication centre (AUC), 172
base station controller (BSC), 171
base transceiving station (BTS), 171
enhanced data rates for GSM evolution (EDGE), 172
equipment identity register (EIR), 172
general packet radio service (GPRS), 172
home location register (HLR), 172
international mobile equipment identity (IMEI), 171
international mobile subscriber identity (IMSI), 171
subscriber identity module (SIM), 171
visitor location register (VLR), 171
Modbus, 165, 170, 240, 255
application data unit (ADU), 165
Big Endian, 167
Modbus Application Protocol (MBAP), 167
protocol data unit (PDU), 165
RTU, 165
TCP, 165
model
client/server, 265
instrumentation amplifier, 528
publisher/subscriber, 265
modulation
ASK, 139
FSK, 139
modem, 139
PSK, 139
module testing, 496
MQTT, 171
MTTF
mean time to failure, 18
multicast, 93
IP address, 114
multimedia messaging service, 177
multitasking, 402, 435
multitasking examples in C#
semaphores, 440
threading, 437
trafic light controller, 439
multitasking system, 399
mutex, 412

N2 diagram, 530, 534, 535
NAS
network attached storage, 509
NAT
Network Address Translator, 115
negative testing, 498
network configuration
DHCP, 115
network operating system
nos, 448
network protocol
LLDP, 123
RSTP, 123
SNMP, 123
network time protocol
NTP, 122
network topology, 97
NMEA 183

gps, 197
NMEA-0183
gps, 178
node, 87, 102
non preemptive multitasking
scheduler, 467

OBD, 253
OBD-2
obd, 200
OBD-II
obd, 200
ODBC, 365
On-board diagnostic
obd, 200
OPC
classic, 268
Open Process Control, 260
open process control, 170
OPC AE, 268, 270, 274
OPC CD, 268
OPC classic, 282
OPC DA, 268, 270, 271
OPC DX, 268, 275
OPC Express Interface (Xi), 268
Express Interface (Xi), 285
OPC HDA, 268, 270, 276
OPC UA, 268, 282
OPC Unified Architecture (UA)
Unified Architecture (UA), 281
operating system, 401
os, 462, 469
operational technology
OT, 23, 71, 81, 296, 305
optic fibre, 143
OSI model, 195, 214, 232
application layer, 94, 95
data link layer, 93
network layer, 93
OSI reference model, 90
physical layer, 91
presentation layer, 95
transport layer, 94
outlier, 343
output loading, 529
output signal
analog, 328
DAC, 328
digital, 327

p&id
piping and instrumentation diagram, 28
PAC, 21, 22, 84
programmable automation controller, 3
parallel port
Centronics, 125
IDE, 128
SCSI, 127
password, 300
most used, 300
tips, 300
PATA, 130
PCS
process control system, 3, 21
peer to peer, 247
perception module, 51
pgp
cryptic, 295
pretty good privacy, 295
phishing, 302
platform as a service
PaaS, 58, 192
PLC, 22, 34, 84
programmable logic controller, 3, 21
polling, 338, 429
positive testing, 498
POSIX, 448
Posix.4
Programming, 437
post office protocol
POP3, 123
power debugging, 464
precision, 335
precision time protocol
PTP, 122
predictive maintenance, 75
preemption, 435
preemptive multitasking
scheduler, 467
priority, 431, 435
priority inheritance, 432
priority inversion, 431
private cloud system
cloud - private type, 192
process automation, 14
process control, 14
process control system, 71, 262
PCS, 3, 71, 296
pcs, 4
Profibus, 242
PROFINET, 247
programmable automation controller, 21
PAC, 3, 34
programmable logic controller
PCS, 3
PLC, 21, 32
proportional integral derivative controller
PID controller, 34
protocol, 57, 87, 88, 260, 326
ASCII, 88
CSMA/CD, 418
Ethernet POWERLINK, 418
FireWire, 160
Hart, 161
master/slave, 88
Modbus, 165
MQTT, 171

multiple access, 89
output, 88
output with check, 88
peer to peer, 89
Profinet IRT, 418
selection, 180
server/client, 88
SOAP, 170
token ring, 88, 417
USB, 154
Wireless HART, 164
XML, 169
protocol stack, 90, 109
protocol suite, 90, 109
proxy, 124
pseudo coding, 30, 413, 465, 466, 526
public cloud system
cloud - public system, 192
public keys
cryptic, 295
publisher/subscribe, 224
pwm, 313
pulse width modulation, 318

quality of service (QoS), 93, 147
queue buffer, 530

RAID, 509
ransomware, 302
raspberry pi, 448
real-time
deadline, 434
definition, 434
event, 435
interrupt, 435
interrupt latency, 435
interrupt service routine (ISR), 435
maximum time, 399
multitasking, 435
preemption, 435
priority, 435
resource, 434
scheduler, 435
simultaneousness, 434
real-time operating system
rtos, 448
real-time system, 399
record, 359
redundancy, 19
redundant array of independent disks
RAID, 509
reference voltage, 335
reinforcement learning, 504
remote terminal unit
RTU, 22, 32
repeater, 101
report
article, 542
formal report, 541
imrad, 542
informal report, 541
paper, 542
research, 11
resolution, 335
resource, 420
ring buffer, 530
RIR
regional Internet registries, 112
roaming, 177
international, 177
national, 177
robotic process automation
RPA, 36
Round Robin
scheduler, 466
Round Robin with interrupt
scheduler, 467
router, 94, 101
routing, 119
routing table, 119
RPA
robotic process automation, 36
RS-232C, 84, 132
rs-232c, 195
RS-422, 132
rs-422, 195
RS-485, 84, 104, 132, 134
rs-485, 195
RTOS requirement, 448
RTU
remote terminal unit, 22, 32

sae
society of automotive engineers, 49
safety and automation system
SAS, 39
safety barriers, 484
galvanic isolation type, 485
zener diode type, 485
Sample and Hold (S/H), 340
sampling
aliasing, 342
SATA
Serial ATA, 152
satellite navigation system, 178
SCADA, 15, 21, 24, 28, 31, 32, 84, 262
supervisory control and data acquisition, 3, 21
scheduler, 399, 406, 435
strategies, 410
schema, 359
SCSI
Small Computer System Interface, 127
secret
protection level, 293
secuity map, 291
security
AES, 149, 299

computer, cyber, data, 289
Pre-shared key (PSK), 149, 299
SSID, 149
TKIP, 149, 299
WEP, 149, 299
WPA, 149, 299
security standard
IEC 62443, 81
ISA-62443
62443, 306
semaphore, 413
sensor filter
finite impulse response (FIR), 349
infinite impulse response (IIR), 349
sensor network, 187
sequence diagram, 404
serial port
Close() method, 207
FireWire, 160
GetPortNames() method, 205
Hart, 161
Open() method, 206
Read() method, 207
RS-232C, 132
RS-422, 134
RS-485, 134
SATA, 152
USB, 154
Write() method, 208
serial port adapter, 197
service oriented architecture
SOA, 66, 531
service set identifier
SSID, 299
short message service, 172
sigma-delta, 332
signal
filtering, 345
SIL
Safety Integrity Level, 39
simple mail transfer protocol
SMTP, 123
simplex, 89
Simultaneous Sample and Hold, 342
SMS, 172
SOA, 282
SOAP, 170, 280
socket
Close() method, 215
connection, 214
GetHostName() method, 215
GetStream() method, 215
Ip Address, 215
Read() method, 216
TcpClient() method, 215
TcpListener() method, 215
Write() method, 216
software
process, 414, 427
state machine, 415
task, 428
thread, 414, 427
software as a service
SaaS, 58, 192
source code inspection, 530
SP100, 147
spam emails, 301
SQL, 363, 365
SSID, 149
stack buffer, 530
state machine, 465
static rounting, 119
Storage
CD-ROM, 473
FLASH, 472
NVRAM, 473
PCMCIA, 473
ROM, 472
USB stick, 473
submask, 113
subsea system, 37
successive-approximation, 332
supervised learning, 504
supervisory control and data acquisition
SCADA, 3
switch, 101
managed, 101
unmanaged, 101
symmetric algorithm
cryptic, 294
synchronous communication, 195
SysML, 9, 526
system design, 7, 523
v model, 7
system engineering, 6
system testing, 497
systems engineering, 523

tag, 26, 28, 43, 76, 271, 356
analog, 356
digital, 356
information, 356
target platform, 474
task state
offline, 408
ready, 408
running, 408
waiting, 408
TCP/IP, 57, 109, 111
classes, 112
private network addresses, 114
programming, 117
TCP, 111, 117
UDP, 111, 118
TCP/IP address, 111
TCP/IP protocol, 214

TDMA, 147
test case, 495
test driven development, 530
test levels, 496
test plan, 493
testing tools, 499
Token Ring, 108
    token, 108
training
    reinforcement learning, 504
    supervised learning, 504
    unsupervised learning, 504
transport layer security
    TLS, 307
trojan, 302
TSN
    time sensitive network, 122

UART, 131, 139
uart, 196
UDMA, 128
UDP, 111
UEFI
    unified extensible firmware interface, 293
        BIOS, 462
UML, 9, 372, 526
    class diagram, 526
    deployment diagram, 526
    interaction diagram, 526
    use case diagram, 526
unicast, 93
Unicode, 326
Unified Modeling Language, 372
uninterruptible power supply
    UPS, 517
unit testing, 496
unsupervised learning, 504
ups
    active, 517
    offline, 517
    online, 517
    passive, 517
USB, 154
    Wireless, 159
use case, 203
User Interface (UI), 24

v model, 7
virtual LAN, 124
virtual machine, 192
virtual machine (VM), 190
virtual private network
    VPN, 307
virus, 301
VPN, 124

watchdog, 435, 460
web socket secure
    WSS, 307
wi-fi protection access
    WPA, 299
Wireless
    6LoWPAN, 147
    bluetooth, 146
    HART, 146, 164
    SP100, 147
    USB, 159
    WLAN, 147
    ZigBee, 145
wireless equivalent privacy
    WEP, 299
wireshark
    TCP/IP packet analyzer, 217
worm, 302

XML, 169
    Extensible Markup Language, 280

YA-710, 376
YA-711, 376
YA-711 specification, 383

Zigbee, 145
zigbee, 151