Ainshams University
Faculty of Engineering
Computers & Systems Dept.

Wireless Accessibility to Local Measurements and Control in Industrial Processes

Project by:

1. Akram Mohamed Badr El Deen Abdul Majeed
2. Amro Ahmed Mahmoud Ghanem
3. Marco Louka Mitry Guirguis
4. Mostafa Mohamed Mohamed Elhoushi
5. Motaz Abdo Gaber Mohamed Ahmed
6. Mina Saad Mikhail Gerges
7. Mina Saad Mikhail Kirollos

Supervised by:

Prof. Dr. Mohamed Hassan Al-Shafey
Eng. Ahmed Mohamed Abdel-Fattah Ahmed

Contents

To all our teachers in the Computers and Systems Engineering Department…

Preface

Industrial process control is a fascinating and challenging area of electronics technology and nothing has revolutionized this area like the microcontroller. The microcontroller has added a level of intelligence to the evaluation of data and a level of sophistication in the response to process disturbances. Microcontrollers are embedded as the "brains" in both manufacturing equipment and consumer electronic devices.

Process control involves applying technology to an operation that alters raw materials into a desired product. Virtually everything that you use or consume has undergone some type of automatic process control in its production. In a manufacturing environment, automatic process control also provides higher productivity and better product consistency while reducing production costs.

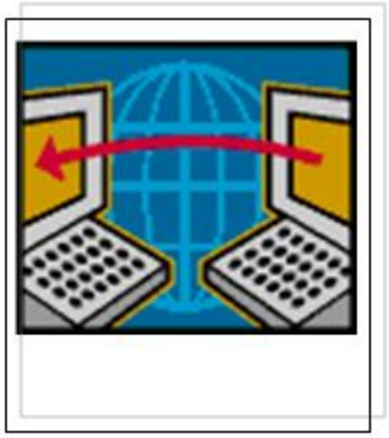
Over the years, the need for control systems which enable engineers to control and monitor a sophisticated industry with many processes from a single point gave rise to SCADA systems. SCADA is an acronym that stands for Supervisory Control and Data Acquisition. SCADA refers to a system that collects data from various sensors at a factory, plant or in other remote locations and then sends this data to a central computer which then manages and controls the data.

SCADA is a term that is used broadly to portray control and management solutions in a wide range of industries. Some of the industries where SCADA is used are Water Management Systems, Electric Power, Traffic Signals, Mass Transit Systems, Environmental Control Systems, and Manufacturing Systems.

The communication protocol needed to transfer data between a control room and the processes has evolved over the years from simple serial protocol to Ethernet protocol. Now, with the emerging of wireless communication, there is no need for complex wiring connections throughout an industry. In fact, a control room may be "portable" with the engineer moving around an industry with his laptop or PDA. This is the goal of the project.

Special thanks to Prof. Dr. Mohamed Hassan Al-Shafey and Eng. Abdel-Fattah for their continuous support and understanding, and thanks to all who helped us in this project.

Project Overview



Objective

The output of the project should be an encased module accompanied with its software to facilitate the control and monitoring of an industry using a PC, laptop or PDA via a wired and/or wireless network. It is in effect, an implementation of a SCADA (System Control And Data Acquisition) system using a wireless network.

Abstract

In an industrial process plant it is common that all local controllers be connected to a central control room where the control engineer has the equipment that enables him to monitor and control the whole process. Common tasks inside the control room are monitoring variables, changing set points, tuning local controllers, and production scheduling. Local controllers usually are microcontrollers with analog interface to the local sensors and actuators, and digital interface to some type of local network to connect to other controllers and to the central control room.

As the control engineer walks around the process plant he needs to have some of the capabilities of process monitoring and control he usually has in the control room. The objective of this project is to develop and implement a wireless local network that connects all local controllers to the control engineer laptop or PDA (Personal Digital Assistant) to allow him to perform tasks that require him to be in the control room. The wireless network is not intended to replace any existing local network, whether a field bus or an Ethernet, but it will work in parallel to any existing network providing the control engineer with a portable control room.

Minimum Implementation Tasks Required

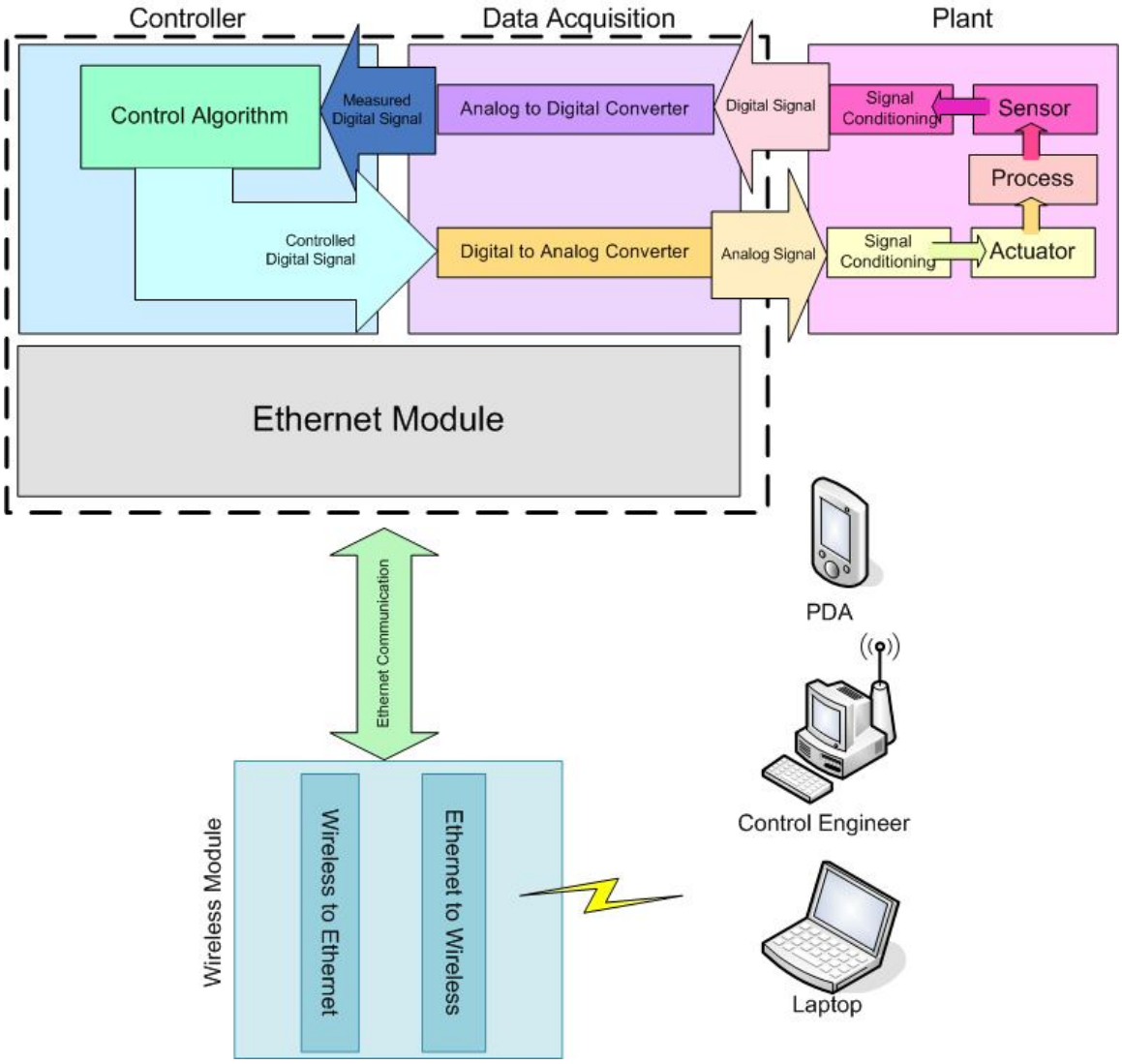


Figure 1 Overall design of the project.

- ***Data Acquisition****: Convert analog signals in their standard voltage and current ranges from the sensors into digital forms, and convert digital signals from the microcontroller to the actuators into analog signals within their standard ranges. It is required to implement such a data acquisition card with 8 input and output channels.*
- ***Control****: Implement a few local control loops (PID and On/Off control) using suitable microcontrollers. The processes to be controlled may be RC electrical circuits which are analogous to hydraulic, thermal and mechanical processes. The controller should be able to implement up to 8 control loops.*
- ***Hardware***: *Add wireless networking capabilities to each local microcontroller via one of its standard interfaces. This networking capability should allow the microcontroller to receive the digital controller parameters (e.g. set point, PID values, On/Off conditions) from the laptop, PDA or PC as well as transmitting output values to the control room.*

- ***Software**: Develop the software that integrates the local controllers and the engineer's portable station to allow the engineer to control and monitor the processes. The software should have different GUI capabilities to display the output values (graphs, graphics simulation of the real process, LCD screen, etc.).*

Extra Implementation Tasks

- ***Data Acquisition**: Allow the control engineer to choose any voltage or current ranges for each input or output channel. That is, the ranges of the I/O channels to be programmable.*
- ***Interface with PLC**: Software to be capable of interfacing with any PLC model given its driver.*
- ***Control**: Implement, along with PID loops and On/Off, RST controller, fuzzy controller and PLC ladder diagram. That is, the control engineer draws on the software a PLC ladder diagram to be implemented by the microcontroller at the plant.*
- ***VPN**: Form a secure VPN (**Virtual Private Network**) so that the control engineer may monitor the process from any part in the world.*
- ***Camera**: The software to support receiving pictures or videos of the process via the wireless network.*
- ***Speaker**: Allow the engineer to provide orders remotely to the workers at the plant to be heard through a speaker at the process.*

How Documentation is Organized

The documentation is divided into 6 parts:

- *Part 1 describes the **Local Controller** module; the part of the project which implements the data acquisition and control algorithms.*
- *Part 2 describes the **Ethernet Controller** module; the part of the project which provides the Ethernet interface to make the controller a node in a network able to be communicated with.*
- *Part 3 describes **Wireless Accessibility**; how the different nodes (whether they are user stations or local controllers) may be accessible remotely without wires.*
- *Part 4 describes the software on the laptop or PC to allow a control engineer to design, control and monitor a process.*
- *Part 5 describes how the software is modified to be implemented on a PDA or PocketPC.*
- *Part 6 describes some case studies of real-life applications or experiments performed in our project.*

Introduction

Remote Control and Monitoring

An industrial facility typically comprises a relatively small control room, surrounded by a relatively large physical plant. The control room is equipped with panels that depict the state of the plant as captured by sensors and input devices that control the actuators, affecting the state of the plant. The actuators and sensors are often relatively inexpensive when compared with the cost of the cable that needs to be used to connect them. The difference becomes even greater when considering the high installation and maintenance costs, the high failure rate of connectors and the difficulty of troubleshooting them.

The information being communicated in industrial environments is typically state information and as such in normal operation it takes the form of repeated streams of small packets. At the same time, these packets are associated with critical tasks having strict timing requirements in harsh environments. The latter may include extremely high or low temperatures, high humidity levels, intense vibrations, explosive atmospheres, corrosive chemicals and excessive electromagnetic noise caused by large motors and conductors. Thus, in general, the required data throughput of the network is relatively low, but its reliability needs to be very high.

In recent years, the desire for connectivity and physical mobility has caused an exponential growth in wireless communication systems. Wireless telephony has entered our daily lives and wireless local area networks increasingly serve as a means to access business and private data. In industrial environments, apart from lower installation and maintenance costs, wireless systems can offer ease of equipment upgrading and practical deployment of mobile robotic systems and micro-electromechanical systems (MEMS).

The rest of this introduction is organized as follows. The next two sections provide an overview of the evolution in wired control and monitoring and present solutions in cable replacement along with examples of successful deployments. The introduction continues with the hybrid wired/wireless fieldbus and mesh networking approaches. Finally, the results of study on how wireless technology is expected to impact industrial control and monitoring are presented in the form of a technology roadmap.

Wired Control and Monitoring

One may identify three main paradigms in wired communication for industrial control and monitoring:

***a)** parallel wiring,*
***b)** fieldbusses and*
***c)** industrial Ethernet.*

According to the first paradigm, each of the field devices is connected with parallel wires to the I/O module of a control or monitoring system. Such a point-to-point wiring approach became obsolete following the introduction of fieldbus technology, which allowed the use of only one two-wire line to provide power, control and configuration functions to devices. Fieldbusses allow many devices to be connected on the same wire and provide the necessary addressing mechanism to support communication with them. The open standardization approach adopted by main fieldbus technologies such as Profibus has facilitated interoperability among systems from multiple vendors and has been proven in many factory, process and building automation applications worldwide.

*Several fieldbus manufacturers have more recently recognized the advantages of Ethernet, which is the established standard bus system in the office world, for industrial applications. The advantages are related to the physical layer, particularly in terms of bandwidth, which can be higher than 100Mb/s rather than up to 12Mb/s for fieldbusses. The higher bandwidth can be utilized by larger packets e.g., for computer vision systems. In the past, the main disadvantage of Ethernet was the **Carrier Sense Multiple Access / Collision Detection** (**CSMA/CD**) contention protocol which does not guarantee time-critical communication. However, by splitting up the network in multiple collision domains using switches (or bridges), the prices of which have dropped dramatically as a result of the Internet revolution, every port on a switch is in its own collision domain and as such no more collisions between devices attached to the switch take place. Ethernet-based solutions offer also improved information sharing between manufacturing and backoffice systems such as asset management and inventory control applications. The data can also be made easily available via a Web server for remote control and monitoring purposes.*

Cable Replacement

Cable replacement is used here to denote the elimination of wires as the physical layer to carry data without requiring any physical changes to the machinery/instrument, the control panel or the underlying software involved. Industrial devices using traditional serial interfaces such as RS232, RS422 and RS485 are good candidates for cable replacement. This is because serial interfaces are typically connected to standard PCs and the connecting software is application dependent or device specific. Apart from serial point-to-point connections, cable replacement has found applications into master/slave multi-point connections, wireless parameterization and diagnosis particularly to do with moving/rotating subsystems, e.g., robotic arms. There are limitations to what cables can be replaced, though, due to the error characteristics of wireless links. When deterministic communication with latency (the time it takes for a packet of data to get from one designated point to another) under 10ms is essential, wireless transmission should be avoided.

*Wireless transmission can take place in various frequency bands and the transmission power is often restricted by law. The 2.4GHz **Industrial Scientific and Medical (ISM**) band is the most widely used. The 900MHz band, which is characterized by lower throughput but better*

range and wall penetration, is only available in a few countries and used by proprietary protocols. The 5.8GHz band holds a lot of potential in terms of higher throughput, better noise immunity and smaller antennas, but products are yet to be proved in the market.

Infrared (IrDA) can also be considered for cable replacement but has the shortcoming of requiring line of sight which limits its applicability.

Standard (Market Name)	802.11 (Wi-Fi)	802.15.1 (Bluetooth)	802.15.4 (Zigbee)
Application focus	Web, e-mail, video	Cable replacement	Control and monitoring
Bandwidth (Kbps)	11000	1000 – 3000	20 – 250
Transmission range (m)	100+	20 (Class 2) 100+ (Class 1)	20-70, 100+ (ext. amplifier)
Nodes supported	32	7	2^{64}
Battery life (days)	0.5 – 5	1 – 7	100 – 1000+
Power consumption (transmitting)	300 mA	45 mA (Class 2) <150 mA (Class 1)	30 mA
Suitability for low duty-cycle applications	Poor (Slow connection time)	Poor (Slow connection time)	Good
Spread spectrum technology	DSSS	FHSS	DSSS
Memory footprint (KB)	70+	50+	40
Success metrics	Speed flexibility	Cost, convenience	Power, cost

Table 1 Summary of different wireless standards.

Table 1 compares three main wireless transmission technologies. These technologies are complementary rather than competing to each other, as they address different needs and have different strengths. Bluetooth requires a low-cost transceiver chip in each device to be connected. Each device has a unique 48-bit address and the transceiver transmits and receives in the ISM band. Connections can be point-to-point or multipoint with a range of 20-100m. Data can be exchanged at a rate of 1-3Mbps and a ***Frequency Hopping Spread Spectrum*** *(****FHSS****) scheme allows devices to communicate even in areas with a great deal of electromagnetic interference. However, this makes it extremely difficult to create extended networks without large synchronization cost. Built-in encryption and simple verification is also provided by Bluetooth.*

ZigBee moves data only a quarter as fast as Bluetooth but can handle orders of magnitude more devices at once and has been optimized for low power consumption. This low power consumption is achieved by the ***Direct Sequence Spread Spectrum*** *(****DSSS****) which allows devices to sleep without the requirement for close synchronization. Another spread spectrum technique under development,* ***Ultra-Wide Band*** *(****UWB****), broadcasts simultaneously on a very large frequency range at low power. The idea is that the signal is spread so thinly that interference will be negligible in any given frequency. UWB is expected to be able to deliver high throughput, particularly in areas with physical obstacles.*

Some examples of industrial applications where cable replacement has been successfully deployed are given below:

- ***Phoenix Contact*** *(Germany): use Bluetooth to replace parallel wiring. A wireless multiplexer can replace wires for up to 32 digital inputs, 32 digital outputs, 2 analogue*

inputs and 2 analogue outputs. The product mirrors the inputs to the outputs and vice versa. The latency from input to output is 10 ms or less.

- ***Expert Monitoring*** *(UK): have developed WiSNet which allows sensors or instruments to be installed wirelessly. A typical WiSNet system consists of an Ethernet / USB network controller and sensor transmitter modules that enable sensors to be positioned anywhere inside or outside manufacturing plants. The transmitters use Bluetooth to send sensor data to a wireless network controller unit connected to a PC.*
- ***EnVision*** *(USA): have developed a wireless sensor that enables bioprocessors to monitor fermentation and cell culture processes directly in reactors. The sensor can be configured by either a Bluetooth wireless browser-based user interface or configured for Foundation Fieldbus communications.*
- ***Bromma Conquip*** *(Sweden): use Bluetooth to replace the cables connecting the control systems of harbor conveyor cranes with configuration and maintenance tools.*
- ***Schneider Toshiba Inverter Europe*** (***STIE***) *(France): use Bluetooth to replace the cables that connect a configuration tool (running on a PC or PDA) to their family of inverter products.*

Hybrid Wired/Wireless Fieldbus Networks

The need to retrofit any support for wireless to existing installations instead of creating new systems from scratch, increases the requirement for hybrid wired/wireless fieldbus networks. The standard ***Profibus Decentralised Periphery*** *(****DP****) network is based on a standard RS485 interface but often using unusual baud rates (93.75 and 187Kbps). The protocol has timing requirements; however, these requirements are adjustable because they were originally introduced to support modems. In order to be able to replace a fieldbus such as Profibus DP with a wireless link, it is very important to keep messages unbroken to support the timing requirements of the fieldbus protocol.*

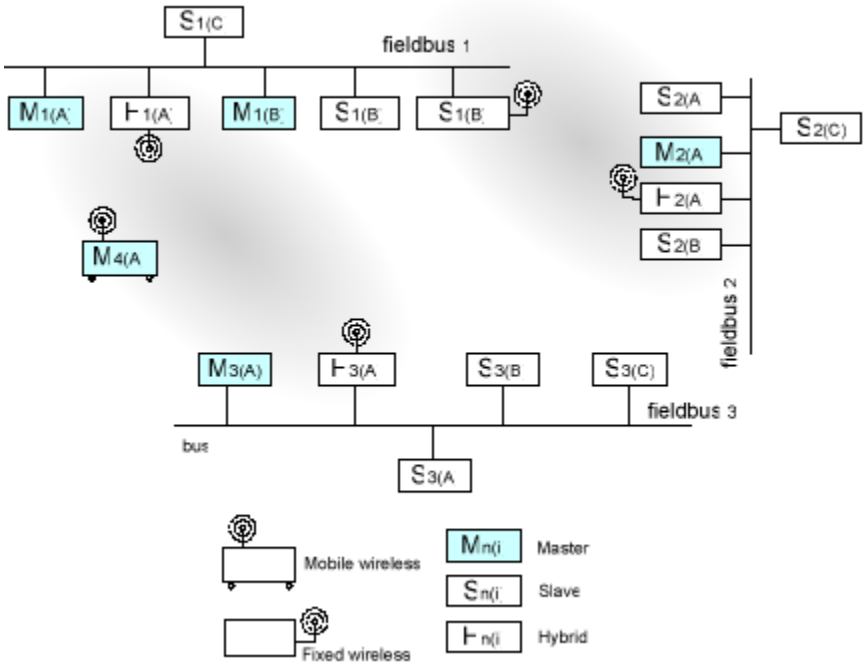


Figure 1 Example of a hybrid wired/wireless profibus topology.

Figure 1 depicts three fieldbusses that incorporate fixed wireless devices. At the low baud rate of 9.6Kbps the full number of 7 wireless nodes in a Bluetooth network can be supported, however at 93.75Kbps only 2-3 nodes can be supported. The maximum bandwidth in a point-to-point configuration is 187Kbps.

Modbus Remote Terminal Unit (RTU) fieldbus also uses RS485 as a transport media and has been successfully replaced by a Bluetooth wireless link both in single-point and multi-point configurations. Examples include the Phoenix Contact and STIE products mentioned above which use Modbus RTU on top of wireless links.

Mesh Networks

The traditional wireless systems have mostly used base station style radio links, with point-to-point or point-to-multipoint transmission. These wireless approaches have liabilities in industrial applications such as rigid structure, meticulous planning requirements and dropped signals. Rather than relying on a central communication coordinator and its associated reliability and efficiency issues, it is possible to use a collection of wireless devices maintaining connectivity to create a path for data packets to travel. This approach is known as a mesh network (Figure 2) and in many ways it resembles an idealized version of a top-level Internet backbone in which physical location is less important than capacity and network topology. In mesh networks each node has a low-power transmitter and communicates directly only with neighbouring nodes and these latter nodes relay data to more distant nodes. Should a link become congested or a node fail, the mesh automatically redirects data packets via an alternative path. This characteristic makes mesh networks virtually immune to localized interference such as that caused by motors turning on or arc welders.

Mesh networks allow applications to grow based on demand (the addition of new nodes is relatively simple), limiting fixed costs and providing great flexibility and capacity. They also minimize the need for elaborate site surveys or physical modifications to plants. Moreover, because of the short range of each transmission, the approach provides better utilization of available bandwidth than systems using high-power transmitters. Unlike other approaches, in mesh networks the major design requirement is the lowest possible node cost.

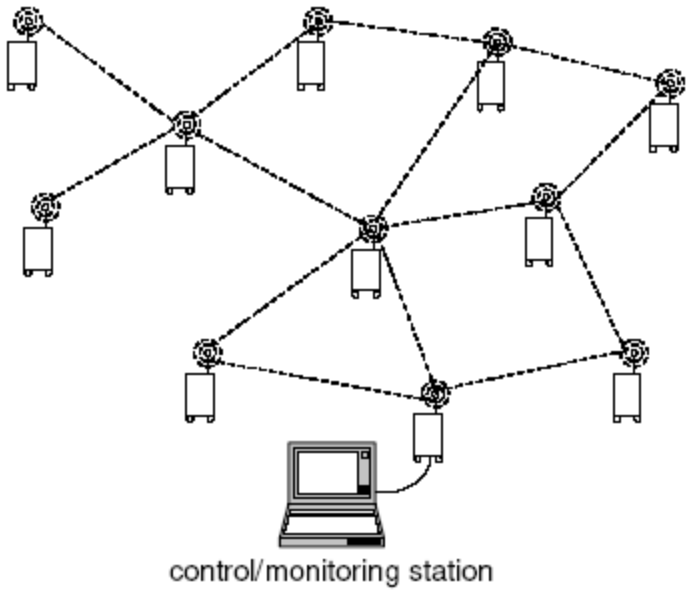


Figure 2 Wireless mesh network topology.

Many important questions remain though with respect to power management, routing strategies and algorithms in mesh networks. Even if good answers were to be given, mesh networks would not displace all existing industrial networks since deterministic operating modes (low latency and jitter) cannot be guaranteed.

Technology Roadmap

Embedded systems are now ubiquitous. They can be found in a diverse range of appliances, from mobile phones to smoke alarms, from refrigerators to trucks. Enabling these systems to communicate opens up new areas of applications: smart buildings, industrial automation, healthcare, power distribution and host of others. Some of the applications will result in a more efficient, accurate or cost effective solution than previous ones. Others will be new, previously unimagined or impossible. We are in the middle of a major technological revolution that will affect many aspects of our lives.

Moving to this exciting new technique in system development necessitates a common language for all systems. Without this, we risk repeatedly 're-inventing the wheel' at a high cost in money and effort, and we compromise the inter-operability of sensors between applications.

*The **RUNES** (**Reconfigurable Ubiquitous Networked Embedded Systems**) project has a vision to enable the creation of large-scale, widely distributed, heterogeneous networked embedded systems that interoperate and adapt to their environments. The inherent complexity of such systems must be simplified for programmers if the full potential for networked embedded systems is to be realized. The widespread use of network embedded systems requires a standardized architecture which allows self-organization to suit a changeable environment.*

RUNES aims to provide an adaptive middleware platform, a common language that will simplify the application creation process. This will allow for a dramatic cut in the cost of new application development and a much faster time to market, transforming applications which are already technically possible into forms that are easy and straightforward for designers to use;

and enabling applications which were previously unattainable. The project will also examine the potential uses and implications of the technology, develop demonstrator systems and design training courses to aid in dissemination of RUNES technology.

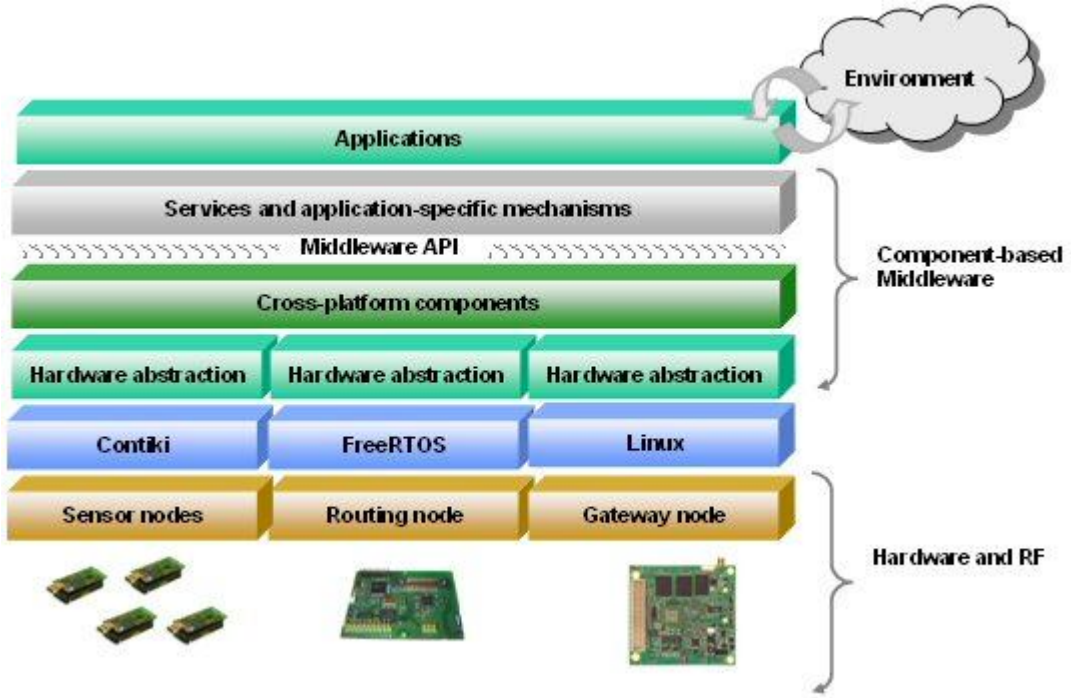


Figure 3 An overview of the RUNES technology which enables the creation of large scale, widely distributed, heterogeneous networked embedded systems that interoperate and adapt to their environments.

A technology roadmap for industrial control and automation was developed in the context of the RUNES project. The goal of technology roadmapping is to plot the future development of a technical field and help set more competitive and realistic targets. The objective of RUNES is to derive architectures and provide middleware and specialized simulation and verification tools that enable the creation of large-scale, widely distributed, heterogeneous networked embedded systems which interoperate and adapt to their environments. The roadmap is composed of the knowledge and views of over 25 organizations (large and small companies and research institutes) collected across Europe between Oct 2004 and Apr 2005. The focus of the roadmap is approximately 10 years although timescales for technological progress are notoriously difficult to predict. Below we summarize the technical, organizational and social issues identified.

Action plans for strategic positioning and resource allocation are also defined. Apart from the textual description of the roadmaps, we have summarized some of the findings in graphical form in Figure 4. The span of different technologies does not necessarily coincide with the first specification releases but mainly with the time adoption started. Regarding the decline of particular technologies, some assumptions, based on the information currently available, are made.

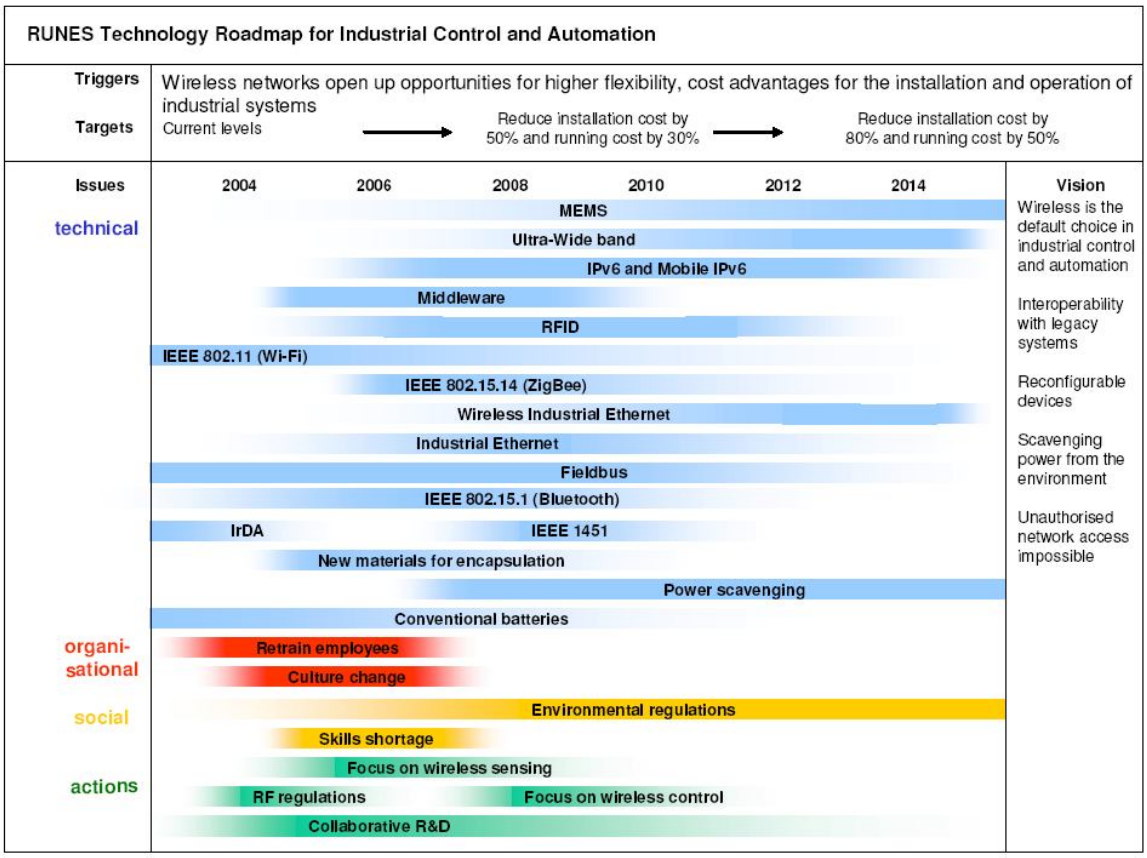


Figure 4 The RUNES technology roadmap for industrial control and automation in graphical form.

Technical Issues

Security

Major concerns about the integrity of signal transmission and reception have been expressed by industrial end-users who are worried about leaving their control and business systems vulnerable to hacking or denial of service attacks. Several solutions are based on proprietary protocols designed specifically with security concerns in mind, although this can be a barrier for future upgrades with standard equipment. Spread spectrum schemes provide inherent security against jamming or interference. Systems can employ 128-bit encryption with dynamically generated keys in order to prevent eavesdropping and unauthorized access. However, this is computationally too expensive for deterministic communication. Continuous monitoring of network activity and attempted access should also be supported. Spread spectrum technology provides inherent security against jamming or interference.

Robustness

The consequences of unreliable control and monitoring are not trivial. Industrial applications entail the risk of substantial losses through equipment damage, personnel injuries, loss of raw materials and environmental pollution. From the perspective of the integrator or end user, communications expertise and comprehensive technical support are key considerations that distinguish ordinary wireless products from the robust wireless communication solutions

required by industrial users. Many of today's wireless solutions offer mean time between failures which are unacceptable to deliver sustained performance in harsh environments. Any early failures could slow adoption of wireless. Encapsulation of sensitive electronic components is particularly important to achieve improved product reliability and extended lifecycles in harsh environments. Positioning of antennas is also critical in order to ensure reliable operation, as it can affect the bandwidth and data transfer rates.

Fail-safe/fail-soft operation

Designers of wireless industrial systems need to provide redundancy and allow degradation. Systems must be designed to go into a safe mode if and when a failure happens (fail-safe). Systems must also be capable of operating at a reduced level of efficiency after the failure of a component or power source (failsoft). Placing devices on different networks provides substantial fail-soft operation in that the failure of a single network only affects the devices that it supported. Devices connected to other, unrelated and perhaps geographically distributed, networks would be unaffected by the events that caused the failure of the first network.

Interference immunity

Multi-path fading is caused when multiple copies of a source signal arrive at a receiver through different
reflected paths and affects wireless communications indoors. The phase variance in the signal copies can result in interference that reduces signal strength, effective range, and data transfer rates. A wireless node has to handle multi-path signals, but interference can be caused by signals from other nearby systems particularly for unlicensed frequency bands. Extremely critical wireless equipment can operate inside a Faraday cage.

Power availability

As the speed of embedded processors increases and more peripherals are integrated into a single chip, the applications that run on these devices become more computationally intensive. However, technological advances of the batteries which power the embedded systems lags significantly behind, and as a result, power consumption is one of the most important issues for mobile wireless embedded systems. The major sources of energy waste include packet collisions, overhearing unwanted packets, control packet overheads and idle listening. For monitoring applications unnecessary high sampling rates also waste energy. One answer to the power problem is to scavenge power from the industrial environment. For instance, this could include development of photoelectric cells that draw energy from lighting. Power scavenging technology is something that industrial sector are thinking about already as supplying power to remote areas is costly.

Interoperability

At present, there are a number of different protocols for fieldbusses and industrial Ethernet, and end-users are concerned about the long-term cost implications of installing closed wireless systems. Until standards have been established and the market has become one for volume suppliers, fears of being left with an obsolete system may hold back widespread adoption. An important component towards interoperability is middleware, which defines appropriate abstractions and mechanisms for dealing with the heterogeneity of devices. Because ideally devices must operate unattended, the middleware has to provide new levels of support for automatic configuration and error handling. Development of middleware is a key activity

within the RUNES project. The IEEE Smart Transducer Interface for Mixed-Mode Communication Protocol (1451.4) was also revived last year in recognition of a need for a standardized approach to device interfaces.

Interfaces

Most industrial processes now modelled and controlled in IT systems, but they require the user to be physically at a terminal. Given the increasing need for workers to access systems remotely more research on machine to human interfaces needs to be undertaken. Human to human interfaces via devices that allow voice, image and video communication is another potentially important field given that they provide natural and real-time exchange of information, e.g. when someone on a factory floor needs to report a fault to an expert. Very few projects have addressed multimodal interaction such as gesture based programming of robotic arms.

Organizational Issues

Culture

The industrial automation sector, in general, is very conservative. Companies do not want to take chances with large investments in new installations and require demonstration of practicality (preferably by someone else). People involved in risk management are not receptive to new technologies. However, cheaper, faster, safer and more reliable options are always there for successful innovators. Often, employees who misunderstand new technology and lack confidence in its ability to improve over previous practice use many new systems in a suboptimal way. Collaboration between control people and radio people is not close enough. Major cultural differences exist between Europe and the less cellular-oriented US.

Work force

Industrial companies are expected to face a strategic human resource issue. The shift from a wired to a wireless plant, particularly one that can operate autonomously, will require adjustments in the work force. Engineers should understand radio technologies and be able to handle false alarms; hence revised technical training will be required. A shortage in embedded software development skills has also been identified.

Social Issues

Workforce

In general new levels of automation devalue unskilled labour through its replacement with less expensive systems. This is expected to increase job security concerns for people who only posses skills in declining technologies.

Environmental

Industrial control and automation have an enormous direct and indirect impact on the environment. New stricter regulations have been introduced and this trend is expected to continue. The deployment of large numbers of wireless sensors can be used for instance in warning systems to reduce the risk of environmental pollution.

Actions

The use of wireless systems for industrial applications is in its infancy. The adoption period is expected to be longer than other sectors (building automation and control, medical care, disaster response and automatic meter reading) reviewed in the RUNES technology roadmaps as end-users migrate incrementally from wire to wireless. Companies dealing with automotive, food processing, petrochemical and asset tracking applications were identified as the early adopters.

A clear difference in the adoption time scales between wireless in control and monitoring was revealed. While technologies are maturing, wireless will not be used for critical control applications. Monitoring in hazardous and inaccessible areas will be given priority in the short/medium term and in moving towards this some lessons can be learnt from successful automatic meter reading deployments. Many wireless systems on the market today do not meet local/national regulations, because they transmit too much power or operate in frequencies that are not approved for unlicensed use. Therefore, it is important to determine whether or not the radio subsystems can be programmed to meet these regulations. Since 2003 the ATEX directive has become mandatory for all electrical and mechanical equipment used in potentially explosive atmospheres and any new networked embedded components will need to comply with it.

The required R&D will involve expertise and engineering skills in communications, sensors and industrial computer systems which are very difficult to find under one corporate roof. Partnerships will therefore be essential. Sources for funding and mechanisms for facilitating collaborative R&D must be identified. Industrial companies collaborating with academia should persuade the latter to reconnect software and hardware education in their curricula.

Concluding Remarks

The traditional approach to the design of industrial control and monitoring systems – designing an architecture that integrates actuators and sensors within a single physical platform under centralized control – is changing. The emerging view, as motivated by new large-scale applications in the factory floor is one in which actuators, sensors, computing, and human interfaces may be distributed across multiple physical platforms. Adopting this new view of industrial systems design requires researchers to address a range of issues, architectures and applications related to wireless connectivity. In industrial control and monitoring there are certainly many future alternatives, however, the process of technology roadmapping helps narrow down the field of possible solutions to those more likely to be pursued. Most of the issues and technical requirements are not orthogonal and as such trade-offs depend on applications.

1

Local Controller

Chapter 1: Industrial Control

Chapter 2: Control Algorithm

Chapter 3: Microcontroller and Data Acquisition

Chapter 4: Controller Software

*The term **Local Controller** in our project refers to the module which performs both data acquisition and control algorithms. It is considered as the core of the project since it performs the main task of sampling a process and controlling its output values.*

In our project, the local controller was implemented as C-code software running on a Motorola HCS12 microcontroller. This microcontroller has accompanied the required hardware to perform analog and digital reading and writing, as well as timer interrupts to perform the actions at a fixed sampling rate.

We would like to note here that the local controller and Ethernet controller were actually implemented on the same microcontroller rather than using two separate hardware modules. This saved time and complexity in communicating between two separate hardware modules.

Chapter 1:
Industrial Control

1.1 Industrial Control

In recent years the performance requirements for process plants have become increasingly difficult to satisfy. Stronger competition, tougher environmental and safety regulations, and rapidly changing economic conditions have been key factors in tightening product quality specifications. A further complication is that modern plants have become more difficult to operate because of the trend toward complex and highly integrated processes. For such plants, it is difficult to prevent disturbances from propagating from one unit to other interconnected units.

In view of the increased emphasis placed on safe, efficient plant operation, it is only natural that the subject of **process control** has become increasingly important in recent years. Without computer-based process control systems it would be impossible to operate modern plants safely and profitably while satisfying product quality and environmental requirements.

The primary objective of process control is to maintain a process at the desired operating conditions, safely and efficiently, while satisfying environmental and product quality requirements. The subject of process control is concerned with how to achieve these goals. In large-scale, integrated processing plants such as oil refineries or ethylene plants, thousands of process variables such as compositions, temperatures, and pressures are measured and must be controlled. Fortunately, large numbers of process variables (mainly flow rates) can usually be manipulated for this purpose. Feedback control systems compare measurements with their desired values and then adjust the manipulated variables accordingly.

The specific objectives of control are:

- Increased product throughput
- Increased yield of higher valued products
- Decreased energy consumption
- Decreased pollution
- Decreased off-spec product
- Increased Safety
- Extended life of equipment
- Improved Operability
- Decreased production labor

Industrial control system (ICS) is a general term that encompasses several types of control systems, including **Supervisory Control And Data Acquisition** (**SCADA**) systems,

Distributed Control Systems (**DCS**), and other smaller control system configurations such as skid-mounted **Programmable Logic Controllers (PLC)** often found in the industrial sectors and critical infrastructures. **ICSs** are typically used in industries such as electrical, water, oil and gas. Based on information received from remote stations, automated or operator-driven supervisory commands can be pushed to remote station control devices, which are often referred to as field devices. Field devices control local operations such as opening and closing valves and breakers, collecting data from sensor systems, and monitoring the local environment for alarm conditions.

1.1.1 PLC

On many industrial sites electronic relays and simple on-off controllers are used to sequence regulator movements; e.g. valves opening and closing. Such controllers are known as **logic controllers** or **sequence controllers**.

These systems are gradually being replaced by sequence controllers based on as programmable logic controllers (PLCs). They have a modular design so they can be expanded to cover more areas of the process operation as it becomes automated. They can also incorporate advanced types of controllers.

A Programmable Logic Controller, PLC, or Programmable Controller is a digital computer used for automation of industrial processes, such as control of machinery on factory assembly lines. Unlike general-purpose computers, the PLC is designed for multiple inputs and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs to control machine operation are typically stored in battery-backed or non-volatile memory. A PLC is an example of a **real-time system** since output results must be produced in response to input conditions within a bounded time, otherwise unintended operation will result.

Figure 1.1 Control panel with PLC.

The main difference from other computers is that PLC are armored for severe condition (dust, moisture, heat, cold, etc) and have the facility for extensive input/output (I/O) arrangements. These connect the PLC to sensors and actuators. PLCs read limit switches, analog process variables (such as temperature and pressure), and the positions of complex positioning systems. Some even use machine vision. On the actuator side, PLCs operate electric motors, pneumatic or hydraulic cylinders, magnetic relays or solenoids, or analog outputs. The input/output arrangements may be built into a simple PLC, or the PLC may have external I/O modules attached to a computer network that plugs into the PLC.

PLCs were invented as replacements for automated systems that would use hundreds or thousands of relays, cam timers, and drum sequencers. Often, a single PLC can be programmed to replace thousands of relays. Programmable controllers were initially adopted by the automotive manufacturing industry, where software revision replaced the re-wiring of hard-wired control panels when production models changed.

Many of the earliest PLCs expressed all decision making logic in simple ladder logic which appeared similar to electrical schematic diagrams. The electricians were quite able to trace out circuit problems with schematic diagrams using ladder logic. This program notation was chosen to reduce training demands for the existing technicians. Other early PLCs used a form of instruction list programming, based on a stack-based logic solver.

The functionality of the PLC has evolved over the years to include sequential relay control, motion control, process control, distributed control systems and networking. The data handling, storage, processing power and communication capabilities of some modern PLCs are approximately equivalent to desktop computers. PLC-like programming combined with remote I/O hardware, allow a general-purpose desktop computer to overlap some PLCs in certain applications.

Under the IEC 61131-3 standard, PLCs can be programmed using standards-based programming languages. A graphical programming notation called **Sequential Function Charts** is available on certain programmable controllers.

PLCs are well-adapted to a certain range of automation tasks. These are typically industrial processes in manufacturing where the cost of developing and maintaining the automation system is high relative to the total cost of the automation, and where changes to the system would be expected during its operational life. PLCs contain input and output devices compatible with industrial pilot devices and controls; little electrical design is required, and the design problem centers on expressing the desired sequence of operations in ladder logic (or function chart) notation. PLC applications are typically highly customized systems so the cost of a packaged PLC is low compared to the cost of a specific custom-built controller design. On the other hand, in the case of mass-produced goods, customized control systems are economic due to the lower cost of the components, which can be optimally chosen instead of a "generic" solution, and where the non-recurring engineering charges are spread over thousands of sales.

1.1.2 DCS

A distributed control system (DCS) refers to a control system usually of a manufacturing system, process or any kind of dynamic system, in which the controller elements are not central in location (like the brain) but are distributed throughout the system with each component sub-system controlled by one or more controllers. The entire system may be networked for communication and monitoring.

Figure 1.2 The part of DCS – DeltaV – Processor with I/O modules.

DCSs are normally used to control large or complex processes. They are modular systems that allow operators to adjust the set-points of many individual controllers from a central control. A central control unit monitors the operation of each of the other controllers and makes data available to other high-level systems, such as fault diagnosis, process optimization and production-scheduling systems.

A DCS typically uses computers (usually custom designed processors) as controllers and use both proprietary interconnections and protocols for communication. Input & output modules form component parts of the DCS. The processor receives information from input modules and sends information to output modules. The input modules receive information from input instruments in the process (aka field) and output modules transmit instructions to the output instruments in the field. Computer buses or electrical buses connect the processor and modules through multiplexers/demultiplexers. Buses also connect the distributed controllers with the central controller and finally to the Human-Machine Interface (HMI) or control consoles.

Distributed control systems (DCSs) are used in industrial, electrical, computer and instrumentation and control engineering applications to monitor and control distributed

equipment with or without remote human intervention; the nomenclature for the former 'manual control' and the latter 'automated control'. DCS is a very broad term that describes solutions across a large variety of industries, including:

- Electrical power grids and electrical generation plants
- Environmental control systems
- Traffic signals
- Water management systems
- Refining and chemical plants
- Pharmaceutical manufacturing
- Sensor Networks

The broad architecture of a solution involves either a direct connection to physical equipment such as switches, pumps and valves or connection via a secondary system such as a SCADA system.

A DCS solution does not require operator intervention for its normal operation but with the line between SCADA and DCS merging, systems claiming to offer DCS may actually permit operator interaction via a SCADA system.

Distributed Control Systems (DCSs) are dedicated systems used to control manufacturing processes that are continuous or batch-oriented, such as oil refining, petrochemicals, central station power generation, pharmaceuticals, food & beverage manufacturing, cement production, steelmaking, and papermaking. DCSs are connected to sensors and actuators and use setpoint control to control the flow of material through the plant. The most common example is a setpoint control loop consisting of a pressure sensor, controller, and control valve. Pressure or flow measurements are transmitted to the controller, usually through the aid of a signal conditioning Input/Output (I/O) device. When the measured variable reaches a certain point, the controller instructs a valve or actuation device to open or close until the fluidic flow process reaches the desired setpoint. Large oil refineries have many thousands of I/O points and employ very large DCSs. Processes are not limited to fluidic flow through pipes, however, and can also include things like paper machines and their associated variable speed drives and motor control centers, cement kilns, mining operations, ore processing facilities, and many others.

A typical DCS consists of functionally and/or geographically distributed digital controllers capable of executing from 1 to 256 or more regulatory control loops in one control box. The input/output devices (I/O) can be integral with the controller or located remotely via a field network. Today's controllers have extensive computational capabilities and, in addition to proportional, integral, and derivative (PID) control, can generally perform logic and sequential control.

DCSs may employ one or several workstations and can be configured at the workstation or by an off-line personal computer. Local communication is handled by a control network with

transmission over twisted pair, coaxial, or fiber optic cable. A server and/or applications processor may be included in the system for extra computational, data collection, and reporting capability.

1.1.3 SCADA

The term SCADA is used differently in North America than in the rest of the world:

- In North America; SCADA refers to a large-scale, distributed measurement and control system.
- In the rest of the world; SCADA is any system that performs Supervisory Control And Data Acquisition, independent of its size or geographical distribution.

SCADA systems are typically used to perform data collection and control at the supervisory level. Some SCADA systems only monitor without doing control, these systems are still referred to as SCADA systems.

SCADA systems can be used to control a wide range of industrial processes and are often used to provide an operator interface for PLC-based control systems. They are software packages designed to run on a computer, with facilities for storing data for analysis. Advanced SCADA systems also incorporate advanced control algorithms that can help operators to automatically optimize process operations. Some advanced SCADA systems also include fault diagnosis and production scheduling systems.

The supervisory control system is a system that is placed on top of a real-time control system to control a process that is external to the SCADA system (i.e. a computer, by itself, is not a SCADA system even though it controls its own power consumption and cooling). This implies that the system is not critical to control the process in real-time, as there is a separate or integrated real-time automated control system that can respond quickly enough to compensate for process changes within the time-constants of the process. The process can be industrial, infrastructure or facility based as described below:

Industrial processes include: manufacturing/production/power generation/fabrication/refining - continuous, batch, repetitive or discrete.

Infrastructure processes may be public or private and include: water treatment and distribution, wastewater collection and wastewater treatment, oil & gas pipelines, electrical power transmission and distribution and large communication systems.

Facility processes in private or public facilities including: buildings, airports, ships or space stations in order to monitor and control: HVAC, access control, energy consumption management

The SCADA systems for these applications all perform *Supervisory Control And Data Acquisition*, even though the use of the systems are very different.

A SCADA system includes input/output signal hardware, controllers, HMI, networks, communication, database and software. It mainly comes in the branch of Instrumentation Engineering.

The term SCADA usually refers to a central system that monitors and controls a complete site or a system spread out over a long distance (kilometres/miles). The bulk of the site control is actually performed automatically by a Remote Terminal Unit (RTU) or by a Programmable Logic Controller (PLC). Host control functions are almost always restricted to basic site over-ride or *supervisory* level capability. For example, a PLC may control the flow of cooling water through part of an industrial process, but the SCADA system may allow an operator to change the control set point for the flow, and will allow any alarm conditions such as loss of flow or high temperature to be recorded and displayed. The feedback control loop is closed through the RTU or PLC; the SCADA system monitors the overall performance of that loop.

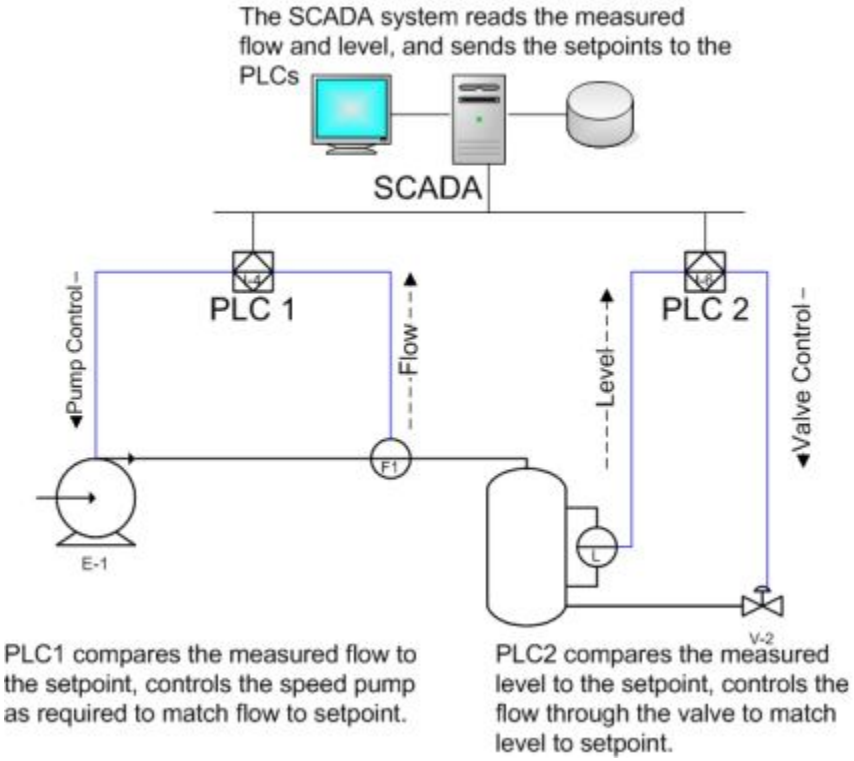


Figure 1.3 An example of a SCADA system.

Data acquisition begins at the RTU or PLC level and includes meter readings and equipment statuses that are communicated to SCADA as required. Data is then compiled and formatted in such a way that a control room operator using the HMI can make appropriate supervisory decisions that may be required to adjust or over-ride normal RTU (PLC) controls. Data may also be collected in to a Historian, often built on a commodity Database Management System, to allow trending and other analytical work.

SCADA systems typically implement a distributed database, commonly referred to as a *tag database*, which contains data elements called *tags* or *points*. A point represents a single input or output value monitored or controlled by the system. Points can be either "hard" or "soft". A hard point is representative of an actual input or output connected to the system, while a soft point represents the result of logic and math operations applied to other hard and soft points. Most implementations conceptually remove this distinction by making every property a "soft" point (expression) that can equal a single "hard" point in the simplest case. Point values are normally stored as value-timestamp combinations; the value and the timestamp when the value was recorded or calculated. A series of value-timestamp combinations is the history of that point. It's also common to store additional metadata with tags such as: path to field device and PLC register, design time comments, and even alarming information.

It is possible to purchase a SCADA system, or Distributed Control System (DCS) from a single supplier. It is more common to assemble a SCADA system from hardware and software components like Telvent, Allen-Bradley, ABB, Siemens, DirectLogic or GE PLCs, HMI packages from Adroit, Wonderware, Iconics, Rockwell Automation, Inductive Automation, Citect, or GE. Communication typically happens over ethernet.

1.1.4 Microcontrollers

A **microcontroller** (or **MCU**) is a computer-on-a-chip. It is a type of microprocessor emphasizing self-sufficiency and cost-effectiveness, in contrast to a general-purpose microprocessor (the kind used in a PC). The only difference between a microcontroller and a microprocessor is that a microprocessor has three parts - ALU, Control Unit and registers (like memory), but the microcontroller has additional elements like ROM, RAM etc.

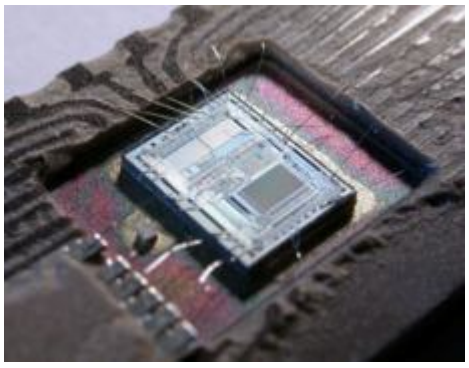


Figure 1.4 An example of a microcontroller: Intel 8742.

This is what is implemented in our project. Microcontrollers are used less frequently in industrial control since they are less robust. But they have the advantage of having fewer restrictions than PLC or SCADA: a skilled programmable may write any code (whether in C or assembly) to implement virtually anything he wants. In fact, a PLC may be implemented using a microcontroller. Therefore, special-cases or complex cases may be implemented using microcontrollers.

1.1.4 Comparisons

For high volume or very simple fixed automation tasks, different techniques are used. For example, a consumer dishwasher would be controlled by an electromechanical cam timer costing only a few dollars in production quantities.

A microcontroller-based design would be appropriate where hundreds or thousands of units will be produced and so the development cost (design of power supplies and input/output hardware) can be spread over many sales, and where the end-user would not need to alter the control. Automotive applications are an example; millions of units are built each year, and very few end-users alter the programming of these controllers. However, some specialty vehicles such as transit busses economically use PLCs instead of custom-designed controls, because the volumes are low and the development cost would be uneconomic.

Very complex process control, such as used in the chemical industry, may require algorithms and performance beyond the capability of even high-performance PLCs. Very high-speed or precision controls may also require customized solutions; for example, aircraft flight controls. In such cases, we need microcontrollers.

PLCs may include logic for single-variable feedback analog control loop, a "proportional, integral, derivative" or "PID controller." A PID loop could be used to control the temperature of a manufacturing process, for example. In fact, some PLCs have ready-made complex modules such as PID tuning and fuzzy control. Historically PLCs were usually configured with only a few analog control loops; where processes required hundreds or thousands of loops, a distributed control system (DCS) would instead be used. However, as PLCs have become more powerful, the boundary between DCS and PLC applications has become less clear-cut.

1.2 Where to take Control?

Most industrial control applications make a complex network of sensors and actuators connected to a single controller in a control room, while some make a group of local controllers to interface with a nearby group of sensors and actuators and give feedback to a central monitoring system. The two systems are illustrated in the following figure.

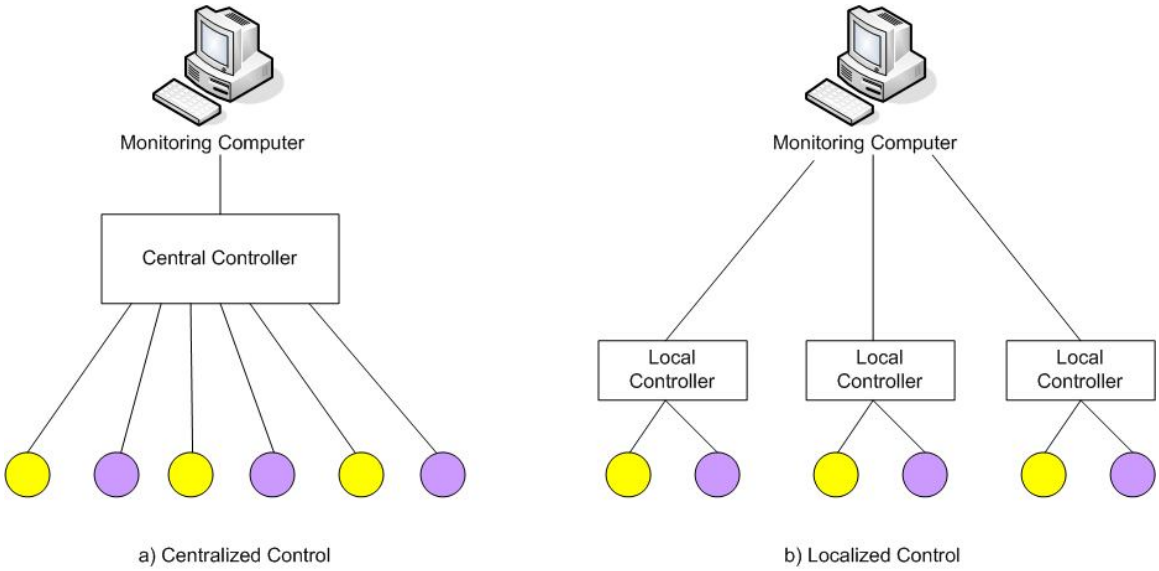


Figure 1.5 Comparing centralized control and localized control.

Centralized control is actually implemented in most industries. This has the advantage that the controller is safe from any process hazards. A certain communication protocol is needed to connect all of the sensors and actuators – each assigned with an address - to the central controller.

Localized control is what we have chosen. This has the advantage that control is not delayed or halted by any communication hazards that may occur between the sensors, actuators and the controller. Also, a controller may control more than one process. Moreover, there is more flexibility in adding new sensors, actuators, since they do not need to have certain interfaces or addresses as in the previous case. A certain communication protocol is needed to connect different controllers – each assigned with an address - to different monitoring stations.

After comparing the advantages and disadvantages of both types, we have chosen to implement localized control; i.e. we form a network of local controllers, each with its own address, to control nearby actuators and have a monitoring computer to receive the sampled values as well as giving control commands to the local controller.

Chapter 2:
Control Algorithms

2.1 Basic Control Concepts

In engineering and mathematics, **control theory** deals with the behavior of dynamical systems. The desired output of a system is called the **reference** or **set-point**, *r*. The value or process variable needed to be controlled is called the **controlled variable,** *c*. The difference between the set-point and the controlled variable is called the **error**, e. The signal which is affected directly into the process and outputted directly by the controller is called the **manipulated variable**, *m*. It is the process variable that can be adjusted in order to keep the controlled variables at or near its set point. The **measured value**, *m,* may be slightly different to the controlled variable due to noise or signal conditioning. When one or more output variables of a system need to follow a certain reference over time, a controller manipulates the inputs to a system to obtain the desired effect on the output of the system.

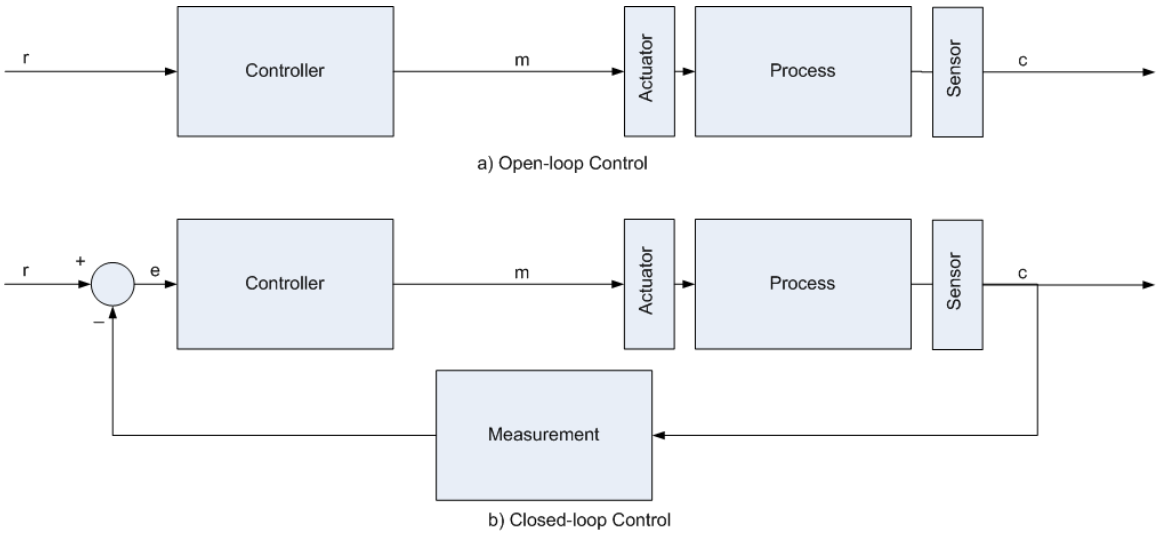


Figure 2.1 Comparing open-loop and closed-loop control.

Common parts found in a typical control system:

- **Process** or **Plant:** the physical system needed to be controlled.
- **Measurement Device:** To control a process accurately, the control loop needs to be able to measure a particular value (such as temperature) on a regular basis. This is usually done by:
    - a **sensor** which takes the measurement,
    - and then a **transmitter** that converts the sensor reading into a standard control signal

- **Transducer:** fgfg…..
- **Actuator** or **Regulator**: This controls the throughput of the process. The type of regulator depends on the need of the process. Examples include: a control valve or variable speed drive that adjusts the flow of a fluid and electronic circuit that supply power to a device (e.g. switch on a heater)
- **Controller:** compares the measured value (e.g. current temperature) with its set-point (the required temperature) and, where there is a difference, it adjusts the regulator for the required process, e.g. a drop in temperature could result in an increase in fuel to the burners in order to bring the temperature back up to the required set-point.

There are mainly two types of control: open-loop and closed-loop. Open-loop control, which is rarely used, simply applies a pre-defined manipulated variable into the process to achieve a certain output. This manipulated variable is a function of the required set point and is either calculated or found by trial-and-error. However, if any external disturbance (i.e. a sudden change in the environment) occurs or any internal disturbance (i.e. the process parameters change due to ageing) occurs or if any noise occurs, the controlled variable will change. Therefore, we need feedback to reduce any undesired change in the controlled variable. This is introduces us to **negative feedback**. Open-loop control can be used in systems sufficiently well-characterized as to predict what outputs will necessarily achieve the desired states. For example, the rotational velocity of an electric motor may be well enough characterized for the supplied voltage to make feedback unnecessary. Drawbacks of open-loop control is that it requires perfect knowledge of the system (i.e. one knows exactly what inputs to give in order to get the desired output), and it assumes there are no disturbances to the system.

Closed-loop control, which is what is implemented in our project, achieves the required feedback. Closed-loop control has another advantage over open-loop control in that we may modify the controller to achieve the **transient response** (i.e. how the controlled variable reaches its steady-state value) we want. The controlled variable is always compared with the required set-point to obtain the error. According to this error, the controller takes the correcting action to exert onto the actuator. How this correcting action is obtained is what differs different types of controllers from each other. The task of a control loop is to hold a particular process variable (e.g. mixer speed, or oven temperature) at the required set-point or value.

The **single-loop controller**, if it is correctly chosen and well tuned, is able to handle about 90% of control tasks found within the process industries. In our project, we have implemented the single-loop controller.

Linear systems, or non-linear systems which are linearized by considering a small portion of its range of operation, have their responses classified into two types of responses: **first-order response** and **second-** (or **higher-**) **order response**:

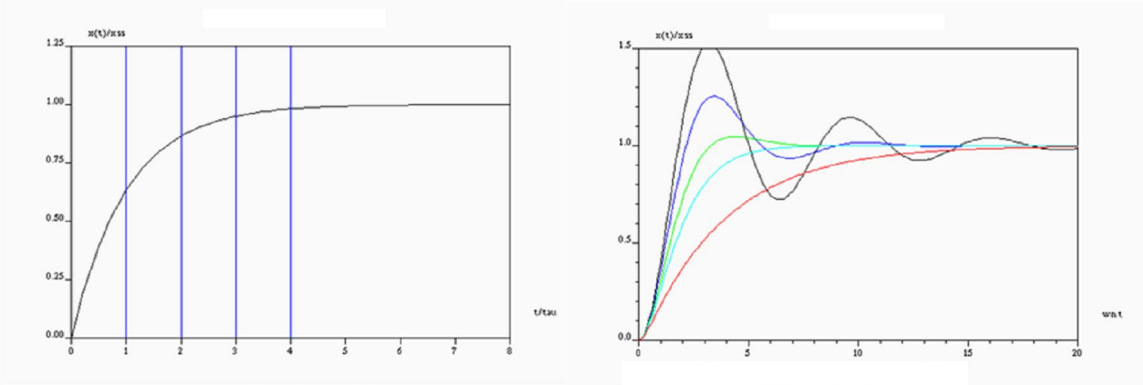


Figure 2.2 Comparing first-order response and second-order response.

- First order response is characterized by the **time-constant,** τ or T_0, which is the time required for the system to reach 63% of its final value. Usually, we would like to have the time-constant as low as possible to make the system response faster. Its transfer function in the s-domain is represented by:

$$H(s) = \frac{1}{1 + sT_0}$$

- Second order response has the following transfer function in the s- domain:

$$H(s) = \frac{{\omega_0}^2}{{\omega_0}^2 + 2\eta\omega_0 + s^2}$$

It is characterized by:

  - **Rise-time,** t_r : refers to the time required for a signal to change from a specified low value to a specified high value. Typically, these values are 10% and 90% of the step height. Usually, we want to minimize this value.
  - **Settling-time,** t_s : refers to the time required for a signal to remain within 2% of its steady-state value. Usually, we also want to minimize this value.
  - **Maximum Overshoot,** *M*: is the maximum positive difference between a signal and its steady-state value. Usually, we also want to minimize this value.

Using closed-loop control we may change the above parameters of a response of a system to virtually any required value.

2.2 Digital Control

For a very long time, feedback control was implemented in continuous-time using analog systems; namely pneumatic system or electronic (op-amp) circuits. However, the rise of

computers paved the way for **digital control**. Since our project is using microcontrollers, we are going to implement digital control.

The term "digital control" usually implies two facts. First, control is taken place in the discrete-time domain not the continuous-time domain. This means that the microcontroller should sample the controlled variable every time sample, T_s , and the control action is determined and taken every sample time. Second, the values of the controlled variable and the manipulated variable are quantized; therefore we have a limited number of values to read and write to.

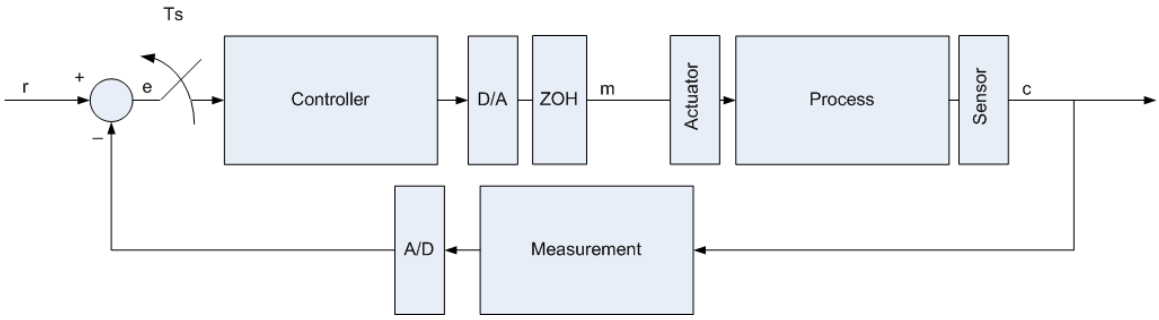
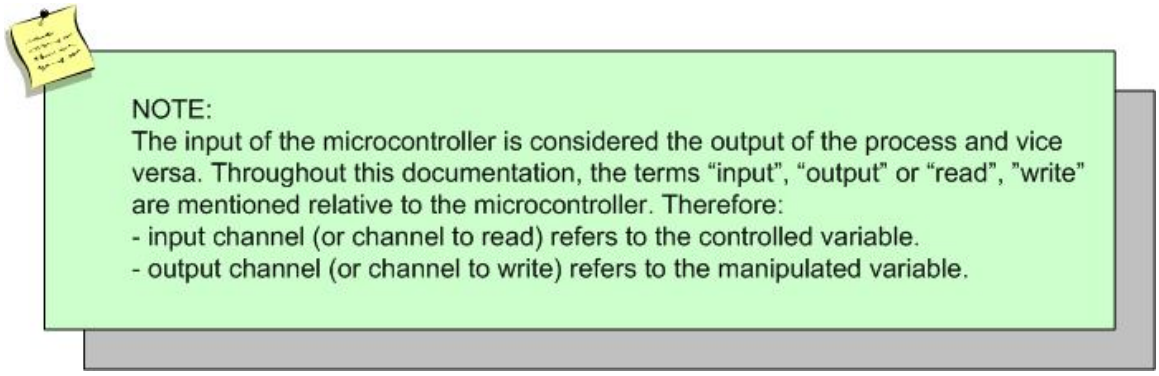


Figure 2.3 Block diagram of digital control closed-loop system.

Digital control systems have introduced 2 more components: **digital-to-analog converter** (**D/A** or **DAC**) and **analog-to-digital converter** (**A/D** or **ADC** or **ATD**). Digital data is represented as a group of bits to be stored in the microcontroller's memory and allow calculation and manipulation.



There are several advantages to digital control over analog control:

- The whole world is moving towards the digital control era.
- Digital components are less susceptible to ageing and environmental variations.
- They are less sensitive to noise.
- Changing a controller does not require an alteration in the hardware (flexibility).
- Reduce the design time.
- Improve the system reliability.

- Easier system integration.
- Low implementation costs.

Considering the quantization of data, a question has emerged: should the data be dealt with as integers or floating numbers and should they be scaled? We have agreed in the project that the user should enter all of his values (for example: set points) as percentage values, and they should be scaled by the monitoring software to a byte value from 0-255. The microcontroller should perform all of its calculations as bytes or integers and return the values as bytes to the monitoring software to scale it back to the percentage range.

There was no need to perform calculations in the microcontroller as floating numbers for 2 reasons:

- The high inefficiency in storage, power and execution time.
- Any calculated values would be casted back to integer numbers (since the analog output should also be discritized), therefore there was no advantage of having high precision values since their precision would be lost again.

Another implementation consideration was the sampling time: would it be large or small?

- A very small sample time means more control actions per unit time which therefore means more power used by the microcontroller.
- A very large sample time means that control actions will be taken too late to perform the desired feedback, causing the system to be unstable. Moreover, the original signal may not be reconstructed

The minimum required sampling time should satisfy Nyquist criterion:

$$f_s > 2f_{max}$$

Where f_{max} is the maximum frequency component in the system, usually equal to its bandwidth.

- For first order systems, we choose $\frac{T_0}{4} < T_s < T_0$, where T_0 is the time constant.
- For second order systems, we choose $0.25 \leq \omega_0 T_s \leq 1.5; \quad 0.7 \leq \eta \leq 1$

We have designed our project so that the sample time is chosen by the user.

2.3 On/Off Control

On/Off controllers are very famous industrial controllers, they can even be found everywhere in our daily life, for example On/Off controllers can be found in air conditioners and refrigerators, heaters and most temperature systems, also in water level regulating systems.

On/Off controllers are famous for the following reasons:

- On/Off controllers are very simple to design and implement.
- They perform very well when chosen correctly to operate on a suitable process (first order process).
- Can be implemented digitally or in analog circuits.
- Are not severely affected with noise, or disturbances.
- Reliable in operation for long times.

On/Off controllers are characterized by a **set point**, which we want the controlled variable to vary around, and the **neutral zone**, which determines the greatest offset from the set point. We need to make a compromise between a small neutral zone value, which will cause high switching rate for actuators which will therefore cause its depreciation, and a large neutral zone value which will cause a large maximum error.

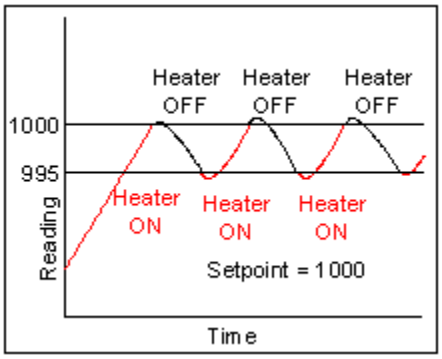


Figure 2.4 An example of On/Off control action.

The output of an On/Off controller is digital since it can be either high or low. On/Off control may be implemented digitally using an if-else statement:

```
if (Analog Input <= (SP + NZ/2)) Digital Output = HIGH

else if (Analog Input >= (SP - NZ/2)) Digital Output = LOW
```

2.4 PID Control and PID Tuning

A **proportional-integral-derivative controller (PID controller)** is a generic control loop feedback mechanism 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, based upon three parameters.

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 determines the reaction based on recent errors and the

derivative determines the reaction based on the rate by which the error has been changing. The weighted sum of these three actions is outputted to a control element such as the position of a control valve or power into a heating element.

By "tuning" the three constants in the PID controller algorithm the PID can provide individualized control specific to process requirements including error responsiveness, overshoot of set-point and system oscillation. Note that the general nature of PID control does not guarantee optimal control of the system.

Some applications may require only using one or two modes (by setting undesired control values to zero) to provide the appropriate system control. A PID controller will be called a PI, PD, P or I controller in the absence of respective control actions. PI controllers are particularly common, since derivative action is very sensitive to measurement noise, and the absence of an integral value prevents the system from ever reaching its target value due to control action.

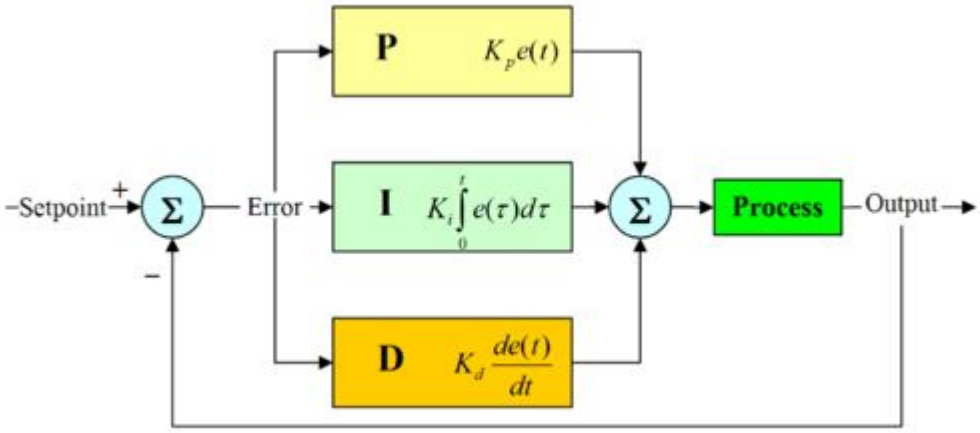


Figure 2.5 A block diagram of a PID controller.

The PID control scheme is named after its three correcting terms, whose sum constitutes the output. The three terms are

Proportional term

The proportional term responds to a change in the process variable proportional to the current measured error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain or proportional sensitivity. The gain is also frequently expressed as a percentage of the proportional band. The proportional term is written as:

$$m = K_p e = \frac{100}{PB} e$$

m: **Output Signal**

K_p: **Proportional Gain**

e: **Error** equal to (set point value - process variable)

PB: **Proportional Band**

A high gain results in a large response to a small error, a more sensitive system (Also called a narrow proportional band). Note that by setting the proportional gain too high, the system can become unstable. In contrast, a small gain results in a small response to a large error, and less sensitive system (Also called a wide proportional band), which is undesirable as the control action may be insufficient to respond to system disturbances.

Finally pure proportional control will never theoretically settle at its target value, but will rather approach the target with a steady state error that is a function of the proportional gain, this is known as a steady state error.

Integral term

The contribution from the integral term is proportional to the past and current values **and** duration of the error signal. The integral term algorithm calculates the accumulated proportional offset over time that should have been corrected previously (finding the offset's *integral*). While this will force the signal to approach the set point quicker than a proportional controller alone and eliminate steady state error, it may also contribute to system instability as the controller will always be responding to past values. This instability causes the process to overshoot the set point since the integral value will continue to be added to the output value, even after the process variable has reached the desired set point.

The responsiveness of the integral function can be calibrated to the specific process by adjusting the constant T_i, called the integral time.

The equation is written as:

$$m = \frac{1}{T_i} \int_0^t e(\tau)\, d\tau$$

m: **Output Signal**

T_i: **Integral Time**

e: **Error** equal to (setpoint value - process variable)

Although mathematically the integral starts at $t = 0$, it is possible to modify the integral to such that it does not "record" all historical values of the error signal. There are many possible schemes for performing such modification, such as windowing the signal or applying a decay term to the integral value itself.

Derivative term

The derivative term provides a braking action to the controller response as the process variable approaches the set point. To accomplish this the process error is predicted at a time in the future T_d, calculated by analyzing the slope of error vs. time (i.e. the rate of change of error, which is its first derivative with respect to time) and adding the anticipated proportional term to the current correction.

Derivative control is used to reduce the magnitude of the overshoot produced by the integral component, but the controller will be a bit slower to reach the set point initially. As differentiation of a signal amplifies the noise levels, this mode of control is highly sensitive to noise in the error term, and can cause a noisy controlled process to become unstable.

By adjusting the constant, T_d, the derivative time, the braking action sensitivity is controlled.

The derivative term is written as:

$$m = T_d \frac{de}{dt}$$

m: **Output Signal**

T_d: **Derivative Time**

e: **Error** equal to (set point value − process variable)

2.4.1 PID Algorithm Implementation

Parallel / "non-interacting" Form

The PID algorithm can be implemented in several ways. The easiest form to introduce is the parallel or "non-interacting" form, where the P, I and D elements are given the same error input in parallel. The output of the controller (i.e. the input to the process) is given by

$$\mathrm{Output(t)} = P_{\mathrm{contrib}} + I_{\mathrm{contrib}} + D_{\mathrm{contrib}}$$

where P_{contrib}, I_{contrib}, and D_{contrib} are the feedback contributions from the PID controller, defined below:

$$P_{\mathrm{contrib}} = K_p\, e(t)$$

$$I_{\mathrm{contrib}} = \frac{1}{T_i} \int_0^t e(\tau)\, d\tau$$

$$D_{\text{contrib}} = T_d \frac{de}{dt}$$

Where $e(\tau) = set\ point - measurement(\tau)$ is the error signal, τ is the time in the past contributing to the integral response and K_p, T_i, T_d are constants that are used to tune the PID control loop:

K_p: Proportional Gain - Larger K_p typically means faster response since the larger the error, the larger the feedback to compensate.

T_i: Integral Time - Smaller T_i implies steady state errors are eliminated quicker. The trade-off is larger overshoot: any negative error integrated during transient response must be integrated away by positive error before we reach steady state.

T_d: Derivative Time - Larger T_d decreases overshoot, but slows down transient response.

Normally the controller is implemented with the K_p gain applied to the I_{contrib}, and D_{contrib} terms as well in the following form, also called the standard form;

$$\text{Output(t)} = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau + T_d \frac{de}{dt} \right)$$

In the ideal parallel form, the standard parameters T_i and T_d are replaced with (K_i and K_d).

$$\text{Output(t)} = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt}$$

In this notation the gain parameters are related to the parameters of the standard form through $K_i = \frac{K_p}{T_i}$ and Kd = KpTd. This parallel form where the parameters are treated as simple gains is the most general and flexible form. However, it is also the form where the parameters have little physical interpretation.

Series / interacting form

Another representation of the PID controller is the series, or "interacting" form. This form essentially consists of a PD and PI controller in series, and it made early (analog) controllers easier to build. When the controllers later became digital, many kept using the interacting form.

Often, one deals with discrete time intervals instead of the continuity. Thus, the PID controller may also be dealt with recursively:

$$\text{Output}_n = \text{Output}_{n-1} + (K_p + K_i + K_d)\, e_n - (K_p + 2K_d)\, e_{n-1} + K_d\, e_{n-2}$$

2.4.2 Loop Tuning

If the PID controller parameters (the gains of the proportional, integral and derivative terms) are chosen incorrectly, the controlled process input can be unstable, i.e. its output diverges, with or without oscillations, and is limited only by saturations or breakage. **Tuning** a control loop is the adjustment of its control parameters (gain/proportional band, integral gain/reset, derivative gain/rate) to the optimum values for the desired control response. The optimum behavior on a process change or set point change varies depending on the application. Some processes must not allow an overshoot of the process variable from the set point. Other processes must minimize the energy expended in reaching a new set point. Generally stability of response is required and the process must not oscillate for any combination of process conditions and set points. Tuning of loops is made more complicated by the response time of the process; it may take minutes or hours for a set point change to produce a stable effect. Some processes have a degree of non-linearity and so parameters that work well at full-load conditions don't work when the process is starting up from no-load. This section describes some traditional manual methods for loop tuning.

There are several methods for tuning a PID loop. The most effective methods generally involve the development of some form of process model, then choosing P, I, and D based on the dynamic model parameters. Manual "tune by feel" methods have proven time and again to be inefficient, inaccurate, and often dangerous.

The choice of method will depend largely on whether or not the loop can be taken "offline" for tuning, and the response speed of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters.

Choosing a Tuning Method		
Method	**Advantages**	**Disadvantages**
Ziegler-Nichols	Proven Method. Online method.	Process upset, some trial-and-error, very aggressive tuning
Tune By Feel	No math required. Online method.	Erratic, not repeatable
Software Tools	Consistent tuning. Online or offline method. May include valve and sensor analysis. Allow simulation before downloading.	Some cost and training involved.
Cohen-Coon	Good process models.	Some math. Offline method. Only good for first-order processes.

Table 2.1 Advantages and disadvantages of different tuning methods.

If the system must remain online, one tuning method is to first set the *I* and *D* values to zero. Increase the *P* until the output of the loop oscillates, then the *P* should be left set to be approximately half of that value for a "quarter amplitude decay" type response. Then increase *I* until any offset is correct in sufficient time for the process. However too much *I* will cause instability. Finally, increase *D*, if required, until the loop is acceptably quick to reach its reference after a load disturbance. However too much *D* will cause excessive response and overshoot. A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly; however, some systems cannot accept overshoot, in which case a "critically damped" tune is required, which will require a *P* setting significantly less than half that of the *P* setting causing oscillation.

Effects of *increasing* parameters				
Parameter	**Rise Time**	**Overshoot**	**Settling Time**	**S.S. Error**
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small Change	Decrease	Decrease	None

Table 2.2 Effects of increasing PID parameters.

Ziegler-Nichols method

Another tuning method is formally known as the "Ziegler-Nichols method", introduced by John G. Ziegler and Nathaniel B. Nichols. As in the method above, the *I* and *D* gains are first set to zero. The "P" gain is increased until it reaches the "critical gain" K_c at which the output of the loop starts to oscillate. K_c and the oscillation period P_c are used to set the gains as shown:

Ziegler-Nichols method			
Control Type	K_p	K_i	K_d
P	$0.5 \cdot K_c$	-	-
PI	$0.45 \cdot K_c$	$1.2K_p / P_c$	-
PID	$0.6 \cdot K_c$	$2K_p / P_c$	$K_p P_c / 8$

Table 2.3 Ziegler-Nichols method.

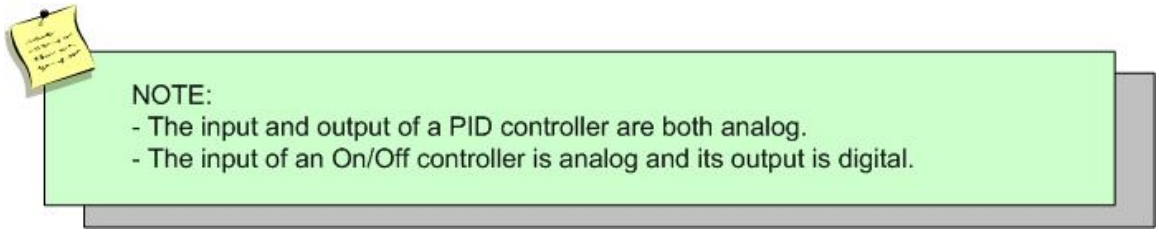
PID Tuning Software

Most modern industrial facilities no longer tune loops using the manual calculation methods shown above. Instead, PID tuning and loop optimization software are used to ensure consistent results. These software packages will gather the data, develop process models, and suggest

optimal tuning. Some software packages can even develop tuning by gathering data from reference changes.

Mathematical PID loop tuning induces an impulse in the system, and then uses the controlled system's frequency response to design the PID loop values. In loops with response times of several minutes, mathematical loop tuning is recommended, because trial and error can literally take days just to find a stable set of loop values. Optimal values are harder to find. Some digital loop controllers offer a self-tuning feature in which very small set point changes are sent to the process, allowing the controller itself to calculate optimal tuning values.

Other formulas are available to tune the loop according to different performance criteria.



Chapter 3:
Microcontroller and Data Acquisition

3.1 What is a Local Controller?

Any process needed to be monitored and controlled should have a **local controller** connected to it. The local controller does two main jobs:

- **Data Acquisition**
- **Control**

The **Data Acquisition** module samples the process data every specified period of time and keeps it ready for any monitoring requests to the process, and the **Control** module takes benefit of the data collected by the Data Acquisition module and calculates the appropriate control action according to the user previously defined configuration of the controller.

In this project both the Data Acquisition and the Control functions were done at ONE module: **Freescale DEMO9S12NE64** which is based on the MC9S12NE64 microcontroller unit (MCU).

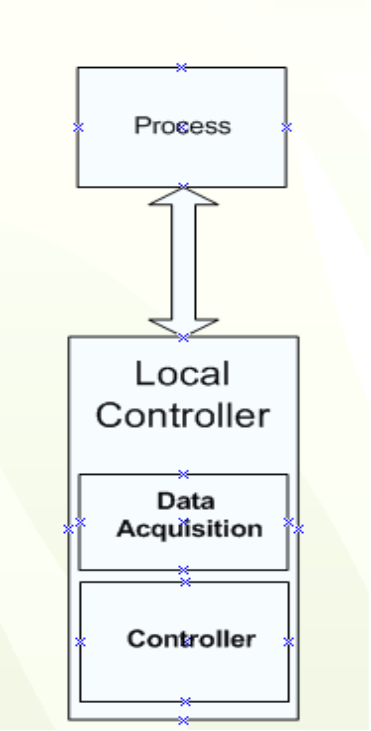


Figure 3.1 Local controller and a process.

3.2 Data Acquisition

DAQ is an abbreviation for **Data AcQuisition**. Though there have been a variety of different abbreviations used over the years (e.g. DAS, DAC, DataAcq, etc), the three letters D-A-Q truly have risen to the level of a universally accepted synonym for its much longer brother. Usage of the term DAQ has expanded to include much more than purely analog input. It is now used to describe an extremely wide variety of system tasks including analog input, analog output, digital input and output, counter/timer and motion monitoring.

Data acquisition is widely used in many areas of industry. Data acquisition is used to acquire data from sensors and other sources under computer control and bring the data together and store and manipulate it. In view of the wide variety of signals and parameters that can be sampled and stored, data acquisition involves many techniques and skills. There are many different components to a data acquisition system including sensors, communication links,

signal processors, computers, databases, data acquisition software, etc. All these items have to operate together to make a successful data acquisition system.

Data acquisition systems can take many forms from very simple manual systems to high complicated computer controlled ones. The simplest form may be a technician manually logging information such as the temperature of an oven. However this form of data acquisition has its limitations. Not only is it expensive because of the fact that someone has to be available to take the measurements, but being manual it can be subject to errors. Readings may not be taken at the prescribed times, and also there can be errors resulting from the manual fashion in which the readings are taken. As can be imagined the problems become worse if a large number of readings need to be taken, as timing may become more of an issue, along with the volume of work required.

To overcome this, the simple answer is to use computer control to perform the data acquisition. As a result a definition of what is normally taken to be data acquisition is gathering information in an automated fashion from analogue and digital measurement sources, i.e. sensors and devices under test. The sensors may range from thermocouples and voltage and current sensors to strain gauges and displacement gauges and much more.

Data Acquisition Measurements

Data acquisition systems may make any number of a huge variety of measurements. These measurements typically measure analogue. Before they can be transferred into any computer system they need to be in a digital format.

Although a huge number of data acquisition measurements can be made, they basically boil down to a very few basic elements:

- Voltage
- Digital signals
- Frequency or time interval

The sensors that are used in data acquisition measurements often return values of voltage in particular that can then be converted to the values of displacement, temperature, or whatever is being measured.

Data Acquisition stage is essential as it prepare the data collected from the process and put them in a form that could be used in the controller stage, noticing that the collected data could have one of two usages:

- Sent to the Software application for monitoring
- Processed and used in taking the right control action

In this project, the process used is an R-C Circuit that is equivalent to some real processes (such as tank systems or heater systems); which have equivalent transfer functions.

For simplicity the sensor's role that transforms the real property into an equivalent electrical signal was canceled and therefore we could say that the input/output to/from the process is pure analog signal.

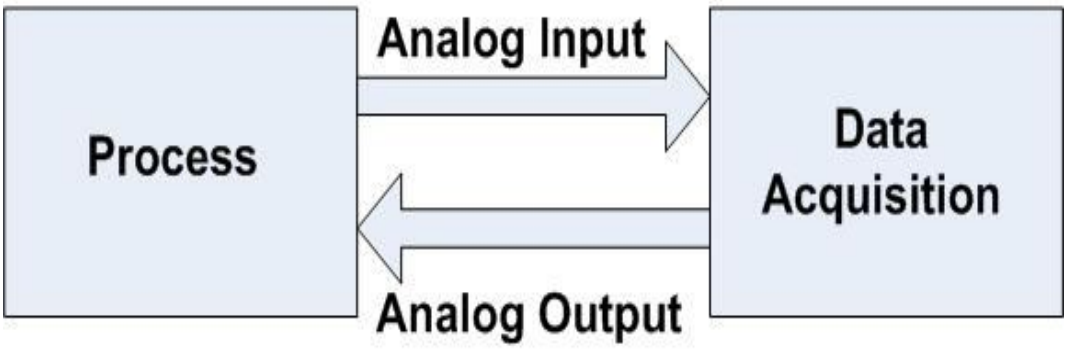
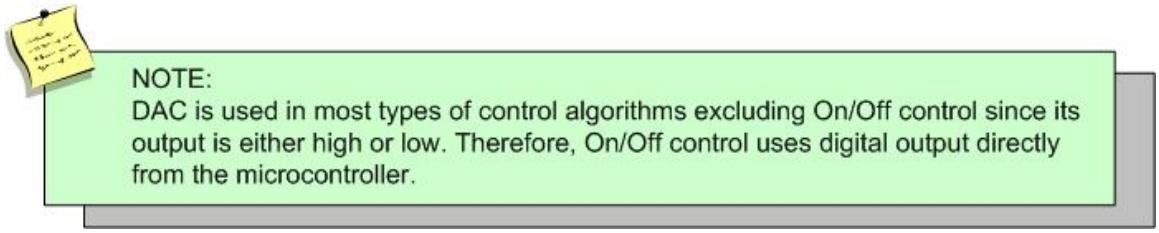


Figure 3.2 The interface between a process and a data acquisition card is mainly analog.

The **Analog Input** to the Data Acquisition is the data which needs to be digitized (sampled) first using the Analog-To-Digital Converter (ADC) and then enter the control stage which send these sampled digital data over the network for monitoring and/or processing to do the control action according to the commands specified.

The **Analog Output** from the Data Acquisition is the control action that first was in the form of digital signal and then passed over the Digital-To-Analog Converter (DAC) to convert it to analog form so as to be used to derive the process and regulate its behavior.



Data Acquisition Functions:

1. Collect data of variables of interest by sampling them every determined period of time and keep them ready for any monitoring requests.

2. Perform any digital or analog conversions needed

3. Communicates with the controllers so they can benefit from the collected data in their control tasks, and configures the controllers to achieve the desired control also interfaces the controllers digital outputs to the processes analog input.

4. Interfaces and communicates somehow with the software application, giving it digital data for monitoring purposes.

For the **Data Acquisition** to perform its tasks it needs the following **sub-modules**:

1. ***Analog-to-Digital*** Conversion sub-module to collect the analog data and convert them to digital form so as to be sent to the monitoring software and also to be used in digital controllers' algorithms applied on the microcontroller.

2. ***Digital-to-Analog*** Conversion sub-module to interface the controllers digital output to the processed analog input (control action).

3.3 HCS12NE64 Microcontroller

As mentioned before, that **"Freescale DEMO9S12NE64"** which is based on the **MC9S12NE64 microcontroller unit** (MCU) was used to do the Data Acquisition and the Control tasks. Let us investigate it in more details.

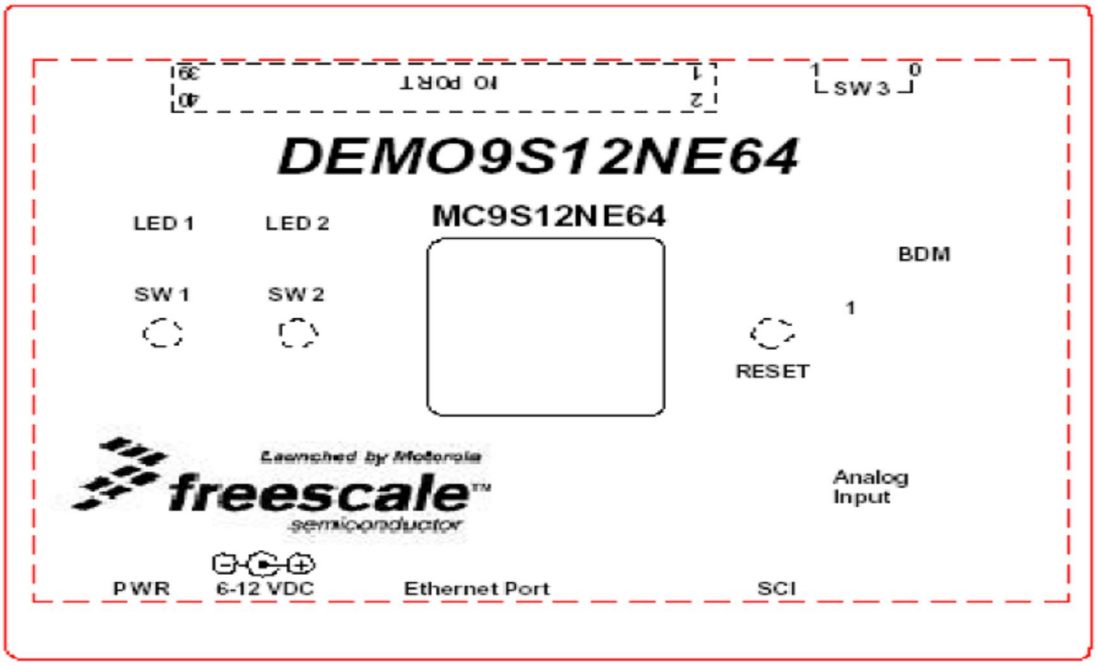


Figure 3.3 The enclosing case of DEMO9S12NE64.

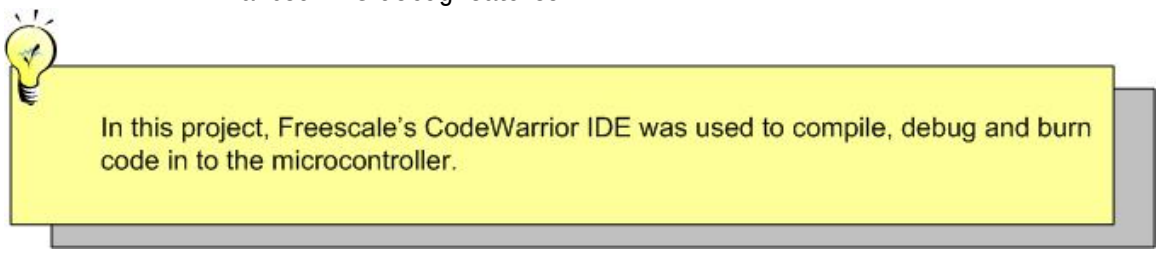
Characteristic	Specification
Maximum Clock Speed	25 MHz at 3.3V (12.5 MHz bus)
Temperature Operating Storage	 -10°C to +50°C -40°C to +85°C
MCU Extension I/O	HCMOS Compatible at 3.3V
Relative Humidity	0 to 90% (non-condensing)
Power Requirements	6 to 12VDC 0.75 Amp supplied externally. A barrel type power plug is required with an outside diameter of 5.5 mm and inside diameter of 2.1 mm.
Dimensions	3.0 x 4.5 x 1.25 inches

Table 3.1 Details of the Demo Board. DEMO9S12NE64.

The Microcontroller's Main Features

- ✓ 16-bit HCS12 core
  - HCS12 CPU
    - Upward compatible with M68HC11 instruction set
    - Interrupt stacking and programmer's model identical to M68HC11
  - Memory map and interface (MMC)
  - Background debug mode (BDM)
- ✓ Memory
  - 64K bytes of FLASH EEPROM
  - 8K bytes of RAM
- ✓ ***Analog-to-Digital converter (ATD)***
  - One 8-channel module with 10-bit resolution
  - External conversion trigger capability
- ✓ ***Timer module (TIM)***
  - 4-channel timer
  - Each channel configurable as either input capture or output compare
  - Simple PWM mode
  - 16-bit pulse accumulator
- ✓ ***Serial interfaces***
  - Two asynchronous serial communications interface (SCI)
  - ***One synchronous serial peripheral interface (SPI)***
  - One inter-IC bus (IIC)
- ✓ Operating frequency
  - 50 MHZ equivalent to 25 MHZ bus speed for single chip
  - 32 MHZ equivalent to 16 MHZ bus speed in expanded bus modes
- ✓ Internal 2.5-V regulator
  - Supports an input voltage range from 3.3 V (+/-)5%
  - Low-power mode capability
  - Includes low-voltage reset (LVR) circuitry
- ✓ Development support
  - Single-wire background debug™ mode (BDM)

- On-chip hardware breakpoints
- Enhanced DBG debug features



Analog-to-Digital converter (ATD)

The ATD that is part of the Data Acquisition is a built-in module in the MCU. The A/D input is programmable and up to 8 analog channels for the digitizing process could be handled.

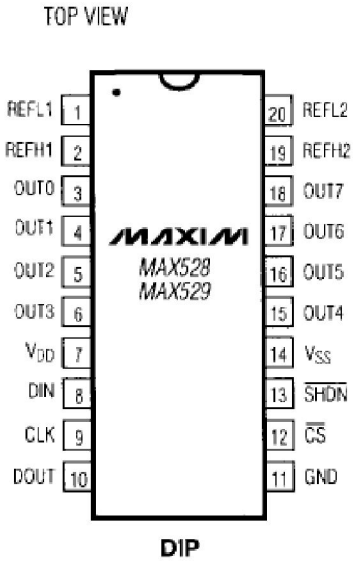
The MCU supports an internal 10 Bit Resolution but it was adjusted so as to be only 8 bits Resolution (1 byte), to ease the calculations, compromise data storage.

3.4 Digital-to-Analog converter (DAC)

This sub-module is needed in the Data Acquisition to convert from Digital to Analog, but it is not included in the MCU.

After a little search, a D/A was chosen that supports many great features: the MAX528 from MAXIM INC.

DAC Pin Configuration



Pin	Function
CS Active-Low Chip Select	When asserted low, this input pin initializes the 528 to start a new frame of serial data. When asserted high, the 16-bit data is latched and the internal shift register is turned off. The DAC registers also are updated with the new data.
DOUT Serial Data Out	This open drain pin serves as the serial output data from the DIN pin.
DIN Serial Data In	This pin serves as the input data line that receives the 16-bit serial data stream.
CLK Serial Data Clock	This pin is an input that drives the serial transmission lines.
SHDN Shutdown	Connect this input pin high for normal operation. Connect it low to conserve power.

Table 3.2 Pin functions of MAX528.

General Description

The MAX528 are monolithic devices combining an octal 8-bit, digital-to-analog converter (DAC), 8 output buffers, and serial-interface logic in space-saving shrink small outline package (SSOP).

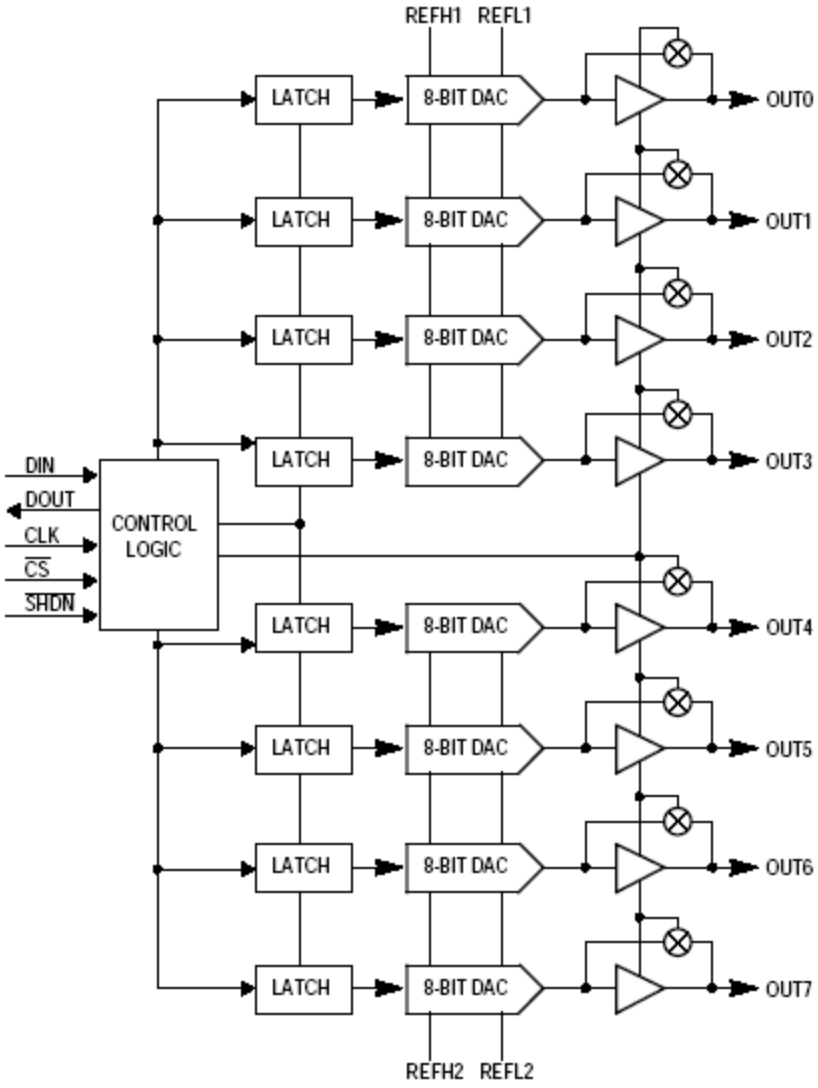


Figure 3.4 Block diagram for MAX528.

Three output modes are serially programmable for each pair of 8 analog outputs:

a) Unbuffered Mode: Connects the internal R-2R DAC network directly to the output pin, reducing power consumption and avoiding the buffer's DC errors.

b) Full-Buffered Mode: Inserts a buffer between the R-2R network and the output, providing +5mA/-2mA output drive.

c) Half-Buffered Mode: It is similar to the Full-Buffered Mode, but uses less power while still providing up to 15mA of output drive in a unipolar output configuration.

Serial data can be "daisy-chained" from one device to another. On power-up, all data bits are reset to 0, and analog outputs enter the unbuffered mode.

DAC Output Range

The MAX528 provides 8 voltage outputs (OUT0 – OUT7) from 2 reference inputs. Each reference voltage has 2 pins; REFH and REFL . The OUT0 – OUT3 output voltages are derived from REFH1 and REFL1, while OUT4 – OUT7 are derived from REFH2 and REFL2.

For each reference, REFH must be more positive than REFL . A DAC output voltage is the product of its programmed 8-bit code and its reference input voltage. For example, the output (analog) voltage of OUT5 is:

OUT5 = (REFH2 – REFL2) (nn/256 + REFL2)

Where nn = 8-bit code for OUT5, with a range of 0 – 255(00 to FF hex)

DAC Programming

The MAX528 are programmed by 16 data bits in two 8-bit bytes, the address pointer (A7 – A0) followed by the data byte (D7 – D0). These bits enter a shift register serially through DIN (pin8) A7 first, and D0 last. The data exits the DOUT (pin10) 16 clocks later in the same order.

Address pointer bits								*Data byte*							
A7	A6	A5	A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0

Data at DIN is shifted into the first register (while all 16 register bits shift forward one stage) on the rising CLK edge, while holding _CS low and _SHDN high. This must occur 16 times to load all the data bits into the shift registers. On the rising edge of _CS, data in the 16 shift registers is transferred as addressed and CLK disabled.

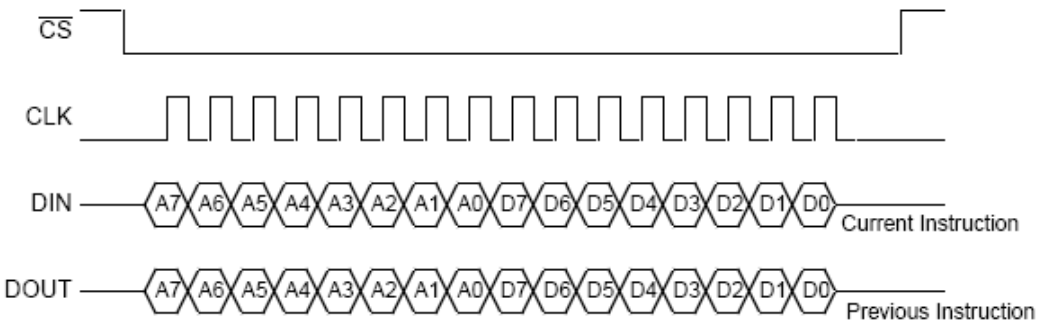


Figure 3.5 Timing diagram for MAX528.

Set Buffer Modes

Set buffer modes is implemented when all 8 address pointer bits (A7 – A0) are logic 0 and data bit D7 is 1. When this instruction is received, data bits D5 – D0 (ignoring D6) are transferred to the mode registers only; and the DAC registers are unchanged.

Enabling and disabling the 8 buffers is done in four pairs by data bits {D1, D2, D4, D5}. D1 controls buffers 6 and 7, D2 controls buffered 4 and 5, D4 controls buffers 2 and 3 and D5 controls buffers 0 and 1. A logic 1 enables a buffer pair (full-buffered or half buffered mode)l a logic 9 disables a buffer pair (unbuffered mode).

Full-buffered and half-buffered modes are set by two data bits, D0 and D3. D0 controls OUT4 through OUT7; D3 controls OUT0 through OUT3. A logic 1 enables full-buffered mode; a logic 0 enables half-buffered mode. These data bits apply only when buffer output pairs are enabled by a 1 in D1, D2, D4, or D5.

Mode	OUT 0,1	OUT 2,3	OUT 4,5	OUT 6,7
Unbuffered (D0 = D3 = X)	D5 = 0	D4 = 0	D2 = 0	D1 = 0
Half-buffered	D5 = 1	D4 = 1	D2 = 1	D1 = 1
	D3 = 0		D0 = 0	
Full-buffered	D5 = 1	D4 = 1	D2 = 1	D1 = 1
	D3 = 1		D0 = 1	

Table 3.3 Buffer mode programming.

3.5 SPI Module

The **Serial Peripheral Interface** Bus or **SPI** bus is a synchronous serial data link standard named by Motorola that operates in full duplex mode. Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave select (chip select) lines. Sometimes SPI is called a "four wire" serial bus, contrasting with three, two, and one wire serial busses. It has the advantage over SCI (Serial Communication Interface) or UART for its very bit rate since its clock may be as high as 70 MHz while the maximum data rate of UART is about 100 kbps only.

Figure 3.4 SPI bus: single-master single-slave.

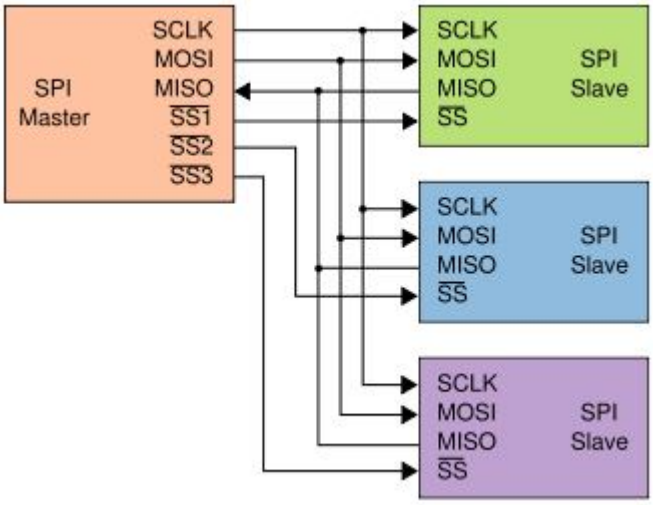


Figure 3.6 SPI bus: single-master multiple-slave.

Advantages	Disadvantages
Full duplex communication	Requires more pins on IC packages than I²C: -Half duplex "3-wire" mode uses one less pin per slave (possible in newer controllers with bidirectional mode; some slaves, like EEPROMs, tristate their outputs when receiving data from the master and don't care about the input when sending data back) -No in-band addressing protocol, so out-of-band chip select signals are required
Higher throughput than I²C	No hardware flow control
Complete protocol flexibility for the bits transferred	No slave acknowledgment (the master could be "talking" to nothing and not know it)
Extremely simple hardware interfacing	Multi-master busses are rare and awkward, and are usually limited to a single slave
Uses many fewer pins on IC packages, and wires in board layouts or connectors, than parallel interfaces	

Table 3.4 Advantages and disadvantages of SPI.

As was mentioned before that there exist a built-in SPI module in the MCU, and this will be used to interface with the DAC (MAX528) module so as to input the digital data through pin-8 DIN in the DAC.

- **The SPI supports:**

  - Full duplex
  - Synchronous serial transmission
  - Serial Communication with peripheral devices

- **Registers used:**
  - SP0CR1: (Control Register 1)
  - SP0CR2: (Control Register 2)
  - SP0BR: (Baud Rate)
  - SP0SR: (Status Register)
  - SP0DR: (SPI data Register)
  - PORTS: (port s)
  - DDRS: (Data Direction register)

Pin	Function
SCK Serial Data Clock	The SCK signal is used to synchronize the movement of data in and out of the SPI module. This pin is an output or an input depending on whether the SPI is configured as a master or a slave. Data is shifted on one side of the clock edge and sampled on the other. The SCK signal can be configured to accommodate different serial peripheral bus structures.
MOSI Master Output, Slave Input	When the SPI is configured as a master, this pin is used as an output to shift the 8-bit serial data out with the most significant bit first. The pin is used as a slave data input when the SPI is configured as a slave.
MISO Master Input, Slave Output	If the SPI is configured as a master, this pin is utilized as an input. When the SPI is in slave mode, the pin is used as an output.
SS Slave Select	When the SPI is a slave, this pin enables the SPI for an incoming transfer. As a master, this pin should be tied high.

Table 3.5 Pin functions of SPI module of HCS12.

Functional Description:

The SPI has two modes of operations:

1 – Master Mode: in which the SPI generates the synchronizing clock and initiates the transmissions. In our project, we use this mode to communicate with the MAX528 DAC.

2- Slave Mode: in which the SPI depends on the Master peripheral to generate the synchronizing clock and initiates transmissions
The SPI operates in Master Mode by setting the MSTR bit in SPICR1.

Baud Rate Generation

The P-clock divisor is selected by the SPIBR[2:1:0] between 2,4,8,16,32,64,128,256 it controls the rate of the shift register. Through the SCK pin PORTS(6), the selected clock signal also controls the rate of the shift register of the slave SPI or other slave peripheral.

Operation Brief Description

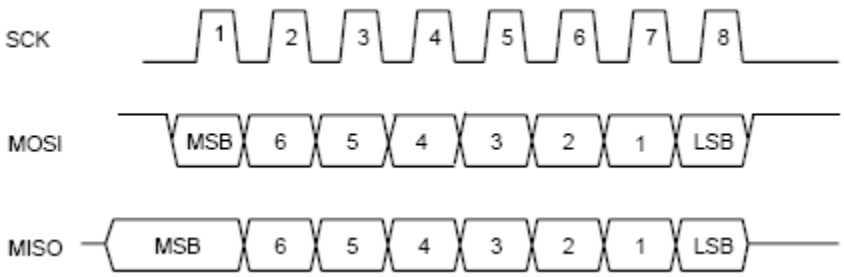


Figure 3.7 SPI timing diagram.

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by writing to the master SPI Data Register. If the shift register is empty, the byte immediately transfers to the shift register. The byte begins shifting out on the MOSI pin under the control of the serial clock.

When a write to the SPI Data Register in the master occurs, there is a half SCK-cycle delay. After the delay, SCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI Control Register 1

Chapter 4:
Controller Firmware

4.1 General Firmware Overview

Both the local controller code and Ethernet controller code were implemented on the same HCS12NE64 microcontroller. Both of them worked independently: the local controller made the sampling and controller action every standard timer interrupt, while the Ethernet controller code made the necessary communication actions every time a TCP packet was received. We have designed our system so that the timer interrupt would have the highest priority amongst all interrupts.

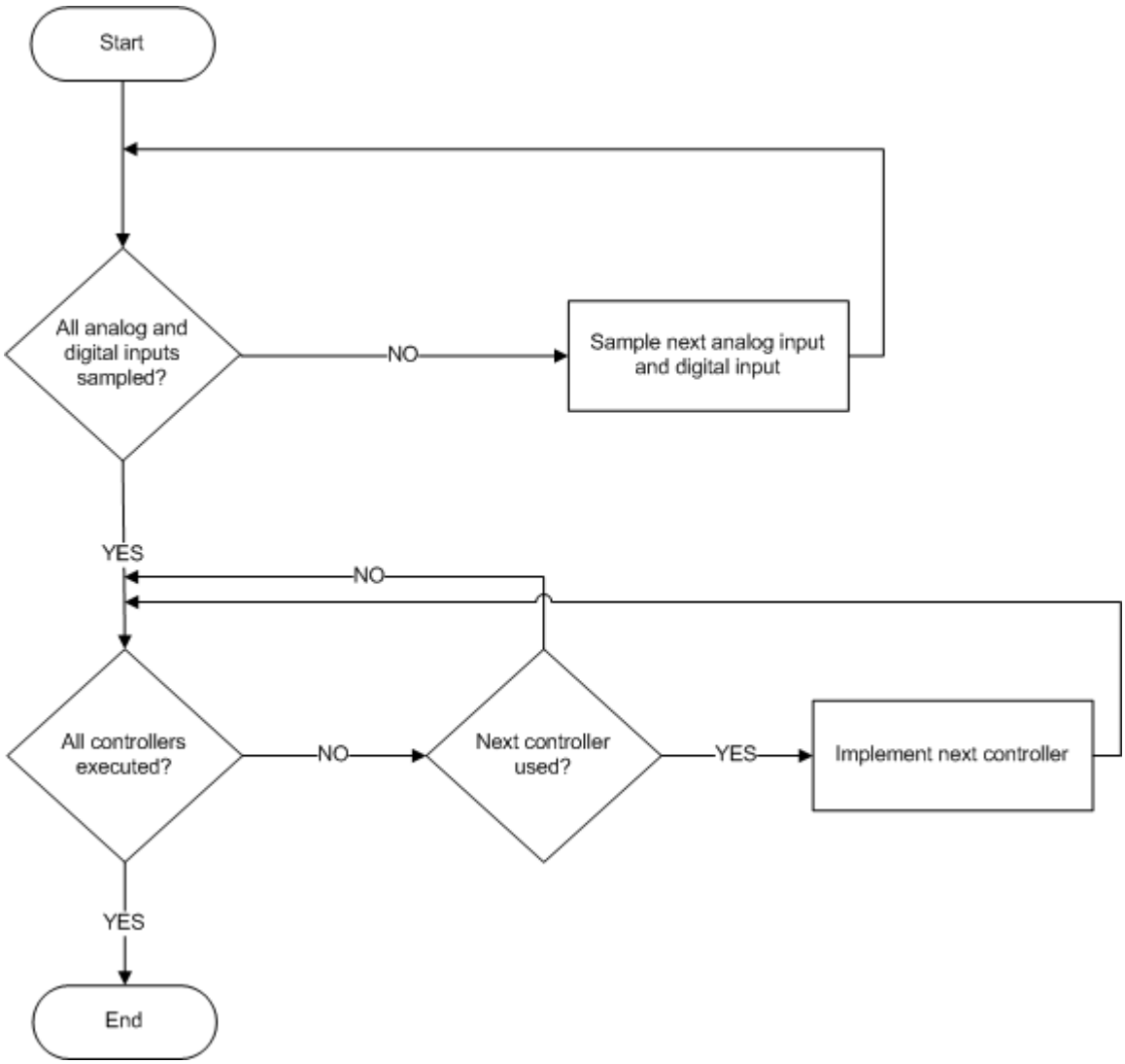


Figure 4.1 Summary of timer interrupt handler.

Every timer interrupt, the local controller samples all of the available channels and assigns them to global array. Therefore, this global array is updated regularly and it is available to be sent to any station on request. Thus, the sampling of the local controller is independent of the sampling of the station.

Each PID or On/Off controller has its own parameters set separately. Moreover, each controller may have its input and output channels set separately. To save time and power, while looping over all of the controllers, each controller is checked first whether it is actually needed or not.

Our local controller supports:

- 8 Analog Input Channels
- 8 Analog Output Channels
- 8 Digital Input Channels
- 8 Digital Output Channels
- 8 On/Off Controllers
- 8 PID Controllers

Each of the above features is implemented in the code as an array of structures. The same structure is used to represent an analog input channel and an analog output channel; but a flag is used to distinguish between them. The same could be said about digital input channels and digital output channels. Each channel has an associated Process ID with it. This is done to prevent any conflict which may occur when two different processes on the same microcontroller access the same channel. Such cases are handled in the **PCSCP** (**Process Control Studio Communication Protocol**).

Each controller has a pointer to its input channel and output channel. Before actually the executing the code for each controller, its mode bit is checked first to see if it is actually needed or not.

When sampling occurs, the value of each input channel structure is updated after reading the actual channel value. After the control action is taken, the actual output channels are written to and their corresponding channel structure values are updated.

Initially, all channels and controllers are set to OFF. The SP and other controller parameters are given some initial value which may be changed by any station. When a station connects to the microcontroller, it starts sending commands to the microcontroller to reserve the required channels and controllers and set the parameters of the required controllers. This is part of PCSCP and is discussed in more detail later in this documentation.

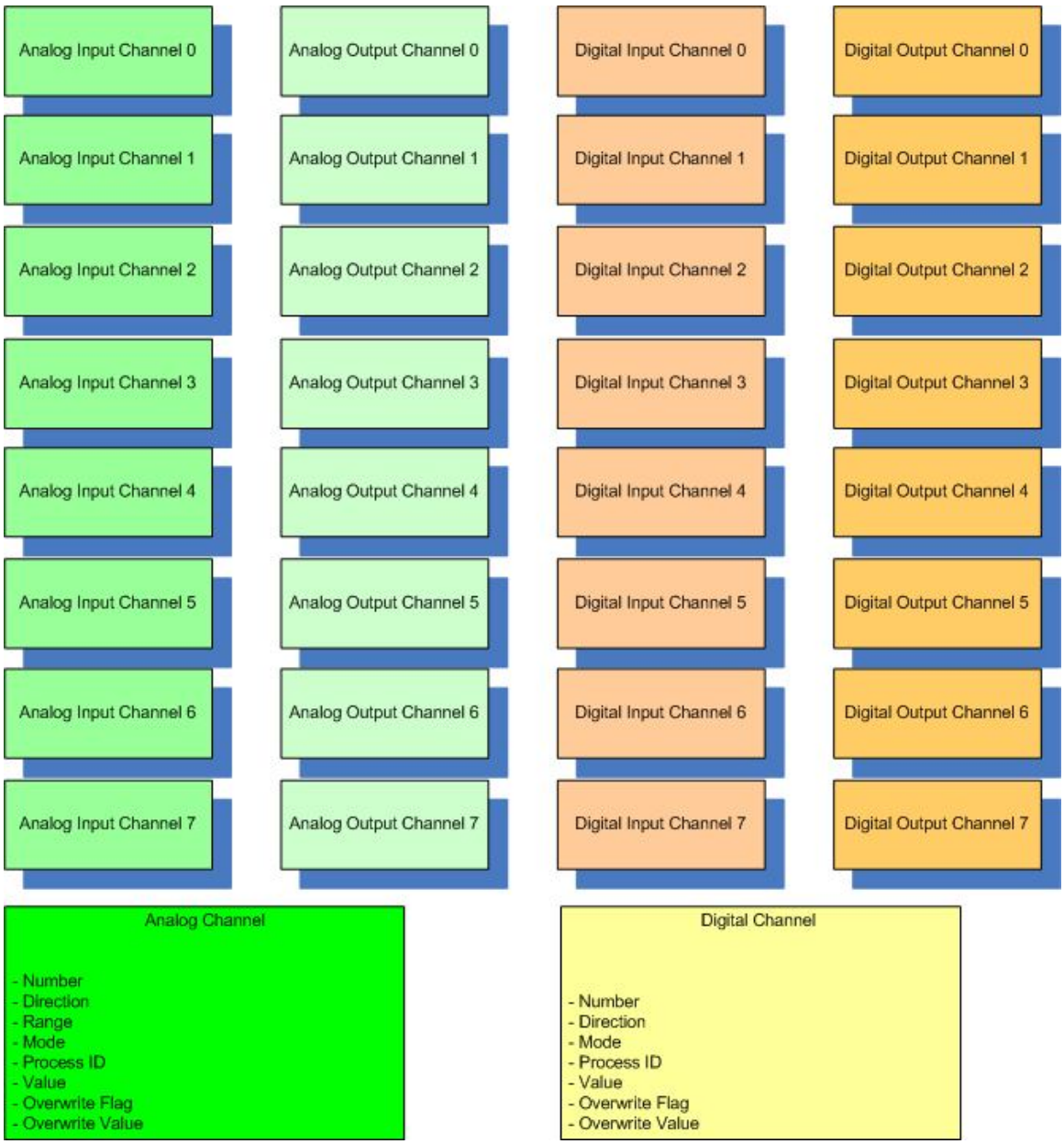


Figure 4.2 Channel structures and their members.

Structure Member	Meaning
Number	the number of the channel (initially set and remains constant and unique)
Mode	whether or not the channel is used
Direction	whether the channel is input or output
Value	the current value read or written to the channel
Overwrite Flag	whether or not a station has overwritten the value. Valid only for output channels.
Overwrite Value	value overwritten by the station.
Range	the electrical range of the channel, either: 0-5V or 0-10V or 4-20 mA.
ProcessID	to distinguish the process it belongs to.

Table 4.1 The meanings of each structure member.

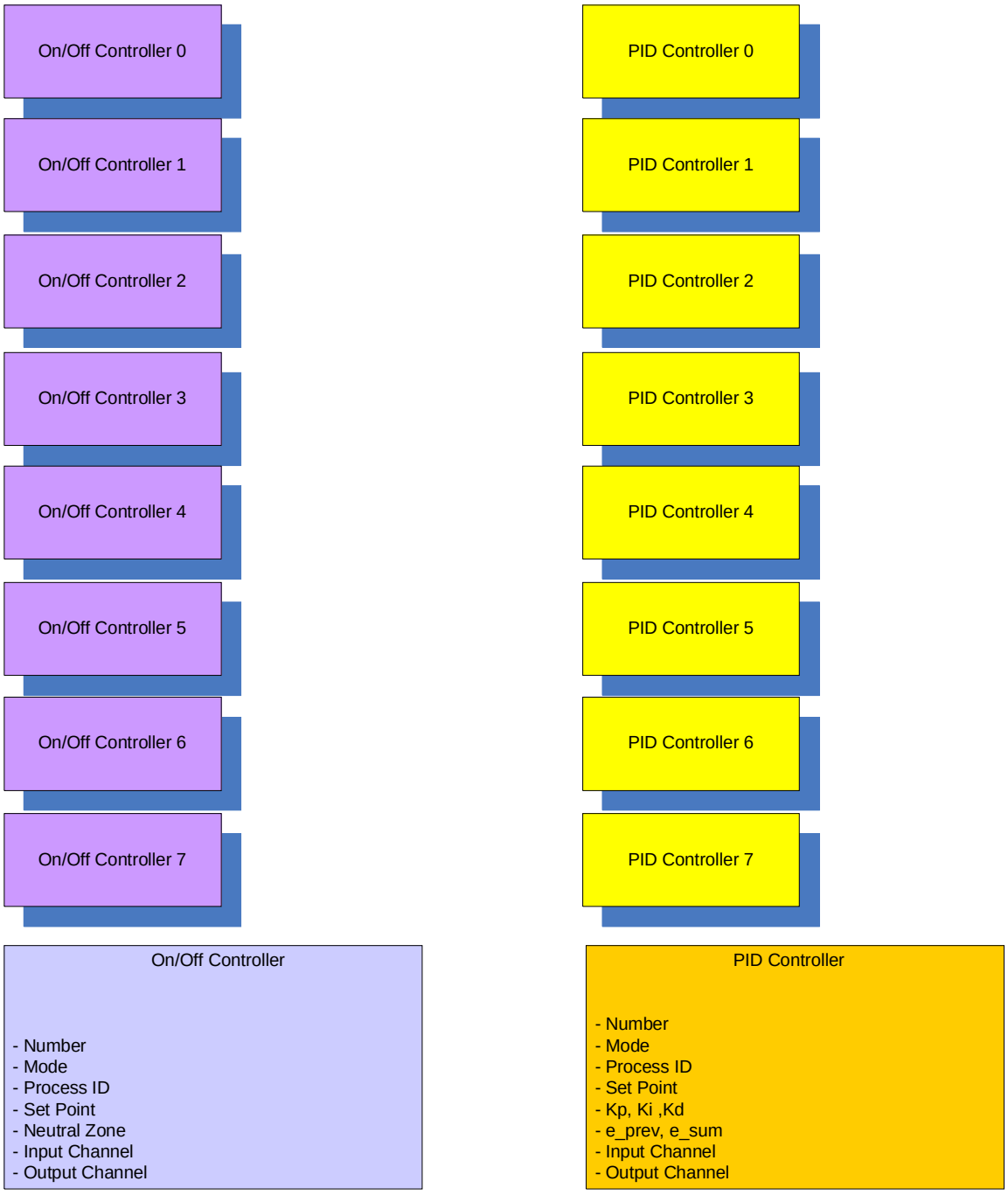


Figure 4.3 Controller structures and their members.

NOTE:
e-prev and e_sum need to be stored in the structure in order to be used in each successive timer interrupt.

4.2 Commands and Requests

The local controller code should be able to receive commands via the Ethernet controller and interpret them in order to modify the controller parameters. Moreover, there is an available option to allow a user to force a value to an actuator. Therefore, the code has the ability to receive commands to overwrite a given analog or digital channel.

The following diagram explains how commands are sent to the local controller to modify any controller parameters. This is considered part of the PCSCP.

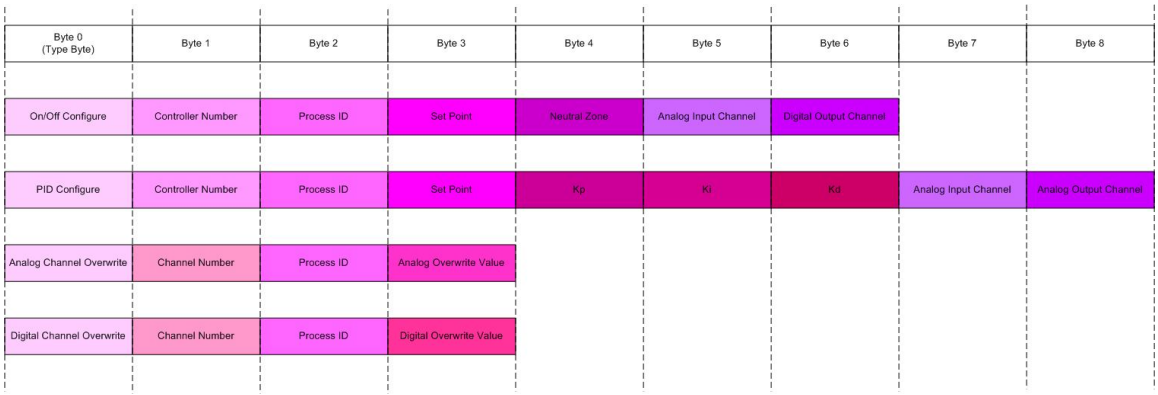


Figure 4.4 Format of the commands to the controller.

NOTE:
Analog overwrite value is in the range 0-255 while digital overwrite value has either the values 0 or 1.

2

Ethernet Controller

Chapter 5: Industrial Control Busses

Chapter 6: Ethernet Interface Hardware

Chapter 7: Ethernet Interface Software

This part includes the hardware and software needed for the implementation of the Ethernet interface between the Software side in the Laptop or PDA, as well as the control loops of the real processes.

After short notes on theoretical background of Reference Models and Protocol Stacks as well as the TCP Protocol used in our project.

Chapter 5:
Industrial Control Busses

5.1 Introduction

A complex automated industrial system — say a manufacturing assembly line — usually needs an organized hierarchy of controller systems to function. In this hierarchy there is usually a Human Machine Interface (HMI) at the top, where an operator can monitor or operate the system. This is typically linked to a middle layer of programmable logic controllers (PLC) via a non time critical communications system (e.g. Ethernet). At the bottom of the control chain is the **fieldbus** which links the PLCs to the components which actually do the work such as sensors, actuators, electric motors, console lights, switches and contactors. A Fieldbus is an industrial network system for real-time distributed control.

Until recently, a PLC would communicate with a slave machine using one of several possible open or proprietary protocols, such as Modbus, Sinec H1, Profibus, CANopen, DeviceNet or FOUNDATION Fieldbus. However, there is now increasing interest in the use of Ethernet as the link-layer protocol, with one of the above protocols as the application-layer (see OSI model). As the following figure shows, Ethernet is now increasing in its popularity over other industrial bus protocols.

5.2 FieldBus

There are a wide variety of concurring fieldbus standards. Some of the most widely used ones include:

- AS-Interface
- CAN
- CANopen
- DeviceNet
- SERCOS_interface
- EtherCAT
- FOUNDATION fieldbus
- HART Protocol
- Industrial Ethernet
- Interbus
- LonWorks
- Modbus
- PROFIBUS
- BITBUS

Fieldbus devices are more flexible than older devices due to the inclusion of a CPU. For example, a pressure transducer can measure process pressure, atmospheric pressure, and also process temperature and supply all three via the fieldbus. Other major cost savings from using fieldbus are due to wiring and installation — the 4–20 mA analogue signal standard requires each device to have its own pair of wires and its own analog connection point at the controller level. Fieldbus eliminates this need by requiring one communication point at the controller level to connect to multiple (100's) of analog and digital points, while at the same time reducing the length of cable runs by connecting to the field devices in a daisy-chain, star, ring, branch, tree style network topology.

5.3 Industrial Ethernet

Industrial Ethernet is the name given to the use of the Ethernet protocol in an industrial environment, for automation and production machine control.

Until recently, a PLC would communicate with a slave machine using one of several possible open or proprietary protocols, such as Modbus, Sinec H1, Profibus, CANopen, DeviceNet or FOUNDATION Fieldbus. However, there is now increasing interest in the use of Ethernet as the link-layer protocol, with one of the above protocols as the application-layer (see OSI model).

During recent years a number of Ethernet based industrial communication system have been established, most of them with extensions for real-time communication. These have the potential to replace the traditional field busses in the long term. Currently the issue stopping most Ethernet fieldbus implementations is the availability of device power. Most industrial measurement & control devices need to be powered from the bus and Power-Over-Ethernet (PoE) does not deliver enough.

- EtherCAT
- Ethernet Powerlink
- SERCOS III
- PROFINET IO
- ETHERNET/IP
- VARAN
- SafetyNET p

A common property of all of these systems seems to be that they are supported by only one PLC/DCS manufacturer for their central logic, and hardly any are compatible with any other.

Some of the advantages are:

- Increased speed, up from 9.6 kbit/s with RS232 to 1 Gbit/s with IEEE 802 over Cat5e/Cat6 cables or optical fiber
- Increased overall performance
- Increased distance
- Ability to use standard access points, routers, switches, hubs, cables and optical fiber, which are immensely cheaper than the equivalent serial-port devices
- Ability to have more than two nodes on link, which was possible with RS485 but not with RS232
- Peer-to-peer architectures may replace master-slave ones
- Better interoperability
- The difficulties of using industrial Ethernet are:
- Migrating existing systems to a new protocol (however many adapters are available)
- Real-time uses may suffer for protocols using TCP (but some use UDP and layer 2 protocols for this reason)
- Managing a whole TCP/IP stack is more complex than just receiving serial data

Serial	Ethernet	Protocol	Network	Standards
Modbus-RTU	Modbus-TCP	TCP/IP		IEC 61158 and IEC 61784
Profibus	PROFINET IO	Isochronous real time protocol (IRT), Real time protocol (RT), Real time over UDP protocol (RTU)	Switches, router and wireless, from 100 Mbit/s up to 1 Gbit/s	IEC 61158 and IEC 61784
DeviceNet (CIP); ControlNet (CIP)	EtherNet/IP (CIP)	TCP/IP; UDP/IP	Switches, router and wireless, from 100 Mbit/s up to 1 Gbit/s	IEC 61158 and IEC 61784; ODVA EtherNet/IP standard
Foundation Fieldbus H1	Foundation Fieldbus High Speed Ethernet (HSE)			
CANopen	Ethernet Powerlink		Ethernet 100Mbit/s	by EPSG (Ethernet Powerlink Standardization Group)
CANopen	EtherCAT	EtherCAT, EtherCAT/UDP	Ethernet 100Mbit/s	IEC 61158, IEC/PAS 62407, IEC 61784-3, ISO 15745-4
	VARAN **V**ersatile **A**utomation **R**andom **A**ccess **N**etwork	VARAN, TCP/IP, Safety	Ethernet 100Mbit/s	VARAN-BUS USER GROUP - VNO

Table 5.1 Summary of different types of Industrial Ethernet busses.

5.4 Theoretical Background in Networking

ISO/OSI Reference Model:

Modern computer networks are designed in a highly structured way. To reduce their design complexity, most networks are organized as a series of layers, each one built upon its predecessor.

The OSI Reference Model is based on a proposal developed by the International Organization for Standardization (ISO). The model is called ISO OSI (Open Systems Interconnection) Reference Model because it deals with connecting open systems - that is, systems that are open for communication with other systems.

The OSI model has seven layers which are:

Layer 1 - Physical

> Physical layer defines the cable or physical medium itself, e.g., thinnet, thicknet, unshielded twisted pairs (UTP). All media are functionally equivalent. The main difference is in convenience and cost of installation and maintenance. Converters from one media to another operate at this level.

Layer 2 - Data Link

> Data Link layer defines the format of data on the network. A network data frame, aka packet, includes checksum, source and destination address, and data. The largest packet that can be sent through a data link layer defines the Maximum Transmission Unit (MTU). The data link layer handles the physical and logical connections to the packet's destination, using a network interface. A host connected to an Ethernet would have an Ethernet interface to handle connections to the outside world, and a loopback interface to send packets to itself.

Layer 3 - Network

> NFS uses Internetwork Protocol (IP) as its network layer interface. IP is responsible for routing, directing datagrams from one network to another. The network layer may have to break large datagrams, larger than MTU, into smaller packets and host receiving the packet will have to reassemble the fragmented datagram. The Internetwork Protocol identifies each host with a 32-bit IP address.

Layer 4 - Transport

Transport layer subdivides user-buffer into network-buffer sized datagrams and enforces desired transmission control. Two transport protocols, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), sits at the transport layer. Reliability and speed are the primary difference between these two protocols. TCP establishes connections between two hosts on the network through 'sockets' which are determined by the IP address and port number. TCP keeps track of the packet delivery order and the packets that must be resent. Maintaining this information for each connection makes TCP a stateful protocol. UDP on the other hand provides a low overhead transmission service, but with less error checking. NFS is built on top of UDP because of its speed and statelessness. Statelessness simplifies the crash recovery.

Layer 5 - Session

The session protocol defines the format of the data sent over the connections. The NFS uses the Remote Procedure Call (RPC) for its session protocol. RPC may be built on either TCP or UDP. Login sessions use TCP whereas NFS and broadcast use UDP.

Layer 6 - Presentation

External Data Representation (XDR) sits at the presentation level. It converts local representation of data to its canonical form and vice versa. The canonical uses a standard byte ordering and structure packing convention, independent of the host.

Layer 7 - Application

Provides network services to the end-users. Mail, ftp, telnet, DNS, NIS, NFS are examples of network applications.

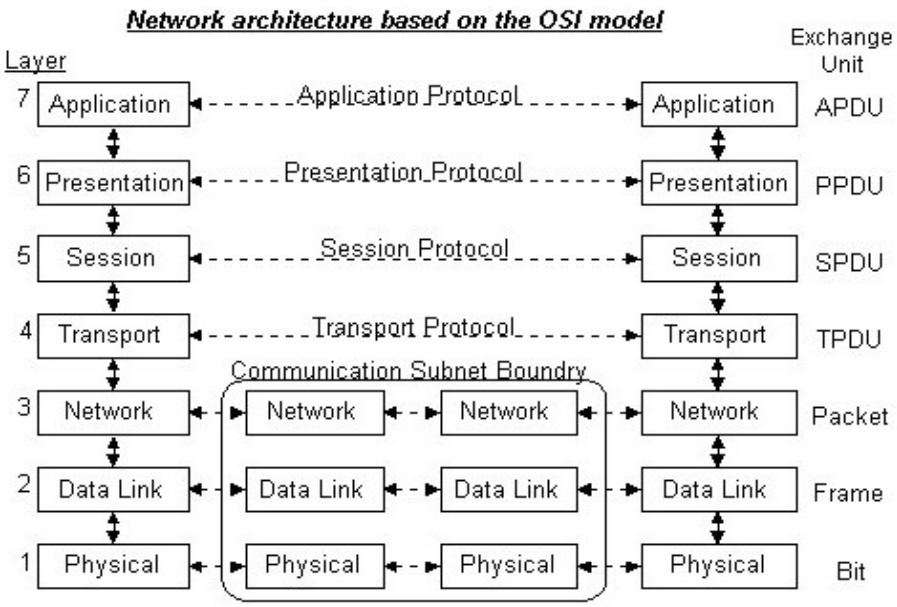


Figure 5.1 Network architecture based on the OSI model.

TCP/IP Network Model

Although the OSI model is widely used and often cited as the standard, TCP/IP protocol has been used by most Unix workstation vendors. TCP/IP is designed around a simple four-layer scheme. It does omit some features found under the OSI model. Also it combines the features of some adjacent OSI layers and splits other layers apart. The four network layers defined by TCP/IP model are as follows.

Layer 1 - Link

This layer defines the network hardware and device drivers.

Layer 2 - Network

This layer is used for basic communication, addressing and routing. TCP/IP uses IP and ICMP protocols at the network layer.

Layer 3 - Transport

This layer handles communication among programs on a network. TCP and UDP fall within this layer.

Layer 4 - Application

End-user applications reside at this layer. Commonly used applications include NFS, DNS, ARP and FTP.

Data Flow within a Protocol Stack

As the reference model indicates, protocols (which compose the various layers) are like a pile of building blocks stacked one upon another. Because of this structure, groups of related protocols are often called *stacks* or *protocol stacks*.

Data is passed down the stack from one layer to the next, until it is transmitted over the network by the network access layer protocols. The four layers in this reference model are crafted to distinguish between the different ways that the data is handled as it passes down the protocol stack from the application layer to the underlying physical network.

At the remote end, the data is passed up the stack to the receiving application. The individual layers do not need to know how the layers above or below them function; they only need to know how to pass data to them.

Each layer in the stack adds control information (such as destination address, routing controls, and checksum) to ensure proper delivery. This control information is called a *header* and/or a *trailer* because it is placed in front of or behind the data to be transmitted. Each layer treats all of the information that it receives from the layer above it as data, and it places its own header and/or trailer around that information.

These wrapped messages are then passed into the layer below along with additional control information, some of which may be forwarded or derived from the higher layer. By the time a message exits the system on a physical link (such as a wire), the original message is enveloped in multiple, nested wrappers—one for each layer of protocol through which the data passed. When a protocol uses headers or trailers to package the data from another protocol, the process is called *encapsulation.*

TCP (Transmission Control Protocol)

- The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks.

- TCP is a connection-oriented transport protocol that sends data as an unstructured stream of bytes. By using sequence numbers and acknowledgment messages, TCP can provide a sending node with delivery information about packets transmitted to a destination node. Where data has been lost in transit from source to destination, TCP can retransmit the data until either a timeout condition is reached or until successful delivery has been achieved. TCP can also recognize duplicate messages and will discard them appropriately. If the sending computer is transmitting too fast for the receiving computer, TCP can employ flow control mechanisms to slow data transfer. TCP can also

communicate delivery information to the upper-layer protocols and applications it supports.

- Unlike TCP's traditional counterpart, UDP (User Datagram Protocol), which can immediately start sending packets on the account of reliability, TCP provides connections that need to be established before sending data. TCP connections have three phases:

    1. connection establishment,
    2. data transfer,
    3. connection termination,

- To establish a connection, TCP uses a three-way handshake. Before a client attempts to connect with a server, the server must first bind to a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. To establish a connection, the three-way (or 3-step) handshake occurs:

    1. The active open is performed by the client sending a SYN to the server.
    2. In response, the server replies with a SYN-ACK.
    3. Finally the client sends an ACK (usually called SYN-ACK-ACK) back to the server.

    At this point, both the client and server have received an acknowledgement of the connection.

- TCP supports many of the Internet's most popular application protocols and resulting applications, including the World Wide Web, e-mail, File Transfer Protocol and Secure Shell.
- TCP has been optimized for wired networks. Any packet loss is considered to be the result of congestion and the window size is reduced dramatically as a precaution. However, wireless links are known to experience sporadic and usually temporary losses due to fading, shadowing, handoff, etc. that cannot be considered congestion. Erroneous back-off of the window size due to wireless packet loss is followed by a congestion avoidance phase with a conservative decrease in window size which causes the radio link to be underutilized. Extensive research has been done on this subject on how to combat these harmful effects. Suggested solutions can be categorized as end-to-end solutions (which require modifications at the client and/or server), link layer solutions (such as RLP in CDMA2000), or proxy based solutions (which require some changes in the network without modifying end nodes).
- For further details and information about TCP operation, the reader is encouraged to read the RFC793 document.

Chapter 6: Ethernet Interface Hardware

(Motorola MC9S12NE64 Ethernet Module)

Ethernet connectivity takes advantage of the very simple TCP/IP protocol set and a microcontroller, thus enabling connectivity from virtually any source with minimal features. To be cost-effective, Ethernet connectivity must be standardized and embedded. Microcontrollers simplify Ethernet networking and bring connectivity to a range of applications. To effectively communicate in these applications, microcontrollers require some fundamental components, including a TCP/IP protocol stack, media access controller (MAC), physical layer (PHY) and sufficient program and data memory resources. The TCP/IP protocol connects the system to the Internet and is the de facto standard of transmitting data over networks. The MAC supports CSMA/CD as defined in the IEEE® 802 standard, and the PHY is a Layer 1 protocol and could be in the form of telephone modem or Ethernet (EPHY).
The Microcontroller used in our project for controlling the real processes as well as an Ethernet Interface is the Motorola MC9S12NE64 microcontroller with Ethernet capability, with the powerful HCS12 core. The MC9S12NE64 Ethernet microcontroller is a single-chip solution that's ideal for affordable embedded connectivity. We used the Freescale demonstration board DEMO9S12NE64 to develop and evaluate our application.

6.1 Motorola MC9S12NE64 Features and Advantages

The MC9S12NE64 is a 112-/80-pin cost-effective, low-end connectivity applications MCU family, it is composed of standard on-chip peripherals including:

- 16-bit central processing unit (HCS12 CPU) which has very interesting features as :

  - Upward compatible with the famous M68HC11 instruction set.
  - Enhanced indexed addressing.
  - Instruction queue.
  - Interrupt control (INT)
  - Multiplexed expansion bus interface (MEBI)
  - Memory map and interface (MMC)
  - Background debug mode (BDM) (Not used in our project)
  - Enhanced debug12 module, including breakpoints and change-of-flow trace buffer (DBG) (Not used in our project)

- 64K bytes of FLASH EEPROM, 8K bytes of RAM.
- Ethernet media access controller (EMAC) with integrated 10/100 Mbps Ethernet physical transceiver (EPHY) which is utilized in our project.
- Two asynchronous serial communications interface modules (SCI), a serial peripheral interface (SPI) which is used for connection with real processes as discussed in the

control section of this documentation. Another serial interface is available which is an inter-IC bus (IIC)

- 4-channel/16-bit timer module (TIM)
- 8-channel/10-bit analog-to-digital converter (ATD), this was used in controlling the real processes as the analog inputs from the processes are converted to digital data so that suitable control action is made, this is discussed in details in the control section of this documentation.
- Up to 21 pins available as keypad wakeup inputs (KWU) for wakeup interrupt function with digital filtering.
- Two additional external asynchronous interrupts.

The inclusion of a PLL circuit allows power consumption and performance to be adjusted to suit operational requirements.

Furthermore, an on-chip bandgap-based voltage regulator (VREG_PHY) generates the internal digital supply voltage of 2.5 V (VDD) from a 3.15 V to 3.45 V external supply range. The MC9S12NE64 has full 16-bit data paths throughout. The 112-pin package version has a total of 70 I/O port pins and 10 input-only pins available.

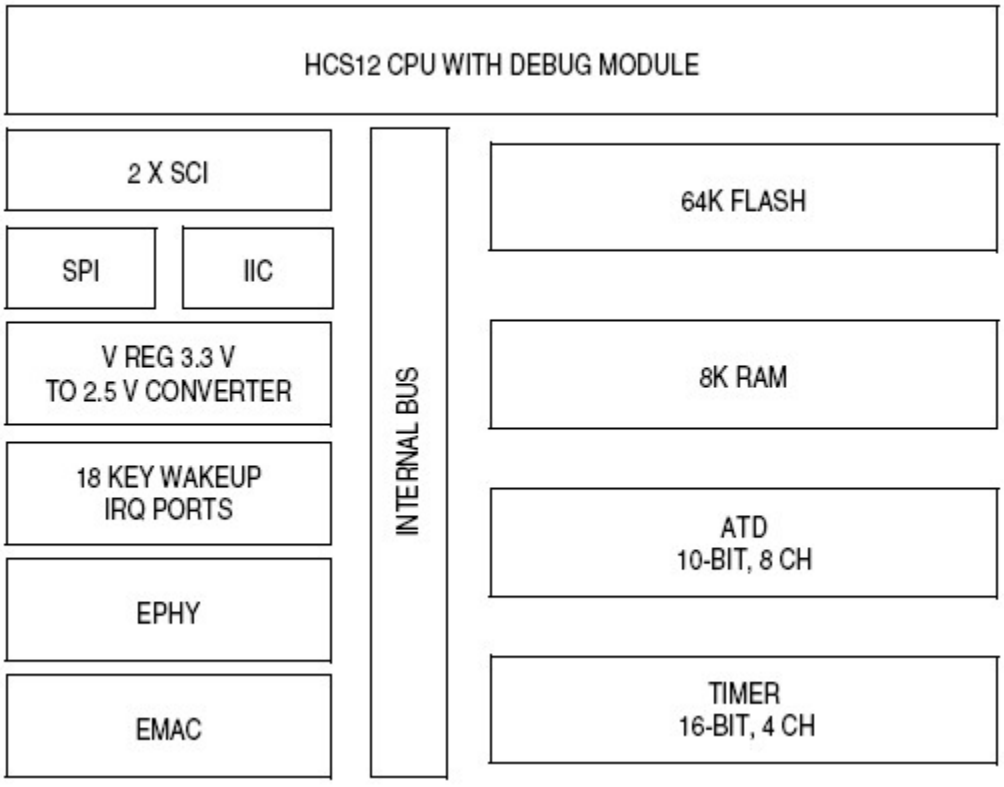


Figure 6.1 Block Diagram of MC9S12NE64 Microcontroller

Ethernet Features:

1- Ethernet Media access controller (EMAC)

- IEEE 802.3 compliant.
- Medium-independent interface (MII)
- Full-duplex and half-duplex modes.
- Flow control using pause frames.
- MII management function.
- Address recognition.
- Frames with broadcast address are always accepted or always rejected.
- Exact match for single 48-bit individual (unicast) address.
- Hash (64-bit hash) check of group (multicast) addresses.
- Promiscuous mode.

2- Ethernet 10/100 Mbps transceiver (EPHY)

- IEEE 802.3 compliant.
- Loop back modes.
- Auto-detection of link capabilities.

3-Two receive and one transmit Ethernet buffer interfaces.

4- Ethertype filter.

5- Loopback mode.

The integrated 10/100 Base-T Ethernet MAC/PHY enables the MC9S12NE64 to keep up with a variety of data rates with flexible memory addressing to support diverse packet sizes. The MC9S12NE64 combines greater processing power and typically larger memory resources (shared 8 KB RAM data/message memory) than an 8-bit solution.

The MC9S12NE64 optimizes the tradeoffs between throughput and memory usage for storing the protocol stack, application code and peripheral drivers, resulting in most of the MCU's resources remaining available for the application. The responsive MC9S12NE64 allows ample time for IP packet disassembly and assembly, for a wide range of IP packet sizes, even in demanding industrial environments. Integrating high internal bus speed and superior addressing modes, such as multi-byte pre/post-decrement indexed addressing, enable the MC9S12NE64 to undertake Ethernet networking and remote monitoring or control which is made in our project.

6.2 Demonstration Board Layout and Specifications

As mentioned before, we used the Freescale demonstration board DEMO9S12NE64 to develop and evaluate our project, it has a Maximum Clock Speed of 25 MHZ at 3.3 V (12.5-MHZ Bus) for 40 ns minimum instruction cycle time, it requires power supply of 6 to 12 VDC 0.75 Amp supplied externally.

The board is supplied with 2 LED's, 2 switches, and an analog input (potentiometer). This was used just for testing and debugging not for real use in the project implementation as the user (client) should be able to control the real-processes remotely via software from a Laptop or a PDA.

The following figures show the top and bottom side layout of the DEMO9S12NE64.

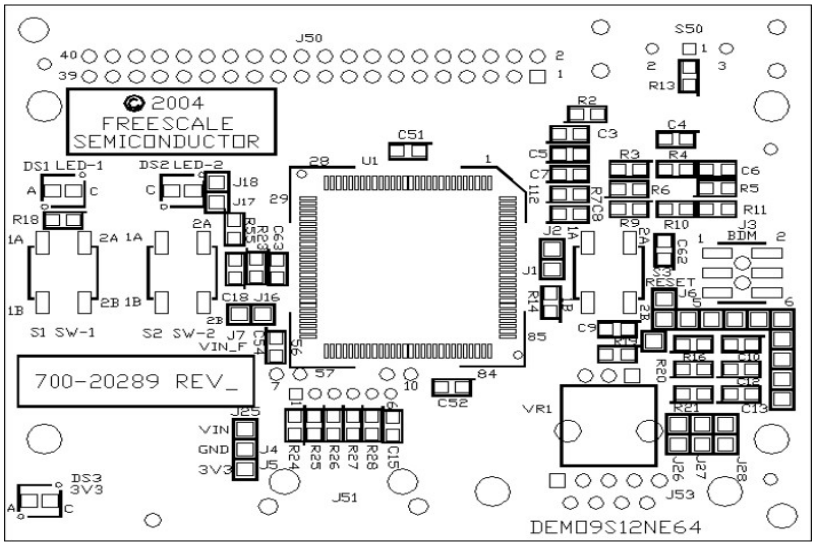


Figure 6.3 DEMO9S12NE64 Top Side

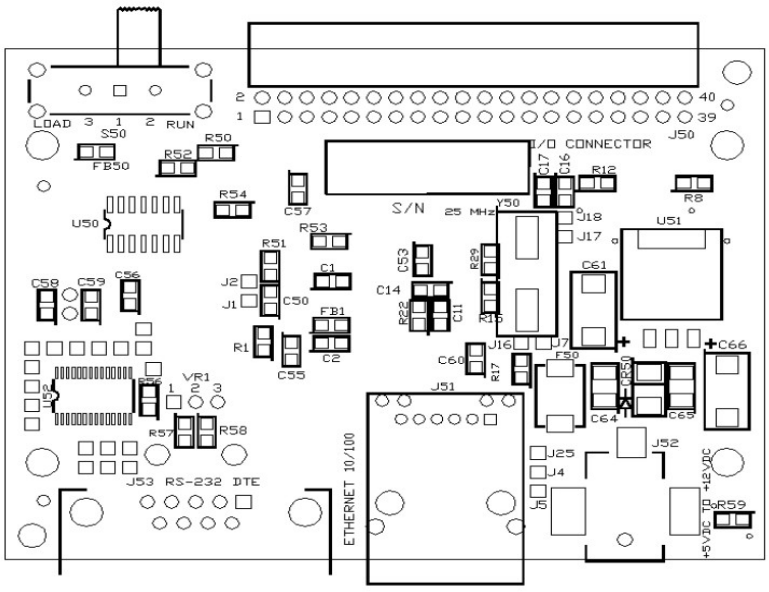


Figure 6.3 DEMO9S12NE64 Bottom Side

Chapter 7: Ethernet Interface Software

This chapter presents all the detailed information about the TCP/IP Stack used to manage the network activity of the microcontroller module over the Ethernet as well as the PCSCP (Process Control Studio Communication Protocol); our main application that enables clients to remotely connect to control loops via portable devices which makes use of the TCP/IP Stack implemented.

Here, the TCP/IP protocols are not discussed in details, readers who are interested in the protocols details are encouraged to read the related RFC documents.

7.1 TCP/IP Stack

A TCP/IP stack defines a set of protocols that allows network devices to connect to a specific device and exchange data on a network. These protocols, defined by RFC (request for comments), enable an embedded device to send email, serve web pages, transfer files, and provide other basic connectivity functions. Figure 3 is a simplified illustration of a user application working through the TCP stack model and illustrates how a TCP/IP stack and the MC9S7NE64 Ethernet controller fit into the system.

The TCP/IP Stack used in our project is the Freescale OpenTCP Stack developed by Viloa Systems. OpenTCP is a highly robust and portable implementation of the TCP/IP and Internet application-layer protocols intended for the implementation of TCP/IP functionality in a constrained environments. This project contains ports, optimizations, and extensions of OpenTCP for Freescale microcontrollers including the 16-bit MC9S7NE-Family.

7.1.1 OpenTCP TCP/IP Stack Structure

The OpenTCP Stack is composed of a suite of programs (C-files) that provide services to custom TCP/IP applications as the Connection Server implemented in our project as well as other standard applications that are not implemented in our project as HTTP Servers, FTP Servers, Mail Clients, … etc. It is entirely written in C which allows easy porting to any existing microcontroller platform for which a C compiler is available.

The TCP/IP Stack is implemented in a modular fashion, with all of its services creating highly abstract layers. Potential users don’t have to know the details of all the intricacies of lower TCP/IP layers specifications in order to effectively use it.

In addition to protocol modules source and header files; such as “tcp.c”, “udp.c”, “ip.c”, “tcp_ip.h”, ”ip.h”, ”ethernet.h”, some global header files reside in the stack; they are essential for the proper execution of the application as they set the necessary global variables and definitions so, these header files must be included in the main application.

Common Header files include:

- “system.h”

  This file holds #defines of Network Transmit Buffer Size, a pointer to that buffer, the master clock (Interrupt driven free-running clock) and the structure that holds all of the network-related information about the network interface.

  It also defines macros to:

  - Read/Write data (either byte or word) from the Ethernet Controller
  - Read data to a buffer in memory
  - Write data from a buffer to Ethernet Controller
  - Check if there is new data in the Ethernet controller (this macro must be invoked periodically)
  - Initialize reading the received Ethernet frame or sending an Ethernet packet from a given address in the Ethernet Controller
  - Discard Ethernet packet when not used anymore

- “datatypes.h”

  This file holds #defines of data types used in the OpenTCP sources so that recompiling for another MCU is easier even when the other MCU is using different size default values; it defines constants for BYTE, WORD, …

- “globalvariables.h”

  It holds declarations of global variables that are commonly used in other OpenTCP modules as well as OpenTCP applications in general. It consists basically of a group of externs.

- “mottypes.h”

  It holds common definitions for core registers block. MCU register definition is done in separate files, describing each peripheral register block as a datastructure.

- “debug.h”

  This file contains debug settings for OpenTCP and its modules. Debugging in this case only assumes a function that sends a null-terminated string over a serial port.

- "mc9s7ne64.h"

  This implements an IO devices mapping.

- "timers.h"

  OpenTCP timers interface file which includes timers function declarations, constants, etc.

In addition the file "address.c" contains definitions for hardware and IP addresses (MAC address, IP address, Subnet Mask, …) of the microcontroller that can be modified as needed.

7.1.2 How the TCP/IP Stack works

The OpenTCP TCP/IP Stack divides the TCP/IP Stack into multiple layers. As mentioned before, data flows upwards and downwards within stack layers sequentially. For our TCP/IP stack software, the C-code of a protocol of a specific layer resides in a separate C-source file, higher levels protocols include C-header files containing functions and variables of these lower levels in order to make use of the lower level protocol. As an example, services and API's (Application Programming Interfaces) call lower levels protocols; HTTP application calls TCP protocol which in turn calls IP protocol and so on.

However, unlike the TCP/IP reference model the OpenTCP TCP/IP stack can directly access one or more layers which are not directly below it, of course that doesn't necessary work for all applications, some services and applications data need intelligent processing in the previous adjacent layer before it can be passed to the next one.

The OpenTCP TCP/IP Stack achieves *multitasking*; it is live in the sense that the different layers not only act when a service is requested but also when events like new packet arrival or time-out occur so, layers are able to perform some timed operations asynchronously and handle different events simultaneously. As a result, user's main application can perform its main task without having to manage the stack as well.

7.1.3 Using the TCP/IP Stack

The OpenTCP Stack consists of source and header files in C which support many protocols and services like ARP, BOOTP, DHCP Client, DNS Client, HTTP Web Server, POP3 Client, SMTP Client, TCP, IP, TFTP Client, UDP, and ICMP(for Ping). The Stack doesn't implement all modules and applications normally present in the TCP/IP stack. However, they can always be implemented as a separate task or module if required, provided that proper C-header files are included.

Our Project application (The Connection Server), however, doesn't implement all protocols and services supported by the OpenTCP TCP/IP Stack. HTTP, FTP, DNS and Mail Clients (POP3, SMTP) for example are not used in our project but they can be implemented in other applications by including the corresponding header files as mentioned before. Including the appropriate header files in the project is critical for correct compilation and execution, since each of the modules comprising the stack resides in its own file.

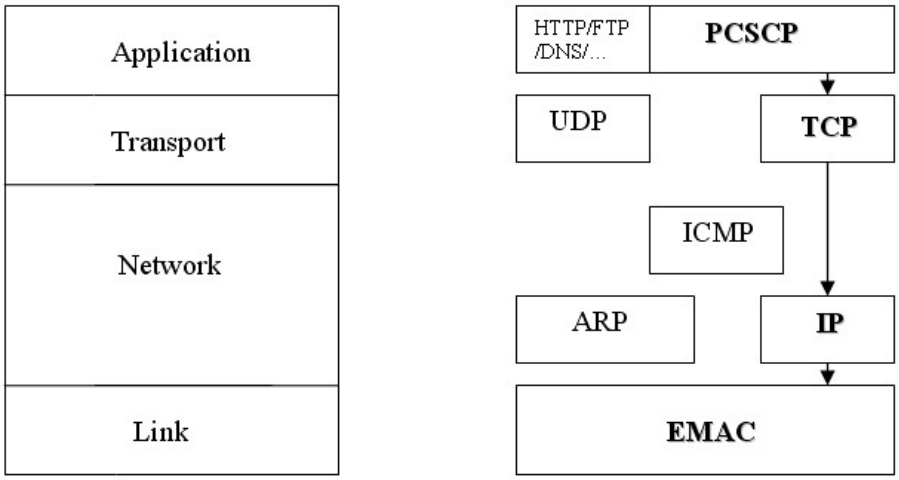


Figure 7.1 Project's utilization of the stacks.

The project executes its main application module ("main.c" source file) which contains the appropriate header files for the used modules in addition to the function calls to the relevant functions in other C-files

The Main module carries out the most important job for the proper execution of the project. In the main module, processor-dependent stuff (I/O ports, timers, …) are initialized, enabling/disabling interrupts (either globally or individually) is decided, the clock source is selected, setup of individual ports is done, bus clock, baud rate, oscillator frequency are set, relevant global variables and definitions are initialized and set.

In our project, we created a source file for our application "pcscp.c" (Process Control Studio Connection Protocol), there; we included the header files for the used modules and the common header files necessary for the proper operation of the whole project. In the main file and at the start of the main function, we put a function call after system initialization that calls a function in the pcscp.c source file; this function is responsible for initialization in addition to other tasks. These details will be thoroughly discussed in the next section.

7.1.4 Protocols Used

In our Connection server application, we used the TCP protocol of the Transport layer, that was due to the fact that our experimental tests showed that TCP transmission through wireless network is far more reliable than UDP despite all its drawbacks including overhead. UDP showed very poor performance and reliability during our tests for wireless connections.

Next, we shall briefly discuss the modules of common protocols implemented in our application, namely, modules of TCP protocol and IP protocol along with frame formats of these protocols as well as the Ethernet Frame Format.

- **TCP:**

  As mentioned before, the TCP module resides in a separate source file which contains the API functions to be called from higher-layers applications such as our main application of the project.

  Functions are available to:

  - Initialize the TCP module; initialize all sockets and corresponding TCP Control Blocks to known state.
  - Create a socket (Obtain a free socket from TCP socket pool)
  - Release a TCP socket once the application does not need the TCP socket anymore.
  - Put TCP socket to listen on a given port (provided that the socket is a server socket)
  - Initialize connection establishment towards remote IP& Port Number (provided that the socket is a client socket)
  - Send user data over TCP using given TCP socket.
  - Close connection over a given socket.
  - Get current state of the socket; whether it is free, connected, listening, closed , ...
  - Check and process the received TCP frame.
  - Create and send TCP packet, based on data supplied as function parameters and data stored in the socket's TCP Control Block.

  The corresponding header file contains function prototypes as well as constants, declarations, ... , it also includes the structure that holds header fields from the received TCP packet *"tcp_frame"* and the structure that holds various fields used to keep track of TCP socket states, settings and event listener function *"tcb"* (TCP Control Block). Instances of these two structures are created and used within functions in the source file.

TCP Frame Format

Apart from data that exist in the network buffer, TCP Frame Header fields are the members of the *"tcp_frame"* structure, they are:

sport: Source port
dport: Destination port
seqno: Sequence number
ackno: Acknowledgement number
hlen_flags : Header length and flags
window : Size of window
checksum : TCP packet checksum
urgent : Urgent pointer
buf_index : Offset from the start of network buffer

- **IP:**

  Functions in the IP source file are available to:

  - Send an IP Frame, it performs all of the necessary preparation in order to send out an IP packet.
  - Process the received Ethernet Frame by checking necessary header information.
  - Construct checksum of the IP header.
  - Check IP frame's checksum; checksum of an IP packet is calculated and compared with the received checksum.

  The corresponding header file contains function prototypes as well as constants, declarations, reserved addresses … , it also includes the structure holding information about various fields of the IPv4 header *"ip_frame"*, an nstance of this structure is created and used within functions in the IP source file.

  IP Packet Format

  IP Packet Header fields are the members of the *"ip_frame"* structure, they are:

  Vihl : Version & Header Length field
  Tos : Type Of Service
  tlen : Total Length
  id : IP Identification number
  frags : Flags & Fragment offset
  ttl : Time to live
  protocol : Protocol over IP
  checksum : Header Checksum
  sip : Source IP address
  dip : Destination IP address
  buf_index : Next offset from the start of network buffer

- **Ethernet:**

  The *"ethernet.h"* header file contains definitions for constants and buffer addresses as well as the structure that holds information about the Ethernet frames. Function to write and read from the Ethernet controller are defined in another file *"system.c"*

  Ethernet Frame Format

  Ethernet Frame Header fields are the members of the *"ethernet_frame"* structure, they are:

  destination[ETH_ADDRESS_LEN] : destination hardware address as read from the received Ethernet frame
  source[ETH_ADDRESS_LEN] : Source hardware address as read from the received Ethernet frame
  frame_size : Size of the received Ethernet frame
  protocol : Protocol field of the Ethernet header, the stack works with two protocols; IP and ARP
  buf_index : Address in the Ethernet controller's buffer where data can be read from.

7.2 PCSCP (Process Control Studio Communication Protocol)

Our main application performs the function of a Connection Server that is responsible for managing requests or commands from remote users; the microcontroller (Ethernet Controller) acts as a server which accepts connections from PCS software via Laptops or PDA's, it then decides what to do depending on the type of received packets; whether it should forward the received packet to the control loop if the packet contains values or set points to be changed in the control loop, or returns acknowledgement for another type of packets representing new connection or channels to be monitored.

These details shall be discussed thoroughly later in this section.

7.2.1 The Main Application Features

- Connection between a computer terminal (Laptop or PDA) and the micro-controller (Ethernet module) is established via a predefined handshaking scenario to handle any errors during connection stage or any illegal requests or commands.
- The operator at the computer terminal can monitor more than one process simultaneously depending on the number of free sockets available at the micro-controller.
- The microcontroller can handle multiple requests at the same time depending on free sockets available, it can also control more than one process depending on free channels available as mentioned in the control part in this documentation.

- After connection establishment, the operator at the computer terminal can just monitor the remote process or control it; by changing set-points or any other control parameters.
- The operator can terminate the connection at any time
- An idle connection is automatically terminated after a predefined period of time.
- A status message is sent to the remote computer terminal to show whether the request or command has been accepted or not, the error messages show clearly where the error lied.
- The application parameters like available sockets, IP address, … can be easily modified and updated as we need depending on the implied conditions.

7.2.2 PCSCP Steps

The protocol steps proceed as follows:

1- The Server (microcontroller) allocates and opens a server socket called *"welcome socket"*. This is the common socket used to receive connection requests from all clients (computer terminals). All clients should first send their requests to this socket before another dedicated one is opened for them.

2- If connection is approved, the server opens a dedicated connection socket for this client from the pool of available sockets, and then it sends a message to the client to inform him that connection is approved and tell him the port number of the connection socket. (Then proceed to step(4))

3- If an error had occurred (probably because no free sockets are available), an error message is reported to the client showing the error type.

4- The client then sends to the connection socket a message with the channels and controllers he wants to monitor/control.

5- If the request is approved, the server sends a message to the client to inform him with the approval. (Then proceed to step(7))

6- If an error had occurred (probably because the requested channels are already in use), an error message is reported to the client showing the error type.

7- At this stage, the client can monitor or control the channels he requested, the connection protocol forwards any incoming messages to the control phase to take the suitable control action, and forwards the reply from control phase to the client.

8- If the client issues a terminate requests, the sever closes the dedicated connection socket and acknowledges the request.

The next diagram shows a simple flowchart of the connection requests and commands sequence.

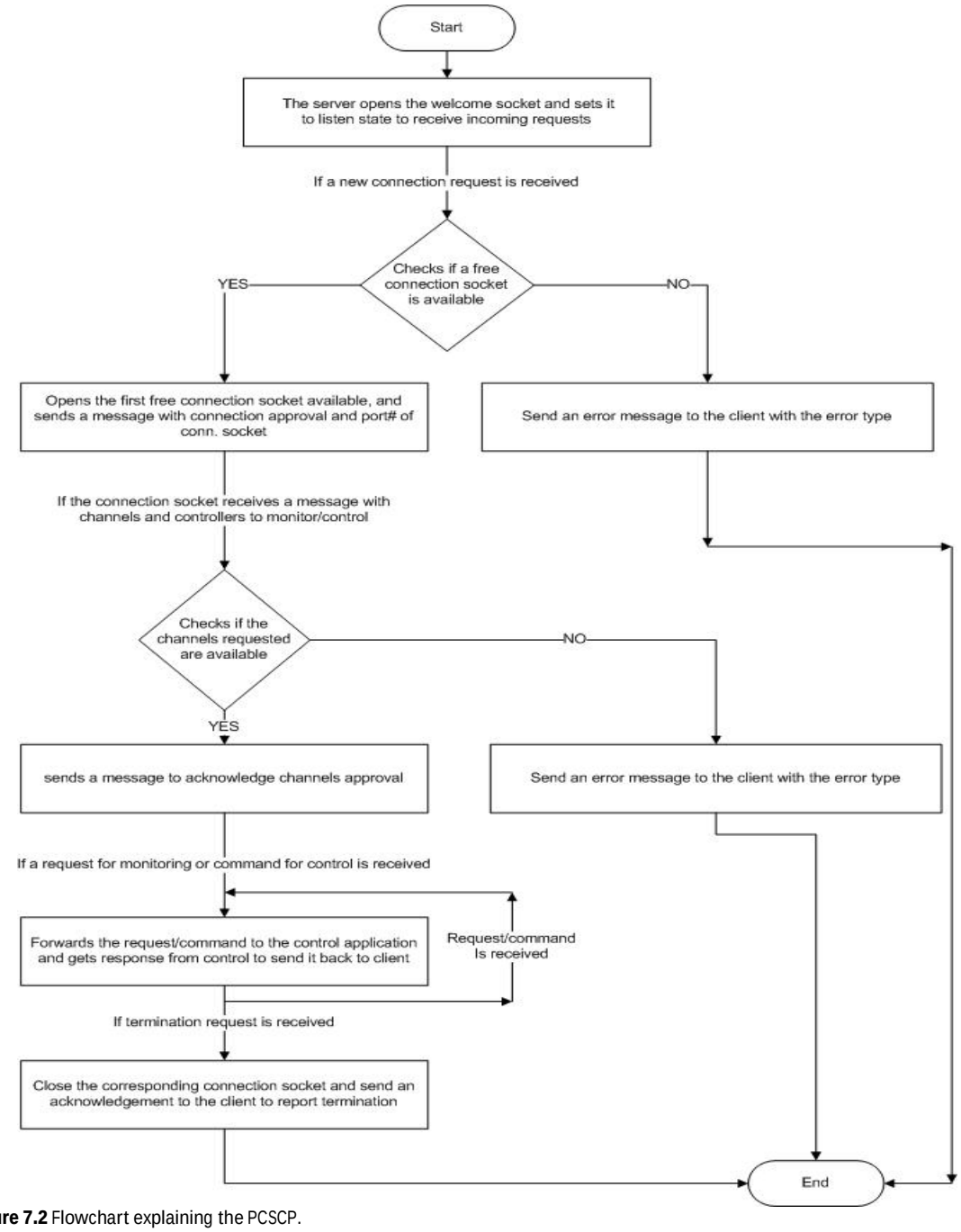


Figure 7.2 Flowchart explaining the PCSCP.

NOTE:

- Any data sent to/from the server or client is accompanied with a *"type byte"* at the start. The purpose of the type byte is to show the meaning of the data after it. The server checks that byte to know whether the data indicates new connection or channels to be monitored or anything else.
- An idle connection socket will be automatically closed after a predefined period of time.

7.2.3 PCSCP Functions

In our *"pcscp.c"* file which is our main application; after includes and definitions for type bytes, welcome socket port number and other stuff, the protocol functions are implemented.
A brief description of the functions implemented in our PCSCP application will be presented in this section.

- ✓ **TCP Application Initialize (tcp_appl_init)**
  Initializes resources needed for the TCP socket application. Here, the welcome socket is created, opened and set to *listen* state in order to receive requests from clients. It also sets the all the flags of connection sockets to zero (not reserved yet)
- ✓ **Find Free Socket (find_free_socket)**
  This function is invoked to find the first free socket from the socket pool; it searched for the first socket whose flag equals zero which indicates that it's free.
- ✓ **TCP Application Event Listener (tcp_appl_eventlistener)**
  It represents the function that will be called when the welcome socket receives a connection. It handles the request depending on the event type on the welcome socket. After regular TCP handshaking operation, packets containing data are stored in an array of bytes and sent to another function (take_action) to start/continue the handshaking scenario between the server and the client at the computer terminal.
- ✓ **TCP Application Event Listener1 (tcp_appl_eventlistener1)**
  It is similar to the previous function but it's called when the dedicated connection sockets receive requests or commands, in case of data event after TCP regular handshaking, it stores the data in an array of bytes then it calls the function (take_action1) to deal with the command or request from the client with the suitable reply.
- ✓ **TCP Take Action (take_action)**
  It consists mainly of a switch-case structure; it checks the type byte of the received data to interpret what this data means and take appropriate decisions. For example, in case of a new connection request, it tries to find a free socket using (find_free_socket) function and if it succeeds it sends back a packet of data indicating connection approval and the port number of the connection socket, else it sends a packet of data indicating an error and the error-type. And so on as explained in the previous section.
- ✓ **TCP Take Action1 (take_action1)**
  Its structure is similar to the previous function but it's for taking decisions for data from connection sockets, not the welcome socket.
- ✓ **TCP Application Send (tcp_appl_send)**
  It is invoked within other functions to send TCP message to some predefined host; it places the message to be sent in the network buffer after an offset for the TCP header, and then invokes another function in the TCP/IP stack that sends the packet through the Ethernet interface.

This concludes the description of the PCSCP protocol, our main application of the Ethernet Interface implemented in our project.

7.3 Other Software Tools Used

During different project phases, we used some software tools for compiling, debugging and testing. These software tools include:

- **Freescale CodeWarrior IDE :**

We used CodeWarrior for compilation of our C code, for building the project and for programming it to the flash memory of the microcontroller. It also has very useful utilities including the debugger that can execute the program step by step to show where the error has occurred (if there is any). All these functions are available through a very user-friendly interface.

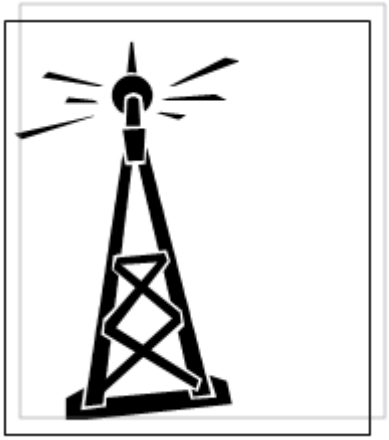
- **Ethereal :**

Ethereal is a perfect packet sniffer, network and protocol analyzer. We used Ethereal extensively in troubleshooting during the testing phase as it captures the out-coming and in-going packets through a network interface card and analyzes the packets contents including header fields of all protocols involved, this was very useful for testing.

This concludes the Software part of the Ethernet Interface implemented in our project.

3

Wireless Accessibility



Chapter 8: Wireless Networking Concepts

Chapter 9: Wireless Security

Wireless technologies*, in the simplest sense, enable one or more devices to communicate 1without physical connections—without requiring network or peripheral cabling. Wireless technologies use radio frequency transmissions as the means for transmitting data, whereas wired technologies use cables. Wireless technologies range from complex systems, such as Wireless Local Area Networks (WLAN) and cell phones to simple devices such as wireless headphones, microphones, and other devices that do not process or store information. They also include infrared (IR) devices such as remote controls, some cordless computer keyboards and mice, and wireless hi-fi stereo headsets, all of which require a direct line of sight between the transmitter and the receiver to close the link. A brief overview of wireless networks, devices, standards, and security issues is presented in this section.*

In our project, we aimed to achieve wireless accessibility to the microcontrollers so that control engineers walking through a process plant may be able to control and monitor remote processes no matter where they are.

Chapter 8:
Wireless Networking Concepts

8.1 Wireless Networks

Wireless networks serve as the transport mechanism between devices and among devices and the traditional wired networks (enterprise networks and the Internet). Wireless networks are many and diverse but are frequently categorized into three groups based on their coverage range: **Wireless Wide Area Networks** (**WWAN**), **WLANs**, and **Wireless Personal Area Networks** (**WPAN**).

WWAN includes wide coverage area technologies such as 2G cellular, Cellular Digital Packet Data (CDPD), Global System for Mobile Communications (GSM), and Mobitex. WLAN, representing wireless local area networks, includes 802.11 (Wi-Fi), Hiper LAN, and several others. WPAN represents wireless personal area network technologies such as Bluetooth and IR.

All of these technologies are "tetherless"—they receive and transmit information using electromagnetic (EM) waves. Wireless technologies use wavelengths ranging from the radio frequency (RF) band up to and above the IR band. The frequencies in the RF band cover a significant portion of the EM radiation spectrum, extending from 9 kilohertz (kHz), the lowest allocated wireless communications frequency, to thousands of gigahertz (GHz). As the frequency is increased beyond the RF spectrum, EM energy moves into the IR and then the visible spectrum.

8.2 Elements of a Wireless Network

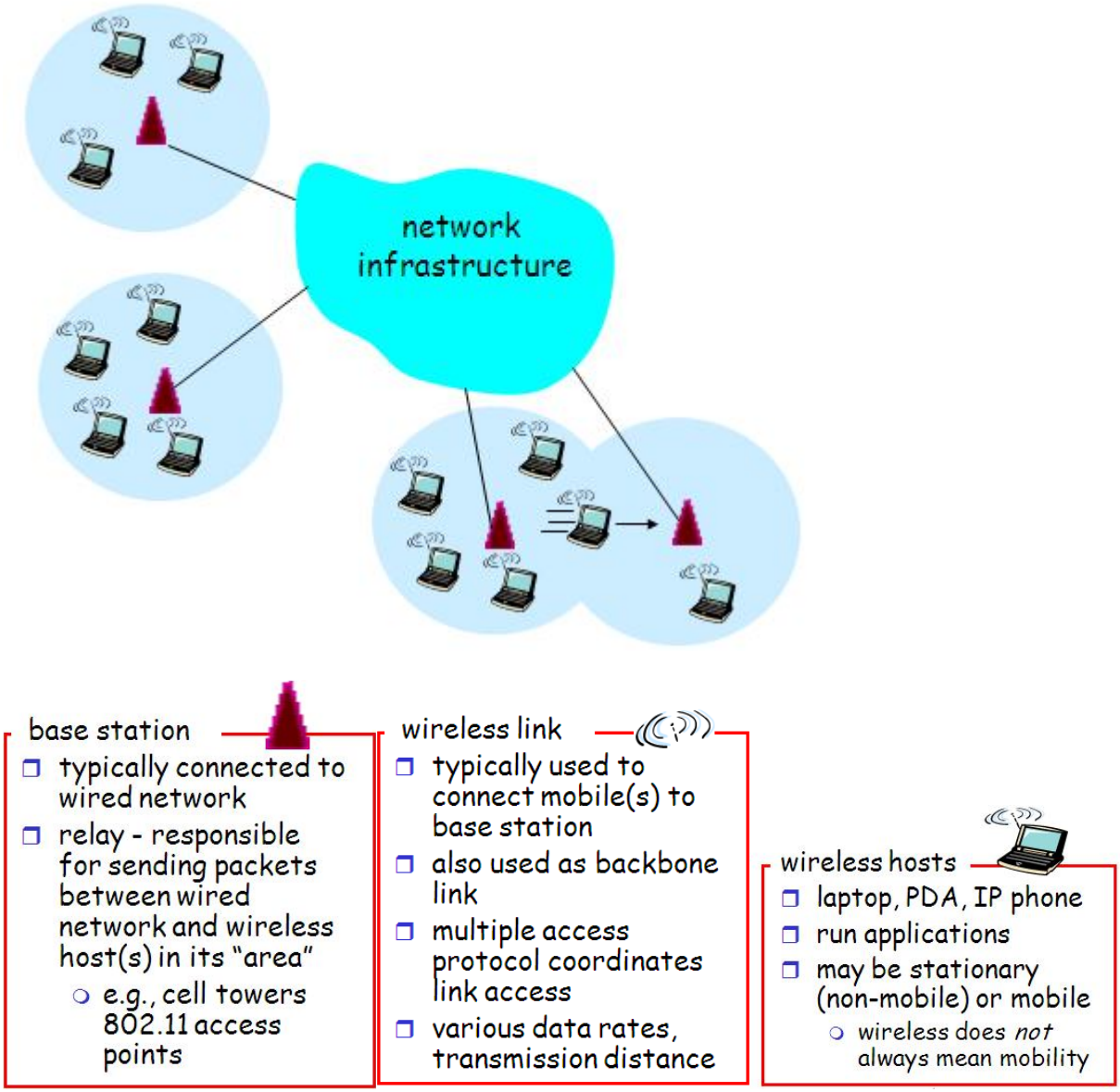


Figure 8.1 Elements of a wireless network.

The basic elements of a wireless network are:

- **Access Point** (or **Base S**tation): a device that connects wireless communication devices together to form a wireless network. The WAP usually connects to a wired network, and can relay data between wireless devices and wired devices. Several WAPs can link together to form a larger network that allows "roaming". WAPs have IP addresses for configuration. There are two types of access points:

1- Dedicated **Hardware Access Points** (**HAP**) such as Lucent's WaveLAN, Apple's Airport Base Station, Linksys, 3com or WebGear's AviatorPRO. Hardware access points offer comprehensive support of most wireless features.

2- **Software Access Points** which run on a computer equipped with a wireless network interface card as used in an ad-hoc or peer-to-peer wireless network. The Vicomsoft InterGate suites are software routers that can be used as a basic Software Access Point, and include features not commonly found in hardware solutions, such as Direct PPPoE support and extensive configuration flexibility, but may not offer the full range of wireless features defined in the 802.11 standard.

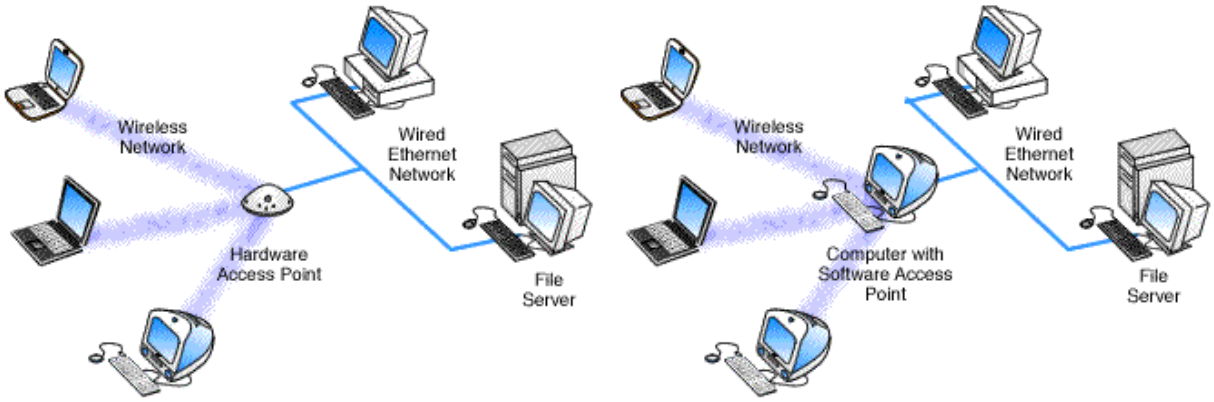


Figure 8.2 Comparing Software Access Point and Hardware Access Point.

- **Wireless Link**: is used to connect the different wireless devices together. It has various data rates and protocols. Also used within a network to connect routers, switches and other network equipment.
- **Wireless Hosts**: usually means the different devices (laptops and PDAs) interconnected in a network. The devices may be stationary or mobile and this shall be discussed later.

8.3 Ad Hoc and Infrastructure Networks

Wireless networks may be classified according to their structures into **Ad Hoc Networks** and **Infrastructure Networks**.

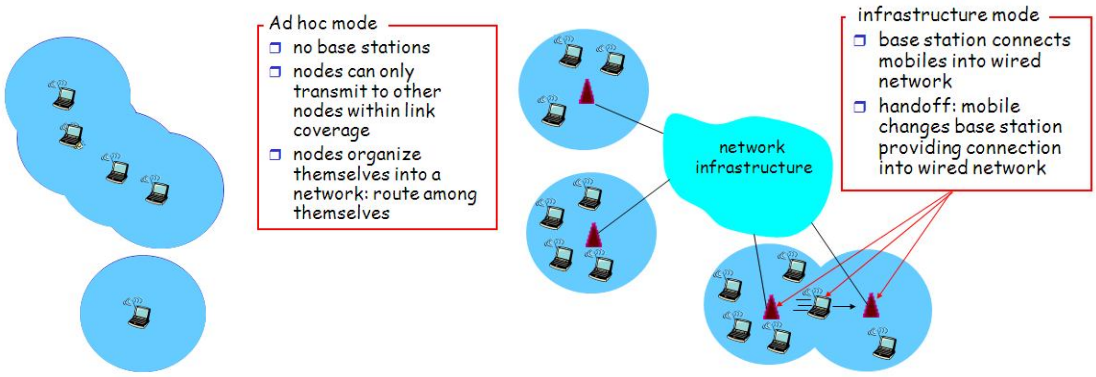


Figure 8.3 Comparing Ad hoc mode and infrastructure mode.

A wireless ad-hoc network, also known as **IBSS** - **Independent Basic Service Set**, is a computer network in which the communication links are wireless. The network is ad-hoc because each node is willing to forward data for other nodes, and so the determination of which nodes forward data is made dynamically based on the network connectivity. This is in contrast to older network technologies, which we shall see discuss in the next paragraph, in which some designated nodes, usually with custom hardware and variously known as routers, switches, hubs, and firewalls, perform the task of forwarding the data. Minimal configuration and quick deployment make ad hoc networks suitable for emergency situations like natural or human-induced disasters, military conflicts, emergency medical situations etc.

In infrastructure mode, a wireless network uses an access point, or base station. In this type of network the access point acts like a hub, providing connectivity for the wireless computers. It can connect (or "bridge") the wireless LAN to a wired LAN, allowing wireless computer access to LAN resources, such as file servers or existing Internet Connectivity.

In our project we have chosen to control and monitor our processes via infrastructure mode since its installation needs less configuration (whereas ad-hoc mode requires setting the SSID for each device) as well as providing more flexibility since an AP may easily allow us to connect to a wired network as well as easily identifying any new wireless host.

8.4 Wireless Standards

Wireless technologies conform to a variety of standards and offer varying levels of security features. The principal advantages of standards are to encourage mass production and to allow products from multiple vendors to interoperate. Each protocol is defined in IEEE standards. WLANs or Wi-Fi follow the IEEE 802.11 standards, Bluetooth follows IEEE 802.15.1 while Zigbee follows IEEE 802.15.4. Ad hoc networks follow proprietary techniques or are based on the Bluetooth standard, which was developed

by a consortium of commercial companies making up the Bluetooth Special Interest Group (SIG). These standards are described below.

The IEEE 802 standards typically create the specifications at the physical layer and portions of the data link layer. The higher layer protocols are left to the industry and the individual applications. Hence the standard and market names are not always interchangeable.

8.4.1 Wi-Fi

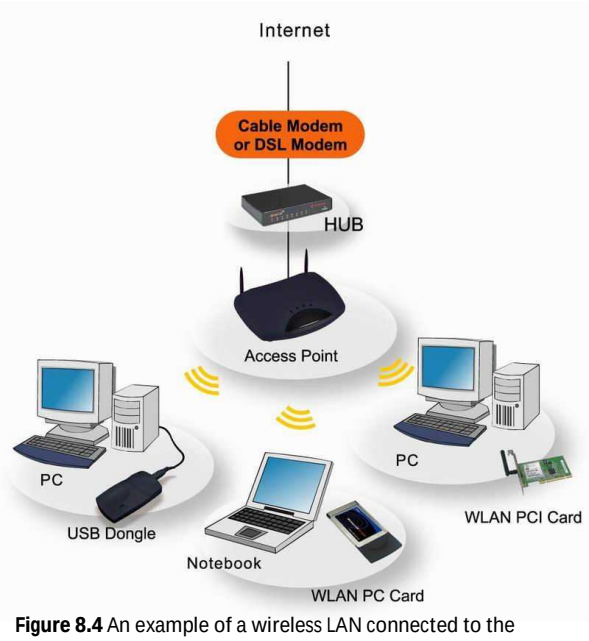


Figure 8.4 An example of a wireless LAN connected to the Internet.

Wi-Fi or WLANs are based on the IEEE 802.11 standard, which the IEEE first developed in 1998. The IEEE designed 802.11 to support medium-range, higher data rate applications, such as Ethernet networks, and to address mobile and portable stations.

802.11 is the original WLAN standard, designed for 1 Mbps to 2 Mbps wireless transmissions. It was followed in 1999 by 802.11a, which established a high-speed WLAN standard for the 5 GHz band and supported 54 Mbps. Also completed in 1999 was the 802.11b standard, which operates in the 2.4 - 2.48 GHz band and supports 11 Mbps. The 802.11b standard is currently the dominant standard for WLANs, providing sufficient speeds for most of today's applications. Because the 802.11b standard has been so widely adopted, the security weaknesses in the standard have been exposed. Another standard, 802.11g, still in draft, operates in the 2.4 GHz wave band, where current WLAN products based on the 802.11b standard operate.

Two other important and related standards for WLANs are 802.1X and 802.11i. The 802.1X, a port-level access control protocol, provides a security framework for IEEE networks, including Ethernet and wireless networks.

We have chosen this standard in our projects in order to be compatible with most devices found in the market as well as its ability to interface smoothly with wired Ethernet networks and the Internet.

Characteristic	Description
Physical Layer	Direct Sequence Spread Spectrum (DSSS), Frequency Hopping Spread Spectrum (FHSS), Orthogonal Frequency Division Multiplexing (OFDM), Infrared (IR).
Frequency Band	2.4 GHz (ISM band) and 5 GHz.
Data Rates	1 Mbps, 2 Mbps, 5.5 Mbps (11b), 11 Mbps (11b), 54 Mbps (11a)
Data and Network Security	RC4-based stream encryption algorithm for confidentiality, authentication and integrity. Limited key management. (AES is being considered for 802.11i.)
Operating Range	Up to 150 feet indoors and 1500 feet outdoors.
Positive Aspects	Ethernet speed without wires; many different products from many different companies. Wireless client cards and access point costs are decreasing.
Negative Aspects	Poor security in native mode; throughput decrease with distance and load.

Table 8.2 Key characteristics of 802.11 WLANs.

8.4.2 Bluetooth

Bluetooth has emerged as a very popular ad hoc network standard today. The Bluetooth standard is a computing and telecommunications industry specification that describes how mobile phones, computers, and PDAs should interconnect with each other, with home and business phones, and with computers using short-range wireless connections. The Bluetooth standard specifies wireless operation in the 2.45 GHz radio band and supports data rates up to 820 kbps. It further supports up to three simultaneous voice channels and employs frequency-hopping schemes and power reduction to reduce interference with other devices operating in the same frequency band. The IEEE 802.15 organization has derived a wireless personal area networking technology based on Bluetooth specifications v1.1.

Ad hoc networks today are based primarily on Bluetooth technology. Bluetooth is an open standard for short-range digital radio. It is touted as a low-cost, low-power, and low-profile technology that provides a mechanism for creating small wireless networks on an ad hoc basis. Bluetooth is considered a wireless PAN technology that offers fast and reliable transmission for both voice and data. Untethered Bluetooth devices will eliminate the need for cables and provide a bridge to existing networks.

Bluetooth can be used to connect almost any device to any other device. An example is the connection between a PDA and a mobile phone. The goal of Bluetooth is to connect disparate devices (PDAs , cell phones , printers, faxes, etc.) together wirelessly in a small environment such as an office or home. According to the leading proponents of the technology, Bluetooth is a standard that will ultimately—

- Eliminate wires and cables between both stationary and mobile devices
- Facilitate both data and voice communications
- Offer the possibility of ad hoc networks and deliver synchronicity between personal devices.

Bluetooth is designed to operate in the unlicensed ISM (industrial, scientific, medical applications) band that is available in most parts of the world, with variation in some locations. The characteristics of Bluetooth are summarized in Table 8.3. Bluetooth-enabled devices will automatically locate each other, but making connections with other devices and forming networks requires user action. As with all ad-hoc networks, Bluetooth network topologies are established on a temporary and random basis.

A distinguishing feature of Bluetooth networks is the master-slave relationship maintained between the network devices. Up to eight Bluetooth devices may be networked together in a master-slave relationship, called a **piconet**. In a piconet, one device is designated as the master of the network with up to seven slaves connected directly to that network. The master device controls and sets up the network (including defining the network's hopping scheme). Devices in a Bluetooth piconet operate on the same channel and follow the same frequency hopping sequence. Although only one device may perform as the master for each network, a slave in one network can act as the master for other networks, thus creating a chain of networks. This series of piconets, often referred to as **scatter-nets**, allows several devices to be inter-networked over an extended distance. This relationship also allows for a dynamic topology that may change during any given session: as a device moves toward or away from the master device in the network, the topology and therefore the relationships of the devices in the immediate network change.

Characteristic	Description
Physical Layer	Frequency Hopping Spread Spectrum (FHSS)
Frequency Band	2.4 – 2.4835 GHz (ISM band).
Hop Frequency	1600 hops/sec.
Data Rates	1 Mbps (raw). Higher bit rates are anticipated.
Data and Network Security	Three modes of security (none, link-level and service-level), two levels of device trust, and three levels of service security. Stream encryption for confidentiality, challenge response for authentication. PIN-derived keys and limited management.
Operating Range	About 10 meters (30 feet); can be extended to 100 meters.
Throughput	Up to approximately 820 kbps.
Positive Aspects	No wires and cables for many interfaces. Ability to penetrate walls and other obstacles. Costs are decreasing with a $5 cost projected. Low power and minimal hardware.
Negative Aspects	Possibility for interference with other ISM band technologies. Relatively low data rates. Signal leak outside desired boundaries.

Table 8.3 Key characteristics of Bluetooth.

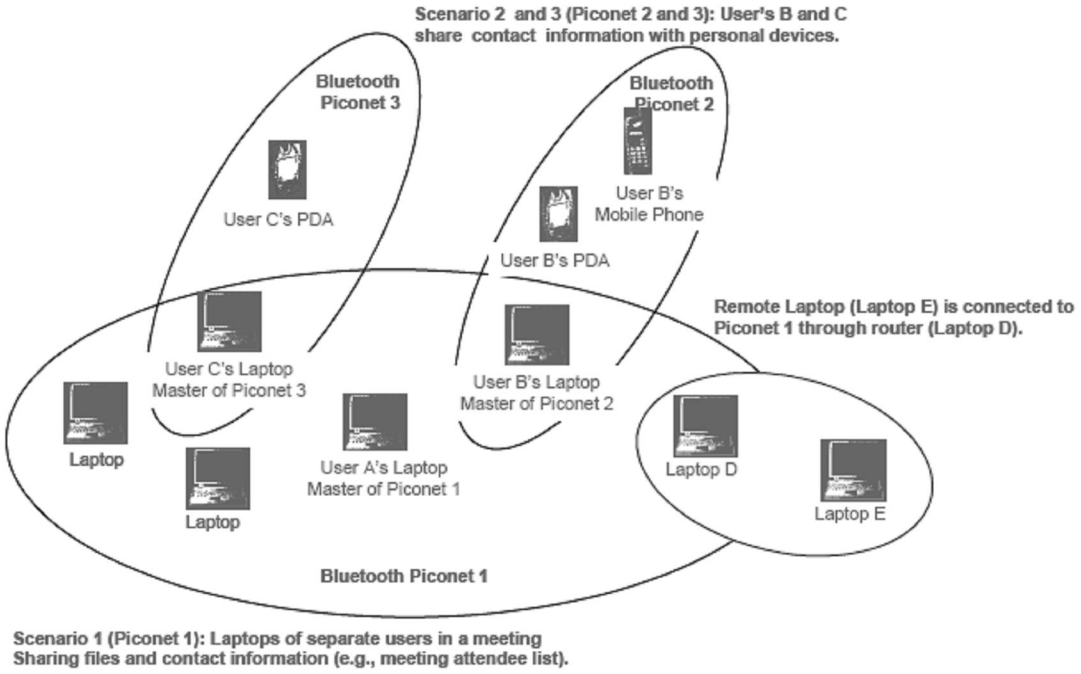


Figure 8.5 Typical Bluetooth Network—A Scatter-net.

Mobile routers in a Bluetooth network control the changing network topologies of these networks. The routers also control the flow of data between devices that are capable of supporting a direct link to each other. As devices move about in a random fashion, these networks must be reconfigured on the fly to handle the dynamic topology. The routing protocols it employs allow Bluetooth to establish and maintain these shifting networks.

Bluetooth transceivers operate in the 2.4 GHz, ISM band, which is similar to the band WLAN devices and other IEEE 802.11 compliant devices occupy. Bluetooth transceivers, which use Gaussian Frequency Shift Keying (GFSK) modulation, employ a frequency hopping (FH) spread spectrum system with a hopping pattern of 1,600 times per second over 89 frequencies in a quasi-random fashion. The theoretical maximum bandwidth of a Bluetooth network is 1 Mbps. However, in reality the networks cannot support such data rates because of communication overhead. The second generation of Bluetooth technology is expected to provide a maximum bandwidth of 2 Mbps.

Bluetooth networks can support either one asynchronous data channel with up to three simultaneous synchronous speech channels or one channel that transfers asynchronous data and synchronous speech simultaneously. Bluetooth uses a combination of packet-switching technology and circuit-switching technology. The advantage of using packet switching in Bluetooth is that it allows devices to route multiple packets of information by the same data path. Since this method does not consume all the resources on a data path, it becomes easier for remote devices to maintain data flow throughout a scatter-net.

Clearly, Bluetooth is not suitable for industrial due to its short-distance range and support for limited number of nodes.

8.4.3 ZigBee

ZigBee is built on top of the IEEE 802.15.4 standard. ZigBee provides routing and multi-hop functions to communication. **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). ZigBee is targeted at RF applications that require a low data rate, long battery life, and secure networking.

ZigBee Key Features:

- **Low Power**

  Low power is one of the key aspects of ZigBee. 802.15.4 was chosen due to it's low power requirements. 802.15.4 radios have the capability to come from an off state to be on the network and transmitting in less then 30ms. This allows the radio to be turned off and powered on only when needed, providing the lowest possible system power. Compare this too many radios that need to remain synchronized with the network or take up to several seconds to connect and transmit from an off state. This delay would be unacceptable for applications such as lighting were requirements are less then 100ms. In addition, during the time the radios are staring up, they are consuming extra power. When combined with Freescale's low power MCUs, 802.15.4 and ZigBee have the potential to last as long as the shelf life of most batteries.

- **Robust**

802.15.4 provides a robust foundation for ZigBee, ensuring a reliable solution in noisy environments. Features such as energy detection, clear channel assessment and channel selection help the device pick the best possible channel, avoiding other wireless networks such as Wi-Fi®. Message acknowledgement helps to ensure that the data was delivered to its destination. Finally, multiple levels of security ensure that the network and data remain intact and secure.

- **Mesh Networking**

The ability to cover large areas with routers is one of the key features of ZigBee that helps differentiate itself from other technologies. Mesh networking can extend the range of the network through routing, while self healing increases the reliability of the network by re-routing a message in case of a node failure

- **Interoperability**

The ZigBee Alliance helps ensure interoperability between vendors by creating testing and certification programs for ZigBee devices. Users can be assured the devices that go through

certification testing and use the ZigBee logo will work with other devices based on the same applications. This provides end customers with the customers with peace of mind while creating brand awareness of products with the ZigBee logo.

Freescale offers a comprehensive ZigBee solution, including RF chipsets, MCUs, sensors, reference designs, protocol stack software, and development tools. For more information on our ZigBee Family offering see www.freescale.com/zigbee.

However, since we wanted to provide both Ethernet and wireless accessibility, we chose to use Wi-Fi rather than ZigBee since the difference in bit rates between Ethernet and ZigBee is greater than that between Ethernet and Wi-Fi. Using ZigBee might have caused problems in transmissions causing less throughput and probably buffer problems.

The relationship between IEEE 802.15.4-2003 and ZigBee is similar to that between IEEE 802.11 and the Wi-Fi Alliance. The ZigBee 1.0 specification was ratified on December 14, 2004 and is available to members of the ZigBee Alliance. An entry level membership, called Adopter, in the ZigBee Alliance costs US$ 3500 annually and provides access to the specifications and permission to create products for market using the specifications. For non-commercial purposes, the ZigBee specification is available to the general public at the ZigBee Specification Download Request. Most recently, the ZigBee 2006 specification was posted in December 2006.

ZigBee operates in the industrial, scientific and medical (ISM) radio bands; 868 MHz in Europe, 915 MHz in the USA and 2.4 GHz in most jurisdictions worldwide. The technology is intended to be simpler and cheaper than other WPANs such as Bluetooth. The most capable ZigBee node type is said to require only about 10% of the software of a typical Bluetooth or Wireless Internet node, while the simplest nodes are about 2%. However, actual code sizes are much higher, closer to 50% of Bluetooth code size. ZigBee chip vendors have announced 128 kilobyte devices

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, domotics, 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.

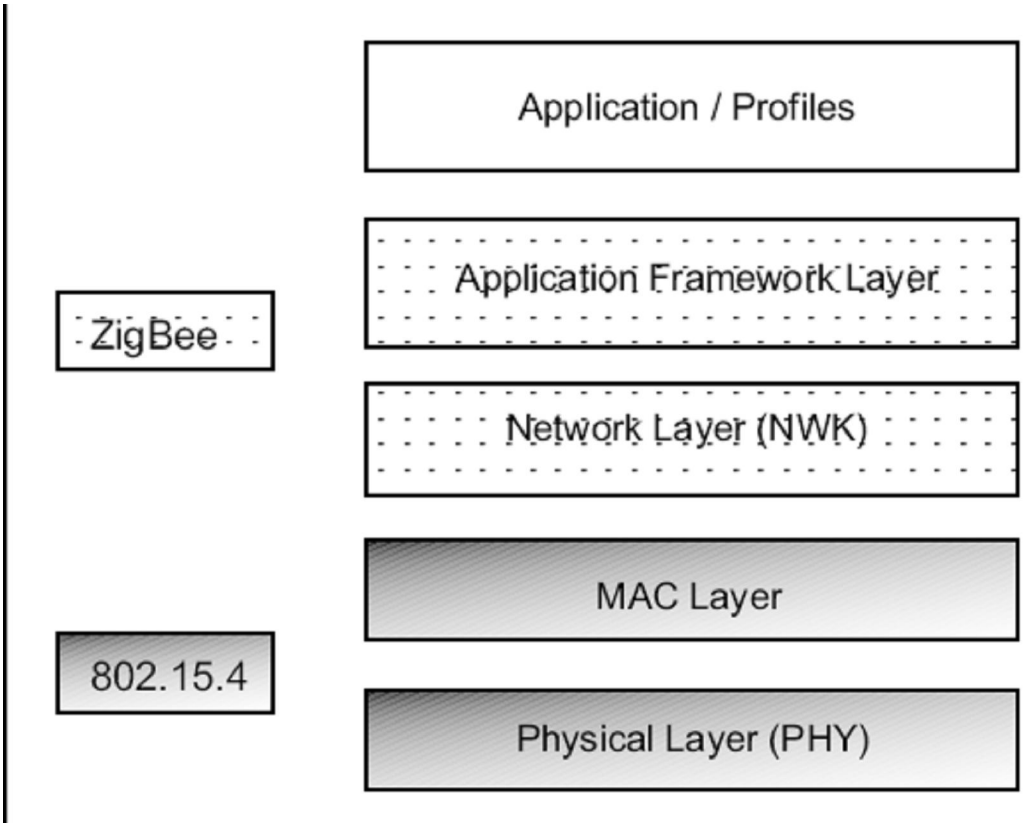


Figure 8.6 ZigBee Stack

8.5 UDP or TCP Over Wireless

During our work, we have discovered that the performance of the UDP protocol over wireless LAN is much lower than that of TCP. Therefore, despite UDP's advantages of having lower overhead, faster response and the need for less complex code (due to its connectionless nature and the absence of handshaking), we had to use TCP protocol in our project.

Tests done by Center for Wireless Communications & Computer Systems Laboratory, Department of Computer Science & Engineering at the University of California, San Diego have agreed with our observation. In their tests, they added an option for UDP tests that uses packet sequence numbers so that the receiver can detect and report packet losses (as they occur and in total), and named this version ettcp.

Their results noticed that in *all* UDP tests with 100 byte packets, 90-95% of packets sent by `ettcp` were never transmitted, and actually did not even reach the interface according to the driver. The reason was buffer shortages at the UDP level, due to the very fast generation rate of short packets, which caused datagrams to be dropped. When running the same tests over the faster wired interfaces for comparison purposes, fewer packets (50%) were dropped as expected. TCP tests with 100 byte packets did not suffer from such drops, because TCP uses window based flow control, with a maximum window of 32 KB in the tests. This prohibits the sending process from passing to the network code huge bursts of data without pause. Even

though `ettcp` used a TCP socket option to transmit data segments immediately, they occasionally saw packets larger than expected, except in the 1500 byte packet tests where the maximum WLAN packet size was reached. The reason is that TCP keeps track of its transmission queue in terms of bytes rather than packets, thus any segments whose immediate transmission is deferred may be later combined into larger packets. Such delays can be caused by MAC layer contention due to the bidirectional traffic of TCP.

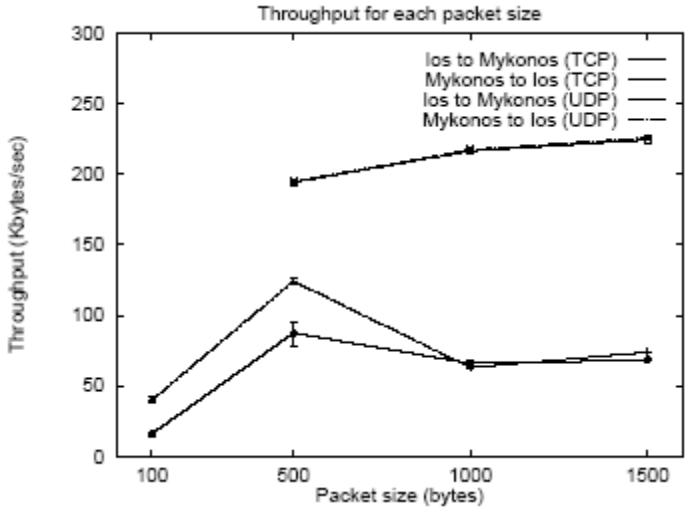


Figure 8.7 Part of the results of the tests done by Center for Wireless Communications & Computer Systems Laboratory at the University of California, showing that the throughput of TCP was higher than that of UDP.

8.6 Roaming

A wireless computer can "roam" from one access point to another, with the software and hardware maintaining a steady network connection by monitoring the signal strength from in-range access points and locking on to the one with the best quality. Usually this is completely transparent to the user; they are not aware that a different access point is being used from area to area. Some access point configurations require security authentication when swapping access points, usually in the form of a password dialog box. Access points are required to have overlapping wireless areas to achieve this as can be seen in the following diagram:

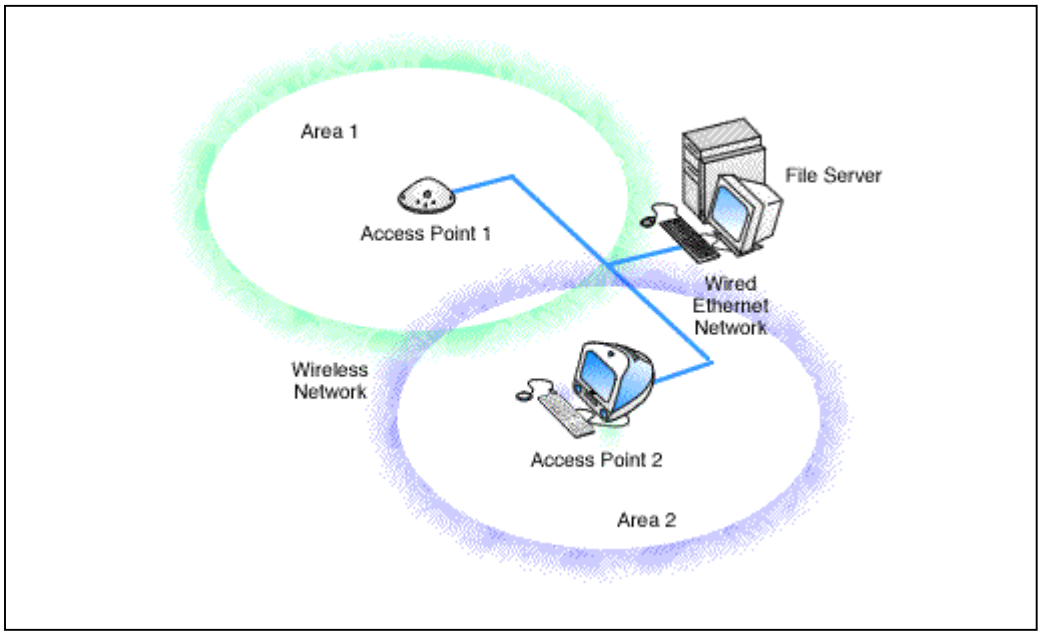


Figure 8.8 Roaming

A user can move from Area 1 to Area 2 transparently. The Wireless networking hardware automatically swaps to the Access Point with the best signal. This handshaking is done by the operating system of the wireless host.

8.7 Ethernet to Wireless Conversion

The microcontroller which we have used, HCS12NE64, provides us with Ethernet connectivity not wireless connectivity. In order to achieve wireless connectivity, we had to find a bridge to convert the Ethernet protocol to the wireless Wi-Fi protocol. This conversion only affects the physical and data-link layer, therefore, the other functions in the upper layers will not be changed and therefore we have achieved transparency between the devices.

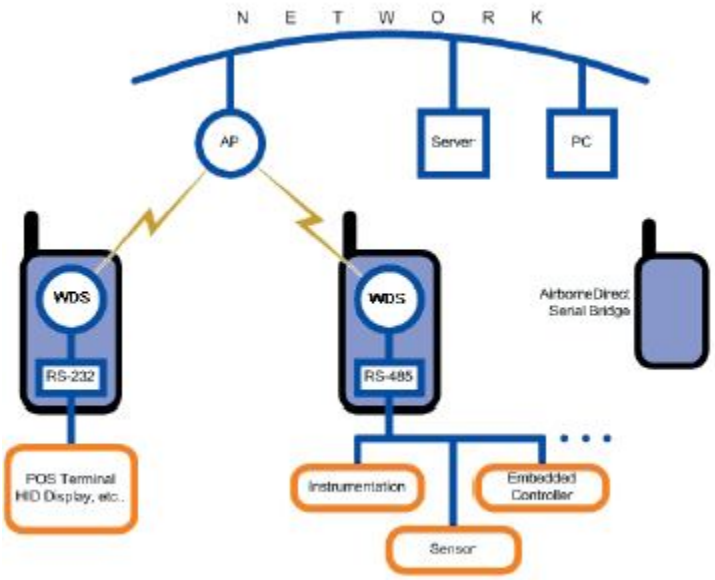


Figure 8.9 Ethernet-to-Wireless Bridge

After a long search, we chose the Airborne wireless-to-Ethernet bridge. AirborneDirect is a family of fully integrated 802.11 wireless LAN device server products designed to provide wireless LAN and Internet connectivity to transportation, medical, warehouse logistics, POS, industrial, military and scientific applications.

The highly integrated hardware and software enables plug-and-play capability. This significantly reduces the complexity of wireless system deployment and network connectivity. Integrating AirborneDirect with existing OEM platforms can significantly enhance the product's value and functionality giving the OEM a competitive advantage.

Figure 8.10 Airborne Ethernet-to-Wireless Bridge used in our project.

AirborneDirect™ is a fully integrated, 802.11 wireless Local Area Network (LAN) connectivity device designed to provide wireless LAN and Internet connectivity in industrial, scientific,

medical, and transportation applications where an existing communications interface already exists. The AirborneDirect™ Ethernet Bridge is well suited to the following applications:

- Point-of-sale devices.
- Medical equipment.
- Manufacturing machinery.
- Bar-code readers.
- Time clocks.
- Scales.
- Data-collection devices.
- Vehicle Diagnostics.
- Telematics.

The AirborneDirect™ Ethernet Bridge provides true plug-and-play wireless connectivity. By delivering convenient, easy-to-deploy wireless network connectivity, the Bridge significantly reduces the complexities of wireless system deployment and network implementation. At the same time, users can move equipment without the cost and time associated with wired network drops and environment restrictions. This provides flexibility for seasonal demands, line and staffing changes, and more.

The AirborneDirect™ Ethernet Bridge provides a bridge between the 802.11 wireless LAN and any Ethernet-ready device with an RJ-45 connector. It acts transparently between the device and a wireless LAN. By integrating AirborneDirect™ into existing and legacy platforms, OEMs can significantly enhance their products by delivering increased value and functionality to their entire customer base.

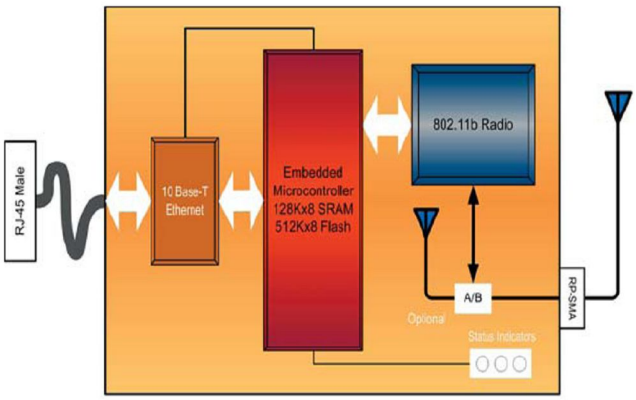


Figure 8.11 Block diagram of Airborne Ethernet-to-Wireless Bridge.

Serial Interface	RS-232, RS-422 or RS-485 (2 wire)
Wireless Network Interface	IEEE 802.11b DSSS, WiFi Compliant
BAUD Rates	Up to 230 Kbps
Frequency	2.4 ~ 2.4835 GHz (US, Europe, Canada, Japan) 2.471 ~ 2.497 GHz (Japan)
Channels	11 US/Canada 13 Europe 14 Japan 4 France
Wireless Raw Data Rates	11Mbps, 5.5Mbps, 2Mbps, 1Mbps
RF Power	+18dBi (typical)
Security	WEP (64 & 128 bit), WPA (PSK & TKIP), WPA with LEAP
Antenna	Integrated Omni-directional RP-SMA +3dBi antenna
Protocols Data Transfer Protocols	TCP/IP, ARP, ICMP, DHCP, DNS, HTTP, UDAP Discovery, UDP TCP/IP, HTTP
Status Indicators	Power, Link, Comm
Power Input	110/240VAC to 5VDC external power supply (wall wart)
Power Consumption	2.5W max (AC Adapter)
Power Supply Connector	2.1mm Barrel Jack
Software Configuration	Web page-based configuration interface, Command Line Interface
Web Server	Built-in Web Server capable of serving dynamic web pages with embedded JavaScript
Management	Device Discovery, Configuration and Remote Firmware Upgrade
OS Compatibility	Airborne Evaluation Utility - Win9x/ME/NT/2000/XP Airborne VCOM - Win2000/XP
Agency Approvals	U.S. = FCC Part 15 Class B, C/UL, CE Europe = CE Canada = RSS-210
Operating Temperature	-40°C to +85°C
Storage Temperature	-40°C to +125°C
Dimensions (without mounting bracket)	99.78mm L x 60.5mm W x 36.1mm T (3.93 in. x 2.38 in. x 1.42 in.)
Enclosure	Nylon - Gray

Table 8.4 Specifications of the Airborne bridge.

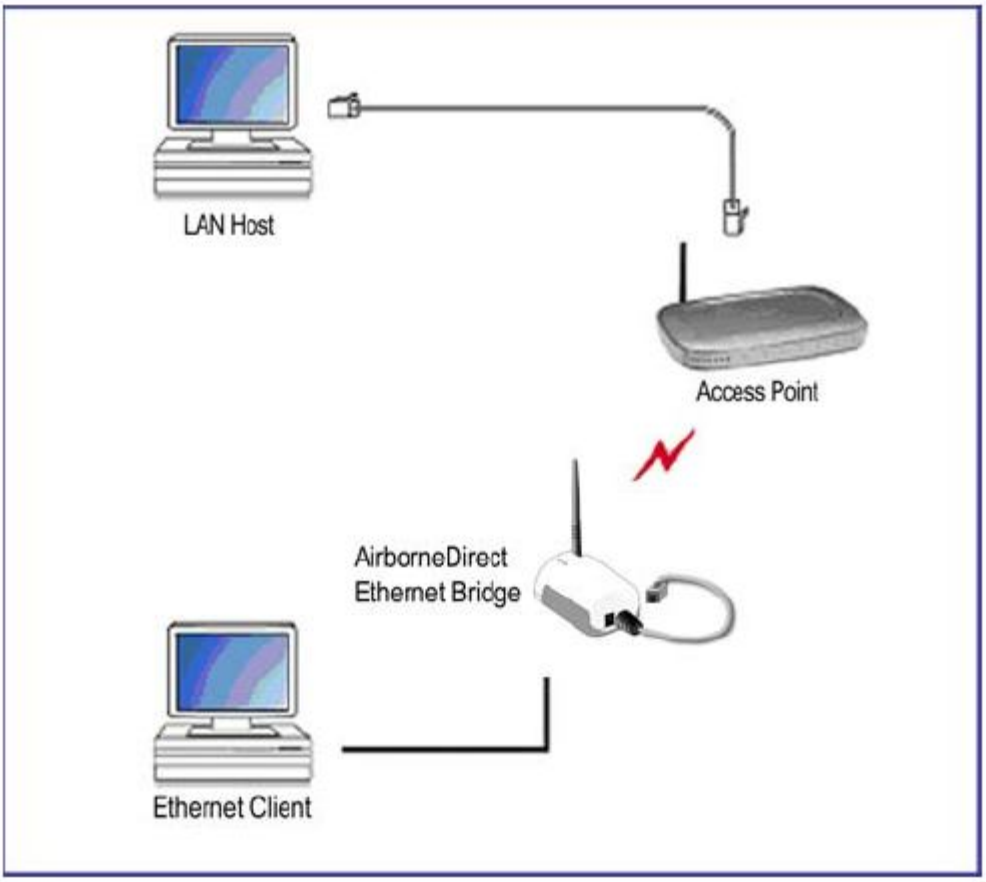


Figure 8.12 Basic Application Involving a LAN Host and Ethernet Client

Referring to the above figure, a LAN host connects to an Ethernet client by setting the destination IP of the Ethernet client rather than the IP of the bridge itself. The IP of the bridge itself is only used to configure it.

This bridge gave us two choices of operations: ad-hoc and infrastructure (where we can only connect to it via a wireless access point). We had chosen infrastructure mode due to the reasons mentioned before.

Chapter 9:
Wireless Security

9.1 Overview

Wireless communications obviously provide potential security issues, as an intruder does not need physical access to the traditional wired network in order to gain access to data communications. However, 802.11 wireless communications cannot be received --much less decoded-- by simple scanners, short wave receivers etc. This has led to the common misconception that wireless communications cannot be eavesdropped at all. However, eavesdropping is possible using specialist equipment.

To protect against any potential security issues, 802.11 wireless communications have a function called WEP (Wired Equivalent Privacy), a form of encryption which provides privacy comparable to that of a traditional wired network. If the wireless network has information that should be secure then WEP should be used, ensuring the data is protected at traditional wired network levels.

9.2 Security of 802.11 Wireless LANs

This section discusses the built-in security features of 802.11. It provides an overview of the inherent security features to better illustrate its limitations and provide a motivation for some of the recommendations for enhanced security. The IEEE 802.11 specification identified several services to provide a secure operating environment. The security services are provided largely by the Wired Equivalent Privacy (WEP) protocol to protect link-level data during wireless transmission between clients and access points. WEP does not provide end-to-end security, but only for the wireless portion of the connection as shown in Figure 9.1.

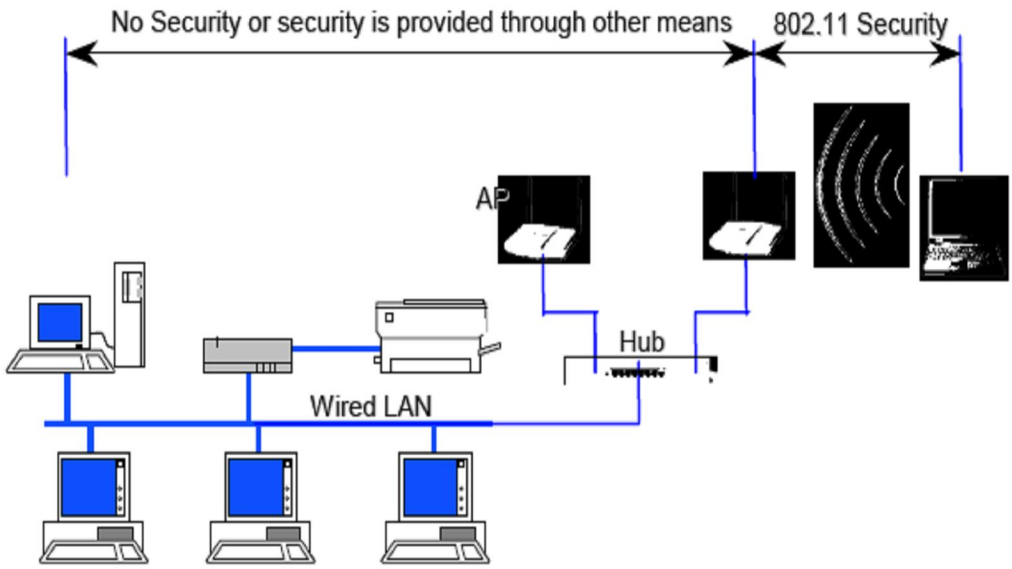


Figure 9.1 Wireless Security of 802.11 in Typical Network

Encryption is the process of transforming information to make it unreadable to anyone except those possessing special knowledge, usually referred to as a Key. The result of the process is **encrypted** information In many contexts, the word **encryption** also implicitly refers to the reverse process, **decryption** , to make the encrypted information readable again (i.e. to make it unencrypted).

Encryption has long been used by militaries and governments to facilitate secret communication. Encryption is now used in protecting information within many kinds of civilian systems, such as computers and networks, mobile telephones, and bank automatic teller machines. Encryption is also used in digital rights management to restrict the use of copyrighted material and in software copy protection to protect against reverse engineering and software piracy.

Encryption, by itself, can protect the confidentiality of messages, but other techniques are still needed to verify the integrity and authenticity of a message; for example, a message authentication code (MAC) or digital signatures. Standards and cryptographic software and hardware to perform encryption are widely available, but successfully using encryption to ensure security is a challenging problem. A single slip-up in system design or execution can allow successful attacks. Sometimes an adversary can obtain unencrypted information without directly undoing the encryption.

9.2.1 Wired Equivalent Privacy (WEP)

Wired Equivalent Privacy (**WEP**) is a security protocol, specified in the IEEE Wireless Fidelity (WI-Fi) standard, 802.11b, that is designed to provide a wireless local area network (WLAN) with a level of security and privacy comparable to what is usually expected of a wired LAN. A wired local area network (LAN) is generally protected by physical security mechanisms

(controlled access to a building, for example) that are effective for a controlled physical environment, but may be ineffective for WLANs because radio waves are not necessarily bound by the walls containing the network. WEP seeks to establish similar protection to that offered by the wired network's physical security measures by encrypting data transmitted over the WLAN. Data encryption protects the vulnerable wireless link between clients and access points; once this measure has been taken, other typical LAN security mechanisms such as password protection, end-to-end encryption, virtual private networks (VPNs), and authentication can be put in place to ensure privacy.

How does it work?

WEP uses secret keys to encrypt data. Both AP and the receiving stations must know the secret keys.

There are two kinds of WEP with keys of either 64bits or 128bits. The longer key gives a slightly higher level of security (but not as much as the larger number would imply). In fact the user keys are 40bits and 104bits long, the other 24bits in each case being taken up by a variable called the Initialization Vector (IV).

When a packet is to be sent it is encrypted using a combination of the IV and the secret key. The IV is different (in theory) for each packet, while the secret key is fixed. The resulting packet data looks like random data and therefore makes the original message unreadable to an outsider not knowing the key. The receiving station reverses the encryption process to retrieve the message in clear text.

What is wrong with WEP?

- **Values can be reused**

  In fact the standard does not specify that the value needs to change at all. Reusing keys is a major cryptographic weakness in any security system.

- **Length is too short**

  24 bit keys allow for around 16.7 million possibilities. Sounds a lot, but on a busy network this number can be achieved in a few hours. Reuse is then unavoidable.

  Some manufacturers use 'random' keys. This is not the best way to ensure against reuse. A better solution is to start with a key and increment by one for each subsequent key. Unfortunately many devices revert to the same value at start up and then follow the same sequence providing lots of duplicate values for hackers to work on.

- **Weak keys are susceptible to attack**

  Certain keys value combinations, 'Weak IVs', do not produce sufficiently random data for the first few bytes. This is the basis of the highly publicized attacks on WEP and the reason that keys can be discovered.

Manufacturers often deliberately disallow Weak IV values. This is good in that it reduces the chances of a hacker capturing weak keys, but also has the effect of reducing the already limited key possibilities further, increasing the chance of reuse of keys.

- **Master keys are used directly**

  From a cryptographic point of view using master keys directly is not at all recommended. Master keys should only be used to generate other temporary keys. WEP is seriously flawed in this respect.

- **Key Management and updating is poorly provided for**

  Administration of WEP keys is not well designed and difficult to do on large networks. Users tend to change keys very infrequently which gives a potential hacker lots of time to collect enough packets to launch an attack.

- **Message integrity checking is ineffective**

  WEP does have a message integrity check but hackers can change messages and recompute a new value to match. This makes the checking ineffective against tampering.

9.2. 2 Wi-Fi Protected Access (WPA)

WPA is an encryption algorithm that takes care of a lot of the vunerablities inherent in WEP. WEP is, by design, flawed. No matter how good or crappy, long or short, your WEP key is, it can be cracked. WPA is different. A WPA key *can* be made good enough to make cracking it unfeasible. WPA is also a little more cracker friendly. By capturing the right type of packets, you can do your cracking offline. This means you only have to be near the AP for a matter of seconds to get what you need. Advantages and disadvantages.

WPA basically comes in two flavours RADIUS or PSK. PSK is crackable, RADIUS is not so much.

PSK uses a user defined password to initialize the TKIP, temporal key integrity protocol. There is a password and the user is involved, for the most part that means it is flawed. The TKIP is not really crackable as it is a per-packet key but upon the initialization of the TKIP, like during an authentication, we get the password (well the PMK anyways). A robust dictionary attack will take care of a lot of consumer passwords.

Radius involves physical transferring of the key and encrypted channels blah blah blah, look it up to learn more about it but 90% of commerical APs do not support it, it is more of an enterprise solution then a consumer one.

WPA Key Management

Rekeying of unicast encryption keys is optional with 802.1x. Additionally, 802.11 and 802.1x provide no mechanism to change the global encryption key that is used for multicast and broadcast traffic. With WPA, rekeying of both unicast and global encryption keys is required. The Temporal Key Integrity Protocol (TKIP) changes the unicast encryption key for every frame and each change is synchronized between the wireless client and the wireless AP. For the global encryption key, WPA includes a facility for the wireless AP to advertise changes to the connected wireless clients.

Temporal Key Integrity Protocol (TKIP)

WEP encryption is optional for 802.11. For WPA, encryption using TKIP is required. TKIP replaces WEP with a new encryption algorithm that is stronger than the WEP algorithm, yet can be performed using the calculation facilities present on existing wireless hardware. TKIP also provides for:

- The verification of the security configuration after the encryption keys are determined.
- The synchronized changing of the unicast encryption key for each frame.
- The determination of a unique starting unicast encryption key for each pre-shared key authentication.

Michael

With 802.11 and WEP, data integrity is provided by a 32-bit ICV that is appended to the 802.11 payload and encrypted with WEP. AIthough the ICV is encrypted, it is possible through cryptanalysis to change bits in the encrypted payload and update the encrypted ICV without being detected by the receiver.

With WPA, a method known as Michael specifies a new algorithm that calculates an 8-byte message integrity code (MIC) with the calculation facilities available on existing wireless hardware. The MIC is placed between the data portion of the 802.11 frame and the 4-byte ICV. The MIC field is encrypted with the frame data and the ICV.

Michael also provides replay protection. A new frame counter in the 802.11 frame is used to prevent replay attacks.

Advanced Encryption Standard (AES) Support

WPA defines the use of AES as an additional optional replacement for WEP encryption. Because adding AES support through a firmware update might not be possible for existing wireless equipment, support for AES on wireless network adapters and wireless APs is not required.

Supporting a Mixture of WPA and WEP Wireless Clients

To support the gradual transition of a WEP-based wireless network to WPA, it is possible for a wireless AP to support both WEP and WPA clients simultaneously. During the association, the wireless AP determines which clients are using WEP and which are using WPA. The disadvantage

to supporting a mixture of WEP and WPA clients is that the global encryption key is not dynamic. All other security enhancements for WPA clients are preserved.

9.3 Authentication

The IEEE 802.11 specification defines two means to “validate” wireless users attempting to gain access toa wired network: open-system authentication and shared-key authentication. One means, shared-keyauthentication, is based on cryptography, and the other is not. The open-system authentication techniqueis not truly authentication; the access point accepts the mobile station without verifying the identity of thestation. It should be noted also that the authentication is only one-way: only the mobile station isauthenticated. The mobile station must trust that it is communicating to a real AP. A taxonomy of thetechniques for 802.11 is depicted in Figure 9.2.

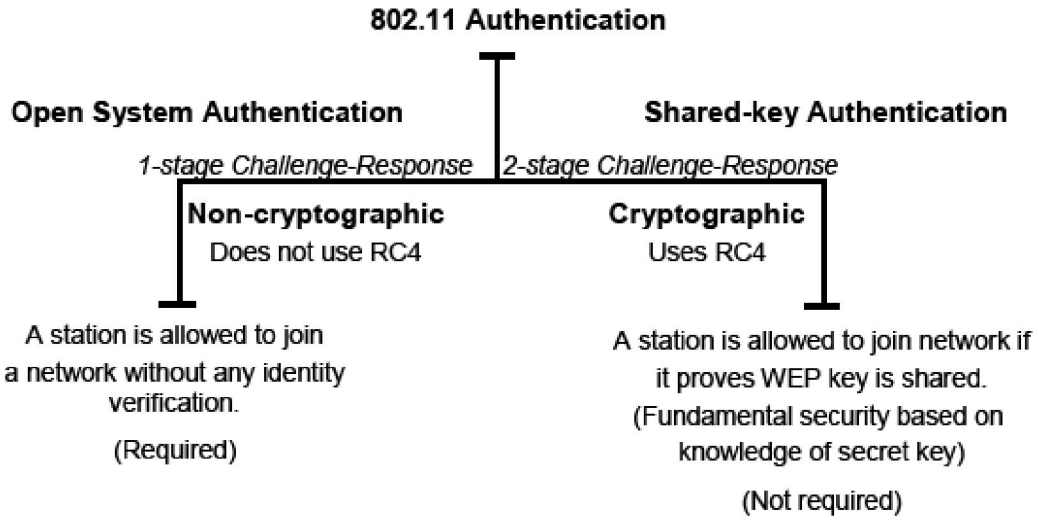


Figure 9.2 Taxonomy of 802.11 Authentication Techniques

With Open System authentication, a client is authenticated if it simply responds with a MAC addressduring the two-message exchange with an access point. During the exchange, the client is not trulyvalidated but simply responds with the correct fields in the message exchange. Obviously, with out cryptographic validatedation, open-system authentication is highly vulnerable to attack and practically invites unauthorized access. Open-system authentication is the only required form of authentication by the802.11 specification.

Shared key authentication is a cryptographic technique for authentication. It is a simple “challenge response ”scheme based on whether a client has knowledge of a shared secret. In this scheme, as depicted conceptually in Figure 2-5, a random challenge is generated by the access point and sent to the wireless client. The client, using a cryptographic key that is shared with the AP, encrypts the challenge (or“ nonce,” as it is called in security vernacular) and returns the result to the AP. The AP decrypts the result computed by the client and allows access only if

the decrypted value is the same as the random challenge transmitted. The algorithm used in the cryptographic computation and for the generation of the 128-bit

challenge text is the RC4 stream cipher developed by Ron Rivest of MIT. It should be noted that the authentication method just described is a rudimentary cryptographic technique, and it does not provide mutual authentication. That is, the client does not authenticate the AP, and therefore there is no assurance that a client is communicating with a legitimate AP and wireless network. It is also worth noting that simple unilateral challenge-response schemes have long been known to be weak. They suffer from numerous attacks including the infamous "man-in-the-middle" attack. Lastly, the IEEE 802.11specification does not require shared-key authentication.

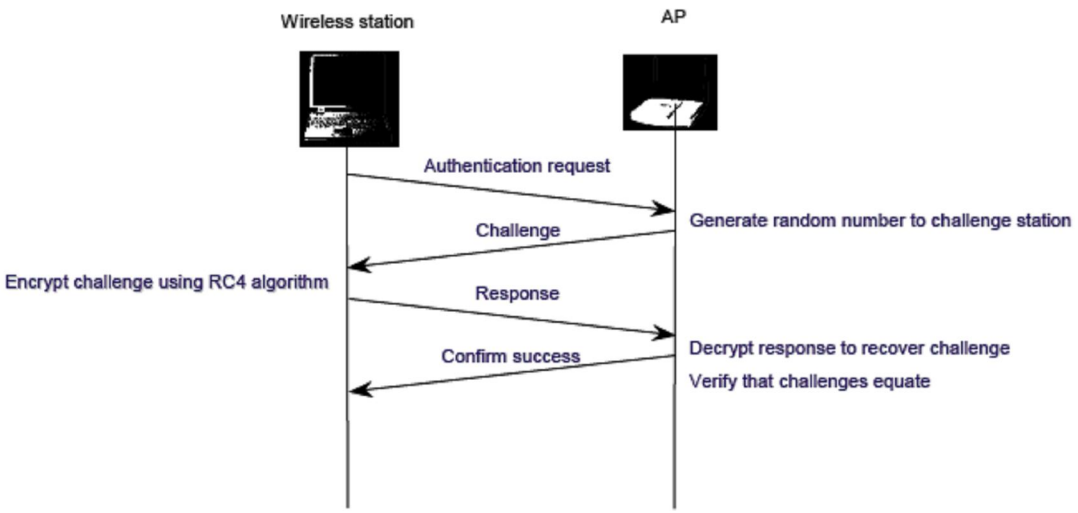
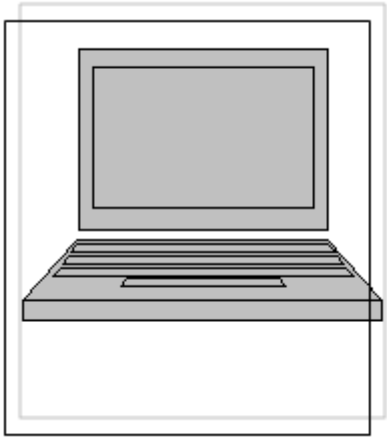


Figure 9.3 Shared-key Authentication Message Flow

4

Software_Application_____



Chapter 10: Process Control Studio

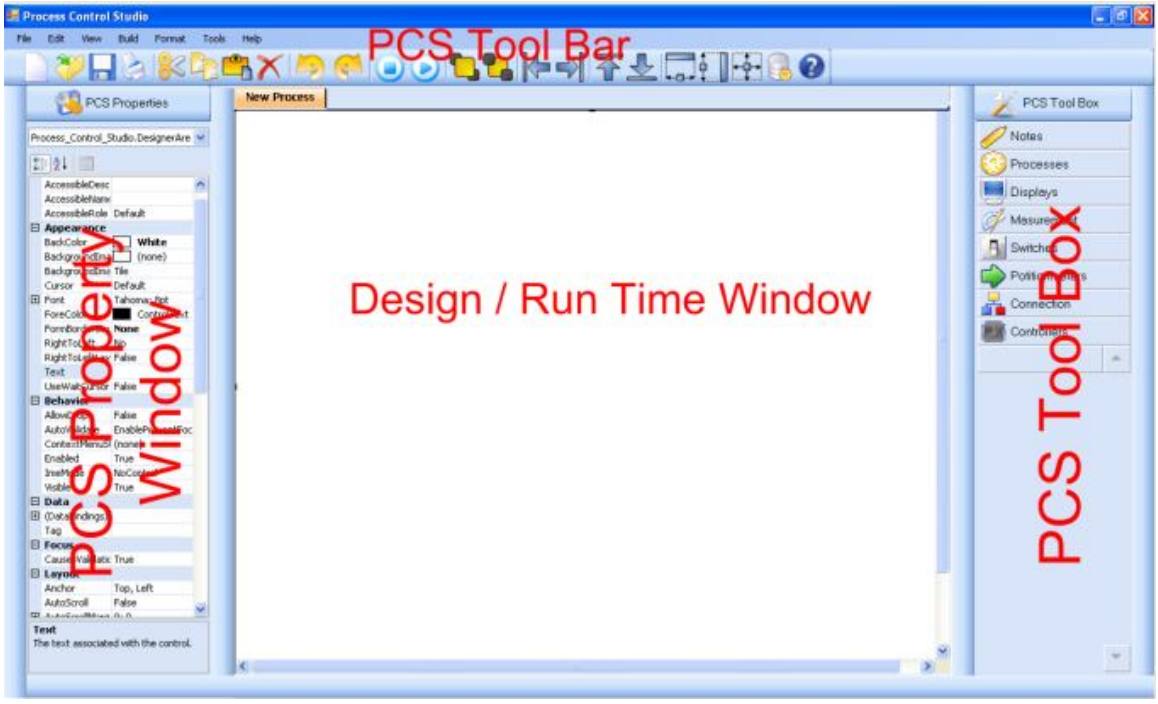
Chapter 11: Process Control Studio Mobile Edition

In our project, we have implemented 2 types of software: one for the PC or laptop and the other for the PDA or PocketPC. We have called the former Process Control Studio and the latter Process Control Studio Mobile Edition. The software should enable an engineer to design a pictorial model of a process and monitor its performance as well as provide control actions.

Chapter 10:
Process Control Studio

10.1 Getting Started with Process Control Studio

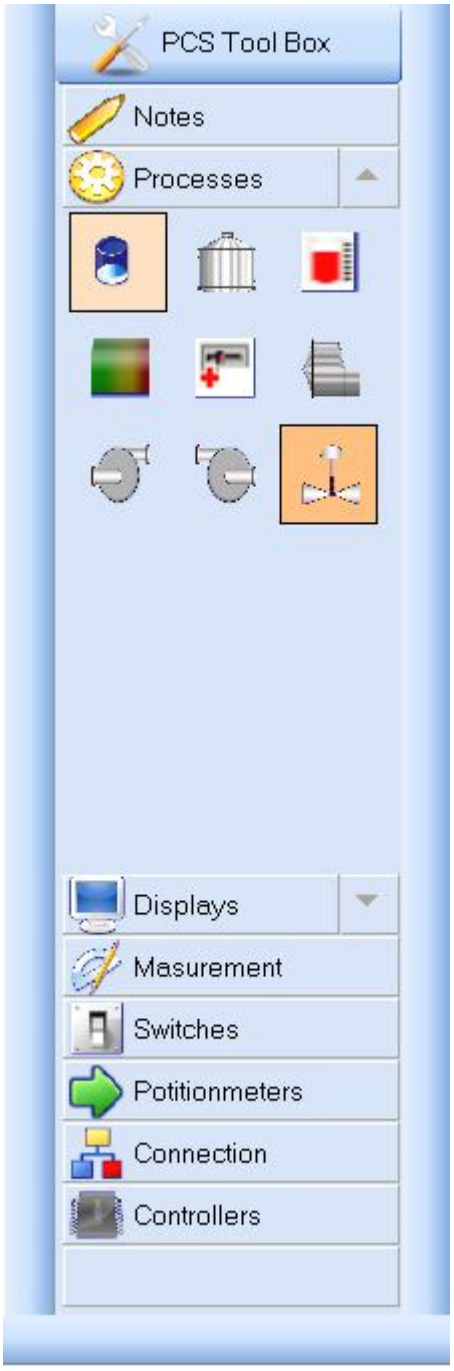
Process Control Studio is designed to achieve the principle of **WYSIWYG** (**What You See is What You Get**).

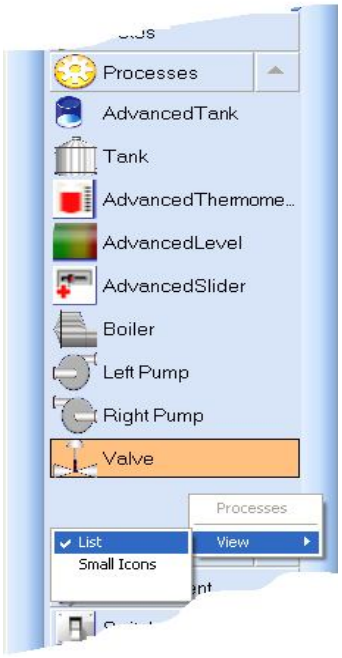


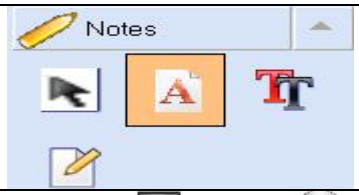
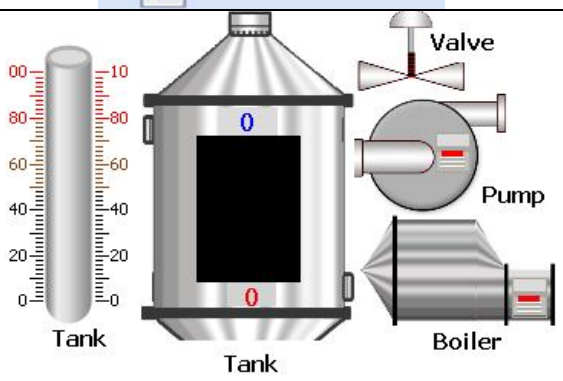
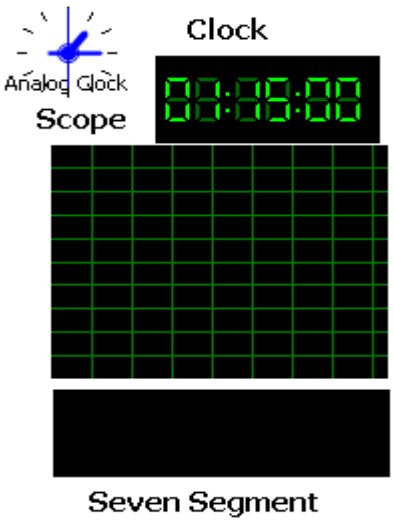
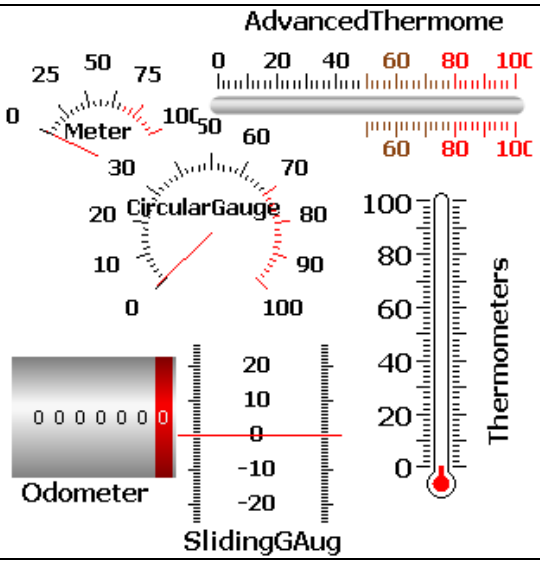
Process Control Studio is Consists of:

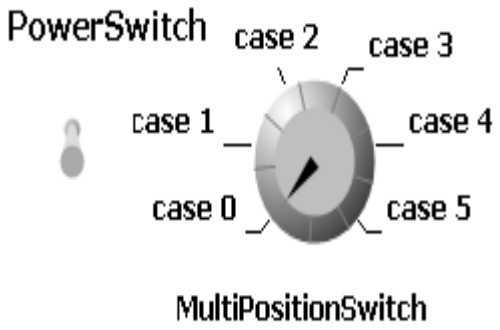
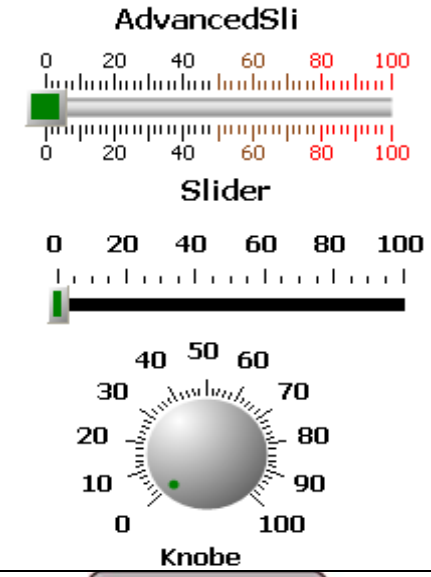
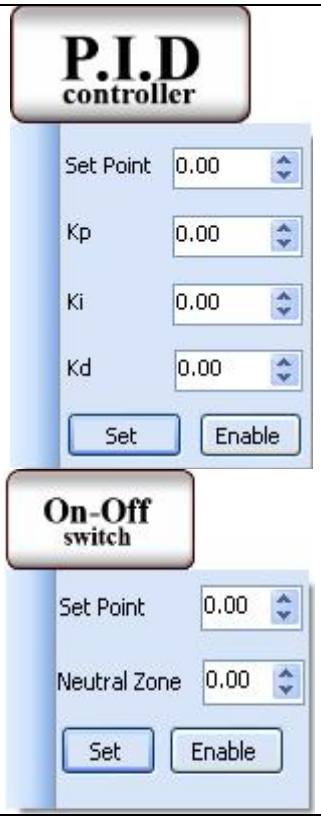
10.1.1 ToolBox

- Consists of number of tabs; each tab consists of number of components.
- It includes all components used in designing the process.
- Expands the members of a group by clicking the name of the group.
- There is a tool tip that appears when mouse moves over every component to tell you the name of this component.
- Control engineer can "drag and drop" controls onto a design form
- Control engineer can change tool box icon by right click inside it and chose **View** > **List.**



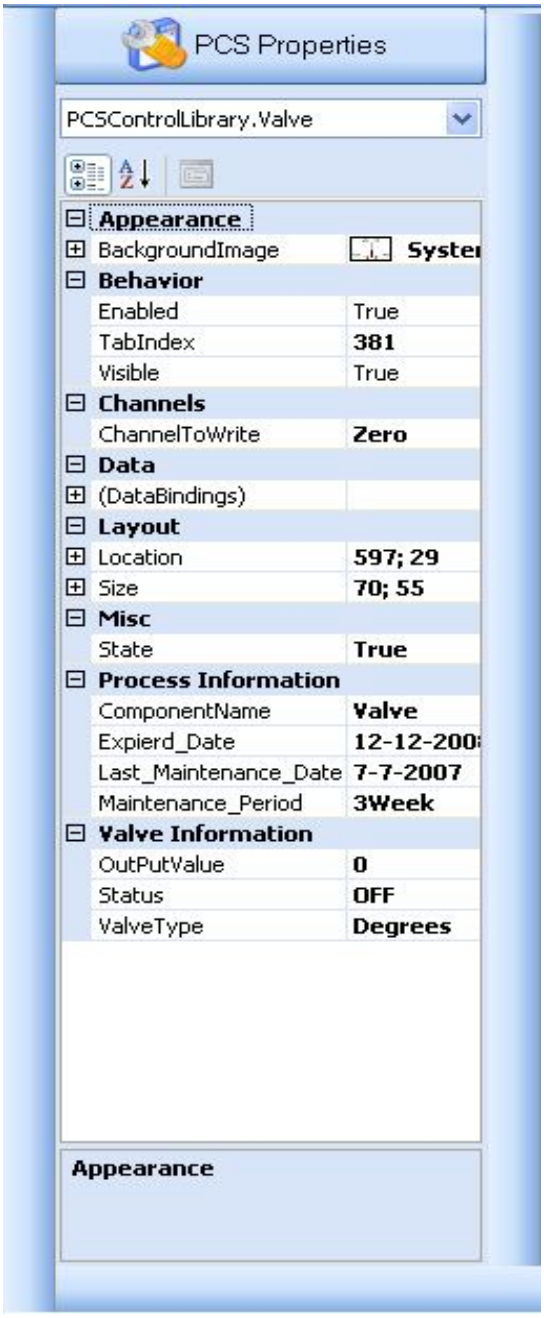


a) Notes Tab • It helps the user to add notes, and labels on the design area	
b) Process Tab • This group is considered as the backbone of any industrial process. • Control should be applied mostly on the components of this group	
c) Displays Tab • It is used mainly for displaying output data or response of the process to be observable by the operator. • Data displayed in the component of this type occurs in real time. • Properties of displays can be easily modified to change color, data type, etc.	
d) Measurement Tab • Each component of this type is responsible for measuring a certain type of physical quantity. • Circular gauges & meters can be used for pressure reading while thermometers are used in thermal processes. • Maximum and minimum values and measurement units can be modified for gaining the highest accuracy.	

e) Switches Tab • Different types of switches can be used by the designer. • On/Off switch & multi-position switch can be used In process.	
f) Potentiometer Tab • Potentiometer is mainly used for varying a certain parameter in the system. • It can also act as an output display. • Setting potentiometer range can be easily handled from the property grid of the software.	
g) Controller Tab • There are two types of controllers to help the designer to control the required process. • ON-OFF & PID controllers are widely used in industrial process. • Required tuning parameters for both controllers can be set from the property grid or during run-time. • Set point, proportional gain, integral gain and differential gain are the required parameters for tuning the PID Controller. • Set point and neutral zone are the required parameters for the ON-OFF controller.	

10.1.2 Property Window

The second window in Process Control Studio is the **Property Widow** which allows manipulation of the properties for a form or control component by:



- Setting all values of any controller or component.
- Setting any general property for any control
  - Size ,Position
  - Name , Type, Color
  - Expired Date
  - Last Maintenance Date
  - Maintenance Period
  - Channel to Read or Write
- Setting a specific property of some component like:
  i. Set Point, Kp, Ki, Kd for PID Controller
  ii. Set Point , Neutral Zone for ON-OFF Controller
- Any Change in any property or value will be shown in the Design area.

- The bottom of the Properties window contains a description of the selected property.
- The left column of the Properties window shows the properties of the control the right column displays their current values.
- The Properties window allows programmers to modify controls visually.

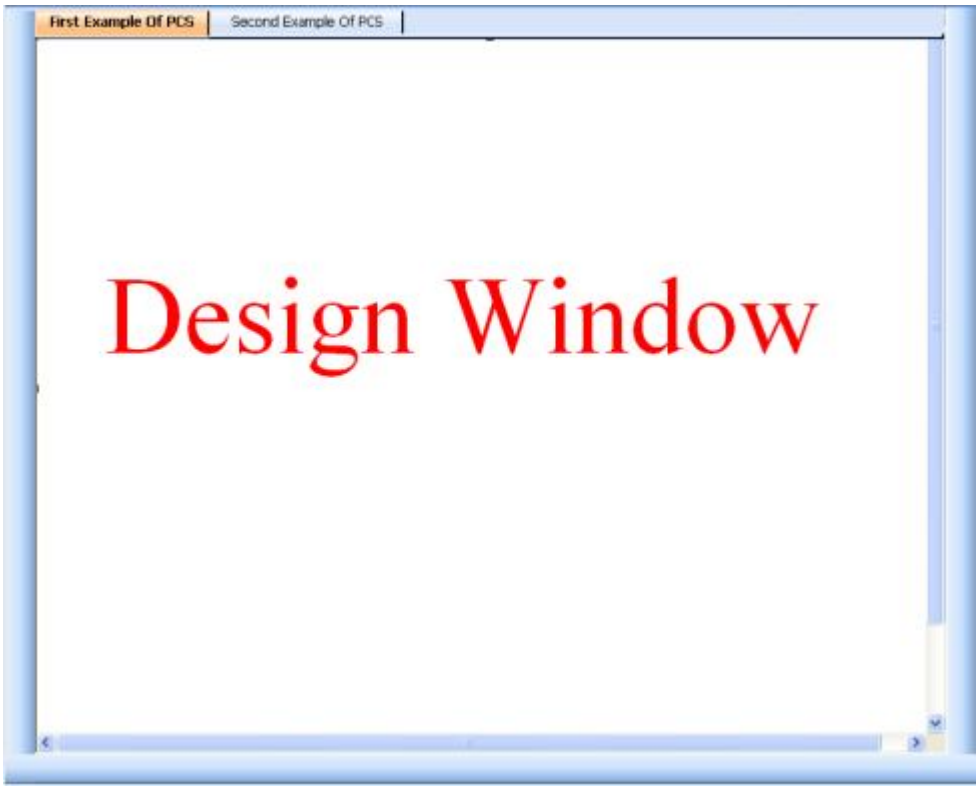
10.1.3 Design Window

- This is the main area of designing a process.
- Any connections like pipes or wires shall be implemented in the design area.
- As shown the user can open more than one design window at the same time and can switch between them using tabs.

- Designing the process is the control engineer's responsibility ,setting the channel of the coming data is handled also by the engineer.

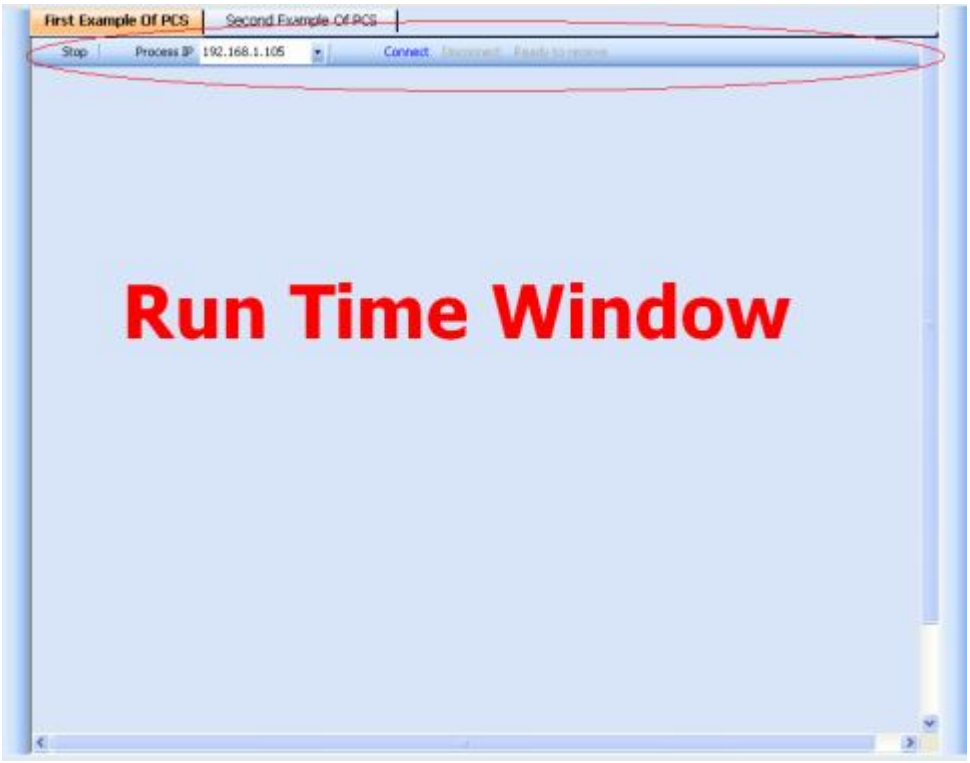
- By using the control components from PCS Toolbox a process can be built.
- Designing the process means creating a model for a real process on the program, connecting them, and setting their states, initializations and properties exactly as it is in the real process.

- The better design model the better and easier monitoring and controlling becomes.
- The design model can be modified at any time by an engineer after stopping the process.
- As shown, the background color of design window is white.

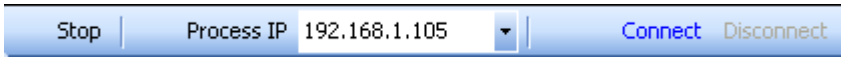


10.1.4 Process Run Time Window

- Perform Connecting & Disconnecting Events with the industrial process.
- Communicate with the process (sending & receiving data and commands).
- Take control action by changing parameters of the controllers (PID and ON-OFF Controller.
- Plot process response.
- Show History for each component in the process.
- Monitoring and controlling process in real-time.
- Monitoring and controlling more than one process at the same time using multiple tabs.



10.1.4.1 Runtime Toolbar



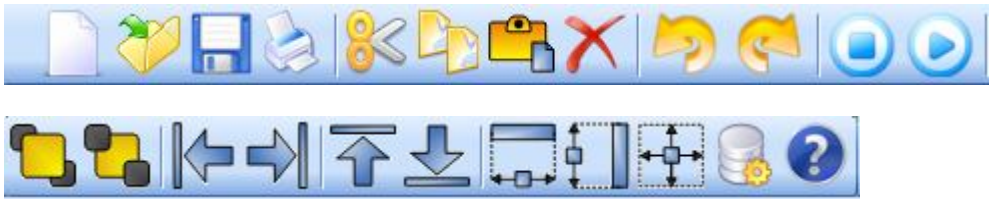
- Choose IP address of the local controller
- **Connect** is used to connect to the process and start sending and receiving data.
- **Stop** is used to stop receiving data from process but not to disconnect from the process.
- **Disconnect** is used to disconnect from the process.

10.1.4.2 How Process Control Studio Performs Communication through a Network

- Using Ethernet controller (refer to Part 2 of the documentation) PCS can simply access any process over the LAN whether the process is wired or wireless and the operator can easily apply many actions on the process.
- Protocol used in communication between the PC and the process is TCP rather than UDP. This is because UDP has a very high error rate along wireless links.

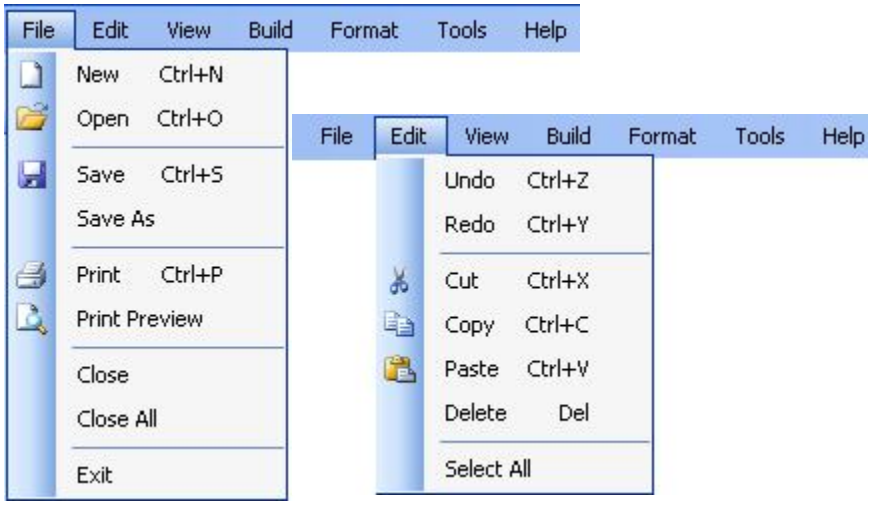
10.1.5 Process control Studio Toolbar & Menu bar

- Toolbar consists of usual features:



  - **New Process:** to create a new process.
  - **Open Process:** to open a process previously designed and saved.
  - **Save Process:** to save a process after designing it.
  - **Print Process:** to print a copy of a process and its information.
  - **Cut , Copy, Past, Delete, Undo, Redo**
  - **Run:** to run the design and switch from design window to runtime window.
  - **Stop:** to stop runtime window and switch to design window.
  - **Bring to Front** and **Send To Back**
  - **Align Left**, **Right**, **Top**, **Bottom**
  - **Make Same Width**, **Height** and **Size**
  - **Database:** to go to PCS Database
  - **Help**

- Menu Bar consists of the common features and some extra features:



10.2 How To Use Process Control Studio

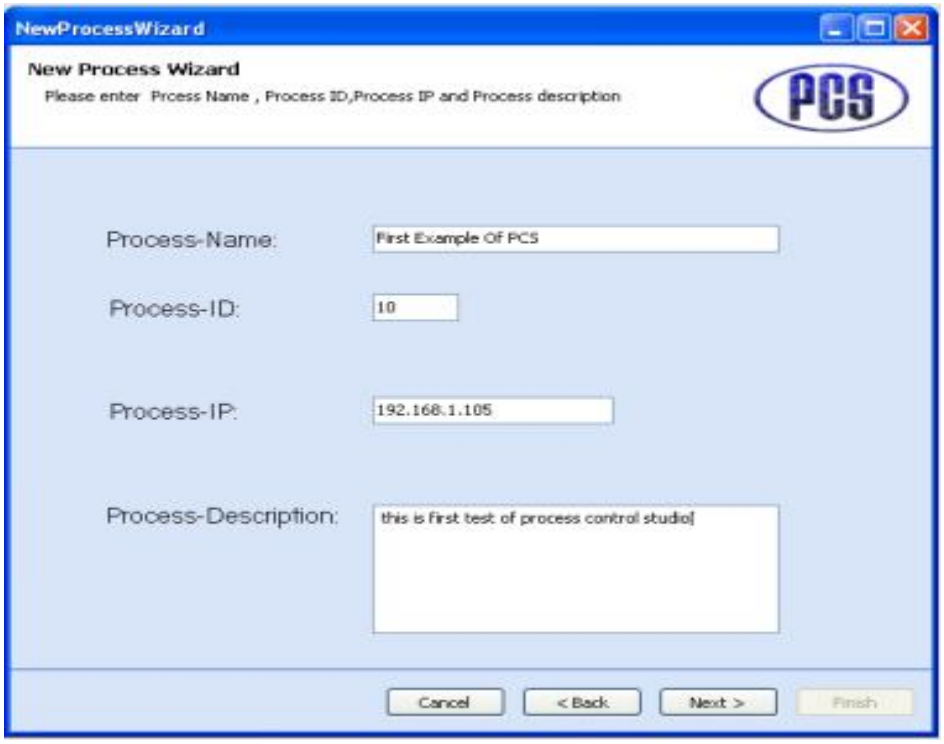
Now we are going to explain briefly in easy steps how to use PCS to design a process and how to monitor and control it.

1-Check the minimum requirements.
2-Insert the Setup CD intro your CD Drive.
3-Double click **setup.exe** to install PCS.
4-Click on your Start Menu: **Start**> **Programs**> **Process Control Studio**.
PCS has now started.

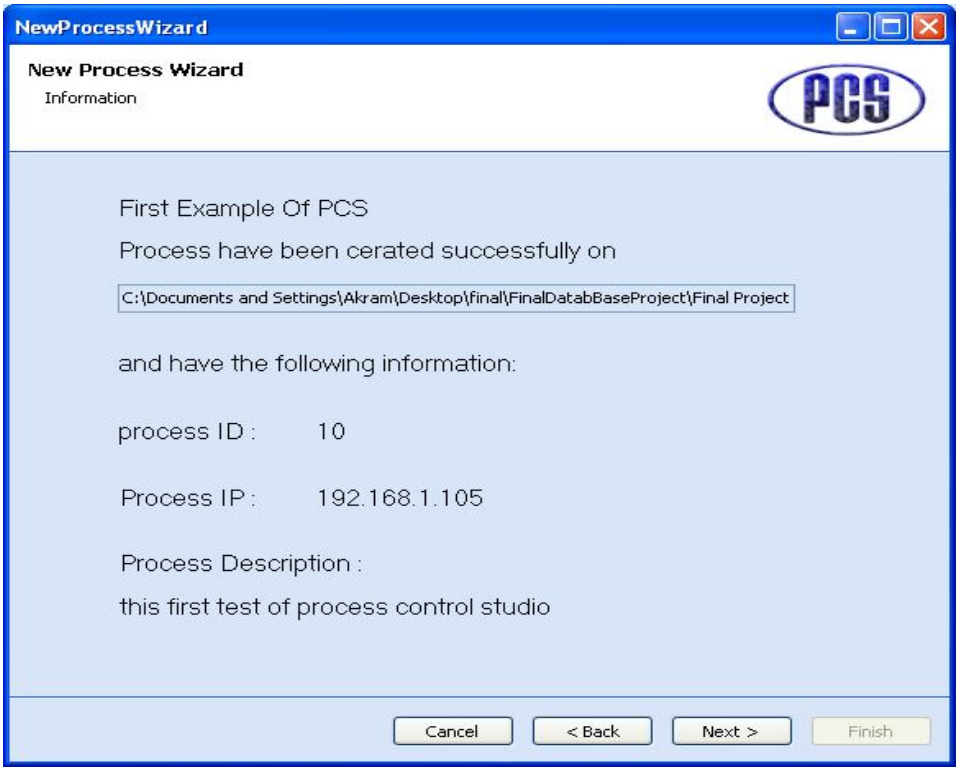
5-Choose **New Process** to create a new process design layout.
This Wizard will help you to create a new process easily.

5.1-After clicking **Next,** enter **Process Name** ,**ID** ,**IP** and **Process Description**.

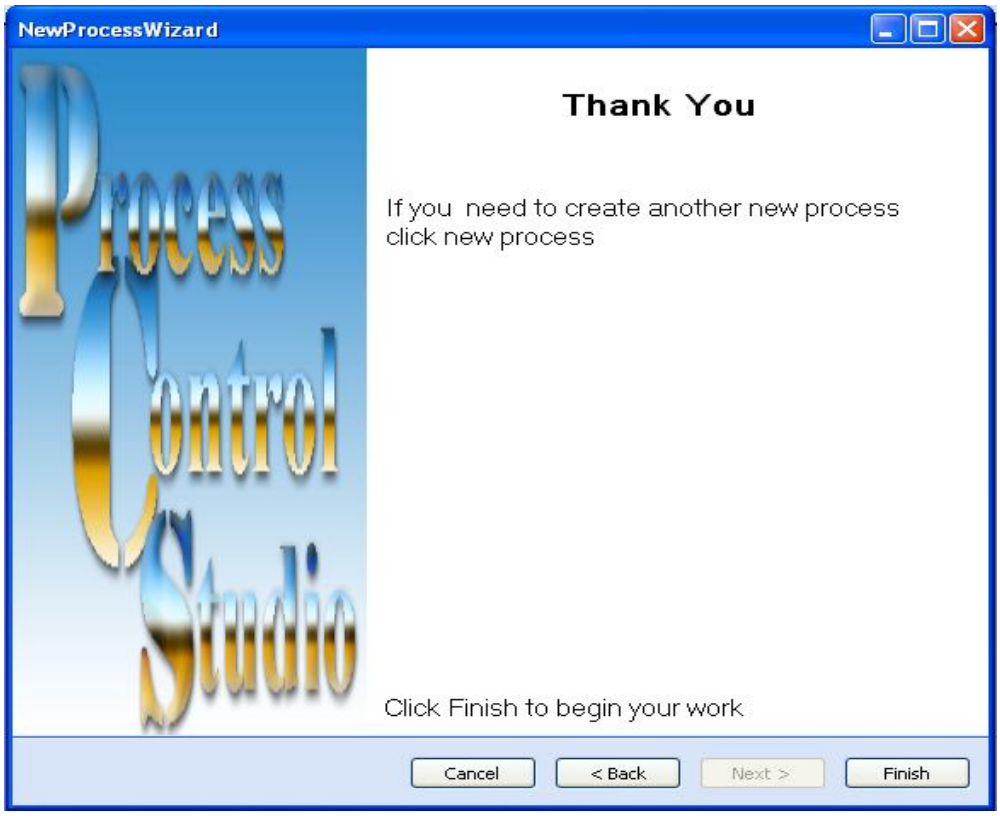
This information will be stored in database and will be use during communication.



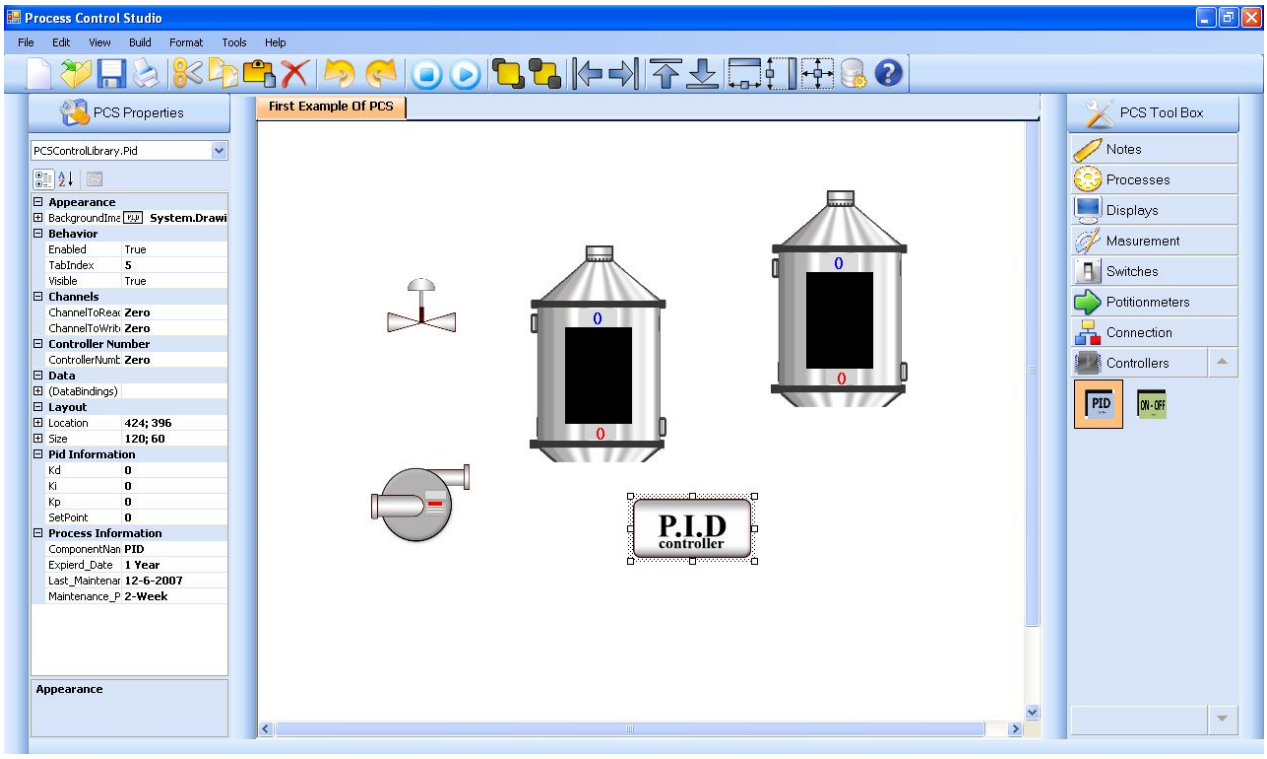
5.2- Click **Next** and if there is no duplicated information you will be told that the process file has been created and this information is stored in database



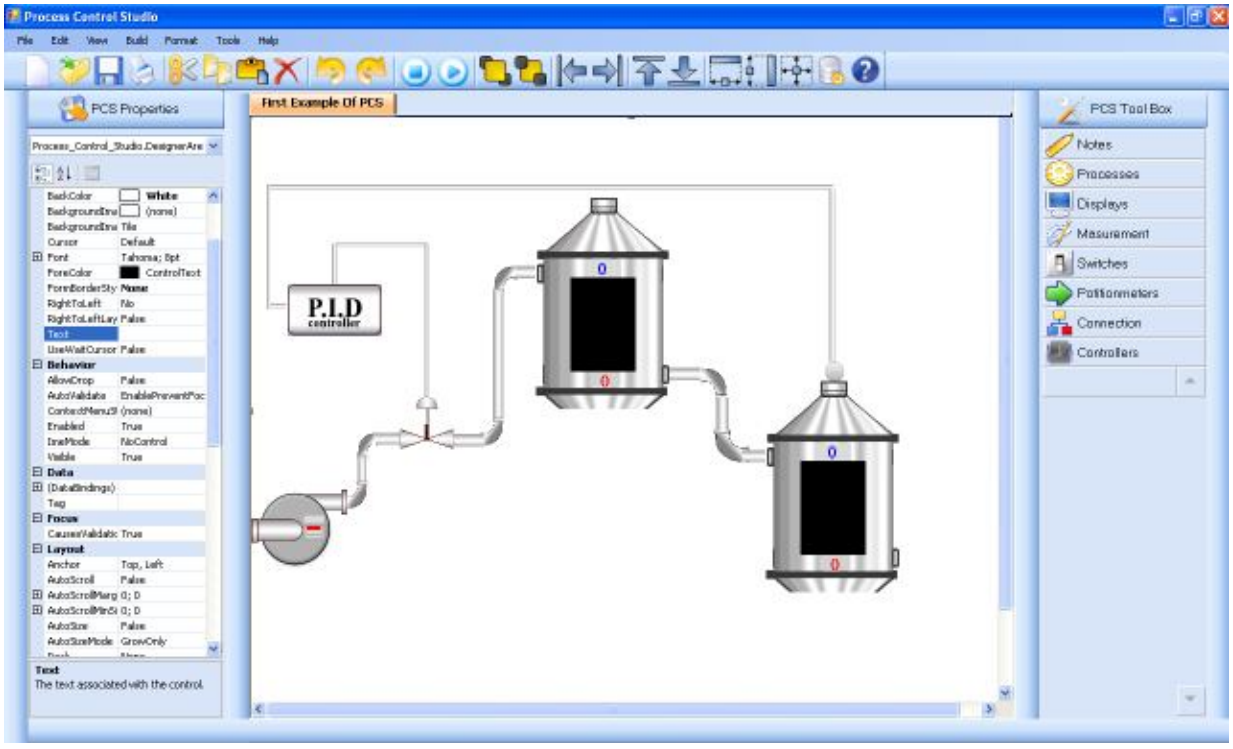
5.3- Click **Next** and **Finish** to start your work.



6-Start dropping the required process components from the PCS Toolbox on to the design area. As an example here, we select two tanks, a pump and a valve.



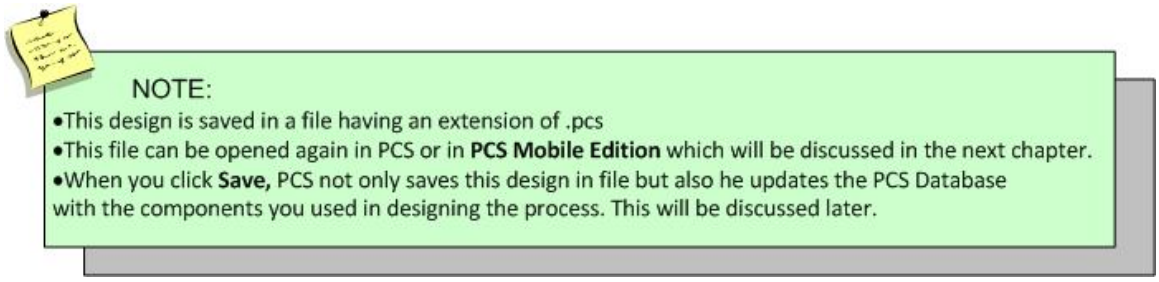
7- Connect the process components together using pipes, wires and connectors.



8- Set the process properties, values and parameters using the property grid.

9- Set the channels for each input or output signal for each component.

10- After your finish click **Save** to save the design with its initial values and properties so it can be easy for you to open it again.



10- Finally click Run from PCS Tool bar to start acting run mode

10.3 Process Control Studio Examples

Example 1: Alarming System

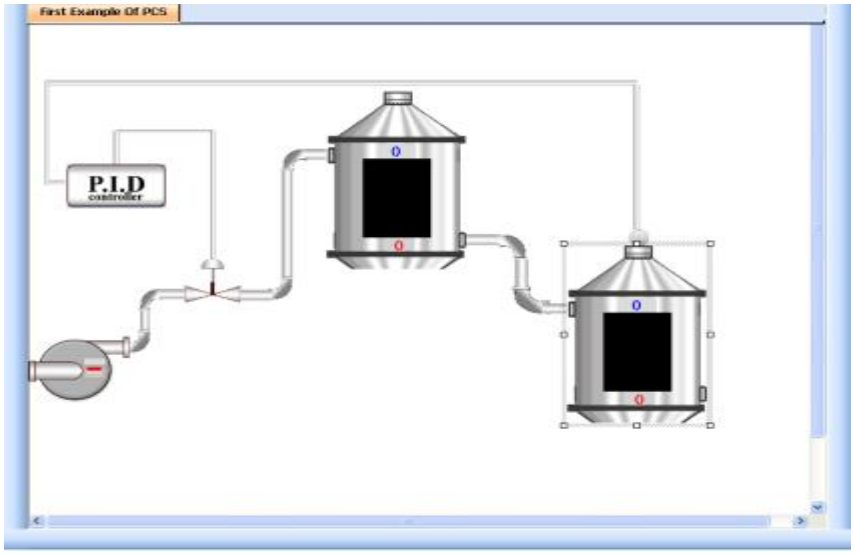
This example tell us about the alarming system that Process Control Studio provides.

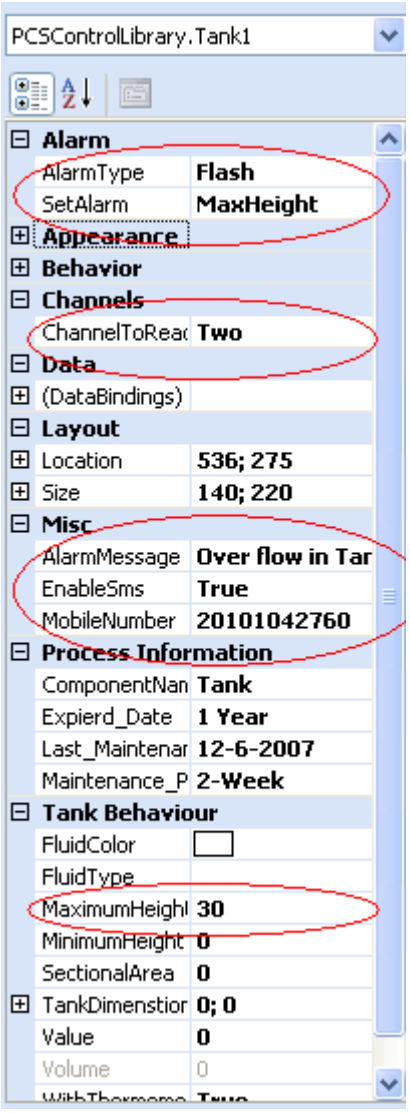
PCS provides three type of alarms:

- Sound
- Flash and notification window
- SMS using SMS through a web server

Also any event of alarming occurs will be stored in PCS database to help control engineers to know which process has a large number of alarms.

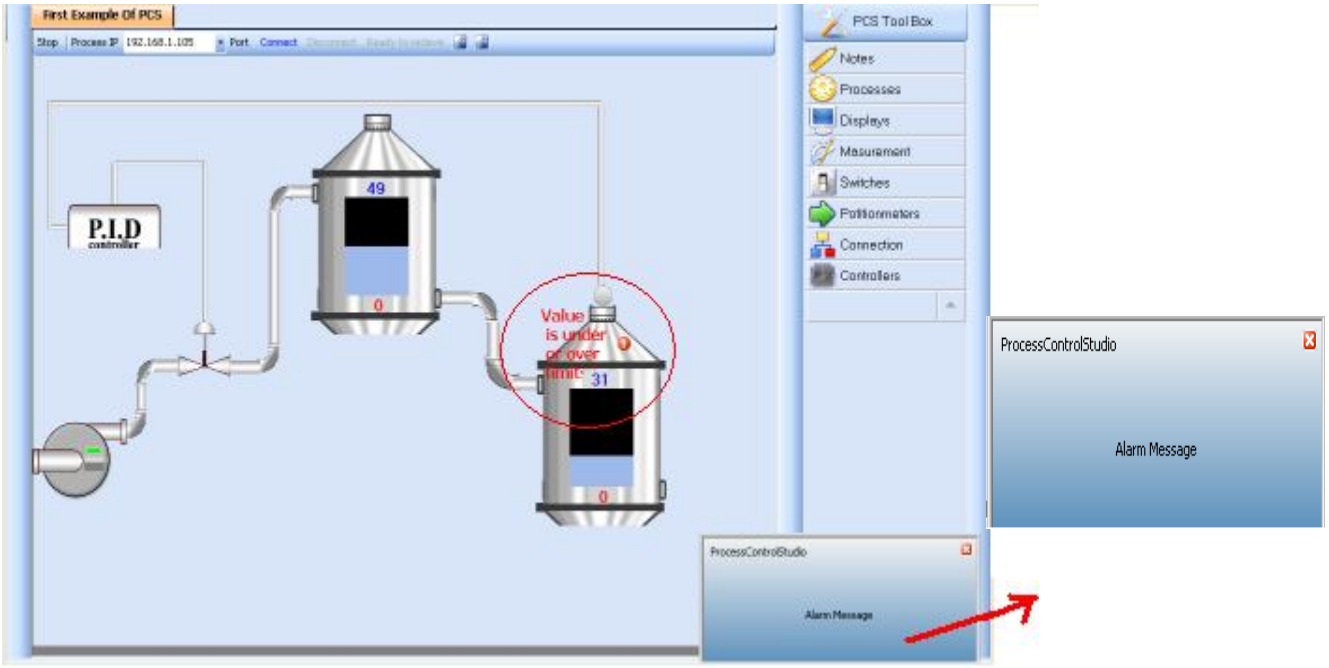
Looking at the previous example, we can set the properties of a tank as follows to enable the alarming system:





If the value of the tank becomes more than 30, which is the maximum height of the tank, the alarming system will notify the control engineer.

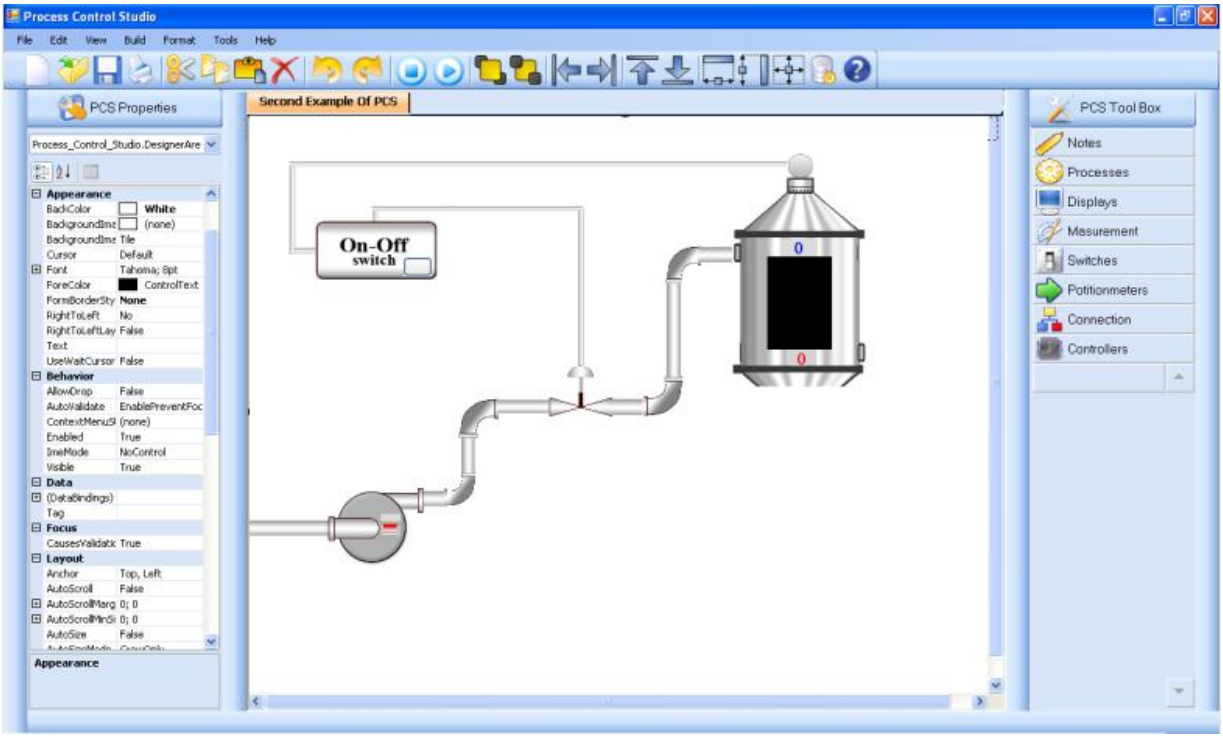
By: Flash , SMS and a notification window will appear which means that this alarm has been stored in PCS Database.



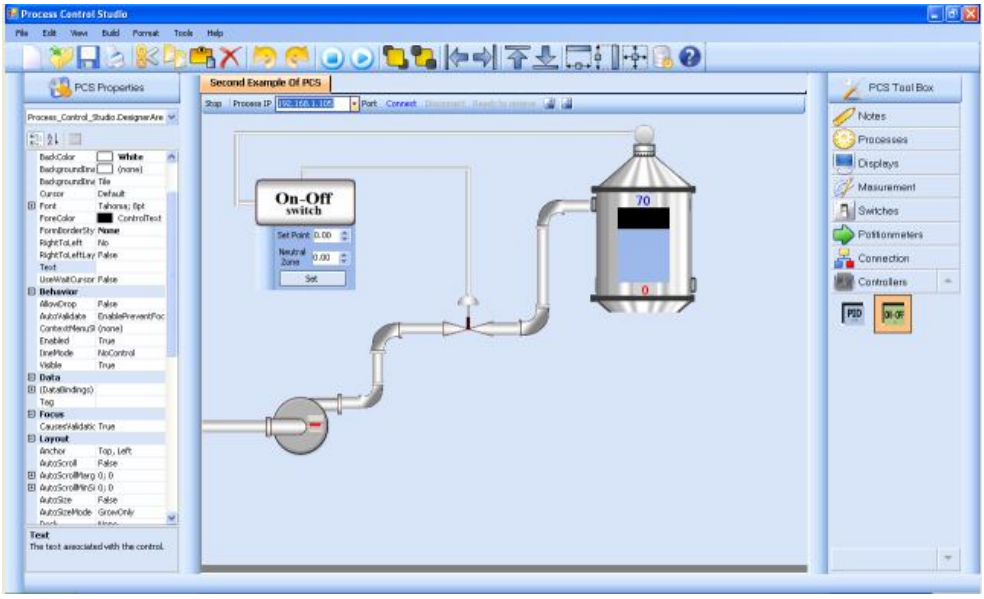
Example 2 :ON –OFF Controller

1- Run Process Control Studio
2- Start New Process
3- Start dragging and dropping the required components and connect them using pipes and connectors.
4- Start setting the properties of each component.

i. Set ON-OFF Controller on channel **Two** for **Channel to Read** and channel **Six** for **Channel to Write**.
ii. Set **Neutral Zone** to 20 , and **Set Point** to 80
iii. Set tank on channel Two for **Channel to Read**.
iv. Set valve on channel Six for **Channel to Write.**



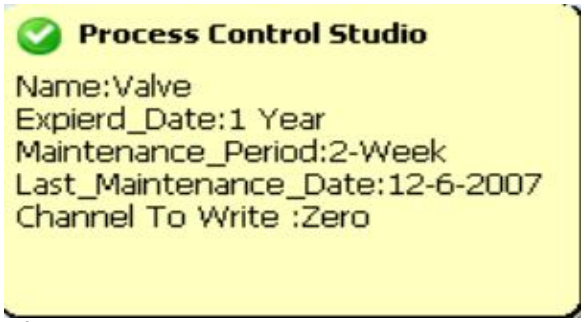
5- Click **Save** to save the design.
6- Click **Run** to start the run mode.
7- From run mode toolbar, choose the IP address of the local controller then click **Connect**.

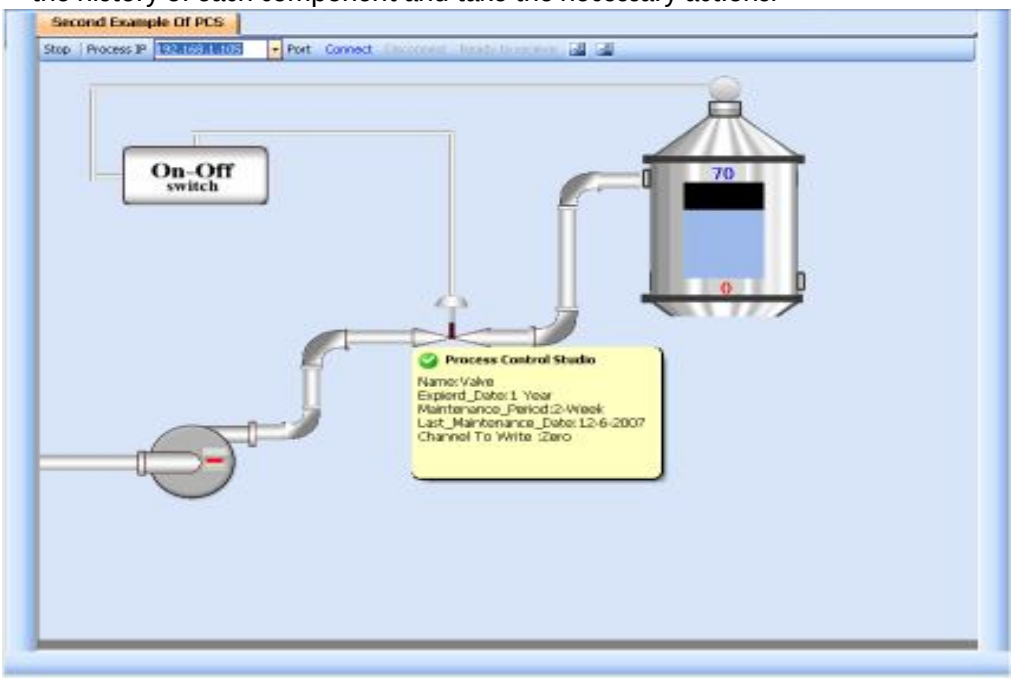


8- While you are in run mode and monitoring your process, you can take a control action by changing the ON-OFF controller **Set Point** or **Neutral Zone** by right clicking on **ON-OFF controller.**

9- After your finish your work click **Disconnect** and **Stop**.

10- If you move the mouse over any component during run mode, a tool tip appears displaying information about this component such as maintenance date, maintenance period, expired date, channel to read and channel to write. This information will help the control engineer to know the history of each component and take the necessary actions.





10.4 Process Control Studio Database

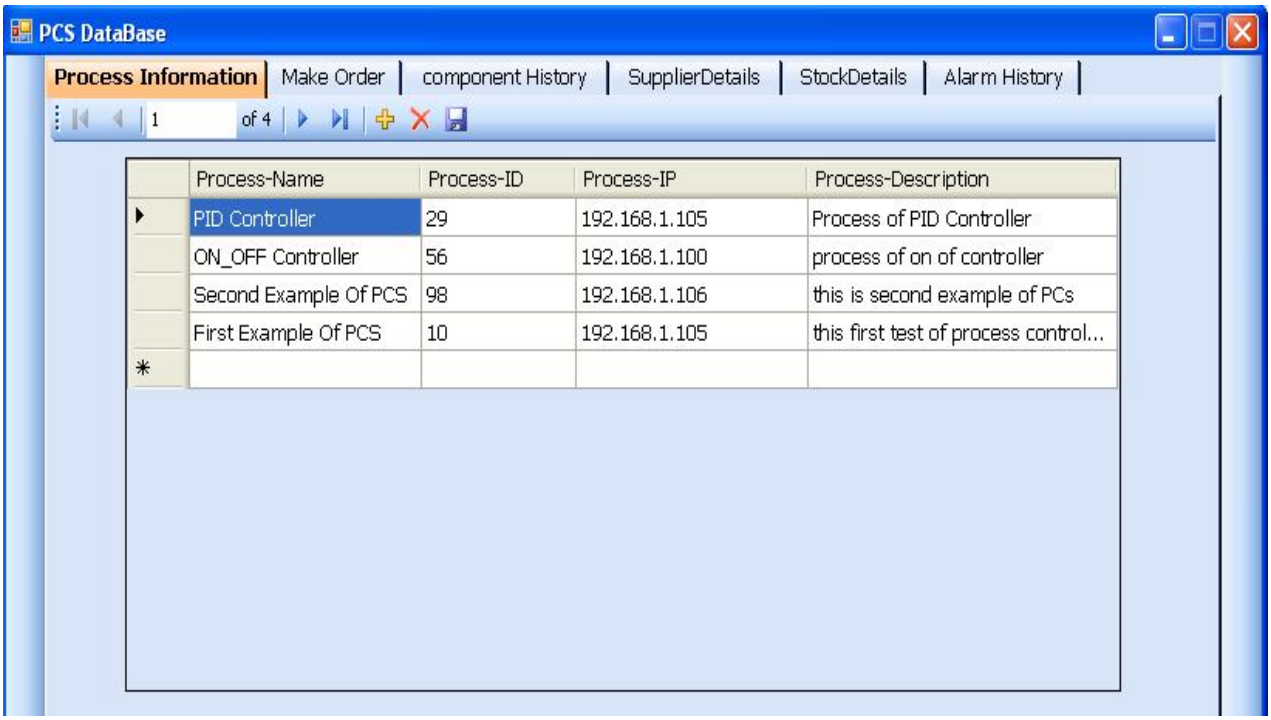
Process control Studio Database consists of six tables:

1-Process Information Table
2-Component History Table
3-Stock Details Table
4- Supplier Details Table
5-Make Order Table
6-Alarm History Table

Tables Description

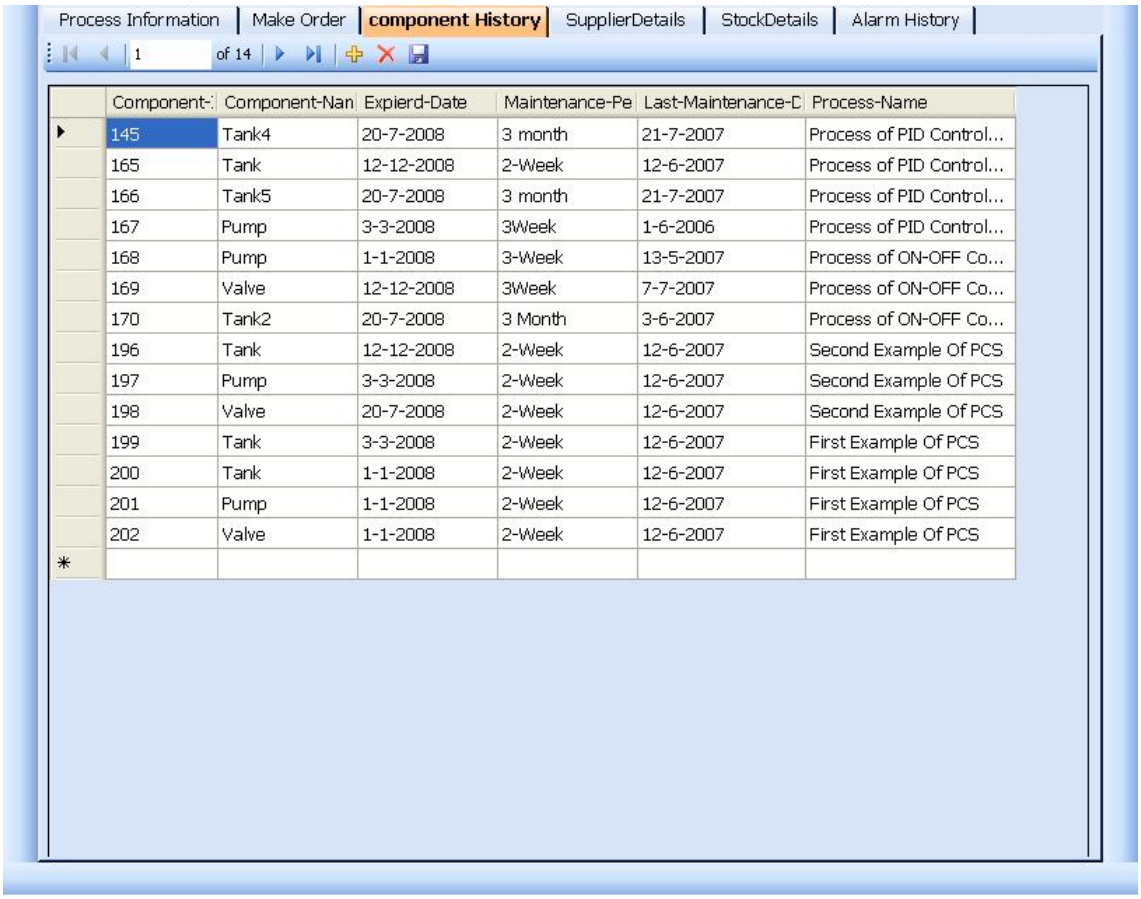
Process Information Table :

- This table stores information about processes like Process Name, ID, IP and Process Description.
- This information is stored into the database after you click **Finish** in New Process Wizard.
- This information will be useful in connecting with the local controller.
- Database must not have duplicated Process Name or Process ID. PCS will refused storing information duplication occurs, telling you that you must check that there is no duplication in Process Name or ID.



Component History Table:

- It stores information about each component in process such as Name, Maintenance Date, Maintenance Period and Expired Date.
- This information is stored into the database once you click **Save**.
- At the first when you design the process you should enter the Maintenance Period and Expired Date.
- It is very useful to know the states of each process and generating reports.

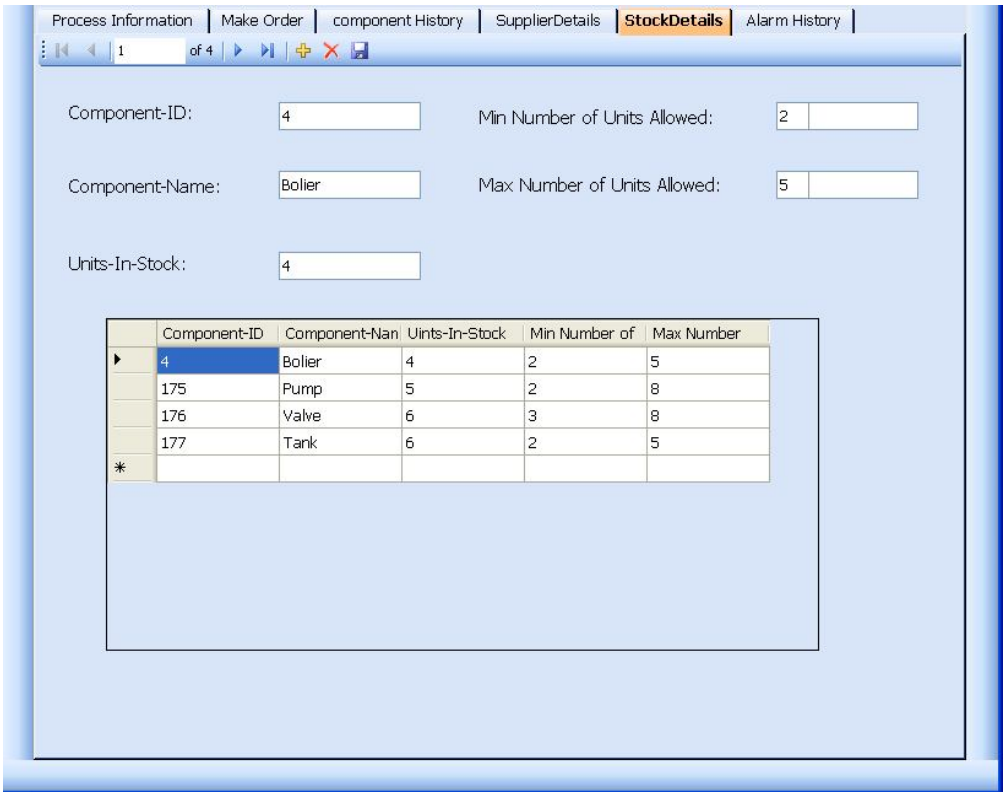


Component-:	Component-Nan	Expierd-Date	Maintenance-Pe	Last-Maintenance-D	Process-Name
145	Tank4	20-7-2008	3 month	21-7-2007	Process of PID Control...
165	Tank	12-12-2008	2-Week	12-6-2007	Process of PID Control...
166	Tank5	20-7-2008	3 month	21-7-2007	Process of PID Control...
167	Pump	3-3-2008	3Week	1-6-2006	Process of PID Control...
168	Pump	1-1-2008	3-Week	13-5-2007	Process of ON-OFF Co...
169	Valve	12-12-2008	3Week	7-7-2007	Process of ON-OFF Co...
170	Tank2	20-7-2008	3 Month	3-6-2007	Process of ON-OFF Co...
196	Tank	12-12-2008	2-Week	12-6-2007	Second Example Of PCS
197	Pump	3-3-2008	2-Week	12-6-2007	Second Example Of PCS
198	Valve	20-7-2008	2-Week	12-6-2007	Second Example Of PCS
199	Tank	3-3-2008	2-Week	12-6-2007	First Example Of PCS
200	Tank	1-1-2008	2-Week	12-6-2007	First Example Of PCS
201	Pump	1-1-2008	2-Week	12-6-2007	First Example Of PCS
202	Valve	1-1-2008	2-Week	12-6-2007	First Example Of PCS

Stock Details Table:

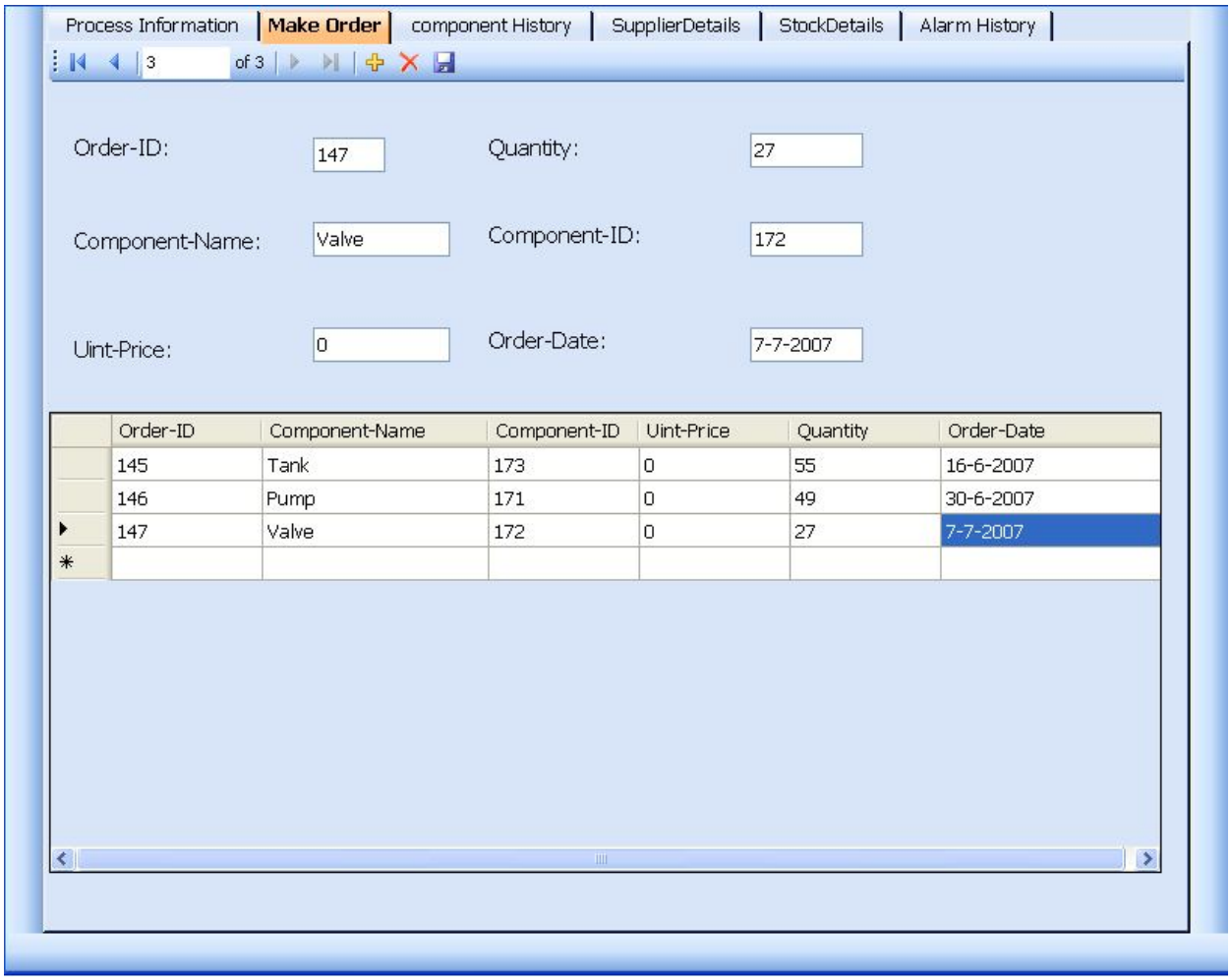
- It stores information about inventory and this information will help control engineers to know which component has a shortage and how to make order for it.
- This information is stored into the database when you click **Save Process** at the first time only.
- For example, if you design a process that contains two tanks and two valves and a pump, this means that process at real-time contains these components and you take them from the store when you click **Save**.
  PCS will deduce stock of tanks by value two and also valves and pumps.

- The two fields: Max Number of Units Allowed and Min Number of Units Allowed, are used to know which component have shortage:
  i. **Max Number of Units Allowed** tells the maximum number of this component to be in the stock when an order is made.
  ii. **Min Number of Units Allowed** tells the minimum number of this component that PCS will notify the control engineer to make order if the units in the stock of this component becomes less than **Min Number of Units Allowed.**

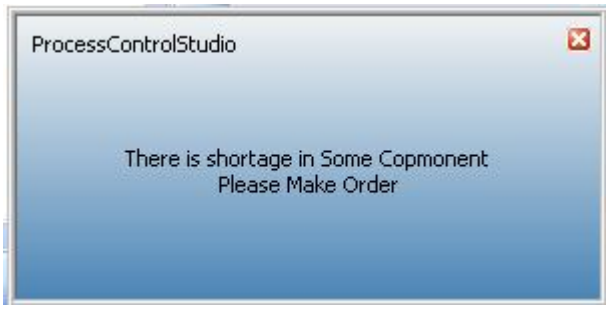


Make Order Table:

- It stores information about components which have previously faced shortages and had orders made for it.
- As explained previously, if any shortage happens PCS will notify the control engineer to make an order by displaying a notification window. If the engineer clicks on it, PCS database form will open to show which component is facing shortage.
- **Quantity** ordered = **Max Number of Units Allowed - Units in Stock**

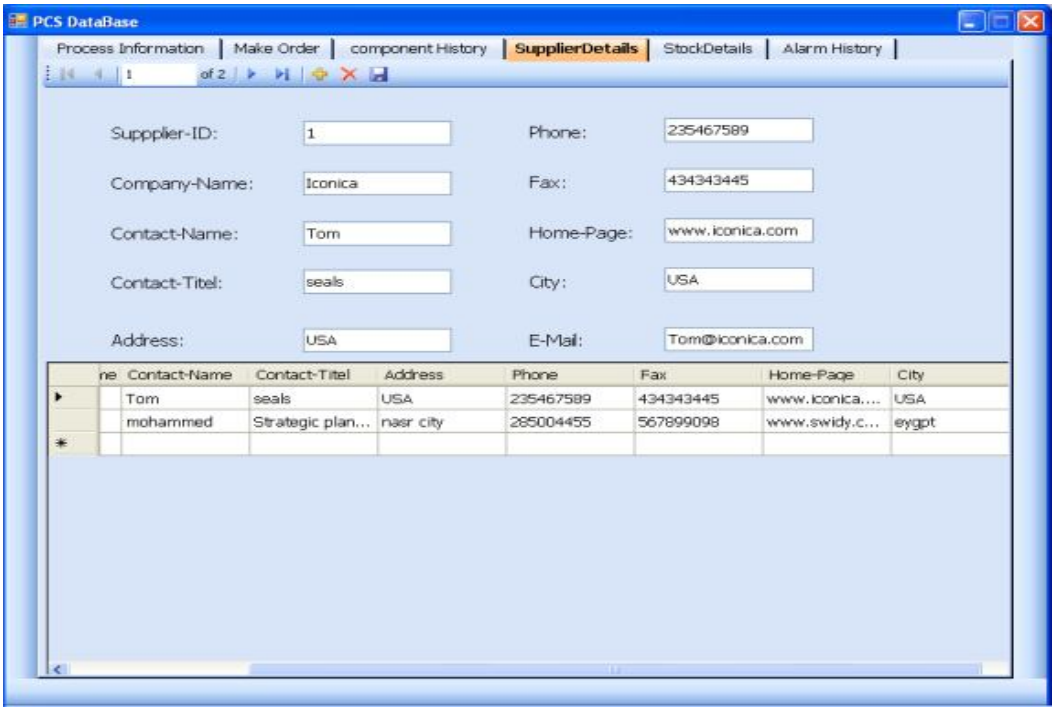


- This is the notification window which will appear when a shortage happens in some component. If you click on it, PCS Database will open to show you which component has shortage and therefore make order for it.



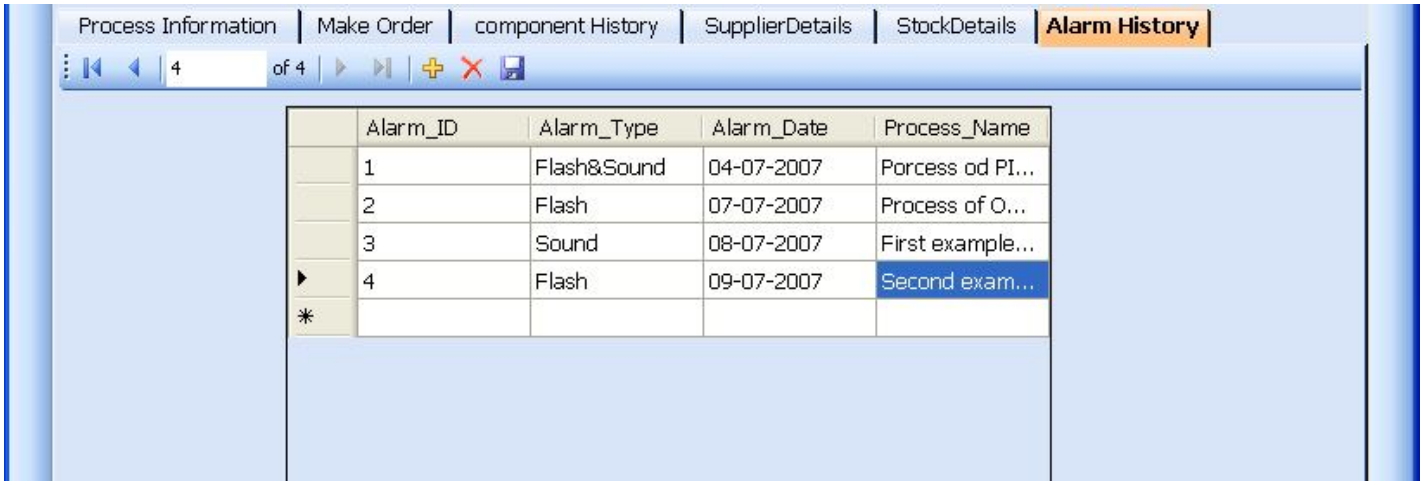
Supplier Details Table:

- It stores information about suppliers which supply the plants components.



Alarming History Table:

- It stores information about alarming such as
  - i. Alarm Type
  - ii. Alarm Date
  - iii. Process Name
- It is updated in real time when an alarm occurs due to a certain process.
- This information will be useful in generating reports to show the status of each process.



10.5 Process Control Studio Users

PCS has two types of user s (default users):

1- Control Engineer
2- Operator

1- Control Engineering can perform the following jobs:

- Designing the process.
- Setting the properties of each component in the process.
- Add and delete users form the database.
- All jobs can done by operator user.

2- Operator can perform the following jobs:

- Load process form .pcs file.
- Monitor and take control actions.

When you run PCS, you will be asked about your user type, as shown:

If you are a control engineering

If you are an operator

Chapter 11:
Process Control Studio Mobile Edition

11.1 Introduction

11.1.1 What is PDA? [1]

Personal digital assistants (PDAs) are handheld computers that were originally designed as personal organizers, but became much more versatile over the years. For more information refer to Appendix.

11.1.2 .NET Compact Framework (.NET CF)

The Microsoft **.NET Compact Framework** (.NET CF) is a version of the .NET Framework that is designed to run on mobile devices such as PDAs, mobile phones and set-top boxes. .NET CF features are listed below [2]

- .NET Compact Framework Capabilities

- Supported Runtime Elements

- The .NET Compact Framework Provides CPU Independence

11.1.3. Operating System Used in This Project

Team of the project came to an agreement to deal with **Windows Mobile for PDAs** using one of the most recent Microsoft's Pocket PC Operating Systems which is **Windows Embedded CE 5 (WinCE 5)** shown in Fig .2**.**

Nowadays Microsoft has launched Windows Embedded CE 6 (WinCE 6), but still having no bread popularity. [3]

---

[1] For more information refer to "What is PDA?" in the Appendix.

[2]For more information refer to "**.**NET Compact Framework (.NET CF)" in the Appendix

[3]For more information refer to "Windows CE Overview" in the Appendix.

Figure 11.1 PDA running WinCE 5 Platform

11.2 Program Description

From the previous introduction about the .NET Compact Framework (.NET CF) world, we move then to discussion of our program that is used in process control that's why we named **Process Control Studio Mobile Edition**.

Process Control Studio Mobile Edition is a program that runs on PDA under WinCE 5 operating system which is used for process view and control; from this goal of our program we merged the program into two tasks:

A. Loading the displayed form of the program by reading an XML file that saves the forms contents and attributes in it, this XML file is the same as the PC (Personal Computer) XML file used to load controls on PC display form. This was done to achieve data consistency between PC and PDA.

B. Dealing with Network Sockets on PDA to receive and send data required for controlling parameters under the constrain of the same protocol implemented on the PC , this is another way of making real world problems come to an agreement to the Handheld PCs world.

A.1 Reading an XML File

The **Extensible Markup Language** (**XML**) is a general-purpose markup language .Markup language as it doesn't do anything except describing data; it is classified as an extensible language because it allows its users to define their own tags. Its primary purpose is to facilitate the sharing of data across different information systems.

A.2. Scaling

This is what is done in our project, the same XML file that is read on the PC program, is also read on the PDA program, but with scaling the controls and the display form to be managed by the size of the PDA display screen.

A.3. Program Sequence

1. When user opens the program (named Process Control Studio Mobile Edition), he can choose to click File menu and select from it Open choice, shown in Fig.11.2.

2. By clicking Open, an Open File Dialog Box will be opened presenting all PCS (Process Control Studio) extension files that now present on PDA specified folder named PCS Files.

3. User at this moment can select which file would he like to open to view its display on the PDA screen, shown in Fig.3.

4. After selecting the file two tabs are created on the form, the first one is for the display form and the other is for viewing a graph representing a process graph, shown in Fig.4.

5. When the form is displayed, the user can click on any control on the form to display its panel to set and change parameters of that control, shown in Fig.5.

6. Program termination occurs when user clicks Close at any time of the program sequence.

A flowchart is shown in Fig.11.6 to illustrate the program sequence

Figure 11.2 User clicks Open from File menu

Figure 11.3 User selects a file to open

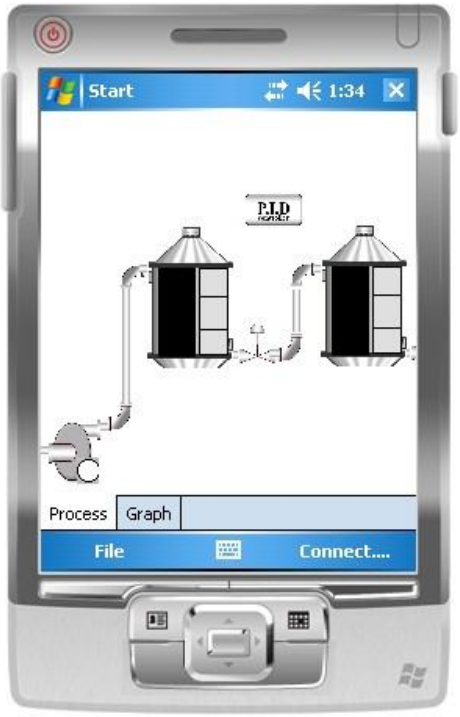


Figure 11.4 Form display

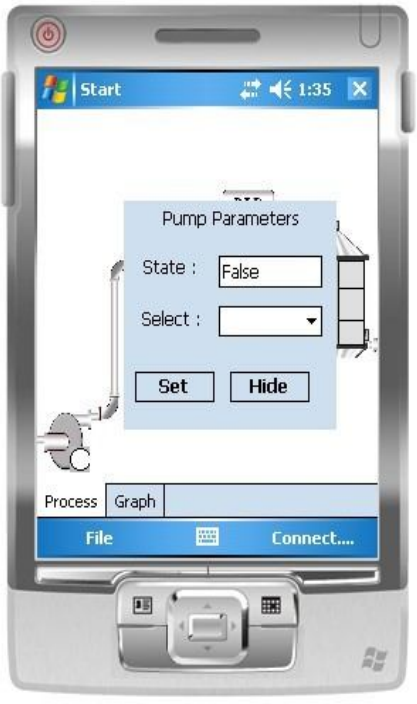


Figure 11.5 Control parameters

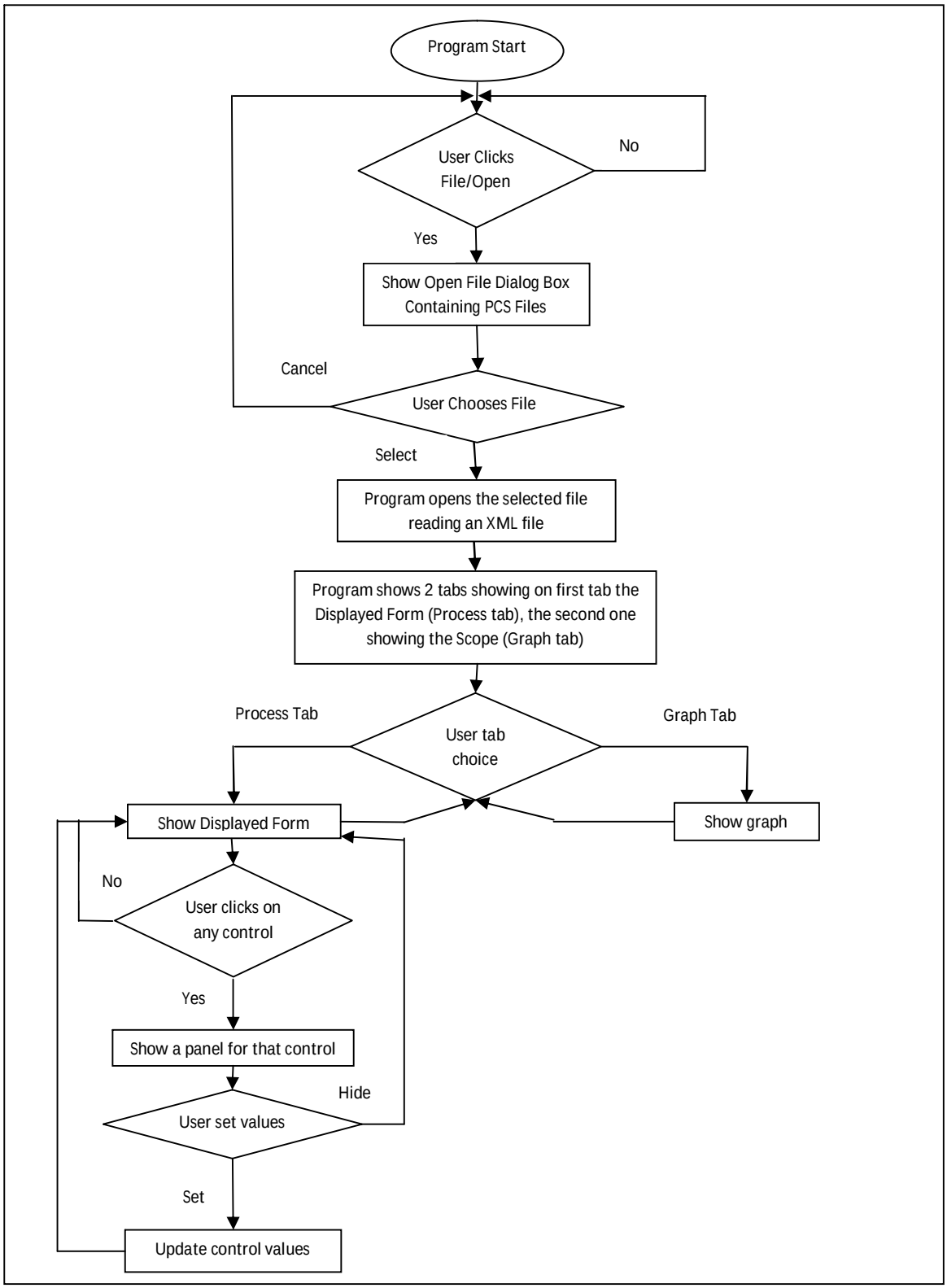


Figure 11 .6 Flowchart showing sequence of XML reading part in the program, note that termination of the program can occur at anytime of clicking Close

B.1 Socket Programming [4]

Network programming in WinCE is possible with sockets. A socket is like a handle to a file. Socket programming resembles the file IO as does the Serial Communication. Sockets are a communication mechanism.

A **socket** is defined by a group of four numbers, these are

- The remote host identification number or address
- The remote host port number
- The local host identification number or address
- The local host port number

B.2 Protocol Used in This Project

As shown in the figure each layer of the Network Protocol Standards consists of series of protocols our concern will be the protocols of the Transport layer which are TCP/IP (Transmission Control Protocol/Internet Protocol) and UDP (User Datagram Protocol).In our project we agreed to use the TCP/IP protocol for it's efficiency and connection reliability (Connection Oriented Protocol).Also TCP/IP is also in widespread use on commercial and private networks. In according to that we managed to create a protocol of our own that is included in the index of this documentation.

B.3 Program Sequence:

1. After reading all the controllers from the xml file the user would have to press the connect button and select the name of the process he wants to connect to.

2. The program goes through the protocol illustrated in the index to connect to the server built on the microcontroller.

---

[4]For more information about sockets refer to “PDA Sockets Programming” in the Appendix.

3. If no error occurred or nothing goes wrong the program will start a thread to listen and receive data from the server obtaining all information about the process and redrawing all components and controllers with their received values on the process tab of the form.

4. Pressing the graph tab will display a tab page that contains a graph that represents the response of each tank on the displayed form with different colors for different types of controlled elements, shown in Fig.7 and Fig.8 .

5. The user is now able to see the current running process with its values and he can adjust any of the control parameters of any component in a friendly user interface way

6. If any of the control parameters is adjusted a send button will become visible to the user.

7. Pressing that send button will start a loop whose task is to get all the parameters of the controls displayed on the form and send them to the microcontroller which in turn will adjust the actual process to the received values.

8. It will start the thread that receives from the server the parameters of the controllers and redraw them with their adjusted values.

9. When ever the displayed form is closed the socket built to handle this connection is closed and the connection is terminated.

A flow chart is shown in Fig.9 representing the sequence of the connection part of the program.

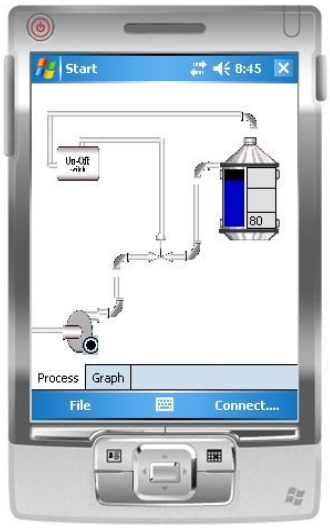


Figure 11.7 A process running on the process panel

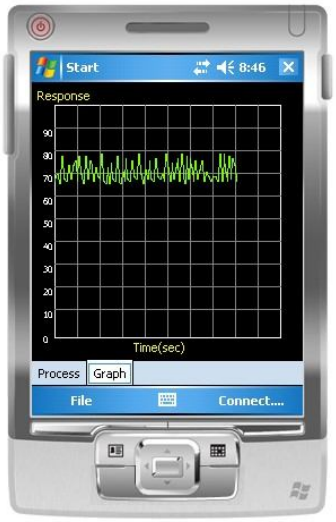


Figure 11.8 The corresponding graph for the tank shown in Fig11.7

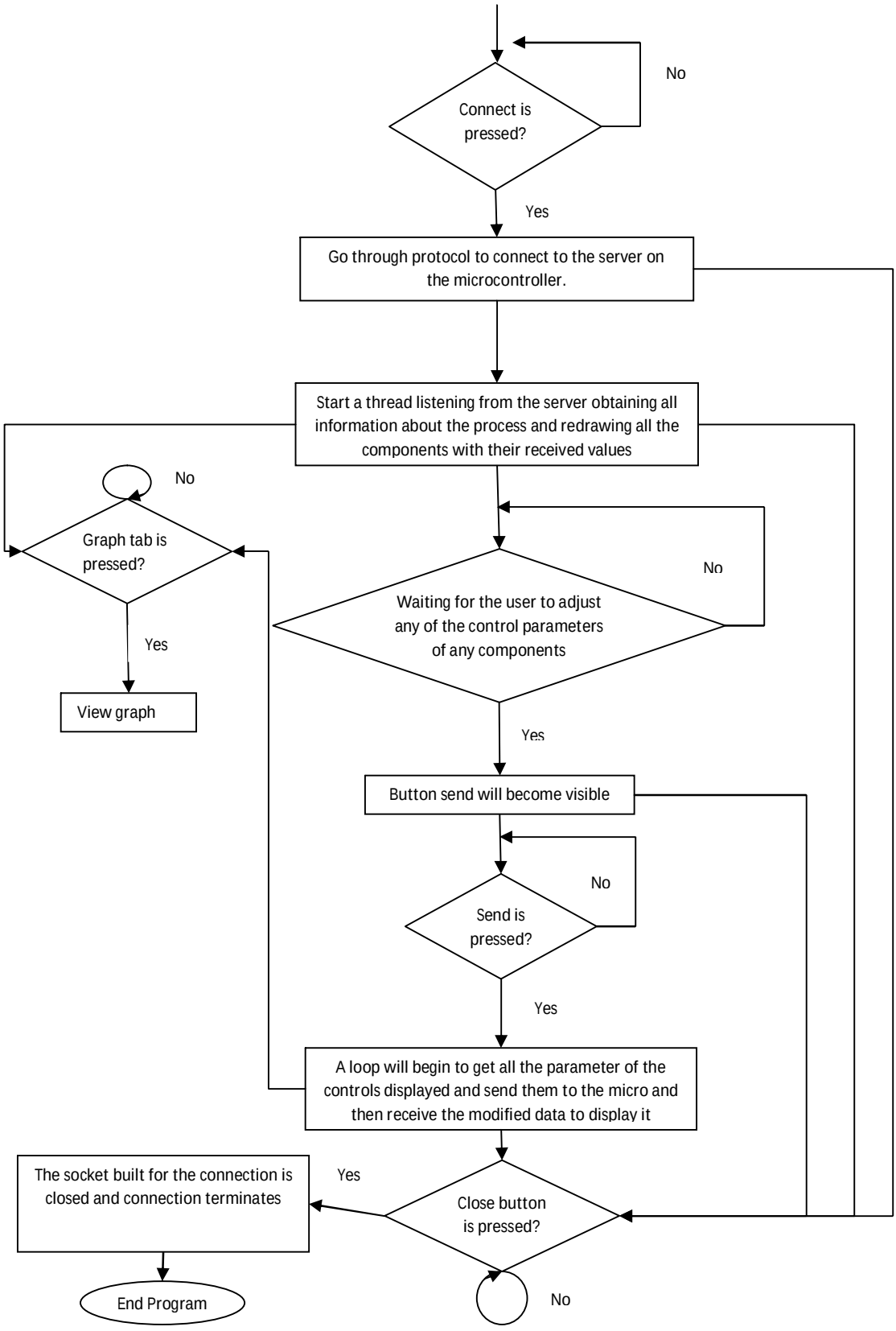


Figure 11.7 Flow chart of the connection procedure.

Appendix 1: Microcontroller Peripheral Functions

```
// Peripheral Functions Definitions-----------------------------
```

```
//******** ADC_Init ***************

// Initialize ADC

void ADC_Init(void){

  ATDCTL2 = 0x80; // enable ADC

}
```

```
//******** ADC_In ***************

// perform 4 ADC samples on chan return average 4 results

unsigned char ADC_In(unsigned char chan){

  ATDCTL5 = chan;                 // start sequence

  while((ATDSTAT0&0x80)==0){};   // wait for SCF

  return (ATDDR0H+ATDDR1H+ATDDR2H+ATDDR3H)/4;

}
```

```
//******** SPI_Init ***************
// Initialize SPI interface make.
void SPI_Init(void){     // PS7=_CS=1
  DDRS = 0xE0;           // PS6=CLK=SPI clock out
  PTS = 0x50;          // PS5=SRI=SPI master data out
/* bit SP0CR1
 7 SPIE = 0   no interrupts
 6 SPE  = 1   enable SPI
 5 SPTIE = 0   disable transmit interruptregular outputs
 4 MSTR = 1   master
 3 CPOL = 0   output changes on +ve edge
 2 CPHA = 0   and input clocked in on activation of CS
 1 SSOE = 0   PS7 regular output
 0 LSBF = 0   most significant bit first */
  SPICR1 = 0x50;
/* bit SP0CR2
 3 PUPS = 0   no internal pullups
 2 RDS  = 0   regular drive
 4 MODFEN = 0 Don't use SS
 3 BIDROE = 0
 1 SPISWAI =0 operate normally in wait mode
 0 SPC0 = 0   normal mode */
  SPICR2 = 0x00;
  SPIBR = 0x02; // 3.125 MHz
  WOMS = 0x00;    // Enable Port S Pull-Up
}
//******** SPI_Out8 ***************
```

```
// Output 8-bit data using SPI port

void SPI_Out8(unsigned char code){

  unsigned char dummy;

  PTS_PTS7 = 0;             // PS7=_CS=0

  while((SPISR_SPTEF)==0);    // wait until write is permissible




  SPIDR = code;             // data

  while((SPISR_SPIF)==0);    // wait until write operation is
complete




  dummy = SPIDR;            // clear the SPIF flag

  PTS_PTS7 = 1;             // PS7=_CS=1


}
```

```
//******** SPI_Out16 ***************
```

```
// Output 2 8-bit data using SPI port

void SPI_Out16(unsigned char code1, unsigned char code2){

  unsigned char dummy;

  PTS_PTS7 = 0;               // PS7=_CS=0


  while((SPISR_SPTEF)==0);    // wait until write is permissible




  SPIDR = code1;             // 1st data

  while((SPISR_SPIF)==0);    // wait until write operation is
complete




  dummy = SPIDR;            // clear the SPIF flag


  while((SPISR_SPTEF)==0);    // wait until write is permissible


  SPIDR = code2;             // 2nd data

  while((SPISR_SPIF)==0);    // wait until write operation is
complete

   dummy = SPIDR;            // clear the SPIF flag

   PTS_PTS7 = 1;             // PS7=_CS=1


}
```

```
//******** MAX528_Init ***************
```

```
// Initialize MAX528 interface
void MAX528_Init(void){
  SPI_Out16(0x00,0x80);  // Set all channels to no buffer mode


}
```

```
//******** MAX528_Out ***************
// Set required analog voltage to required channel.
void MAX528_Out(unsigned char channelNo , unsigned char value){
  unsigned char code1, temp      = 0x01;
      code1      = temp<<channelNo;
  SPI_Out16(code1,value);
}
```

```
//********* MAX528_NOP ***************
// No Operation
void MAX528_NOP() {
  SPI_Out16(0x00,0x00);  // NOP
}
```

```
//********* MAX528_NOP ***************
// No Operation
void MAX528_NOP() {
  SPI_Out16(0x00,0x00);  // NOP
}
```

Appendix 2: Microcontroller Layouts and Schematics

Pin	Pin Number	Purpose	Connect to
VDD	7	+ve terminal of supply	+12V
VSS	14	-ve terminal of supply	0V
GND	11	Ground	0V
$\overline{CS}$	12	Active Low Chip select	SPI slave select (PS7)
$\overline{SHDN}$	13	Active Low Shutdown	+5V
REFL1 and REFL2	1 & 20	-ve terminal of reference	0V
REFH1 and REFH2	2 & 19	+ve terminal of reference	+5V
DIN	8	Serial digital data input	SPI master data out of HC12 (PS5)
CLK	9	Clock	SPI clock out of HC12 (PS6)
OUT0-OUT7	3-6 & 15-18	Analog output	To processes

Table A2.1 Pin functions and connections of MAX528.

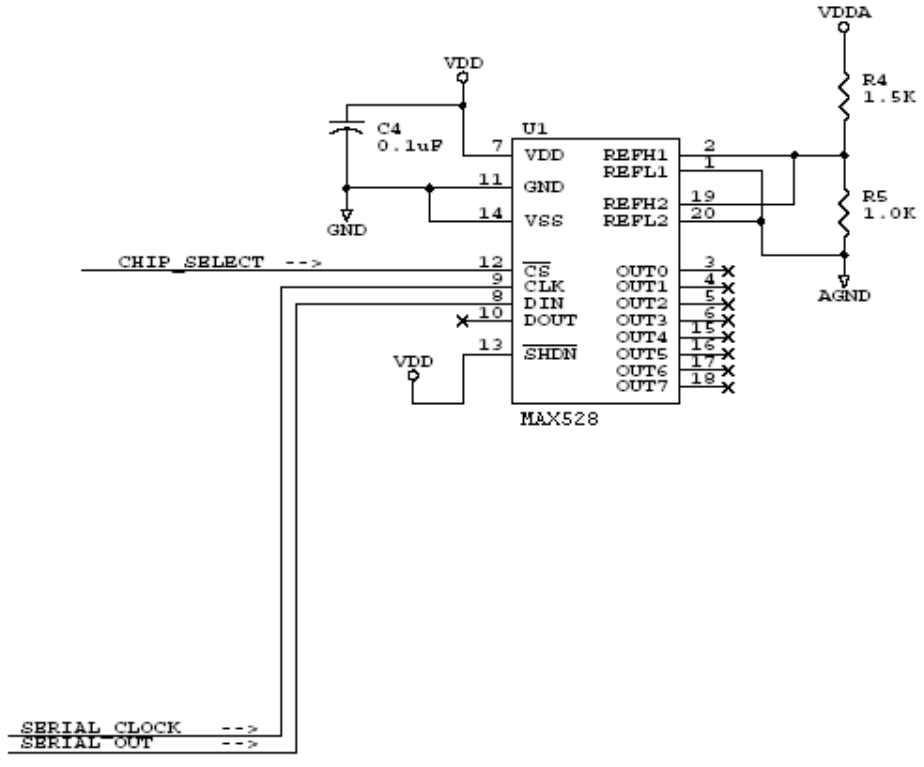


Figure A2.1 Circuit layout connection of MAX528.

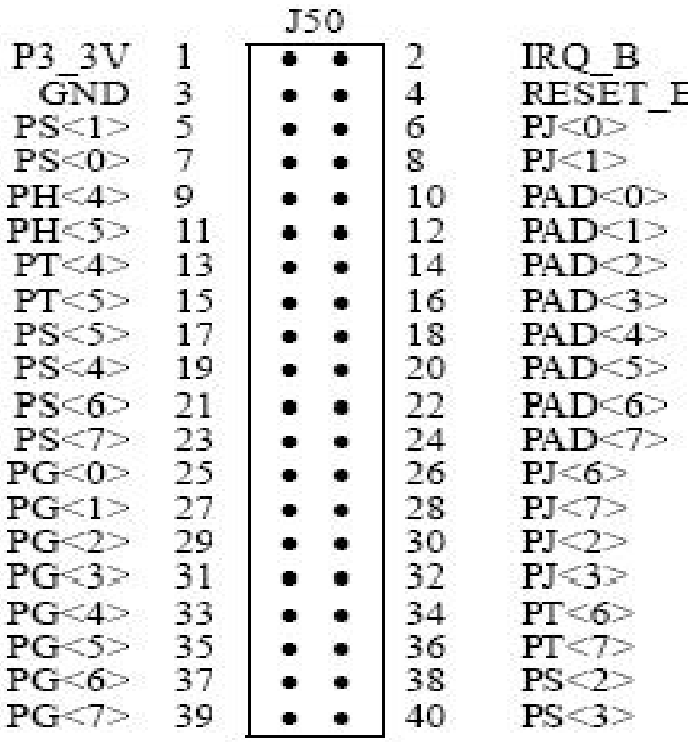


Figure A2.2 Pin connections of DEMO9S12NE64 board.

Pin	Label	Signal
1	P3_3V	3.3 VDC supplied from the DEMO9S12NE64
2	IRQ_B	IRQ_B, which is also PE1, is always an input and can always be read. This input is used for requesting an asynchronous interrupt to the MCU. When used as an interrupt pin, this signal is active-low
3	GND	GROUND
4	RESET_B	Active low bidirectional control signal that acts as an input to initialize the MCU to a known start-up state. It also acts as an open-drain output to indicate that an internal failure has been detected in either the clock monitor or COP watchdog circuit
5	PS<1>	PS1 is a general purpose input or output. When the Serial Communications Interface 0 (SCI0) transmitter is enabled the PS1 pin is configured as the transmit pin, TXD, of SCI0
6	PJ<0>	PJ0 is a general purpose I/O pin. When the EMAC MII interface is enabled it becomes the management data clock(MII_MDC) signal
7	PS<0>	PS0 is a general purpose input or output. When the Serial Communications Interface 0 (SCI0) receiver is enabled the PS0 pin is configured as the receive pin RXD0 of SCI0
8	PJ<1>	PJ1 is a general purpose I/O pin. When the EMAC MII interface is enabled it becomes the Management Data I/O (MII_MDIO) signal
9	PH<4>	PH4 is a general purpose input or output pin. When the EMAC MII interface is enabled it becomes the transmit Clock (MII_TXCLK) signal

Table A2.2 Pin functions of DEMO9S12NE64 board.

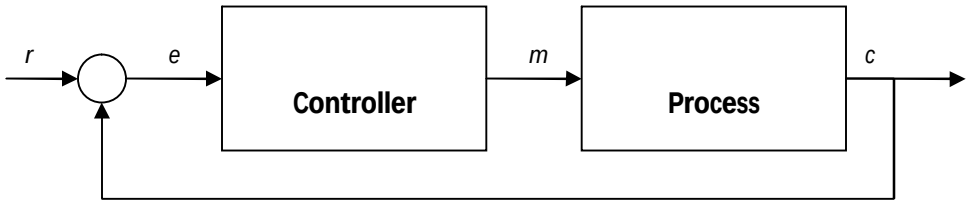
Pin	Label	Signal
10, 12, 14, 16, 18, 20, 22, 24	PAD<0> - PAD<7>	PAD[7:0] are the analog inputs for the analog to digital converter (ADC). They can also be configured as general purpose digital input
11	PH<5>	PH5 is a general purpose input or output pin. When the EMAC MII interface is enabled it becomes the transmit Enabled (MII_TXEN) signal
13, 15, 34, 36	PT<4>, PT<5>, PT<6>, PT<7>	PT[7:4] are general purpose input or output pins. When the Timer system 1 (TIM1) is enabled they can also be configured as the TIM1 input capture or output compare pins IOC1[7-4]
17	PS<5>	PS5 is a general purpose input or output pin. When the Serial Peripheral Interface (SPI) is enabled PS5 is the master output (during master mode) or slave input (during slave mode) pin (MOSI)
19	PS<4>	PS4 is a general purpose input or output pin. When the Serial Peripheral Interface (SPI) is enabled PS4 is the master input (during master mode) or slave output (during slave mode) pin (MISO)
21	PS<6>	PS6 is a general purpose input or output pin. When the Serial Peripheral Interface (SPI) is enabled PS6 becomes the serial clock pin, SCK
23	PS<7>	PS7 is a general purpose input or output. When the Serial Peripheral Interface (SPI) is enabled PS7 becomes the slave select pin SS
25	PG<0>	PG6 is a general purpose input or output pin. When the EMAC MII interface is enabled it becomes the receive data (MII_RXD0) signal
26	PJ<6>	PJ6 is a general purpose input or output pin. When the IIC module is enabled it becomes the Serial Data Line (IIC_SDL) for the IIC module (IIC)
27	PG<1>	PG1 is a general purpose input or output pin. When the EMAC MII interface is enabled it becomes the receive data (MII_RXD1) signal
28	PJ<7>	PJ7 is a general purpose input or output pin. When the IIC module is enabled it becomes the serial clock line (IIC_SCL) for the IIC module (IIC)
29	PG<2>	PG2 is a general purpose input or output pin. When the EMAC MII interface is enabled it becomes the receive data (MII_RXD2) signal
30	PJ<2>	PJ2 is a general purpose input or output pin. When the EMAC MII interface is enabled it becomes the carrier sense (MII_CRS) signal
31	PG<3>	PG3 is a general purpose input or output pin. When the EMAC MII interface is enabled it becomes the receive data (MII_RXD3) signal
32	PJ<3>	PJ3 is a general purpose input or output pin. When the EMAC MII interface is enabled it becomes the collision (MII_COL) signal
33	PG<4>	PG4 is a general purpose input or output pin. When the EMAC MII interface is enabled it becomes the receive clock (MII_RXCLK) signal
35	PG<5>	PG5 is a general purpose input or output pin. When the EMAC MII interface is enabled it becomes the receive data valid (MII_RXDV) signal
37	PG<6>	PG6 is a general purpose input or output pin. When the EMAC MII interface is enabled it becomes the receive error (MII_RXER) signal
38	PS<2>	PS2 is a general purpose input or output. When the Serial Communications Interface 1 (SCI1) receiver is enabled the PS2 pin is configured as the receive pin RXD of SCI1
39	PG<7>	PG7 is a general purpose input or output pin. It can be configured to generate an interrupt(KWG7) causing the MCU to exit STOP or WAIT mode
40	PS<3>	PS3 is a general purpose input or output. When the Serial Communications Interface 1 (SCI1) transmitter is enabled the PS3 pin is configured as the transmit pin, TXD, of SCI1

Table A2.2 Pin functions of DEMO9S12NE64 board (continued).

Pin Functions	Data Direction	Pin Name	Pin Number
Ground	-	GND	3
High	-	P3_3V	1
Demo Board Interface:			
- Run/Load Switch	Input	PG4	33
- SW1	Input	PE0	-
- SW2	Input	PH4	9
- Reset	Input	RESET_B	4
- LED1	Output	PG0	25
- LED2	Output	PG1	27
- Pot	Input	PAD0	10
Analog Outputs:			
MOSI	Output	PS5	17
_SS	Output	PS7	23
SPI Clock	Output	PS6	21
Analog Inputs:			
Analog Channel 0	Input	PAD0	10
Analog Channel 1	Input	PAD1	12
Analog Channel 2	Input	PAD2	14
Analog Channel 3	Input	PAD3	16
Analog Channel 4	Input	PAD4	18
Analog Channel 5	Input	PAD5	20
Analog Channel 6	Input	PAD6	22
Analog Channel 7	Input	PAD7	24
Digital Inputs:			
Digital Input 0	Input	PJ6	26
Digital Input 1	Input	PJ7	28
Digital Input 2	Input	PJ2	30
Digital Input 3	Input	PJ3	32
Digital Input 4	Input	PT6	34
Digital Input 5	Input	PT7	36
Digital Input 6	Input	PS2	38
Digital Input 7	Input	PJ0	6
Digital Outputs:			
Digital Output 0	Output	PG0	25
Digital Output 1	Output	PG1	27
Digital Output 2	Output	PG2	29
Digital Output 3	Output	PG3	31
Digital Output 4	Output	PS4	19
Digital Output 5	Output	PG5	35
Digital Output 6	Output	PG6	37
Digital Output 7	Output	PS1	5

Table A2.3 Pin connections of DEMO9S12NE64 board.

Appendix 3:
PID Implementation

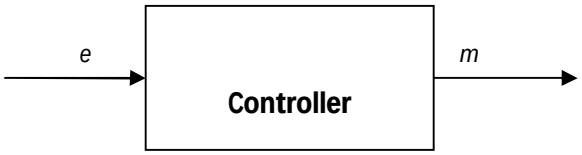


***r* = set point**

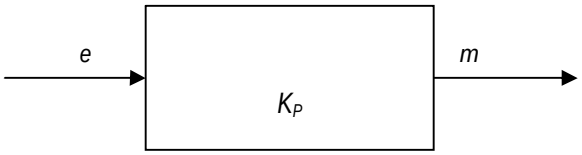
e* = error = *r* – *c

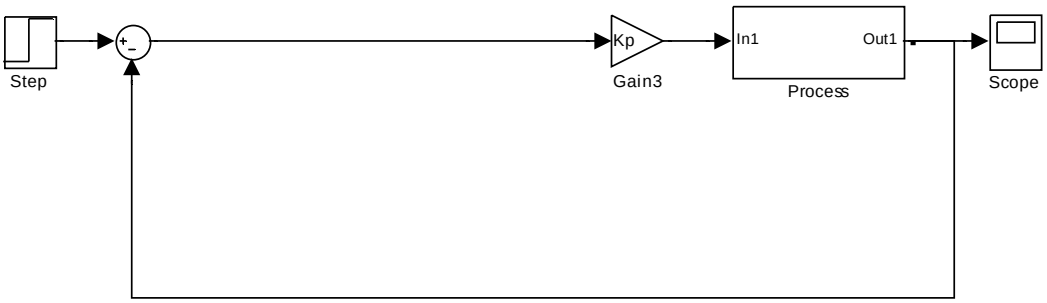
***m* = manipulated variable**

***c* = controlled variable**



P Controller



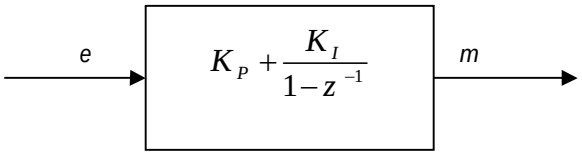


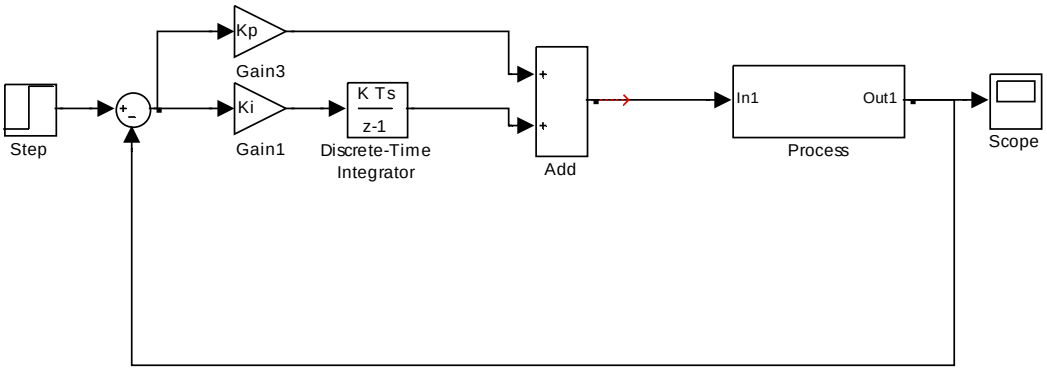
$$\frac{m}{e} = K_P$$

$$m = K_P e$$

```
c = get_c();
e = r-c;
m = Kp*e;
```

PI Controller 1





$$\frac{m}{e} = \left(K_P + \frac{K_I}{1-z^{-1}} \right)$$

$$m = \left(K_P + \frac{K_I}{1-z^{-1}} \right) e$$

$$m\left(1-z^{-1}\right) = \left(K_P\left(1-z^{-1}\right) + K_I\right) e$$

$$m - mz^{-1} = K_P e - K_P e z^{-1} + K_I e$$

$$m = K_P e - K_P e z^{-1} + K_I e + mz^{-1}$$

$$m = \left(K_P + K_I\right) e - K_P e z^{-1} + mz^{-1}$$

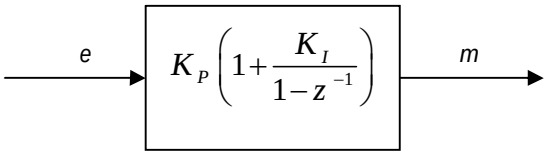
$$m(k) = \left(K_P + K_I\right) e(k) - K_P e(k-1) + m(k-1)$$

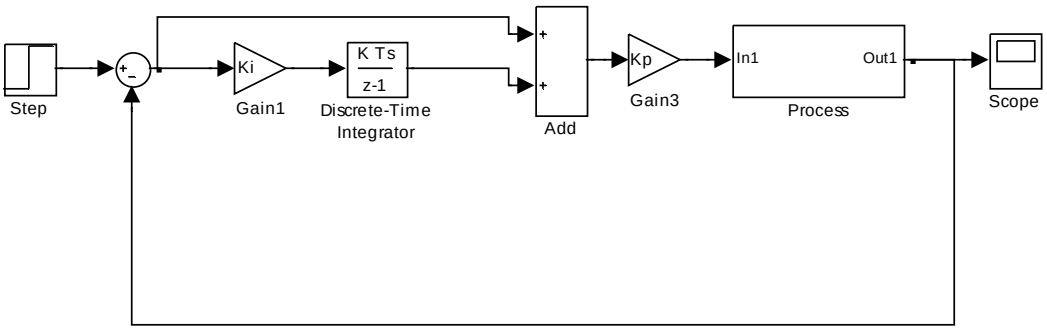
```
c = get_c();
e = r-c;
e_sum = e_sum + e;
m = Kp*e + Ki*e_sum;
```

OR

```
e_prev = e;
m_prev = m;
c = get_c();
e = r-c;
m = (Kp + Ki)*e – Kp*e_prev + m_prev;
```

PI Controller 2





$$\frac{m}{e}=K_P\left(1+\frac{K_I}{1-z^{-1}}\right)$$

$$m=K_P\left(1+\frac{K_I}{1-z^{-1}}\right)e$$

$$m\left(1-z^{-1}\right)=K_P\left(\left(1-z^{-1}\right)+K_I\right)e$$

$$m-mz^{-1}=K_Pe-K_Pez^{-1}+K_PK_Ie$$

$$m=K_Pe-K_Pez^{-1}+K_PK_Ie+mz^{-1}$$

$$m=\left(K_P+K_PK_I\right)e-K_Pez^{-1}+mz^{-1}$$

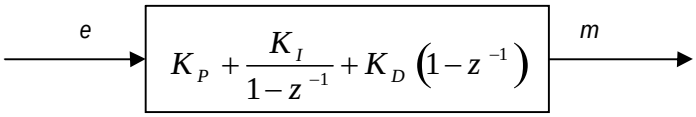
$$m(k)=\left(K_P+K_PK_I\right)e(k)-K_Pe(k-1)+m(k-1)$$

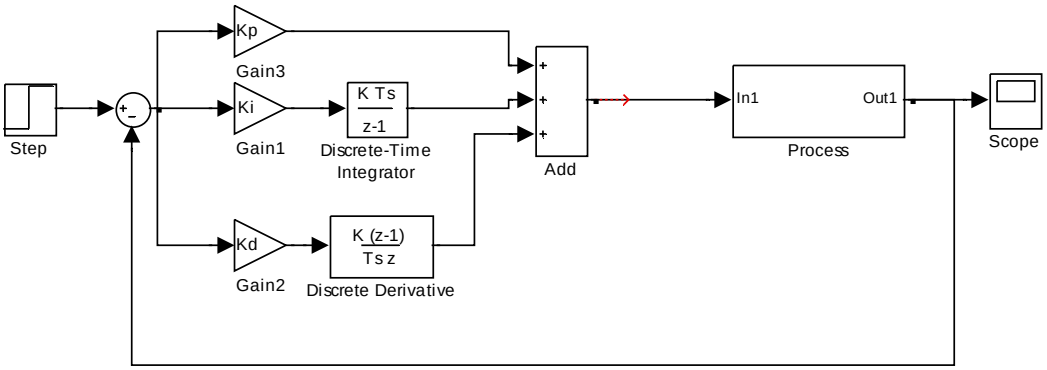
```
c = get_c();
e = r-c;
e_sum = e_sum + e;
m = Kp*(e + Ki*e_sum);
```

OR

```
e_prev = e;
m_prev = m;
c = get_c();
e = r-c;
m = (Kp + Kp*Ki)*e – Kp*e_prev + m_prev;
```

PID Controller 1





$$\frac{m}{e} = \left(K_P + \frac{K_I}{1-z^{-1}} + K_D\left(1-z^{-1}\right)\right)$$

$$m = \left(K_P + \frac{K_I}{1-z^{-1}} + K_D\left(1-z^{-1}\right)\right)e$$

$$m\left(1-z^{-1}\right) = \left(K_P\left(1-z^{-1}\right) + K_I + K_D\left(1-z^{-1}\right)^2\right)e$$

$$m - mz^{-1} = K_P e - K_P e z^{-1} + K_I e + K_D\left(1-2z^{-1}+z^{-2}\right)e$$

$$m = K_P e - K_P e z^{-1} + K_I e + mz^{-1} + K_D e - 2K_D e z^{-1} + K_D e z^{-2}$$

$$m = \left(K_P + K_I + K_D\right)e - \left(K_P + 2K_D\right)e z^{-1} + mz^{-1} + K_D e z^{-2}$$

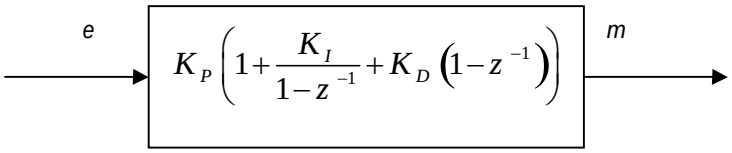
$$m(k) = \left(K_P + K_I + K_D\right)e(k) - \left(K_P + 2K_D\right)e(k-1) + m(k-1) + K_D e(k-2)$$

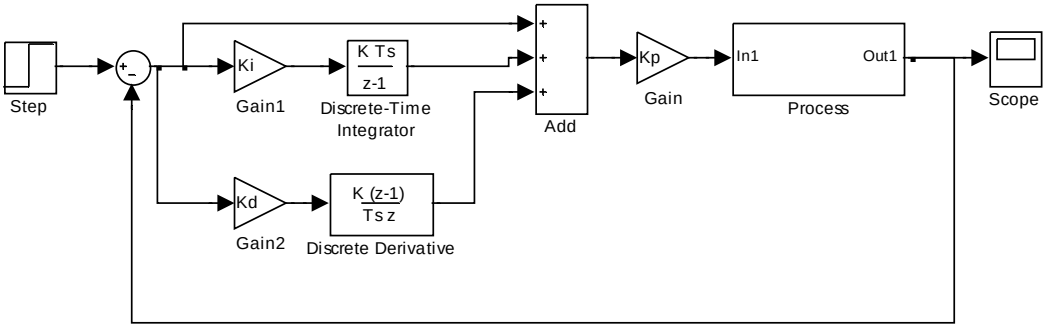
```
e_prev = e;
c = get_c();
e = r-c;
e_sum = e_sum + e;
m = Kp*e + Ki*e_sum + Kd*(e-e_prev);
```

OR

```
e_prev_prev = e_prev;
e_prev = e;
m_prev = m;
c = get_c();
e = r-c;
m = (Kp+Ki+Kd)*e–(Kp+2Kd)*e_prev+m_prev+Kd*e_prev_prev;
```

PID Controller 2





$$\frac{m}{e}=K_P\left(1+\frac{K_I}{1-z^{-1}}+K_D\left(1-z^{-1}\right)\right)$$

$$m=K_P\left(1+\frac{K_I}{1-z^{-1}}+K_D\left(1-z^{-1}\right)\right)e$$

$$m\left(1-z^{-1}\right)=K_P\left(\left(1-z^{-1}\right)+K_I+K_D\left(1-z^{-1}\right)^2\right)e$$

$$m-mz^{-1}=K_Pe-K_Pez^{-1}+K_PK_Ie+K_PK_D\left(1-2z^{-1}+z^{-2}\right)e$$

$$m=K_Pe-K_Pez^{-1}+K_PK_Ie+mz^{-1}+K_PK_De-2K_PK_Dez^{-1}+K_PK_Dez^{-2}$$

$$m=\left(K_P+K_PK_I+K_PK_D\right)e-\left(K_P+2K_PK_D\right)ez^{-1}+mz^{-1}+K_PK_Dez^{-2}$$

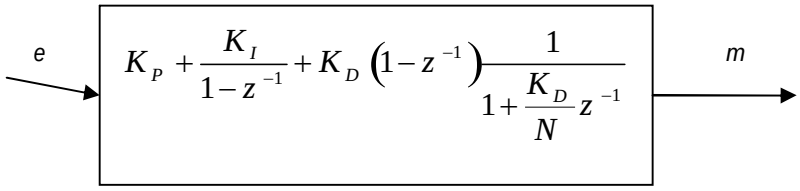
$$m(k)=\left(K_P+K_PK_I+K_PK_D\right)e(k)-\left(K_P+2K_PK_D\right)e(k-1)+m(k-1)+K_PK_De(k-2)$$

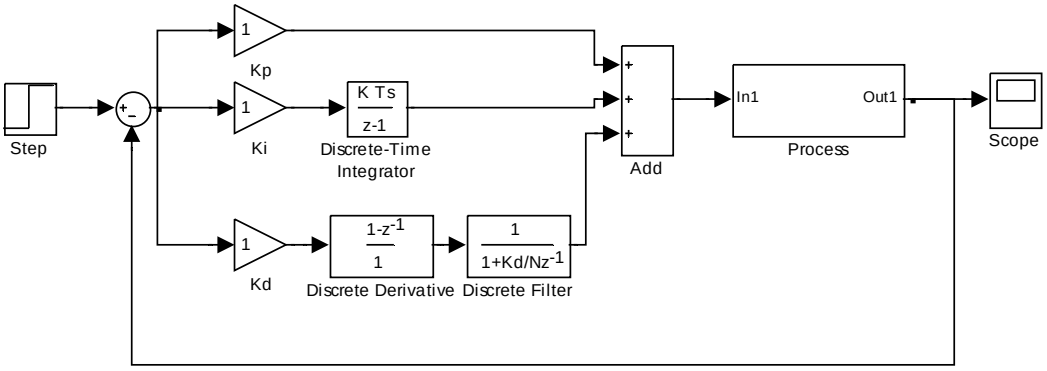
```
e_prev = e;
c = get_c();
e = r-c;
e_sum = e_sum + e;
m = Kp*(e + Ki*e_sum + Kd*(e-e_prev));
```

OR

```
e_prev_prev = e_prev;
e_prev = e;
m_prev = m;
c = get_c();
e = r-c;
m = (Kp + Kp*Ki + Kp*Kd)*e – (Kp + 2Kp*Kd)*e_prev +
      m_prev + Kp*Kd*e_prev_prev;
```

PID Controller 3





$$\frac{m}{e} = \left(K_P + \frac{K_I}{1-z^{-1}} + K_D\left(1-z^{-1}\right)\frac{1}{1+\frac{K_D}{N}z^{-1}} \right)$$

$$m = \left(K_P + \frac{K_I}{1-z^{-1}} + K_D\left(1-z^{-1}\right)\frac{1}{1+\frac{K_D}{N}z^{-1}} \right)e$$

$$m\left(1-z^{-1}\right)\left(1+\frac{K_D}{N}z^{-1}\right) = \left(K_P\left(1-z^{-1}\right)\left(1+\frac{K_D}{N}z^{-1}\right) + K_I\left(1+\frac{K_D}{N}z^{-1}\right) + K_D\left(1-z^{-1}\right)^2 \right)e$$

$$m\left(1+\frac{K_D}{N}z^{-1}-z^{-1}-\frac{K_D}{N}z^{-2}\right) = \left(K_P\left(1+\frac{K_D}{N}z^{-1}-z^{-1}-\frac{K_D}{N}z^{-2}\right) + K_I + \frac{K_I K_D}{N}z^{-1} + K_D\left(1-2z^{-1}+z^{-2}\right) \right)e$$

$$m\left(1+\left(\frac{K_D}{N}-1\right)z^{-1}-\frac{K_D}{N}z^{-2}\right) = \left(K_P + \frac{K_P K_D}{N}z^{-1} - K_P z^{-1} - \frac{K_P K_D}{N}z^{-2} + K_I + \frac{K_I K_D}{N}z^{-1} + K_D - 2K_D z^{-1} + K_D z^{-2} \right)e$$

$$m\left(1+\left(\frac{K_D}{N}-1\right)z^{-1}-\frac{K_D}{N}z^{-2}\right) = \left(\left(K_P+K_I+K_D\right) + \left(\frac{K_D}{N}\left(K_P+K_I\right) - K_P - 2K_D\right)z^{-1} + \left(K_D - \frac{K_P K_D}{N}\right)z^{-2} \right)e$$

$$m(k) + \left(\frac{K_D}{N}-1\right)m(k-1) - \frac{K_D}{N}m(k-2) = \left(K_P+K_I+K_D\right)e(k) + \left(\frac{K_D}{N}\left(K_P+K_I\right) - K_P - 2K_D\right)e(k-1) + \left(K_D - \frac{K_P K_D}{N}\right)e(k-2)$$

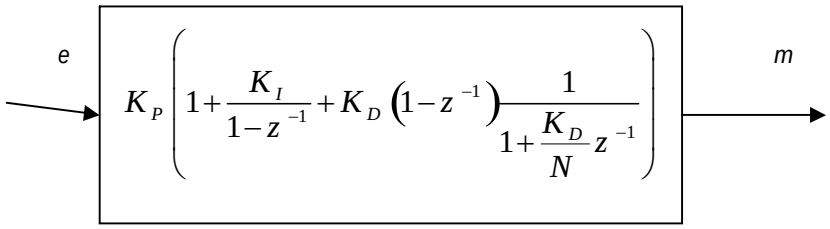
$$m(k) = \left(K_P+K_I+K_D\right)e(k) + \left(\frac{K_D}{N}\left(K_P+K_I\right) - K_P - 2K_D\right)e(k-1) + \left(K_D - \frac{K_P K_D}{N}\right)e(k-2) - \left(\frac{K_D}{N}-1\right)m(k-1) + \frac{K_D}{N}m(k-2)$$

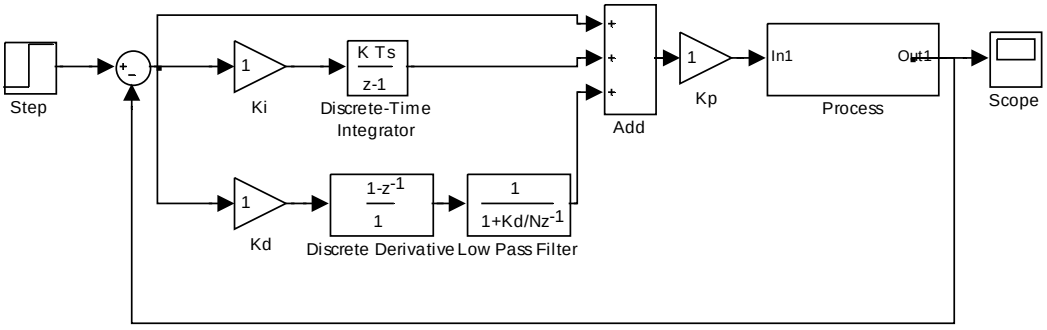
```
e_prev = e;
c = get_c();
e = r-c;
e_sum = e_sum + e;
m1_prev = m1;
m1 = Kd*(e-e_prev) – Kd/N*m1_prev;
m = Kp*e + Ki*e_sum + m1;
```

OR

```
e_prev_prev = e_prev;
e_prev = e;
m_prev_prev = m_prev;
m_prev = m;
c = get_c();
e = r-c;
m = (Kp+Ki+Kd)*e–(Kd/N*(Kp+Ki)-Kp-2Kd)*e_prev +
     (Kd-Kp*Kd/N)*e_prev_prev – (Kd/N-1)*m_prev +
Kd/N*m_prev_prev;
```

PID Controller 4





$$\frac{m}{e} = K_P\left(1+\frac{K_I}{1-z^{-1}}+K_D\left(1-z^{-1}\right)\frac{1}{1+\frac{K_D}{N}z^{-1}}\right)$$

$$m = K_P\left(1+\frac{K_I}{1-z^{-1}}+K_D\left(1-z^{-1}\right)\frac{1}{1+\frac{K_D}{N}z^{-1}}\right)e$$

$$m\left(1-z^{-1}\right)\left(1+\frac{K_D}{N}z^{-1}\right) = K_P\left(\left(1-z^{-1}\right)\left(1+\frac{K_D}{N}z^{-1}\right)+K_I\left(1+\frac{K_D}{N}z^{-1}\right)+K_D\left(1-z^{-1}\right)^2\right)e$$

$$m\left(1+\frac{K_D}{N}z^{-1}-z^{-1}-\frac{K_D}{N}z^{-2}\right) = K_P\left(\left(1+\frac{K_D}{N}z^{-1}-z^{-1}-\frac{K_D}{N}z^{-2}\right)+K_I+\frac{K_IK_D}{N}z^{-1}+K_D\left(1-2z^{-1}+z^{-2}\right)\right)e$$

$$m\left(1+\left(\frac{K_D}{N}-1\right)z^{-1}-\frac{K_D}{N}z^{-2}\right) = \left(K_P+\frac{K_PK_D}{N}z^{-1}-K_Pz^{-1}-\frac{K_PK_D}{N}z^{-2}+K_PK_I+\frac{K_PK_IK_D}{N}z^{-1}+K_PK_D-2K_PK_Dz^{-1}+K_PK_Dz^{-2}\right)e$$

$$m\left(1+\left(\frac{K_D}{N}-1\right)z^{-1}-\frac{K_D}{N}z^{-2}\right) = \left(\left(K_P+K_PK_I+K_PK_D\right)+\left(\frac{K_PK_D}{N}\left(1+K_I\right)-K_P-2K_PK_D\right)z^{-1}+\left(K_PK_D\left(1-\frac{1}{N}\right)\right)z^{-2}\right)e$$

$$m(k)+\left(\frac{K_D}{N}-1\right)m(k-1)-\frac{K_D}{N}m(k-2) = \left(K_P+K_PK_I+K_PK_D\right)e(k)+\left(\frac{K_PK_D}{N}\left(1+K_I\right)-K_P-2K_PK_D\right)e(k-1)+\left(K_PK_D\left(1-\frac{1}{N}\right)\right)e(k-2)$$

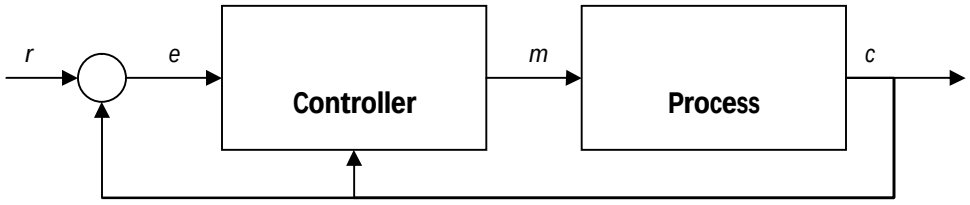
$$m(k) = \left(K_P+K_PK_I+K_PK_D\right)e(k)+\left(\frac{K_PK_D}{N}\left(1+K_I\right)-K_P-2K_PK_D\right)e(k-1)+\left(K_PK_D\left(1-\frac{1}{N}\right)\right)e(k-2)-\left(\frac{K_D}{N}-1\right)m(k-1)+\frac{K_D}{N}m(k-2)$$

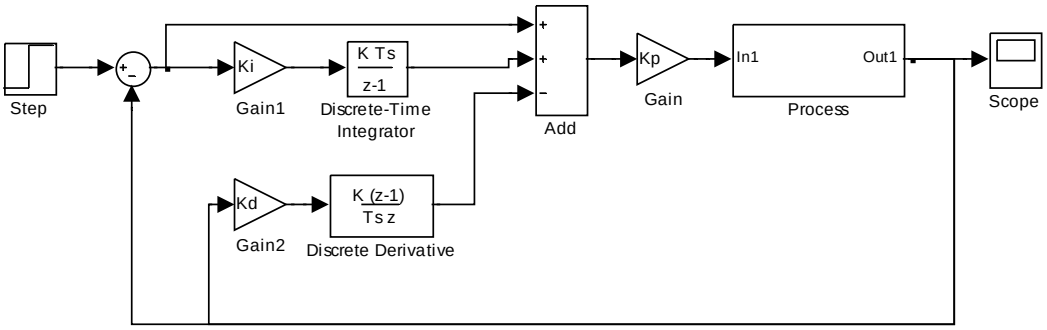
```
e_prev = e;
c = get_c();
e = r-c;
e_sum = e_sum + e;
m1_prev = m1;
m1 = Kd*(e-e_prev) – Kd/N*m1_prev;
m = Kp*(e + Ki*e_sum + m1);
```

OR

```
e_prev_prev = e_prev;
e_prev = e;
m_prev_prev = m_prev;
m_prev = m;
c = get_c();
e = r-c;
m = Kp(1+Ki+Kd)*e–Kp*(Kd/N*(1+Ki)-1-2Kd)*e_prev +
     Kp*Kd(1-1/N)*e_prev_prev – (Kd/N-1)*m_prev +
Kd/N*m_prev_prev;
```

Enhancement 1.0 : Derivative Action Suppressing for Error Step Changes





$$m = K_P\left(\left(1+\frac{K_I}{1-z^{-1}}\right)e-\left(K_D\left(1-z^{-1}\right)\right)\right)c$$

$$m\left(1-z^{-1}\right)=K_P\left(\left(1-z^{-1}+K_I\right)e-\left(K_D\left(1-z^{-1}\right)^2\right)c\right)$$

$$m-mz^{-1}=K_P\left(e-ez^{-1}+K_Ie-K_Dc\left(1-2z^{-1}+z^{-2}\right)\right)$$

$$m-mz^{-1}=K_P\left(e-ez^{-1}+K_Ie-K_D\left(c-2cz^{-1}+cz^{-2}\right)\right)$$

$$m=K_P\left(\left(1+K_I\right)e-ez^{-1}-K_D\left(c-2cz^{-1}+cz^{-2}\right)\right)+mz^{-1}$$

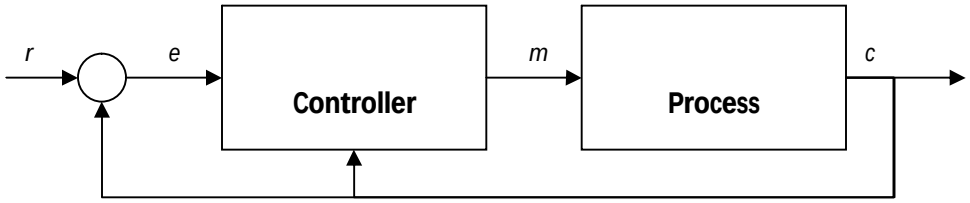
$$m(k)=K_P\left(\left(1+K_I\right)e(k)-e(k-1)-K_D\left(c(k)-2c(k-1)+c(k-2)\right)\right)+m(k-1)$$

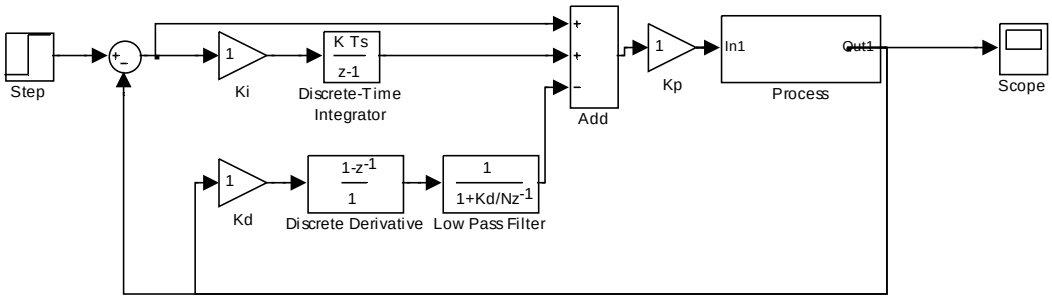
```
c_prev = c;
c = get_c();
e = r-c;
e_sum = e_sum + e;
m = Kp*(e + Ki*e_sum – Kd*(c – c_prev));
```

OR

```
c_prev_prev = c_prev;
c_prev = c;
m_prev = m;
c = get_c();
e = r-c;
m = Kp*((1+Ki)*e – e_prev – Kd*(c – 2*c_prev +
     c_prev_prev)) + m_prev;
```

Enhancement 1.1 : Derivative Action Suppressing for Error Step Changes with Low Pass Filter





$$m = K_P\left(\left(1+\frac{K_I}{1-z^{-1}}\right)e - \left(K_D\left(1-z^{-1}\right)\frac{1}{1+\frac{K_D}{N}z^{-1}}\right)\right)c$$

$$m\left(1-z^{-1}\right) = K_P\left(\left(1-z^{-1}+K_I\right)e - \left(K_D\left(1-z^{-1}\right)^2\frac{1}{1+\frac{K_D}{N}z^{-1}}\right)c\right)$$

$$m\left(1-z^{-1}\right)\left(1+\frac{K_D}{N}z^{-1}\right) = K_P\left(\left(1-z^{-1}+K_I\right)\left(1+\frac{K_D}{N}z^{-1}\right)e - \left(K_D\left(1-z^{-1}\right)^2\right)c\right)$$

$$m\left(1+\frac{K_D}{N}z^{-1}-z^{-1}-\frac{K_D}{N}z^{-2}\right) = K_P\left(\left(1+K_I+\left(K_I+\frac{K_IK_D}{N}-1\right)z^{-1}-\frac{K_D}{N}z^{-2}\right)e - K_D\left(1-2z^{-1}+z^{-2}\right)c\right)$$

$$m\left(1+\left(\frac{K_D}{N}-1\right)z^{-1}-\frac{K_D}{N}z^{-2}\right) = K_P\left(\left(\left(1+K_I\right)+\left(K_I+\frac{K_IK_D}{N}-1\right)z^{-1}-\frac{K_D}{N}z^{-2}\right)e - K_D\left(1-2z^{-1}+z^{-2}\right)c\right)$$

$$m(k)+\left(\frac{K_D}{N}-1\right)m(k-1)-\frac{K_D}{N}m(k-2) = K_P\left(1+K_I\right)e(k)+K_P\left(K_I+\frac{K_IK_D}{N}-1\right)e(k-1)-\frac{K_PK_D}{N}e(k-2)$$
$$-K_PK_Dc(k)+2K_PK_Dc(k-1)-K_PK_Dc(k-2)$$

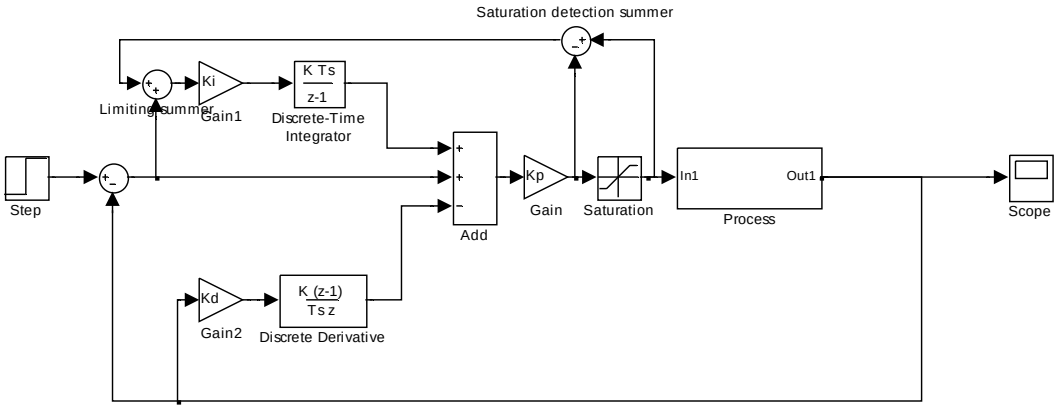
$$m(k) = K_P\left(1+K_I\right)e(k)+K_P\left(K_I+\frac{K_IK_D}{N}-1\right)e(k-1)-\frac{K_PK_D}{N}e(k-2)-K_PK_Dc(k)+2K_PK_Dc(k-1)-K_PK_Dc(k-2)$$
$$-\left(\frac{K_D}{N}-1\right)m(k-1)+\frac{K_D}{N}m(k-2)$$

```
c_prev = c;
c1_prev = c1;
c = get_c();
e = r-c;
e_sum = e_sum + e;
c1 = Kd*(e-e_prev) – Kd/N*c1_prev;
m = Kp*(e + Ki*e_sum –c1);
```

OR

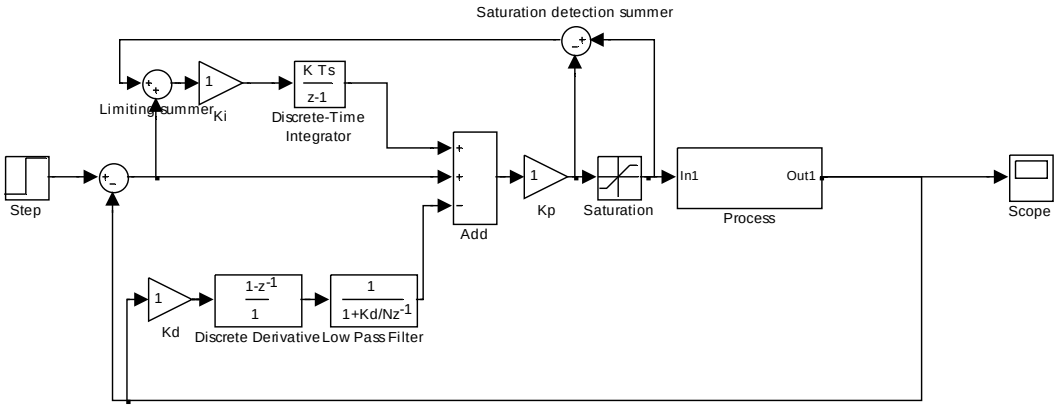
```
c_prev_prev = c_prev;
c_prev = c;
m_prev_prev = m_prev;
m_prev = m;
c = get_c();
e = r-c;

m = Kp*((1+Ki)*e + (Ki+Ki*Kd/N-1)*e_prev –
     Kp*Kd/N*e_prev_prev) – Kd*(c -2*c_prev+c_prev_prev))   -
(Kd/N-1)*m_prev + Kd/N*m_prev_prev;
```

Enhancement 2.0 : Anti-Wind Up Mechanism




```
c_prev = c;
c = get_c();
e = r-c;
e_mod = e + m_diff;
e_sum = e_sum + e_mod;
m = Kp*(e + Ki*e_sum – Kd*(c – c_prev));
m_sat = sat(m);
m_diff = m_sat – m;
```

Enhancement 2.1 : Anti-Wind Up Mechanism with Low Pass Filter



```
c_prev = c;
c1_prev = c1;
c = get_c();
e = r-c;
e_mod = e + m_diff;
e_sum = e_sum + e_mod;
c1 = Kd*(e-e_prev) – Kd/N*c1_prev;
m = Kp*(e + Ki*e_sum – c1);
m_sat = sat(m);
m_diff = m_sat – m;
```

Appendix 4: PDA

1. What is PDA?

Personal digital assistants (PDAs) are handheld computers that were originally designed as personal organizers, but became much more versatile over the years. PDAs are also known as pocket computers or palmtop computers. PDAs have many uses: calculation, use as a clock and calendar, playing computer games, accessing the Internet, sending and receiving E-mails, video recording, typewriting and word processing, use as an address book, making and writing on spreadsheets, use as a radio or stereo, recording survey responses, and Global Positioning System (GPS). Newer PDAs also have both color screens and audio capabilities, enabling them to be used as mobile phones (Smartphone), web browsers, or portable media players. Many PDAs can access the Internet, intranets or extranets via Wi-Fi, or Wireless Wide-Area Networks (WWANs).One of the most significant PDA characteristics is the presence of a touch screen.

1.1 Architecture

Many PDAs run using a variation of the **ARM architecture** . This encompasses a class of **RISC microprocessors** that are widely used in mobile devices and embedded systems, and its design was influenced strongly by a popular 1970s/1980s CPU.

1.2 Operating Systems Supported (OS)

The currently major PDA operating systems are:

- **Palm OS** - owned by PalmSource.
- **Windows Mobile (Pocket PC)** - based on the **Windows CE kernel** owned by Microsoft.
- **BlackBerry OS** - owned by Research In Motion.
- **Many operating systems based on the Linux kernel** - free (not owned by any company) These include
    - Familiar (comes in three flavours: GPE, Opie and barebone)
    - OpenZaurus (for Zaurus PDAs)
    - Intimate (for PDAs with lots and lots of memory)
- **Symbian OS** (formerly EPOC) owned by Ericsson, Motorola, Panasonic, Nokia, Samsung, Siemens and Sony Ericsson.
- **Windows Vista** - owned by Microsoft; has seldom use on PDAs.

2. Increasing Popularity

According to some modern studies, the overall market for PDAs grew by 20.7% in the third quarter of 2005, compared to the third quarter of 2004, with marketshare resolving as follows (by operating system):

- **Windows Mobile for PDAs** that comply with the Microsoft's Pocket PC specifications - 49.2% (increasing)
- **RIM BlackBerry for BlackBerry PDA** - 25.0% (increasing)
- **Symbian OS** - 5.8% (increasing)
- **Various operating systems based on the Linux kernel for various special designed PDAs** - 0.7% (stable)
- **Palm OS for Palm**- 14.9% (declining)

3. .NET Compact Framework (.NET CF)

The Microsoft **.NET Compact Framework** (.NET CF) is a version of the .NET Framework that is designed to run on mobile devices such as PDAs, mobile phones and set-top boxes. The .NET Compact Framework uses some of the same class libraries as the full .NET Framework and also a few libraries designed specifically for mobile devices such as Windows CE Input Panel.

It is possible to develop applications which use the .NET Compact Framework in Visual Studio.NET 2003 or in Visual Studio 2005, in C# or Visual Basic.NET. The resulting applications are designed to run on a special, mobile-device, high performance JIT compiler.

To be able to run applications powered by the .NET Compact Framework, the platform must support the Microsoft .NET Compact Framework runtime. Some operating systems which do include .NET CF are Windows CE 4.1 or later, Microsoft Pocket PC, Microsoft Pocket PC 2002 and Smartphone 2003. .NET Compact Framework applications don't run on the desktop computers with full .NET Framework, because, although the executable files are binary compatible, they use a different digital signature. This is used to prevent user from trying to run a full .NET application in a .NET Compact Framework device. The .NET Compact Framework is for devices that have the following characteristics:

- A capable CPU
- RAM for program and data storage
- Persistent storage, such as a disk drive or RAM disk

And this what is in our case for devices run on Windows CE 5 (WinCE 5)

3.1 .NET Compact Framework Capabilities

The .NET Compact Framework was designed to be consistent with the desktop .NET Framework. Like it or not, the desktop version always seems to be lurking in the background, like a worried parent or a homeless puppy. A crude way to compare the frameworks is by number of files and disk space occupied. The .NET Compact Framework has 18 libraries (including optional ones) and occupies about 2.5MB. The .NET Framework version 1.1, by contrast, has 86 libraries and occupies about 40MB.

3.2 Supported Runtime Elements

So what can the .NET Compact Framework library do for us? First, the .NET Compact Framework implements most of the Common Language Infrastructure (CLI), so the standard data types found on the desktop are also present for smart-device programming. At present, Microsoft supports two programming languages for the .NET Compact Framework: C# and Visual Basic .NET. This is quite a reduction from the 20 or more languages that support desktop .NET Framework. And yet, support for C# and Visual Basic .NET in the .NET Compact Framework shows that interoperability between languages is more than a promise; in the .NET Compact Framework, as on the desktop, we have direct evidence in the form of working, commercial-quality compilers that allow assemblies created in one language to freely interoperate with assemblies created in other languages.

The .NET Compact Framework supports an execution engine that provides Just in Time (JIT) compilation of IL and that verifies the type safety of managed modules before loading them. The runtime for the .NET Compact Framework provides the same memory management as on the desktop, enabling memory allocation, heap management, and garbage collection of unreachable objects.

The .NET Compact Framework supports application domains, which is a kind of lightweight process. Multiple application domains can run in a single operating system process. On some operating systems, processes have high overhead, and the cost of context switching from one process to another is high. That certainly describes processes on desktop versions of Windows, and the application domains provide a lightweight alternative.

A side effect of having lightweight processes is that there is a fairly low limit to the number of processes that can be created on a Windows CE system. That limit is 32, and it is so tightly ingrained in the memory and process architecture of Windows CE that we usually call this a hard limit, by which we mean that this limit is unlikely to change in future versions of the operating system. The operating system itself uses a few of these processes, which leaves 24 or 25 available for application programs. A .NET Compact Framework programmer might use application domains to avoid using up all the available operating system processes for large, complex applications.

The .NET Compact Framework runtime does support P/Invoke (Platform Invoke), the ability for managed code to call through to unmanaged code. There are some limitations in comparison to the desktop. One involves the complexity of the parameters that can be passed. Simple value types can be passed as parameters, as you might expect. Most reference types can also be passed as parameters, including strings, structures, and objects. But complex structures and complex objects—meaning those with embedded structures or embedded objects—cannot be marshaled automatically.

3.3 The .NET Compact Framework Provides CPU Independence

By meeting the goals for a framework with small size and good performance, the .NET Compact Framework team created a class library that is attractive to a wide range of application developers. For some developers, though, the most important feature is that .NET Compact Framework programs are CPU-independent. Five different processor families have been validated to run Windows CE: ARM, MIPS, PowerPC, SH, and x86. To build and ship native programs or libraries requires a large development and testing effort, especially when a target market needs support for several different processor families.

Things are much simpler with a .NET Compact Framework program or library. Instead of building and shipping five (or more) different versions, a single version can support all supported processors. (A single .NET Compact Framework program or library may require separate installation files, however, to accommodate CPU-specific setup programs.)

4. Windows CE Overview

As a result, the .NET Compact Framework supports all versions of the Pocket PC, including the first two generations (the Pocket PC 2000 and the Pocket PC 2002), which are built on Windows CE 3.0. This is noteworthy because, except for the Pocket PC, all Windows CE–powered platforms must run Windows CE version 4.1 or later to support the .NET Compact Framework (The third generation of Pocket PC, the Pocket PC 2003, runs this later version of the operating system.) It is worth noting that the .NET Compact Framework comes preinstalled on the PocketPC 2003 in device ROM but must be manually installed for the Pocket PC and Pocket PC 2002 devices.

To many people, Pocket PC and Windows CE are one and the same. This is due, in part, to the use of the term Pocket PC OS to refer to the software that runs a Pocket PC. Some people seem to assume that the Pocket PC is running not Windows CE but some other operating system. However, that is not the case: the Pocket PC is built on the Windows -CE operating system.

When starting to look at any technology, it helps to know what the creators of the technology had in mind during their development process. We start, then, with a discussion of the **design goals for Windows CE**. We have found that each design goal has a real impact on what an application programmer can do and that learning about these design goals helps programmers understand a wide range of choices made by the Windows CE development teams.

Design goals for Windows CE are the following aspects: Small, Modular, Portable, Compatible, Connected and Real time. Each of these aspects is needed to build the desired operating system.

5. Why .NET Compact Framework in Windows CE?

The reason is simple: Microsoft sees Windows CE–powered devices—including, of course, the Pocket PC—as part of its larger .NET strategy. Using the .NET Compact Framework, programmers can build traditional GUI applications that consume the services of Web Service servers.

With two different browsers—Pocket Internet Explorer (PIE) and Generic Internet Explorer (GENIE)—Windows CE–powered devices can also run Web Forms applications. An add-on to the ASP.NET Web server, ASP.NET Mobile Controls, allows for server-side customizations to Web-based applications for small-screen devices.

The .NET Compact Framework team adopted the following design goals for the .NET Compact Framework:

- Build on the benefits of the .NET Framework
- Maintain consistency with the desktop version.
- Ensure that the framework runs well on mobile and embedded devices.
- Expose the richness of target platforms.
- Preserve the look and feel of platforms.
- Provide portability to other mobile operating systems.

Some PDAs figures are shown at Fig .1

Fig .1 Three of devices running Windows CE derived operating systems: O2 XDA , iPaq H3870 and Sendo Z100.

6. PDA Sockets Programming

Network programming in WinCE is possible with sockets. A socket is like a handle to a file. Socket programming resembles the file IO as does the Serial Communication. You can use sockets programming to have two applications communicate with each other. The application is typically on the different computers but they can be on same computer. Sockets are a communication mechanism. A socket is normally identified by a small integer which may be called the socket descriptor. The socket mechanism was first introduced in the 4.2 BSD Unix systems in 1983. This stated that:

A **socket** is defined by a group of four numbers, these are

- The remote host identification number or address
- The remote host port number
- The local host identification number or address
- The local host port number

For any communication mechanism or socket mechanism the concept of protocol appears .A protocol is a set of rules that governs how information is delivered. For example, to use the telephone network, the protocol is to pick up the phone, listen for a dial tone, dial a number having a specific number of digits, wait for the phone to ring, and say hello. In radio, the station

uses amplitude or frequency modulation with a specific carrier frequency and transmission bandwidth, and you know to turn on the radio and tune in the station. In technical terms, no one protocol or set of protocols can be used for any communication situation. Be that as it may, communication engineers have found that a common thread runs through the *organization* of the various protocols. This grand design of information transmission organization runs through all modern networks today.

7. Protocol Architecture

What has been defined as a networking standard is a layered, hierarchical protocol organization. As shown in Fig.8, protocols are organized by function and level of detail.

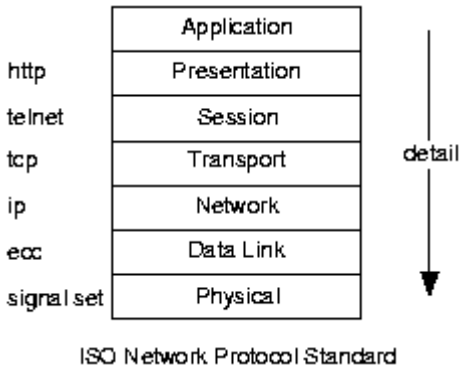


Figure .2: Protocols are organized according to the level of detail required for information transmission. Protocols at the lower levels (shown toward the bottom) concern reliable bit transmission. Higher level protocols concern how bits are organized to represent information, what kind of information is defined by bit sequences, what software needs the information, and how the information is to be interpreted. Bodies such as the IEEE (Institute for Electronics and Electrical Engineers) and the ISO (International Standards Organization) define standards such as this. Despite being a standard, it does not constrain protocol implementation so much that innovation and competitive individuality are ruled out.