2DT903 Embedded Systems

Course Introduction

5.0

Hemant Ghayvat

Department of Computer Science and Media Technology
hemant.ghayvat@lnu.se

Course information

Formal aspects

Course language is English only

• You must register to the course in order to get any grades

• For registration issues, please contact:

– Eva Puschl at eva.puschl@lnu.se

• Grading scale: A-F (F = fail)

• All materials, information and deadlines are reported on MyMoodle, including details of retake options.

• You are responsible for checking that page regularly.

Exam pattern and other details

- The exam would online and verbal exam.
- Exam time: 15 mins

Teaching staff

• Course coordinator and Teacher:

– Hemant Ghayvat, Hemant.ghayvat@lnu.se

• Teacher:

– Mehdi Saman Azari, mehdi.samanazari@lnu.se

Teaching staff and communication rules

- Please ask any (not personal) questions in course forum on MyMoodle, Slack or both and avoid using direct emails
- Before asking any questions, please look for answers within course materials on the MyMoodle course page
- If you need to send an email, please clearly specify the following info:
  - Email subject: course code and motivation/issue
  - E-mail body:
    - who you are
    - which course you are talking about
    - clear explanation of your issue/need

Course Information

- Course syllabus
  - https://kursplan.lnu.se/kursplaner/syllabus-1DT302-1.pdf
- Prerequisites –

  Knowledge about fundamentals of computer science, computer architecture / technology and progr amming languages (eg, 1DV004, 7.5 credits + 1DV507, 7.5 credits + 1DT301, 7.5 credits or equiv alent).
- Course objectives

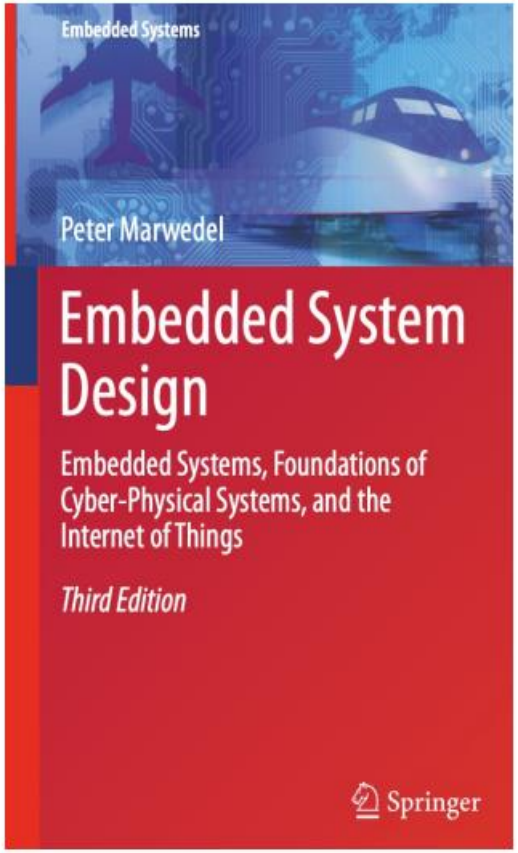
Introduction to Embedded System, Categories, Requirements, Applications, Challenges and Issues. Core of Embedded system, Memory, Sensors and Actuators, communication interface (such as RF Radios), Embedded firmware, system components (conditioning circuits).(Hemant)

- ❖ Fundamental issues of hardware software co-design, computational models in embedded Design data flow graph, control flow graph, state machine model, sequential programmed model, concurrent model, unified modeling language. .(Hemant)
- ❖ describe and explain architectures and programming languages for embedded

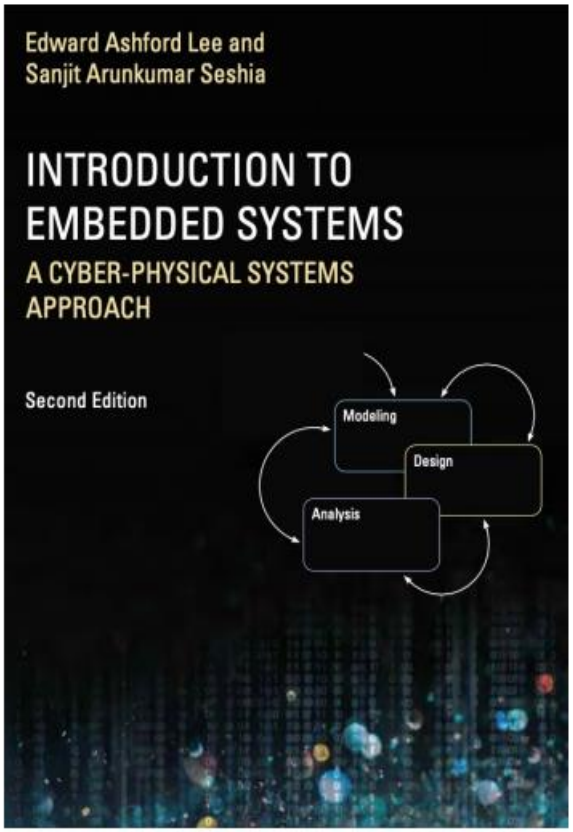
systems (Mehdi)

- ❖ use the main realtime scheduling algorithms (Fredrik)
- ❖ model concurrency in embedded systems (Fredrik)
- ❖ describe and explain the main standards and certification activities for embedded system (Hemant)

Reference material



Peter Marwedel, **Embedded System Design Embedded Systems**, Foundations of Cyber-Physical Systems and the Internet of Things, Third Edition (2018)



E. A. Lee and S. A. Seshia, **Introduction to Embedded Systems - A Cyber-Physical Systems Approach**, Second Edition, MIT Press, 2017.

Course Assignerats

- ADC of Raspberry Pi Pico using Micro-Python.
- Perform multithreading using dual-core programming on Raspberry Pi Pico
- Real-time Hardware Project with RTOS

Tentative Course plan

- The schedule may be subject to changes !!
- Any update will be notified on the MyMoodle course page
- Please refer to TimeEdit for additional available timeslots, that might be used as a backup if needed.

Embedded Systems

What we know about ES?

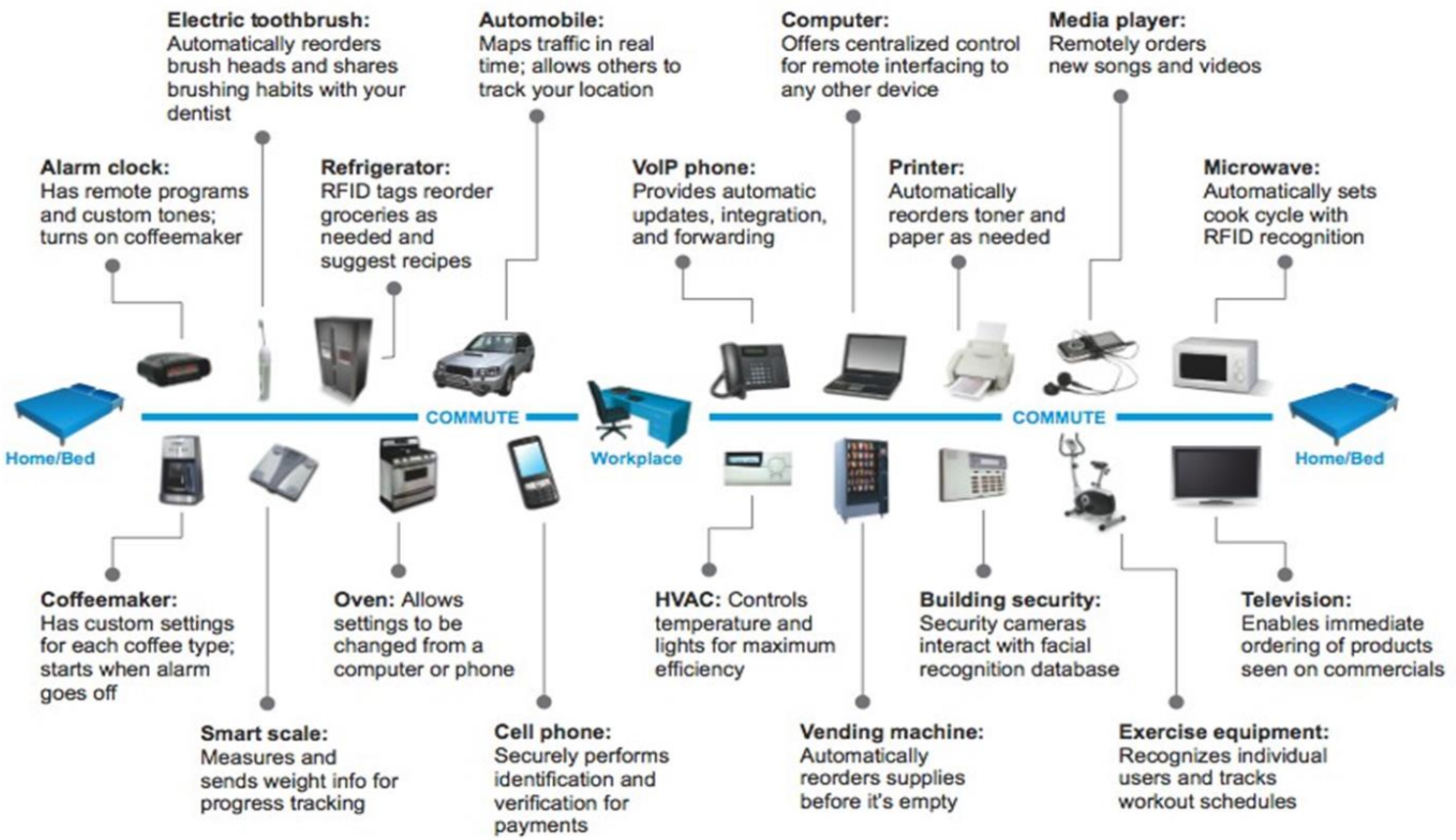
Embedded Systems

An embedded system is an arrangement of computer hardware and software planned for a particular function. Embedded systems may also function within a larger system.

Examples of embedded systems:

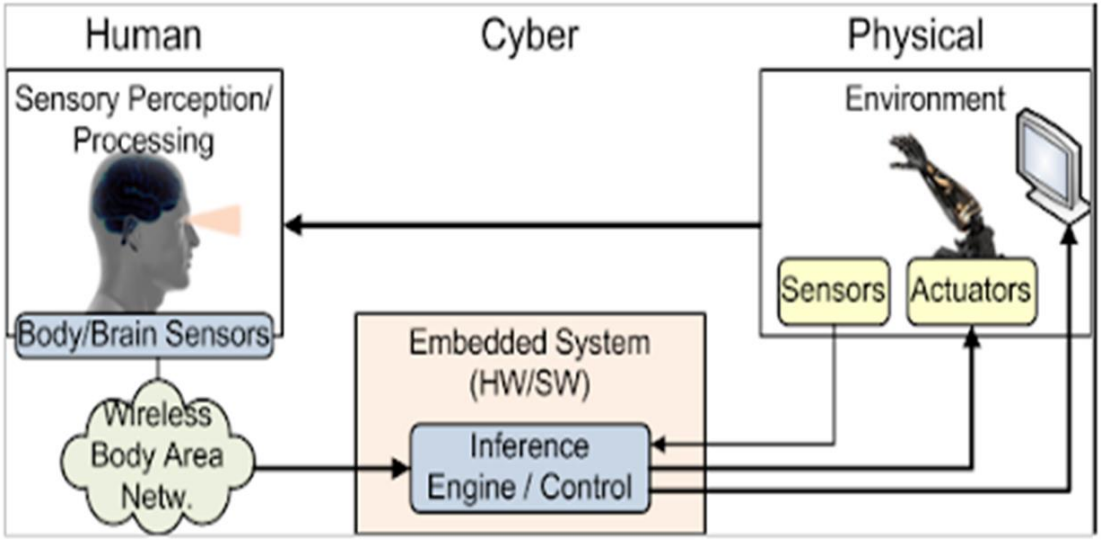
- Health Informatics Devices
- Industrial Informatics and Control Devices
- Mobile Phones
- Automobile

Embedded System



Cyber Physical System

Cyber-Physical Systems (CPS) are integrations of computation with physical Processes; to address the technical problem is managing time and concurrency in computational systems [1].

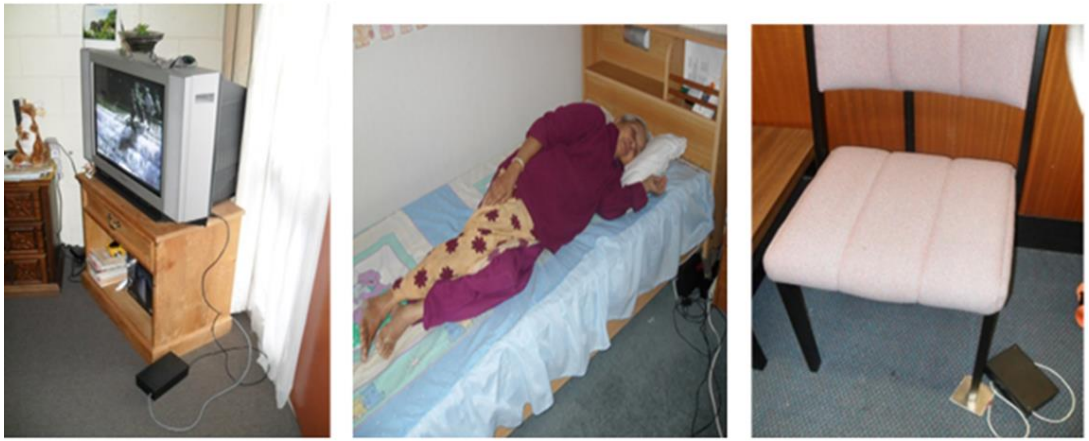


Human in the loop with CPS

IoT

Internet of things is the network of the embedded system devices (such as sensors, actuators) connected to the internet.

Example: Designing a Smart Home for wellbeing monitoring



Example: Designing a Smart Home for wellbeing monitoring

The owner of a home asks you to design a system that will allow to monitor one's wellbeing and generate their behavioral pattern.

What can go wrong? What are the possible failure points?

- The sensor get faulty or get offline or disconnected from power supply
- Someone remotely get unauthorized access of the system
- Data analytics algorithm give faulty output

What is the cost of a potential failure?

- Loss of data is loss privacy
- Threat to life
- Loss of reputation for us
- Lawsuit against us

Example: Designing a Smart Home for wellbeing monitoring

- Determine what happens if some get the unauthorized access
- Resolve the cyber security issue
- Determine how exactly the data analytics algorithm fails
- Make additional ML or DL algorithm to address it
- Test the system again and again to upgrade it
- Cope with broken links you didn’t find or that break

General Purpose Computing System Vs ES

General Purpose Computing System	Embedded System
combination of a generic hardware and a General Purpose Operating System for Generalised Task	special purpose hardware and embedded OS for the specific set of applications
Contains a General Purpose Operating System (GPOS)	May or may not contain an operating system for functioning
Applications are alterable (programmable) by the user	The firmware of the embedded system is pre-programmed and it is non-alterable by the end-user
Performance is the key deciding factor in the selection of the system. Always, 'Faster is Better'	Application-specific requirements (Safety critical system vs normal ES)
Not in resource constraint Environment	Highly resource constraint
Response requirements are not time-critical	For certain category of embedded systems like mission critical systems, the response time requirement is highly critical (safety critical ES)

Classification of Embedded Systems

Some of the criteria used in the classification of embedded systems are:

1. Based on generation
2. Complexity and performance requirements
3. Based on deterministic behaviour
4. Based on triggering

Classification Based on Generation

- First Generation
- Second Generation
- Third Generation
- Fourth Generation
- Next Generation

Classification Based on Generation (continued)

- First Generation
  - Early embedded systems were built around 8-bit microprocessors like 8085 and Z80 and 4-bit microcontrollers.
  - Simple in hardware circuits with firmware developed in assembly code.
  - **E.g.**: Digital telephone keypads, stepper motor control units, etc.

Classification Based on Generation (continued)

- Second Generation
  - Embedded systems built around 16-bit microprocessors and 8-bit or 16-bit microcontrollers.
  - Instruction set were much **more complex and powerful** than the first generation.
  - Some of the second generation embedded systems contained embedded operating systems for their operation.
  - **E.g.**: Data acquisition systems, SCADA systems, etc.

Classification Based on Generation (continued)

- Third Generation
  - Embedded systems built around 32-bit microprocessors and 16-bit microcontrollers.
  - Application and domain specific processors/controllers like Digital Signal Processors (DSP) and Application Specific Integrated Circuits (ASICs) came into picture.
  - The instruction set of processors became **more complex and powerful** and the concept of **instruction pipelining** also evolved.
  - Dedicated embedded real time and general purpose operating systems entered into the embedded market.
  - Embedded systems spread its ground to areas **like** robotics, media, industrial process control, networking, etc.

Classification Based on Generation (continued)

- Fourth Generation
  - The advent of **System on Chips (SoC), reconfigurable processors and multicore processors** are bringing high performance, tight integration and miniaturisation into the embedded device market.
  - The **SoC technique implements a total system** on a chip by implementing different functionalities with a processor core on an integrated circuit.
  - They make use of high performance real time embedded operating systems for their functioning.
  - **E.g.**: Smart phone devices, Mobile Internet Devices (MIDs), etc.

Classification Based on Generation (continued)

- Next Generation
  - The processor and embedded market is highly dynamic and demanding.
  - The next generation embedded systems are expected to meet growing demands in the market.
  - Eg: Cyber Physical system, Digital Twin, Cyber Twin

Classification Based on Complexity and Performance

- Small-Scale Embedded Systems
- Medium-Scale Embedded Systems
- Large-Scale Embedded Systems/Complex Systems

Classification Based on Complexity and Performance (continued)

- Small-Scale Embedded Systems
  - Simple in application needs and the performance requirements are not time critical.
  - **E.g.**: An electronic toy
  - Usually built around low performance and low cost 8-bit or 16-bit microprocessors/microcontrollers.
  - May or may not contain an operating system for its functioning.

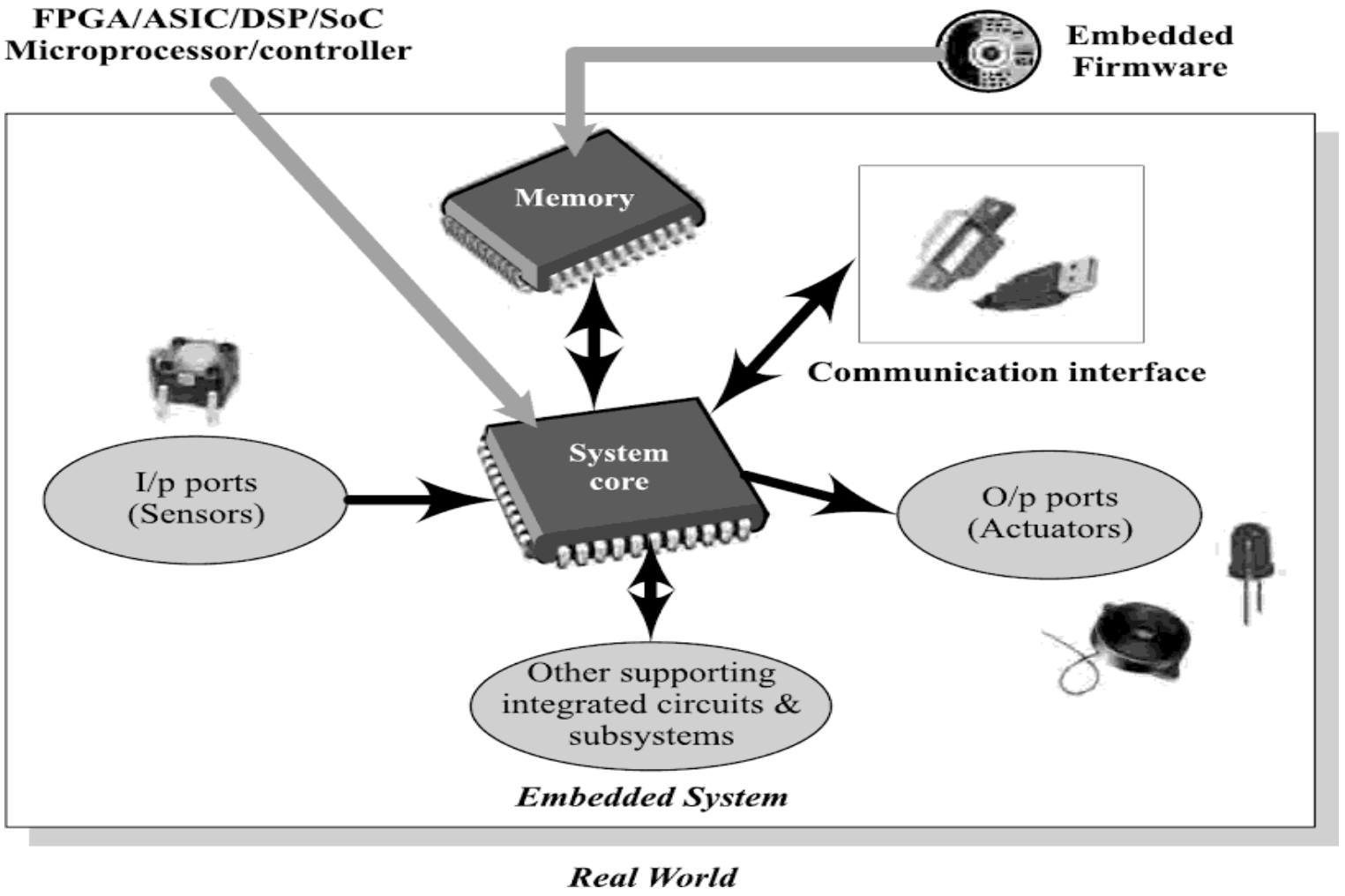
Classification Based on Complexity and Performance (continued)

- Medium-Scale Embedded Systems
  - Slightly complex in hardware and firmware (software) requirements.
  - Usually built around medium performance, low cost 16-bit or 32-bit microprocessors/microcontrollers or digital signal processors.
  - Usually contain an embedded operating system (either general purpose or real time operating system) for functioning.
  - Example: smart home monitoring and automation

Classification Based on Complexity and Performance (continued)

- Large-Scale Embedded Systems/Complex Systems
  - Highly complex in hardware and firmware (software) requirements.
  - They are employed in **mission critical applications demanding high performance.**
  - Usually built around high performance 32-bit or 64-bit RISC processors/controllers or Reconfigurable System on Chip (RSoC) or multi-core processors and programmable logic devices.
  - May contain multiple processors/controllers and co-units/hardware accelerators for offloading the processing requirements from the main processor of the system.
  - **Decoding/encoding of media, cryptographic function implementation, etc. are examples** of processing requirements which can be implemented using a co-processor/hardware accelerator.
  - Usually contain a high performance real time operating system (RTOS) for task scheduling, prioritization and management.

The Typical Embedded System



Core of the Embedded System

- Embedded systems are domain and application specific and are built around a central core.
- The core of the embedded system falls into any one of the following categories:
  1. General Purpose and Domain Specific Processors
     1. Microprocessors
     2. Microcontrollers
     3. Digital Signal Processors
  2. Application Specific Integrated Circuits (ASICs)
  3. Programmable Logic Devices (PLDs)
  4. Commercial off-the-shelf Components (COTS)

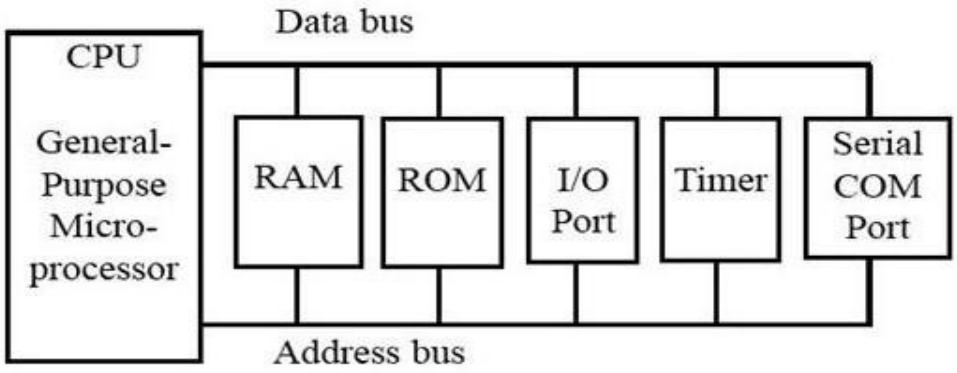
Microprocessor vs. Microcontroller

- A Microprocessor is a silicon chip representing a central processing unit (**CPU**), which is capable of **performing arithmetic** as well as logical operations according to a **pre-defined set of instructions**.
- A Microcontroller is a **highly integrated chip that contains a CPU**, scratch pad **RAM**, special and general purpose **register arrays**, on chip **ROM**/FLASH **memory** for program storage, timer and interrupt control units and dedicated **I/O ports**.

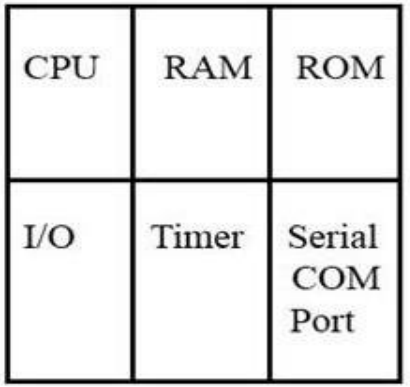
Microprocessor vs. Microcontroller (continued)

Microprocessor	Microcontroller
A silicon chip representing a central processing unit (CPU)	A microcontroller is a highly integrated chip that contains a CPU, scratchpad RAM, register arrays, on chip ROM/ FLASH memory, timer and interrupt and dedicated I/O ports
It is a dependent unit. It requires the combination of other chips like timers, program and data memory chips, interrupt controllers, etc. for functioning	It is a self-contained unit and it doesn't require external interrupt controller, timer, UART, etc. for its functioning
Most of the time, general purpose in design and operation	Mostly application-oriented or domain-specific
Doesn't contain a built in I/O port.	multiple built-in I/O ports
Targeted for high end market where performance is important	Targeted for embedded market where performance is not so critical
Limited power saving options	power saving features

Microprocessor vs. Microcontroller (continued)



Microprocessor-based system



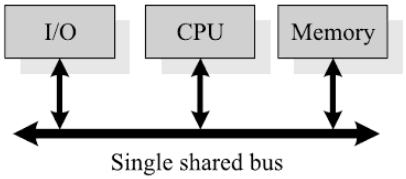
Microcontroller

RISC vs CISC

- RISC stands for Reduced Instruction Set Computing.
  - All RISC processors/controllers possess lesser number of instructions, typically in the range of 30 to 40.
  - E.g.: Atmel AVR microcontroller – its instruction set contains only 32 instructions.
- CISC stands for Complex Instruction Set Computing.
  - The instruction set is complex and instructions are high in number.
  - E.g.: 8051 microcontroller – its instruction set contains 255 instructions.

Harvard vs. Von-Neumann Processor/ControlIer Architecture (continued)

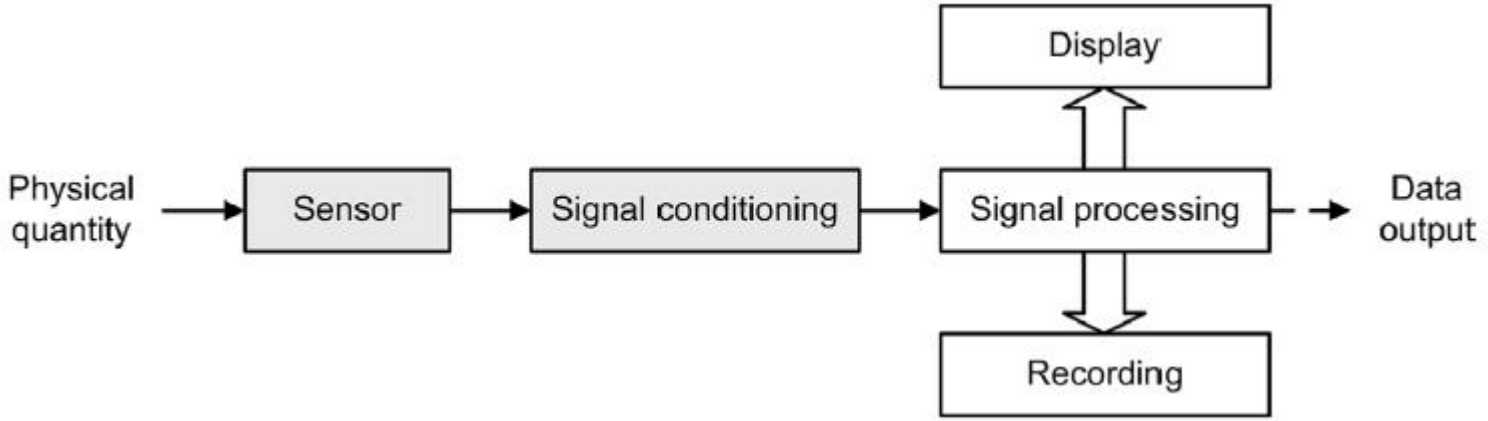
Harvard Architecture	Von-Neumann Architecture
Separate buses for instruction and data fetching	Single shared bus for instruction and data fetching
Easier to pipeline, so high performance can be achieved	Low performance compared to Harvard architecture
Comparatively high cost	Cheaper
No memory alignment problems	Allows self modifying codes
Since data memory and program memory are stored physically in different locations, no chances for accidental corruption of program memory	Since data memory and program memory are stored physically in the same chip, chances for accidental corruption of program memory



Sensors and Actuators

- A sensor is a transducer device that converts energy from one form to another for any **measurement** or control purpose. (more or less it is a measurement device such as a scale)
  - E.g.: Temperature sensor, magnetic hall effect sensor, humidity sensor, etc.
- An actuator is a form of transducer device (mechanical or electrical) which converts signals to corresponding **physical action** (motion).
  - Actuator acts as an output device.
  - E.g.: Stepper motor

Sensor and Interfacing module



- Fig. shows the arrangement of an instrumentation system for the sensor linked with the conditioning Circuit

Why conditioning circuit needed ?

- The physical quantity to be measured (e.g. temperature) acts upon a sensor that produces an electrical output signal.
- This signal is an electrical analogue of the physical input but there may not be a linear relationship between the physical quantity and its electrical equivalent.
- Because of this and since the output produced by the sensor may be small or may suffer from **the presence of noise** (i.e. unwanted signals), further **signal conditioning will be required** before the signal will be at an acceptable level and in an acceptable form for **signal processing**, display and recording.
- Furthermore, because the signal processing may use digital rather than analogue signals an additional stage of analogue-to-digital conversion may be required.

Instrumentation and Control Systems (continued)

- Fig. shows the arrangement of a control system.

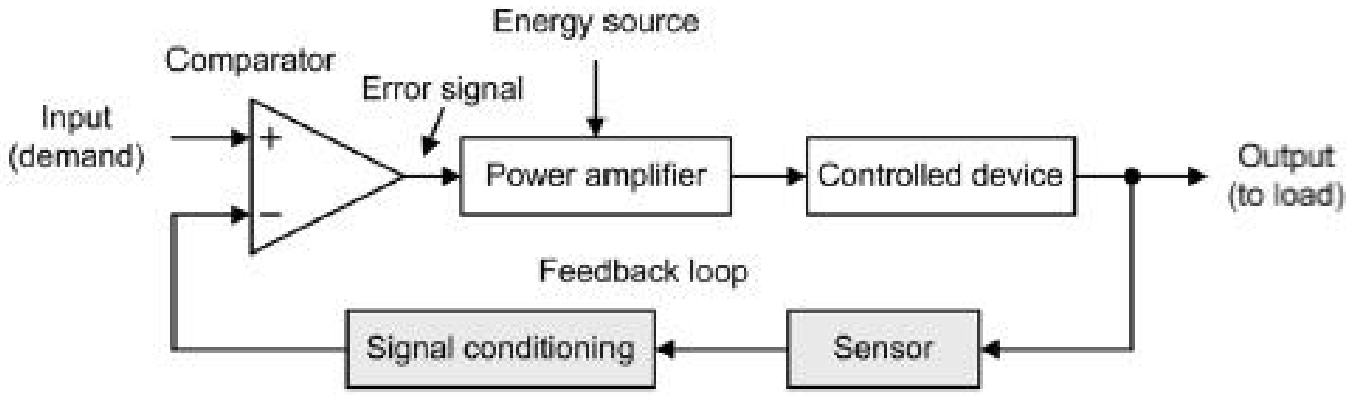


Fig: A Control System

Communication Interface

Communication Interface

- Why do we need ?
- Two types:
- Onboard Communication Interface (Device/board level communication

interface): An **onboard communication interface** is used to facilitate communication **within the same circuit board** (or system). It's typically used for **communication between components** that are physically part of the same system or located close to each other on the same PCB (Printed Circuit Board). E.g.: Serial interfaces like I2C, SPI, UART, 1-Wire, etc and parallel bus interface.

- External Communication Interface (Product level communication interface): An **external communication interface** is used to facilitate communication between an embedded system and **external devices or systems**

E.g.: Wireless interfaces like Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves (RF), GPRS, etc. and wired interfaces like RS-232C/RS-422/RS-485, USB, Ethernet IEEE 1394 port, Parallel port, CF-II interface, SDIO, PCMCIA/PCIex, etc

Lnu.se