Unit-I: Introduction

- Introduction: Definition of Embedded Systems, Typical examples, and Application domains (Automotive, Consumer, etc), Characteristics, Typical block diagram, Input, Core, Output, Commercial Off the Shelf Components (COTS). Processing Components, Microprocessors & Microcontrollers, Indicative Examples (Microcontrollers on Arduino boards), Development boards (Arduino boards), Concepts and brief introduction to Memory, Interrupts, Power Supply, Clocks, Reset. Case Studies: Washing Machine, Antilock Brake Systems (Block diagram & Working Principle).

Definition of Embedded Systems

- Embedded Systems: An embedded system is a combination of computer hardware and software designed for a specific function.
- Embedded systems may also function within a larger system. The systems can be programmable or have a fixed functionality.
- They are dedicated to a specific function or set of functions.
- Embedded systems have limited resources (memory, processing power) and are often real-time systems.

Embedded System Block Diagram

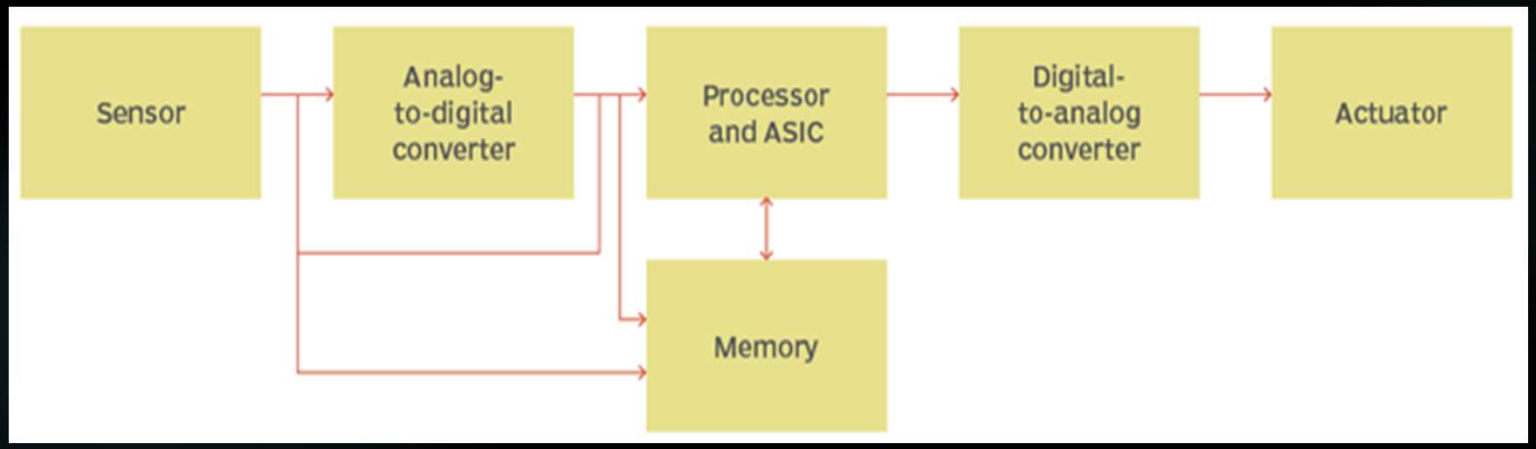


Figure1.1: A diagram of the basic structure and flow of information in embedded systems.

How does an embedded system work?

- Embedded systems always function as part of a complete device -- that's what's meant by the term embedded.
- Generally, they comprise a processor, power supply, and memory and communication ports. Embedded systems use the communication ports to transmit data between the processor and peripheral devices.
- The processor interprets this data with the help of minimal software stored on the memory.
- Often, embedded systems are used in real-time operating environments and use a real-time operating system (RTOS) to communicate with the hardware.

Structure of embedded systems

Embedded systems vary in complexity but, generally, consist of three main elements:

- **Hardware:** The hardware of embedded systems is based around microprocessors and microcontrollers. Microprocessors are very similar to microcontrollers and, typically, refer to a CPU (central processing unit) that is integrated with other basic computing components such as memory chips and digital signal processors (DSPs). Microcontrollers have those components built into one chip.
- **Software and firmware:** Software for embedded systems can vary in complexity. However, industrial-grade microcontrollers and embedded IoT systems usually run very simple software that requires little memory.
- **Real-time operating system:** These are not always included in embedded systems, especially smaller-scale systems. RTOSes define how the system works by supervising the software and setting rules during program execution.

Typical Examples of Embedded Systems
(But not limited to these)

- **Smartphones:**
  - Incorporate various embedded systems like the processor, memory, and sensors.
  - Perform functions such as communication, multimedia, and application execution.
- **Automotive Systems:**
  - Engine control units (ECUs) regulate fuel injection, ignition timing, and emissions.
  - Anti-lock braking systems (ABS) provide improved vehicle control during braking.
- **Consumer Electronics:**
  - Digital cameras, MP3 players, and smart TVs utilize embedded systems for image processing, audio playback, and content streaming.
- **Medical Devices:**
  - Implantable pacemakers and insulin pumps provide life-saving functionality.
  - Medical monitoring devices track vital signs and provide real-time feedback.

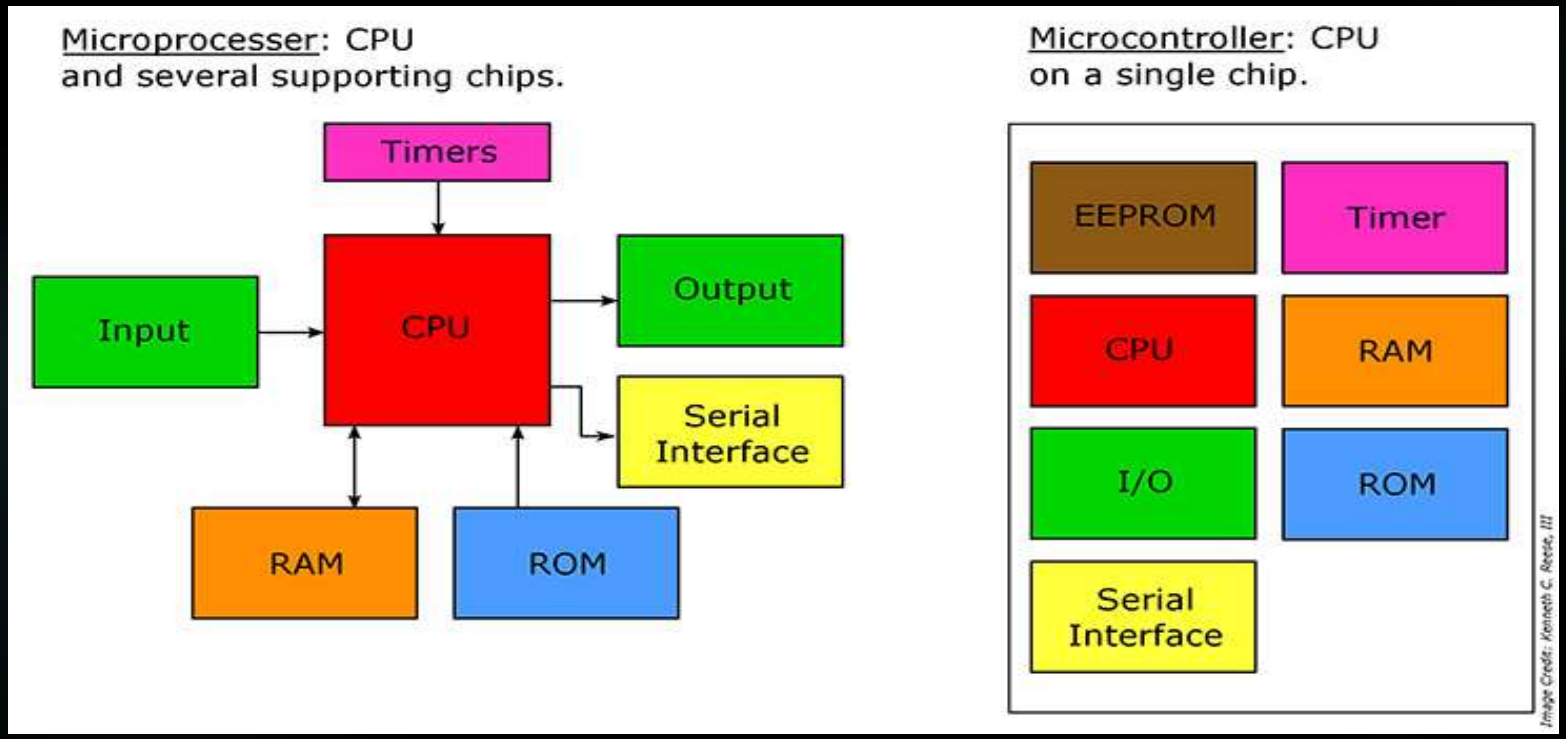
Application Domains of Embedded Systems **(But not limited to these)**

- **Automotive:**
  - Engine management systems
  - Infotainment systems
  - Advanced driver-assistance systems (ADAS)
  - Connected car technologies
- **Consumer Electronics**:
  - Smart home automation
  - Wearable devices
  - Gaming consoles
  - Home entertainment systems
- **Industrial Automation:**
  - Programmable logic controllers (PLCs)
  - Robotics systems
  - Process control systems
  - Machine vision systems
- **Healthcare:**
  - Medical imaging devices
  - Patient monitoring systems
  - Drug delivery systems
  - Prosthetic devices

Conclusion

- Embedded systems are specialized computer systems designed for specific tasks within larger systems or devices.
- They have limited resources and are often real-time systems.
- Typical examples include smartphones, automotive systems, consumer electronics, and medical devices.
- Embedded systems find applications in various domains such as automotive, consumer electronics, industrial automation, and healthcare.

Microprocessor and Microcontroller



Microprocessor and Microcontroller

Microprocessor

- Intel Core i7
- AMD Ryzen
- Qualcomm Snapdragon
- IBM POWER9
- ARM Cortex-A

Microcontroller

- Arduino
- Raspberry Pi
- Texas Instruments MSP430
- Atmel AVR
- STMicroelectronics STM32

Difference between Microcontroller & Microprocessor

Microcontroller

- Application-specific
- Integrated components (CPU, memory, I/O)
- On-chip memory (ROM, RAM)
- Reduced Instruction Set (RISC)
- Lower clock speeds
- Lower power consumption
- Cost-effective
- Embedded systems, control applications
- Robotics, automotive systems, home appliances

Microprocessor

- General-purpose
- CPU-focused
- External memory
- Complex Instruction Set (CISC)
- Higher clock speeds
- High computational power
- Expensive
- Wide range of applications
- Personal computers, servers

Commercial Off the Shelf Components (COTS)

- Commercial Off-the-Shelf (COTS) components refer to ready-made, pre-built products that are readily available in the market and not specifically developed for a particular customer or application.
- These components are mass-produced by manufacturers and sold to a wide range of customers for various purposes.
- examples of Commercial Off-the-Shelf (COTS) components:
  - Computer Processors
  - Memory Modules
  - Display Panels
  - Microcontrollers
  - Sensors and not limited to these.

Microcontrollers on Arduino boards

- Microcontrollers on Arduino boards play a central role in enabling the functionality and versatility of Arduino-based projects.
- Arduino boards are designed to provide an accessible platform for both beginners and experienced users to develop interactive electronic projects.

Key features and examples of microcontrollers used in Arduino boards

ATmega Series Microcontrollers:

The majority of Arduino boards are based on microcontrollers from the ATmega series, developed by Atmel (now Microchip Technology). Some common examples include:

- **ATmega328P:** This is the microcontroller used in Arduino Uno, one of the most widely used Arduino boards. It offers 32KB of flash memory, 2KB of SRAM, and 1KB of EEPROM.
- **ATmega2560:** Arduino Mega 2560 utilizes this microcontroller, which provides more memory and I/O pins compared to Arduino Uno. It has 256KB of flash memory, 8KB of SRAM, and 4KB of EEPROM.
- **ATmega32U4:** The Arduino Leonardo and Arduino Micro boards feature this microcontroller. It includes 32KB of flash memory, 2.5KB of SRAM, and 1KB of EEPROM. The ATmega32U4 also has built-in USB support, allowing the board to appear as a virtual keyboard or mouse when connected to a computer.

Key features and examples of microcontrollers used in Arduino boards

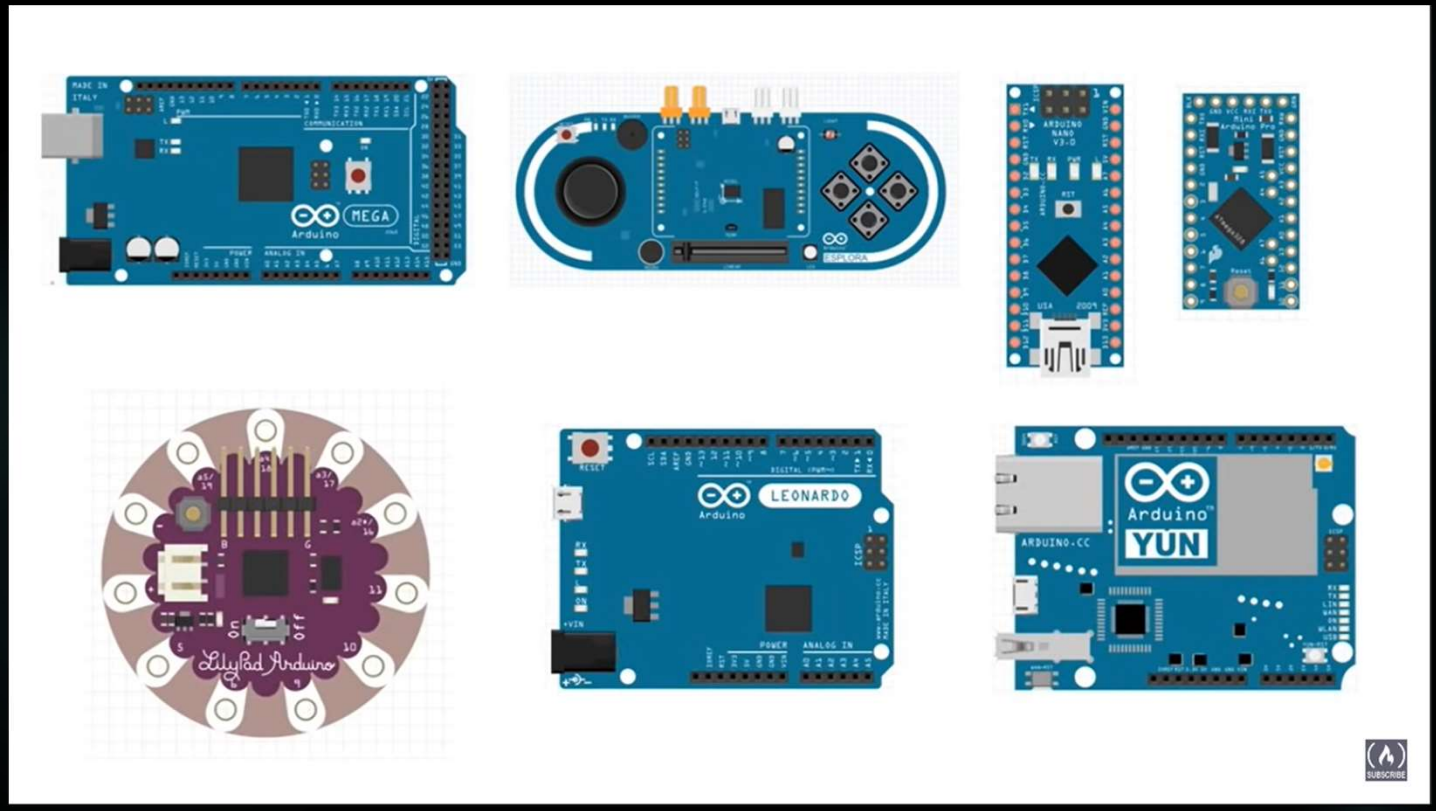
ARM-based Microcontrollers:

Arduino has also introduced boards based on ARM-based microcontrollers, offering more processing power and advanced features. Some examples include:

- **Arduino Due:** The Arduino Due is based on the Atmel SAM3X8E microcontroller, which is powered by an ARM Cortex-M3 processor. It provides 512KB of flash memory, 96KB of SRAM, and various advanced peripherals.

- **Arduino MKR Family:** Arduino MKR boards, such as MKR1000, MKR WiFi 1010, or MKR Zero, are based on various ARM Cortex-M0 or Cortex-M4 microcontrollers. These boards offer different combinations of memory, connectivity options, and power-saving features, catering to diverse IoT and wireless communication applications.

Version of Arduino



Concepts of Memory, Interrupts, Power Supply, Clocks, Reset.

These concepts form the foundation of embedded systems and are essential for understanding and developing efficient and reliable embedded applications.

Memory

Memory in embedded systems refers to the storage used to store program instructions and data. There are typically two types of memory used:

- **ROM (Read-Only Memory):** ROM contains non-volatile data, such as firmware or fixed program instructions that are permanently stored and cannot be modified during runtime.
- **RAM (Random Access Memory):** RAM is volatile memory that allows read and write operations during runtime. It is used to store data and variables that can be accessed by the processor.

Memory plays a critical role in executing instructions and storing temporary or permanent data in embedded systems.

Interrupts

- Interrupts are signals generated by external events or internal conditions that pause the normal execution of a program.
- When an interrupt occurs, the processor temporarily suspends the current task to handle the interrupt request.
- Interrupts are used to respond to time-sensitive events, such as sensor inputs, communication requests, or hardware errors.
- Interrupts ensure that critical events are processed promptly and allow the processor to efficiently handle multiple tasks concurrently.

Power Supply

- Power supply is the provision of electrical energy required to operate an embedded system.
- Embedded systems typically require a stable and reliable power source to ensure proper functioning.
- The power supply may come from various sources, including batteries, AC mains, or power adapters.
- Power management techniques are often implemented to optimize power consumption and extend the battery life in battery-powered devices.

Clocks

- Clocks provide a regular and synchronized timing signal to the processor and other components in an embedded system.
- The clock signal determines the pace at which instructions are executed, data is processed, and peripherals operate.
- Clocks ensure proper coordination and synchronization of different components, allowing the system to function accurately and reliably.

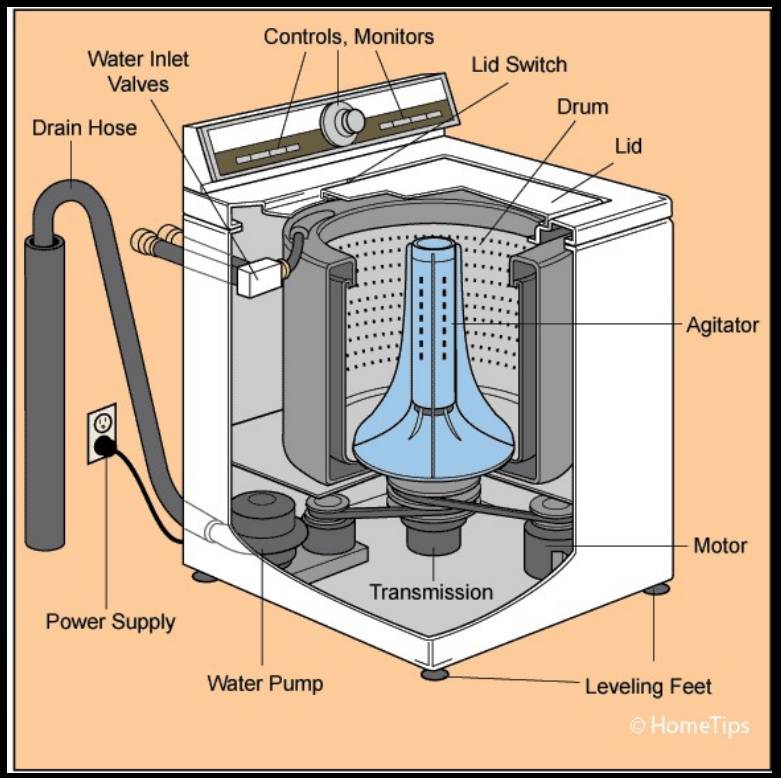
Reset

- Reset is the process of restarting an embedded system or restoring it to its initial state.
- A reset can be triggered by various events, such as power-on, manual reset button press, or a signal from a watchdog timer.
- When a reset occurs, the system restarts, and the processor initializes itself, including clearing memory, resetting peripheral devices, and executing startup routines.
- Reset ensures a clean and predictable state of the system, enabling proper initialization and preventing any lingering effects from previous operations.

Case Studies: Washing Machine, Antilock Brake Systems

These case studies demonstrate the block diagrams and working principles of a washing machine and antilock brake systems, highlighting the key components and operations involved in their functioning.

washing machine



washing machine

The embedded system components and the microcontroller play a crucial role in controlling and coordinating the operations of the washing machine, enabling efficient and automated washing processes.

- **Program Selection:** The user selects a wash program using the control panel. The microcontroller receives the input and processes it.
- **Sensor Monitoring:** The microcontroller continuously monitors the sensor inputs, such as water level sensors, temperature sensors, and rotational speed sensors. It uses this information to ensure proper control and synchronization of the washing process.
- **Control Algorithm Execution:** Based on the selected program and sensor inputs, the microcontroller executes control algorithms to regulate the motor speed, water inlet valve, and other components. It ensures the right amount of water is added, the drum rotates at the desired speed, and the temperature is maintained as per the program.

washing machine (Cont)

- **User Feedback and Display:** The microcontroller communicates with the user interface to provide feedback on the current stage of the wash cycle, display program options, and indicate any errors or notifications.
- **Safety Features:** The microcontroller incorporates safety features, such as detecting abnormal sensor readings, monitoring water overflow, and ensuring proper door lock status, to ensure safe and reliable operation.
- **Power Management:** The microcontroller manages power consumption by controlling the operation of various components, optimizing energy usage, and putting the system into low-power modes when not in use.