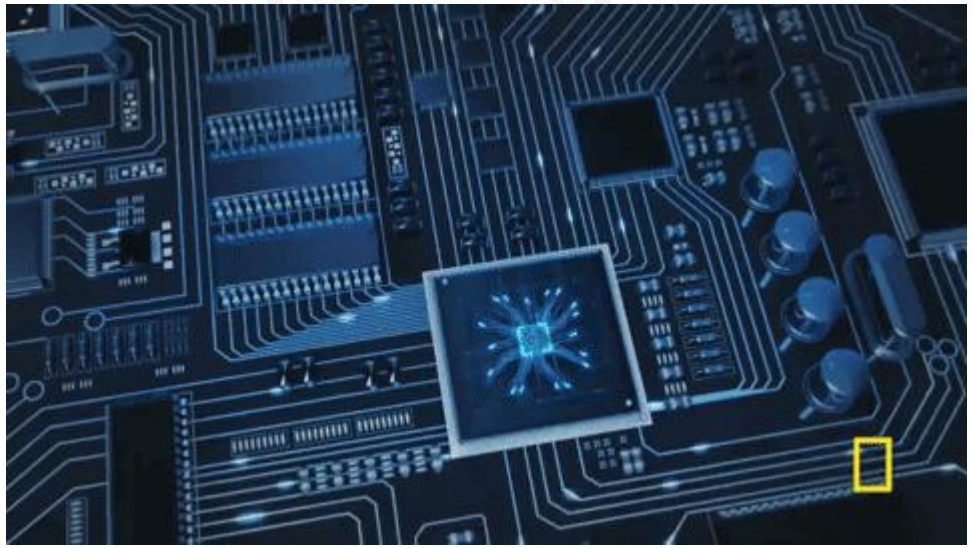
ATME College of Engineering

Department of EEE
Emitting Elite Energy

BEE613B
Embedded Systems Design
Module-1: Introduction



Presented by,

Mr.Shreeshayana R

Assistant Professor

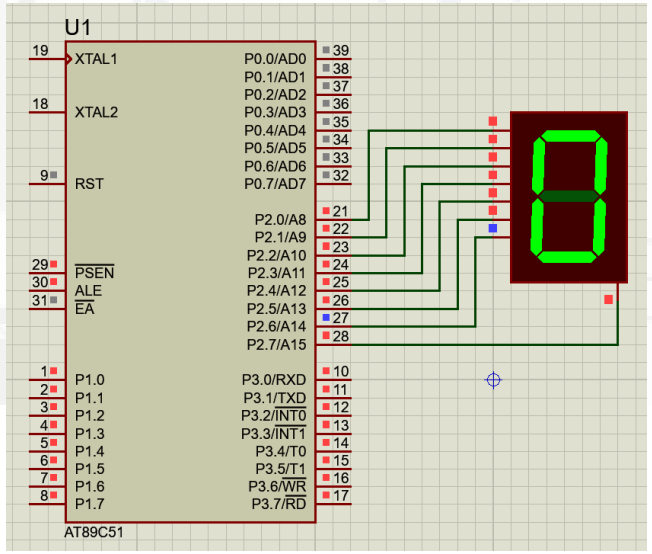
Electrical and Electronics Engineering

ATME College of Engineering, Mysuru

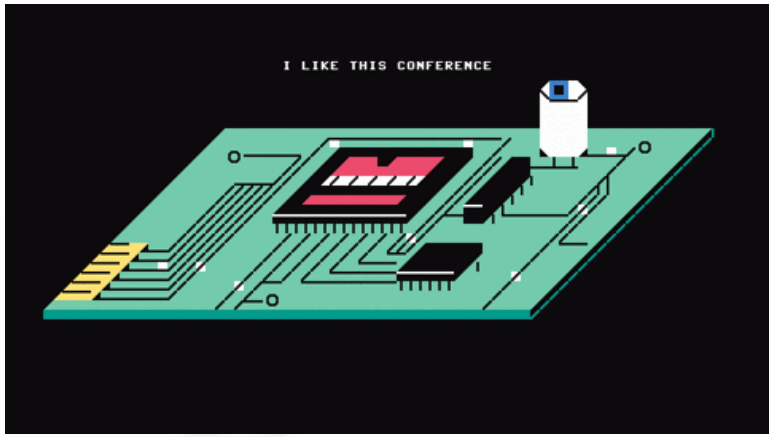
ATME College of Engineering

Department of EEE
Emitting Elite Energy

SESSION-1



Course Overview



- **Course Code**: BEE613B
- **Course Title**: Embedded Systems Design
- **Type:** Professional Elective
- **Prerequisite**: C Programming, Microcontrollers Fundamentals
- **Contact Hours**: 40 Hours

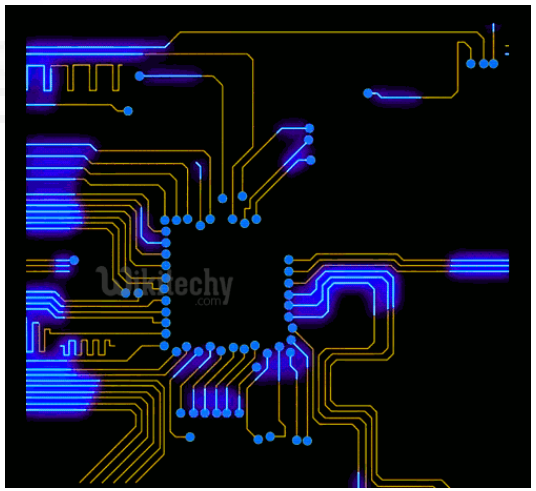
Course Objectives

1. Introduction to **Embedded System Design**
2. **Characteristics & attributes** of Embedded Systems
3. **Overview** of Embedded System Applications
4. **Software & Hardware development** for RTOS-based Embedded Systems

Module 1: Introduction to Embedded Systems

- - History, Classifications & Applications
- - Core Components: Microprocessors, Microcontrollers, ASICs, Sensors, Actuators
- - Communication Interfaces, Embedded Firmware, PCB Design

Delivery Plan: Week-1 to Week-3



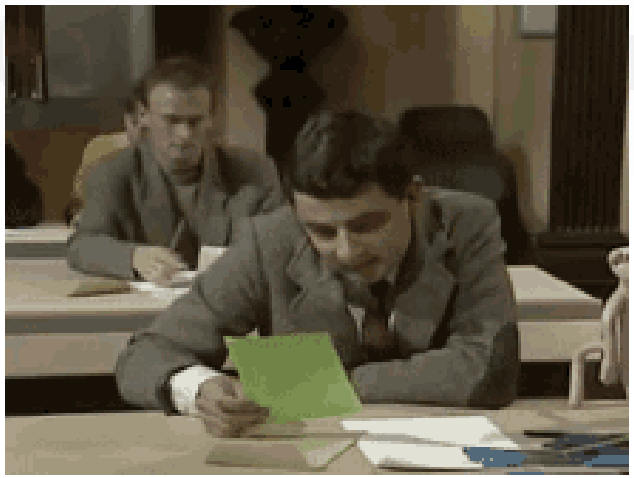
Module 2: Characteristics & Quality Attributes

- - Operational & Non-Operational Quality Attributes
- - Application-Specific Embedded Systems (e.g., Washing Machine, Automotive)

Delivery Plan: Week-4 to Week-5

IA

IA	Module	COs
IA-1	Module-1	CO-1
	Module-2	CO-2
IA-2	Module-3	CO-3
	Module-4	CO-4
IA-3	Module-5	CO-5
	Module-4	CO-4



Module 3: Hardware-Software Co-design & Development

- - Computational Models in Embedded Systems
- - Embedded Hardware: Analog/Digital Components, VLSI Design, EDA Tools

Module 4: Embedded Firmware & Development Environments

- - Firmware Design Approaches & Development Languages
- - Cross Compilation, Debugging Tools: Simulators, Emulators, Disassemblers

Module 5: RTOS-Based Embedded System Design

- - Operating System Basics, Multitasking & Scheduling
- - Tasks, Processes, and Threads in Embedded Systems

Textbooks & References

1. Shibu K V, “Introduction to Embedded Systems”, McGraw Hill

2. NPTEL: IIT Delhi (Prof. Santanu Chaudhary)

3. Arduino & Raspberry Pi Documentation

4. Vijay Madisetti, "Internet of Things: A Hands-on Approach"

5. Raj Kamal, "Internet of Things: Architecture and Design"

Activity-Based Learning

- - Design a simple Embedded System (e.g., Remote Control)
- - Microcontroller Experiments: LED, LCD, DAC Interfacing

As per VTU

Course Outcomes & Bloom's Taxonomy

CO-1. Explain Embedded System characteristics (L2)

CO-2. Understand circuit emulators & debugging (L2)

CO-3. Analyze hardware & software requirements (L3)

CO-4. Develop programming skills for Embedded Applications (L4)

CO-5. Design Embedded Systems for real-time applications (L4)

Assessment Structure (CIE & SEE)

- - CIE: 50% (Assignments: 25 marks, Tests: 25 marks)
- - SEE: 50% (University Exam, 3-hour duration)
- - Minimum passing marks: 40% (40 out of 100)

ATME College of Engineering

Department of EEE
Emitting Elite Energy

Module-1
Introduction

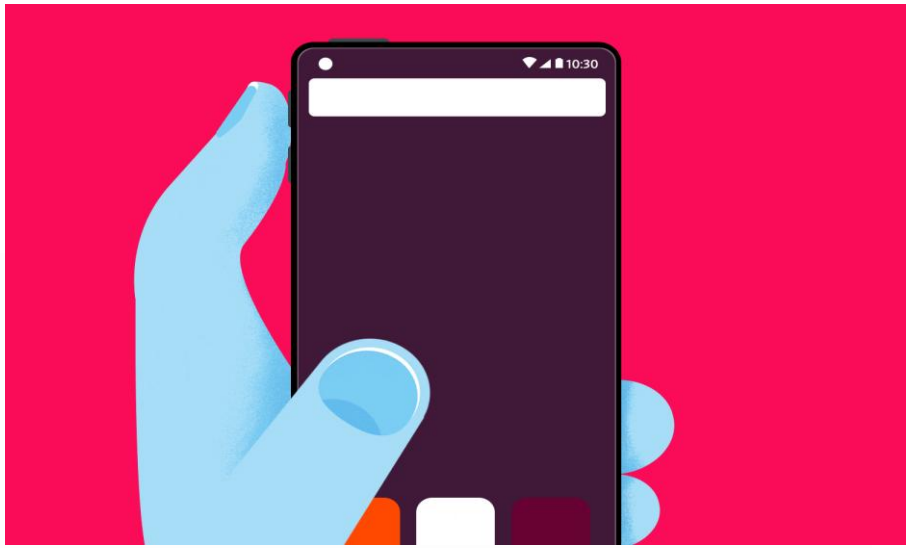
SYLLABUS

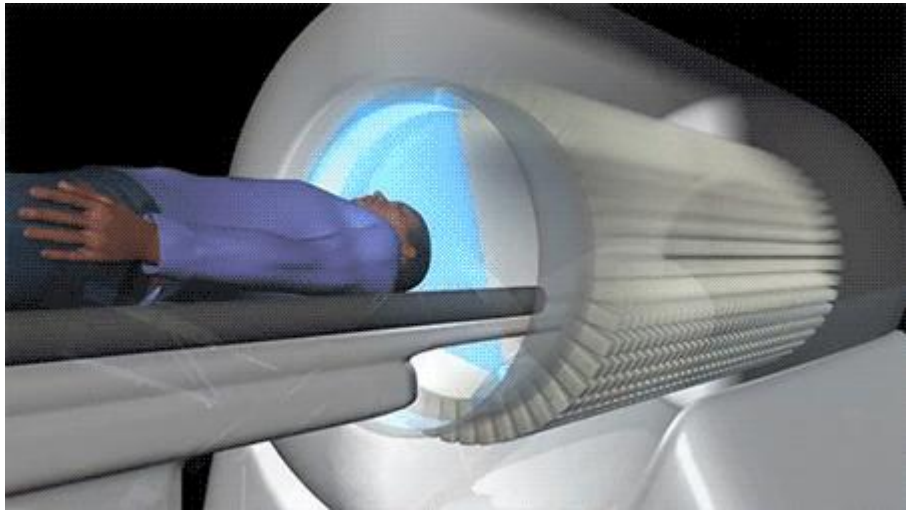
1.1 Introduction: Embedded Systems and general purpose computer systems,
1.2 History, classifications, applications and purpose of embedded systems)
1.3 Core of Embedded Systems : Microprocessors and microcontrollers,
1.4 RISC and CISC controllers,
1.5 Big endian and Little-endian processors,
1.6 Application specific ICs,
1.7 Programmable logic devices,
1.8 COTS, sensors and actuators,
1.9 Communication interface, Embedded firmware,
1.10 other system components, PCB and passive components

Comparison Between Embedded Systems and General Purpose Computing Systems

Embedded System????


MICROWAVE


10:30



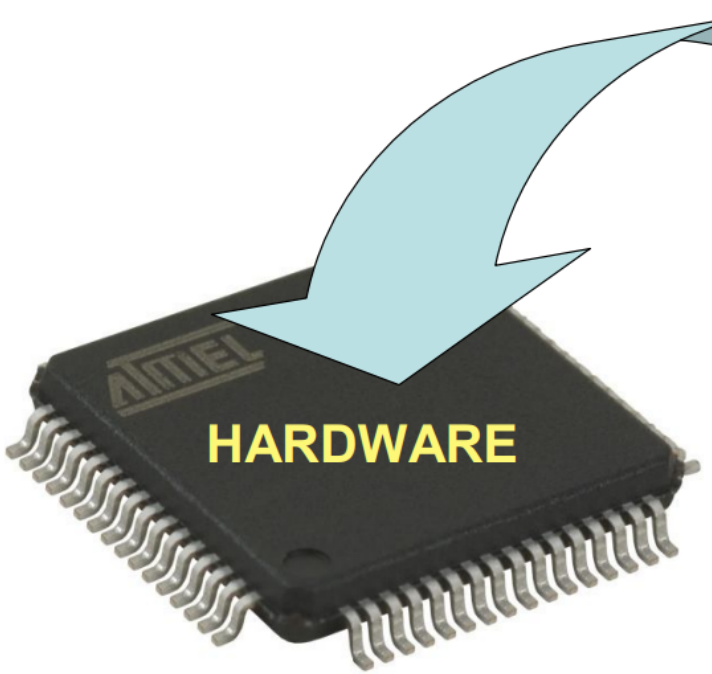
Embedded System is a **specialized computing system** designed to perform a dedicated function or **set of functions within a larger system.**

Unlike general-purpose computers, embedded systems are typically designed to be **efficient, reliable, and real-time in nature**, and they are usually built into the hardware they control.
specifically written to control the hardware.

EMBEDDED SYSTEM

Definition: An Embedded System is one that has computer hardware with software embedded in it as one of its important components.

Its software embeds in ROM (Read Only Memory). It does not need secondary memories as in a computer



SOFTWARE PROGRAM

```
#include <16f876a.h>
#use delay (clock=20000000)
#byte PORTB=6
main()
{
set_tris_b(0);
portb=255;        //decimal
delay_ms(1000);
portb=0x55;       //hexadecimal
delay_ms(1000);
portb=0b10101010;  //binary
delay_ms(500);
}
```

Key Characteristics of Embedded Systems:

1. **Task-Specific**: Designed to perform a specific task or a set of tasks.
2. **Real-time Operation**: Often operates within strict timing constraints.
3. **Integration**: Embedded systems are often integrated into larger devices or systems.
4. **Hardware and Software Combination**: Typically combines both hardware and software, with embedded software (firmware) specifically written to control the hardware.

Examples of Embedded Systems:

1. Microwave ovens
2. Smartphones
3. Automobiles (engine control, infotainment systems)
4. Medical devices (pacemakers, MRI machines)
5. Industrial machines

Comparison Table

Feature	Embedded System	General Purpose Computing System
Purpose	Designed for a specific function or task	Designed for multiple applications and tasks
Operating System	Fixed or custom OS, often non-modifiable	User can install or change OS (Windows, Linux, etc.)
Hardware Flexibility	Limited hardware customization	Highly flexible, supports multiple peripherals
User Applications	Cannot install third-party applications	Users can install and run multiple applications
Interfaces	Minimal, task-specific interfaces (e.g., IR remote)	Multiple interfaces (USB, Bluetooth, Wi-Fi, Ethernet, etc.)

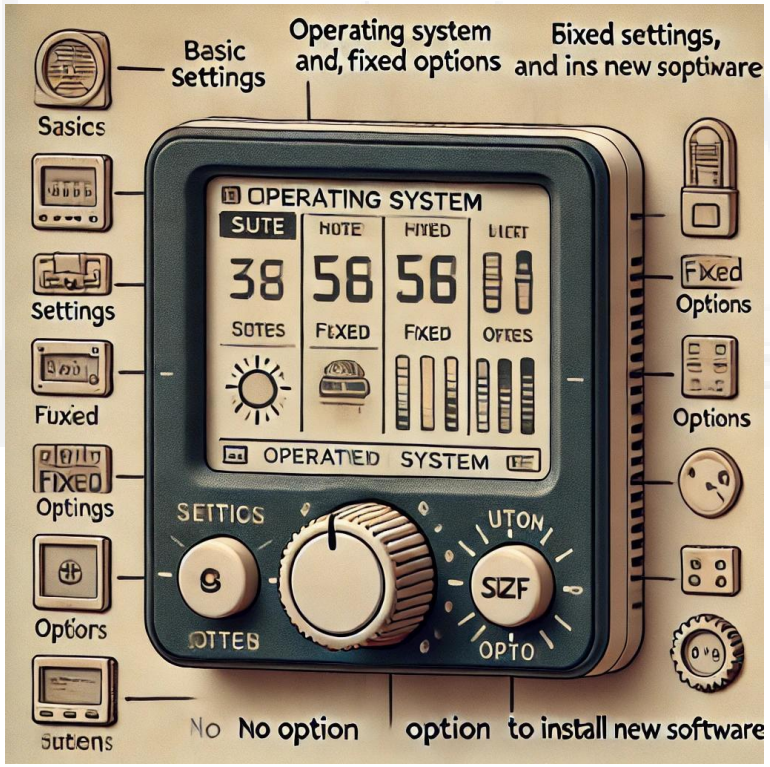
Comparison Table

Feature	Embedded System	General Purpose Computing System
Computational Requirements	Real-time processing with strict deadlines	General-purpose processing, not always real-time
Power Efficiency	Optimized for lower power consumption	Higher power consumption due to multitasking capability
Memory Availability	Limited and optimized for function	Large memory (RAM, HDD/SSD) for multiple applications
Software Modification	Not user-modifiable	Users can modify, update, and install new software
Functionality Expansion	Fixed function, cannot be easily repurposed	Can be repurposed for various tasks
Example Devices	DVD Player, Washing Machine, Car ECU	Desktop PC, Laptop, Palmtop, Workstation

Purpose of an Embedded System: A washing machine with predefined functions.

Purpose of a General-Purpose Computing System: A laptop running multiple applications.

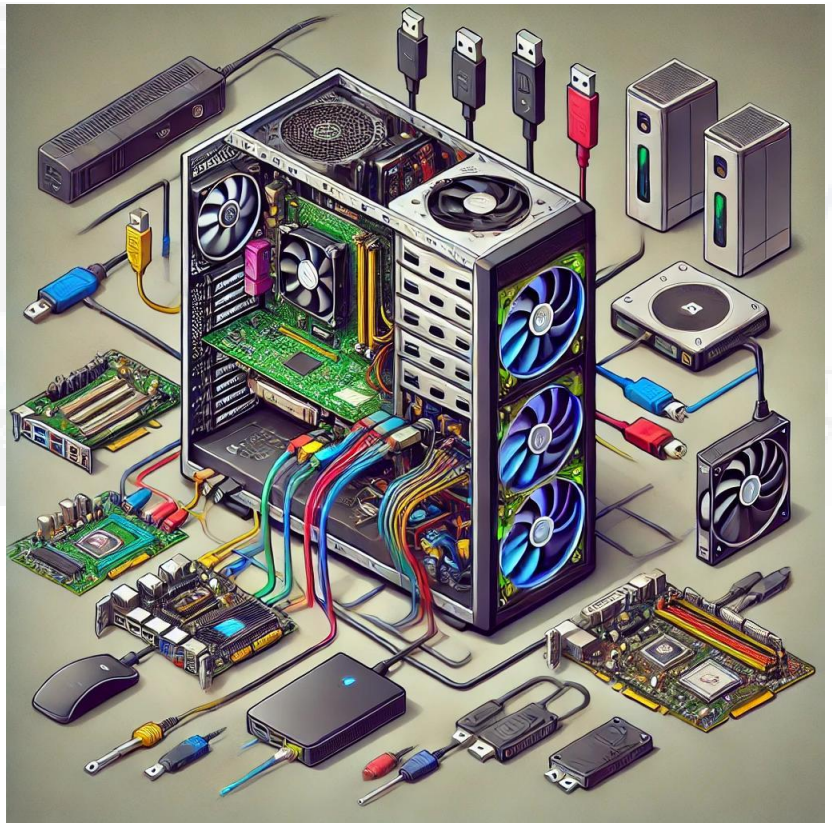
Operating System in an Embedded System: A digital thermostat with a fixed interface.



Operating System in a General-Purpose Computing System: A desktop PC with a flexible OS.

Hardware Flexibility in an Embedded System: A microwave oven with fixed buttons.

Hardware Flexibility in a General-Purpose Computing System: A desktop PC with multiple peripherals.



User Applications in an Embedded System: A smart thermostat with limited options.

User Applications in a General-Purpose Computing System: A smartphone with multiple third-party apps.

ATME College of Engineering

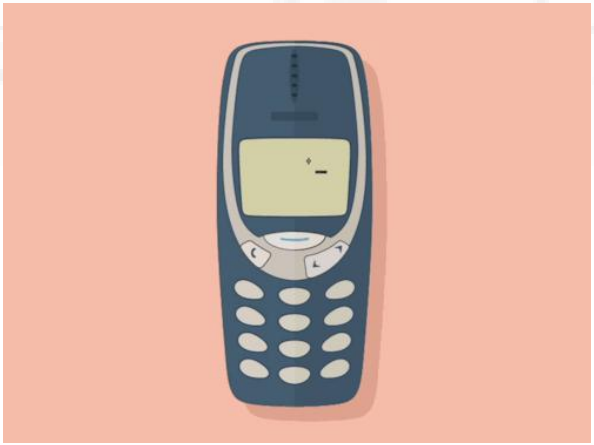
Department of EEE
Emitting Elite Energy

Classification of Embedded Systems

Based on Evolution and Complexity

First Generation Embedded Systems

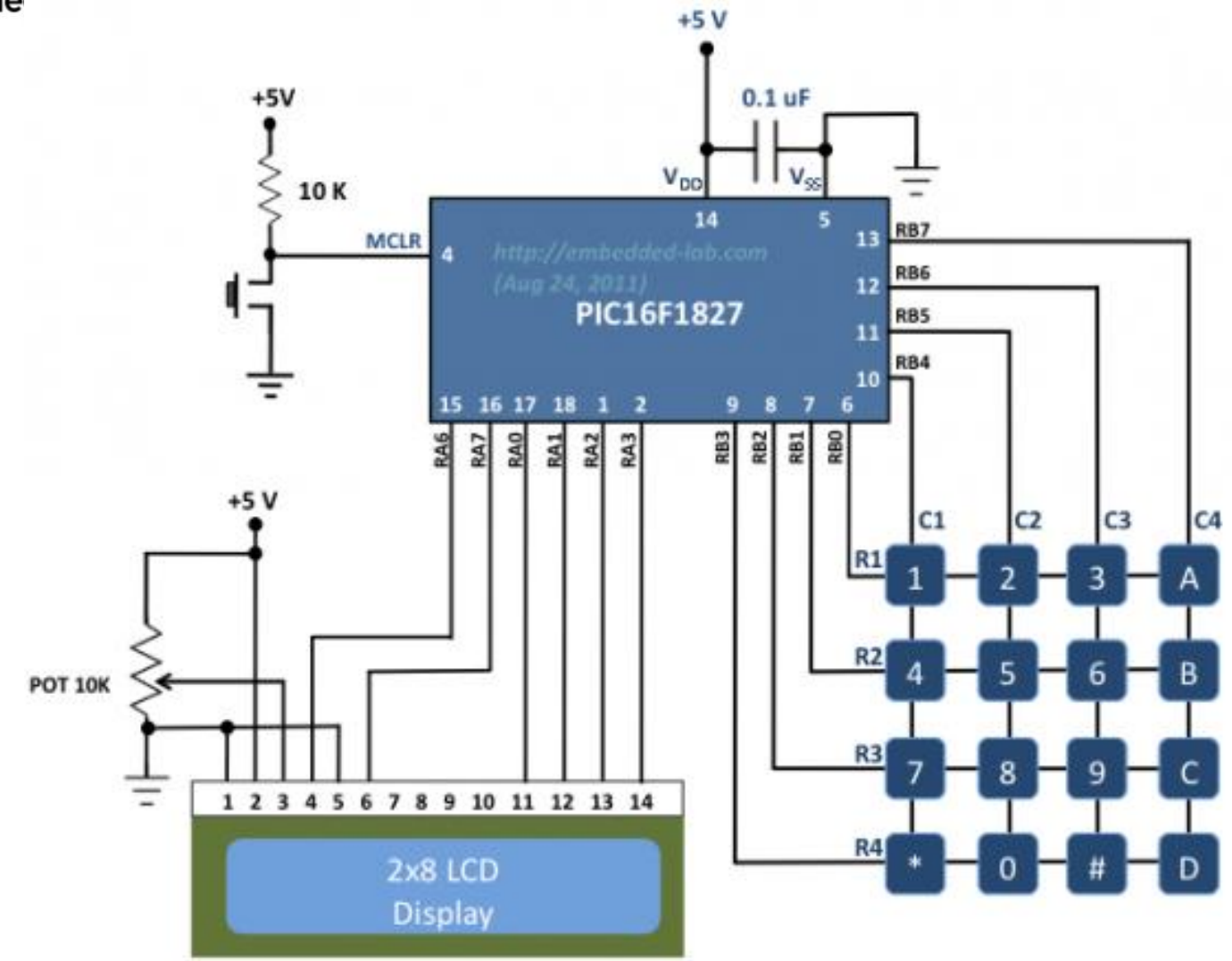
- Built around 8-bit microprocessors (e.g., 8085, Z80) and 4-bit microcontrollers. Simple circuits with Assembly language firmware.
- **Examples**: Digital telephone keypads, stepper motor control units.



Motorola MicoTac from 1991

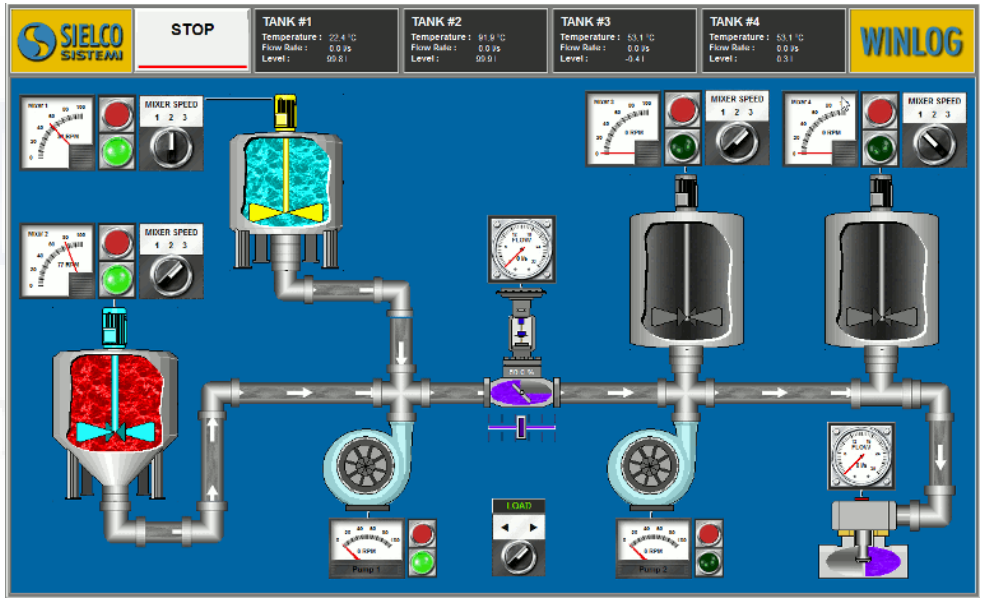
First Generation Embedded Systems

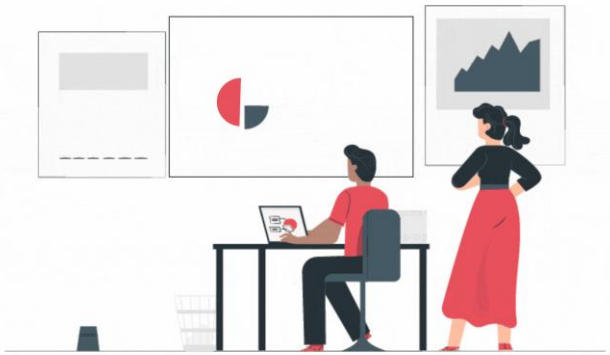
One of the first recognizably modern embedded systems was the Apollo Guidance Computer, developed ca. 1965 by Charles Stark Draper at the MIT Instrumentation Laboratory.


+5 V
0.1 uF
+5V
10 K
MCLR
VDD
VSS
14
5
4
http://embedded-lab.com
(Aug 24, 2011)
PIC16F1827
13
12
11
10
RB7
RB6
RB5
RB4
15 16 17 18 1 2 9 8 7 6
RA6 RA7 RA0 RA1 RA2 RA3 RB3 RB2 RB1 RB0
+5 V
POT 10K
C1 C2 C3 C4
R1 1 2 3 A
R2 4 5 6 B
R3 7 8 9 C
R4 * 0 # D
1 2 3 4 5 6 7 8 9 10 11 12 13 14
2x8 LCD
Display

Second Generation Embedded Systems

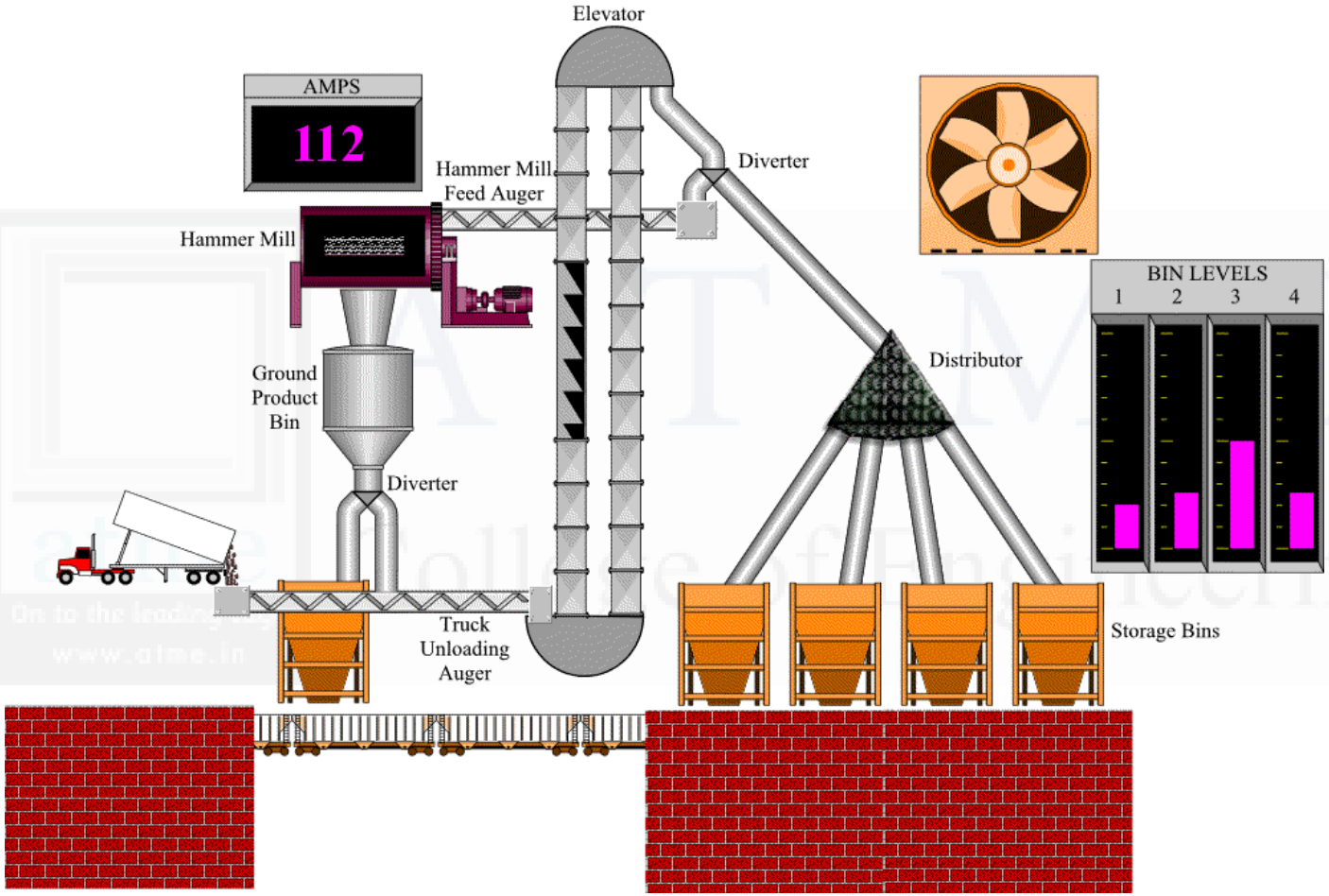
- Uses 16-bit microprocessors and 8/16-bit microcontrollers. More complex instruction sets, some with embedded OS.
- **Examples**: Data Acquisition Systems, SCADA systems.





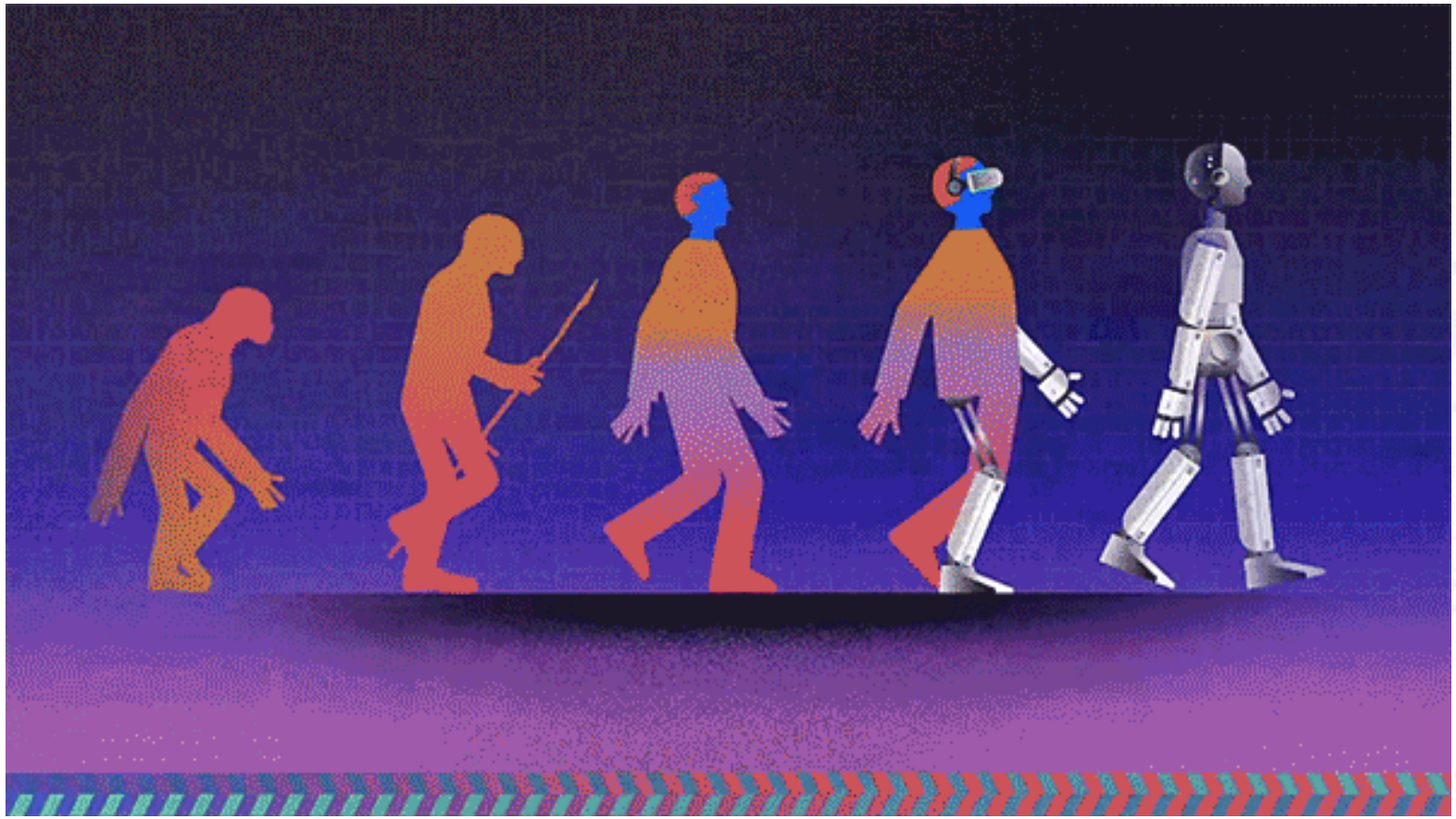
Third Generation Embedded Systems

- Introduced 32-bit processors, 16-bit microcontrollers, and specialized chips like DSPs and ASICs. Instruction pipelining and real-time OS became common.
- Examples: Robotics, industrial process control, networking.


Elevator
AMPS
112
Hammer Mill Feed Auger
Diverter
Hammer Mill
BIN LEVELS
1 2 3 4
Distributor
Ground Product Bin
Diverter
Truck Unloading Auger
Storage Bins

Fourth Generation Embedded Systems

- Involves System-on-Chip (SoC), multicore processors, and high-performance embedded OS. Greater miniaturization and integration.
- **Examples:** Smartphones, Mobile Internet Devices (MIDs).



Classification Based on Complexity and Performance

1. Small-Scale Embedded Systems

- Simple applications, not time-critical.
- Built on low-cost 8/16-bit microprocessors/microcontrollers.
- May or may not have an OS.
- The need to limit power dissipation when system is running continuously

Usually "C" is used for developing these system.

- Example: Electronic toys.

Programming tools: Editor, Assembler and Cross Assembler

Classification Based on Complexity and Performance

2. Medium-Scale Embedded Systems

- Moderately complex hardware and software.
- Uses 16/32-bit microprocessors/microcontrollers or DSPs or RISC.
- Usually includes an embedded OS.
- Both hardware and software complexity

Programming tools: RTOS, Source code Engineering Tool, Simulator, Debugger and Integrated Development Environment (IDE)

3.Large-Scale/Complex Embedded Systems

- High-performance, mission-critical applications.
- Uses 32/64-bit RISC processors, RSoC, multi-core processors.
- Includes multiple processors, co-processors, and accelerators.
- Uses high-performance Real-Time Operating Systems (RTOS).
- Examples: Media encoding/decoding, cryptographic processing.

Programming Tools: For these systems may not be readily available at a reasonable cost or may not be available at all. A compiler or retargetable compiler might have to be developed for this.

Applications of embedded systems

1. **Consumer Electronics** – Camcorders, cameras.
2. **Household Appliances** – TVs, DVD players, washing machines, fridges, microwaves.
3. **Home Automation & Security** – Air conditioners, sprinklers, alarms, CCTV cameras.
4. **Automotive Industry** – ABS, engine control, ignition, automatic navigation.
5. **Telecom** – Cell phones, telephone switches, multimedia applications.
6. **Computer Peripherals** – Printers, scanners, fax machines.
7. **Networking** – Routers, switches, hubs, firewalls.
8. **Healthcare** – Scanners, EEG, ECG machines.
9. **Measurement & Instrumentation** – Digital multimeters, CROs, logic analyzers, PLC systems.
10. **Banking & Retail** – ATMs, currency counters, POS systems.
11. **Card Readers** – Barcode scanners, smart card readers, handheld devices.

Purpose of Embedded Systems

1.Data Collection & Processing: Converts, stores, or displays data (**e.g., ECG monitors, cameras**).
2.Data Communication: Enables wired/wireless data transfer (**e.g., routers, home automation**).
3.Signal Processing: Enhances audio/video signals (**e.g., digital hearing aids**).
4.Monitoring: Observes real-time data without control (**e.g., ECG machines, oscilloscopes**).
5.Control Systems: Regulates processes using sensors & actuators (**e.g., air conditioners**).

1. Data Collection/Storage/Representation

Embedded systems collect and process data from the external environment, which can be in analog or digital form.

Analog data is converted into digital form using Analog-to-Digital (A/D) converters.

The collected data may be:

Stored for further processing.

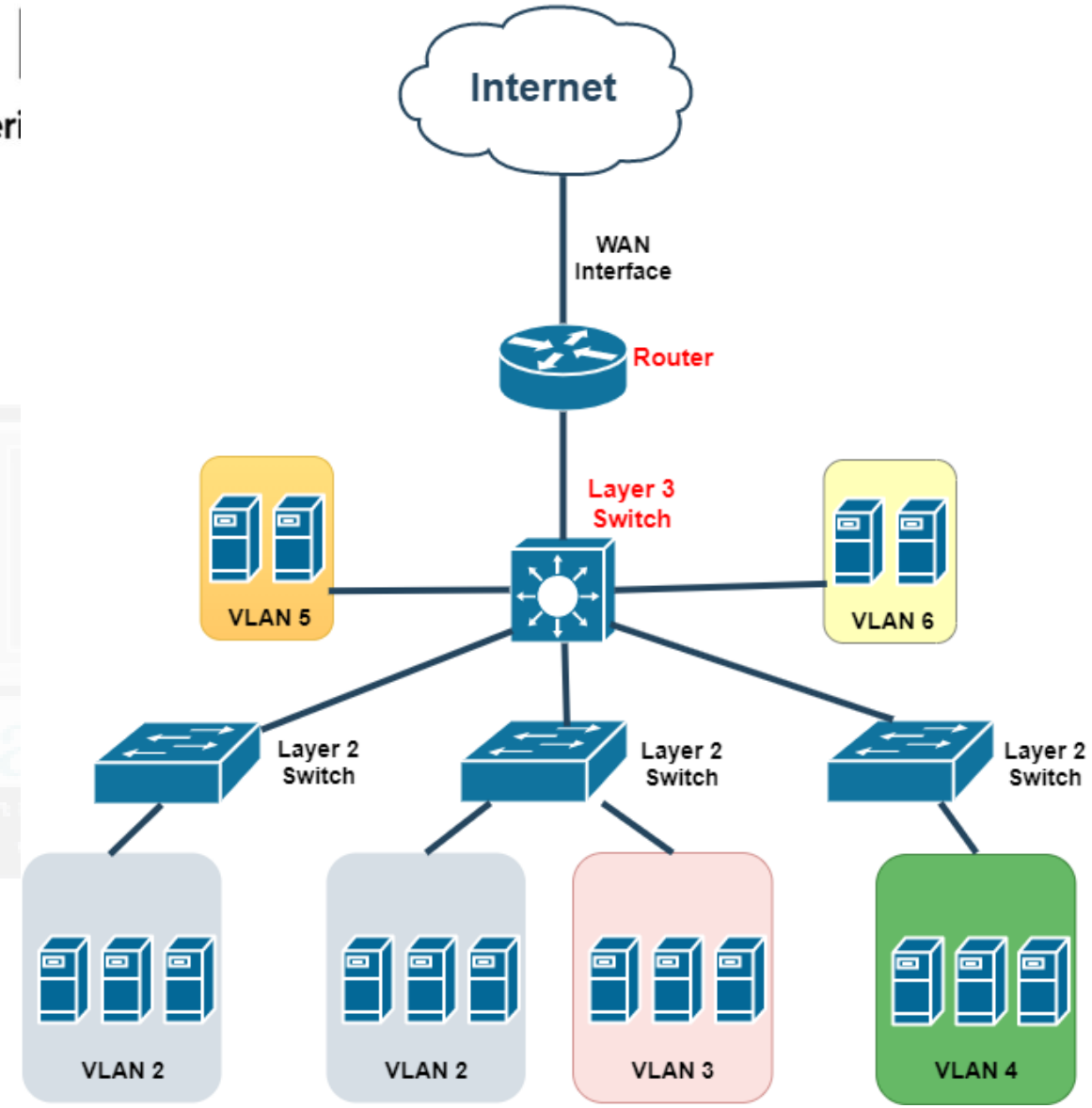
- Represented visually (LCD, LED) or audibly (buzzers, alarms).
- Deleted after processing.

Examples: Digital multimeters, ECG monitoring devices, digital cameras (store and display images).

2. Data Communication

Embedded systems enable communication between devices, either through wired (USB, RS-232, TCP/IP) or wireless (Bluetooth, Wi-Fi, ZigBee) methods.

- Data transmission can be analog or digital, with modern systems favoring digital communication.
- Some embedded systems serve as dedicated transmission units, ensuring secure and efficient data exchange.
- **Examples**: Network routers, switches, telephone systems, home automation devices.


Internet
WAN
Interface
Router
Layer 3
Switch
VLAN 5
VLAN 6
Layer 2
Switch
Layer 2
Switch
Layer 2
Switch
VLAN 2
VLAN 2
VLAN 3
VLAN 4

3. Data (Signal) Processing

Embedded systems are used in applications requiring signal processing, such as audio, video, and electrical signal manipulation.

They perform functions like speech coding, audio synthesis, and video encoding.

Example: Digital hearing aids, which amplify and process sound to improve hearing for individuals with impairments.

4. Monitoring

Designed to observe variables but not control them.
Sensors gather real-time data, which is displayed for monitoring purposes without altering the system.
Common in healthcare and industrial applications.
Examples: Medical: ECG machines (monitor heartbeat but do not regulate it).
Industrial: Digital storage oscilloscopes (CROs), logic analyzers, and multimeters (monitor voltage, current, etc.).

5. Control

Embedded systems that regulate processes by taking corrective actions based on sensor input.

These systems consist of sensors (input), actuators (output), and a control unit that adjusts the output to maintain the desired conditions.

Example: Air conditioners: The sensor (thermistor) measures the current temperature.

The user input (desired temperature) is set via a remote control.

The actuator (compressor) adjusts airflow to maintain the desired temperature

Feature	Microprocessor	Microcontroller
Definition	A silicon chip representing a CPU that performs arithmetic and logical operations.	A highly integrated chip with CPU, RAM, ROM, timers, and I/O ports for embedded systems.
Dependency	Dependent on **external hardware** (memory, timers, interrupt controllers, etc.).	Self-contained and does **not require external hardware for functioning.**
Purpose	General-purpose usage in industrial and high-performance computing.	Designed for specific tasks in embedded systems.
I/O Ports	Does not include built-in I/O ports; requires external components like programmable peripheral interface chips.	Includes multiple built-in I/O ports (8, 16, or 32-bit) for direct use.
Applications	Used in computers, servers, and high-performance devices.	Used in IoT, robotics, automotive systems, and consumer electronics.

Feature	Microprocessor	Microcontroller
Architecture	Based on Harvard or Von-Neumann architecture with RISC or CISC instruction sets.	Similar architectures, often domain-specific instruction sets (e.g., AVR for automotive).
Power Efficiency	Less efficient in terms of power consumption.	Optimized for power efficiency and compactness.
Target Market	High-end markets where performance is critical.	Embedded markets, where cost and size are important.
Cost	Expensive due to dependency on external hardware.	Cost-effective, with all required components integrated.

1. CISC (Complex Instruction Set Computing) Microprocessors
Example: Intel x86, AMD Ryzen

- Executes multiple instructions in a single cycle, making it versatile but complex.
- Used in **personal computers and servers**.

2. RISC (Reduced Instruction Set Computing) Microprocessors
Example: ARM Cortex, Apple M1, IBM POWER

- Executes simple, optimized instructions faster than CISC.
- Used in **smartphones, tablets, and embedded systems**.

3. DSP (Digital Signal Processing) Microprocessors
Example: Texas Instruments TMS320, Qualcomm Hexagon

- Specialized for **real-time signal processing** in **audio, video, and communication**.

4. Embedded Microprocessors
Example: Raspberry Pi (Broadcom), ESP32, ATmega328

- Designed for **specific applications** like IoT, industrial control, and automation.

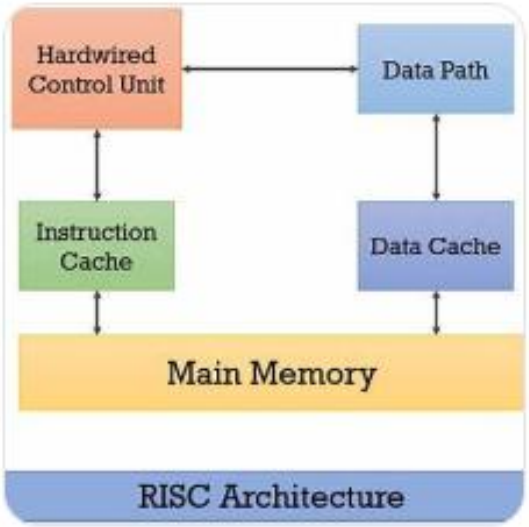
5. GPU (Graphics Processing Unit) Microprocessors
Example: NVIDIA RTX Series, AMD Radeon

- Parallel processing power for **gaming, AI, and scientific computing**.

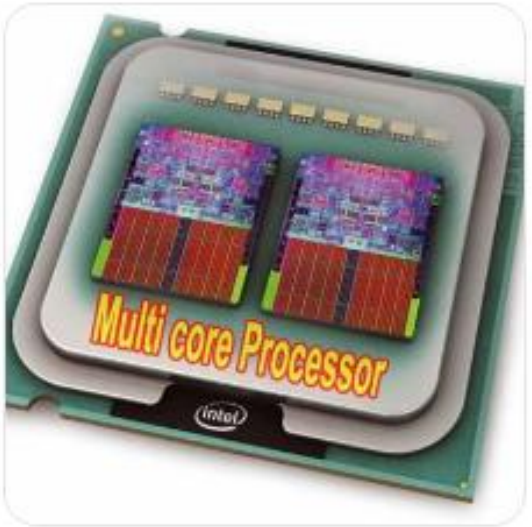
6. Multi-Core Microprocessors
Example: Intel Core i9, AMD Threadripper

- Multiple cores for **parallel computing and high-performance tasks**.


INTEL CORE i5
i5-10600
SRH37 3.30GHZ


Hardwired Control Unit
Data Path
Instruction Cache
Data Cache
Main Memory
RISC Architecture


DSP
TMS320
VC5509APGE
4A-46AFKTW
G4


Multi core Processor
intel

Feature	RISC (Reduced Instruction Set Computing)	CISC (Complex Instruction Set Computing)
Instruction Set	Lesser number of instructions.	Greater number of instructions.
Instruction Pipelining	Supports instruction pipelining for increased execution speed.	Generally, no instruction pipelining is present.
Instruction Set Orthogonality	Orthogonal instruction set (any instruction can operate on any register or use any addressing mode).	Non-orthogonal instruction set (instructions are instruction-specific regarding registers and addressing modes).
Operations	Operations are performed only on registers; memory operations are limited to load and store instructions.	Operations can be performed directly on registers or memory, depending on the instruction.
Number of Registers	Large number of general-purpose registers are available.	Limited number of general-purpose registers.

Feature	RISC (Reduced Instruction Set Computing)	CISC (Complex Instruction Set Computing)
Programming	Requires more lines of code as the instructions are simpler and perform smaller operations.	Instructions are like macros in C language, allowing complex operations with a single instruction, reducing the code length.
Instruction Length	Single, fixed-length instructions.	Variable-length instructions.
Hardware Complexity	Uses less silicon and has a lower pin count due to simpler instruction decoding.	Requires more silicon and additional decoder logic to handle complex instruction decoding.
System Architecture	Uses Harvard architecture (separate buses for program and data memory).	Can use either Harvard or Von-Neumann architecture (shared bus for program and data memory).

Endianness in Memory Storage

Endianness defines **how multi-byte data is stored in memory by a processor**.

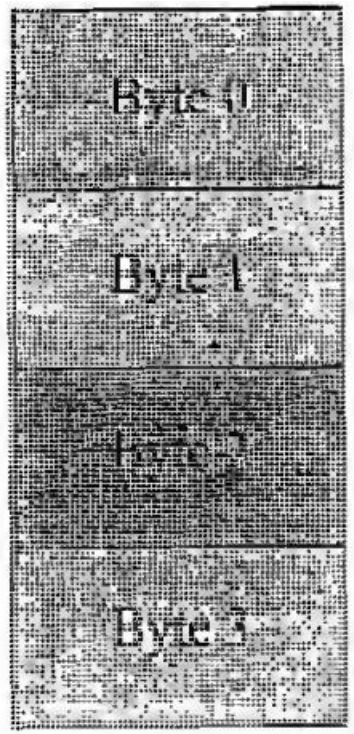
It determines the order in which bytes are arranged in a system where the word size is greater than one byte.

Types of Endianness:

1. Little-Endian:

a. The **lower-order byte** is stored at the **lowest memory address**.
b. The **higher-order byte** is stored at the **next higher memory address**.
c. Example (for a 4-byte integer Byte3 Byte2 Byte1 Byte0):
    i. Memory Order: **Byte0 → Byte1 → Byte2 → Byte3** (Lowest to Highest Address).
d. Commonly used in **Intel x86** processors.

Little-Endian

Base Address + 0	Byte 0		0×20000 (Base Address)
Base Address + 1	Byte 1		0×20001 (Base Address + 1)
Base Address + 2	Byte 2		0×20002 (Base Address + 2)
Base Address + 3	Byte 3		0×20003 (Base Address + 3)

1. Big-Endian:

a. The **higher-order byte** is stored at the **lowest memory address**.

b. The **lower-order byte** is stored at the **next higher memory address**.

c. Example (for a 4-byte integer Byte3 Byte2 Byte1 Byte0):

   i. Memory Order: **Byte3 → Byte2 → Byte1 → Byte0** (Lowest to Highest Address).

d. Used in **Motorola, PowerPC, and some network protocols**.

Base Address + 0	Byte 3	Byte 3	0×20000 (Base Address)
Base Address + 1	Byte 2	Byte 2	0×20001 (Base Address + 1)
Base Address + 2	Byte 1	Byte 1	0×20002 (Base Address + 2)
Base Address + 3	Byte 0	Byte 0	0×20003 (Base Address + 3)

ASIC –Application-Specific Integrated Circuit

1.Definition: Custom microchip for specific applications, replacing general-purpose logic.

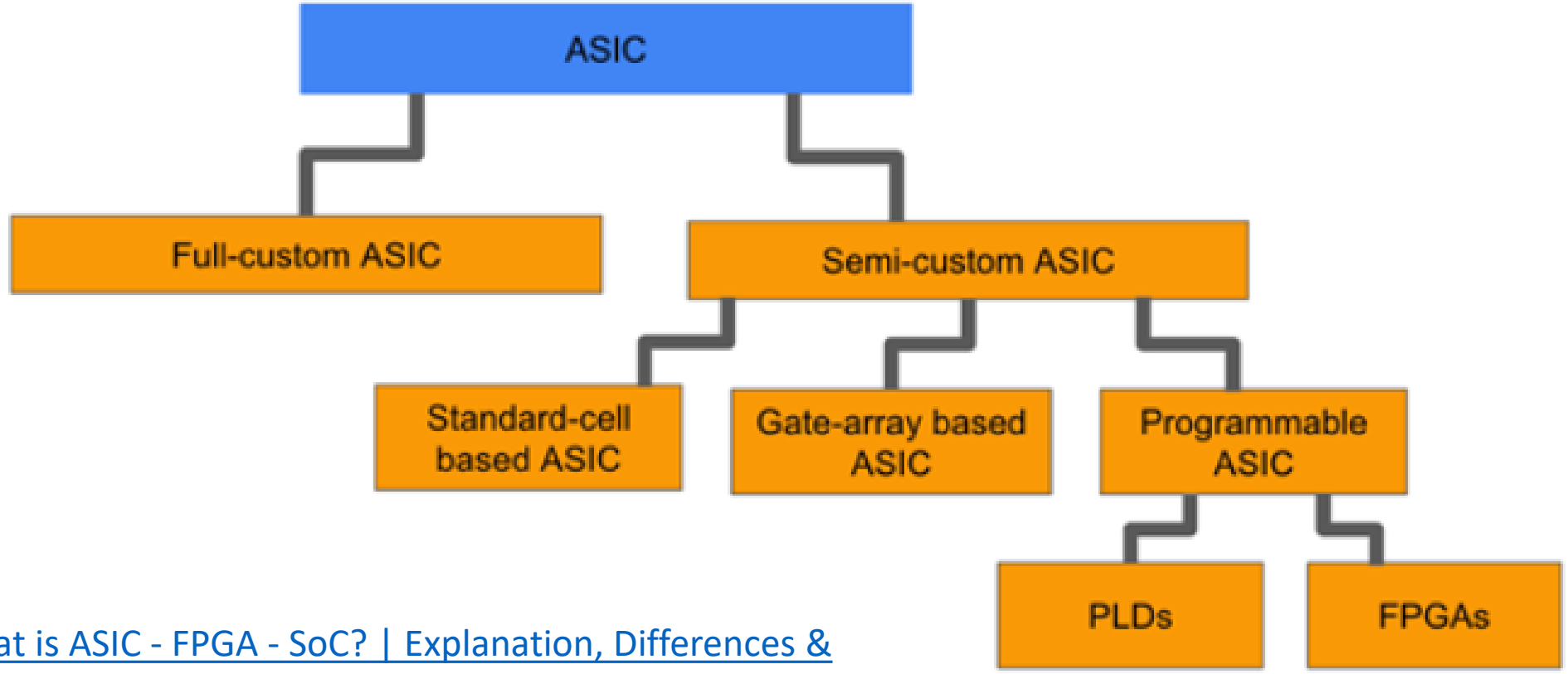
2.Advantages: Reduces cost, compact, high-performance, and task-optimized.

3.Types: Pre-fabricated (specialized) & **Custom** (tailored for users).

4.NRE Cost: One-time investment, viable for **large-scale production**.

5.ASSP: Publicly available ASICs for **specific applications** (**e.g., ADE7760 Energy Meter**).

6.Proprietary: Confidential designs, limited public disclosure.



(512) What is ASIC - FPGA - SoC? | Explanation, Differences & Applications - YouTube

The primary difference between these devices lies in circuit complexity and the number of available logic cells. **An SPLD typically consists of a few hundred gates, whereas a CPLD consists of a few thousand logic gates.**

Programmable Logic Devices (PLDs) – Key Points

1.Definition & Types: Perform system functions; categorized as **Fixed Logic Devices** (non-changeable) and **PLDs** (reconfigurable).

2.Advantages: Fast development, cost-effective, flexible inventory, **field reprogrammability**.

3.Types:

- **FPGAs:** High density, high performance (**e.g., Xilinx Virtex**).
- **CPLDs(Complex Programmable Logic Device):** Lower density, predictable timing (**e.g., Xilinx CoolRunner**).

4.PLDs vs Fixed Logic: PLDs allow **modification & testing**, while fixed logic devices are **permanent**.

5.FPGA Evolution: From **40 MHz, costly** to **300 MHz, affordable, with integrated features**.

CPLD (Complex Programmable Logic Device)

A CPLD is an integrated circuit that can be programmed to implement a variety of digital logic functions.
It's like a "customizable" logic circuit that can be reprogrammed multiple times for different tasks. CPLDs have a moderate number of logic gates and are typically used for tasks requiring high-speed logic but lower complexity than an FPGA.

Due to their small size and low power consumption, CPLDs are ideal for use in portable and handheld digital devices.

FPGA (Field-Programmable Gate Array)

An FPGA is an integrated circuit that can be configured after manufacturing to carry out a variety of tasks.

Unlike CPLDs, FPGAs offer a higher density of logic elements, making them capable of implementing more complex systems.

They contain thousands to millions of logic gates, flip-flops, and other elements that can be programmed to perform various tasks.

COTS & Sensors/Actuators – Key Points

Commercial Off-the-Shelf (COTS):

- **Definition:** Ready-made components for easy system integration.
- **Examples:** Remote circuits, ADCs, IR detectors, TCP/IP modules (**e.g., WIZnet**).
- **Advantages:** Fast development, cost-effective, pre-built firmware.
- **Disadvantages: Vendor lock-in, compatibility issues, risk of discontinuation**.

Sensors & Actuators:

1. Temperature Sensor (e.g., Thermistor) in Smart Thermostats

- **Sensor:** Measures temperature changes in the environment and provides data to the thermostat to control heating and cooling systems.
- **Actuator:** Adjusts the heating or cooling unit based on temperature data from the sensor.

Sensors & Actuators:

2. Pressure Sensor in Car Tire Monitoring Systems

- **Sensor:** Measures the pressure inside tires and provides data to the car's onboard computer system.

- **Actuator:** Illuminates the dashboard light or triggers an alert when tire pressure is too low.

Sensors & Actuators:

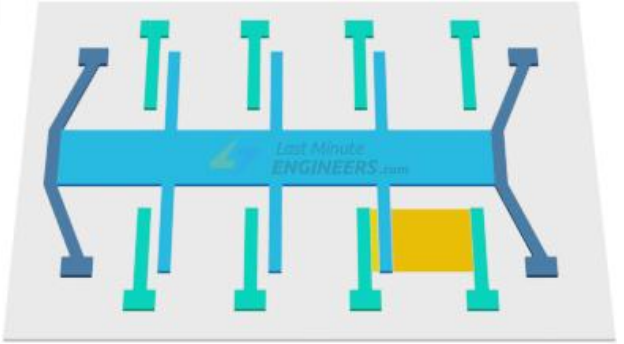
3. Proximity Sensor in Automatic Doors

- **Sensor:** Detects the presence of a person approaching the door (e.g., infrared sensor).
- **Actuator:** Triggers the door to open automatically.

Sensors & Actuators:

4. Accelerometer in Smartphones (Shake Detection)

- **Sensor:** Measures the phone's acceleration to detect movement or tilting.
- **Actuator:** Triggers functions like screen rotation, gaming inputs, or shake-to-wake features.



Communication Interfaces

I2C (Inter-Integrated Circuit):

•This protocol typically uses two lines: **SCL (Clock Line)** and **SDA (Data Line)**.

•It's a serial communication method that allows multiple devices to communicate with one or more master controllers.

SPI (Serial Peripheral Interface):

•SPI uses **four lines**: **MISO (Master In Slave Out)**, **MOSI (Master Out Slave In)**, **SCLK (Serial Clock)**, and **SS (Slave Select)**.

UART (Universal Asynchronous Receiver-Transmitter):

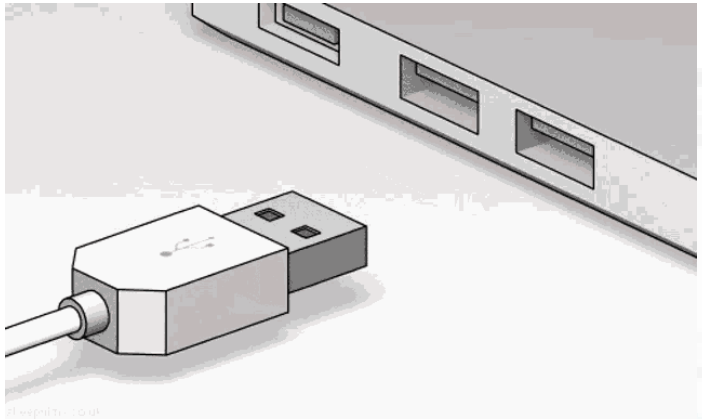
•UART typically has two lines: **TX (Transmit)** and **RX (Receive)**. It's used for asynchronous communication, commonly found in serial ports.

Parallel Communication:

•This protocol uses multiple data lines, often 8, 16, or more, allowing data to be sent simultaneously (in parallel) rather than serially.

Communication Interfaces –Types:

External: Device communication (**USB, Ethernet, Wi-Fi, Bluetooth, RF, GPRS**)



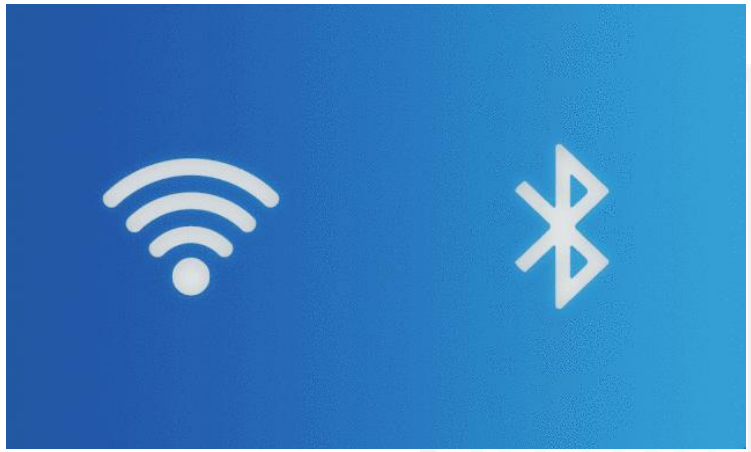
USB (Universal Serial Bus):

- A standard for connecting devices to a computer or other devices.
- Allows for data transfer and power supply over a single cable.

- A wired networking protocol used to connect computers and devices in a local area network (LAN).
- Typically uses a twisted-pair cable or fiber-optic cables for data transmission.

Communication Interfaces –Types:

External: Device communication (**USB, Ethernet, Wi-Fi, Bluetooth, RF, GPRS**)



Wi-Fi:

•A wireless networking protocol that allows devices to connect to the internet or local networks without using physical cables.

Bluetooth:

•A short-range wireless communication technology, typically used to connect devices like headphones, speakers, and smartphones.

•Operates over small distances (usually within 100 meters)

Communication Interfaces – Key Points

1.Types:

- **Onboard:** Internal connections (**I2C, SPI, UART, Parallel**).
- **External:** Device communication (**USB, Ethernet, Wi-Fi, Bluetooth, RF, GPRS**).

2.I2C Bus:

- **Synchronous, bi-directional, half-duplex (SCL & SDA)**.
- **Master-Slave Model:** Master controls, slaves respond, supports multi-master.

3.I2C Communication Sequence:

- **Start → Address → Read/Write Bit → Acknowledge → Data Transfer → Stop**.

4.I2C Data Rates:

- **Standard:** 100 kbps, **Fast:** 400 kbps, **High-Speed:** 3.4 Mbps.

Note: I2C stands for **Inter-Integrated Circuit**.

I2C Communication Sequence

1. **Start Condition:** Master pulls **SDA LOW** while **SCL is HIGH**.
2. **Addressing:** Master sends **7-bit/10-bit slave address**.
3. **Read/Write Bit: 0 = Write, 1 = Read**.
4. **ACK from Slave:** Slave pulls **SDA LOW** if address matches.
5. **Data Transfer:**
   - **Write:** Master sends **8-bit data**.
   - **Read:** Slave sends **8-bit data**.
6. **ACK After Each Byte:** Receiver sends acknowledgment.
7. **Stop Condition:** Master **releases SDA HIGH** while **SCL is HIGH**.

Serial Communication Interfaces – Key Points

SPI (Serial Peripheral Interface):

- Synchronous, full-duplex, four-wire interface (Motorola).
- Master-Slave System: One master, multiple slaves (selected via SS).
- Signal Lines: MOSI, MISO, SCLK, SS.
- Fast, ideal for continuous data transfer but lacks acknowledgment mechanism.

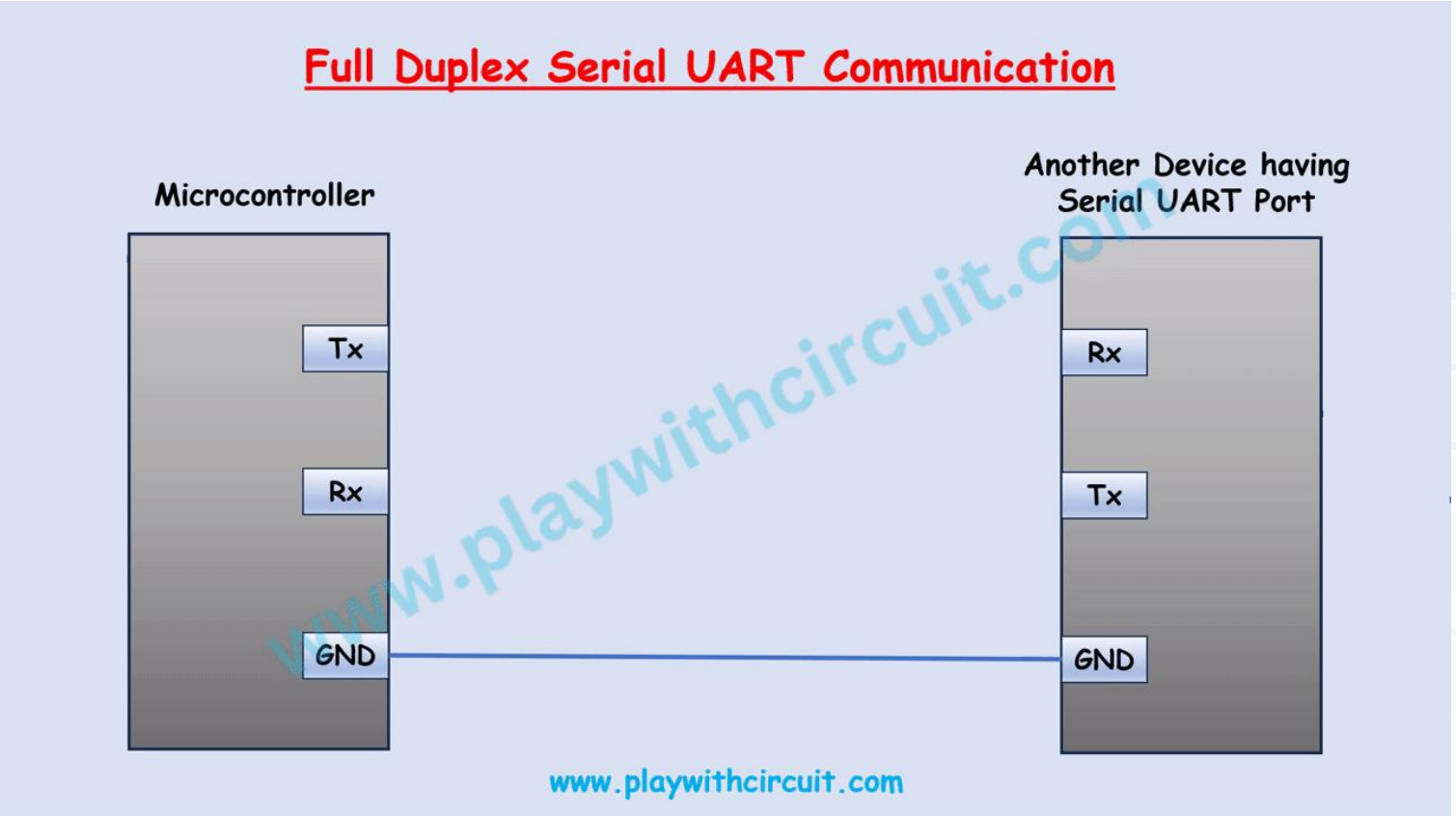
1. SPI Serial Peripheral Interface is communication **between two devices**, **one bit at a time** sequential one bit at time over a computer bus (a subsystem that transfers data Between components inside of a computer) or a communication channel (wire or wave transmission).
2. Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating short distances.
3. It can also be used for communication between two microcontrollers.

Serial Communication Interfaces – Key Points

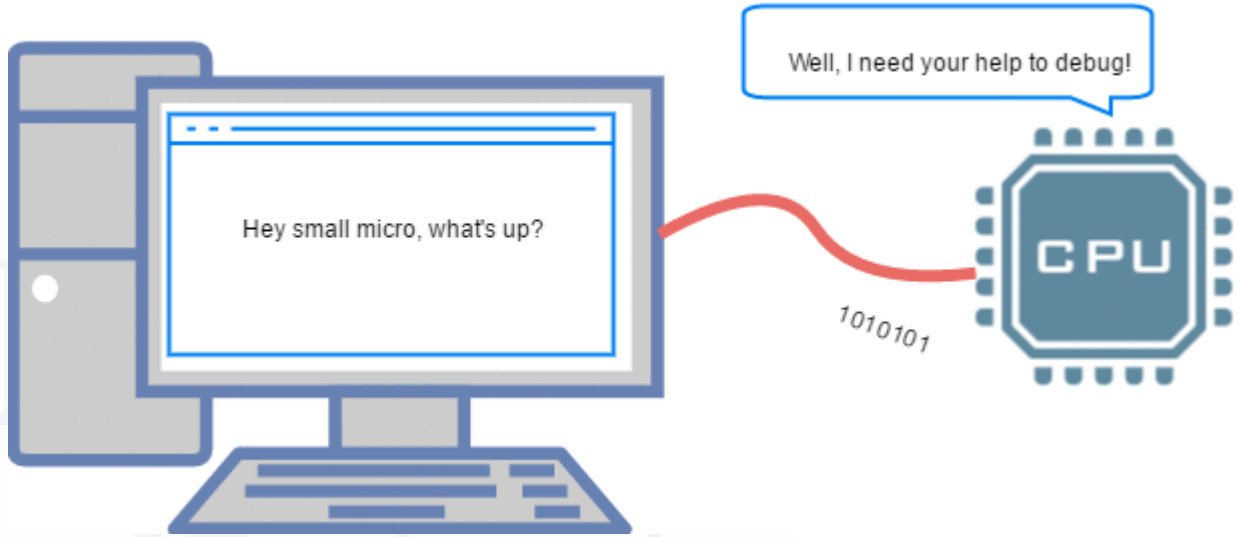
UART (Universal Asynchronous Receiver Transmitter):

- Asynchronous serial communication, no clock signal needed.
- Uses Start/Stop bits, optional Parity for error detection.
- TX (Transmit) → RX (Receive) connection.
- Supports hardware handshaking for flow control.

Data Frame of UART for Transmission of Letter 'H'


Full Duplex Serial UART Communication
Microcontroller
Another Device having Serial UART Port
Tx
Rx
GND
Rx
Tx
GND
www.playwithcircuit.com
www.playwithcircuit.com

USART	UART
In USART, half duplex mode is used.	While in UART, full duplex mode is used.
The speed of USART is more than the speed of UART.	While the speed of UART is comparatively less.
USART uses both data signals and clock for its functioning.	While UART entails data signals only for its functioning.
In USART, data is transmitted in the form of blocks.	While in UART, data is transmitted in the form of bytes(one byte at a time).



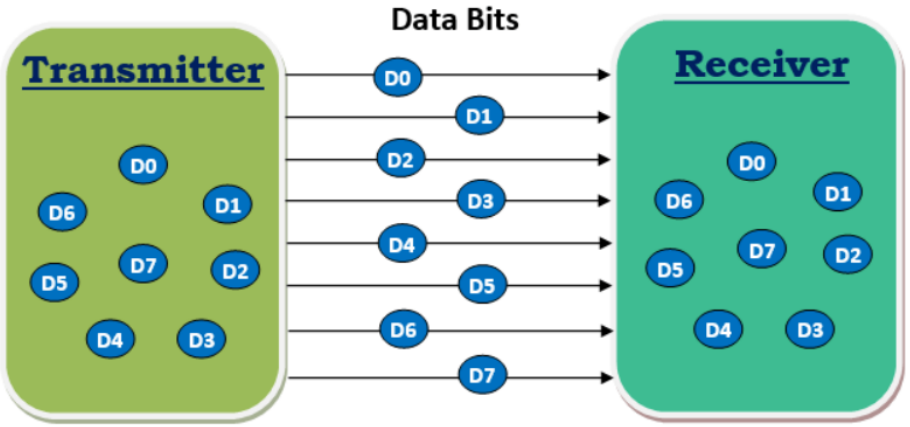
Serial Communication Interfaces – Key Points

1-Wire Interface:

- **Asynchronous half-duplex**, single-wire (DQ) protocol (Maxim).
- Master-Slave model, each device has a unique 64-bit ID.
- Sequence: Reset → Presence → Address → Function → Data Transfer.
- **Bit Writing**: 1 = Short LOW pulse, 0 = Long LOW pulse.
- **Bit Reading**: 1 = Released bus, 0 = LOW bus.

Parallel Interface – Key Points

- **Definition:** On-board interface for **memory-mapped peripheral communication** with the host processor.
- **Communication Control:** Managed by **Read/Write & Chip Select signals**.
- **Data Transfer: Host-initiated**, direction controlled by **RD/WR signals**.
- **Interrupt Handling:** Devices notify the processor via **interrupt lines**.
- **Bus Width:** Matches processor's **data bus** (4-bit, 8-bit, 16-bit, etc.).
- **Timing:** Follows **strict protocols** for reliable data transfer.



External Communication Interfaces – Key Points

RS-232:

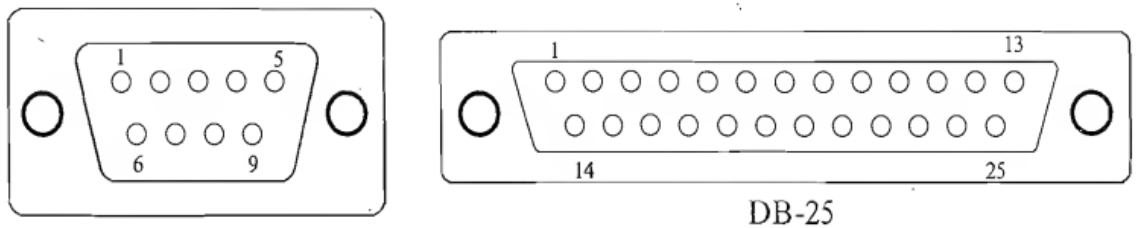
- Point-to-point serial communication (EIA standard).
- Baud Rate: Up to 19.2 Kbps, max 50 ft distance.
- Connectors: DB-9/DB-25, uses voltage signaling (+3V to +25V for 0, -3V to -25V for 1).
- Requires level converters (e.g., MAX232) for TTL compatibility.

RS-422:

- Differential communication, 100 Kbps, max **400 ft distance.**
- Supports multi-drop: 1 transmitter, 10 receivers.

RS-485:

- Enhanced RS-422, multi-drop support for 32 transmitters & 32 receivers.
- Uses addressing for device identification.
- Ideal for **long-distance industrial communication.**

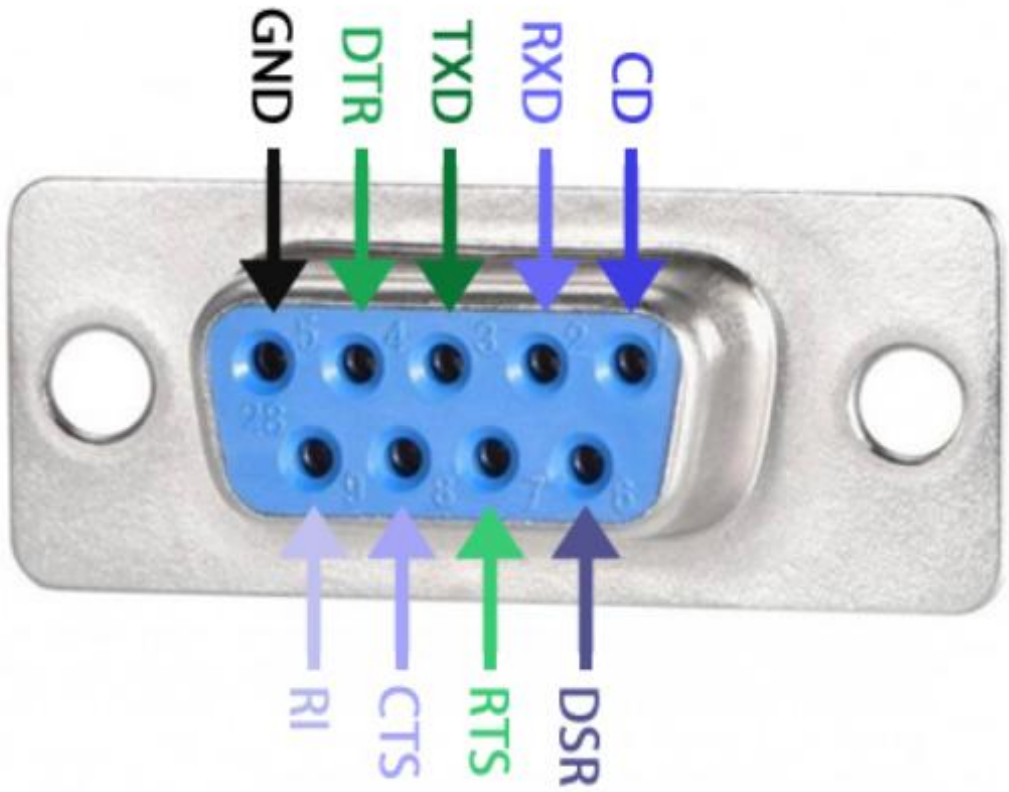


DB-9 DB-25

RS-232 Connector Pin Details (DB-9 and DB-25)

Key Notes:

- **DB-9 connectors** are more common in modern systems, while **DB-25 connectors** are mostly obsolete.
- For simple data transmission, only **TXD, RXD, and GND** are essential.
- Here are all the pins for **RS-232 connectors (DB-9 and DB-25)**:

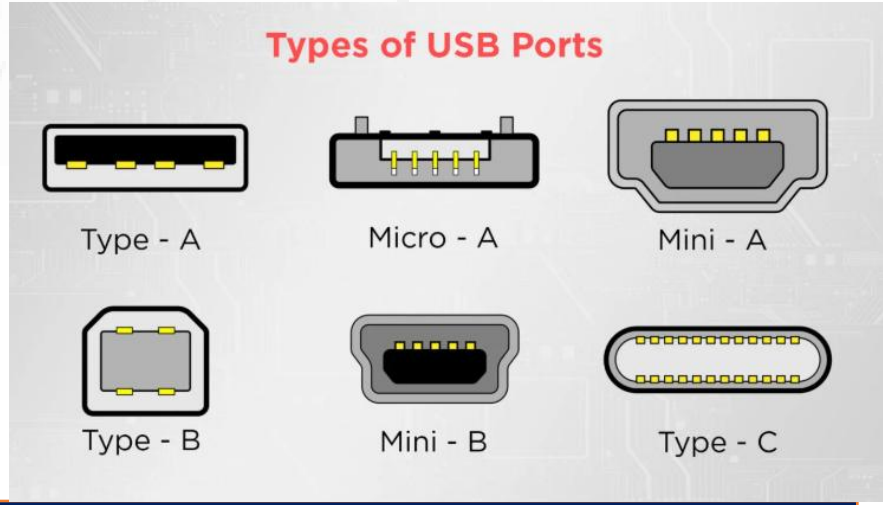


Name	Direction	Description
CD	IN	Carrier Detect Input Signal
RXD	IN	Receive Data Signal
TXD	OUT	Transmit Data Signal
DTR	OUT	Data Terminal Ready Signal
GND	-	System Ground
DSR	IN	Data Set Ready Signal
RTS	OUT	Request to Send Signal
CTS	IN	Clear to Send Signal
RI	IN	Ring Indicator Signal

USB – Key Points

- **High-speed wired serial bus** (1995, Intel & Microsoft).
- **Star topology**, supports **127 devices**.
- **Connectors: Type A (host), Type B (slave), Mini/Micro USB**.
- **Data Types: Control, Bulk, Isochronous, Interrupt**.
- **Data Rates: USB 1.0 (1.5–12 Mbps), USB 2.0 (480 Mbps), USB 3.0 (4.8 Gbps)**.
- **Power: 5V, 500mA**.

USB Connector Pins:



USB 1.0	USB 2.0	USB 3.0	USB 4.0
12 Mbps	480 Mbps	5 Gbps	40 Gbps
5V	5V	5V	5V-8V
0.5 amps	0.5 amps	0.9 amps	0.5 - 0.9 amps
2.5 watts	2.5 watts	4.5 watts	240 watts
Not Fast Charging	Not Fast Charging	Not Fast Charging	Fast Charging

Wireless Communication Interfaces – Key Points

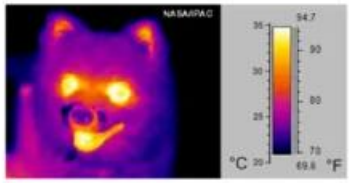
Infrared (IrDA):
•Half-duplex, line-of-sight wireless.
•Range: 10 cm – 1 m, Data Rate: 9.6 kbps – 16 Mbps.
•Used in: TV remotes, file transfer, early mobile phones.

Bluetooth (BT):
•Short-range (2.4 GHz), low-power.
•Range: ~30 feet, Data Rate: Up to 1 Mbps.
•Used in: Phones, headsets, wearables, speakers.

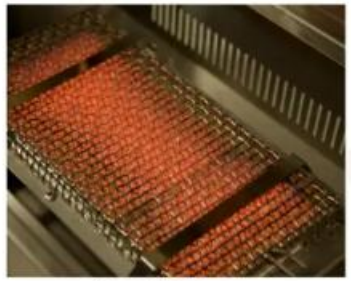
Wi-Fi (IEEE 802.11):
•IP-based wireless networking.
•Range: 100-300 feet, Data Rate: 1 Mbps – 150 Mbps.
•Used in: Internet access, IoT, smart devices.

ZigBee (IEEE 802.15.4):
•Low-power, low-data WPAN.
•Range: Up to 100m, Data Rate: 20 – 250 Kbps.
•Used in: Home automation, smart meters, IoT.

GPRS (General Packet Radio Service):
•Packet-switched mobile data (GSM).
•Max Data Rate: 171.2 kbps.
•Used in: Mobile internet, GPS trackers, M2M communication.

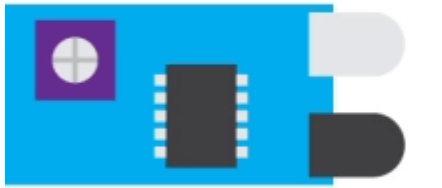

Thermal imaging


Night vision

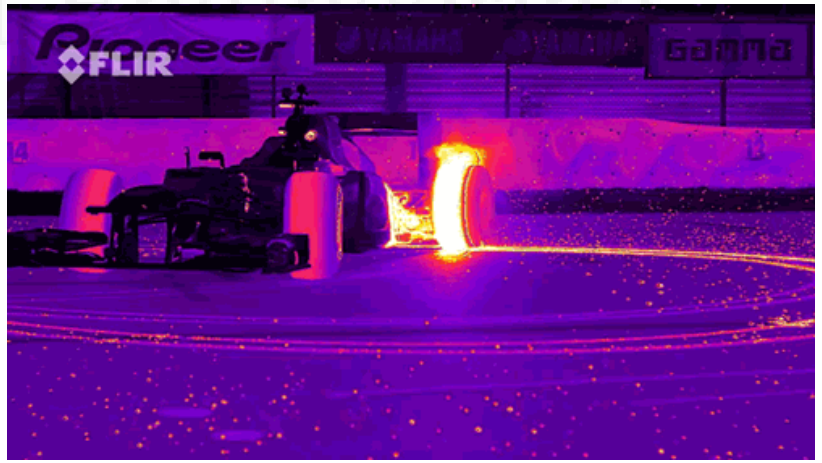

Cooking


Short range communication


Remote controls for TV/VCR

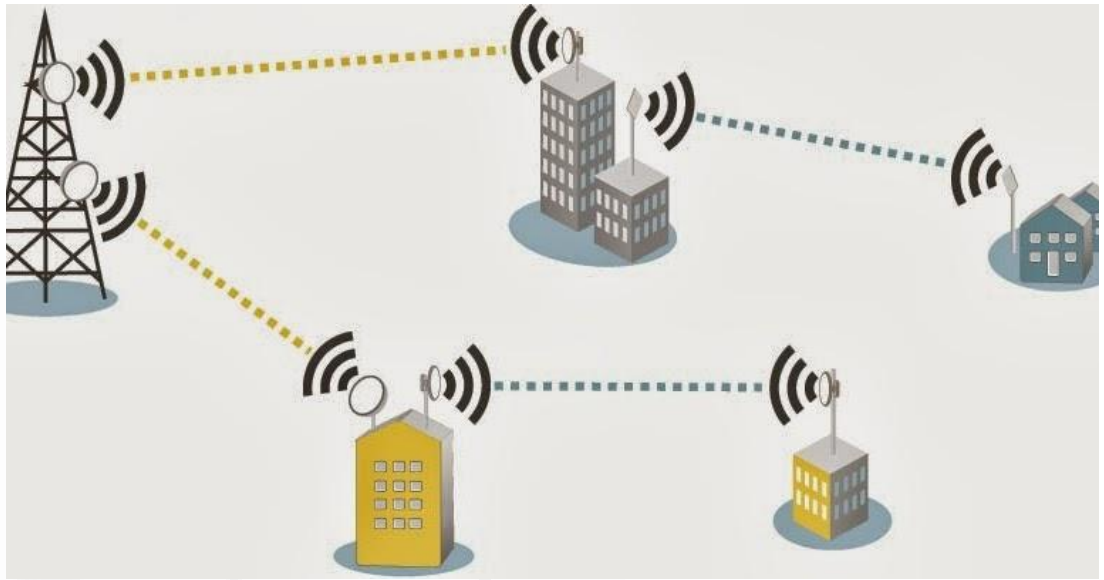


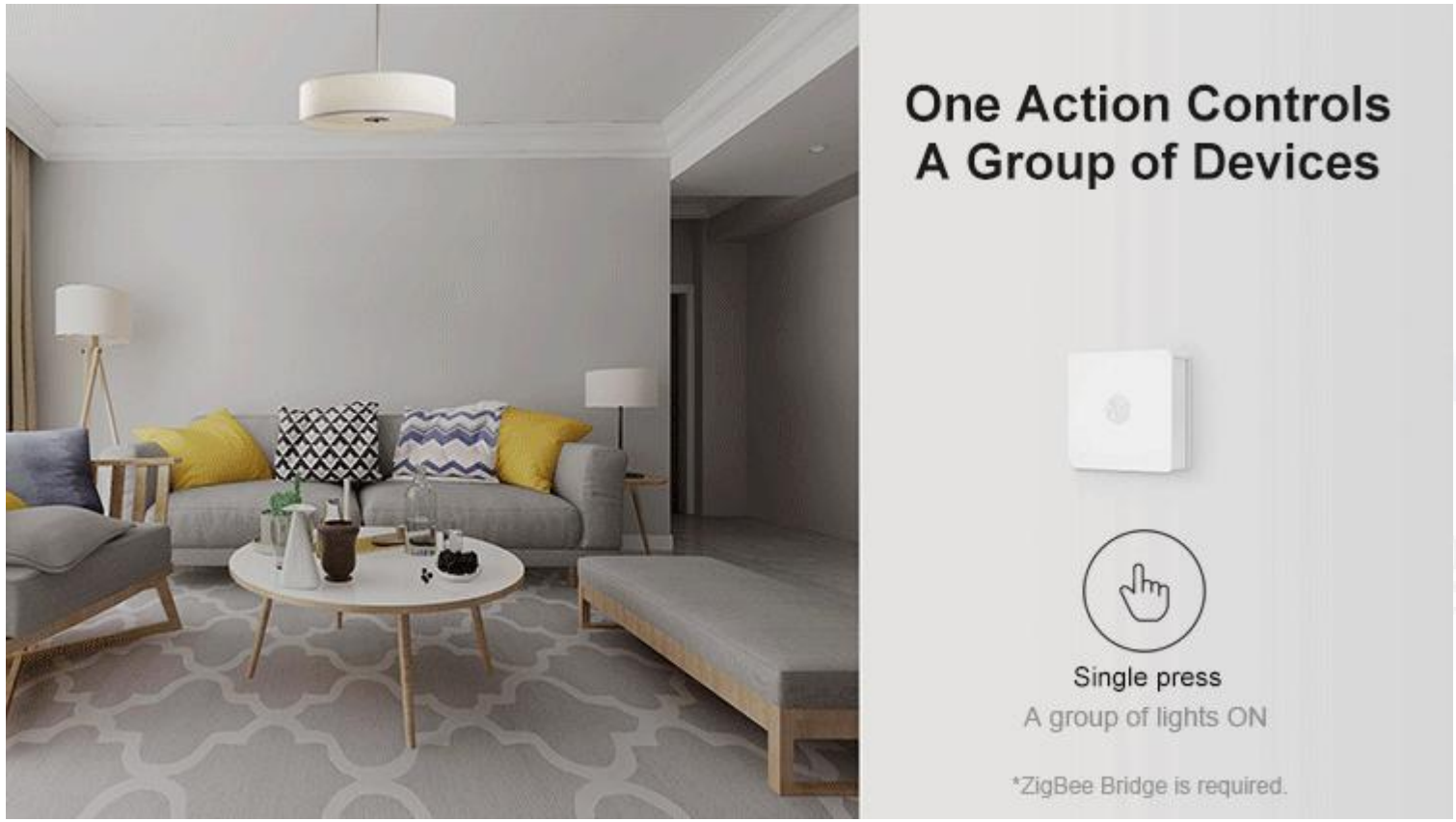

OBJECT

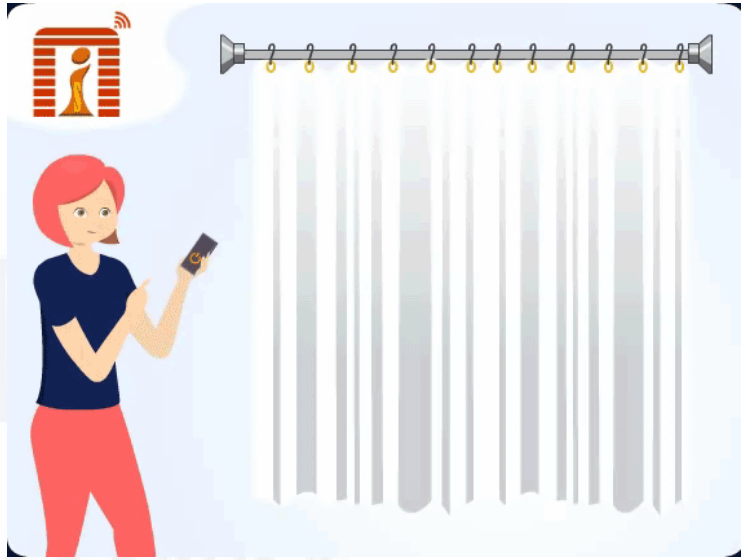


Transfer files wirelessly over the Bluetooth Low Energy protocol using BleuIO


802.11ax




One Action Controls
A Group of Devices
Single press
A group of lights ON
*ZigBee Bridge is required.



PARAMETERS	GSM	GPRS	UMTS
Abbreviation	Global System for Mobile Communication.	General Packet Radio Service.	Universal Mobile Telecommunications System.
Data rates	14.4 Kbps	57.6 Kbps	2 Mbps
Carrier Size	200 KHz TDMA	200 KHz	5 Mhz CDMA
System generation	2G	2.5G	3G
Based System	TDMA	GSM	GSM,GPRS
Users per channel	8	8	-
Type of connection	Circuit-Switched Technology.	Packet-Switched Technology.	Both Circuit & Packet-Switched Technology.
Frame Duration	4.615 ms.	4.615 ms.	10 ms.
Features	SMS	MMS	Video calls & TV applications.

Embedded Firmware – Key Points

- **Definition:** Program instructions & settings stored in an embedded system.

- **Development:**
  - **High-Level (C/C++)** – Easier, portable, IDE-supported.
  - **Assembly** – Processor-specific, harder to debug.

- **HEX File Creation:** **Converts code to machine-readable binary.**

- **Control Algorithms:**
  - **Super Loop:** Continuous execution (**while(1){}**).
  - **Task Scheduler:** RTOS-based task management.

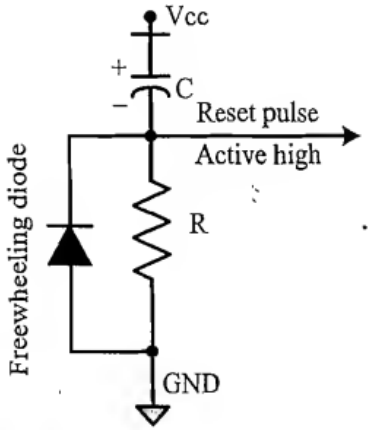
Embedded Software	Firmware
Acts similar to an OS on a computer	Not like an OS; performs only one specific function or purpose
Performs high-level functions of a device	Performs low-level functions of a device
Does not depend on hardware	Hardware dependent
Typically will have user or system interaction	Does not have user or system interaction

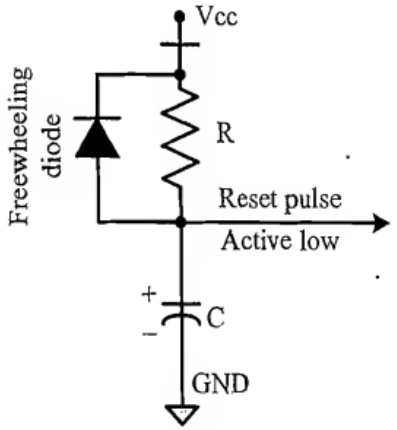
Applications of embedded firmware:

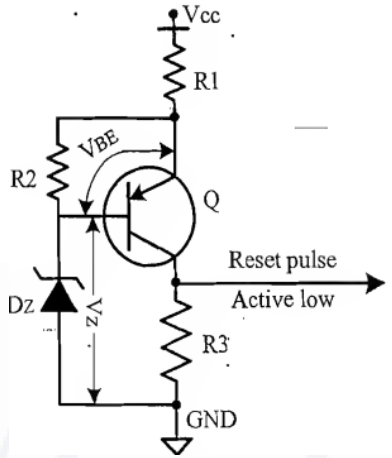
1. **Consumer Electronics**: Smartphones, smart TVs, wearables.
2. **Automotive Systems**: Engine control, infotainment, ADAS.
3. **Industrial Automation**: Robotics, factory machines, sensors.
4. **Healthcare Devices**: Medical monitoring devices, insulin pumps.
5. **IoT Devices**: Smart home systems, connected appliances, sensors.

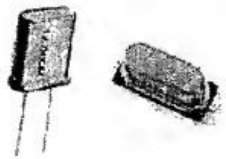
Other System Components – Key Points

- **Definition:** Essential **circuits/ICs** required for proper **functioning** of an embedded system.
- **Examples:**
  - **Watchdog Timer** – Prevents system hang-ups.
  - **Reset IC/Circuit** – Ensures proper system startup.
  - **Brown-out Protection** – Prevents malfunction during power drops.
- **Integration:** Some **controllers/SoCs** have these built-in; others require **external components**.
- **Additional Components: Level translators, specific function ICs**, and interface circuits as needed.

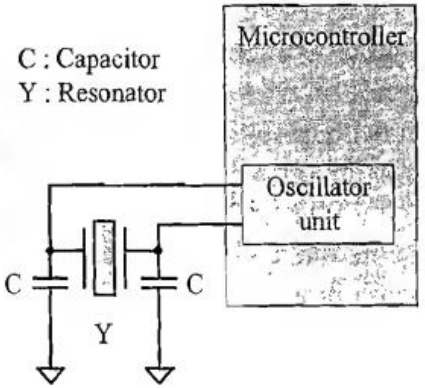


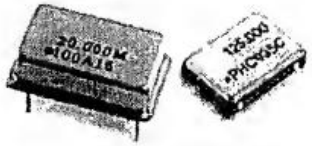




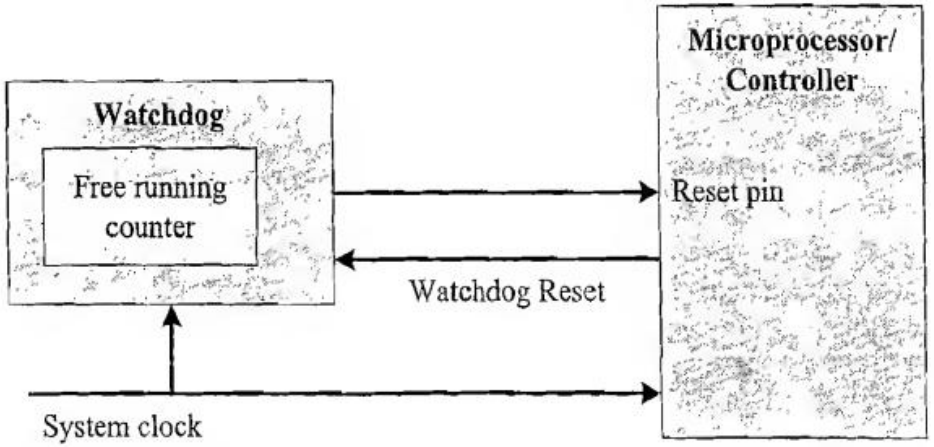


Quartz crystal resonator





Crystal oscillator



PCB and Passive Components – Key Points

- **Printed Circuit Board (PCB):** The **foundation** of an embedded system, designed based on schematics and used for **component mounting and firmware testing**.
- **Passive Components:** Essential **supporting elements** like **resistors, capacitors, diodes**, etc., ensuring stable operation.
  - **Example:** Regulator IC with **filter capacitors** for a stable, ripple-free power supply.

I/O Subsystem for Embedded Systems

1. Overview:

- The I/O subsystem enables interaction between the embedded system and the external world, typically through sensors and actuators connected to input and output ports.
- The sensors detect changes in the system environment or variables and send this information to the system. Actuators, on the other hand, help make necessary adjustments by acting on the output based on the system's commands.

1. **Input: Sensors**:

- Sensors are transducer devices that convert energy from one form to another, allowing for measurement or control.
- Examples include magnetic hall effect sensors used in a "smart" running shoe to measure the distance between the cushion and magnet.
- Sensors are typically interfaced with the system via signal conditioning or translation components like **ADCs (Analog-to-Digital Converters) or optocouplers.**

2. **Output: Actuators**:

- Actuators are devices that convert **signals into physical actions** (such as motion).
- An example is the use of a micro stepper motor in the smart running shoe to adjust the position of the cushioning element.

LED Interfacing:

1. LED Characteristics:

- **LED Type:** LED is a **p-n junction diode** with an **anode** and **cathode**.
- **Power Supply Connection:**
  - The **anode** should be connected to the **+ve** terminal of the supply voltage.
  - The **cathode** should be connected to the **-ve** terminal of the supply voltage.

- **Current Flowing Through the LED:**
  - The current through the LED needs to be **limited** to a value lower than the maximum current that the LED can handle.
  - A **resistor** is used in series to limit the current and prevent damage to the LED.

LED Interfacing Methods:

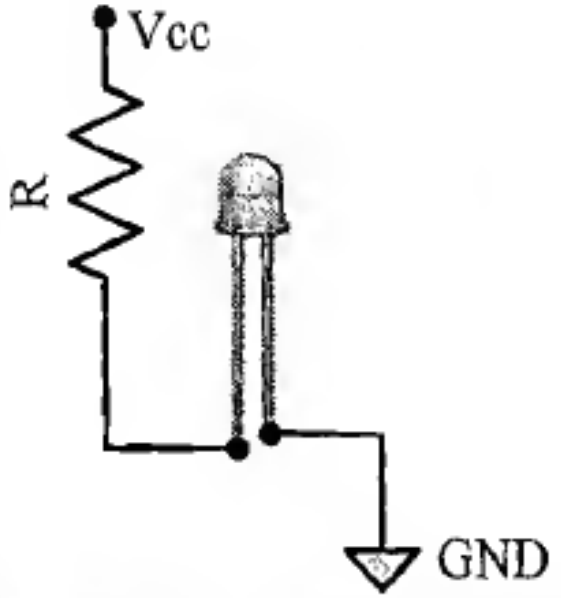
- **Method 1: Port Pin as Current Source**
  - In this method, the **port pin** of the processor/controller drives the LED.
  - The **port pin** is set to **logic high (1)** to source current through the LED.
- **Method 2: Port Pin as Current Sink**
  - Here, the LED is connected such that the **port pin** "sinks" the current.
  - The **port pin** is set to **logic low (0)** to sink current from the LED.

Brightness Consideration:

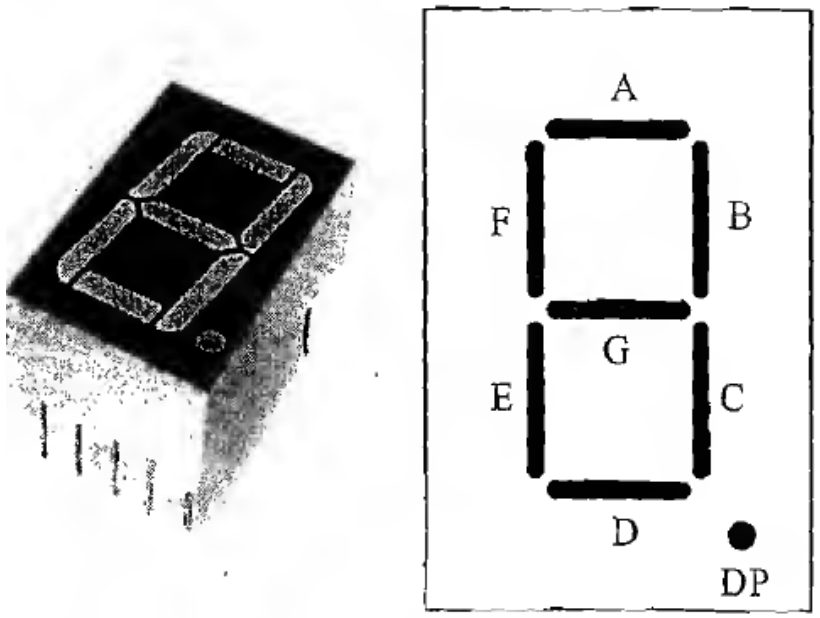
- In both cases, the LED's brightness is determined by the current flowing through it.
- If the port pin is used as a source, the current may not be sufficient for the required brightness, and the **resistor** ensures proper current flow.

Diagram of LED Interfacing:

- The diagram shows a simple LED interfacing circuit where:
  - The **resistor (R)** limits the current flowing through the LED.
  - The **Vcc** is the positive supply, and the **GND** is the ground.



7 Segment Display



7-Segment LED Display:

Overview:

- A 7-segment LED display is a type of output device used to **display alpha-numeric characters.**
- It contains 8 LED segments arranged in a figure-eight pattern.
- Out of the 8 segments, 7 are used to display alpha-numeric digits, and the remaining segment is used to display the "decimal point."

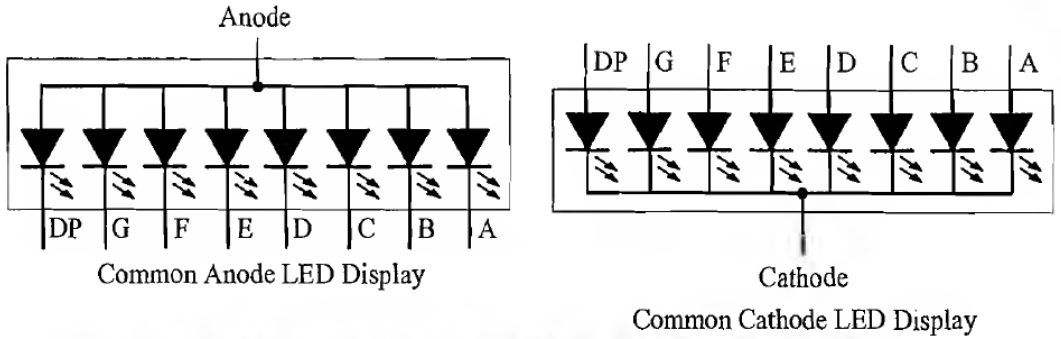
- **LED Segments:**
  - The 7 segments are named A, B, C, D, E, F, G, with DP representing the decimal point.
  - For each character or digit, different combinations of these segments are lit.
  - For example:
    - To display the number 4, segments F, G, B, C, and D are lit.
    - **For displaying the number 3, segments A, B, C, D, and G are lit.**

- **Configuration:**
  - **There are two common configurations for the 7-segment display:**
    - **Common Anode Configuration**: In this configuration, all the anodes of the 8 LED segments are connected to the power supply. The cathodes of each segment are connected to the port pins of a processor/controller.
    - **Common Cathode Configuration**: In this setup, all the cathodes of the 8 LED segments are connected to the ground. The anodes are connected to the processor/controller's port pins.
  - The display uses either common anode or common cathode configuration based on the LED's power supply setup.

- **Connection:**
  - Each segment is connected to a port pin of the processor/controller through a current-limiting resistor to protect the LED from excessive current flow.
  - The logic for turning a segment on is:
    - **For common anode**: A segment is lit when the port pin is set to logic 0.
    - **For common cathode:** A segment is lit when the port pin is set to logic 1.
- **Usage:**
  - These displays are commonly used in embedded systems that require alpha-numeric data display, such as in public telephone call monitoring devices and point of sale terminals.

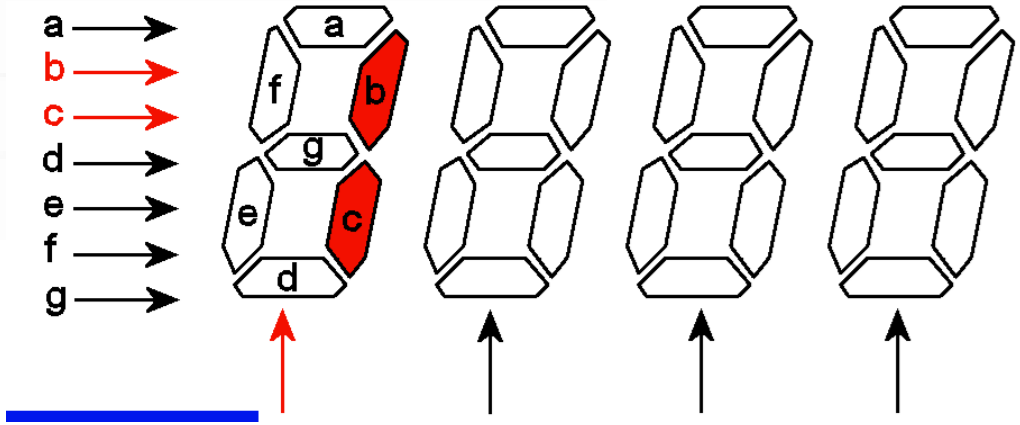
Current Limiting Resistor:

- The current flowing through each LED segment should be limited to the maximum safe value (typically around 20mA). This is achieved by using a current-limiting resistor.
- The value of the resistor can be calculated based on the parameters provided by the LED manufacturer, particularly considering the supply voltage and the LED's forward voltage.

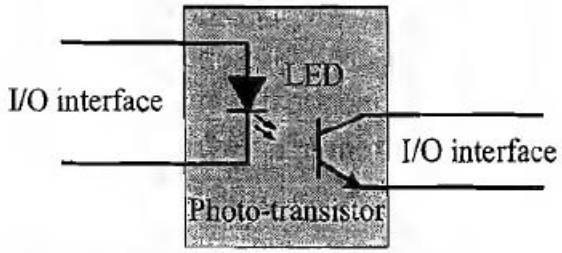


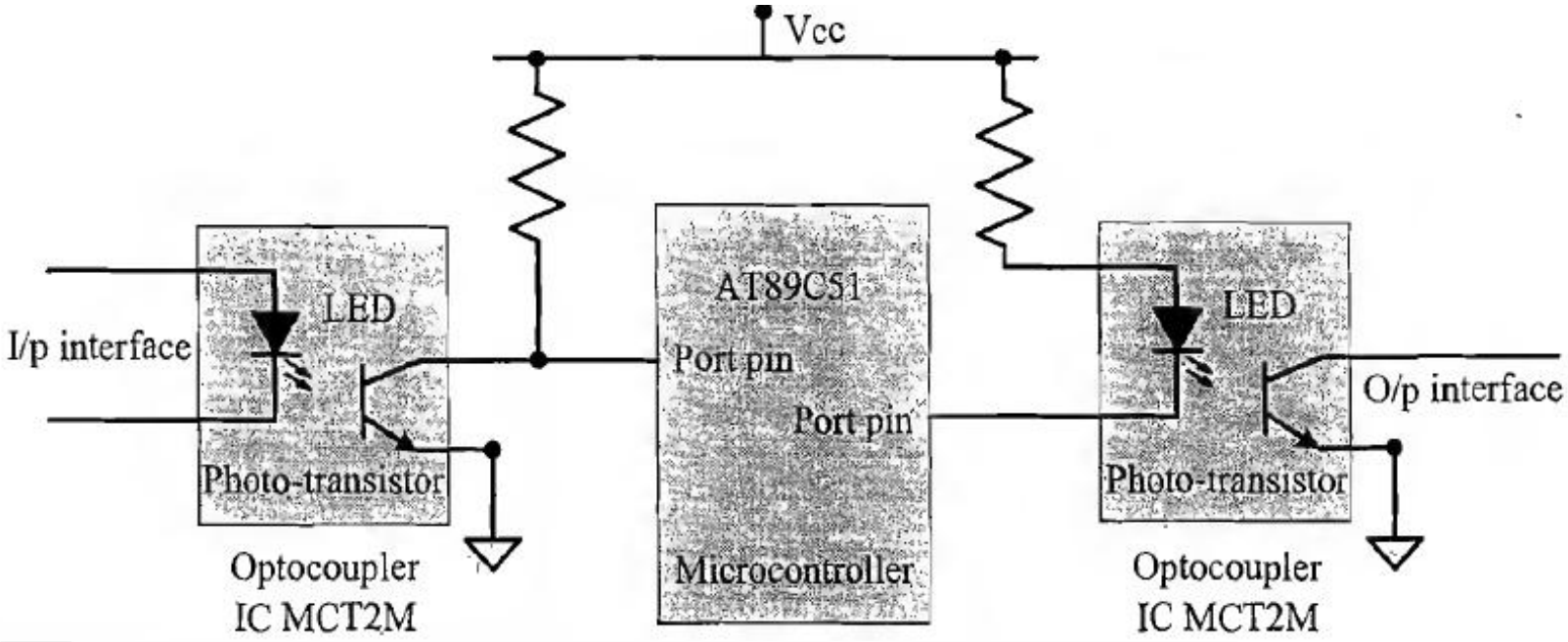
Applications:

- The 7-segment display is an important part of embedded systems for visual output, commonly seen in devices like digital clocks, calculators, and other digital displays in various consumer electronics.

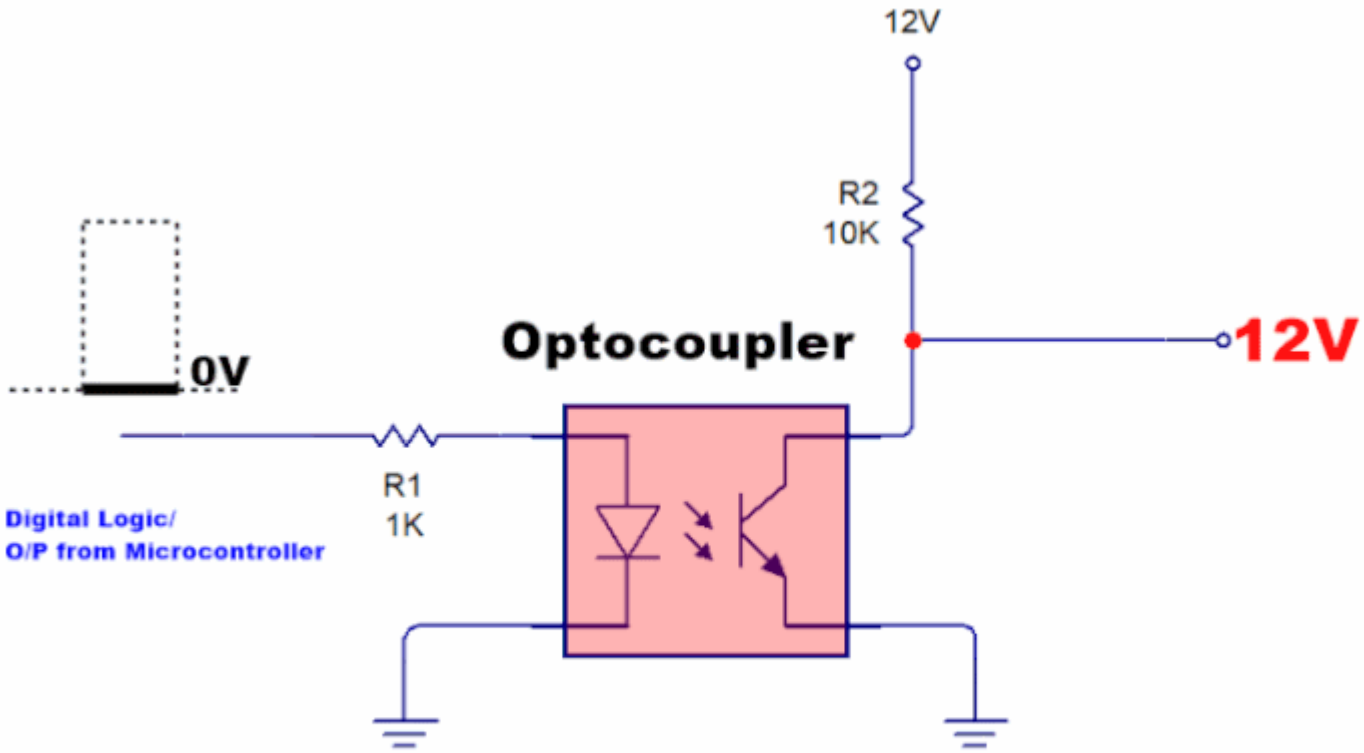


Opto Coupler





Opto Coupler



An **Optocoupler** is a solid-state device used to isolate two parts of a circuit while transferring electrical signals between them. **It integrates an LED and a phototransistor within the same housing.** The LED in the optocoupler is responsible for emitting light when powered, and the light is received by the phototransistor, which produces an output current proportional to the light intensity.

Brief explanation of its use and configuration:

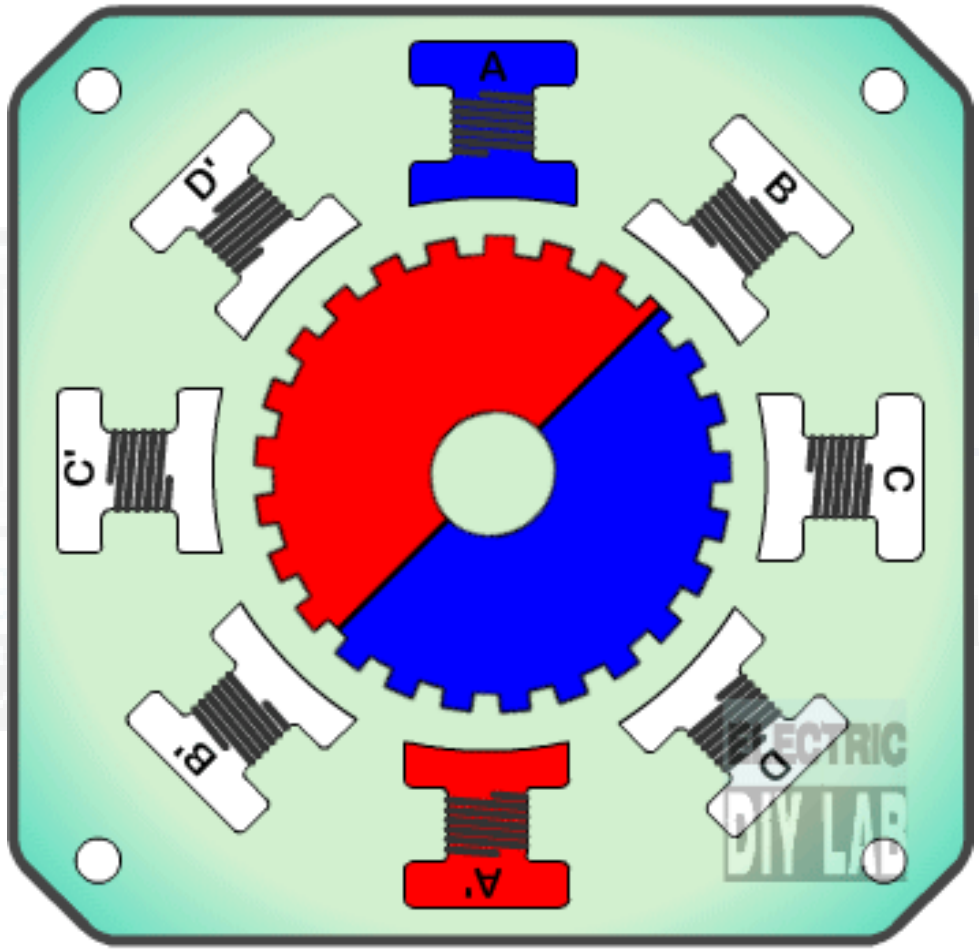
1. **Working Principle**: The LED in the optocoupler receives an electrical signal from one side of the circuit and emits light.
2. This light activates the phototransistor on the other side, which then passes the signal onto the next circuit. This isolates the two circuits electrically but allows signal transmission through light.

1. **Applications**: Optocouplers are widely used in electronic circuits for:
   - **Signal Isolation**: Prevents high-voltage spikes from one part of the circuit affecting other parts.
   - **Protection**: Helps protect sensitive components, like microcontrollers or CPUs, from voltage surges or spikes.
   - **Noise Suppression**: Used to eliminate noise in the signal transmission.
   - **Voltage Isolation**: Ensures there's no direct electrical connection between the two sides of the circuit, which is crucial in high-voltage applications.
2. **Common Uses**:
   - Used in power supply circuits, microcontrollers, and communication systems.
   - Common in industrial control systems, switching power supplies, and interfacing different parts of embedded systems that require isolation.

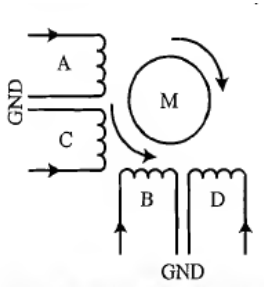
A **stepper motor** is an electromechanical device that moves in discrete steps when driven by electrical pulses. Unlike a DC motor, which rotates continuously when powered, a stepper motor produces precise and controlled rotation. These motors are commonly used in applications like robotics, industrial systems, and printers.

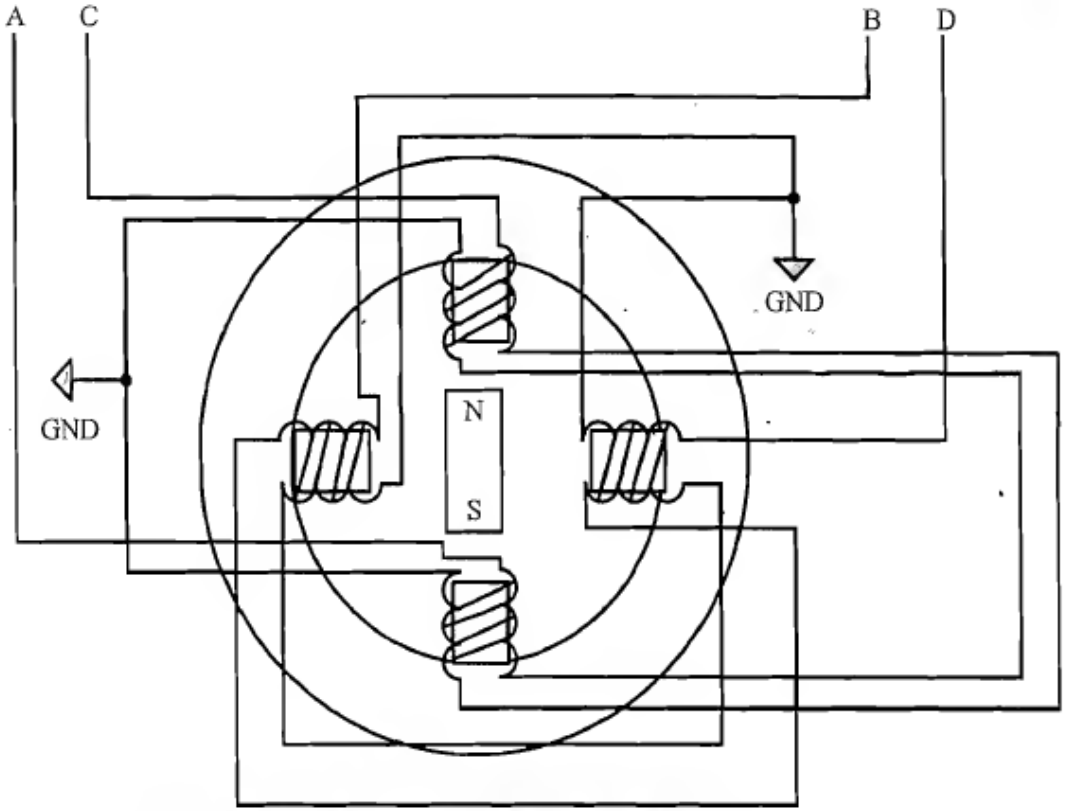
Types of Stepper Motors:

1. **Unipolar Stepper Motor**:
   - Has two windings per phase.
   - The rotation is controlled by reversing the current direction through each coil.
   - A simple way to change the direction of rotation is by switching the coil connections.
2. **Bipolar Stepper Motor**:
   - Has a single winding per phase.
   - The direction of rotation is controlled by reversing the current in the windings.
   - Requires more complex circuitry than unipolar motors.


A
B
C
D
A'
B'
C'
D'

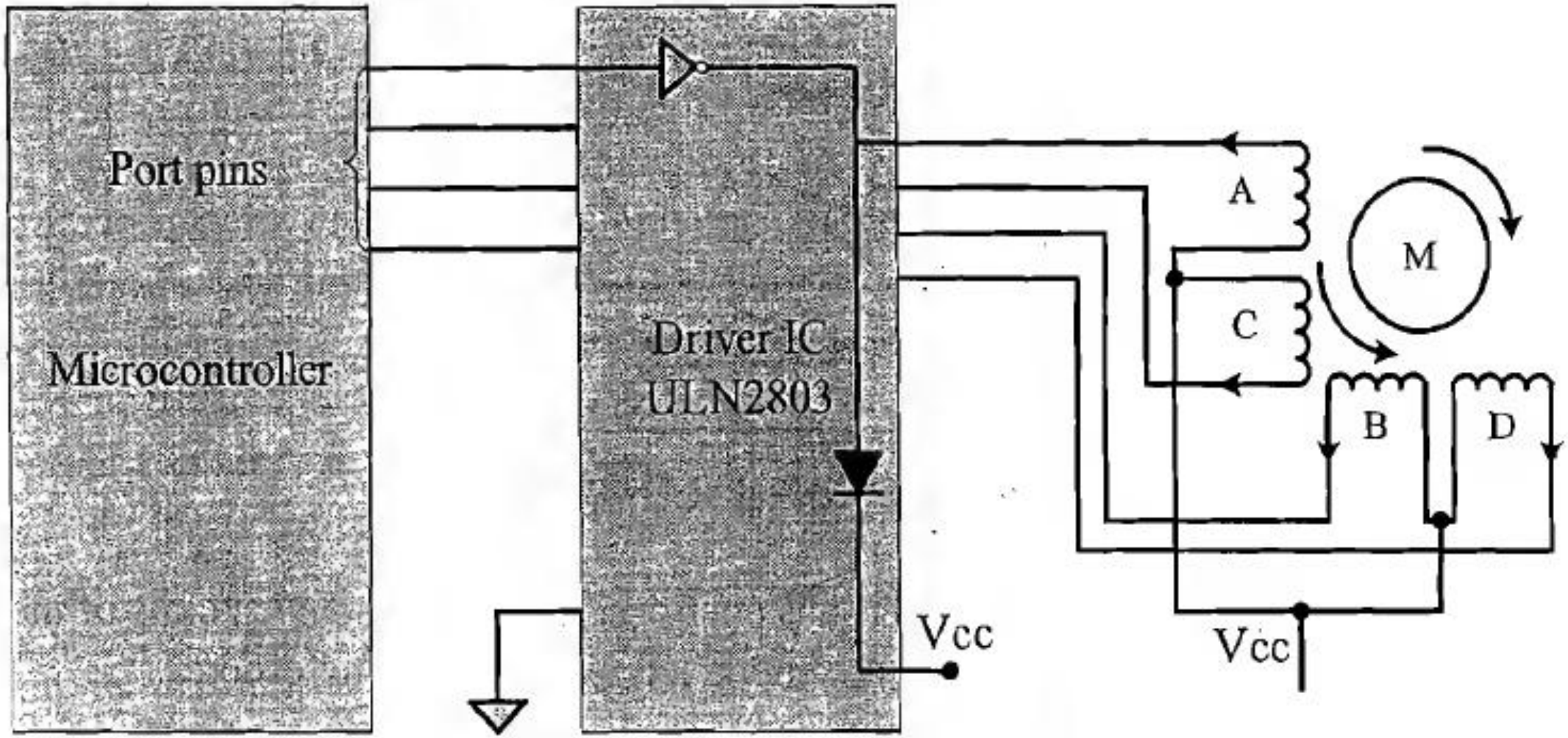
Stepper Motor





StatorWinding details for a 2 Phase unipolar stepper motor

Interfacing of stepper motor through driver circuit



Stepping Modes:

- **Full Step**: In this mode, both phases are energized at the same time. This results in movement in discrete steps, and the coils (A, B, C, D) are energized in a specific sequence to move the motor.
- **Wave Step**: Only one phase is energized at a time, and each coil is alternately powered. This mode is less stable than full step but saves energy.
- **Half Step**: Combines both wave and full step modes, offering a balance of torque and stability.

Motor Control:

To reverse the direction of the motor, the order in which the coils are energized can be altered.

Stepper Motor Interfacing:

- Stepper motors require a higher current than microcontroller pins can provide directly. Thus, **driver circuits** are used to interface the motor with the microcontroller.
- **ICs like ULN2803** can drive a 5V stepper motor effectively.
- Alternatively, simple circuits using **transistors** can be used to control the motor.

Full Step Energizing Sequence:

In the full step mode, both phases are energized simultaneously.
The coils A, B, C, and D are energized in the following order:

Step	Coil A	Coil B	Coil C	Coil D
1	High	Low	Low	Low
2	Low	High	Low	Low
3	Low	Low	High	Low
4	Low	Low	Low	High

In each step, one coil from each phase is energized, which leads to rotation in one direction. To reverse the rotation, the energizing order is reversed.

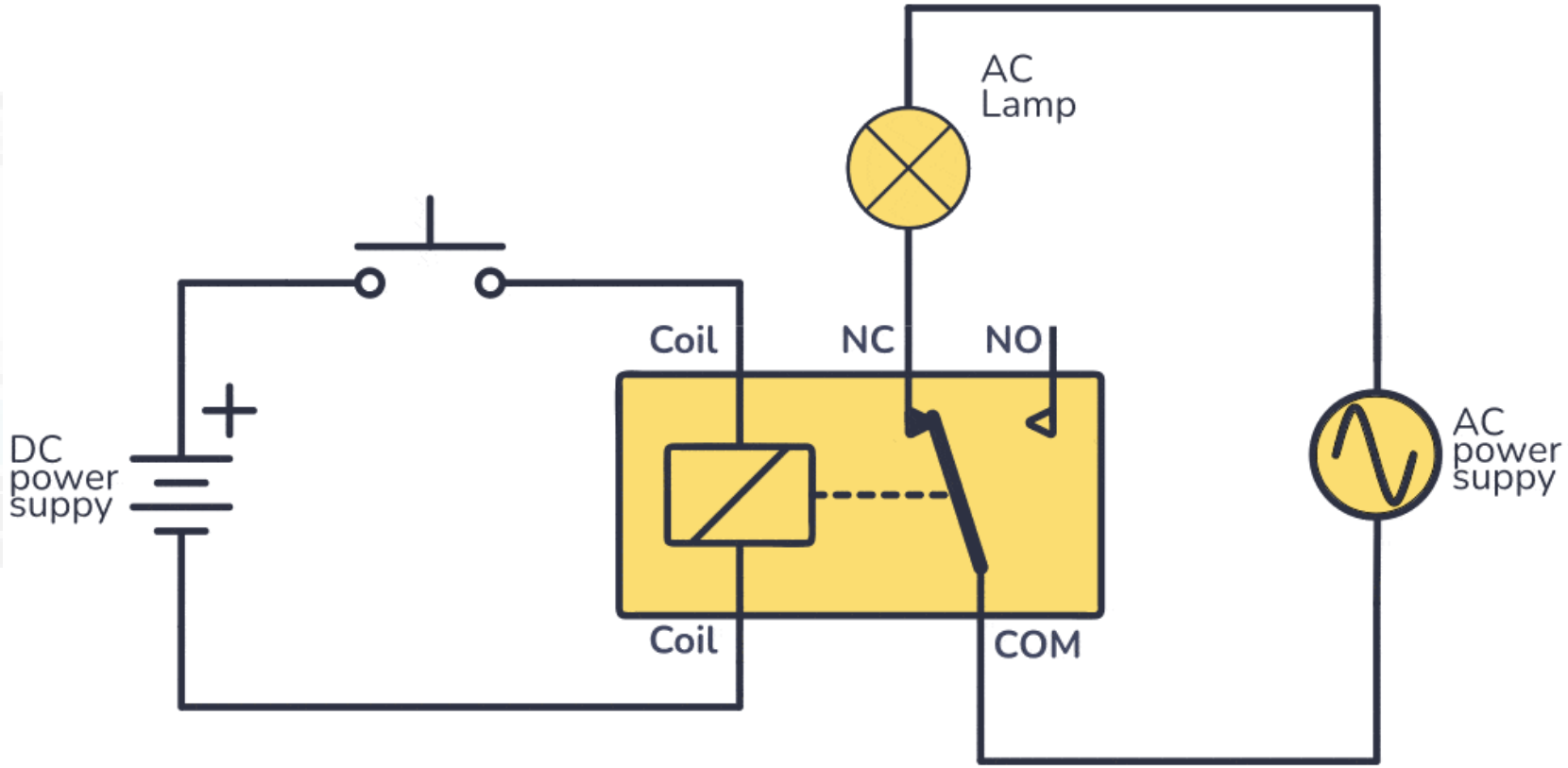
Half Step Energizing Sequence:

In the half step mode, a combination of wave and full step is used. The energizing sequence is as follows:

In half-step mode, each phase alternates between being powered individually and being powered with the other phase. This increases the resolution and torque compared to wave stepping, giving the motor smoother movement.

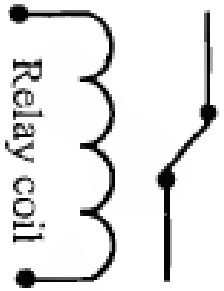
Step	Coil A	Coil B	Coil C	Coil D
1	High	Low	Low	Low
2	High	Low	Low	High
3	Low	Low	High	High
4	Low	High	High	High
5	Low	High	High	Low
6	Low	High	Low	Low
7	High	High	Low	Low
8	High	Low	Low	Low

Relay

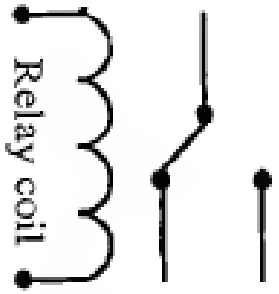


Relay

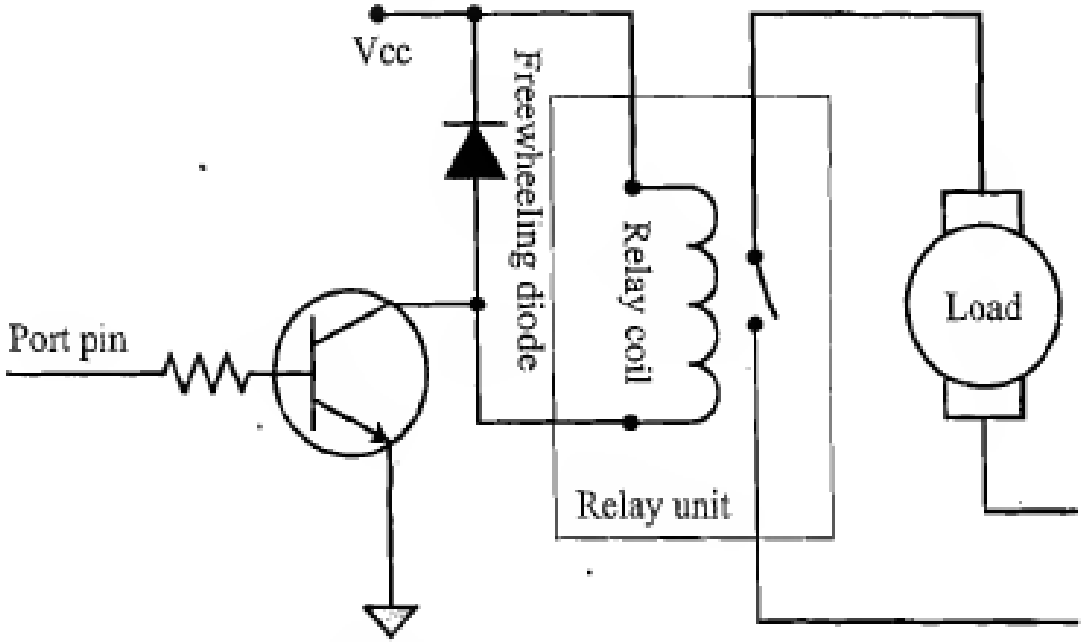
Single pole single throw normally open



Single pole single throw normally closed



Single pole double throw



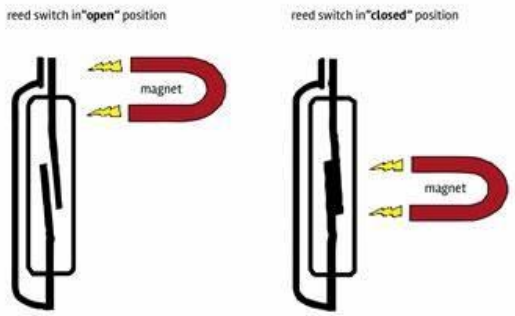
A **relay** is an electro-mechanical device used to control power or signals in embedded systems. It consists of a **coil** (made of insulated wire) and a **metal armature** with one or more contacts.

Working Principle:

When a voltage is applied to the relay coil, it creates a magnetic field that attracts the armature, changing the position of the contacts. This action opens or closes the circuit, allowing or interrupting the flow of current or signal.

Relay Configurations:

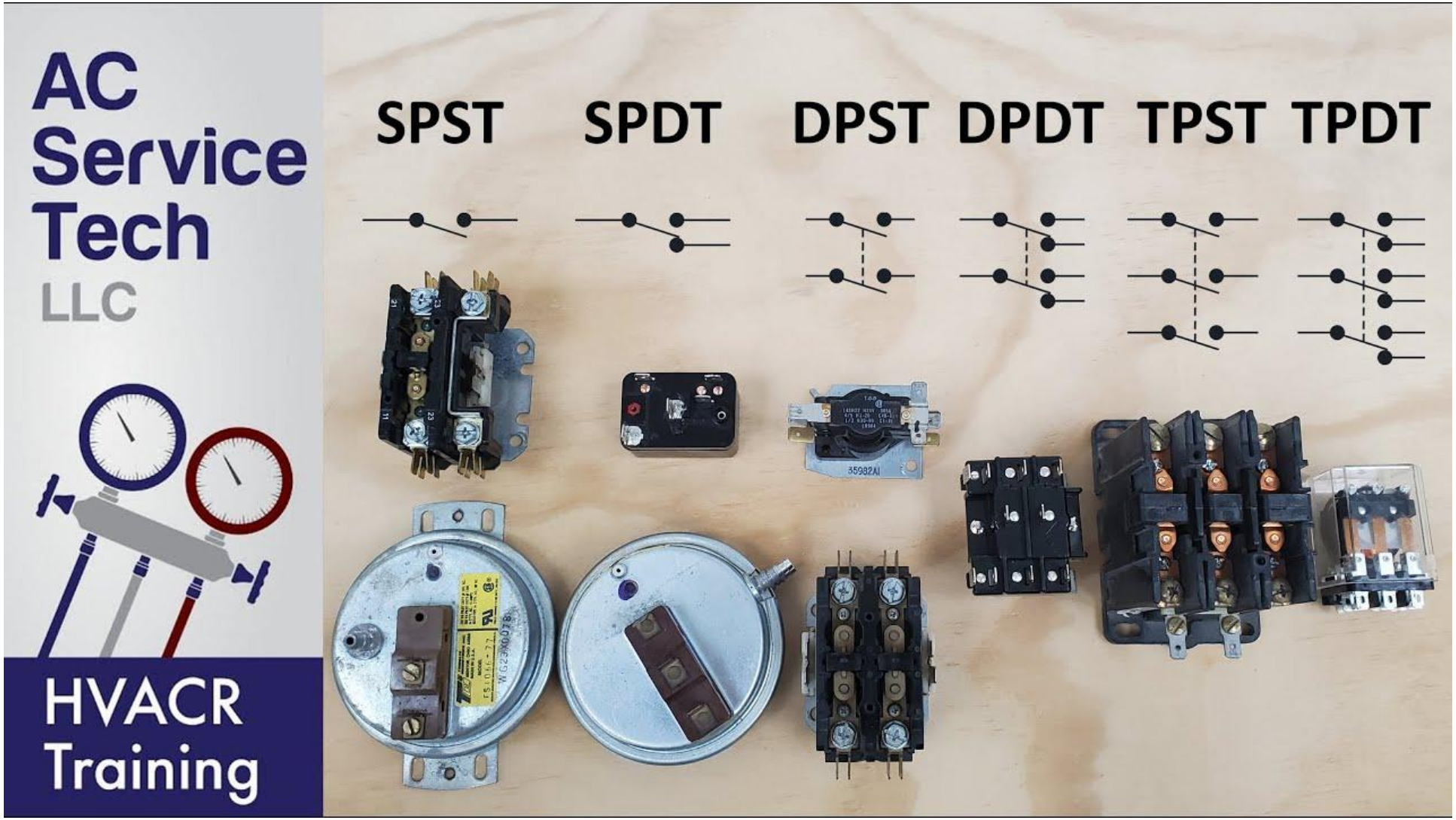
1. **Single Pole Single Throw (SPST)**:
   - **Normally Open (NO)**: The circuit is open until the relay is energize closing the path.
   - **Normally Closed (NC)**: The circuit is closed until the relay energized, opening the path.



2. **Single Pole Double Throw (SPDT)**:
   - Has two paths for signal flow, selected by energizing or de-energizing the relay.

Relays are typically controlled by a **relay driver circuit** using a **transistor**. A **free-wheeling diode** is used to protect the relay and transistor by absorbing voltage spikes when the coil is de-energized.

For low-voltage applications, **Reed relays** are used, as they are small and efficient in switching low DC signals.

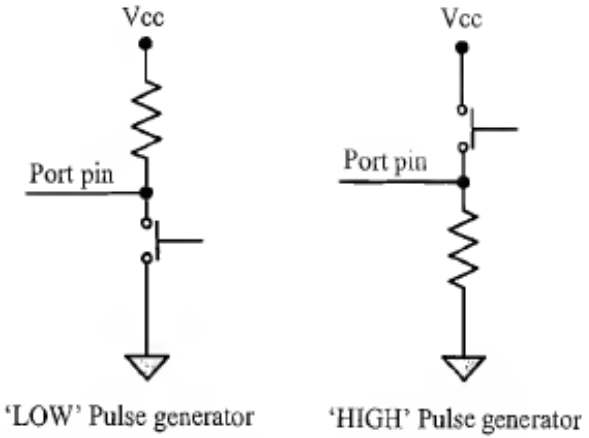

AC
Service
Tech
LLC
HVACR
Training
SPST
SPDT
DPST
DPDT
TPST
TPDT

Piezo Buzzer:

A **piezo buzzer** is a piezoelectric device that generates sound when voltage is applied to it.

- **Self-driving buzzers** generate a fixed tone without external components.
- **External driving buzzers** allow for tone variation by applying a pulse train.
- They can be directly interfaced to a microcontroller, but for higher current requirements, a transistor-based driver circuit can be used.

Push Button Switch:



A **push button** is an input device that can be configured as:

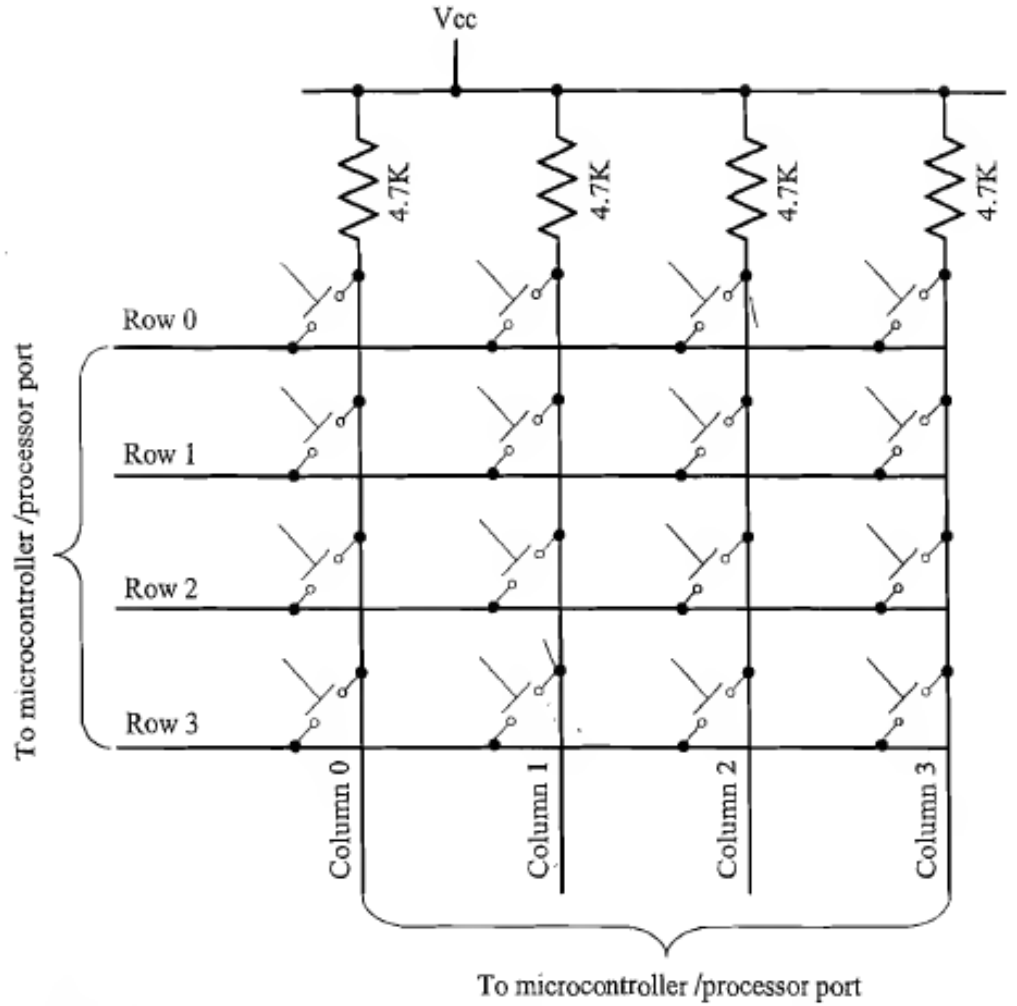
- **Push to Make**: Normally open, closes the circuit when pressed.
- **Push to Break**: Normally closed, opens the circuit when pressed. It is commonly used for generating momentary pulses, such as a reset or start signal in embedded systems. The push button is typically connected to the microcontroller and can generate either a **HIGH** or **LOW** pulse based on how it is interfaced.

Keyboard Interfacing

A keyboard is an input device used for user interfacing. For a limited number of keys, push buttons can be used directly with port pins. However, for a large number of keys (e.g., an alphanumeric keypad), a matrix keyboard is a more efficient solution, reducing the number of required connections.

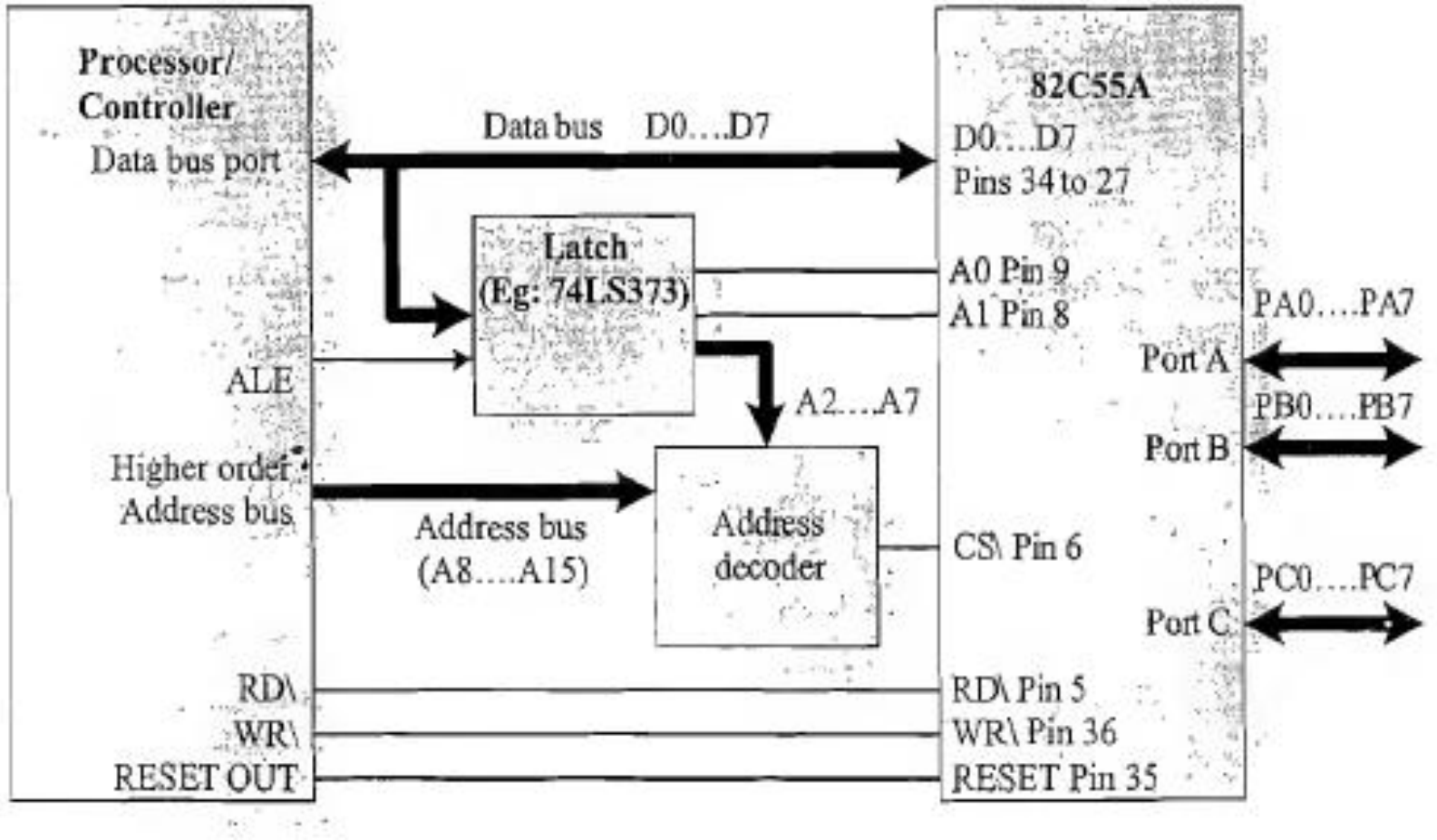
- Matrix Keyboard: The keys are arranged in a row and column matrix. For example, a 4x4 matrix for 16 keys requires only 8 lines instead of 16 port pins.
- Scanning Technique: The keyboard scans the rows one by one. For each row, the columns are read. When a key is pressed, it connects the row and column, generating a logic 0 in the corresponding column.
- Debouncing: Mechanical keys can cause multiple inputs for a single press due to contact bounce. Software debouncing solves this by re-reading the key after a short delay, confirming the press. Hardware debouncing can also be used for more reliable results.

Pull-up resistors are used in the columns to limit the current during key presses.


Vcc
4.7K
4.7K
4.7K
4.7K
Row 0
Row 1
Row 2
Row 3
To microcontroller /processor port
Column 0
Column 1
Column 2
Column 3
To microcontroller /processor port

A **Programmable Peripheral Interface (PPI)** is used to extend the I/O capabilities of processors/controllers, especially when the built-in I/O ports are insufficient for the application. A commonly used PPI device is the **8255A**, which is designed for 8-bit processors/controllers.

- **8255A** provides 24 I/O pins, which can be grouped into:
  - **Three 8-bit parallel ports** (Port A, Port B, and Port C)
  - **Two 8-bit parallel ports** (Port A and Port B), with Port C being configurable as:
    1. **8 individual I/O pins** (Port C as a general I/O port)
    2. **Two 4-bit ports**: **Port C Upper (Cy)** and **Port C Lower (CL)**


Processor/
Controller
Data bus port
Data bus D0....D7
82C55A
D0....D7
Pins 34 to 27
Latch
(Eg: 74LS373)
A0 Pin 9
A1 Pin 8
PA0....PA7
Port A
PB0....PB7
Port B
ALE
A2....A7
Higher order
Address bus
Address bus
(A8....A15)
Address
decoder
CS\ Pin 6
PC0....PC7
Port C
RD\
WR\
RESET OUT
RD\ Pin 5
WR\ Pin 36
RESET Pin 35

Thank You