DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE MATERIAL

SUBJECT: CS3691 - EMBEDDED SYSTEMS AND IOT

CS3691 Embedded Systems and IoT Syllabus

UNIT I 8-BIT EMBEDDED PROCESSOR

8-Bit Microcontroller – Architecture – Instruction Set and Programming – Programming Parallel Ports – Timers and Serial Port – Interrupt Handling.

UNIT II EMBEDDED C PROGRAMMING

Memory And I/O Devices Interfacing – Programming Embedded Systems in C – Need For RTOS –Multiple Tasks and Processes – Context Switching – Priority Based Scheduling Policies.

UNIT III IOT AND ARDUINO PROGRAMMING

Introduction to the Concept of IoT Devices – IoT Devices VersusComputers – IoT Configurations – Basic Components – Introduction to Arduino – Types of Arduino– Arduino Toolchain – Arduino Programming Structure – Sketches – Pins – Input/Output From Pins Using Sketches – Introduction to Arduino Shields – Integration of Sensors and Actuators with Arduino.

UNIT IV IOT COMMUNICATION AND OPEN PLATFORMS

IoT Communication Models and APIs – IoT Communication Protocols – Bluetooth – WiFi – ZigBee– GPS – GSM modules – Open Platform (like Raspberry Pi) – Architecture – Programming –Interfacing – Accessing GPIO Pins – Sending and Receiving Signals Using GPIO Pins – Connecting to the Cloud.

UNIT V APPLICATIONS DEVELOPMENT

Complete Design of Embedded Systems – Development of IoT Applications – Home Automation –Smart Agriculture – Smart Cities – smart Healthcare.

TEXTBOOKS

1. Muhammed Ali Mazidi, Janice Gillispie Mazidi, Rolin D. McKinlay, “The 8051Microcontroller and Embedded Systems”, Pearson Education, Second Edition, 2014.

2. Robert Barton, Patrick Grossetete, David Hanes, Jerome Henry, Gonzalo Salgueiro, “IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things”, CISCO Press, 2017.

REFERENCES

1. Michael J. Pont, “Embedded C”, Pearson Education, 2007.
2. Wayne Wolf, “Computers as Components: Principles of Embedded Computer System Design”, Elsevier, 2006.
3. Andrew N Sloss, D. Symes, C. Wright, “Arm System Developer’s Guide”, Morgan Kauffman/ Elsevier, 2006.
4. Arshdeep Bahga, Vijay Madisetti, “Internet of Things – A hands-on approach”, Universities Press, 2015

UNIT – 1

8-BIT EMBEDDED PROCESSOR

What is a Microprocessor?

Computer's Central Processing Unit (CPU) built on a **single Integrated Circuit (IC)** is called a **microprocessor**.

A digital computer with one microprocessor which acts as a CPU is called microcomputer.

It is a programmable, multipurpose, clock -driven, register-based electronic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions and provides results as output.

The microprocessor contains millions of tiny components like transistors, registers, and diodes that work together.

Evolution of Microprocessors

We can categorize the microprocessor according to the generations or according to the size of the microprocessor:

First Generation (4 - bit Microprocessors)

The first generation microprocessors were introduced in the year 1971-1972 by Intel Corporation. It was named **Intel 4004** since it was a 4-bit processor.

It was a processor on a single chip. It could perform simple arithmetic and logical operations such as addition, subtraction, Boolean OR and Boolean AND.

I had a control unit capable of performing control functions like fetching an instruction from storage memory, decoding it, and then generating control pulses to execute it.

Second Generation (8 - bit Microprocessor)

The second generation microprocessors were introduced in 1973 again by Intel. It was a first 8 - bit microprocessor which could perform arithmetic and logic operations on 8-bit words. It was Intel 8008, and another improved version was Intel 8088.

Third Generation (16 - bit Microprocessor)

The third generation microprocessors, introduced in 1978 were represented by **Intel's 8086, Zilog Z800 and 80286**, which were 16 - bit processors with a performance like minicomputers.

Fourth Generation (32 - bit Microprocessors)

Several different companies introduced the 32-bit microprocessors, but the most popular one is the **Intel 80386**.

Fifth Generation (64 - bit Microprocessors)

From 1995 to now we are in the fifth generation. After 80856, Intel came out with a new processor namely Pentium processor followed by **Pentium Pro CPU**, which allows multiple CPUs in a single system to achieve multiprocessing.

Other improved 64-bit processors are **Celeron, Dual, Quad, Octa Core processors**.

Basic Terms used in Microprocessor

Here is a list of some basic terms used in microprocessor:

Instruction Set - The group of commands that the microprocessor can understand is called Instruction set. It is an interface between hardware and software.

Bus - Set of conductors intended to transmit data, address or control information to different elements in a microprocessor. A microprocessor will have three types of buses, i.e., data bus, address bus, and control bus.

IPC (Instructions Per Cycle) - It is a measure of how many instructions a CPU is capable of executing in a single clock.

Clock Speed - It is the number of operations per second the processor can perform. It can be expressed in megahertz (MHz) or gigahertz (GHz). It is also called the Clock Rate.

Bandwidth - The number of bits processed in a single instruction is called Bandwidth.

Word Length - The number of bits the processor can process at a time is called the word length of the processor. 8-bit Microprocessor may process 8 -bit data at a time. The range of word length is from 4 bits to 64 bits depending upon the type of the microcomputer.

Data Types - The microprocessor supports multiple data type formats like binary, ASCII, signed and unsigned numbers.

Working of Microprocessor

The microprocessor follows a sequence to execute the instruction: Fetch, Decode, and then Execute. Initially, the instructions are stored in the storage memory of the computer in sequential order. The microprocessor fetches those instructions from the stored area (memory), then decodes it and executes those instructions till STOP instruction is met. Then, it sends the result in binary form to the output port. Between these processes, the register stores the temporary data and ALU (Arithmetic and Logic Unit) performs the computing functions.

MICROCONTROLLERS – Overview

In 1981, Intel introduced an 8-bit microcontroller called the 8051. It was referred as system on a chip because it had 128 bytes of RAM, 4K byte of on-chip ROM, two timers, one serial port, and 4 ports (8-bit wide), all on a single chip.

What is a Microcontroller?

- A **microcontroller** is a small and low-cost microcomputer, which is designed to perform the specific tasks of embedded systems like displaying microwave's information, receiving remote signals, etc.
- The general microcontroller consists of the processor, the memory (RAM, ROM, EPROM), Serial ports, peripherals (timers, counters), etc.

Criteria for Choosing Microcontroller

While choosing a microcontroller, make sure it meets the task at hand and that it is cost effective. We must see whether an 8-bit, 16-bit or 32-bit microcontroller can best handle the computing needs of a task. In addition, the following points should be kept in mind while choosing a microcontroller −

Speed − What is the highest speed the microcontroller can support?

Packaging − Is it 40-pin DIP (Dual-inline-package) or QFP (Quad flat package)? This is important in terms of space, assembling, and prototyping the end-product.

Power Consumption − This is an important criteria for battery-powered products.

Amount of RAM and ROM on the chip.

Count of I/O pins and Timers on the chip.

Cost per Unit − This is important in terms of final cost of the product in which the microcontroller is to be used.

Difference between Microprocessor and Microcontroller

The following table highlights the differences between a microprocessor and a microcontroller –

Microcontroller	Microprocessor
Microcontrollers are used to execute a single task within an application.	Microprocessors are used for big applications.
Its designing and hardware cost is low.	Its designing and hardware cost is high.
Easy to replace.	Not so easy to replace.

It is built with CMOS technology, which requires less power to operate.	Its power consumption is high because it has to control the entire system.
It consists of CPU, RAM, ROM, I/O ports.	It doesn't consist of RAM, ROM, I/O ports. It uses its pins to interface to peripheral devices.

Features of 8051:

4KB on-chip program memory (ROM/EPROM).
128 bytes on-chip data memory.
Four register banks.
64KB each program and external RAM addressability.
One microsecond instruction cycle with 12MHz crystal.
32 bidirectional I/O lines organized as four 8-bit ports.
Multiple modes, high-speed programmable serial port (UART).
16-bit Timers/Counters.
Direct byte and bit addressability.

Applications of 8051 Microcontroller

Some of the applications of 8051 is mainly used in daily life & industrial applications also some of that applications are shown below

Light sensing and controlling devices
Temperature sensing and controlling devices
Fire detections and safety devices
Automobile applications
Defense applications

8051 Microcontroller Applications in Embedded Systems

The applications of 8051 microcontroller involves in 8051 based projects. The list of 8051 projects is listed below.

Arduino Managed High Sensitive LDR based Power Saver for Street Light Control System
The Temperature Humidity Monitoring System of Soil Based on Wireless Sensor Networks using Arduino
RFID based Electronic Passport System for Easy Governance using Arduino
Arduino based RFID Sensed Device Access
Arduino based DC Motor Speed Control
Arduino Based Line Following Robot
Zigbee based Automatic Meter Reading System

Block Diagram of 8051:

Registers in microcontrollers are mainly used to store data and short-term instructions which are mainly used to process addresses to fetch data. This microcontroller includes 8-bit registers which have 8-bit start from D0 to D7. Here, D0 to D7 is LSB (least significant bit) and D7 is the most significant bit (MSB).

To make the data process better than 8-bit, then it must be separated into eight different bit parts. It includes several registers however general-purpose type registers are frequently available to programmers. There are classified into two types like General purpose & Special purpose. So, most of the general-purpose registers are listed below.

An accumulator is mainly used to execute arithmetic & logic instructions.

Registers like B, R0 toR7 are used for storing instruction addresses & data.

Data Pointers or DPTR is used to allow & process data in dissimilar addressing modes. This register includes DPH (high byte) & a DPL (low byte) which is mainly used to hold a 16-bit address. So, it

can be used as a base register within not direct jumps, lookup table instructions & external data transfer.

Program counter or PC is a 16-bit register used to store the next instruction's address to be performed

These registers are 8-bits other than program counter & data pointer registers.

Program Status Word

The term PSW stands for Program status word and it is one kind of register in the microcontroller. It is also called a flag register, used to demonstrate the position of arithmetic logic instructions such as zero carry bit, carry bit, etc. PSW or flag register is an 8-bit register where 6-bits are used. This register includes 8-flags where these flags are known as conditional flags. These flags will perform instruction simply if the condition is satisfied.

These conditional flags are overflow, parity, auxiliary carry & carry. The Program status word registers bit numbers like 3 & 4 are used to alter the bank registers whereas 1 & 5 are not used but they can be used by the programmer for executing a specific task.

MSB LSB

CY	AC	F0	RS1	RS0	OV	—	P

BIT	SYMBOL	FUNCTION
PSW.7	CY	Carry flag.
PSW.6	AC	Auxilliary Carry flag. (For BCD operations.)
PSW.5	F0	Flag 0. (Available to the user for general purposes.)
PSW.4	RS1	Register bank select control bit 1. Set/cleared by software to determine working register bank. (See Note.)
PSW.3	RS0	Register bank select control bit 0. Set/cleared by software todetermine working register bank. (See Note.)
PSW.2	OV	Overflow flag.
PSW.1	—	User-definable flag.
PSW.0	P	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the Accumulator, i.e., even parity.

NOTE: The contents of (RS1, RS0) enable the working register banks as follows:
(0,0)— Bank 0 (00H–07H)
(0,1)— Bank 1 (08H–0fH)
(1,0)— Bank 2 (10H–17H)
(1,1)— Bank 3 (18H–17H)

Stack Pointer

In the 8051 microcontrollers, the stack is 8-bit wide and it can hold data from 00 – FFH. The stack pointer can be used through the CPU to allow the stack. This microcontroller includes an 8-bit stack pointer that means it can allow values from 00H to FFH. Once it is activated, then the stack pointer includes the 07 value.

Ports 0 to 3

P0, P1, P2, and P3 are the SFR latches of Ports 0, 1, 2, and 3, respectively. Writing a one to a bit of a port SFR (P0, P1, P2, or P3) causes the corresponding port output pin to switch high. Writing a zero causes the port output pin to switch low. When used as an input, the external state of a port pin will be held in the port SFR (i.e., if the external state of a pin is low, the corresponding port SFR bit will contain a 0; if it is high, the bit will contain a 1).

Serial Data Buffer

The Serial Buffer is actually two separate registers, a transmit buffer and a receive buffer. When data is moved to SBUF, it goes to the transmit buffer and is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

Timer Registers Basic to 8051

Register pairs (TH0, TL0), and (TH1, TL1) are the 16-bit Counting registers for Timer/Counters 0 and 1, respectively.

Control Register for the 8051

Special Function Registers IP, IE, TMOD, TCON, SCON, and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port.

PSEN (Program Store Enable)

The 8051 has four dedicated bus control signals. It is a control signal that enables external program (code) memory**.** It usually connects to an EPROM's Output Enable (OE) pin to permit reading of program bytes.

The PSEN signal pulses low during the fetch stage of an instruction. When executing a program from internal ROM (8051/8052), PSEN remains in the inactive (high) state.

ALE (Address Latch Enable)

The 8051 similarly uses ALE for demultiplexing the address and data bus. When Port 0 is used in its alternate mode—as the data bus and the low-byte of the address bus—ALE is the signal that latches the address into an external register during the first half of a memory cycle.

EA (External Access)

The EA input signal is generally tied high (+5 V) or low (ground). If high, the 8051 executes programs from internal ROM when executing in the lower 4K of memory. If low, programs execute from external memory only (and PSEN pulses low accordingly).

RST (Reset)

The RST input is the master reset for the 8051. When this signal is brought high for at least two machinecycles, the 8051 internal registers are loaded with appropriate values for an orderly system start-up.

Timers/Counters

8051 microcontroller has two 16 bit timers and counters. These counters are again divided into a 8 bit register. The timers are used for measurement of intervals to determine the pulse width of pulses.

On-chip Oscillator Inputs

The 8051 features an on-chip oscillator. The nominal crystal frequency is 12 MHz for most ICs in the MCS-51™ family.

Memory Organization

Most microprocessors implement a shared memory space for data and programs. This is reasonable, since programs are usually stored on a disk and loaded into RAM for execution; thus both the data and programs reside in the system RAM. Microcontrollers have limited memory, and there is no disk drive or disk operating system. The control program must reside in. For this reason, the 8051 implements a separate memory space for programs (code) and data. Both the code and data may be internal; however, both expand using external components to a maximum of 64K code memory and 64K data memory.

The internal memory consists of on-chip ROM and on-chip data RAM. ***The on-chip RAM contains a rich arrangement of general-purpose storage, bit-addressable storage, register banks, and special function registers.***

ADDRESSING MODES

The way in which an operand is given to an instruction is known as addressing modes in 8051 microcontroller.

8051 has four addressing modes.

1. Immediate Addressing

Data is immediately available in the instruction.

For example -

ADD A, #77; Adds 77 (decimal) to A and stores in A

ADD A, #4DH; Adds 4D (hexadecimal) to A and stores in A

MOV DPTR, #1000H; Moves 1000 (hexadecimal) to data pointer

2. Register Addressing

This way of addressing accesses the bytes in the current register bank. Data is available in the register specified in the instruction. The register bank is decided by 2 bits of Processor Status Word (PSW).

For example-

ADD A, R0; Adds content of R0 to A and stores in A

3. Direct Addressing

The address of the data is available in the instruction.

For example -

MOV A, 088H; Moves content of SFR TCON (address 088H)to A

4. Register Indirect Addressing

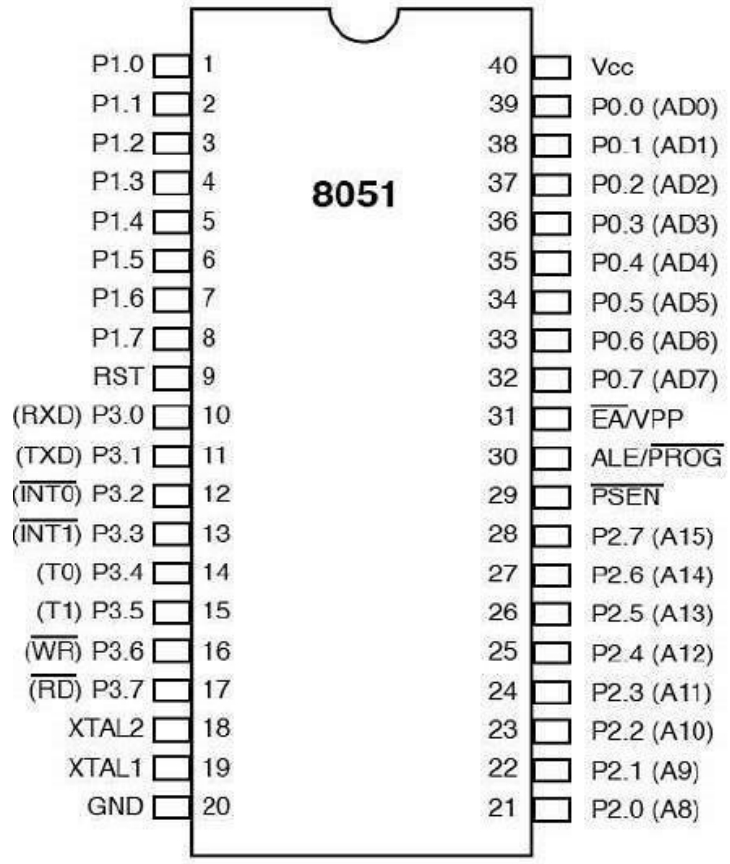
The address of data is available in the R0 or R1 registers as specified in theinstruction.

For example -

MOV A, @R0 moves content of address pointed by R0 to A

8051 PIN DESCRIPTION

The pin diagram of 8051 microcontroller looks as follows −



Pins 1 to 8 − These pins are known as Port 1. This port doesn't serve any other functions. It is internally pulled up, bi-directional I/O port.

Pin 9 − It is a RESET pin, which is used to reset the microcontroller to its initial values.

Pins 10 to 17 − These pins are known as Port 3. This port serves some functions like interrupts, timer input, control signals, serial communication signals RxD and TxD, etc.

Pins 18 & 19 − These pins are used for interfacing an external crystal to get the system clock.

Pin 20 − This pin provides the power supply to the circuit.

Pins 21 to 28 − These pins are known as Port 2. It serves as I/O port. Higher order address bus signals are also multiplexed using this port.

Pin 29 − This is PSEN pin which stands for Program Store Enable. It is used to read a signal from the external program memory.

Pin 30 − This is EA pin which stands for External Access input. It is used to enable/disable the external memory interfacing.

Pin 31 − This is ALE pin which stands for Address Latch Enable. It is used to demultiplex the address-data signal of port.

Pins 32 to 39 − These pins are known as Port 0. It serves as I/O port. Lower order address and data bus signals are multiplexed using this port.

Pin 40 − This pin is used to provide power supply to the circuit.

8051 microcontrollers have 4 I/O ports each of 8-bit, which can be configured as input or output. Hence, total 32 input/output pins allow the microcontroller to be connected with the peripheral devices.

Pin configuration, i.e. the pin can be configured as 1 for input and 0 for output as per the logic state.

- **Input/Output (I/O) pin** − All the circuits within the microcontroller must be connected to one of its pins except P0 port because it does not have pull-up resistors built-in.
- **Input pin** − Logic 1 is applied to a bit of the P register. The output FE transistor is turned off and the other pin remains connected to the power supply voltage over a pull-up resistor of high resistance.

Port 0 − The P0 (zero) port is characterized by two functions −

- When the external memory is used then the lower address byte (addresses A0A7) is applied on it, else all bits of this port are configured as input/output.
- When P0 port is configured as an output then other ports consisting of pins with built-in pull-up resistor connected by its end to 5V power supply, the pins of this port have this resistor left out.

Input Configuration

If any pin of this port is configured as an input, then it acts as if it ‒floats‖, i.e. the input has unlimited input resistance and in-determined potential.

Output Configuration

When the pin is configured as an output, then it acts as an ‒open drain‖. By applying logic 0 to a port bit, the appropriate pin will be connected to ground (0V), and applying logic 1, the external output will keep on ‒floating‖.

In order to apply logic 1 (5V) on this output pin, it is necessary to build an external pull up resistor.

Port 1

P1 is a true I/O port as it doesn't have any alternative functions as in P0, but this port can be configured as general I/O only. It has a built-in pull-up resistor and is completely compatible with TTL circuits.

Port 2

P2 is similar to P0 when the external memory is used. Pins of this port occupy addresses intended for the external memory chip. This port can be used for higher address byte with addresses A8-A15. When no memory is added then this port can be used as a general input/output port similar to Port 1.

Port 3

In this port, functions are similar to other ports except that the logic 1 must be applied to appropriate bit of the P3 register.

Instruction Set

8051 Microcontroller have set of instruction to perform different operations. There arefive group of instruction which are listed below.

Arithmetic Instructions

Logic Instructions

Data Transfer Instructions

Branch Instructions

Bit-oriented Instructions

Data Transfer Instructions

Operation : MOV
Syntax : MOV destination, source

Description: MOV copies the value of source into destination. The value of source is not affected. Both destination and source must be in Internal RAM. No flags are affected unless the instruction is moving the value of a bit into the carry bit in which case the carry bit is affected or unless the instruction is moving a value into the PSW register (which contains all the program flags).

Operation : MOVC
Function : Move Code Byte to Accumulator
Syntax : MOVC A,@A+*register*

Description: MOVC moves a byte from Code Memory into the Accumulator. The Code Memory address from which the byte will be moved is calculated by summing the value of the Accumulator with either DPTR or the Program Counter (PC). In the case of the Program Counter, PC is first

incremented by 1 before being summed with the Accumulator.

Operation : MOVX
Function : Move Data To/From External Memory (XRAM)
Syntax : MOVX *operand1,operand2*

Description: MOVX moves a byte to or from External Memory into or from theAccumulator.

If *operand1* is @DPTR, the Accumulator is moved to the 16-bit External Memory address indicated by DPTR. This instruction uses both P0 (port 0) and P2 (port 2) to output the 16-bit address and data. If *operand2* is DPTR then the byte is moved from External Memory into the Accumulator.

If *operand1* is @R0 or @R1, the Accumulator is moved to the 8-bit External Memory address indicated by the specified Register. This instruction uses only P0 (port 0) to output the 8-bit address and data. P2 (port 2) is not affected. If *operand2* is @R0 or @R1 then the byte is moved from External Memory into the Accumulator.

Operation : SWAP
Function : Swap Accumulator Nibbles
Syntax : SWAP A

Description: SWAP swaps bits 0-3 of the Accumulator with bits 4-7 of theAccumulator. This instruction is identical to executing "RR A" or "RL A" four times.

Operation : XCH
Function : Exchange Bytes
Syntax : XCH A,*register*

Description: Exchanges the value of the Accumulator with the value contained inregister.
Ex: XCH A, R1

Operation : PUSH
Function : Push Value Onto Stack
Syntax : PUSH

Description: PUSH "pushes" the value of the specified *iram addr* onto the stack. PUSH first increments the value of the Stack Pointer by 1, then takes the value stored in *iram addr* and stores it in Internal RAM at the location pointed to by the incremented Stack Pointer.

Operation : POP
Function : Pop Value From Stack
Syntax : POP

Description: POP "pops" the last value placed on the stack into the *iram addr* specified. In other words, POP will load *iram addr* with the value of the InternalRAM address pointed to by the current Stack Pointer. The stack pointer is then decremented by 1.

Arithmetic Instructions

Operation: ADD, ADDC
Function: Add Accumulator, Add Accumulator With Carry

Description: Description: ADD and ADDC both add the value *operand* to the value of the Accumulator, leaving the resulting value in the Accumulator. The value *operand* is not affected. ADD and ADDC function identically except that ADDC adds the value of operand as well as the value of the Carry flag whereas ADD does not add the Carry flag to the result.

Operation: SUBB
Function: Subtract from Accumulator With Borrow

Description: SUBB subtract the value of *operand* from the value of the Accumulator, leaving the resulting value in the Accumulator. The value *operand* is not affected.

The **Carry Bit (C)** is set if a borrow was required for bit 7, otherwise it is cleared. In other words, if the unsigned value being subtracted is greater than the Accumulatorthe Carry Flag is set.

Operation : MUL
Function : Multiply Accumulator by B
Syntax : MUL AB

Description: Multiples the unsigned value of the Accumulator by the unsigned value of the "B" register. The least significant byte of the result is placed in the Accumulator and the most-significant-byte is placed in the "B" register.

The **Carry Flag (C)** is always cleared.

Operation: DIV
Function: Divide Accumulator by B
Syntax: DIV AB

Description: Divides the unsigned value of the Accumulator by the unsigned value of the "B" register. The resulting quotient is placed in the Accumulator and the remainder is placed in the "B" register.

The **Carry flag (C)** is always cleared.

Operation: INC
Function: Increment Register
Syntax: INC *register*

Description: INC increments the value of *register* by 1. If the initial value of *register* is 255 (0xFF Hex), incrementing the value will cause it to reset to 0. Note: The Carry Flag is NOT set when the value "rolls over" from 255 to 0.
In the case of "INC DPTR", the value two-byte unsigned integer value of DPTR is incremented. If the initial value of DPTR is 65535 (0xFFFF Hex), incrementing the value will cause it to reset to 0. Again, the Carry Flag is NOT set when the value of DPTR "rolls over" from 65535 to 0.

Operation : DEC
Function : Decrement Register
Syntax : DEC *register*

Description: DEC decrements the value of *register* by 1. If the initial value of *register* is 0, decrementing the value will cause it to reset to 255 (0xFF Hex). Note: The Carry Flag is NOT set when the value "rolls over" from 0 to 255.

Logical Instructions

Operation : ORL
Function : Bitwise OR
Syntax : ORL *operand1*,*operand2*

Description: ORL does a bitwise "OR" operation between *operand1* and *operand2*, leaving the resulting value in *operand1*. The value of *operand2* is not affected. A logical "OR" compares the bits of each operand and sets the corresponding bit in the resulting byte if the bit was set in either of the original operands, otherwise the resulting bit is cleared.

Operation : ANL
Function : Bitwise AND
Syntax : ANL *operand1, operand2*

Description: ANL does a bitwise "AND" operation between *operand1* and *operand2*, leaving the resulting value in *operand1*. The value of operand2 is not affected. A logical "AND" compares the bits of each operand and sets the corresponding bit in theresulting byte only if the bit was set in both of the original operands, otherwise the resulting bit is cleared.

Operation : XRL
Function : Bitwise Exclusive OR
Syntax : XRL *operand1*,*operand2*

Description: XRL does a bitwise "EXCLUSIVE OR" operation between *operand1* and *operand2*, leaving the resulting value in *operand1*. The value of operand2 is not affected. A logical "EXCLUSIVE OR" compares the bits of each operand and sets thecorresponding bit in the resulting byte if the bit was set in either (but not both) of the original operands, otherwise the bit is cleared.

Operation : CPL
Function : Complement Register
Syntax : CPL *operand*

Description: CPL complements *operand*, leaving the result in *operand*. If *operand* is a single bit then the state of the bit will be reversed. If *operand* is the Accumulator then all the bits in the Accumulator will be reversed. This can be thought of as "Accumulator Logical Exclusive OR 255" or as "255-Accumulator." If the *operand* refers to a bit of an output Port, the value that will be complemented is based on the last value written to that bit, not the last value read from it.

Operation : CLR
Function : Clear Register
Syntax : CLR *register*

Description: CLR clears (sets to 0) all the bit(s) of the indicated register. If the register is a bit (including the carry bit), only the specified bit is affected. Clearing the Accumulator sets the Accumulator's value to 0.

Operation : RL
Function : Rotate Accumulator Left
Syntax : RL A

Description: Shifts the bits of the Accumulator to the left. The left-most bit (bit 7) of the Accumulator is loaded into bit 0.

Operation : RR
Function : Rotate Accumulator Right
Syntax : RR A

Description: Shifts the bits of the Accumulator to the right. The right-most bit (bit 0) of the Accumulator is loaded into bit 7.

Operation : RLC
Function : Rotate Accumulator Left Through Carry
Syntax : RLC A

Description: Shifts the bits of the Accumulator to the left. The left-most bit (bit 7) of the Accumulator is loaded into the Carry Flag, and the original Carry Flag is loaded into bit 0 of the Accumulator. This function can be used to quickly multiply a byte by 2.

Operation : RRC
Function : Rotate Accumulator Right Through Carry
Syntax : RRC A

Description: Shifts the bits of the Accumulator to the right. The right-most bit (bit 0) of the Accumulator is loaded into the Carry Flag, and the original Carry Flag is loaded into bit 7. This function can be used to quickly divide a byte by 2.

Branching Instructions

Operation : JMP
Function : Jump to Data Pointer + Accumulator
Syntax : JMP @A+DPTR

Description: JMP jumps unconditionally to the address represented by the sum of thevalue of DPTR and the value of the Accumulator.

Operation : JC
Function : Jump if Carry Set
Syntax : JC *reladdr*

Description: JC will branch to the address indicated by *reladdr* if the Carry Bit is set. If the Carry Bit is not set program execution continues with the instruction following the JC instruction.

Operation : JNC
Function : Jump if Carry Not Set
Syntax : JNC *reladdr*

Description: JNC branches to the address indicated by *reladdr* if the carry bit is not set. If the carry bit is set program execution continues with the instruction following the JNB instruction.

Operation : JZ
Function : Jump if Accumulator Zero
Syntax : JNZ *reladdr*

Description: JZ branches to the address indicated by *reladdr* if the Accumulator contains the value 0. If the value of the Accumulator is non-zero program execution continues with the instruction following the JNZ instruction.

Operation : JNZ
Function : Jump if Accumulator Not Zero
Syntax : JNZ *reladdr*

Description: JNZ will branch to the address indicated by *reladdr* if the Accumulator contains any value except 0. If the value of the Accumulator is zero program execution continues with the instruction following the JNZ instruction.

Operation : LCALL
Function : Long Call
Syntax : LCALL *code addr*

Description: LCALL calls a program subroutine. LCALL increments the program counter by 3 (to point to the instruction following LCALL) and pushes that value ontothe stack (low byte first, high byte second). The Program Counter is then set to the 16-bit value which follows the LCALL opcode, causing program execution to continue atthat address.

Operation : ACALL
Function : Absolute Call Within 2K Block
Syntax : ACALL *code address*

Description: ACALL unconditionally calls a subroutine at the indicated *code address*. ACALL pushes the address of the instruction that follows ACALL onto the stack, least-significant-byte first, most-significant-byte second. The Program Counter is then updated so that program execution continues at the indicated address.

Operation : RET
Function : Return From Subroutine
Syntax : RET

Description: RET is used to return from a subroutine previously called by LCALL or ACALL. Program execution continues at the address that is calculated by popping thetopmost 2 bytes off the stack. The most-significant-byte is popped off the stack first, followed by the least-significant-byte.

Bit-Wise Instructions

Operation : JB
Function : Jump if Bit Set
Syntax : JB *bit addr, reladdr*

Description: JB branches to the address indicated by *reladdr* if the bit indicated by*bit addr* is set. If the bit is not set program execution continues with the instruction following the JB instruction.

Operation : JNB
Function : Jump if Bit Not Set
Syntax : JNB *bit addr,reladdr*

Description: JNB will branch to the address indicated by *reladdress* if the indicated bit is not set. If the bit is set program execution continues with the instructionfollowing the JNB instruction.

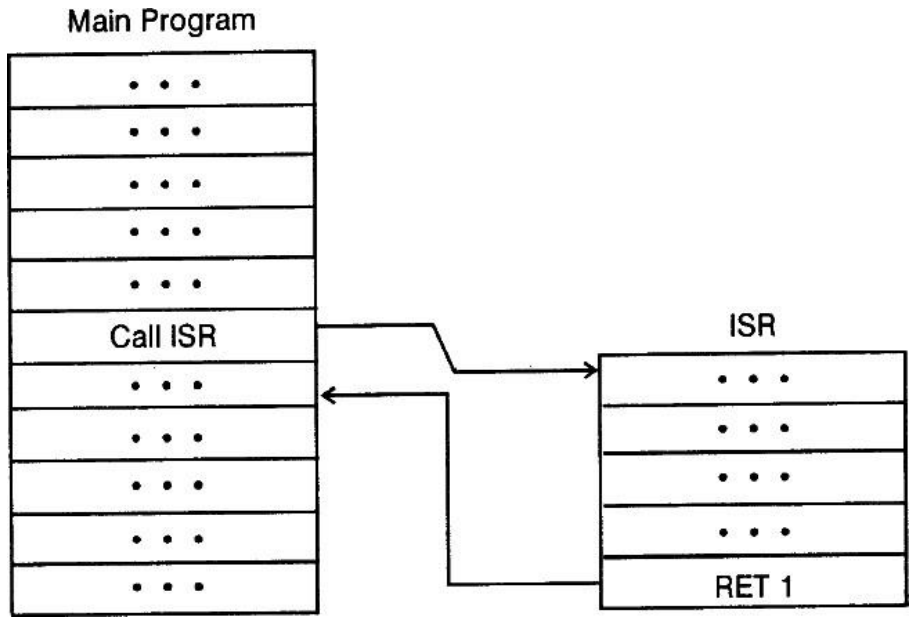
INTERRUPTS

An interrupt is an internal or external event that interrupts the microcontroller to inform it that a device needs its service. Whenever any device needs its service, the device notifies the microcontroller by sending it as interrupt signal. Upon receiving an interrupt signal, the microcontroller interrupts whatever it is doing and serves the device. The program which is associated with the interrupt is called interrupt Service Routine (ISR). The microcontroller can serve many devices based on the priority assigned to it.

Execution of an Interrupt

In order to use any interrupt, the following steps must be taken.

- It finishes the instruction it is executing and saves the address of the next instruction (PC) on the stack.
- It also saves the current status of all the interrupts internally.
- It jumps to a fixed location in memory called the interrupt vector or table that holds the address of the Interrupt Service Routine (ISR).
- The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it. It starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine which is RET 1.
- Upon executing RET 1 instruction, the microcontroller returns to the place where it was interrupted. First it gets the program counter (PC) address from the stack by popping the top two byes of the stack into the PC. Then it starts to execute from that address.



Interrupt Structure

Interrupts in 8051

Five interrupts are provided in the 805l.

Three of these a regenerated by internal operations: Timer Flag 1 & 0, and the serial port interrupt (RI or TI).

Two interrupts are triggered by external signals provided by circuitry that is connected to pin

INT0' and **INT1'**(port pins P3.2 and P3.3)

- It also saves the current status of all the interrupts internally.
- It jumps to a fixed location in memory called the interrupt vector or table that holds the address of the Interrupt Service Routine (ISR).
- The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it. It starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine which is RET 1.
- Upon executing RET 1 instruction, the microcontroller returns to the place where it was interrupted. First it gets the program counter (PC) address from the stack by popping the top two byes of the stack into the PC. Then it starts to execute from that address.

Types		Interrupt	Vector Address
Internal	TF0	Timer flag 0 interrupt	$000B_H$
	TF1	Timer flag 1 interrupt	$001B_H$
	RI/TI	Serial port interrupt	0023_H
External	$\overline{INT0}$	External interrupt 0	0003_H
	INT1	External interrupt 1	0013_H

Interrupt Vector

Timer flag interrupts

When a timer / counter overflows, the corresponding timer flag TF0 or TFI (location: 000B H or 001B H) is set to l.

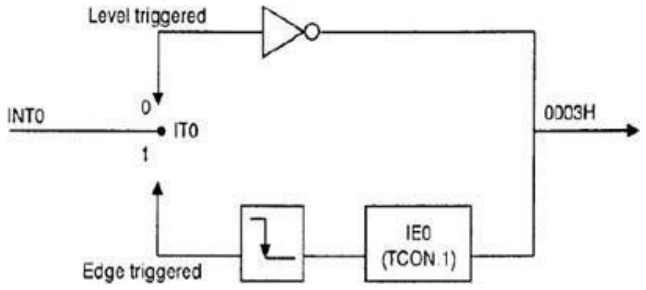
flag is cleared to 0 when the resulting interrupt generates aprogram call to the appropriate timer subroutine in memory.

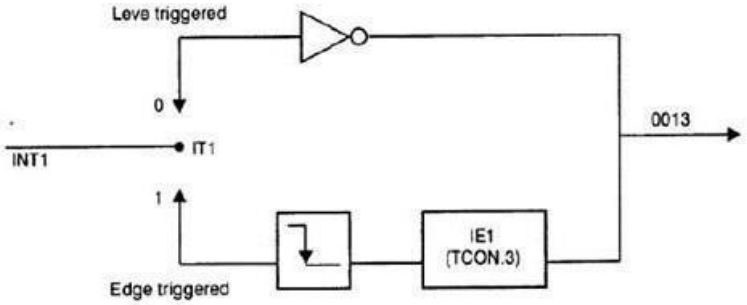
External interrupts

The external hardware interrupts INT0 and INT1 are located on pins P3.2 and P3.3.

Inputs on these pins can set the interrupt flags IE0 and IE1 in the TCON register to 1 by level triggering or edge-triggering.

Fig. Shows the activation of INT0 and INT1





Serial Port Interrupt

In SCON, if RI = l, a data byte is received If TI = 1, a data byte has been transmitted.

These are ORed together to provide a single interrupt to the processor.

The interrupt bit in the IE register is used to both send and receive data.

If IE.4 [ES- Enable serial port interrupt] is enabled, when RI or TI is raised and 8051gets interrupted and jumps to memory address location 0023H to execute the ISR.

The Fig.6.Showsthe serial interrupt is invoke by TI or RI flags.

Serial Port Interrupt

INTERRUPT CONTROL

All interrupt functions are under the control of the program. The programmer is able to alter control bits in the:

Interrupt Enable Register (IE)

Interrupt Priority Register (IP) and

Timer Control Register (TCON)

Interrupt Enable Register (IE)

The IE register holds the programmable bits that can enable or disable all the interrupts.

Bit D7 of the IE register (EA) must be set high to allow the rest of the register to take effect.

If EA = l, interrupts are enabled and will be responded to if their corresponding bits in IE are high.

If EA = 0, no interrupt will be responded to, even if the associated bit in the EI registeris high.

D7	D6	D5	D4	D3	D2	D1	D0
EA	-	-	ES	ET1	EX1	ET0	EX0

IE Register

EA: Enable interrupts bits.

Set to 1 to permit individual interrupts to be enabled by their enable bits.

Cleared to 0 by program to disable all interrupts.

ES: Enable serial port interrupt.

Set to 1 to enable by program.

Cleared to 0 to disable serial port interrupt.

ET1: Enable/ disable the Timer 1 overflow interrupt.

EX1: Enable external interrupt 1.

Set to 1 by program to enable **INT1'**interrupt.

Clearedto 0 to disable **INT1**'interrupt.

ET0: Enable / disable the Timer 0 overflow interrupt.

EX0: Enable/ disable the external interrupt 0.

Set to 1 by program to enable **INT0'** interrupt.

Clearedto 0 to disable**INT0**' interrupt.

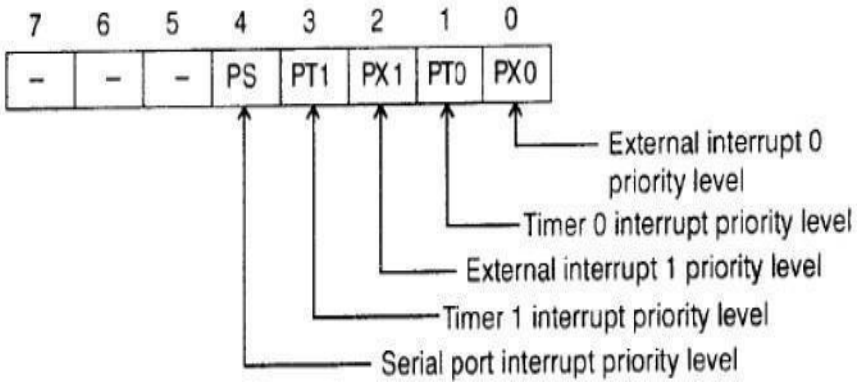
Interrupt Priority Register (IP)

Interrupt priority (IP) register determines the interrupt priority.

Bits in IP registers set to 1 give the accompanying interrupt a high priority; a 0 assignsa low priority.

Interrupts with a high priority can interrupt another interrupt with a lower priority and the lower priority continues after the higher is finished.

If two interrupts with the same priority occur at the same time, then they have the following ranking



1. IE0
2. TF0
3. IE1
4. TF1
5. RI/TI

The bit addressable IP register is shown inFig.9. If the bit is 0, the corresponding interrupts has a lower priority, otherwise higher priority.

Parallel I /O Ports :

The 8051 microcontroller has four parallel I/O ports , each of 8-bits .So, it provides the user 32 I/O lines for connecting the microcontroller to the peripherals. The four ports are P0 (Port 0), P1(Port1) ,P2(Port 2) and P3 (Port3). Upon reset all the ports are output ports. In order to make them input, all the ports must be set i.e a high bit mustbe sent to all the port pins. This is normally done by the instruction –SETB‖.

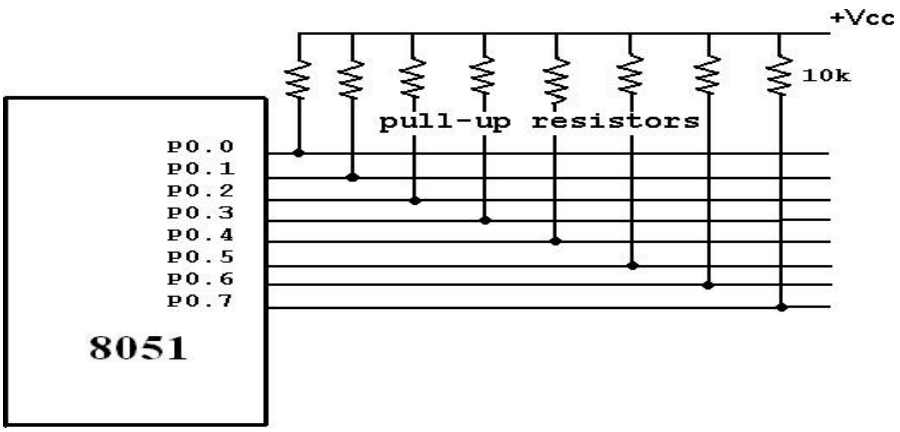
```
Ex:   MOV A,#0FFH      ; A = FF
      MOV P0,A         ; make P0 an input port
```

PORT 0:

Port 0 is an 8-bit I/O port with dual purpose. If external memory is used, these port pins are used for the lower address byte address/data (AD0-AD7), otherwise all bitsof the port are either input or output.. Unlike other ports, Port 0 is not provided with pull- up resistors internally ,so for PORT0 pull-up resistors of nearly 10k are to be connected externally as shown.

Dual role of port 0:

Port 0 can also be used as address/data bus(AD0-AD7), allowing it to be used for both address and data. When connecting the 8051 to an external memory, port 0 providesboth address and data. The 8051 multiplexes address and data through port 0 to save the pins. ALE indicates whether P0 has address or data. When ALE = 0, it provides data D0- D7, and when ALE =1 it provides address and data with the help of a 74LS373 latch.



Dual role of port 0

PORT 1:

Port 1 occupies a total of 8 pins (pins 1 through 8). It has no dual application and acts only as input or output port. In contrast to port 0, this port does not need any pull-up resistors since pull-up resistors connected internally. Upon reset, Port 1 is configured as an output port. To configure it as an input port, port bits must be set i.e a high bit must

be sent to all the port pins. This is normally done by the instruction –SETB‖.

Ex: MOV A, #0FFH; A=FF HEX

MOV P1, A; make P1 an input port by writing 1's to all of its pins

PORT 2:

Port 2 is also an eight-bit parallel port. (pins 21- 28). It can be used as input or output port. As this port is provided with internal pull-up resistors it does not need any external pull-up resistors. Upon reset, Port 2 is configured as an output port. If the port isto be used as input port, all the port bits must be made high by sending FF to the port. For

Ex: MOV A, #0FFH ; A=FF hex

MOV P2, A ; make P2 an input port by writing all 1's to it

Dual role of port 2:

Port2 lines are also associated with the higher order address lines A8-A15. In systems based on the 8751, 8951, and DS5000, Port2 is used as simple I/O port. But, in 8031-based systems, port 2 is used along with P0 to provide the 16-bit address for the external memory. Since an 8031 is capable of accessing 64K bytes of external memory, it needs a path for the 16 bits of the address. While P0 provides the lower 8 bits via A0- A7, it is the job of P2 to provide bits A8-A15 of the address. In

other words, when 8031is connected to external memory, Port 2 is used for the upper 8 bits of the 16-bit address,and it cannot be used for I/O operations.

PORT 3:

Port3 is also an 8-bit parallel port with dual function. (pins 10 to 17). The port pins can be used for I/O operations as well as for control operations. The details of these additional operations are given below in the table. Port 3 also do not need any externalpull-up resistors as they are provided internally similar to the case of Port2 & Port 1. Upon reset port 3 is configured as an output port. If the port is to be used as input port, all the port bits must be made high by sending FF to the port.

```
Ex:     MOV A, #0FFH      ; A= FF hex
        MOV P3, A         ; make P3 an input port by writing all 1's to it
```

Alternate Functions of Port 3 :

P3.0 and P3.1 are used for the RxD (Receive Data) and TxD (Transmit Data) serial communications signals. Bits P3.2 and P3.3 are meant for external interrupts. Bits P3.4 and P3.5 are used for Timers 0 and 1 and P3.6 and P3.7 are used to provide the write andread signals of external memories connected in 8031 based systems

S.No	Port 3 bit	Pin No	Function
1	P3.0	10	RxD
2	P3.1	11	TxD
3	P3.2	12	$\overline{\text{INT0}}$
4	P3.3	13	$\overline{\text{INT1}}$
5	P3.4	14	T0
6	P3.5	15	T1
7	P3.6	16	$\overline{\text{WR}}$
8	P3.7	17	$\overline{\text{RD}}$

Serial communication

Serial communication uses only one or two data lines to transfer data and is generally used for long distance communication. In serial communication the data is sentas one bit at a time in a timed

sequence on a single wire. Serial Communication takes place in two methods, Asynchronous data Transfer and Synchronous Data Transfer.

Asynchronous data transfer:

It allows data to be transmitted without the sender having to send a clock signal tothe receiver. Instead, special bits will be added to each word in order to synchronize the sending and receiving of the data. When a word is given to the UART for Asynchronoustransmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter.

After the Start Bit, the individual bits of the word of data are sent. Here each bit inthe word is transmitted for exactly the same amount of time as all of the other bits. Whenthe entire data word has been sent, the transmitter may add a Parity Bit that the transmitter generates. The Parity bit may be used by the receiver to perform simple error checking. Then at least one Stop Bit is sent by the transmitter. If the Stop Bit does not appear when it is supposed to, the UART considers the entire word to be corrupted and will report aFraming Error.

Baud rate is a measurement of transmission speed in asynchronous communication, it represents the number of bits/sec that are actually being sent over the serial link. The Baud count includes the overhead bits Start, Stop and Parity that are generated by the sending UART and removed by the receiving UART.

Synchronous data transfer:

In this method the receiver knows when to ―read‖ the next bit coming from the sender. This is achieved by sharing a clock between sender and receiver. In most forms of serial Synchronous communication, if there is no data available at a given time to transmit, a fill character will be sent instead so that data is always being transmitted. Synchronous communication is usually more efficient because only data bits are transmitted between sender and receiver, however it will be costlier because extra wiring and control circuits are required to share a clock signal between the sender and receiver. Devices that use serial cables for their communication are split into two categories.

1. DTE (Data Terminal Equipment). Examples of DTE are computers, printers & terminals.
2. DCE (Data Communication Equipment). Example of DCE is modems.

Parallel Data Transfer:

Parallel communication uses multiple wires (bus) running parallel to each other, and can transmit data on all the wires simultaneously. i.e all the bits of the byte are transmitted at a time. So, speed of the parallel data transfer is extremely high compared to serial data transfer. An 8-bit parallel data transfer is 8-times faster than serial data transfer. Hence with in the computer all data transfer is mainly based on Parallel data transfer. But only limitation is due to the high cost, this method is limited to only short distance communications.

S.No	Serial Communication	Parallel Communication
1	Data is transmitted bit after the bit ina single line	Data is transmitted simultaneously through group of lines(Bus)
2	Data congestion takes place	No, Data congestion
3	Low speed transmission	High speed transmission
4	Implementation of serial links is not an easy task.	Parallel data links are easilyimplemented in hardware
5.	In terms of transmission channel costs such as data bus cable length, data bus buffers, interface connectors, it is less expensive	It is more expensive
6	No , crosstalk problem	Crosstalk creates interference between the parallel lines.
7	No effect of intersymbol interference and noise	Parallel ports suffer extremely from inter-symbol interference (ISI) and noise, and therefore the data can be corrupted over long distances.
8	The bandwidth of serial wires ismuch higher.	The bandwidth of parallel wires is much lower.
9	Serial interface is more flexible to upgrade , without changing the hardware	Parallel data transfer mechanismrely on hardware resources and hence not flexible to upgrade.
10	Serial communication workeffectively even at high frequencies.	Parallel buses are hard to run at high frequencies.

PIPELINING

To improve the performance of a processor we have two options: 1) Improve the hardware by introducing faster circuits. 2) Arrange the hardware such that more than one operation can be performed at the same time. Since there is a limit on the speed of hardware and the cost of faster circuits is quite high, we have to adopt the 2nd option.

Pipelining is a process of arrangement of hardware elements of the processor such that its overall performance is increased. Simultaneous execution of more than one instruction takes place in a pipelined processor.

Design of a basic pipeline

In a pipelined processor, a pipeline has two ends, the input end and the output end. Between these ends, there are multiple stages/segments such that the output of one stage is connected to the input of the next stage and each stage performs a specific operation.

Interface registers are used to hold the intermediate output between two stages. These interface registers are also called latch or buffer.

All the stages in the pipeline along with the interface registers are controlled by a common clock.

Execution in a pipelined processor Execution sequence of instructions in a pipelined processor can be visualized using a space-time diagram. For example, consider a processor having 4 stages and let there be 2 instructions to be executed. We can visualize the execution sequence through the following space-time diagrams:

Non-overlapped execution:

Stage / Cycle	1	2	3	4	5	6	7	8
S1	I_1				I_2			
S2		I_1				I_2		
S3			I_1				I_2	
S4				I_1				I_2

Total time = 8 Cycle

Overlapped execution:

Stage / Cycle	1	2	3	4	5
S1	I_1	I_2			
S2		I_1	I_2		
S3			I_1	I_2	
S4				I_1	I_2

Total time = 5 Cycle **Pipeline Stages** RISC processor has 5 stage instruction pipeline to execute all the instructions in the RISC instruction set. Following are the 5 stages of the RISC pipeline with their respective operations:

Stage 1 (Instruction Fetch) In this stage the CPU reads instructions from the address in the memory whose value is present in the program counter.

Stage 2 (Instruction Decode) In this stage, instruction is decoded and the register file is accessed to get the values from the registers used in the instruction.

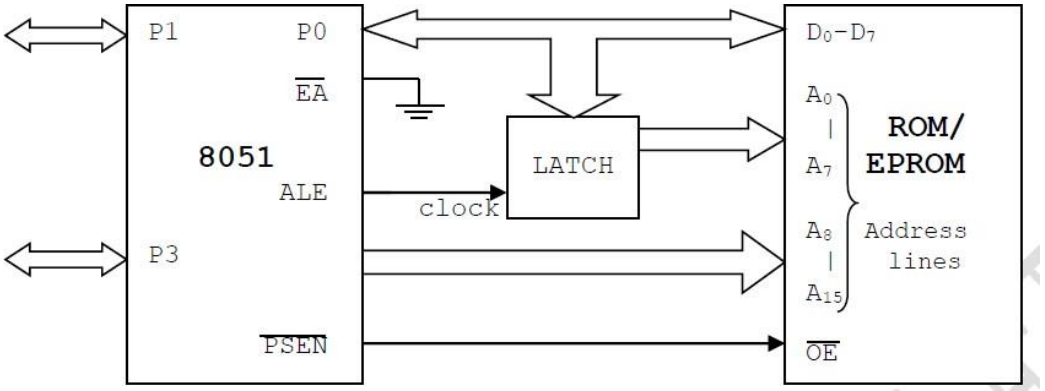
Stage 3 (Instruction Execute) In this stage, ALU operations are performed.

Stage 4 (Memory Access) In this stage, memory operands are read and written from/to the memory that is present in the instruction.

Stage 5 (Write Back) In this stage, computed/fetched value is written back to the register present in the instructions.

EXTERNAL MEMORY INTERFACE

EXTERNAL ROM (PROGRAM MEMORY) INTERFACING

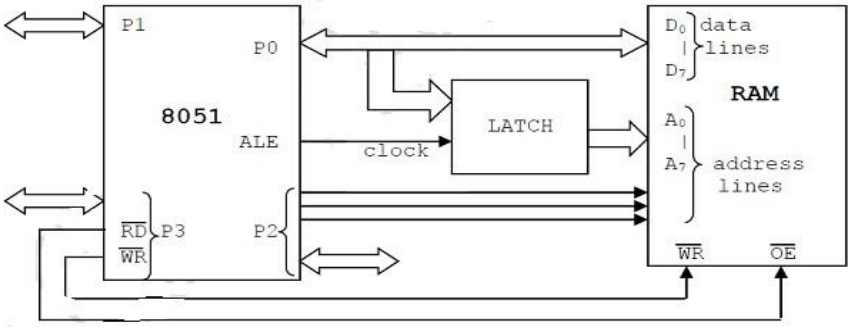


Interfacing of ROM/EPROM to 8051

Figure shows how to access or interface ROM to 8051.

- Port 0 is used as multiplexed data & address lines. It gives lower order (A7-A0) 8bit address in initial T cycle & higher order (A8-A15) used as data bus.
- 8 bit address is latched using external latch & ALE signal from 8051.
- Port 2 provides higher order (A15-A8) 8 bit address.
- PSEN is used to activate the output enable signal of external ROM/EPROM.

EXTERNAL RAM (DATA MEMORY) INTERFACING



Interfacing of RAM to 8051

Figure shows how to connect or interface external RAM (data memory) to 8051.

- Port 0 is used as multiplexed data & address lines.
- Address lines are decoded using external latch & ALE signal from 8051 to providelower order (A7-A0) address lines.
- Port 2 gives higher order address lines.
- RD & WR signals from 8051 selects the memory read & memory write operationsrespectively.

Note:RD & WR signals: generally P3.6 & P3.7 pins of port 3 are used to generate memory read and memory write signals. Remaining pins of port 3 i.e. P3.0-P3.5 can be used for other functions.

Solved Examples:

Example 1: Design a μController system using 8051 to Interface the external RAM of size 16k x 8.

Solution: Given, Memory size: 16k

Which means, we require 2^n=16k: n address lines

- Here n=14: A0 to A13 address lines are required.
- A14 and A15 are connected through OR gate to CS pin of external RAM.
- When A14 and A15 both are low (logic '0'), external data memory (RAM) is selected.

Address Decoding (Memory Map) For 16k X 8 Ram

Address	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Hex Addr
Start	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
End	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH

Figure shows interfacing of 16k x 8 RAM to 8051.

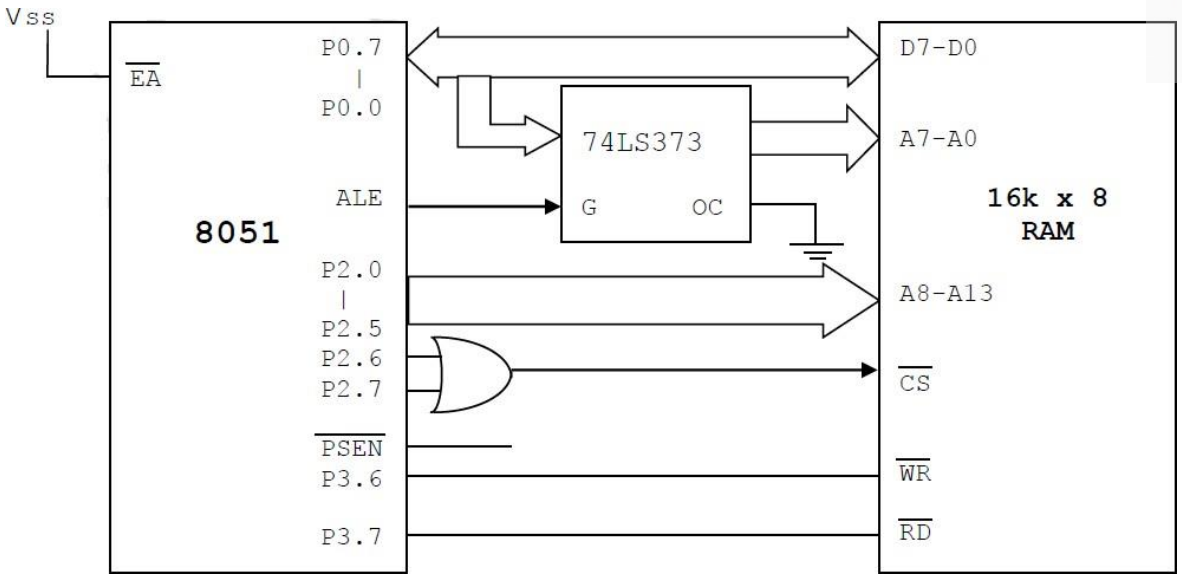


Figure 16Kx8 Memory (RAM) Interfacing with 8051

Example 2: Design a µController system using 8051 to interface the external ROM ofsize 4k x 8.

Solution: Given, Memory size: 4k

i.e we require 2^n=4k :: n address lines

- Here n=12 :: A0 to A11 address lines are required.
- Remaining lines A0, A0, A0, A0 & PSEN are connected though OR gate to CS &RD of external ROM.
- When A0 to A0 are low (logic '0'), only then external ROM is selected.

Address Decoding (Memory Map) for 4k x 8 RAM

Address	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Hex Addr
Start	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
End	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFFH

Figure shows interfacing of 4k x 8 ROM to 8051

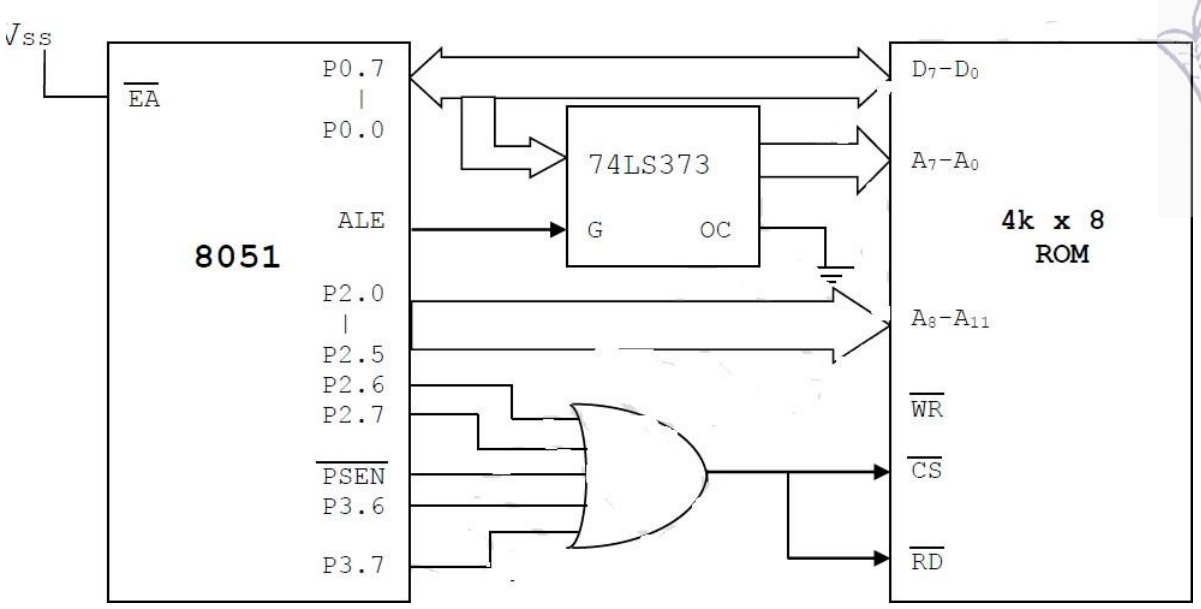


Figure 4Kx8 Memory (ROM) Interfacing with 8051

Example 3: Design a μController system using 8051, 16k bytes of ROM & 32k bytes of RAM. Interface the memory such that starting address for ROM is 0000H & RAM is 8000H.

Solution:

Given, Memory size- ROM : 16k

i.e we require 2^n=16k :: n address lines

Here n=14 :: A0 to A13 address lines are required.

A14,A15,PSEN ORed CS *when low – ROM is selected*

Memory size- RAM :32k

i.e we require 2^n=32k :: n address lines

Here n=15 :: A0 to A15 address lines are required.

A15 inverted(NOT Gate) CS when high- RAM is selected.For RAM selection

- PSEN is used as chip select pin ROM.
- RD is used as read control signal pin..
- WR is used as write control signal pin.

Address Decoding (Memory Map) for 16k x 8 ROM.

Address	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Hex Addr
Start	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000 H
End	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF H

Address Decoding (Memory Map) for 32k x 8 RAM.

Address	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Hex Addr
Start	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000 H
End	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF H

Figure shows the interfacing of 16Kx8 Memory (ROM) and 32Kx8 RAM with 8051

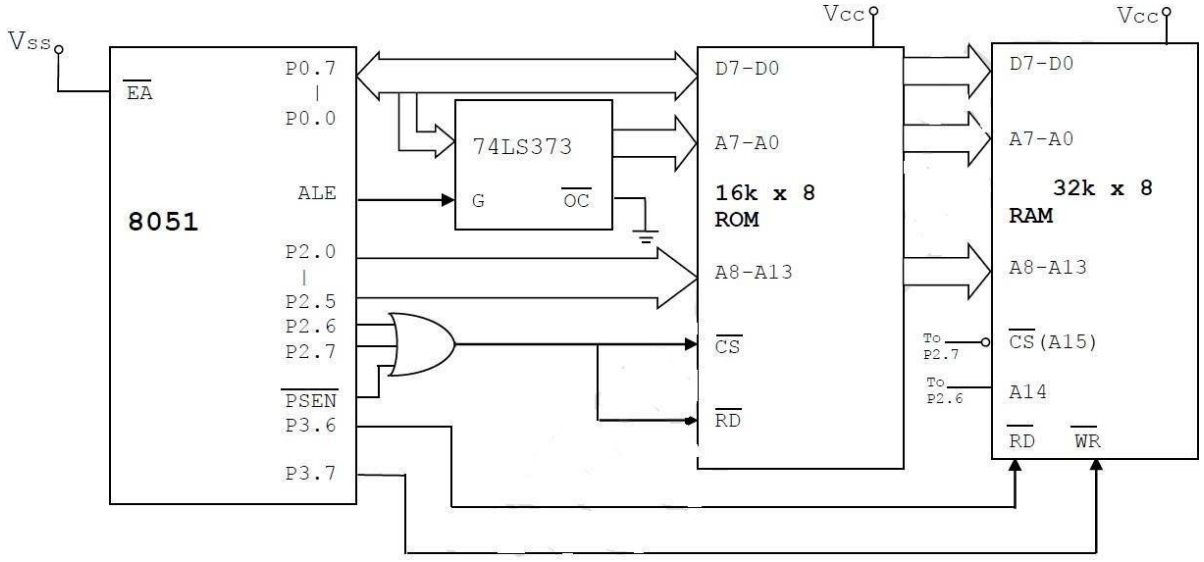
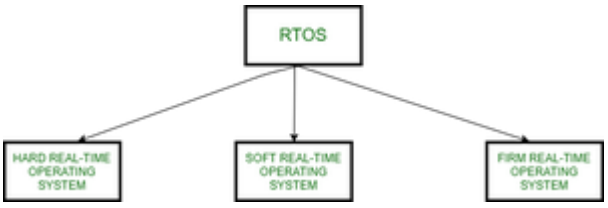


Figure 16Kx8 Memory (ROM) and 32Kx8 RAM Interfacing with 8051

RTOS

Real-time **operating systems (RTOS)** are used in environments where a large number of events, mostly external to the computer system, must be accepted and processed in a short time or within certain deadlines. such applications are industrial control, telephone switching equipment, flight control, and real-time simulations

The real-time operating systems can be of 3 types –



1. **Hard Real-Time operating system:**
   These operating systems guarantee that critical tasks be completed within a range of time.
   For example, a robot is hired to weld a car body. If the robot welds too early or too late, the car cannot be sold, so it is a hard real-time system that requires complete car welding by robot hardly on the time., scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

2. **Soft real-time operating system:**
   This operating system provides some relaxation in the time limit.
   For example – Multimedia systems, digital audio systems etc. Explicit, programmer-defined and controlled processes are encountered in real-time systems. A separate process is changed with handling a single external event. The process is activated upon occurrence of the related event signalled by an interrupt.

   Multitasking operation is accomplished by scheduling processes for execution independently of each other. Each process is assigned a certain level of priority that corresponds to the relative importance of the event that it services. The processor is allocated to the highest priority processes. This type of schedule, called, priority-based preemptive scheduling is used by real-time systems.

3. **Firm Real-time Operating System**:
   RTOS of this type have to follow deadlines as well. In spite of its small impact, missing a

deadline can have unintended consequences, including a reduction in the quality of the product. Example: Multimedia applications.

Advantages:
The advantages of real-time operating systems are as follows-

1. **Maximum consumption –**
   Maximum utilization of devices and systems. Thus more output from all the resources.

2. **Task Shifting –**
   Time assigned for shifting tasks in these systems is very less. For example, in older systems, it takes about 10 microseconds. Shifting one task to another and in the latest systems, it takes 3 microseconds.

3. **Focus On Application –**
   Focus on running applications and less importance to applications that are in the queue.

4. **Real-Time Operating System In Embedded System –**
   Since the size of programs is small, RTOS can also be embedded systems like in transport and others.

5. **Error Free –**
   These types of systems are error-free.

6. **Memory Allocation –**
   Memory allocation is best managed in these types of systems.

Disadvantages:
The disadvantages of real-time operating systems are as follows-

1. **Limited Tasks –**
   Very few tasks run simultaneously, and their concentration is very less on few applications to avoid errors.

2. **Use Heavy System Resources –**
   Sometimes the system resources are not so good and they are expensive as well.

3. **Complex Algorithms –**
   The algorithms are very complex and difficult for the designer to write on.

4. **Device Driver And Interrupt signals –**
   It needs specific device drivers and interrupts signals to respond earliest to interrupts.

5. **Thread Priority –**
   It is not good to set thread priority as these systems are very less prone to switching tasks.

6. **Minimum Switching –** RTOS performs minimal task switching.

Comparison of Regular and Real-Time operating systems:

Regular OS	Real-Time OS (RTOS)
Complex	Simple
Best effort	Guaranteed response
Fairness	Strict Timing constraints
Average Bandwidth	Minimum and maximum limits
Unknown components	Components are known
Unpredictable behavior	Predictable behavior
Plug and play	RTOS is upgradeable

MULTIPLE TASKS AND MULTIPLE PROCESSES:

Tasks and Processes

Many (if not most) embedded computing systems do more than one thing that is, the environment can cause mode changes that in turn cause the embedded system to behave quite differently. For example, when designing a telephone answering machine,

We can define recording a phone call and operating the user's control panel as distinct tasks, because they perform logically distinct operations and they must be performed at very different rates. These different **tasks** are part of the system's functionality, but that application-level organization of functionality is often reflected in the structure of the program as well.

A ***process*** is a single execution of a program. If we run the same program two different times, we have created two different processes. Each process has its own state that includes not only its registers but all of its memory. In some OSs, the memory management unit is used to keep each process in a separate address space. In others, particularly lightweight RTOSs, the processes run in the same address space. Processes that share the same address space are often called ***threads***.

Multirate Systems

Implementing code that satisfies timing requirements is even more complex when multiple rates of computation must be handled. ***Multirate*** embedded computing systems are very common, including automobile engines, printers, and cell phones. In all these systems, certain operations must be executed periodically, and each operation is executed at its own rate.

Timing Requirements on Processes

Processes can have several different types of timing requirements imposed on them by the application. The timing requirements on a set of processes strongly influence the type of scheduling that is appropriate. A scheduling policy must define the timing requirements that it uses to determine whether a schedule is valid. Before studying scheduling proper, we outline the types of process timing requirements that are useful in embedded system design.

Two important requirements on processes: ***release time*** and ***deadline***.

The release time is the time at which the process becomes ready to execute; this is not necessarily the time at which it actually takes control of the CPU and starts to run. An aperiodic process is by definition initiated by an event, such as external data arriving or data computed by another process.

The release time is generally measured from that event, although the system may want to make the process ready at some interval after the event itself. For a periodically executed process, there are two common possibilities.

In simpler systems, the process may become ready at the beginning of the period. More sophisticated systems, such as those with data dependencies between processes, may set the release time at the arrival time of certain data, at a time after the start of the period.

A deadline specifies when a computation must be finished. The deadline for an aperiodic process is generally measured from the release time, since that is the only reasonable time reference. The deadline for a periodic process may in general occur at some time other than the end of the period.

Rate requirements are also fairly common. A rate requirement specifies how quickly processes must be initiated.

The ***period*** of a process is the time between successive executions. For example, the period of a digital filter is defined by the time interval between successive input samples.

The process's ***rate*** is the inverse of its period. In a multirate system, each process executes at its own distinct rate.

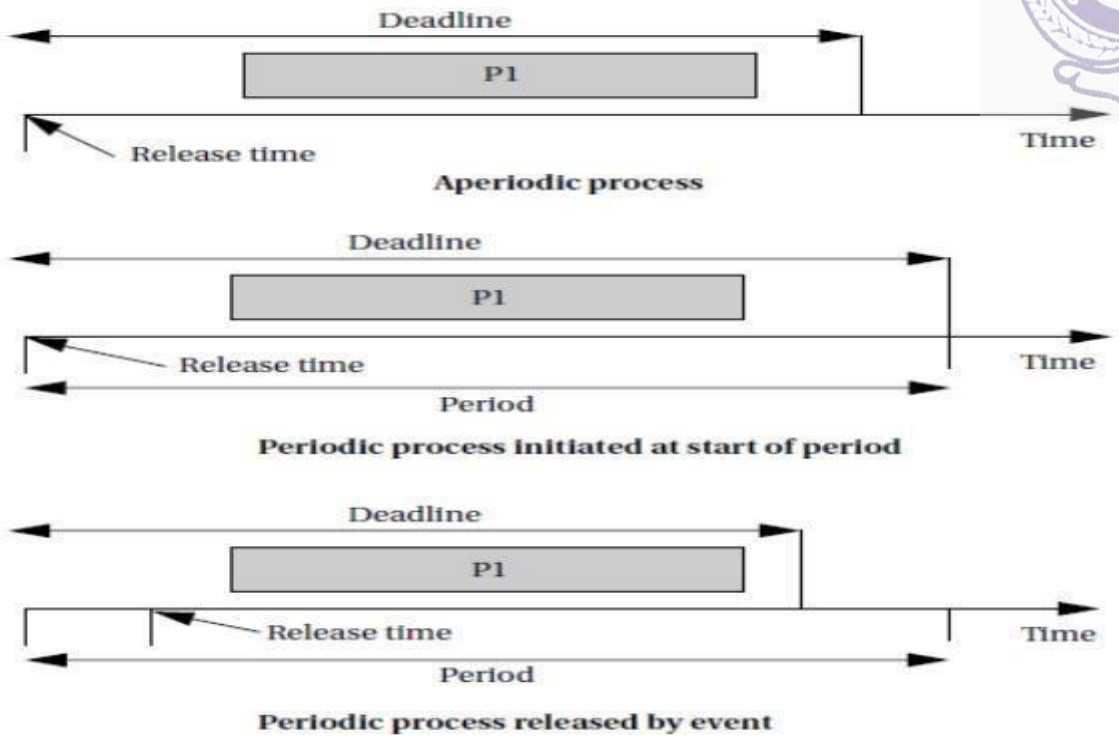


Fig 3.2 Example definitions of release times and deadlines.

The most common case for periodic processes is for the initiation interval to be equal to the period. However, pipelined execution of processes allows the initiation interval to be less than the period. Figure 3.3 illustrates process execution in a system with four CPUs.

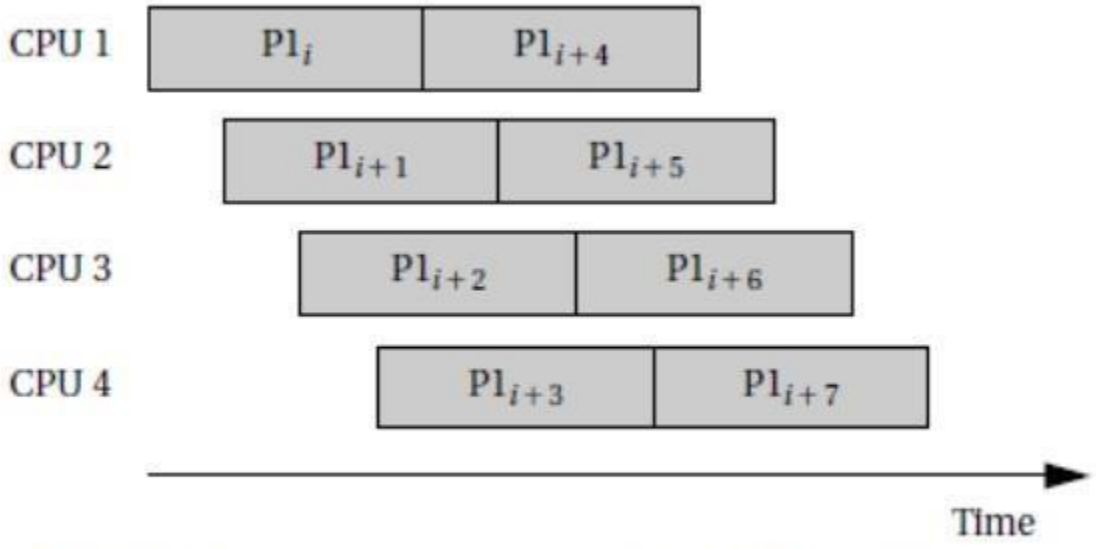


Fig 3.3 A sequence of processes with a high initiation rate.

CPU Metrics

We also need some terminology to describe how the process actually executes. The ***initiation time*** is the time at which a process actually starts executing on the CPU. The ***completion time*** is the time at which the process finishes its work.

The most basic measure of work is the amount of ***CPU time*** expended by a process. The CPU time of process *i* is called *Ci* . Note that the CPU time is not equal to the completion time minus initiation time; several other processes may interrupt execution. The total CPU time consumed by a set of processes is

$$T = \sum Ti$$

We need a basic measure of the efficiency with which we use the CPU. The simplest and most direct measure is ***utilization***:

U=CPU time for useful work/total available CPU time

Utilization is the ratio of the CPU time that is being used for useful computations to the total available CPU time. This ratio ranges between 0 and 1, with 1 meaning that all of the available CPU time is being used for system purposes. The utilization is often expressed as a percentage. If we measure the total execution time of all processes over an interval of time *t*, then the CPU utilization is

$$U=T/t.$$

Context Switching in OS (Operating System)

The Context switching is a technique or method used by the operating system to switch a process from one state to another to execute its function using CPUs in the system. When switching perform in the system, it stores the old running process's status in the form of registers and assigns the CPU to a new process to execute its tasks. While a new process is running in the system, the previous process must wait in a ready queue. The execution of the old process starts at that point where another process stopped it. It defines the characteristics of a multitasking operating system in which multiple processes shared the same CPU to perform multiple tasks without the need for additional processors in the system.

The need for Context switching

A context switching helps to share a single CPU across all processes to complete its execution and store the system's tasks status. When the process reloads in the system, the execution of the process starts at the same point where there is conflicting.

Following are the reasons that describe the need for context switching in the Operating system.

1. The switching of one process to another process is not directly in the system. A context switching helps the operating system that switches between the multiple processes to use the CPU's resource to accomplish its tasks and store its context. We can resume the service of the process at the same point later. If we do not store the currently running process's data or context, the stored data may be lost while switching between processes.
2. If a high priority process falls into the ready queue, the currently running process will be shut down or stopped by a high priority process to complete its tasks in the system.
3. If any running process requires I/O resources in the system, the current process will be switched by another process to use the CPUs. And when the I/O requirement is met, the old process goes into a ready state to wait for its execution in the CPU. Context switching stores the state of the process to resume its tasks in an operating system. Otherwise, the process needs to restart its execution from the initials level.
4. If any interrupts occur while running a process in the operating system, the process status is saved as registers using context switching. After resolving the interrupts, the process switches from a wait state to a ready state to resume its execution at the same point later, where the operating system interrupted occurs.
5. A context switching allows a single CPU to handle multiple process requests simultaneously without the need for any additional processors.

Example of Context Switching

Suppose that multiple processes are stored in a Process Control Block (PCB). One process is running state to execute its task with the use of CPUs. As the process is running, another process arrives in the ready queue, which has a high priority of completing its task using CPU. Here we used context switching that switches the current process with the new process

requiring the CPU to finish its tasks. While switching the process, a context switch saves the status of the old process in registers. When the process reloads into the CPU, it starts the execution of the process when the new process stops the old process. If we do not save the state of the process, we have to start its execution at the initial level. In this way, context switching helps the operating system to switch between the processes, store or reload the process when it requires executing its tasks.

Context switching triggers

Following are the three types of context switching triggers as follows.

1. Interrupts
2. Multitasking
3. Kernel/User switch

Interrupts: A CPU requests for the data to read from a disk, and if there are any interrupts, the context switching automatic switches a part of the hardware that requires less time to handle the interrupts.

Multitasking: A context switching is the characteristic of multitasking that allows the process to be switched from the CPU so that another process can be run. When switching the process, the old state is saved to resume the process's execution at the same point in the system.

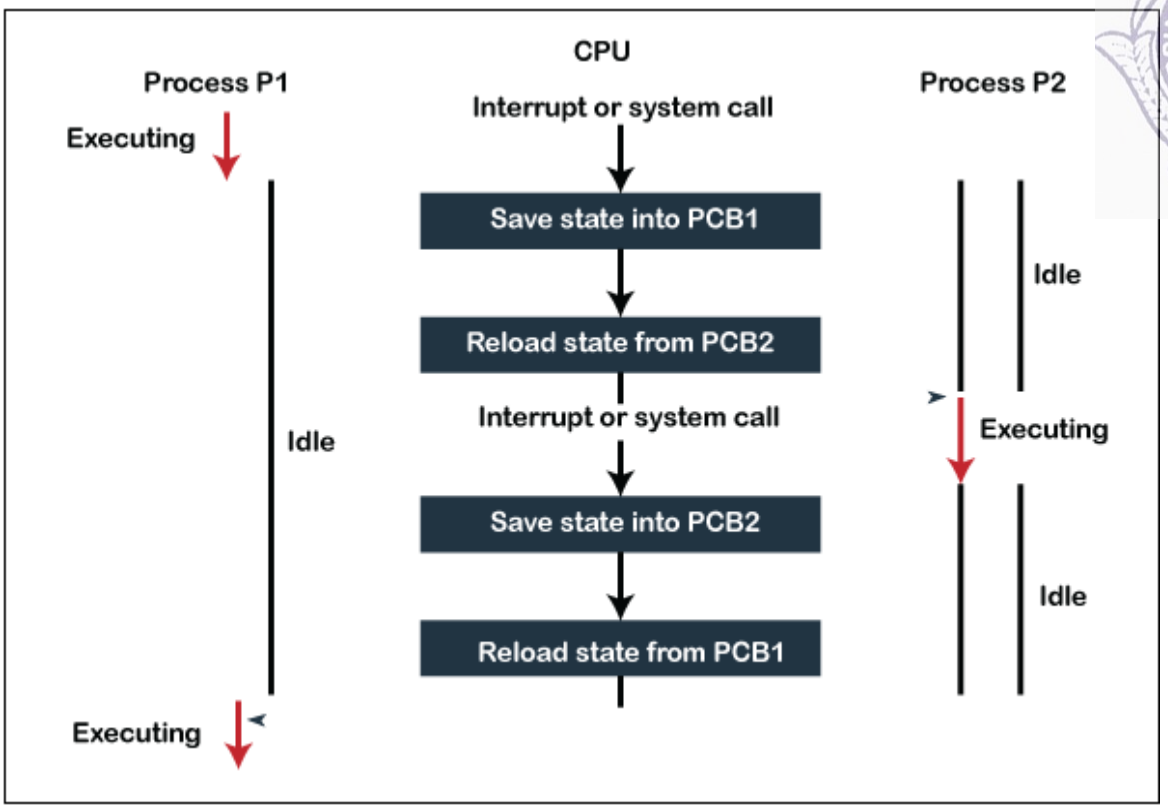
Kernel/User Switch: It is used in the operating systems when switching between the user mode, and the kernel/user mode is performed.

What is the PCB?

A PCB (Process Control Block) is a data structure used in the operating system to store all data related information to the process. For example, when a process is created in the operating system, updated information of the process, switching information of the process, terminated process in the PCB.

Steps for Context Switching

There are several steps involves in context switching of the processes. The following diagram represents the context switching of two processes, P1 to P2, when an interrupt, I/O needs, or priority-based process occurs in the ready queue of PCB.



As we can see in the diagram, initially, the P1 process is running on the CPU to execute its task, and at the same time, another process, P2, is in the ready state. If an error or interruption has occurred or the process requires input/output, the P1 process switches its state from running to the waiting state. Before changing the state of the process P1, context switching saves the context of the process P1 in the form of registers and the program counter to the **PCB1**. After that, it loads the state of the P2 process from the ready state of the **PCB2** to the running state.

The following steps are taken when switching Process P1 to Process 2:

1. First, these context switching needs to save the state of process P1 in the form of the program counter and the registers to the PCB (Program Counter Block), which is in the running state.
2. Now update PCB1 to process P1 and moves the process to the appropriate queue, such as the ready queue, I/O queue and waiting queue.
3. After that, another process gets into the running state, or we can select a new process from the ready state, which is to be executed, or the process has a high priority to execute its task.
4. Now, we have to update the PCB (Process Control Block) for the selected process P2. It includes switching the process state from ready to running state or from another state like blocked, exit, or suspend.

5. If the CPU already executes process P2, we need to get the status of process P2 to resume its execution at the same time point where the system interrupt occurs.

Similarly, process P2 is switched off from the CPU so that the process P1 can resume execution. P1 process is reloaded from PCB1 to the running state to resume its task at the same point. Otherwise, the information is lost, and when the process is executed again, it starts execution at the initial level.

Priority Based Scheduling

Earliest-Deadline-First Scheduling

Earliest Deadline First (EDF) is one of the best known algorithms for realtime processing. It is an optimal dynamic algorithm. In dynamic priority algorithms, the priority of a task can change during its execution. It produces a valid schedule whenever one exists.

EDF is a preemptive scheduling algorithm that dispatches the process with the earliest deadline. If an arriving process has an earlier deadline than the running process, the system preempts the running process and dispatches the arriving process.

A task with a shorter deadline has a higher priority. It executes a job with the earliest deadline.

Tasks cannot be scheduled by rate monotonic algorithm.

EDF is optimal among all scheduling algorithms not keeping the processor idle at certain times.

Upper bound of process utilization is 100 %.

Whenever a new task arrive, sort the ready queue so that the task closest to the end of its period assigned the highest priority. System preempt the running task if it is not placed in the first of the queue in the last sorting.

If two tasks have the same absolute deadlines, choose one of the two at random (ties can be broken arbitrarily). The priority is dynamic since it changes for different jobs of the same task.

EDF can also be applied to aperiodic task sets. Its optimality guarantees that the maximal lateness is minimized when EDF is applied.

Many real time systems do not provide hardware preemption, so other algorithm must be employed.

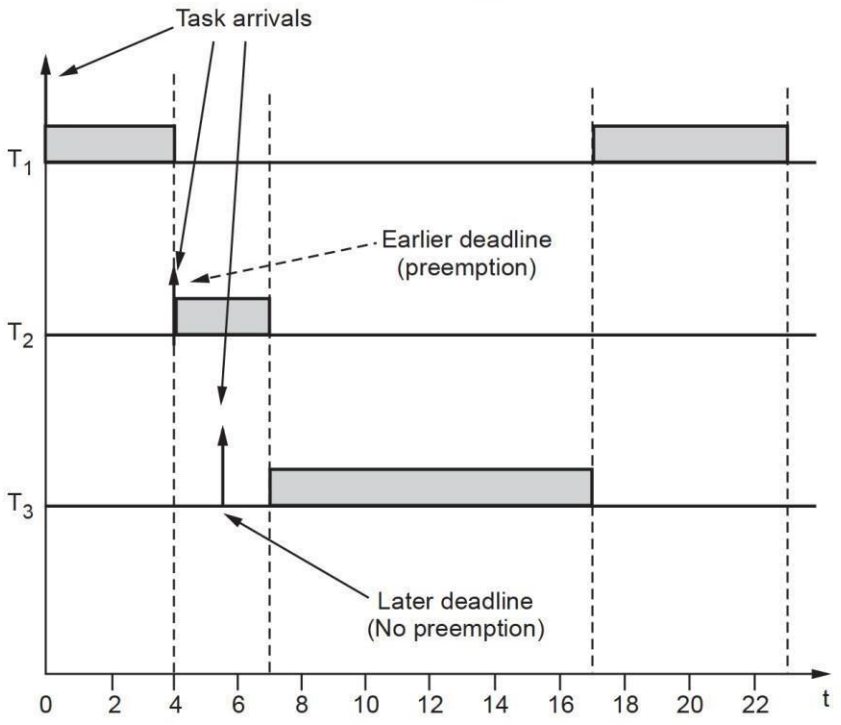
In scheduling theory, a real-time system comprises a set of real-time tasks;each task consists of an infinite or finite stream of jobs. The task set can be scheduled by a number of policies including fixed priority or dynamic priority algorithms.

The success of a real-time system depends on whether all the jobs of all the tasks can be guaranteed to complete their executions before their deadlines. If they can, then we say the task set is schedulable.

The schedulability condition is that the total utilization of the task set must be less than or equal to 1.

Implementation of earliest deadline first : Is it really not feasible to implement EDF scheduling ?

Task	Arrival	Duration	Deadline
T1	0	10	33
T2	4	3	28
T3	5	10	29



Problems for implementations :

1. Absolute deadlines change for each new task instance, therefore the priorityneeds to be updated every time the task moves back to the ready queue.
2. More important, absolute deadlines are always increasing, how can we associate a finite priority value to an ever increasing deadline value.
3. Most important, absolute deadlines are impossible to compute a-priori.

EDF properties :

1. EDF is optimal with respect to feasibility (i.e. schedulability).
2. EDF is optimal with respect to minimizing the maximum lateness.

Advantages

1. It is optimal algorithm.
2. Periodic, aperiodic and sporadic tasks are scheduled using EDF algorithm.
3. Gives best CPU utilization.

Disadvantages

1. Needs priority queue for storing deadlines
2. Needs dynamic priorities
3. Typically no OS support
4. Behaves badly under overload
5. Difficult to implement.

Rate Monotonic Scheduling

Rate Monotonic Priority Assignment (RM) is a so called static priority round robin scheduling algorithm.

In this algorithm, priority is increases with the rate at which a process mustbe scheduled. The process of lowest period will get the highest priority.

The priorities are assigned to tasks before execution and do not change over time. RM scheduling is preemptive, i.e., a task can be preempted by a task with higher priority.

In RM algorithms, the assigned priority is never modified during runtime of the system. RM assigns priorities simply in accordance with its periods, i.e. thepriority is as higher as shorter is the period which means as higher is the activationrate. So RM is a scheduling algorithm for periodic task sets.

If a lower priority process is running and a higher priority process becomesavailable to run, it will preempt the lower priority process. Each periodic task is assigned a priority inversely based on its period :

1. The shorter the period, the higher the priority.
2. The longer the period, the lower the priority.

The algorithm was proven under the following assumptions :

1. Tasks are periodic.
2. Each task must be completed before the next request occurs.
3. All tasks are independent.
4. Run time of each task request is constant.
5. Any non-periodic task in the system has no required deadlines.

RMS is optimal among all fixed priority scheduling algorithms forscheduling periodic tasks where the deadlines of the tasks equal their periods.

Advantages :

1. Simple to understand. 2. Easy to implement. 3. Stable algorithm.

Disadvantages :

1. Lower CPU utilization.
2. Only deal with independent tasks.
3. Non-precise schedulability analysis

Comparison between RMS and EDF

Parameters	RMS	EDF
Priorities	Static	Dynamic
Works with OS with fixed priorities	Yes	No
Uses full computational power of processor	No	Yes
Possible to exploit full computational power of	No	Yes

Priority Inversion

Priority inversion occurs when a low-priority job executes while some ready higher-priority job waits.

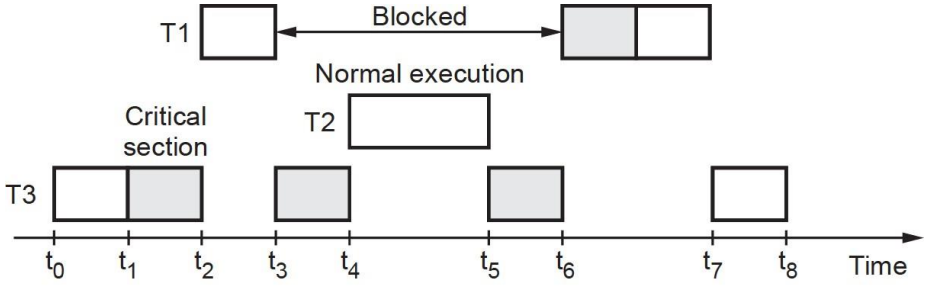
Consider three tasks Tl, T2 and T3 with decreasing priorities. Task T1 and T3 share some data or resource that requires exclusive access, while T2 does not interact with either of the other two tasks.

Task T3 starts at time t0 and locks semaphore s at time tv At time t2, Tl arrives and preempts T3 inside its critical section. After a while, Tl requests to use the shared resource by attempting to lock s, but it gets blocked, as T3 is currently using it. Hence, at time t3

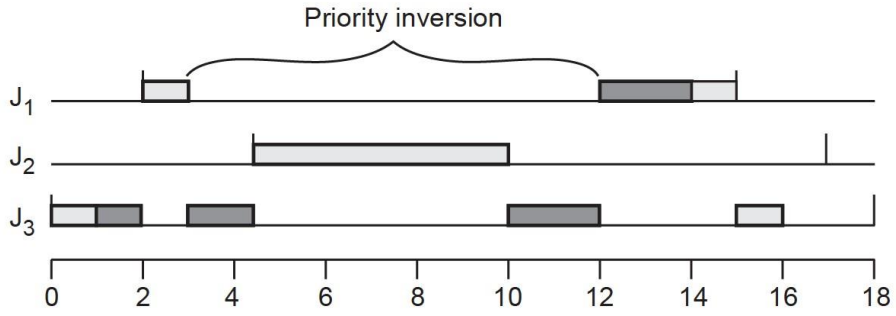
continues to execute inside its critical section. Next, when T2 arrives at time t4, it preempts T3, as it has a higher priority and does not interact with either Tl or T3.

The execution time of T2 increases the blocking time of Tl, as it is no longer dependent solely on the length of the critical section executed by T3.

When tasks share resources, there may be priority inversions.



Priority inversion method



Priority inversion example

Priority inversion is not avoidable; However, in some cases, the priorityinversion could be too large.

Simple solutions :

1. Make critical sections non-preemptable.
2. Execute critical sections at the highest priority of the task that could use it.

The solution of the problem is rather simple; while the low priority task blocks an higher priority task, it inherits the priority of the higher priority task; in this way, every medium priority task cannot make preemption.

Embedded C

Embedded C is an extension of C language and it is used to develop micro- controller-based applications. The extensions in the Embedded C language from normal C Programming Language are the I/O Hardware Addressing, fixed-point arithmetic operations, accessing address spaces, etc.

Embedded C Program has five layers of Basic Structures. They are:

- **Comment:** These are simple readable text, written in code to make it more understandable to the user. Usually comments are written in // or /* */.
- **Pre-processor directives:** The Pre-Processor directives tell the compiler which files to look in to find the symbols that are not present in the program.
- **Global Declaration:** The part of the code where global variables are defined.
- **Local Declaration:** The part of the code where local variables are defined.
- **Main function:** Every C program has a main function that drives the whole code. It basically has two parts the declaration part and the execution part. Where, the declaration part is where all the variables are declared, and the execution part defines the whole structure of execution in the program.

In nature it uses a cross-platform development scheme, i.e., the development of the application by it is **platform-independent** and can be used on multiple platforms.

Difference between Embedded C and C

The following table list the major differences between the C and Embedded C programming languages:

Parameters	C	Embedded C
General	• C is a general-purpose programming language, which can be used to design any type of desktop-based application. • It is a type of high-level language.	• Embedded C is simply an extension of C language and it is used to develop micro-controller-based applications. • It is nothing but an extension of C.

Parameters	C	Embedded C
Dependency	• C language is a hardware-independent language. • C compilers are OS-dependent.	• Embedded C is a fully hardware-dependent language. • Embedded C is OS-independent.

Compiler	• For C language, the standard compilers can be used to compile and execute the program.
	• Popular Compiler to execute a C language program are:
	  ◦ GCC (GNU Compiler collection)
	  ◦ Borland turbo C,
	  ◦ Intel C++
		• For Embedded C, specific compilers that are able to generate particular hardware/micro-controller-based output are used.
		• Popular Compiler to execute an Embedded C language program are:
		  ◦ Keil compiler
		  ◦ BiPOM ELECTRONIC
		  ◦ Green Hill Software

Usability and Applications	• C language has a free format of program coding.
	• It is specifically used for desktop applications.
	• Optimization is normal.
	• It is very easy to read and modify the C language.
	• Bug fixing is very easy in a C language program.
	• It supports other various programming languages during application.
	• Input can be given to the program while it is running.
	• Applications of C Program:
	  ◦ Logical programs
	  ◦ System software programs
		• Formatting depends upon the type of microprocessor that is used.
		• It is used for limited resources like RAM and ROM.
		• High level of optimization.
		• It is not easy to read and modify the Embedded C language.
		• Bug fixing is complicated in an Embedded C language program.
		• It supports only the required processor of the application and not the programming languages.
		• Only the pre-defined input can be given to the running program.
		• Applications of Embedded C Program:
		  ◦ DVD
		  ◦ TV
		  ◦ Digital camera

UNIT III
IOT AND ARDUINO PROGRAMMING

Introduction to the Concept of IoT Devices – IoT Devices Versus Computers – IoT Configurations – Basic Components – Introduction to Arduino – Types of Arduino – Arduino Toolchain – Arduino Programming Structure – Sketches – Pins –Input/Output From Pins Using Sketches – Introduction to Arduino Shields –Integration of Sensors and Actuators with Arduino.

WhatIsIot:

IoT stands for Internet of Things. It refers to the interconnectedness of physical devices, such as appliances and vehicles, that are embedded with software, sensors, and connectivity which enables these objects to connect and exchange data. This technology allows for the collection and sharing of data from a vast network of devices, creating opportunities for more efficient and automated systems.

Internet of Things (IoT) is the networking of physical objects that containelectronics embedded within their architecture in order to communicate and senseinteractionsamongsteachotherorwithrespecttotheexternalenvironment.Inthe upcomingyears,IoT-basedtechnologywillofferadvancedlevelsofservicesand practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of thecategorical examples where IoT is strongly established.

IoT is network of interconnected computing devices which are embedded in everyday objects, enabling them to send and receive data. Over 9 billion „Things" (physical objects) are currently connected to the Internet, as of now. In the near future, this number is expected to rise to a whopping 20 billion.

There are two ways of building IoT:

1. For ma separate internet work including only physical objects.
2. Make the Internet ever more expansive, but this requires hard-core technologies such as rigorous cloud computing and rapid big data storage (expensive).

In the near future, IoT will become broader and more complex in terms of scope. It will change the world in terms of "anytime, anyplace, anything in connectivity."

IoTEnablers:

RFIDs: uses radio waves in order to electronically track the tag attached to each physical object.

Sensors: devices that are able to detect changes in an environment (ex:motion detectors).

Different types of Sensors:

1. Temperature Sensors
2. Image Sensors
3. Gyro Sensors
4. Obstacle Sensors
5. RF Sensor
6. IR Sensor
7. MQ-02/05GasSensor
8. LDR Sensor
9. Ultrasonic Distance Sensor

Nano technology: as the name suggests, these are extremely small devices with dimensions usually less than a hundred nanometers.

Smart networks:(ex:meshtopology).

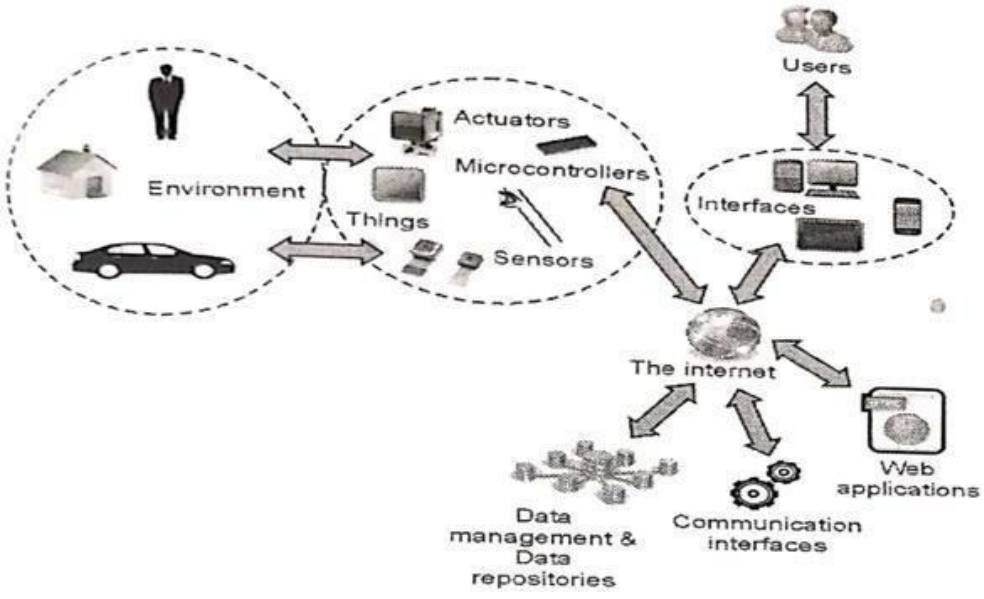
Working with IoT Devices:

CollectandTransmitData: Forthispurposesensorsarewidely usedtheyareusedas per requirements in different application areas.

Actuate device based on triggers produced by sensors or processing devices : If certain condition is satisfied or according to user‟s requirements if certain trigger is activatedthen which action to performed that is shown by Actuator devices.

Receive Information : From network devices user or device can take certain information also for their analysis and processing purposes.

Communication Assistance : Communication assistance is the phenomena of communication between 2 network or communication between 2 or more IoT devices of same or different Networks. This can be achieved by different communication protocols like : MQTT , Constrained Application Protocol, ZigBee, FTP, HTTP etc.



WorkingofIoT

CharacteristicsoftheInternetofThings

The Internet of Things (IoT) is characterized by the following key features that arementioned below.

1. Connectivity

Connectivity is an important requirement of the IoT infrastructure. Things of IoT should be connected to the IoT infrastructure. Anyone, anywhere, anytime can connect, this should be guaranteedatalltimes. For example,the connectionbetweenpeople throughInternet devices like mobile phones, and other gadgets, also a connection between Internet devices such as routers, gateways, sensors, etc.

2. IntelligenceandIdentity

The extraction of knowledge from the generated data is very important. For example, asensor generates data, but that data will only be useful if it is interpreted properly. Each IoT device has a unique identity. This identification is helpful in tracking the equipment and at times for querying its status.

3. Scalability

The number of elements connected to the IoT zone is increasing day by day. Hence, an IoT setup should be capable of handling the massive expansion. The data generated as anoutcome is enormous, and it should be handled appropriately.

4. Dynamic and Self-Adapting (Complexity)
IoT devices should dynamically adapt themselves to changing contexts and scenarios. Assume a camera meant for surveillance. It should be adaptable to work in different conditions and different light situations (morning, afternoon, and night).

1. **Architecture**
   IoT architecture cannot be homogeneous in nature. It should be hybrid, supporting different manufacturers' products to function in the IoT network. IoT is not owned by any one engineering branch. IoT is a reality when multiple domains come together.
2. **Safety**
   There is a danger of the sensitive personal details of the users getting compromised when all his/her devices are connected to the internet. This can cause a loss to the user. Hence, data security is the major challenge. Besides, the equipment involved is huge. IoT networks may also be at risk. Therefore, equipment safety is also critical.
3. **Self-Configuring**
   This is one of the most important characteristics of IoT. IoT devices are able to upgrade their software in accordance with requirements with a minimum of user participation. Additionally, they can set up the network, allowing for the addition of new devices to an already-existing network.
4. **Interoperability**
   IoT devices use standardized protocols and technologies to ensure they can communicate with each other and other systems. Interoperability is one of the key characteristics of the Internet of Things (IoT). It refers to the ability of different IoT devices and systems to communicate and exchange data with each other, regardless of the underlying technology or manufacturer.
5. **Embedded Sensors and Actuators**
   Embedded sensors and actuators are critical components of the Internet of Things (IoT). They allow IoT devices to interact with their environment and collect and transmit data.
6. **Autonomous Operation**
   Autonomous operation refers to the ability of IoT devices and systems to operate independently

and make decisions without human intervention. This is a crucial characteristic of the Internet of Things (IoT) and enables a wide range of new applications and services.

Here is the paragraph with proper spacing:

Configuring an Internet of Things (IoT)
Identify your IoT components: Determine the devices and sensors you will be using in your IoT network. These could include sensors, actuators, gateways, and edge devices.

Choose a Communication Protocol:
Decide on the communication protocol you'll use for your devices to exchange data. Common protocols include MQTT, CoAP, HTTP, and AMQP. Choose a protocol based on factors like the type of data you'll be transmitting, the reliability needed, and the network environment.

Set Up Network Infrastructure:
Ensure you have a stable and secure network infrastructure in place. This might include Wi-Fi, Ethernet, cellular, or even LoRaWAN for long-range low-power communication.

Device Registration and Onboarding:
Register each IoT device to your network. This might involve provisioning them with necessary credentials (such as certificates) to ensure secure communication.

Security Measures:
Implement security mechanisms to protect your IoT ecosystem from unauthorized access, data breaches, and cyberattacks. This could involve using encryption, secure bootstrapping, device authentication, and firewalls.

Data Collection and Transmission:
Configure devices to collect the required data from sensors and other sources. Set up rules for data transmission frequency, conditions, and thresholds. Ensure that data is sent efficiently to minimize network usage.

Data Processing at the Edge: If you're using edge computing, configure devices or gateways to process data locally before sending it to the cloud. This reduces latency and conserves bandwidth.

Cloud Integration: Integrate your IoT system with cloud platforms like AWS IoT, Microsoft Azure IoT, or Google Cloud IoT. Configure end points, authentication, and data routing to the cloud services.

Data Storage and Analytics: Set up data storage and analytics pipelines to process and analyze the incoming data. This might involve using databases, data lakes, and analytics tools to gain insights from the collected data.

Remote Device Management: Implement mechanisms for remote device management, such as firm ware updates, configuration changes, and diagnostics. Over-the-air(OTA)update scan be critical for maintaining device security and functionality.

Scalability Planning: Consider how your IoT system will scale as you add more devices and sensors. Design the architecture in a way that allows easy scalability without major disruptions.

Monitoring and Maintenance: Set up monitoring tools to keep track of the health, performance, and security of your IoT devices and network. Implement automated alerts for

abnormal conditions.

Compliance and Regulations: Ensure that your IoT system complies with relevant regulations and standards, especially if it involves sensitive or personal data.

Testing and Iteration: Thoroughly test your IoT configuration in a controlled environment before deploying it at scale. Make necessary adjustments based on testing results and user feedback.

Documentation: Maintain comprehensive documentation of your IoT configuration, including device specifications, network settings, security measures, and communication protocols. This documentation will be valuable for trouble shooting and future enhancements.

TypesofIoT Applications:

IoT devices span a wide range of industries and applications, including:

Home Automation: Smart thermostats, doorbell cameras,smart lights, and connected appliances.

Healthcare: Wearable fitness trackers, remote patient monitoring devices, and medical implants.

Industrial IoT(IIoT): Sensors and monitors used in manufacturing, supply chain management, and predictive maintenance.

Agriculture: Soil moisture sensors, GPS trackers for livestock, and automated irrigation systems.

SmartCities: Connected street lights,waste management systems, and traffic monitoring.

Automotive: Connected vehicles, vehicle-to-vehicle(V2V) communication,and self-driving cars.

BenefitsofIoT Devices:

Efficiency: IoTdevicescanoptimizeprocesses,reducewaste,andenhanceresource utilization.

Automation: Taskscanbeautomatedbasedonreal-timedata,leadingtoimproved productivity.

Data-DrivenInsights: IoTdevicesprovidedatathatcanbeanalyzedtogaininsights,make informed decisions, and identify trends.

RemoteMonitoringand Control: Devicescanbemonitoredandcontrolledremotely, enabling real-time adjustments.

EnhancedUserExperience: IoTdevicescreatepersonalizedexperiencesandconvenience for users.

ChallengesandConsiderations:

Securityand Privacy: IoTdevicescanbevulnerabletocyberattacks ifnotproperlysecured.

Interoperability: Ensuringthatdevicesfromdifferentmanufacturerscancommunicate seamlessly.

Scalability: Handlingalargenumber ofdevices anddata pointscan bechallenging.

PowerManagement: Ensuringdeviceshaveadequatepowerandoptimizingenergy consumption.

DataManagement: Efficientlyhandlingandprocessingthevastamountofdatagenerated by IoT devices.

BasiccomponentsofIoT:

Devices/Things: These are physical objects embedded with sensors, actuators, and communication modules that enable them to collect data, perform actions, and communicate over the internet. Devices can range from simple sensors to complex machinery.

Sensors: Sensors are components that detect changes in the environment and convert physical phenomena (like temperature, light, pressure, humidity, etc.) into electrical signals that can be processed by IoT devices.

Actuators: Actuators are devices that perform actions based on the data received from sensors or commands from remote systems. Examples include motors, servos, solenoids, and relays.

Computing Units: These are the computing units within IoT devices. They process data from sensors, control actuators, and execute programmed instructions. They also manage communication with other devices and systems.

Communication Modules: IoT devices need communication capabilities to send and receive data over the internet. Communication modules can include Wi-Fi, Bluetooth, Zigbee, LoRa, cellular, Ethernet, and more.

Network Infrastructure: This includes the underlying network that enables devices to connect to the internet. It can be wired (like Ethernet) or wireless (like Wi-Fi, cellular, or satellite).

Internet Connectivity: The devices need access to the internet to communicate with other devices and central systems. This connectivity can be through Wi-Fi, cellular networks, satellite connections, or other means.

Cloud Platforms: Cloud platforms provide storage, computing power, and services that enable data processing, analysis, storage, and remote control of IoT devices. Cloud services facilitate the scalability and management of IoT solutions.

Data Processing and Storage: IoT devices generate a vast amount of data. Data processing involves analyzing and interpreting this data to extract meaningful insights. Data storage involves storing the collected data for future reference and analysis.

User Interfaces: User interfaces can be web applications, mobile apps, or dashboards that allow users to monitor and control IoT devices remotely. These interfaces provide real-time information and enable users to set preferences or receive alerts.

Security: Security is critical in IoT to protect data, devices, and networks from unauthorized access and cyberattacks. Security measures include encryption, authentication, access controls, and regular software updates.

Power Management: IoT devices often operate on limited power sources such as batteries. Power management strategies are essential to optimize energy consumption and extend the device's operational lifespan.

Data Analysis: Analyzing the data collected from IoT devices can yield valuable insights for making informed decisions, predicting trends, and improving operations.

Machine Learning and Artificial Intelligence (AI): IoT systems can leverage machine learning and AI to improve automation, anomaly detection, and decision-making based on real-time data.

ModernApplications:

Smart Grids and energy saving
Smart cities
Smart homes/Home automation
Healthcare
Earthquake detection
Radiation detection/hazardous gas detection
Smartphone detection
Water flow monitoring
Traffic monitoring

Wearables
Smart door lock protection system
Robots and Drones
Healthcare and Hospitals, Telemedicine applications
Security
Biochip Transponders (For animals in farms)

Heart monitoring implants (Example: Pacemaker, ECG real-time tracking)

AdvantagesofIoT:

1. Improved efficiency and automation of tasks.
2. Increased convenience and accessibility of information.
3. Better monitoring and controlof devices and systems.
4. Greater ability together and analyze data.
5. Improved decision-making.
6. Cost savings.

Disadvantages of IoT:

Security concerns and potential for hacking or data breaches.
Privacy issues related to the collection and use of personal data.
Dependence on technology and potential for system failures.
Limited standardization and interoperability among devices.
Complexity and increased maintenance requirements.
High initial investment costs.
Limited battery life on some devices.
Concerns about job displacement due to automation.
Limited regulation and legal framework for IoT, which can lead to confusion and uncertainty.

Overview of IoT Vs Computers:

One big difference between IoT devices and computers is that the main function of IoT devices is not to compute(not to be a computer) and the main function of a computer is to compute functions and to run programs. But on IoT devices that is not its main point, it hassomeotherfunctionbesidesthat.Asanexamplelikeincars,thefunctionofIoTdevicesare not to compute anti-lock breaking or to do fuel injection, their main function from the point of view of a user is to be driven and to move you from place to place and the computer is just to help that function. For example, The main function of the car is not to compute like anti-lock breaking or to do fuel injection their main function from the point of view of a user isto drive, to move you from place to place. But when we embed software in it then the software can beable for fuel limit detection.

Difference between IoT devices and Computers:

IOT Devices	Computers
IoT devices are special-purpose devices.	Computers are general-purpose devices.
IoT devices can do only a particular task for which it is designed.	Computer scan do so many tasks.
The hardware and software built-in the IoT devices are streamlined for that particular task.	The hardware and software built-in the computers are streamlined to do many tasks(such as calculation, gaming, music player, etc.)
IoT devices can be cheaper and faster at a particular task than computers, as IoT devices are made to do that particular task.	A computer can be expensive and slower at a particular task than an IoT device.
Examples: Music Player-iPod, Alexa, smart cars, etc.	Examples :Desktop computers, Laptops, etc.

Introduction to Arduino

Arduino is a project, open-source hardware, and software platform used to design and build electronic devices. It designs and manufactures microcontroller kits and single-board interfaces for building electronics projects.

The Arduino boards were initially created to help the students with the non-technical background. The designs of Arduino boards use a varietyof controllers and microprocessors.

The Arduino board consists of sets of analog and digital I/O (Input / Output) pins, which are further interfaced to **breadboard, expansion boards,** and other**circuits**. Such boardsfeaturethemodel,UniversalSerialBus(**USB**),and**serialcommunicationinterfaces**, which are used for loading programs from the computers.

Types ofArduino

The flexibility of the Arduino board is enormous so that one can do anything they imagine. This board can be connected very easily to different modules such as obstacle sensors, presence detectors, fire sensors, GSM Modules GPS modules, etc.

The main function of the Arduino board is to control electronics through reading inputs &changing it into outputs because this board works like a tool. This boardis also used to make different electronics projects in the field of electronics, electrical, robotics, etc.

Features of Different Types of Arduino Boards

The features of different types of Arduino boards are listed in tabular form.

Arduino Board	Processor	Memory	Digital I/O	Analogue I/O
Arduino Uno	16MhzATmega328	2KBSRAM, 32KB flash	14	6 input, 0 output
Arduino Due	84MHz AT91SAM3X8E	96KBSRAM, 512KB flash	54	12 input, 2 output
Arduino Mega	16MHz ATmega2560	8KBSRAM, 256KB flash	54	16 input, 0 output
Arduino Leonardo	16MHz ATmega32u4	2.5KBSRAM, 32KB flash	20	12 input, 0 output

Different Types Of Arduino Boards

The list of Arduino boards includes the following such as

Arduino Uno(R3)
Arduino Nano
Arduino Micro

ArduinoDue
LilyPadArduinoBoard
Arduino Bluetooth
Arduino Diecimila
RedBoardArduino Board
ArduinoMega(R3) Board
ArduinoLeonardoBoard
Arduino Robot
Arduino Esplora
ArduinoPro Mic
Arduino Ethernet
ArduinoZero
FastestArduino Board

Arduino Uno(R3)

The Uno is a huge option for your initial Arduino. This Arduino board depends on an ATmega328P based microcontroller. As compared with other types of arduino boards, it is verysimpletouselikettheArduinoMegatypeboard..Itconsistsof14-digital I/Opins, where 6-pins can be used as PWM(pulse width modulation outputs), 6-analog inputs, a reset button, a power jack, a USB connection, an In-Circuit Serial Programming header (ICSP), etc.

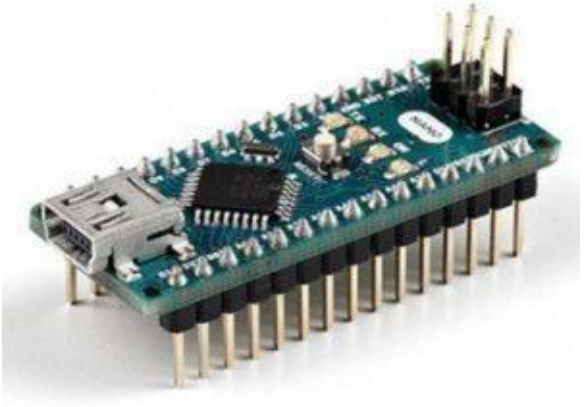
It includes everything required to hold up the microcontroller; simply attach it to a PC with the help of a USB cable and give the supply to get started with an AC-to-DC adapter or battery.

ArduinoUno(R3)

Arduino Uno is the most frequently used board and it is the standard form apart from all the existing Arduino Boards. This board is very useful for beginners. Please refer to this link to know more about Arduino Uno Board

Arduino Nano

This is a small board based on the microcontrollers like ATmega328P otherwise ATmega628 but the connection of this board is the same as to the Arduino UNO board. This kind of microcontroller board is very small in size, sustainable, flexible, and reliable.

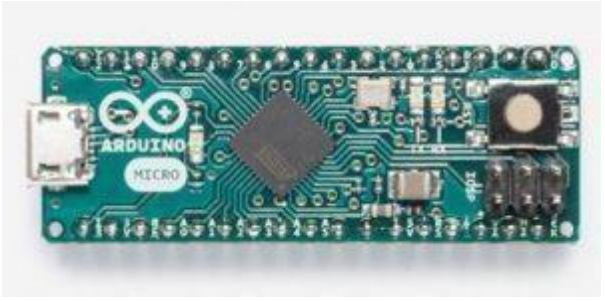


ArduinoNano

As compared with the Arduino Uno board, it is small in size. The devices like mini USB and Arduino IDE are necessaryto build the projects. This board mainlyincludes analog pins-8, digital pins-14 with the set of an I/O pin, power pins-6 & RST (reset) pins-2. Please refer to this link to know more about Arduino Nano Board.

Arduino Micro

The Arduino Micro board mainly depends on the ATmega32U4 based Microcontroller that includes 20-sets of pins wherethe7-pins are PWM pins, 12-analoginput pins. This board includes different components like an ICSP header, RST button, small USB connection, crystal oscillator-16MHz. The USB connection is inbuilt and this board is the shrunk version of the Leonardo board.



Arduino Micro

Arduino Due

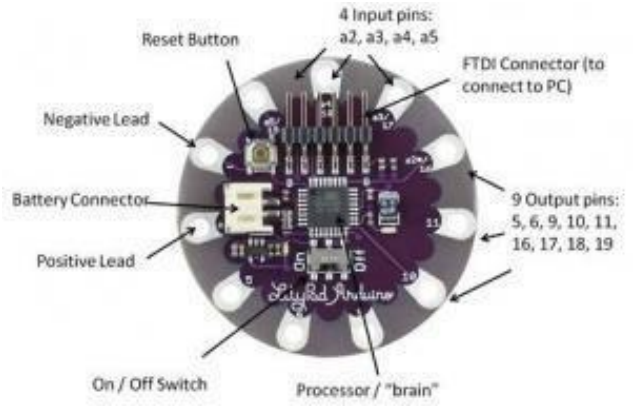
This Arduino board depends on the ARM Cortex-M3 and it is the first Arduino microcontroller board. This board includes digital I/O pins-54 where 12-pins are PWM o/p pins, analog pins -12, UARTs-4, a CLK with 84 MHz, an USB OTG, DAC-2, a power jack, TWI-2, a JTAG header, an SPI header, two buttons for reset & erase.

ArduinoDue

This board works with 3.3V where the highest voltage that the pins of input/outputcan stand is 3.3V because providing a high voltage to any I/O pin can lead to damage the board This board is simply connected to a computer through a small USB cable otherwise it can be powered through an AC to DC adapter. This Arduino Due board is suitable with all shields of Arduino at 3.3V.

LilyPad Arduino Board

The Lily Pad Arduino board is a wearable e-textile technology expanded by Leah " Buechley" and considerately designed by "Leah and SparkFun". Each board was imaginatively designed with huge connecting pads & a smooth back to let them to be sewn into clothing using conductive thread. This Arduino also comprises of I/O, power, and also sensor boards which are built especially for e-textiles. These are even washable!



LilyPadArduinoBoards

Arduino Bluetooth

This Bluetooth mainly depends on the microcontroller like ATmega168 and thisboard is also called Arduino BT. This kind of board includes different components likedigital pins-16, analog pins-6, crystal oscillator-16MHz, reset button, screw terminals, ICSP header. In this board, the screw terminals are mainly used for power. The programming ofthis Bluetooth microcontroller can be done with Bluetooth like a wireless connection.

Arduino Diecimila

The microcontroller board like Arduino Diecimila mainly depends on the ATmega168. This board includes digital I/O pins -14 where 6-pins can be used like PWM outputs & analog inputs-6, a USB connection, a crystal oscillator-16 MHz, an ICSP header, a reset button & a power jack. This board can be connected to a computer through a USB cable and it can be activated using a battery and an AC-DC adapter.

Arduino Diecimila

As the name suggests, the meaning of Diecimila in Italian is 10,000 which means that marks the truth that above 10k Arduino boards have been designed. In a set of USB Arduino boards, it is the latest one as compared with other versions.

Red Board Arduino Board

The Red Board Arduino board can be programmed using a Mini-B USB cable using the Arduino IDE. ItwillworkonWindows8withouthavingtomodifyyoursecuritysettings. It is more constant due to the USB or FTDI chip we used and also it is entirely flat on the back. Creating it is very simple to utilize in the project design. Just plug the board, select the menu option to choose an Arduino UNO and you are ready to upload the program. You can control the Red Board over a USB cable using the barrel jack.

RedBoardArduino Boards

Arduino Mega(R3)Board

The Arduino Mega is similar to the UNO‟s big brother. It includes lots of digital I/O pins (from that, 14-pins can be used as PWM o/ps), 6-analog inputs, a reset button, a power jack, a USB connection, and a reset button. It includes everything required to hold up the microcontroller; simply attach it to a PC with the help of a USB cable and give the supply to get started with an AC-to-DC adapter or battery. The huge number of pins make this Arduino board very helpful for designing projects that need a bunch of digital i/ps or o/ps like lots of buttons. Please refer to this link to know more about Arduino Mega (R3) Board

ArduinoMega(R3) Board

Arduino Leonardo Board

The first development board of an Arduino is the Leonardo board. This board usesone microcontroller along with the USB. That means, it can be very simple and cheap also. Because this board handles USB directly, program libraries are obtainable which let the Arduino board to follow a keyboard of the computer, mouse, etc.

ArduinoLeonardoBoard

Arduino Robot

This kind of board is the first Arduino over wheels. This Arduino robot includes two processors on each of its boards. The two boards are the motor board and control board where the motor board controls the motors & the control board is used to read the sensors for operating. Every board is a complete Arduino board and its programming can be done through the Arduino IDE. These are microcontroller boards that depend on the ATmega32u4.

The pins of this Robot are mapped to actuators and sensors onboard.The process of programming the robot is the same as the Arduino Leonardo board. It is also named a small computer and it is extensively used in robotics.

This board includes the speaker, color screen, buttons-5, motors-2, a digital compass, an SD card reader, potentiometers-2 & floor sensors-5. The library of this robot can be used for controlling the sensors as well as the actuators.

Arduino Esplora

The Arduino Esplora includes a small computer known as a microcontroller including a number of inputs & outputs. The inputs of this board are a light sensor, four buttons, a microphone, an accelerometer, joystick, a slider, a temperature sensor, etc whereas the outputs are a 3 color LED, a buzzer. This kind of Arduino board looks like a videogame controller.

The programming of this board can be done using Arduino Software like IDE which takes the data from the inputs and controls the output like a keyboard or a mouse. As compared with all other types of Arduino boards, this esplora is totally different because the inputs, as well as outputs, are connected to the board already.

Arduino Esplora

So connecting the components like actuators or sensors is very simple. Thus, programming is somewhat different as compared with other types of Arduino boards. This esplora board includes its own library so that the data from the sensors & actuators are very easy to read and write.

Arduino ProMic

The Arduino Pro Micro board is the same as the Arduino Mini board apart from the ATmega32U4 Microcontroller. This pro mic board includes digital I/O pins-12, pulse width modulation (PWM) pins-5, serial connections of Tx & Rx &10-bit ADC.

Arduino Ethernet

The Arduino Ethernet board depends on the microcontroller like ATmega328. This kind of microcontroller board includes analog pins-5, digital I/O pins-14, RST button, an RJ45 connection, crystal oscillator, a power jack, ICSP header, etc. The connection of the Arduino board can be done through the Ethernet shield to the internet.

Arduino Zero

This is a powerful as well as simple 32-bit board and it provides the best platform for innovative projects like wearable technology, smart IoT devices, crazy robotics, high-tech automation, etc. This board expands by providing improved performance, permitting a range of project opportunities & performs like a great educational tool.

ArduinoZero

Thisboardincludesanaloginputpins-6,digitalI/Opins-14,apowerjack,AREF button, UART port pins, a USB connector & an In-Circuit Serial Programming (ICSP) header, a power header, etc.

This board is power-driven through the SAMD21 microcontroller based on Atmel. The main feature of this is EDBG (Embedded Debugger) based on Atmel and it provides a complete debug interface without using extra hardware.

Fastest Arduino Board

Designing one of the best Arduino development boards that are familiar with Arduino MEGA & UNO is the hifive1 board that includes a 320 MHz RISC-V microcontroller unit. This kind of fastest board has Cortex M-7 with a 400 MHz microcontroller unit.

Flash memory– upto2Mbytes

RAM– 1 Mbyte

DMAcontrollers-4

Communication peripherals-Upto 35

16-bitMaxResolutionwith3×ADCs

D/Aconverterswith2× 12-bit

Hardware with JPEG Codec

Timers-22&Watchdogs– 200Mhz

HW Calendar &RTC with Sub-second Accuracy

Cryptographic Acceleration

Toolchain(IDE)

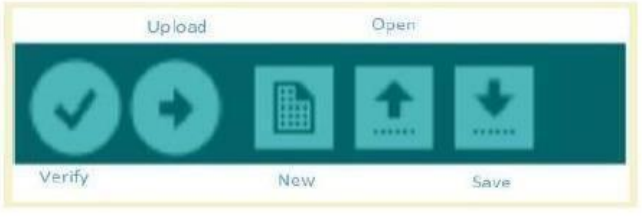
A toolchain is a set of programming tools that is used to perform a complex set of operations. In the Arduino Software (IDE), the toolchain is hidden from the user, but it is used to compile and upload the user Sketch. It includes a compiler, assembler, linker, and Standard C & math libraries.The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL.

Sketch− Program coded in Arduino IDE is called a SKETCH

1. To create a new sketch: File -> New

   To open an existing sketch: File -> Open
   There are some basic ready-to-use sketches available in the **EXAMPLES** section
   File -> Examples -> Select any program

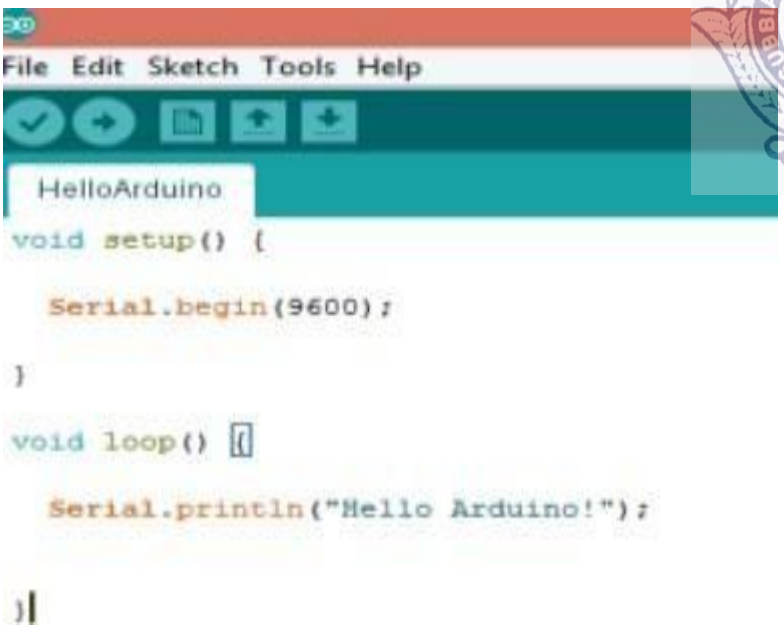
2. **Verify:** Checks the code for compilation errors
3. **Upload:** Uploads the final code to the controller board
4. **New:** Creates a new blank sketch with basic structure
5. **Open:** Opens an existing sketch
6. **Save:** Saves the current sketch



Compilation and Execution

SerialMonitor: Opens the serial console

All the data printed to the console are displayed here



StructureofSketch

A sketch can be divided into two parts:

Setup()

Loop()

Thefunctionsetup()isthepointwherethecodestarts,justlikethemain()functionin C and C++

I/OVariables, pinmodes are initializedintheSetup() function

Loop()function, as the namesuggests, iterates the specified task in the program

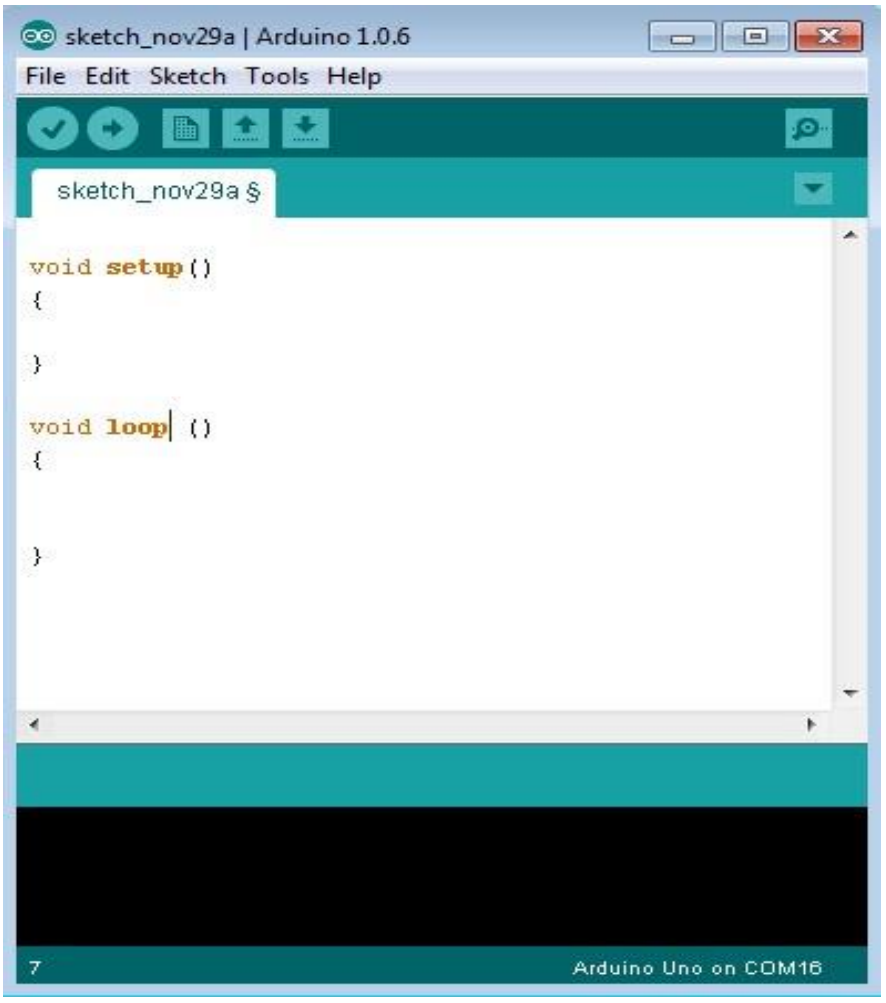
Structure

Arduino programs can be divided in three main parts: **Structure, Values** (variables and constants), and **Functions**. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the structure. Software structure consists of two main functions. −

Setup( ) function - The setup() function is called when a sketch starts. Use it to initializethe variables (not declaration),pin modes, start using libraries, etc. Thesetup function will only run once, after each power up or reset of the Arduino board.

Loop( ) function - After creating a setup() function, which initializes and sets theinitial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to activelycontrol the Arduino board



Voidsetup ( ){

}

PURPOSE− The**setup()**function is called when a sketch starts. Use it to initializethe variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

INPUT −-

OUTPUT−-

RETURN−-

VoidLoop( ){

}

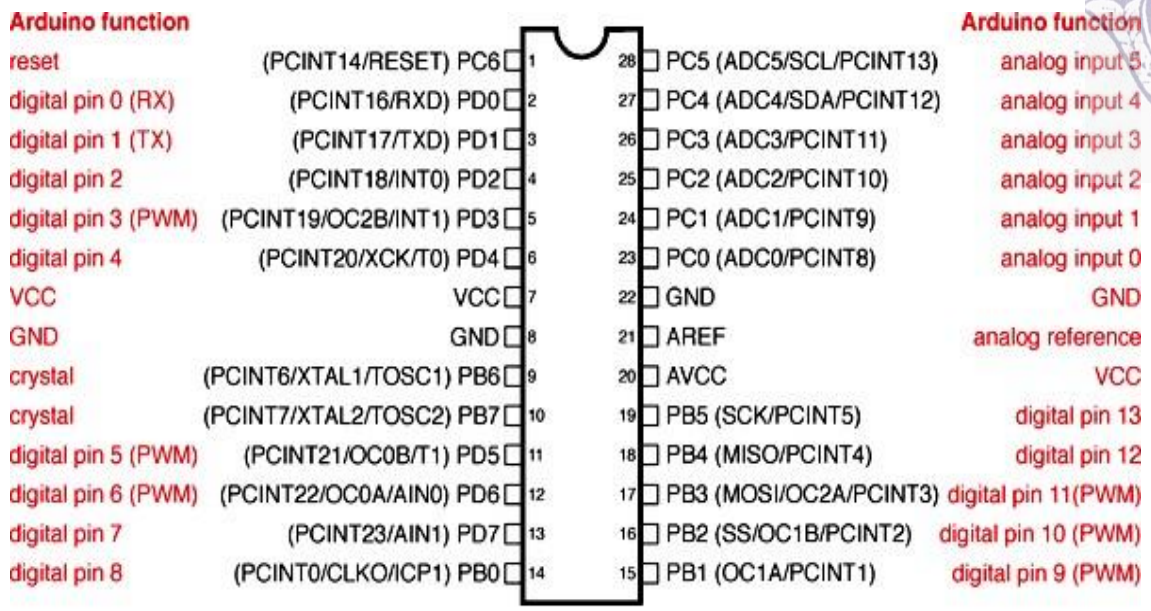
PURPOSE− After creating a**setup()**function, which initializes and sets the initial values, the**loop()**function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to activelycontrol the Arduino board.

INPUT −-

OUTPUT−-

RETURN−-

ArduinoUnoPinsDiagram



Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

ARDUINOI/O(Input,Output)Functions

1. Pin Mode()-The pin Mode()function is used to configure a specific pin to behave either as an input or an output.

pinMode() Function Syntax void setup () { pinMode (*pin , mode*); }	• ***pin*** – the number of the pin whose mode you wish to set • ***mode*** – INPUT, OUTPUT, or INPUT_PULLUP.

2. **digitalWrite()**-
ThedigitalWrite()functionisusedtowriteaHIGHoraLOWvaluetoadigitalpin.

digitalWrite() Function Syntax digitalWrite (*pin ,value*) **Example -** void loop() { digitalWrite (4 ,HIGH); }	• ***pin*** – the number of the pin whose mode you wish to set • ***value*** – HIGH, or LOW.

3. digitalRead()-Readsthevaluefromaspecifieddigitalpin,eitherHIGHorLOW.

digitalRead() Function Syntax
digitalRead(*pin*);
Example -
void loop()
{
val = digitalRead(4); *// read the input pin*
digitalWrite(7, val); *// sets the LED to the button's value*
}
	• ***pin*** - the number of the digital pin you want to read (*int*).
	
	Return -
	• **val** – HIGH, or LOW.

4. analogRead( ) function- we can read the voltage applied to one of the pins. This functionreturnsanumberbetween0and1023,whichrepresentsvoltagesbetween0 and 5 volts.

analogRead() function Syntax analogRead(*analogPin*); **Example -** void **loop**() { val = analogRead(analogPin); }	• ***analogPin*** – the number of the analog input pin to read from (0 to 5 on most boards, 0 to 7 on the Mini and Nano, 0 to 15 on the Mega) **Return -** • int (0 to 1023)

5. analogWrite()function-Writesananalogvaluetoapin.

analogWrite() function Syntax
 	
analogWrite(*Pin, value*)
 	
Example -
void **loop**()
{
 	
val = analogRead(Pin); *// read the input pin*
 	
analogWrite(ledPin, val / 4); *// analogRead values go from 0 to 1023, analogWrite values from 0 to 255*
 	
}
	• ***Pin*** - the pin to write to.
	• ***value*** - the duty cycle: between 0 (always off) and 255 (always on).

Input/Ouput(I/O)inArduino

These functions allow access to the pins

void pinMode(pin, Mode)

Setsapintoactaseitheraninputoranoutput
pinisthenumberofthepin
1-13forthedigitalpins
A0-A5fortheanaloginputpins
ModeistheI/Omodethepinissetto
INPUT,OUTPUT

DigitalInput

Int digital Read(pin)

Returns the state of an input pin
Returns either LOW(0volts)or HIGH(5volts) int pin

val;

Pin val=digitalRead(3);

pinvalissettothestateofdigitalpin3

Digital Output

Void digital Write(pin,value)

Assigns thestateofanoutputpin
AssignseitherLOW(0volts)orHIGH(5volts)

digitalWrite(3, HIGH);

Digitalpin3issetHIGH(5volts)

Analog Input

Int analog Read(pin)

Returns the state of analog input pin
Returns an integer from 0to1023
0 for 0 volts, 1023for 5volts int

pin val;

Pin val=analogRead(A3);

P in must be an analog pin

Analog Output

No pins can generate a true analog output
But PWMc an be used

ARDUINO SHIELDS

The advantages of using Arduino shields are listed below:

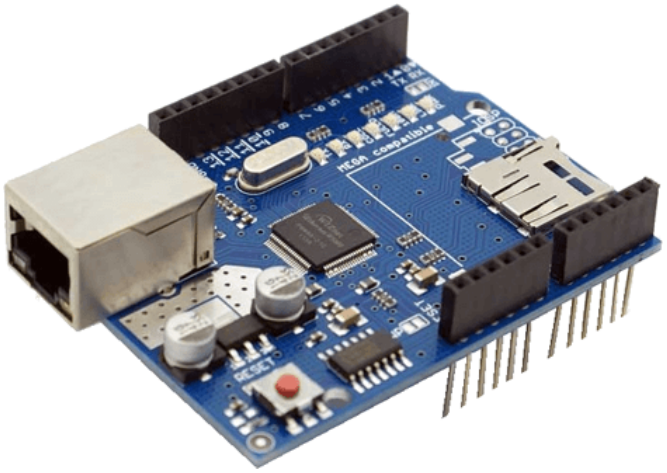
It adds new functionalities to the Arduino projects.
The shields can be attached and detached easily from the Arduino board. It does not require any complex wiring.
It is easy to connect the shields by mounting them over the Arduino board.
The hardware components on the shields can be easily implemented.

TypesofShields

The popular Arduino shields are listed below:

Ethernet shield
XBee shield
Proto shield
Relay shield
Motor shield
LCD shield
Bluetooth shield
Capacitive Touchpad shield
The Ethernet shields are used to connect the Arduino board to the Internet. We need to mount the shield on the top of the specified Arduino board.
The USB port will play the usual role to upload sketches on the board.
The latest version of Ethernet shields consists of a micro SD card slot. The micro SD card slot can be interfaced with the help of the SD card library.

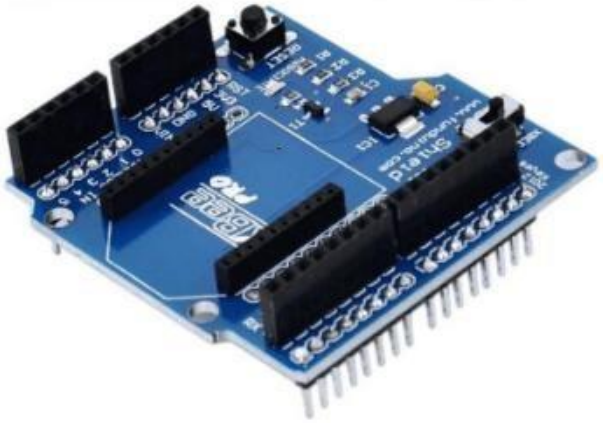
The Ethernet shield is shown below:



- We can also connect another shield on the top of the Ethernet shield. It means that we can also mount two shields on the top of the Arduino board.

Xbee Shield

- WecancommunicatewirelesslywiththeArduinoboardbyusingtheXbeeShield with Zigbee.
- Itreducesthehassleofthecable,whichmakesXbeeawirelesscommunication model.
- TheXbeewirelessmoduleallowsustocommunicateoutdoorupto300feetand indoor upto 100 feet.
- TheXbeeshield is shown below:

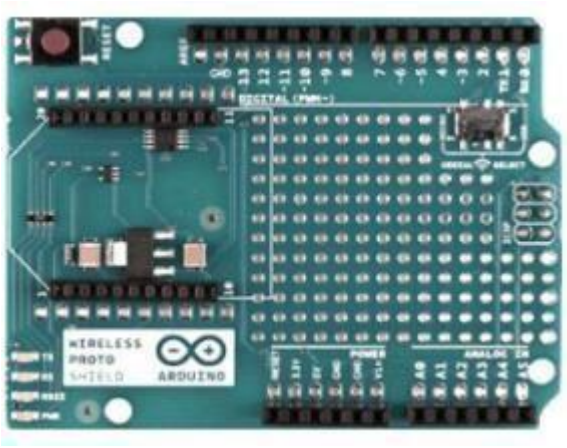


It can also be used with different models of XBee.

Proto shield

Proto shields are designed for custom circuits.
We can solder electronic circuits directly onto the shield.
The shield consists of two LED pads, two power lines, and SPI signal pads.
The IOREF (Input Output voltage REFerence) and GND (Ground) are the two power lines on the board.

The proto shield is shown below:



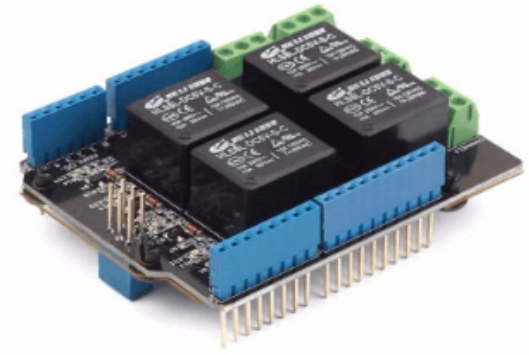
- We can also solder the SMD (Surface Mount Device) ICs on the prototyping area. A maximum of 24 pins can be integrated onto the SMD area.

Relay shield

Here is the text with proper spacing:

The Arduino digital I/O pins cannot bear the high current due to its voltage and current limits. The relay shield is used to overcome such situations. It provides a solution for controlling the devices carrying high current and voltage.
The shield consists of four relays and four LED indicators.
It also provides NO/NC interfaces and a shield form factor for simple connection to the Arduino board.
The LED indicators depict the ON/OFF condition of each relay.
The relay used in the structure is of high quality.
The NO (Normally Open), NC (Normally Closed), and COM pins are present on each relay.

The relay shield is shown below:



- The applications of the Relay shield include remote control, etc.

Motor shield

The motor shield helps us to control the motor using the Arduino board.
It controls the direction and working speed of the motor. We can power the motor shield either by the external power supply through the input terminal or directly by the Arduino.We can also measure the absorption current of each motor with the help of the motor shield.

- The motor shield is based on the L298 chip that can drive a step motor or two DC motors. L298 chip is a full bridge IC. It also consists of the heat sinker, which increases the performance of the motor shield.
- It can drive inductive loads, such as solenoids, etc.
- Theoperatingvoltageisfrom5Vto12V.

The Motor shield is shown below:

- Theapplications of themotor shieldareintelligentvehicles, micro-robots, etc.

LCDshield

The keypad of the LCD (Liquid Crystal Display) shield includes five buttons called up, down, left, right, and select.
There are 6 push buttons present on the shield that can be used as a custom menu control panel.
It consists of the 1602 white characters, which are displayed on the blue backlight LCD.
The LED present on the board indicates the power ON.
The five keys present on the board help us to make selections on menus and from the board to our project.

The LCD shield is shown below:

- The LCD shield is popularly designed for the classic boards such as Duemilanove, UNO, etc.

Bluetooth shield

- The Bluetooth shield can be used as a wireless module for transparent serial communication.
- It includes a serial Bluetooth module. D0 and D1 are the serial hardware ports in the Bluetooth shield, which can be used to communicate with the two serial ports (from D0 to D7) of the Arduino board.
- We can install Groves through the two serial ports of the Bluetooth shield called a Grove connector. One Grove connector is digital, while the other is analog.

The Bluetooth shield is shown below:

The communication distance of the Bluetooth shield is up to 10 m at home without any obstacle in between.

Capacitive Touchpad shield

It has a touchpad interface that allows integration of the Arduino board with the touch shield. The Capacitive touchpad shield consists of 12 sensitive touch buttons, which include 3 electrode connections and 9 capacitive touch pads.

The capacitive shield is shown below:

The board can work with the logic level of 3.3 V or 5 V.
We can establish a connection to the Arduino project by touching the shield.

INTEGRATION OF SENSORS AND ACTUATORS WITH ARDUINO.

Arduino is a popular microcontroller platform that allows you to easily interface with various sensors and actuators to create interactive projects. Integrating sensors and actuators with Arduino involves the following steps:

Selecting the appropriate sensors and actuators: Identify the specific sensors and actuators you need for your project. Arduino supports a wide range of sensors such as temperature sensors, motion sensors, light sensors, and actuators such as motors, servos, and LEDs.

Actuators are basically mechanical or electro mechanical devices. They convert energy or signals into motion. And mainly use to provide controlled motion to other components of various mechanical structures or devices.

Power supply: Ensure that your sensors and actuators have a suitable power supply. Arduino boards can usually provide power to low-power sensors and actuators directly from theirdigitaloranalogpins.However,high-powerdeviceslikemotorsmayrequirean external power source.

Wiring connections: Connect the sensors and actuators to the Arduino board using jumper wires or appropriate connectors. Arduino boards typically have digital input/output (I/O) pins, analog input pins, and power pins that you can use for wiring.

For digital sensors and actuators, you can connect them to the digital I/O pins. These pins can be configured as either input or output. Analog sensors can be connected to the analog input pins, which can read continuous voltage values.

Power and ground connections should be made to provide appropriate voltage and common reference for the sensors and actuators.

Library installation: Some sensors and actuators require specific libraries to be installed in the Arduino Integrated Development Environment (IDE). These libraries provide pre-written code and functions that make it easier to interface with the devices. You can install libraries by navigating to Sketch -> Include Library -> Manage Libraries in the Arduino IDE.

Code development: Write the Arduino code to read sensor data and control the actuators. The code should include appropriate functions and commands to initialize the sensors, read their values, and control the actuators based on the sensor inputs. The specific code will depend on the sensors and actuators you are using, so refer to their respective documentation and examples.

Uploading and testing: Upload the code to the Arduino board and test the integration. Use the Serial Monitor in the Arduino IDE to view sensor readings or debug any issues.Make sure the sensors and actuators are functioning as expected.

By following these steps, you can integrate sensors and actuators with Arduino to create interactive projects and prototypes. The possibilities are vast, ranging from simple projects like temperature monitoring and LED control to complex robotics and automation systems.

UNIT- 4 IOT COMMUNICATION AND OPEN PLATFORMS

IoT Communication Models and APIs – IoT Communication Protocols – Bluetooth – WiFi – ZigBee– GPS – GSM modules – Open Platform (like Raspberry Pi) – Architecture – Programming – Interfacing – Accessing GPIO Pins – Sending and Receiving Signals Using GPIO Pins – Connecting to the Cloud.

Communication Models in IoT (Internet of Things)

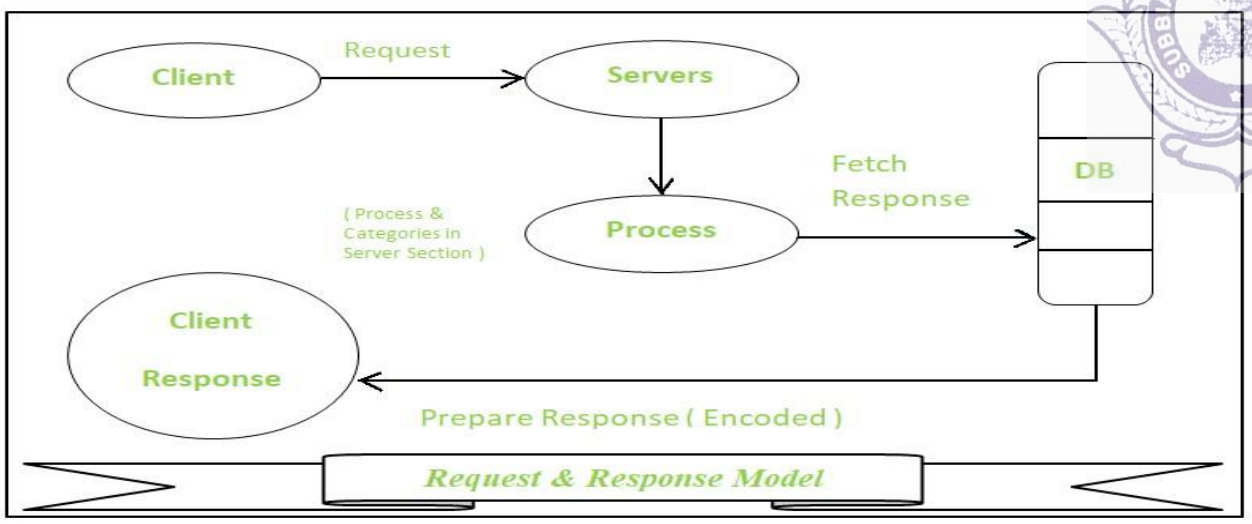
IoT devices are found everywhere and will enable circulatory intelligence in the future. For operational perception, it is important and useful to understand how various IoT devices communicate with each other. Communication models used in IoT have great value. The IoTs allow people and things to be connected any time, any space, with anything and anyone, using any network and any service.

Types of Communication Model :

1.Request & Response Model

This model follows a client-server architecture.

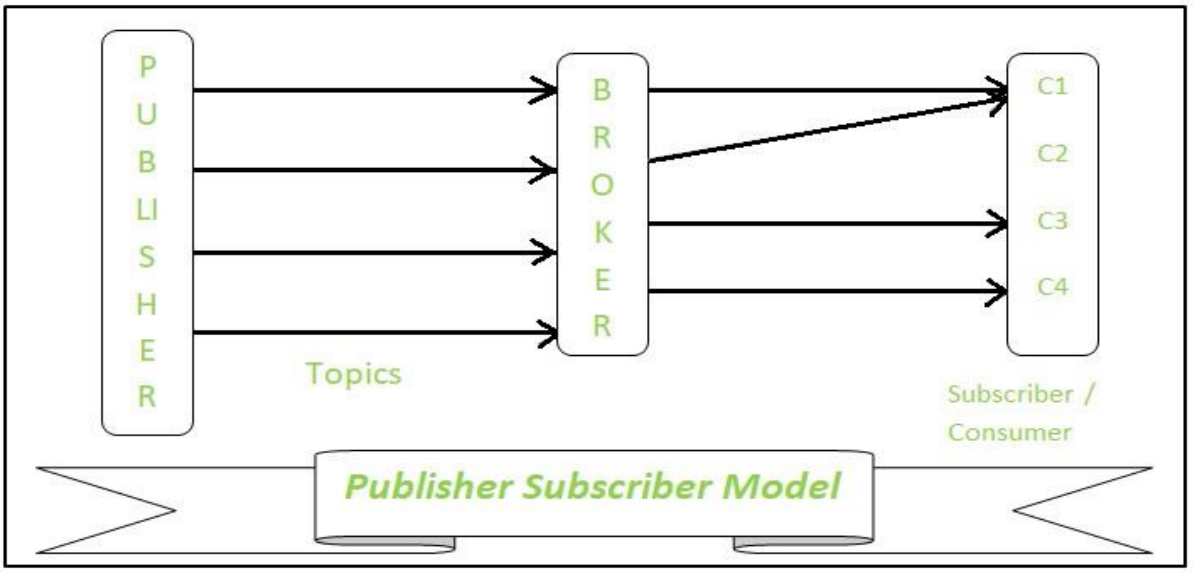
- The **client**, when required, requests the information from the server. This request is usually in the encoded format.
- This model is stateless since the data between the requests is not retained and each request is independently handled.
- The server Categories the request, and fetches the data from the database and its resource representation. This data is converted to response and is transferred in an encoded format to the client. The client, in turn, receives the response.
- On the other hand — In **Request-Response** communication model client sends a request to the server and the server responds to the request. When the server receives the request it decides how to respond, fetches the data retrieves resources, and prepares the response, and sends it to the client.



2. Publisher-Subscriber Model

This model comprises three entities: Publishers, Brokers, and Consumers.

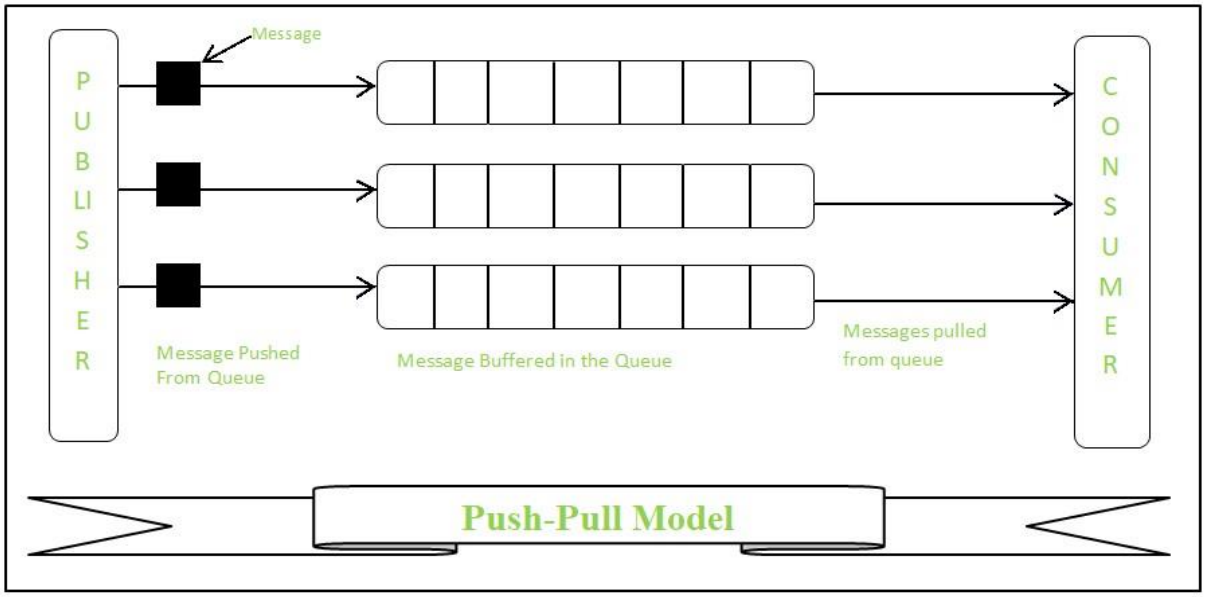
- **Publishers** are the source of data. It sends the data to the topic which are managed by the broker. They are not aware of consumers.
- **Consumers** subscribe to the topics which are managed by the broker.
- Hence, **Brokers** responsibility is to accept data from publishers and send it to the appropriate consumers. The broker only has the information regarding the consumer to which a particular topic belongs to which the publisher is unaware of.



3. Push-Pull Model

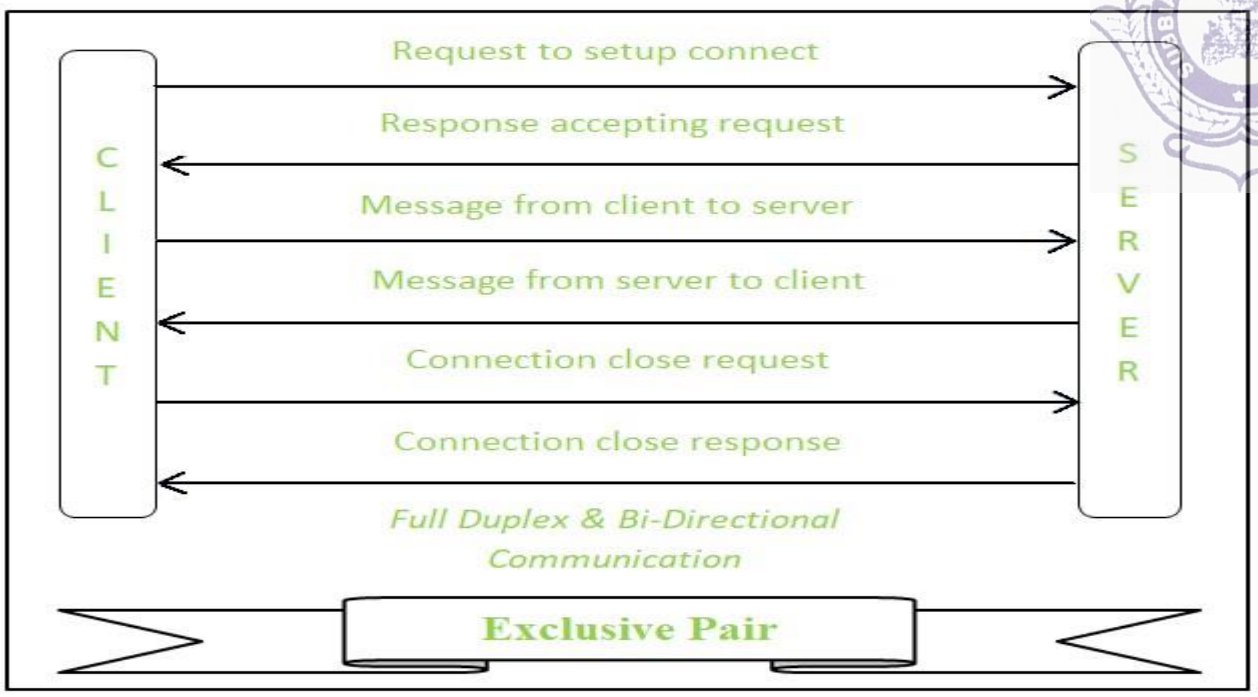
The push-pull model constitutes data publishers, data consumers, and data queues.

- **Publishers** and **Consumers** are not aware of each other.
- Publishers publish the message/data and push it into the queue. The consumers, present on the other side, pull the data out of the queue. Thus, the queue acts as the buffer for the message when the difference occurs in the rate of push or pull of data on the side of a publisher and consumer.
- **Queues** help in decoupling the messaging between the producer and consumer. Queues also act as a buffer which helps in situations where there is a mismatch between the rate at which the producers push the data and consumers pull the data.



4. Exclusive Pair

- **Exclusive Pair** is the bi-directional model, including full-duplex communication among client and server. The connection is constant and remains open till the client sends a request to close the connection.
- The **Server** has the record of all the connections which has been opened.
- This is a state-full connection model and the server is aware of all open connections.
- WebSocket based communication API is fully based on this model.



IoT Communication APIs

Generally we used Two APIs For IoT Communication. These IoT Communication APIs are:

- REST Based Communication APIs
- Web Socket Based Communication APIs

Web service can either be implemented using REST principles or using Web Socket Protocol –

1. REST Based Communication API :

REpresentational State Transfer (REST) is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred. REST APIs follow the request-response communication model. The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system.

Advantages of REST API:

- Simplicity: REST APIs are relatively simple to design and implement, making them a popular choice for building APIs for web applications.

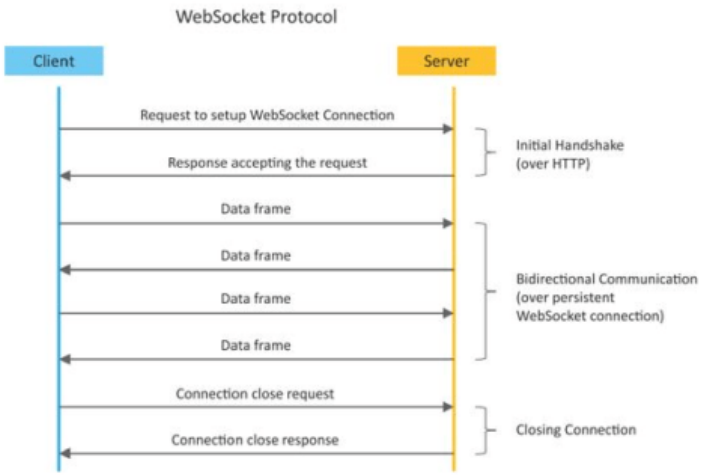
- Flexibility: REST APIs can be used to support a wide range of applications and services, from simple web applications to complex enterprise systems.
- Caching: REST APIs can leverage caching to improve performance and reduce server load.
- Stateless: REST APIs are stateless, meaning that each request is processed independently of any previous requests, making them easy to scale and distribute.

Disadvantages of REST API:

- Limited real-time support: REST APIs do not support real-time communication between the server and client, making them less suitable for applications that require real-time updates.
- Performance overhead: REST APIs require more overhead than WebSocket APIs, as each request and response must contain all the necessary information to complete the request.
- Complexity: REST APIs can be complex to design and implement for large, distributed systems.

2. Web Socket Based Communication APIs :

Web Socket APIs allow bi-directional, full-duplex communication between clients and servers. It follows the exclusive pair communication model. This Communication API does not require a new connection to be set up for each message to be sent between clients and servers. Once the connection is set up the messages can be sent and received continuously without any interruption. WebSocket APIs are suitable for IoT Applications with low latency or high throughput requirements.



Advantages of WebSocket API:

- Real-time communication: WebSocket APIs enable real-time communication between the server and client, making them ideal for applications that require real-time updates.
- Efficiency: WebSocket APIs are more efficient than REST APIs for real-time applications, as they use a persistent connection to enable bidirectional communication.
- Scalability: WebSocket APIs are highly scalable, as they can support thousands of connections per server.
- Reduced overhead: WebSocket APIs have lower overhead than REST APIs, as they use a single connection to transmit data.

Disadvantages of WebSocket API:

- **Complexity**: WebSocket APIs are more complex to design and implement than REST APIs, requiring additional programming skills and knowledge.
- **Security**: WebSocket APIs can be vulnerable to security threats if not properly secured.
- **Compatibility**: WebSocket APIs are not supported by all browsers, requiring fallback mechanisms for older browsers.

Similarities between REST API and WebSocket API:

- Both REST API and WebSocket API are used to build APIs for web applications.
- Both REST API and WebSocket API are standardized interfaces that enable communication between the server and client.
- Both REST API and WebSocket API can be customized to suit the specific needs of a particular application or system.
- Both REST API and WebSocket API can be secured using various authentication and encryption methods.

Difference between Rest API and Web Socket API :

S.NO.	REST API	WEB SOCKET API
1.	It is Stateless protocol. It will not store the data.	It is Stateful protocol. It will store the data.
2.	It is Uni-directional. Only either server or client will communicate.	It is Bi-directional. Messages can be received or sent by both server or client.
3.	It is Request-response model.	It is Full duplex model.
4.	HTTP request contains headers like head section, title section.	It is suitable for real-time applications. It does not have any overhead.
5.	New TCP connection will be set up for each HTTP request.	Only Single TCP connection.
6.	Both horizontal and vertical scaling (we can add many resources and number of users both horizontally and vertically).	Only vertical scaling (we can add resources only vertically).
7.	It depends upon the HTTP methods to retrieve the data..	It depends upon the IP address and port number to retrieve the data
8.	It is slower than web socket regarding the transmission of messages.	web socket transmits messages very fastly than REST API.
9.	It does not need memory or buffers to store the data.	It requires memory and buffers to store the data.

MQTT

Message Queuing Telemetry Transport (MQTT): The message query telemetry transport protocol is a communication-based protocol that is used for IoT devices. This protocol is based on the publish-subscribe methodology in which clients receive the information through a broker only to the subscribed topic. A broker is a mediator who categorizes messages into labels before being delivered.

Characteristics of the Protocol

- **Light-weight and reliable**: The MQTT message is compact, which can realize stable transmission on severely limited hardware equipment and network with low bandwidth and high delay.
- **Publish/subscribe mode**: Based on the publish/subscribe mode, the advantage of publishing and subscribing mode is that the publisher and subscriber are decoupled: Subscribers and publishers do not need to establish a direct connection or be online at the same time.
- **Created for the IoT**: It provides comprehensive IoT application features such as heartbeat mechanism, testament message, QoS quality level
- **Better ecosystem**: It covers all-language platform's clients and SDKs, and it has mature Broker server software, which can support massive Topic and ten-million-level device access and provide rich enterprise integration capabilities.

CoAP

Constrained Application Protocol (COAP): The constrained application protocol is a client server-based protocol. With this protocol, the COAP packet can be shared between different client nodes which are commanded by the COAP server. The server is responsible to share the information depending on its logic but has not acknowledged it. This is used with the applications which support the state transfer model.

Characteristics of the Protocol

CoAP refers to many design ideas of HTTP, and it also improves many design details and adds many practical functions according to the specific situation of limited resource-limited devices.

- It is based on message model

- Based on UDP Protocol, transport layer supports restricted devices
- It uses request/response model similar to HTTP request, and HTTP is text format, while CoAP is binary format, which is more compact than HTTP
- It supports two-way communication
- It has the characteristics of light-weight and low power consumption
- It supports reliable transmission, data re-transmission, and block transmission to ensure reliable arrival of data
- It supports IP multicast
- It supports observation mode
- It supports asynchronous communication

LoRaWAN

LoRaWAN refers to Long Rage Wide Area Network which is a wide area network protocol. It is an optimized low-power consumption protocol design to support large-scale public networks with millions of low-power devices. A single operator operates the LoRaWAN. The LoRaWAN network is a bi-directional communication for IoT application with low cost, mobility, and security..

An end device can connect to a network with LoRaWAN in two ways:

- Over-the-air Activation (OTAA): A device has to establish a network key and an application session key to connect with the network.
- Activation by Personalization (ABP): A device is hardcoded with keys needed to communicate with the network, making for a less secure but easier connection.

Properties of LoRaWAN protocol

- **Standard:** LoRaWAN
- **Frequency:** Various
- **Range:** 2-5km (urban environment), 15km (suburban environment)
- **Data Rates:** 0.3-50 kbps.

6LoWPAN

The 6LoWPAN protocol refers to IPv6 Low Power Personal Area Network which uses a lightweight IP-based communication to travel over low data rate networks. It has limited processing ability to transfer information wirelessly using an internet protocol. So, it is mainly used for home and building automation. The 6LoWPAN protocol operates only within the 2.4 GHz frequency range with 250 kbps transfer rate. It has a maximum length of 128-bit header packets.

6LowPAN Security Measure

Security is a major issue for 6LowPAN communication Protocol. There are several attacks issues at the security level of 6LoWPAN which aim is to direct destruction of the network. Since it is the combination of two systems, so, there is a possibility of attack from two sides that targets all the layer of the 6LoWPAN stack (Physical layer, Data link layer, Adaptation layer, Network layer, Transport layer, Application layer).

Basic Requirements of 6LoWPAN:

1. The device should be having sleep mode in order to support the battery saving.
2. Minimal memory requirement.
3. Routing overhead should be lowered.

Features of 6LoWPAN:

1. It is used with IEEE 802.15,.4 in the 2.4 GHz band.
2. Outdoor range: ~200 m (maximum)
3. Data rate: 200kbps (maximum)
4. Maximum number of nodes: ~100

Advantages of 6LoWPAN:

1. 6LoWPAN is a mesh network that is robust, scalable, and can heal on its own.
2. It delivers low-cost and secure communication in IoT devices.
3. It uses IPv6 protocol and so it can be directly routed to cloud platforms.
4. It offers one-to-many and many-to-one routing.

5. In the network, leaf nodes can be in sleep mode for a longer duration of time.

Disadvantages of 6LoWPAN:

1. It is comparatively less secure than Zigbee.
2. It has lesser immunity to interference than that Wi-Fi and Bluetooth.
3. Without the mesh topology, it supports a short range.

Applications of 6LoWPAN:

1. It is a wireless sensor network.
2. It is used in home-automation,
3. It is used in smart agricultural techniques, and industrial monitoring.
4. It is utilised to make IPv6 packet transmission on networks with constrained power and reliability resources possible.

Difference between COAP and MQTT protocols:

Basis of	COAP	MQTT
Abbreviation	Constrained Application Protocol	Message Queuing Telemetry Transport
Communication Type	It uses Request-Response model.	It uses Publish-Subscribe model
Messaging Mode	This uses both Asynchronous and Synchronous.	This uses only Asynchronous
Transport layer protocol	This mainly uses User Datagram protocol(UDP)	This mainly uses Transmission Control protocol(TCP)

Basis of	COAP	MQTT
Header size	It has 4 bytes sized header	It has 2 bytes sized header
RESTful based	Yes it uses REST principles	No it does not uses REST principles
Persistence support	It does not has such support	It supports and best used for live data communication
Message Labelling	It provides by adding labels to the messages.	It has no such feature.
Usability/Security	It is used in Utility area networks and has secured mechanism.	It is used in IoT applications and is secure
Effectiveness	Effectiveness in LNN is excellent.	Effectiveness in LNN is low.
Communication Model	Communication model is one-one.	Communication model is many-many.

Bluetooth

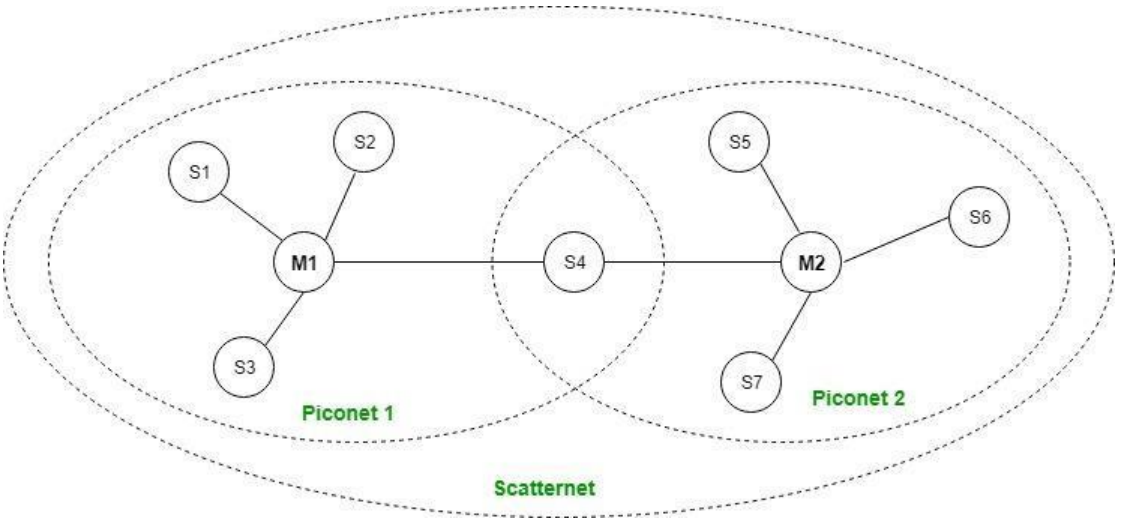
Bluetooth is universal for short-range wireless voice and data communication. It is a Wireless Personal Area Network (WPAN) technology and is used for exchanging data over smaller distances. This technology was invented by Ericson in 1994. It operates in the unlicensed, industrial, scientific, and medical (ISM) band from 2.4 GHz to 2.485 GHz. Maximum devices that can be connected at the same time are 7. Bluetooth ranges up to 10 meters. It provides data rates up to 1 Mbps or 3 Mbps depending upon the version. The

spreading technique that it uses is FHSS (Frequency-hopping spread spectrum). A Bluetooth network is called a **piconet** and a collection of interconnected piconets is called **scatternet**.

Bluetooth Architecture:

The architecture of Bluetooth defines two types of networks:

1. Piconet
2. Scatternet



Piconet:

Piconet is a type of Bluetooth network that contains **one primary node** called the master node and **seven active secondary nodes** called slave nodes. Thus, we can say that there is a total of 8 active nodes which are present at a distance of 10 meters. The communication between the primary and secondary nodes can be one-to-one or one-to-many. Possible communication is only between the master and slave; Slave-slave communication is not possible. It also has **255 parked nodes**, these are secondary nodes and cannot take participation in communication unless it gets converted to the active state.

Scatternet:

It is formed by using **various piconets**. A slave that is present in one piconet can act as master or we can say primary in another piconet. This kind of node can receive a message

from a master in one piconet and deliver the message to its slave in the other piconet where it is acting as a master. This type of node is referred to as a bridge node. A station cannot be mastered in two piconets.

Advantage:

- It is a low-cost and easy-to-use device.
- It can also penetrate through walls.
- It creates an Ad-hoc connection immediately without any wires.
- It is used for voice and data transfer.

Disadvantages:

- It can be hacked and hence, less secure.
- It has a slow data transfer rate: of 3 Mbps.
- It has a small range: 10 meters.
- Bluetooth communication does not support routing.
- The issues of handoffs have not been addressed.

Applications:

- It can be used in laptops, and in wireless PCs, printers.
- It can be used in wireless headsets, wireless PANs, and LANs.
- It can connect a digital camera wirelessly to a mobile phone.
- It can transfer data in terms of videos, songs, photographs, or files from one cell phone to another cell phone or computer.
- It is used in the sectors of Medical health care, sports and fitness, Military.

WIFI - Wireless Fidelity

Wifi is also known as **Wireless Fidelity.**

We are all familiar with Wi-Fi, which is available on our mobile phones, laptops, or wherever Wi-Fi is supported. Wi-Fi is **a wireless networking technology that permits to connect**

wirelessly to a network or to other computer or mobile device. A circular radio frequency range is used to transmit data in Wi-Fi.

Wireless Fidelity (Wi-Fi) is a generic term for the **wireless network in the communication norm**. Wifi operates like a local area network without the use of a wire or cables.

WLAN stands for **Wireless Local Area Network**. IEEE 802.11 is the rule for communication. WiFi uses the **Physical Data Link Layer (PDLL)** to operate.

Types or Kinds of Wifi

As mentioned earlier, Wi-Fi has numerous kinds or standards. Here, the names of the standards are defined.

- **Wi-Fi-1 (802.11b, launched in 1999**) - This version has link speed from 2Mb/s to 11 Mb/s over 2.4 GHz frequency band
- **Wi-Fi-2 (802.11a) launched in 1999**. After a month of releasing the previous version, 802.11a, was released, and it provides upto 54 Mb/s link speed over the 5 GHz band
- **Wi-Fi-3 (802.11g) was launched in 2003.** In this version, the speed was risen up to 54 to 108 Mb/s over 2.4 GHz
- **802.11i launched in 2004**. This is equivalentto**802.11g**, but only the security feature was enhanced in this version
- **802.11e launched in 2004**. This is also the same as **802.11g**; only Voice over Wireless LAN and multimedia streaming are included.
- **Wi-Fi-4 (802.11n) launched in 2009**. This version holds up both 2.4 GHz and 5 GHz radio frequencies, and it provides up to 72 to 600 Mb/s speed.
- **Wi-Fi-5 (802.11ac) launched in 2014**. It supports a speed of 1733 Mb/s in the 5 GHz band.

Advantages of WIFI

The advantages of Wi-Fi Tt include

- A **versatile network connection** and the absence of complicated wiring requirements for installation.
- Everywhere in the Wi-Fi range can access it.

- Independent users are not required to obtain regulatory approval.
- In addition, Wi-Fi Extenders make it possible to expand the network.
- It's easy and quick to set up.
- Only the SSID and password need to be configured.
- As part of its security measures, Wi-Fi networks encrypt radio signals using WPA encryption.
- It is also more affordable.
- Hotspots are another feature that it offers.
- Roaming is supported as well.

Wi-Fi Disadvantages

- Mobile phones, laptops, and other devices with batteries consume a lot of power when using Wi-Fi.
- Even when encryption is in place, security issues can still arise.
- Wi-Fi can be attacked and accessed in the same way that recognised devices become unidentified to the router.
- In comparison to a direct cable connection, the speed is slower.
- People can be harmed by it because it emits radiation like cell phones.
- Thunderstorms, for example, can interfere with Wi-Fi signals.
- Because it **lacks a firewall, unauthorised access** to Wi-Fi is possible.
- Since a router is required to access the internet via Wi-Fi, we can't access the internet if the power goes out.

Zigbee

What is Zigbee? – Zigbee is a **low power**, low data rate (250kbps) wireless protocol used primarily for Home automation and industrial control, building automation,sensor data collection etc

Zigbee devices have a range (1 hop) of 80 to 100m.

Devices can be split into:

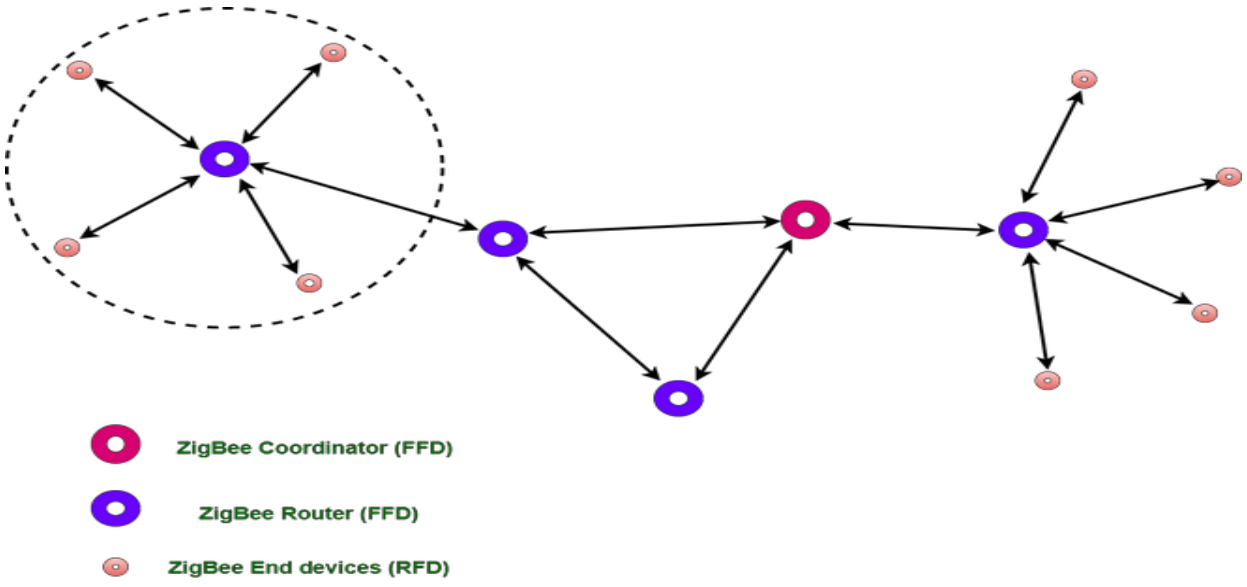
- **Full Function Device** (FFD)

Full Function Device (FFD)- Can communicates with all node types and can operate in one of three modes:

- **Zigbee Coordinator Device:** It communicates with routers. This device is used for connecting the devices.
- **Zigbee Router:** It is used for passing the data between devices.

Zigbee End Device: It is the device that is going to be controlled. There must be 1 coordinator on a Zigbee network.

Reduced Function Device (RFD): Can only talk to a single FFD. These are end nodes. Example a smart lock,switch ect



General Characteristics of Zigbee Standard:

- Low Power Consumption
- Low Data Rate (20- 250 kbps)
- Short-Range (75-100 meters)
- Network Join Time (~ 30 msec)
- Support Small and Large Networks (up to 65000 devices (Theory); 240 devices (Practically))
- Low Cost of Products and Cheap Implementation (Open Source Protocol)
- Extremely low-duty cycle.
- 3 frequency bands with 27 channels.

Operating Frequency

1. **Channel 0**: 868 MHz (Europe)
2. **Channel 1-10**: 915 MHz (the US and Australia)
3. **Channel 11-26**: 2.4 GHz (Across the World)

Features of Zigbee:

1. Stochastic addressing: A device is assigned a random address and announced. Mechanism for address conflict resolution. Parents node don't need to maintain assigned address table.

2. Link Management: Each node maintains quality of links to neighbors. Link quality is used as link cost in routing.

3. Frequency Agility: Nodes experience interference report to channel manager, which then selects another channel

4. Asymmetric Link: Each node has different transmit power and sensitivity. Paths may be asymmetric.

5. Power Management: Routers and Coordinators use main power. End Devices use batteries.

Advantages of Zigbee:

1. Designed for low power consumption.
2. Provides network security and application support services operating on the top of IEEE.
3. Zigbee makes possible completely networks homes where all devices are able to communicate and be
4. Use in smart home
5. Easy implementation
6. Adequate security features.
7. **Low cost**: Zigbee chips and modules are relatively inexpensive, which makes it a cost-effective solution for IoT applications.
8. **Mesh networking:** Zigbee uses a mesh network topology, which allows for devices to communicate with each other without the need for a central hub or router. This makes it ideal for use in smart home applications where devices need to communicate with each other and with a central control hub.
9. **Reliability:** Zigbee protocol is designed to be highly reliable, with robust mechanisms in place to ensure that data is delivered reliably even in adverse conditions.

Disadvantages of Zigbee :

1. **Limited range:** Zigbee has a relatively short range compared to other wireless communications protocols, which can make it less suitable for certain types of applications or for use in large buildings.
2. **Limited data rate:** Zigbee is designed for low-data-rate applications, which can make it less suitable for applications that require high-speed data transfer.
3. **Interoperability:** Zigbee is not as widely adopted as other IoT protocols, which can make it difficult to find devices that are compatible with each other.
4. **Security:** Zigbee's security features are not as robust as other IoT protocols, making it more vulnerable to hacking and other security threats.

Zigbee Applications:

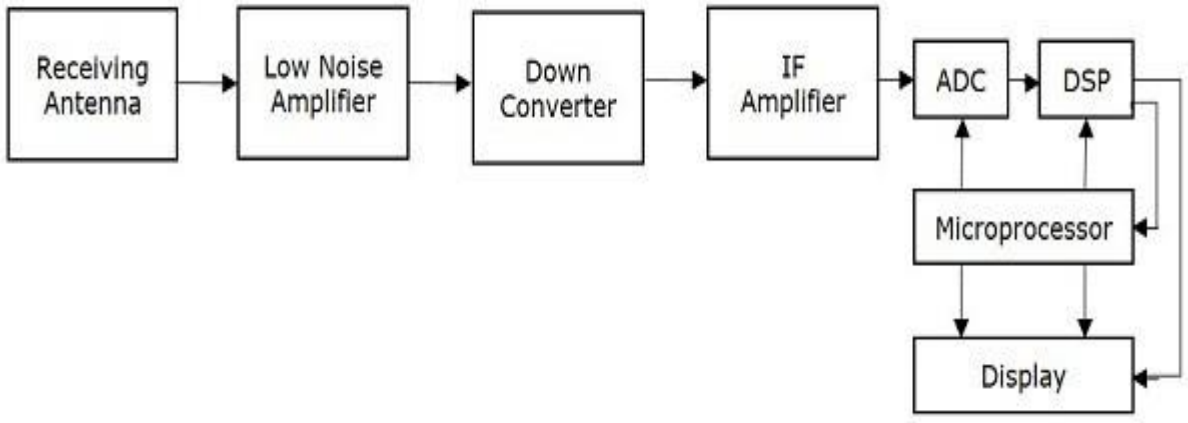
1. Home Automation
2. Medical Data Collection
3. Industrial Control Systems
4. meter reading system
5. light control system
6. Commercial
7. Government Markets Worldwide
8. Home Networking

GPS

Global Positioning System **(GPS)** is a navigation system based on satellite. It has created the revolution in navigation and position location. It is mainly used in positioning, navigation, monitoring and surveying applications.

GPS receiver basically consists of three components:

- An Antenna (tuned to the frequencies transmitted by the satellites).
- Receiver processor.
- Highly Stable Clock (Commonly a Crystal oscillator).



- **Receiving Antenna** receives the satellite signals. It is mainly, a circularly polarized antenna.
- **Low Noise Amplifier** (LNA) amplifies the weak received signal
- **Down converter** converts the frequency of received signal to an Intermediate Frequency (IF) signal.
- **IF Amplifier** amplifies the Intermediate Frequency (IF) signal.
- **ADC** performs the conversion of analog signal, which is obtained from IF amplifier to digital. Assume, the sampling & quantization blocks are also present in ADC (Analog to Digital Converter).
- **DSP** (Digital Signal Processor) generates the C/A code.
- **Microprocessor** performs the calculation of position and provides the timing signals in order to control the operation of other digital blocks. It sends the useful information to Display unit in order to display it on the screen.

Usage of GPS:

There are five most uses of the GPS.

- **Location:-** with the help of GPS we can find the exact position of the object.
- **Navigation:**– we can navigate one location to another with the help of GPS.GPS technology is also useful for Transportation Management and breathing of Ship at docks.
- **Tracking**: -with the help of GPS we can Monitor object movement like speed, distance, position.
- **Mapping:**– GPS also helps in creating maps of the World.
- **Timing**:- GPS also provides the estimated time for reaching destination measurement its depend on speed and object movement.

GSM

GSM stands for **Global System for Mobile Communication.** GSM is an open and digital cellular technology used for mobile communication. It uses 4 different frequency bands of 850 MHz, 900 MHz, 1800 MHz and 1900 MHz . It uses the combination of FDMA and TDMA.

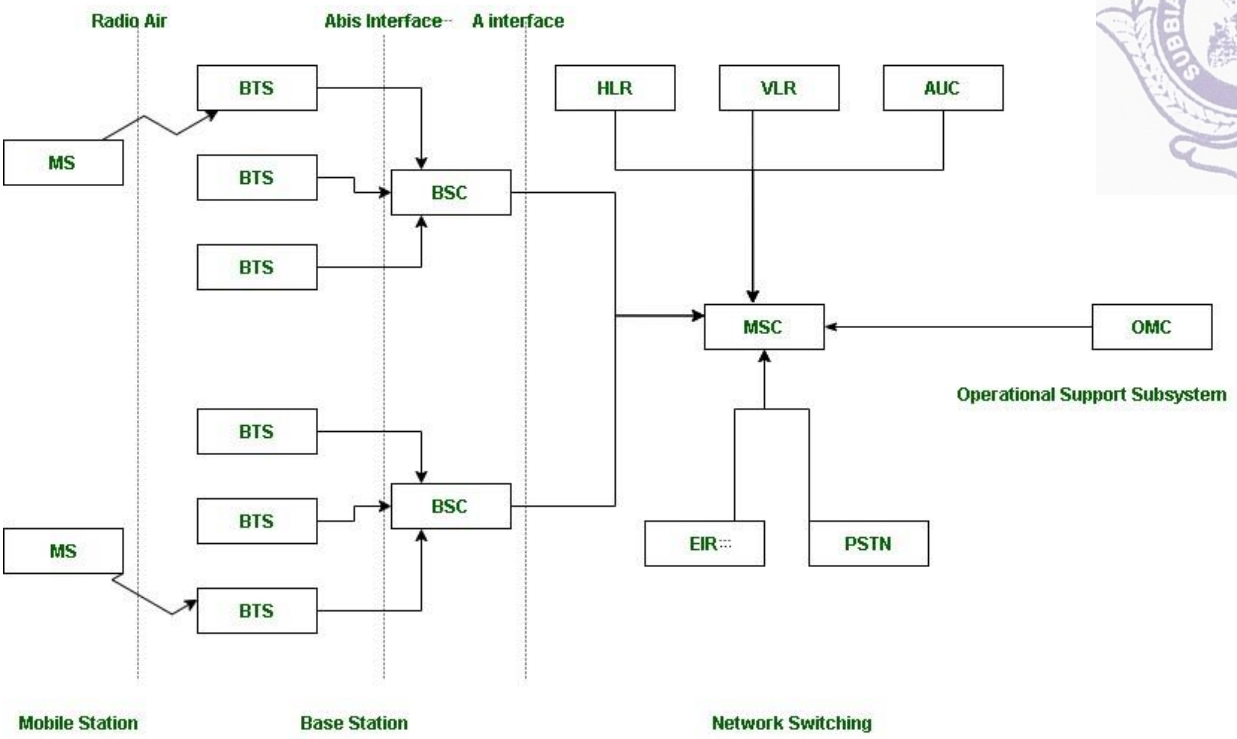
GSM is having 4 different sizes of cells are used in GSM :

1. Macro : In this size of cell, Base Station antenna is installed.
2. Micro : In this size of cell, antenna height is less than the average roof level.
3. Pico : Small cells' diameter of few meters.
4. Umbrella : It covers the shadowed (Fill the gaps between cells) regions.

Features of GSM are :

1. Supports international roaming
2. Clear voice clarity
3. Ability to support multiple handheld devices.
4. Spectral / frequency efficiency
5. Low powered handheld devices.
6. Ease of accessing network
7. International ISDN compatibility.
8. Low service cost.
9. New features and services.

1. **BSS :** BSS stands for Base Station Subsystem. BSS handles traffic and signaling between a mobile phone and the network switching subsystem. BSS having two components **BTS** and **BSC.**
2. **NSS :** NSS stands for Network and Switching Subsystem. NSS is the core network of GSM. That carried out call and mobility management functions for mobile phone present in network. NSS have different components like **VLR, HLR** and **EIR.**
3. **OSS :** OSS stands for Operating Subsystem. OSS is a functional entity which the network operator monitor and control the system. **OMC** is the part of OSS. Purpose of OSS is to offer the customer cost-effective support for all GSM related maintenance services.



Suppose there are 3 Mobile stations which are connected with the tower and that tower is connected to BTS through TRX, then further connected to BSC and MSC. Let's understand the functionality of different components.

1. MS : MS stands for Mobile System. MS comprises user equipment and software needed for communication with a mobile network. Mobile Station (MS) = Mobile Equipment(ME) + Subscriber Identity Module (SIM). Now, these mobile stations are connected to tower and that tower connected with BTS through TRX. TRX is a transceiver which comprises transmitter and receiver. Transceiver has two performance of sending and receiving.

2. BTS : BTS stands for Base Transceiver Station which facilitates wireless communication between user equipment and a network. Every tower has BTS.

3. BSC : BSC stands for Base Station Controller. BSC has multiple BTS. You can consider the BSC as a local exchange of your area which has multiple towers and multiple towers have BTS.

4. MSC : MSC stands for Mobile Switching Center. MSC is associated with communication switching functions such as call setup, call release and routing. Call tracing, call forwarding all functions are performed at the MSC level.

Services of GSM:

- Telephony services or teleservices
- Data services or bearer services
- Supplementary services

Advantages:

Compatibility: GSM is widely used around the world, so it is compatible with many different networks and devices.

Security: GSM offers enhanced security features such as authentication, encryption and confidentiality, which helps to protect the user's privacy and data.

Efficient use of bandwidth: GSM uses a time-division multiplexing (TDM) technique which enables many users to share the same frequency channel at different times, making it an efficient use of the available bandwidth.

Roaming: GSM allows users to roam internationally and use their mobile phones in other countries that use the same GSM standard.

Wide range of features: GSM supports a wide range of features, including call forwarding, call waiting, voicemail, conference calling, and more.

Disadvantages:

Limited coverage: GSM networks may have limited coverage in some remote areas, which can make it difficult for users to make calls or access the internet.

Network congestion: GSM networks may become congested during peak hours, which can lead to dropped calls or poor call quality.

Security vulnerabilities: Although GSM offers enhanced security features, it is still vulnerable to certain types of attacks, such as eavesdropping and spoofing.

Data transfer speed: GSM networks offer relatively slow data transfer speeds compared to newer technologies such as 3G and 4G.

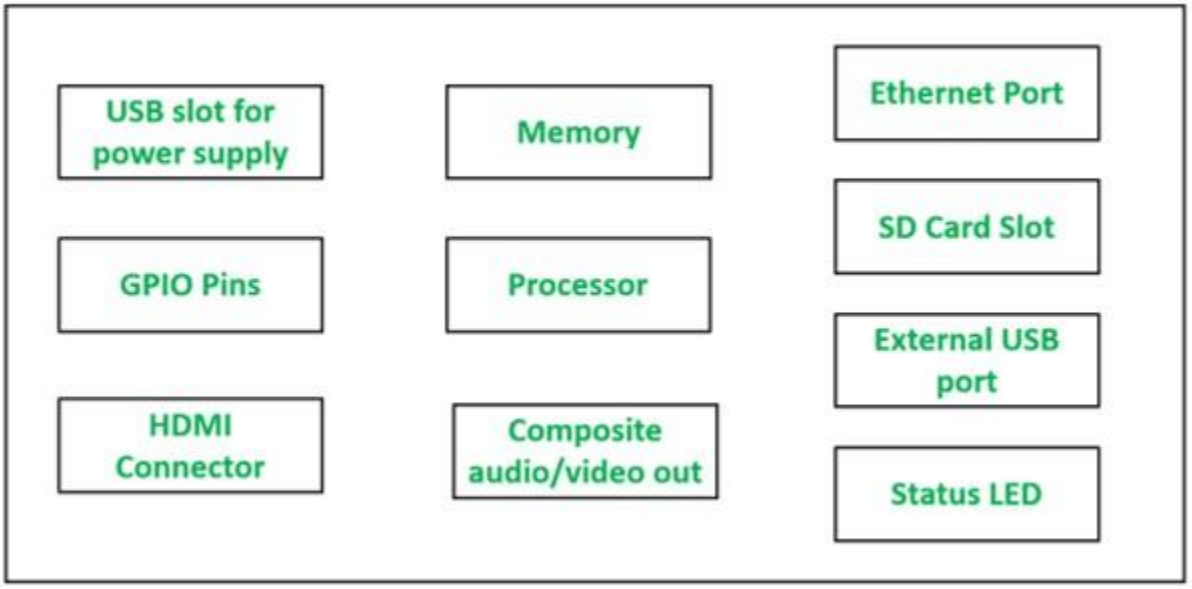
Limited capacity: GSM networks have a limited capacity for handling large volumes of data, which can be a disadvantage for users who require high-speed internet access or other data-intensive applications.

Single Board Computers -Raspberry-Pi

Raspberry Pi is developed by Raspberry Pi Foundation in the United Kingdom. The Raspberry Pi is a series of powerful, small single-board computers.

Raspberry Pi is launched in 2012 and there have been several iterations and variations released since then.

Various versions of Raspberry Pi have been out till date. All versions consist of a Broadcom system on a chip (SoC) with an integrated ARM-compatible CPU and on-chip graphics processing unit (GPU).

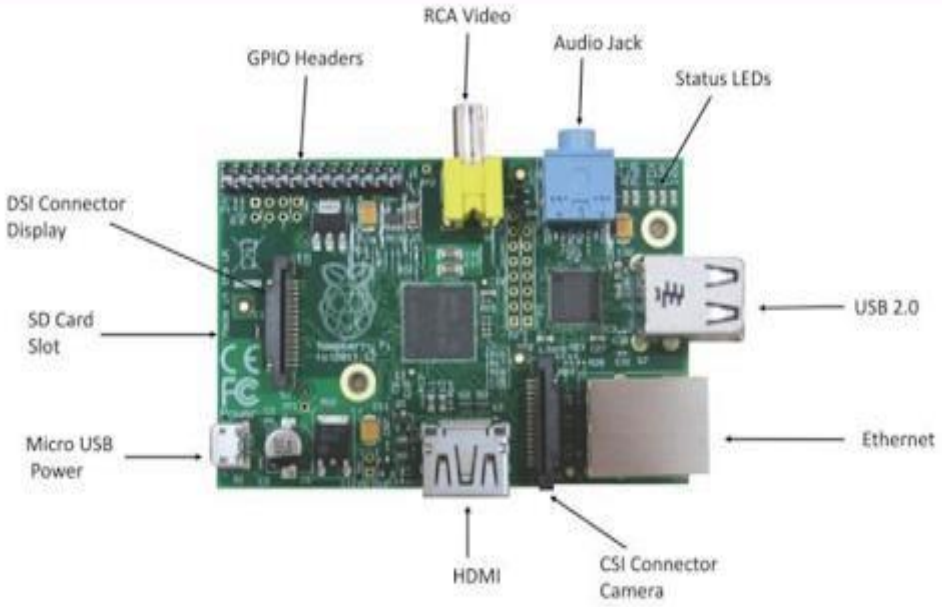


Block Diagram of Raspberry Pi

- A low-cost, **credit-card-sized** minicomputer**.**
- Plugs into a computer monitor or TV.
- Uses a standard keyboard and mouse.
- Use for explore computing as for programmings like Scratch and Python languages.
- Capable as a normal desktop computer to browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games."

- Allowing interface sensors and actuators through the general purpose IO pins
- Supports Python out of box

Structure of board



Raspberry Pi board uses various components and peripherals as follows:-

Processor and Ram

- Based on an ARM processor.
- The latest version of Raspberry Pi (model B, revision 2) comes with a 700 MHz low-power ARM 1176JZ-F processor And a 512 MB SD Ram.

USB ports

- Two USB 2.0 USB ports on Raspberry pi can provide a current of up to 100 mA.
- For connecting devices that draw a current of more than 100 mA, an external USB-powered hub is required.

Ethernet port

- Standard RJ45 Ethernet port.
- Connect an ethernet cable or USB Wi-Fi adaptor to provide internet connectivity.

HDMI outputs

- The HDMI port on Raspberry Pi provides both video and audio output.
- Connect the Raspberry Pi to a monitor by an HDMI cable.
- For monitors that have a DVI port but no HDMI port so use an HDMI to DVI adaptor or cable.

Composite video output with RCA jack

- Support both PAL and NTSC video output.
- RCA jack is used to connect old television that has an RCA input only.

Audio output

- 3.5 mm audio output jack is used for audio output to old television along with the RCA Jack for video.
- The audio quality is inferior to the HDMI output.

Display serial interface(DSI)

- Used to connect an LCD panel to Raspberry Pi.

Camera serial interface(CSI)

- Used to connect a camera module to Raspberry Pi.

Status LEDs

- Raspberry Pi has 5 status LED is which are:-
- ACT:- SD card access.
- PWR:- 3.3 V power is present.
- FDX:- Full duplex LAN connected.
- LNK:- link/ network activity.
- 100:- 100 Mbit LAN connected.

SD card slot

- Not have a built-in operating system and storage so plug in an SD card loaded with a Linux image to an SD card slot.

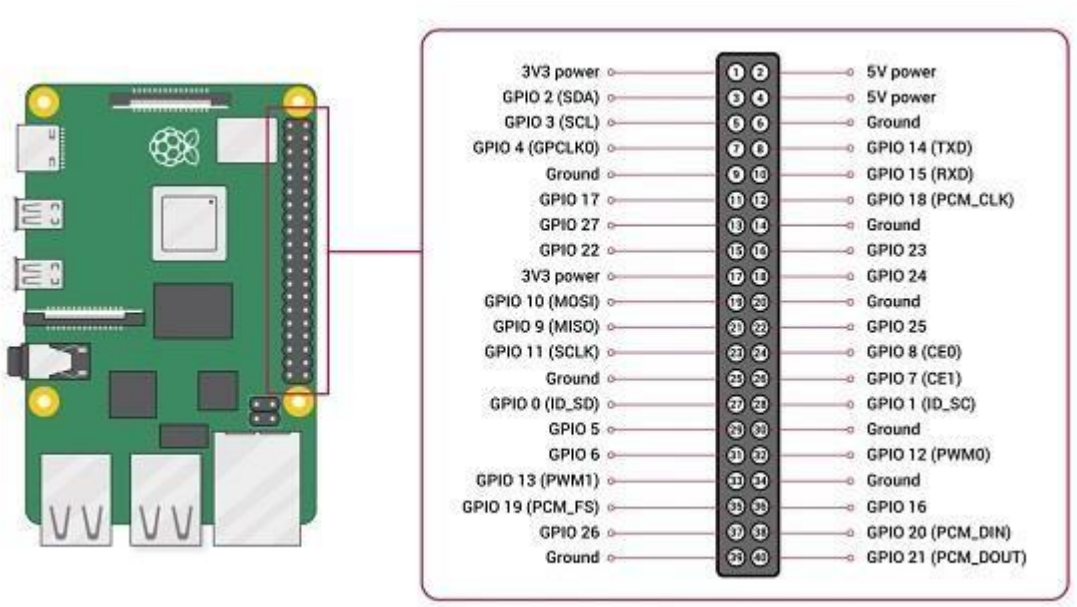
- Required at least an 8GB SD card for the setting up NOOBS software (new out-of-the-box software)

Power input

- A micro-USB connector for power input.

GPIO Pin

- A number of general-purpose Input / Output pins are used by Raspberry Pi.
- There are four types of pins on Raspberry Pi:- True GPIO pins, SPI interface pins, 12C interface pins, and Serial RX and TX pins.



Voltages

From the above diagram, we can see that there are two 5V pins and two 3V3 pins on the board. It also has several ground pins (0V). All these pins are unconfigurable.

Outputs

A GPIO pin can be designated as an output pin. The pin set as output pin can be set to 3V(high) or 0V(low).

Inputs

A GPIO pin can be designated as an input pin. The pin set as input pin can be read as 3V(high) or 0V(low). You can use internal pull-up or pull-down resistors.

above diagram, GPIO2 and GPIO3 pins have fixed pull-up resistors but for the other pins, you can configure it in software.

Alternative Functions

GPIO pins can be used with a variety of alternative functions. Among them, some are available on all pins and others on specific pins.

PWM: Pulse-width modulation

Software PWM are available on all the pins whereas Hardware PWM are available on GPIO12, GPIO13, GPIO18, and GPIO19.

SPI: Serial Peripheral Interface

The SPI are available on the following −

SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)
SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)

I2C: Inter-integrated Circuit

The I2C are available on the following −

Data: (GPIO2); Clock (GPIO3)
EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)

Serial

The serial function is available at the following −

TX(GPIO14)
RX(GPIO15)

Raspberry pi Connecting to the Cloud

Connecting a Raspberry Pi to the cloud allows you to access and manage your Raspberry Pi remotely, exchange data with cloud services, and perform various tasks such as data logging, remote control, and automation. Here's a general overview of how to connect a Raspberry Pi to the cloud:

1. **Select a Cloud Service Provider:** Choose a cloud service provider that fits your needs. Some popular options include AWS (Amazon Web Services), Microsoft Azure, Google Cloud Platform, and various IoT-focused platforms like AWS IoT, Azure IoT, or Google Cloud IoT. Each has its own set of features, pricing, and capabilities.
2. **Set Up Your Raspberry Pi:** Ensure your Raspberry Pi is set up correctly, connected to the internet, and running an appropriate operating system like Raspbian (now called Raspberry Pi OS).
3. **Install Necessary Software:** Depending on your chosen cloud platform, you may need to install specific libraries or SDKs on your Raspberry Pi. For example, if you're using AWS, you would install the AWS SDK for Python (Boto3). If you're using Azure, you might use the Azure IoT SDK or Azure Python SDK.
4. **Create Cloud Resources:** Create the necessary resources on your cloud platform. For example, if you're using AWS IoT, you'd create an IoT Thing, a policy, and certificates.
5. **Connect to the Cloud:** Write code on your Raspberry Pi to connect to your cloud platform of choice. This typically involves using the SDK you installed earlier and providing the necessary authentication credentials (e.g., API keys, certificates, or access tokens).
6. **Publish and Subscribe to Topics (MQTT):** Many IoT applications use the publish-subscribe model for communication. MQTT (Message Queuing Telemetry Transport) is a common protocol for this purpose. You can publish data from your Raspberry Pi to specific MQTT topics and subscribe to topics to receive commands or data from the cloud.
7. **Data Handling and Processing:** Once your Raspberry Pi is connected, you can send and receive data between the Raspberry Pi and the cloud. You may want to implement data processing, storage, or analytics depending on your project's requirements.
8. **Security Considerations:** Ensure proper security practices. Use secure connections (HTTPS, MQTT over TLS), secure your IoT devices and cloud resources, and regularly update your Raspberry Pi's software to patch any security vulnerabilities.

9. **Monitoring and Management:** Set up monitoring and management tools provided by your cloud platform to keep an eye on your Raspberry Pi's health and performance. This might include dashboards, alerts, and logging.
10. **Scale and Optimize:** As your project grows, you may need to scale your cloud resources and optimize your code and infrastructure for performance and cost-efficiency.

UNIT 5 - APPLICATIONS DEVELOPMENT

Complete Design of Embedded Systems – Development of IoT Applications – Home Automation –Smart Agriculture – Smart Cities – Smart Healthcare.

Embedded System

As its name suggests, Embedded means something that is attached to another thing. An embedded system can be thought of as a computer hardware system having software embedded in it. An embedded system can be an independent system or it can be a part of a large system. An embedded system is a microcontroller or microprocessor based system which is designed to perform a specific task. For example, a fire alarm is an embedded system; it will sense only smoke.

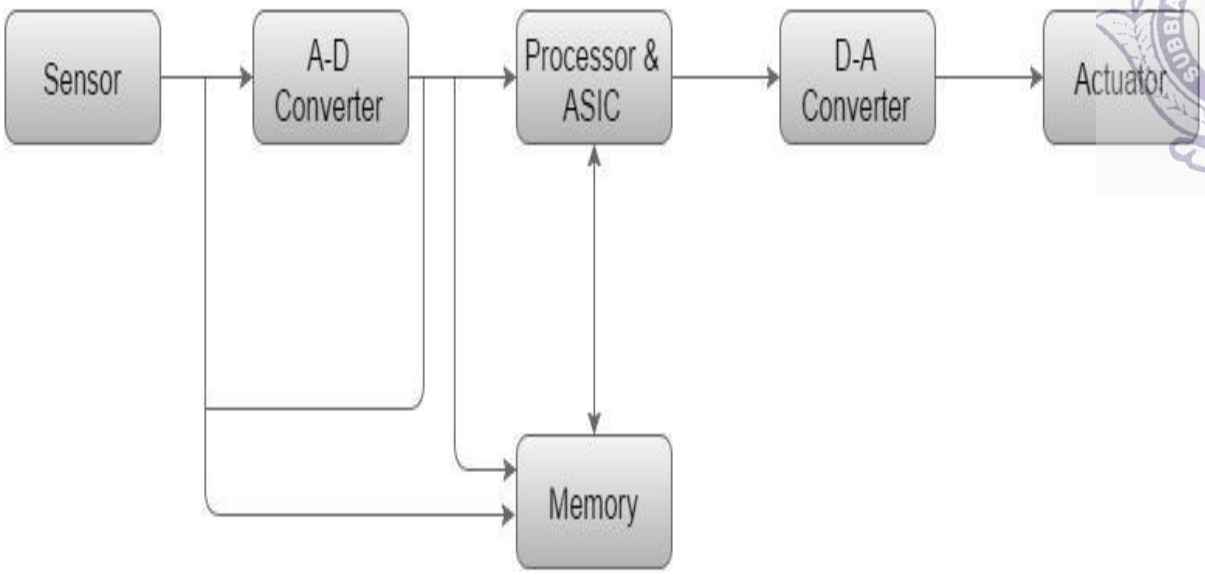
An embedded system has three components −

- It has hardware.
- It has application software.
- It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies. RTOS defines the way the system works. It sets the rules during the execution of application program. A small scale embedded system may not have RTOS.

So we can define an embedded system as a Microcontroller based, software driven, reliable, real-time control system.

Basic Structure of an Embedded System

The following illustration shows the basic structure of an embedded system −



- **Sensor** − It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.
- **A-D Converter** − An analog-to-digital converter converts the analog signal sent by the sensor into a digital signal.
- **Processor & ASICs** − Processors process the data to measure the output and store it to the memory.
- **D-A Converter** − A digital-to-analog converter converts the digital data fed by the processor to analog data
- **Actuator** − An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

Advantages

- Easily Customizable
- Low power consumption
- Low cost
- Enhanced performance

Disadvantages

- High development effort
- Larger time to market

Home Automation

Home Automation is a system that allows users to control various appliances of varying kinds and also makes controlling of home appliances easier and saves energy. Nowadays, home automation is used more and more. On the other hand, it provides increased comfort especially when everyone is busy with their work. Home automation installed in houses does not only increase comfort but also allows centralized control of heating, ventilation, air-condition, and lighting. Hence, they contribute to an overall cost reduction and also useful in energy saving which is certainly the main problem today.

In present years, wireless systems like Wi-Fi, Bluetooth have become more and more common in home networking. Also in home automation, the use of wireless technologies gives several advantages that could not be achieved using a wired network only.

Home Automation Components: At the most initial level, home automation systems are made up of three elements-

1. A smart device.
2. A hub.
3. A connected application.

While some other home automation systems work with just two elements which include a single device that works with the help of an app on mobile or a tablet or a system that includes a hands-free hub that controls home automation system while most of the systems work using all the above three components.

1. Smart Devices: These are the real powerhouse of any home automation system. These are the main parts that actually implement the whole system commands. Examples of the smart devices which can be added to any home automation to complete the whole system are as follows:

- **Access Control**
- **Security Devices:** This includes security cameras, smart locks.
- **Home Appliances:** Smart refrigerators, washing machines, dishwashers, and ovens already exist.
- **Smaller Appliances:** As automatic coffee pots and electric kettle have been also around for a while too
- **Climate Controls:** Climate control system with energy management systems
- **Smart Thermostats.**

- **Entertainment Pieces:** Entertainment includes smart TVs, wireless speakers, and film projectors
- **Health Care Devices:** Smart humidifiers and smart scales are two common examples of health care devices.
- **Lighting Controls:** They include dimmers, light bulbs, light strips, and switches, etc.

A high-speed internet plays an important role in smooth connectivity and also plays an important reliable performance between Wi-Fi-enabled devices.

2. Smart Hubs: The hub is the controlling center of the home automation system. It is the piece that connects your individual devices and helps them talk to one another.

3. Mobile Apps: The mobile application provides an interface between the user and the system. It gives you the ability to control or monitor your smart devices remotely. They can be easily downloaded with the help of a provided application on mobile and provide access control of the system, power controls, timer access, and many more things.

How Home Automation Works?

Home automation works with the help of a network of devices that are connected to the Internet through different communication systems like Wi-Fi, Bluetooth, ZigBee, and others. Through these devices can be managed remotely through controllers through an app. Many of these IoT devices have sensors that monitor changes in motion, temperature, and light so the user can gain information about the device's surroundings.

Three steps are followed in Home automation as follows:

1. **Monitoring:** This means keeping the control of the system using an app on a device remotely.
2. **Control:** This means that the system can be controlled remotely from anywhere through the app by the user.
3. **Automation:** Automation means making almost all devices automatic for making it a better system.

Applications: Some of the most common applications of home automation are as follows-

- Heating, ventilation, and air conditioning.
- Lighting control system.

- Occupancy-aware control system.
- Leak detection.
- Smoke sensors.
- Indoor positioning systems.
- Home automation for the elderly and disabled.
- Air quality control.
- Smart Kitchen.
- Connected Cooking.
- Voice control devices like Amazon Alexa or Google Home used to control home appliances or systems etc.

Advantages of Home Automation:

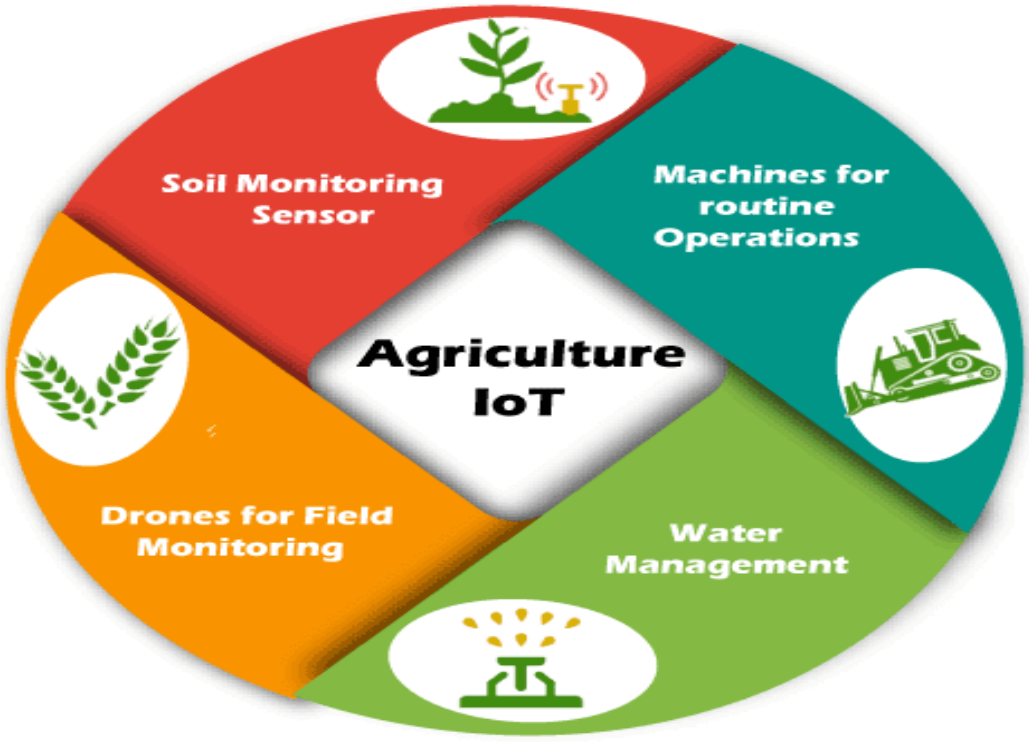
- **Energy Savings:** Self-automated light bulbs, fans, and switchboards save energy, cutting utility costs over time.
- **Home Safety:** Home automation provides the best technologies for home security. Consumers purchase these devices because they want to make their homes safer and more secure. Automatic lighting systems and motion sensors help people to enter doors and walk late at night.
- **User Convenient:** Because home automation performs role tasks automatically, end-users experience great convenience. For instance, you could use sensors indoors to turn on your smart lighting when you unlock the front door.
- **Better Control:** Consumers also choose smart home devices to better control functions within the home. With home automation technology, you can know easily what's happening inside your home at all times.
- **Comfortable Atmosphere:** All Connected devices around our home can also help to create a comfortable atmosphere—they provide intelligent and reliable lighting, sound, and temperature, which can all help to create a comfortable environment.
- **Provide Peace of Mind:** This system may help consumers to invest in home automation for peace of mind.
- **Remote Access:** Being able to control devices remotely means things like unlocking the door for a plant sitter without having to leave a key under the mat.

Disadvantages Of Home Automation:

- **Costs:** These are more expensive than their non-WiFi-connected counterparts.
- **Security Issues:** There are many security issues like the doorbell can be started ringing automatically etc.
- **New Technology:** Since IoT is a relatively new technology, you may run into some bugs, like devices having trouble connecting to the Internet or experiencing lag, depending on the device make and model.
- **Surveillance:** If privacy is a huge concern, then smart security is probably not for you, as users can live stream footage from the camera's respective app. Instead, you might want to opt for a local alarm system.

IOT in Agriculture

Agriculture is another important domain for IOT. IOT systems play an important role for crop and soil monitoring and give a proper solution accordingly. IOT leads to smart farming. Using IOT, farmers can minimize waste and increase productivity. The system allows the monitoring of fields with the help of sensors. Farmers can monitor the status of the area.



Challenges in the modern agriculture industry

The challenges faced by the farming industry and agriculture are listed as follows -

- Lack of workforce and manpower
- Environmental challenges and global warming
- Requirement of large manual intervention
- Lack of proper monitoring
- Challenges in analyzing the large scale unstructured data

There are various uses of IOT in agriculture that are discussed as follows -

IOT analytics in agriculture

The data from smart sensors can be further analyzed for automated decision-making and predictive analysis. Machine learning and predictive analysis will be helpful for farmers to cope up with the weather conditions such as drought, flood, etc.

Drone-based uses

Drones are also useful in smart farming. On one side, drones are useful to monitor the soil, air, moisture quality, and on another side, they can also be used for physical activities such as prevention of physical breakouts in farms, automated spraying of fertilizers, and many more. Although there are some limitations of using a drone, but it is useful to reduce the manual workforce.

Real-time crop monitoring

Motion detectors, light detectors, smart-motion sensing sensors, smart sensors are useful to provide real-time data to farmers of their farms. It will be helpful in the monitoring of the quality of their products.

Smart Irrigation system

It is one of the parts of smart agriculture using IOT. In it, IOT checks the water lanes created by the farmer or the moisture level in the environment.

Infrastructure requirements

There are some infrastructure requirements for adopting smart farming in IOT. Some of the requirements are listed as follows -

- Hardware maintenance cost
- Continuous connectivity to the internet
- Required high investments in drones, sensors
- The requirement to hire highly trained staff for management and to operate
- Requirement of power connectivity to operate and charge the robots and drones

IoT in Smart Cities

1. Water Level Checking

The water supply is one of the most significant perspectives for legislatures. With intelligent sensors, the water levels can be checked progressively.

These sensors can send triggers and alarms to key chiefs for low or high water levels. The spillages and water dispersion can be combined using IoT sensors and ICT frameworks.

All regions with a plentiful water supply can be set apart on the guide; correspondingly, the guides can feature regions with water spillage or deficiency.

A complete outline of the water supply with GPS directions can be given to water specialists with IoT frameworks.

2. Health Cards

Clinics and medical services frameworks are significant marks of administration. The smart city requires a state−of−the−art medical services framework that can follow quantifiable advancement concerning residents' well−being.

A shrewd card−based framework can be utilized by people that might be utilized in all administration and approved clinics.

This card will have the verifiable subtleties of the medicines and so on for people. The robust medical care framework will empower the public authority to look at the clinics and their administrations to residents.

The smart card empowers the framework to work with simple information assortment. The cloud−based framework can give essential knowledge to Medical services experts for a further progressive organization.

3. Waste & Garbage Management

The waste and trash the executive's exercises can be improved with intelligent sensors and IoT Frameworks.

The trash containers can utilize intelligent sensors to demonstrate when they should be discharged. This diminishes the times that vehicles are expected to gather the trash from the receptacles and evades what is going on of waste flood.

Metropolitan organizations can involve shrewd receptacles and IoT frameworks for trash assortment.

4. Transport Systems

The transportation framework for the residents can be improved with IoT−empowered frameworks. The armadas can be overseen and followed utilizing GPS beacons.

Legislatures can finish armadas' organization, planning, ongoing situating, support, and free time for executives with IoT frameworks.

The residents can likewise benefit from transportation administrations with a card−based framework for tickets and so on.

5. Smart Traffic Management

Traffic is one of the significant problem areas for residents. With IoT sensors, traffic can be controlled better.

The sensors are associated with traffic lights and send data to an incorporated server. The approaching vehicles are followed utilizing these sensors.

When the quantity of vehicles arrives at a limit, signals are shipped off to the drivers to redirect. These signs are shown with electronic showcase sheets.

Constant traffic cautions and GIS planning of the streets can further develop gridlocks and blockage during top hours.

6. Infrastructure Assets Management

The brilliant city requires advanced usage of framework resources. The plants, apparatus, and gear are labeled and observed with the brought−together resource of the executive's framework.

The continuous undertaking stock for different advancement works can be followed utilizing the brought−together framework.

The situation with framework resources, their usage, upkeep, and the complete lifecycle of the board should be possible with the brought−together IoT framework.

7. Surveillance Systems

IP cameras and reconnaissance frameworks can assist the public authority with controlling crime percentages in a city.

The IP cameras can be utilized for surveying and monitoring essential foundations. These cameras can be associated with unified frameworks with reinforcements for verifiable information.

A versatile reconnaissance framework can be set up with IoT video arrangements safeguarding individuals, spots, and resources.

8. Pollution Control With Sensors

Urbanization has prompted an uncommon expansion in contamination levels. The rising contamination levels are causing medical problems for residents.

With IoT–empowered sensors, contamination can be estimated progressively. The contamination sensors send data to an incorporated server.

The public authority can make a move given the contamination levels; e.g., they can establish trees in a specific area.

The plant life and contamination levels can likewise be portrayed online with google maps for executives.

9. Smart Energy Management

One of the critical difficulties for state–run administrations is to decrease energy utilization and introduce a proficient appropriation framework set up.

Brilliant framework arrangements, electronic meters, and intelligent lighting frameworks are a portion of the components that are utilized by legislatures to oversee energy effectively.

The power dispersion guides can show on going energy utilization levels, spillages, and upkeep plans. The IoT–empowered arrangements can improve the energy of the board for urban areas.

10. E-Services

This can be overseen through biometric confirmation or smart cards. Residents can benefit from all taxpayer–driven organizations through this card.

The public authority can collect data through these cards for proactive preparation and the executives.

All taxpayer–supported organizations can be incorporated through the e–administrations gateway.

The residents can benefit from these offices for paying their water and power bills, local charges, medical clinic check–ups, etc. Coordinated information additionally helps in strategy–making and organization.

Internet of Things (IoT) in Healthcare

IoT technology brings numerous applications in healthcare, from remote monitoring to smart sensors to medical device integration. It keeps the patients safe and healthy as well as improves the physician delivers care towards the patients.

Healthcare devices collect diverse data from a large set of real-world cases that increases the accuracy and the size of medical data.

Factor affecting IoT Healthcare Application

There are various factors that affect the IoT healthcare application. Some of them are mention below:

- **Continuous Research:** It requires continuous research in every field (smart devices, fast communication channel, etc.) of healthcare to provide a fast and better facility for patients.
- **Smart Devices:** Need to use the smart device in the healthcare system. IoT opens the potential of current technology and leads us toward new and better medical device solutions.
- **Better Care:** Using IoT technology, healthcare professionals get the enormous data of the patient, analysis the data and facilitate better care to the patient.
- **Medical Information Distribution:** IoT technology makes a transparency of information and distributes the accurate and current information to patients. This leads the fewer accidents from miscommunication, better preventive care, and improved patient satisfaction.

Simple Healthcare System Architecture

The application of the Internet of Things (IoT) in healthcare transforms it into more smart, fast and more accurate. There is different IoT architecture in healthcare that brings start health care system.

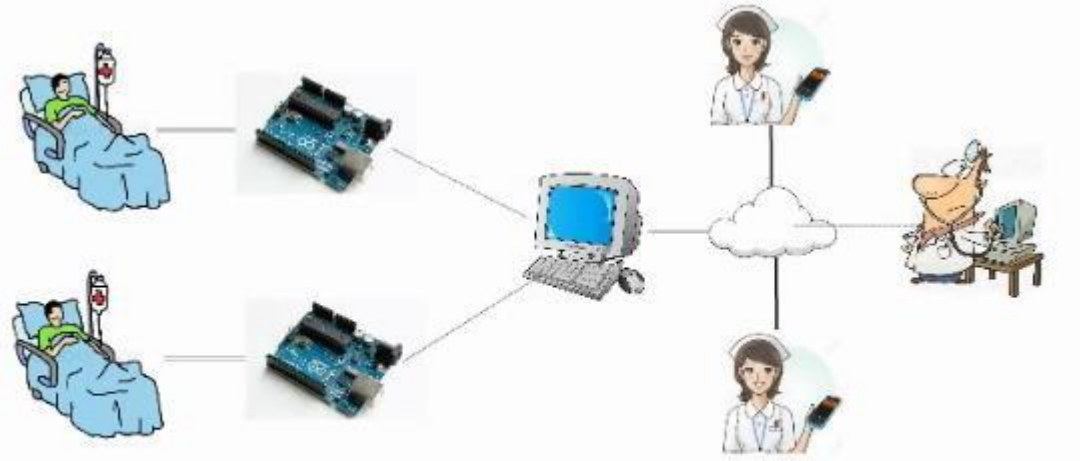
Product Infrastructure: IoT product infrastructure such as hardware/software component read the sensors signals and display them to a dedicated device.

Sensors: IoT in healthcare has different sensors devices such as pulse-oximeter, electrocardiogram, thermometer, fluid level sensor, sphygmomanometer (blood pressure) that read the current patient situation (data).

Connectivity: IoT system provides better connectivity (using Bluetooth, WiFi, etc.) of devices or sensors from microcontroller to server and vice-versa to read data.

Analytics: Healthcare system analyzes the data from sensors and correlates to get healthy parameters of the patient and on the basis of their analyze data they can upgrade the patient health.

Application Platform: IoT system access information to healthcare professionals on their monitor device for all patients with all details.



IoT in Healthcare – a concept

IoT challenges in Healthcare

- Data security & privacy
- Integration: multiple devices & protocols
- Data overload & accuracy
- Cost