ET8491 - Embedded systems and IOT Design.

Unit - I 8051 Microcontroller

DAY 1

* Microcontroller for an Embedded system.

To make a complete microcomputer system, only microprocessor is not sufficient. It is necessary to add other peripherals such as Read only memory (ROM), read / write memory (RAM), decoders, drivers, number of input / output devices to make a complete microcomputer system. In addition, special purpose devices, such as interrupt controller, programmable timers, programmable I/O devices, DMA controllers may be added to improve the capability and performance and flexibility of a microcomputer system.

The microcontroller encapsulates all the features that are found in microprocessor. However, it has also added features to make a complete microcomputer system on its own. The microcontroller has built-in ROM, RAM, parallel I/O, serial I/O, counters and a clock circuit.

Advantages:

* Built-in peripherals have smaller access times hence speed is more.
* Hardware reduces due to single chip microcomputer system.
* Less Hardware, reduces PCB size and increases reliability of the system.

* 8051

Features of 8051 microcontroller :-

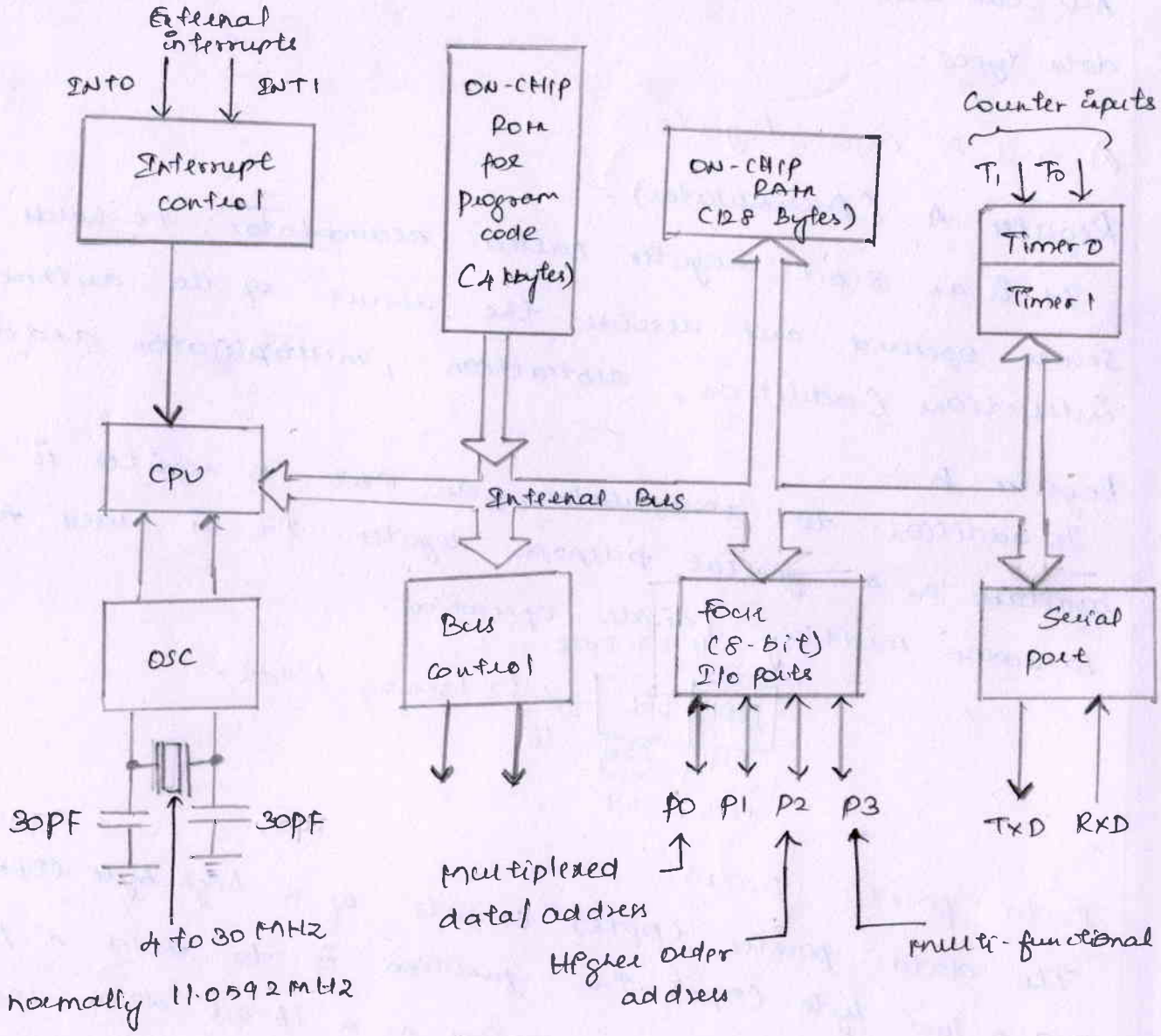
The 8051 is an 8-bit microcontroller designed by Intel. It was optimized for 8-bit math and single bit Boolean operations. Its family includes 8031, 8051, 8052 and 8751 microcontroller.

The features are as follows :-

1) 4096 bytes on-chip program memory.
2) 128 bytes on-chip data memory.
3) Four register banks.
4) 128 user-defined software flags.
5) 64 Kilobytes each program and external RAM addressability.
6) One microsecond instruction cycle with 12MHz crystal.
7) 32 bidirectional I/O lines organized as four 8-bit ports
8) Multiple mode, high-speed programmable serial port.
9) Two multiple mode, 16-bit timers/counters.
10) Two-level prioritized interrupt structure.
11) Full depth stack for subroutine return linkage and data storage.
12) Direct byte and bit addressability.
13) Binary or decimal arithmetic.
14) Signed-overflow detection and parity computation.
15) Hardware multiple and divide in 4 usec.
16) Integrated boolean processor for control applications
17) Upwardly compatible with existing 8051 software.

* Architecture of 8051:-

Day. 2



Block diagram of 8051.

Central processing Unit:-

The CPU of 8051 consists of eight-bit arithmetic and logic unit with associated registers like A, B, PSW, SP, the sixteen bit program counter and "Data pointer" (DPTR) registers. Along with these registers it has a set of special function registers.

The unique feature of 8051 architecture is that the ALU can also manipulate one bit as well as eight-bit data types.

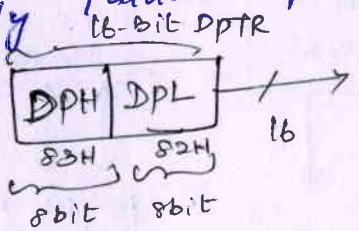
A and B CPU Registers :-

Register A (Accumulator) :-

It is an 8-bit register called accumulator. It holds a source operand and receives the result of the arithmetic instructions (addition, subtraction, multiplication and division)

Register B :-

In addition to accumulator, an 8 bit B register is available as a general purpose register. It is used for hardware multiply / divide operation.



Data pointer (DPTR)

The data pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its function is to hold a 16-bit address. It may be manipulated as a 16-bit data register or as two independent 8-bit registers. It serves as a base register in indirect jumps, lookup table instructions and external data transfer. The DPTR does not have a single internal address; DPH (83H) and DPL (82H) have seperate internal addresses.

Program counter :-

The 8051 has a 16-bit program counter. It is used to hold the address of memory location from which the next instruction is to be fetched.

8051 flag bits and PSW Register:-

The program status word (PSW) is also known as flag register.

B7	B6	B5	B4	B3	B2	B1	B0
CY	AC	F0	RS1	RS0	OV	-	P

The 8051 consists of following flags:-

* CY - carry flag:- This flag is set if there is an overflow out of bit 7. The carry flag also serves as a borrow flag for subtraction. In b
* AC - Auxiliary carry flag:- This flag is set if there is an overflow out of bit 3 ie, carry from lower nibble to higher nibble (D3 bit to D4 bit).
* F0 - Available for user for general purpose.
* RS1 - RS0 (Register Bank select): They select the working register bank as follows:-

RS1	RS0	Bank selection	
0	0	00H - 07H	Bank 0
0	1	08H - 0FH	Bank 1
1	0	10H - 17H	Bank 2
1	1	18H - 1FH	Bank 3

* OV-overflow flag:-
This flag is set whenever the result of a signed number operation is too large, causing the high order bit to overflow into the sign bit.

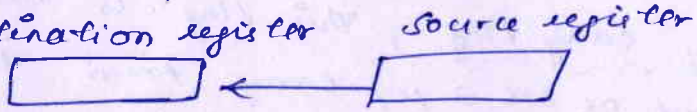
* P-parity flag:-
parity is defined by the number of ones present in the accumulator, P=0, if number of ones are even and P=1, if number of ones are odd.

* Addressing modes :-

The way, using which the data sources or destination addresses are specified in the instruction mnemonic for moving the data, is called addressing mode.

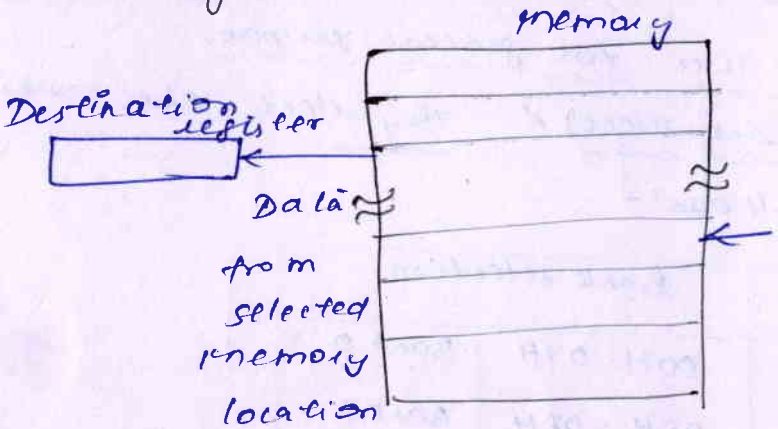
* Register Addressing :-

The 8051 can access eight "working registers" (R0-R7). Three bit code within the instruction selects one of the eight registers from the selected register bank. The programmer can select a register bank by modifying bits 4 and 3 in the PSW.



Example :-
Step 1 :- MOV PSW, #00001000B; select register Bank 2

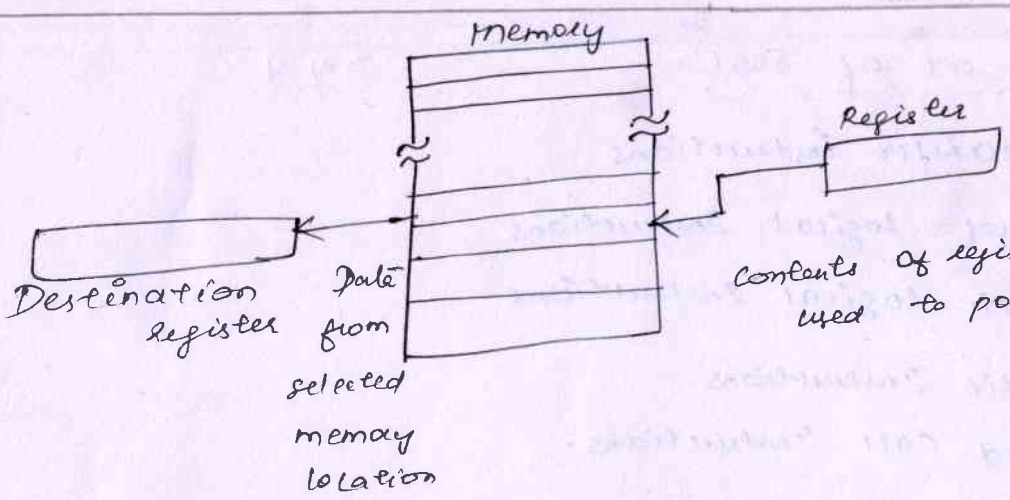
* Direct Byte Addressing :-



Direct addressing can access any on-chip variable or hardware register i.e. on-chip RAM and special function register. The most significant bit of the address decides whether it is a location within on-chip RAM (MSB=0) or it is special function register (MSB=1)

* Register Indirect Addressing :-

In this addressing mode R0 and R1 of each register bank can be used as an index or pointer register. R0 and R1 point to the contents in the RAM. The instruction with indirect addressing uses the '@' sign.



Ro and R1 are the only registers that can be used for pointers in register indirect addressing mode.

* Immediate addressing :-

In this addressing mode source operand is a constant rather than a variable. So the constant can be incorporated in the instruction. Sign "#" indicates it is a immediate addressing mode.

* Register specific :-

Inherent in the instruction, these refer to a specific register such as accumulator or DPTR.

* Index :-

Only program memory can be accessed in the index addressing. Either the DPTR or PC can be used as an index register.

* Stack Addressing mode :-

It is subtype of direct addressing mode in which stack instructions (PUSH and POP) are used. Instruction such as "PUSH A" is invalid. PUSH 0E0H is valid instruction.

Instruction set of 8051:-

Day 4

* Data transfer instructions
* Byte level logical instructions
* Bit level logical instructions
* Arithmetic instructions.
* Jump and call instructions.

* Data transfer Instructions:-

An immediate, direct, register and indirect addressing modes are used in different move instructions.

MOV <dest>, <src>

* Instructions to Access External Data memory.

MOVX A, @Ri

* Instructions to Access External ROM / program memory.

MOVC A, @A+DPTR.

* Data transfer with STACK (PUSH and POP) Instruction:-

PUSH direct : push onto stack

POP direct : pop from stack.

* Data Exchange Instruction.

XCH A, Rn.

* Byte level Logical Instruction.

ANL <dest-byte>, <src-byte>.

* Arithmetic Instructions

Increment, decrement, addition, subtraction, multiplication, division and decimal operations.

* Bit level logical Instructions:-

ANL C, <src-bit>

* Rotate and swap instructions.

RL A :- Rotate Accumulator left.

* Jump and call Instructions :-

AJUMP addr11 → AJmp transfers program execution to the indicated address. since address is 11-bit the destination must therefore be either the same 2K block of program memory as the first byte of the instruction following AJmp. No flags are affected.

* CALL and subroutines :-
There are two subroutine - call instructions. LCALL (Long call) and ACALL (Absolute call).
Each increments the pc to the first byte of the following instruction, then pushes it onto the stack (low byte first).

---

* Data and program memory :- Day 5

Memory Addressing

Program memory

↳ Stores program to be executed

↳ Implemented by EPROM

↳ further divided into on-chip (Internal) - 4KB and external - 64KB

Data memory

↳ stores, intermediate results, variables, constants.

↳ Implemented by RAM

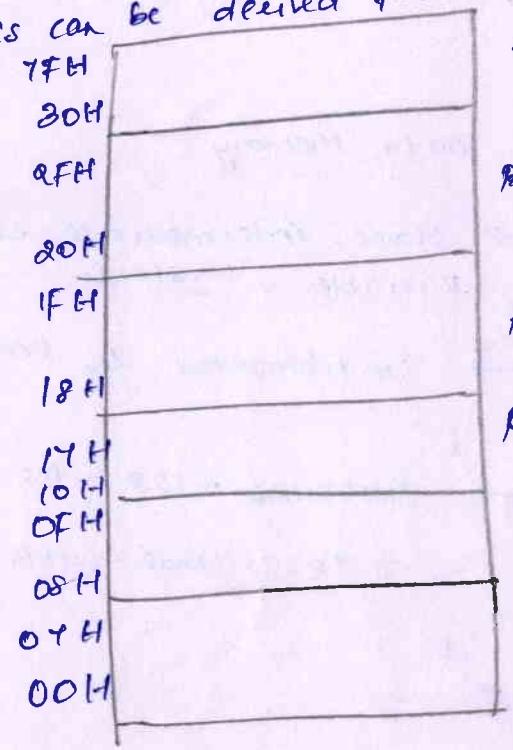
↳ INTERNAL - 128 bytes of RAM + SFR External - 64KB

Program Memory - ROM

* ROM (Read only memory)
  * 8051 can address 4K bytes on chip memory
  * It can address 64KB external memory
  * Memory map of internal and external program overlaps.
  * The internal & external ROM distinguished by PSEN signal.
  * ROM less version of 8051 - PSEN used to access external memory.

Data memory :-

* External RAM
  * 8051 supports 64KB external data memory - range - 0000 to FFFF.
  * Accessed by DPTR
  * 8051 generates RD, WR during external access.
  * CS can be derived from address lines.



PSW :-

CY	AC	F0	RS1	RS0	OV	-	P

CY Carry flag

AC Auxillary carry flag

F0 Flag 0.

RS1 Register Bank address bit 1

RS0 Register Bank address bit 0

OV overflow flag

- User defined flag

P parity flag.

DAY. 6

* Stacks :-

The stack is a section of RAM used by the CPU to store information temporarily. This information could be data or an address. The CPU needs this storage area since there are only a limited number of registers.

How stacks are accessed in the 8051:-

If the stack is a section of RAM, there must be registers inside the CPU to point to it. The register used to access the stack is called the SP (stack pointer) register. The stack pointer in the 8051 is only 8 bits wide, which means that it can take values of 00 to FFH. When the 8051 is powered up, the SP register contains value 07. this means that RAM location 08 is the

first location used for the stack by the 8051. The storing of a cpu register in the stack is called a PUSH, and pulling the contents of the stack back into a cpu register is called a pop. In other words, a register is pushed onto the stack to save it and popped off the stack to retrieve it. The job of SP is very critical when push and pop actions are performed. To push and pop the following steps are involved.

Pushing onto the stack:-

In the 8051 the stack pointer (SP) points to the last used location of the stack. As we push data onto the stack, the stack pointer (SP) is incremented by one. Notice that this is different from many microprocessors, notably X86 processors in which the SP is decremented when data is pushed onto the stack.

Popping from the stack:-

With every pop, the top byte of the stack is copied to the register specified by the instruction and stack pointer is decremented by once.

The stack is a section of RAM used by the cpu to store the information temporarily.

Information could be data or an address.

The register used to access the stack is called stack point register.

DAY-7

* Interrupts :-

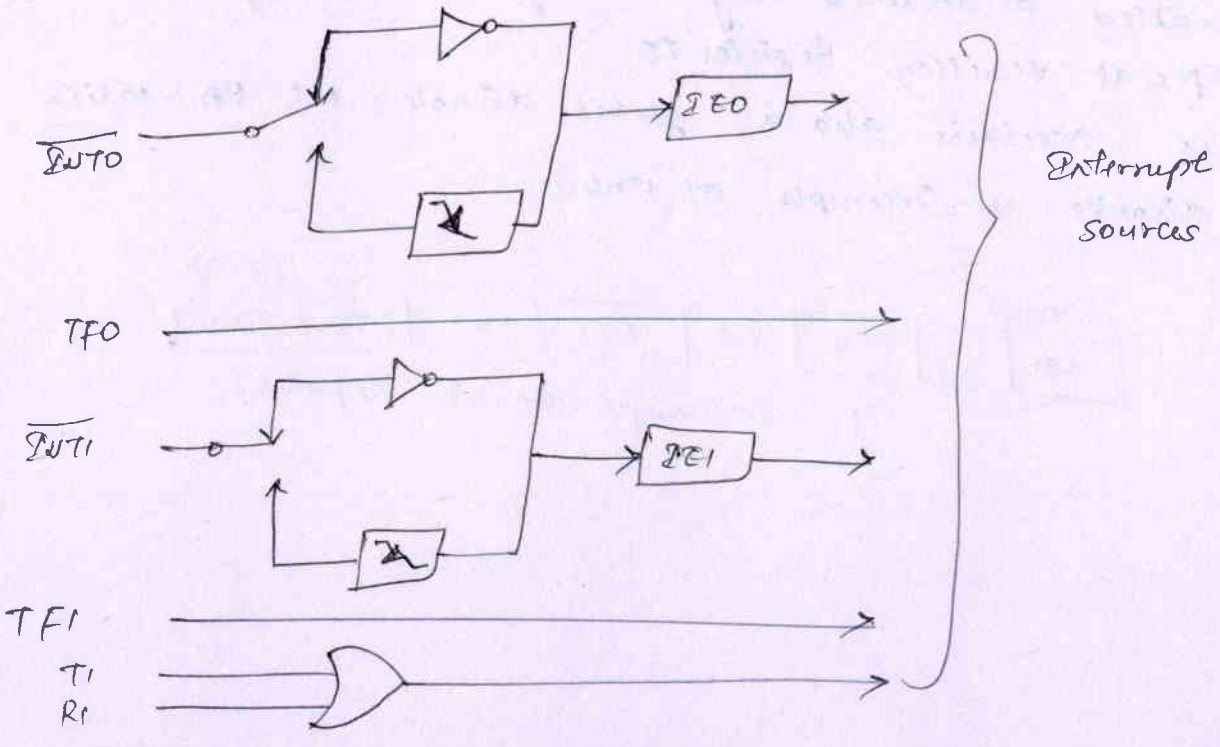
The 8051 provides 5 interrupt sources. The 8052 provides 6.

The external interrupts $\overline{INT0}$ and $\overline{INT1}$ can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in Register TCON. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON.

When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt was transition-activated.

If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

The Timer 0 and Timer 1 Interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter registers.



The timer flag set upon generation of interrupt is cleared by the on-chip hardware when microcontroller starts execution of particular interrupt service routine.

The serial port interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to service routine. In fact, the service routine will normally have to determine whether it was RI or TI that generated the interrupt and the bit will have to be cleared in software.

Interrupt control:-
(Enabling and disabling interrupts using IE)

When 8051 is reset, all interrupts are disable. These are enabled by software. All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is interrupts can be generated or pending interrupts can be canceled in software.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special function Register IE.
IE contains also a global disable bit EA, which disables all interrupts at once.

(MSB)							(LSB)
EA	-	ET2	ES	ET1	EX1	ET0	EX0

IE - Interrupt enable register.

Interrupt priority and Interrupt Destinations

(vector locations)

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in special function register IP. A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low priority interrupt. A high-priority interrupt can't be interrupt by any other interrupt source.

(MSB) (LSB)

-	-	-	PS	PT1	PX1	PT0	PX0

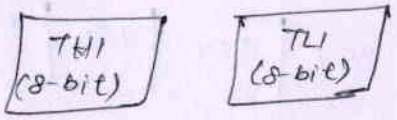
If two requests of different priority levels are received simultaneously, the request of higher priority is serviced. If requests of the same priority level are received simultaneously, an interrupt polling sequence determines which request is serviced.

DAY-8

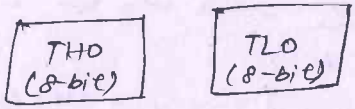
8051 Timers:-

8051 has two timers, timer 0 and timer 1. Basically both, timer 0 and timer 1 are 16-bit registers. Since 8051 is an 8-bit microcontroller, each 16-bit register can be accessed as low-byte register (TL) and high byte register (TH).

Timer 1 register



Timer 0 register



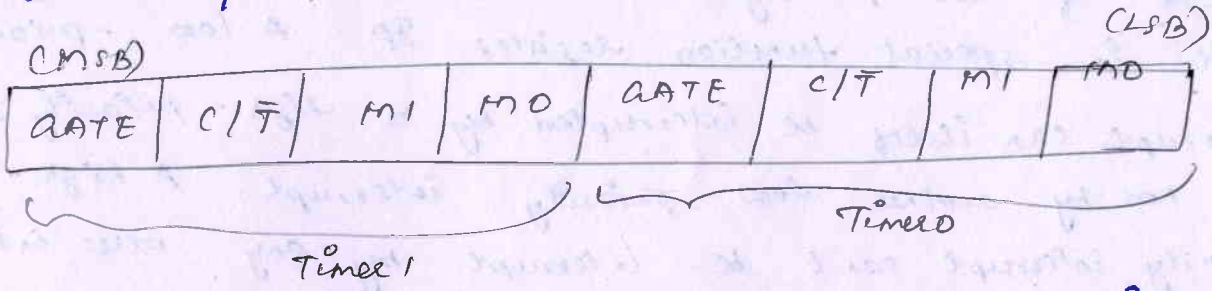
Timer control (TCON) register

Timer mode (TMOD) register

Timer registers

Structure of TMOD Register :-

Timer / counter mode control (TMOD) is the SFR in 8051 having format as shown below :-



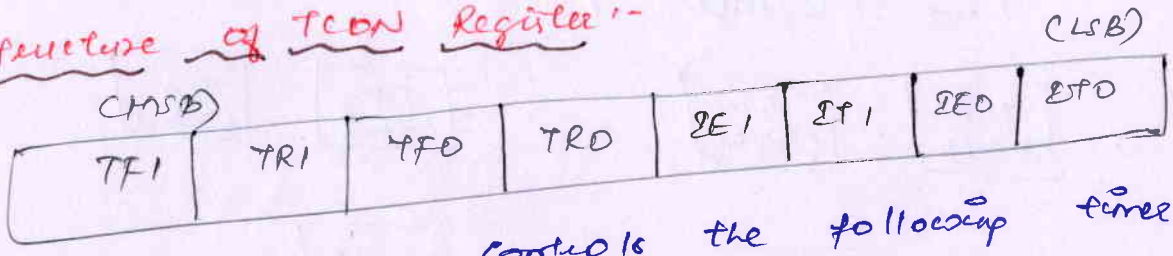
* Select Timer 0 to operate as a counter or timer.
* Select Timer 1 to operate as a counter or timer.
* Select the mode in which timer should operate.

M1, M0 :- These bits select the timer mode. There are four different modes of timer, mode 0, mode 1, mode 2 and mode 3.

C/T :- This bit is cleared (C/T = 0) for selecting 'timer' operation and is set (C/T = 1) for selecting 'counter' operation.

GATE :- Gating control when set, Timer / counter "x" is enabled only while "INTx" pin is high and "TRx" control bit is set. When cleared Timer "x" is enabled whenever "TRx" control bit is set.

Structure of TCON Register :-



The TCON register control is the following timer operations :-

* Start and stop timer 0 and timer 1

* provides status of timer / counter overflows.
* provides status of external interrupts
* configures external interrupts as either low level triggered or falling edge triggered.

8051 Timer modes and programming:-

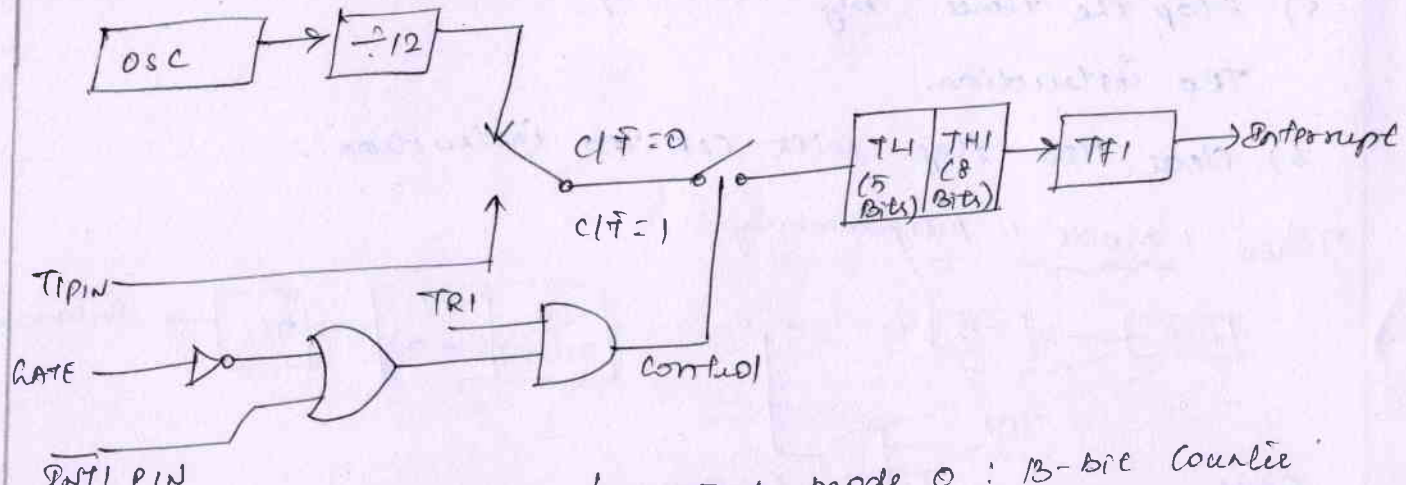
There are 4 modes of timer, mode 0, mode 1, mode 2 and mode 3. All these modes and their programming.

Mode 0:- Both timers in mode 0 are 8-bit counters with a divide-by-32 prescalar.

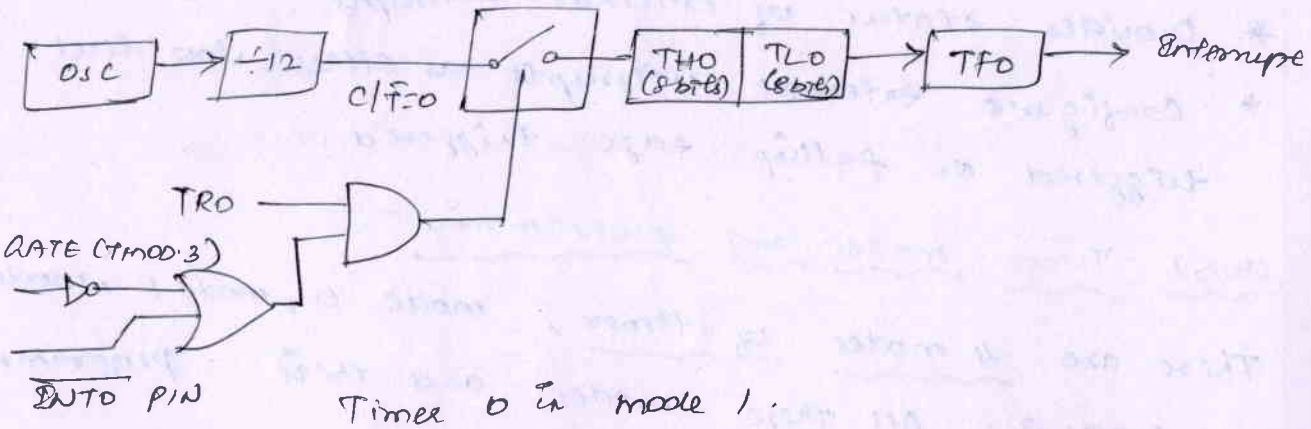
Mode 0 operation is same for timer 0. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD 3).

Mode 1:- Both timers in mode 1 are 16-bit counters. As the count rolls over from all 1s to all 0s, it set the timer interrupt flag TF.

The counted input is enabled to timer when TR=1 & either GATE=0 or $\overline{INT1}$ = 1.



Timer / counter 1 mode 0 : 13-bit counter

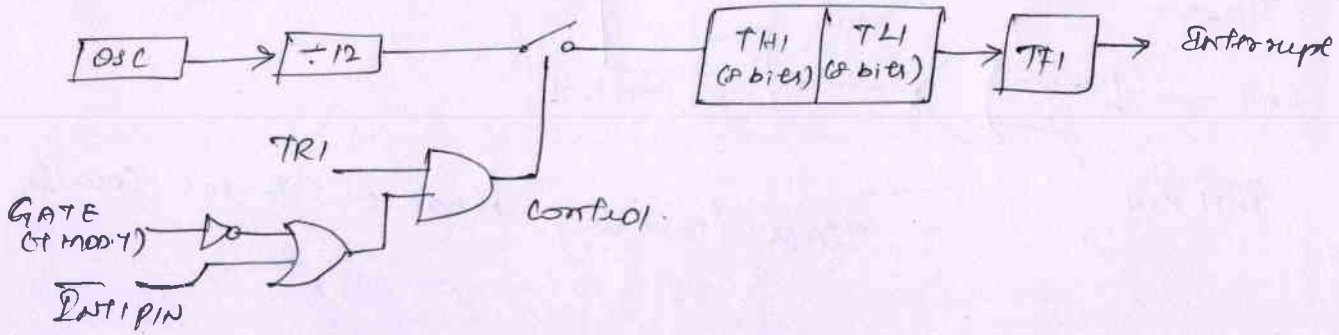


Timer 0 in mode 1.

A time delay can be generated using mode 1 of the timer 0 using following steps:-

1) Load TMOD register indicating timer 0 is used and mode 1 is selected.
2) Load TL0 and TH0 registers with count values.
3) Start the timer by setting TR0 bit = 1.
4) Monitor the timer flag (TF0) with the JNB TF0, target address instruction. When it is raised get out of the loop.
5) Stop the timer by clearing TR0 bit = 0 with CLR TR0 instruction.
6) Clear TF0 flag with CLR TF0 instruction.

Timer 1 mode 1 programming:-

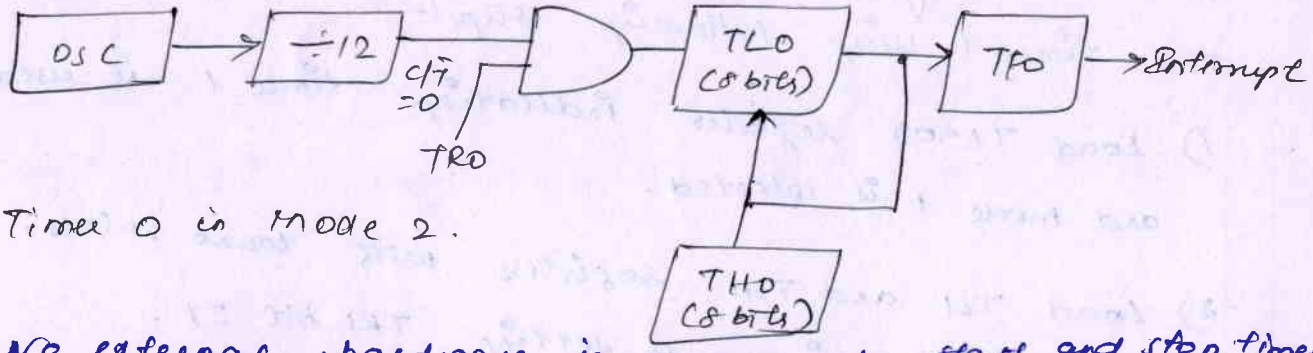


A time delay can be generated using mode 1 of the timer 1 using following steps :-

1) Load TMOD register indicating timer 1 is used and mode 1 is selected.
2) Load TL1 and TH1 registers with count values.
3) Start the timer by setting TR1 bit = 1.
4) Monitor the timer flag (TF1) with the JNB TF1, target address instruction. When it is raised, get out of the loop.
5) stop the timer by clearing TR1 bit = 0 with CLR TR1 instruction.
6) Clear TF1 flag with CLR TR1 instruction.

Timer 0 Mode 2 :-

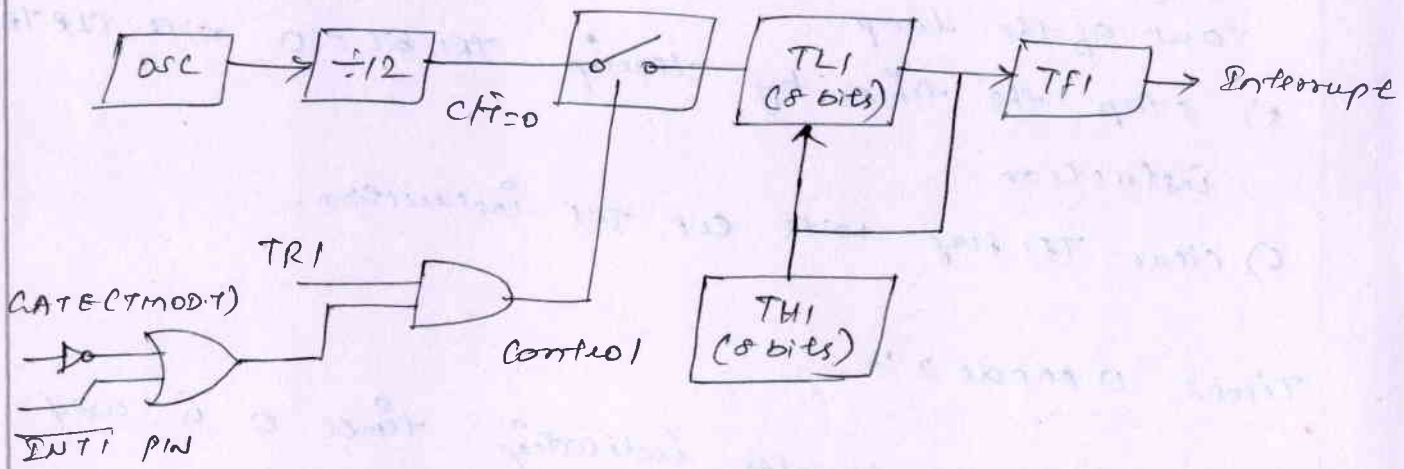
1) Load TMOD register indicating timer 0 is used mode 2 is selected.
2) Load TH0 register with count value.
3) start the timer by setting TR0 bit = 1
4) Monitor the time flag (TF0) with the JNB TF0, target address instruction. When it is raised, get out of the loop.
5) Clear the TF0 flag, with CLR TF0 instruction.
6) Go back to step 4. There is no need to load TH0 register again since mode 2 is auto-reload.



Timer 0 in mode 2.

No external hardware is used to start and stop timer.

Timer 1 mode 2 programming :-



1) Load TMOD register indicating timer 1 is used and mode 2 is selected.
2) Load TH1 register with count value.
3) Start the timer by setting TR1 bit = 1
4) Monitor the timer flag (TF1) with the JNB TF1, target address instruction. When it is raised, get out of the loop.
5) Clear the TF1 flag, with CLR TF1 instruction.
6) Go back to step 4. There is no need to load TH1 register again since mode 2 is auto-reload.

mode 3:-

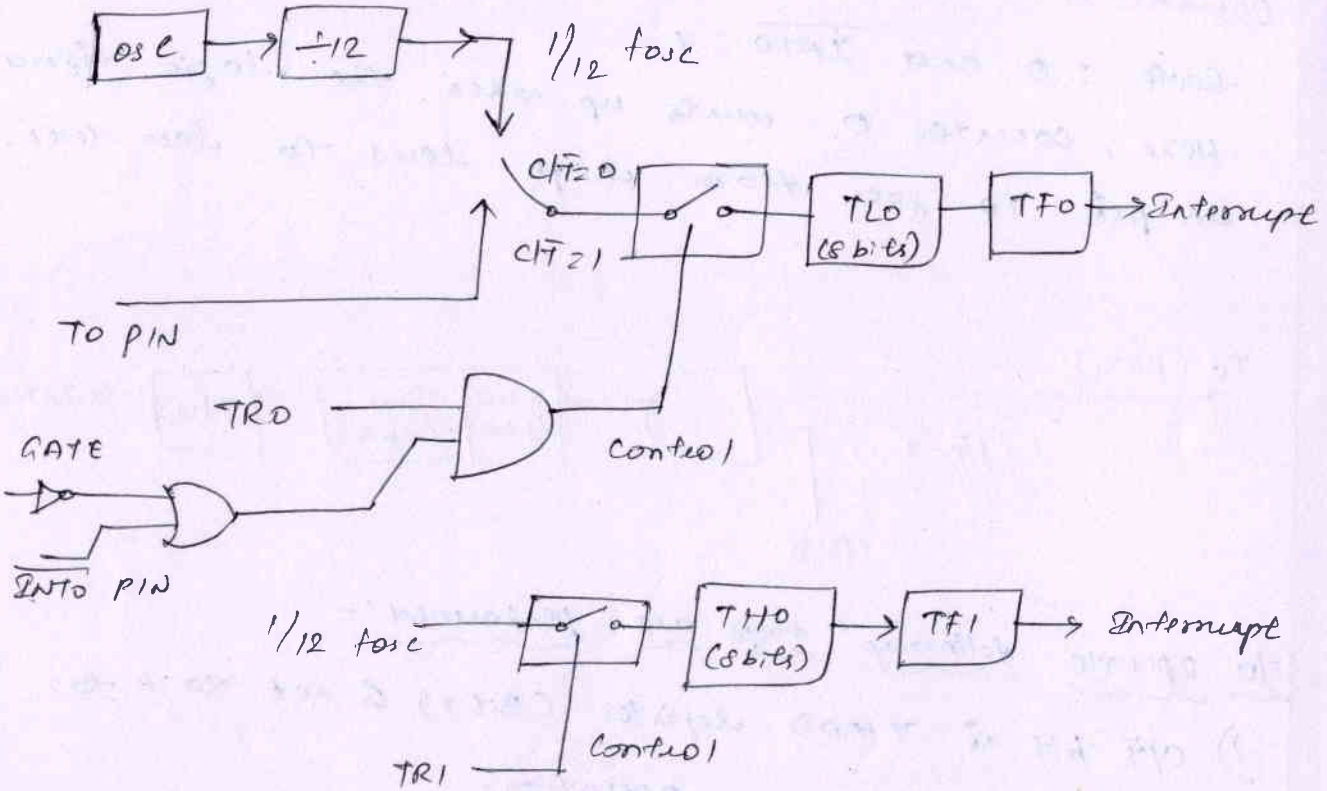
Timer 1 in mode 3 simply holds its count. The effect is the same as setting TR1 = 0. Timer 0 in mode 3 establishes TL0 and TH0 as two seperate counters.

TL0 uses 2 seperate counters.

(1) Logic for mode 3 on Timer 0.

(2) TL0 uses the timer 0 control bits: C/$\overline{T}$, GATE, TR0, $\overline{INT0}$ and TF0.

(3) TH0 is locked into a timer mode and takes over the use of TR1 and TF1 from timer 1



Timer / counter 0 mode 3:- Two 8-bit counters.

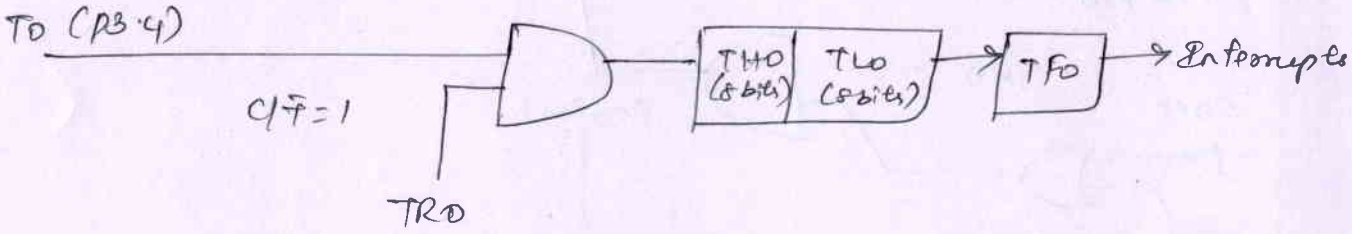
8051 Counters :-

C/T Bit in TMOD Register :- This decides the timer / counter functioning as a counter or a timer. When C/T bit in the TMOD register is 0, the timer mode is selected. When timer / counter is used as a timer, the 8051's crystal is used as a source of the frequency. When C/T bit in the TMOD register is 1, the counter mode is selected. When timer / counter is used as a counter, it gets its pulses from outside the 8051. The counter counts up for each clock pulse applied at this pin. These pins are called T0 (timer 0 clock input) and T1 (timer 1 clock input).

Counter 0 in mode 1 :-

Gate = 0 and $\overline{INT0}$ = 1.

Here, counter 0 counts up when the logic signal on pin T0 goes from high level to low level.



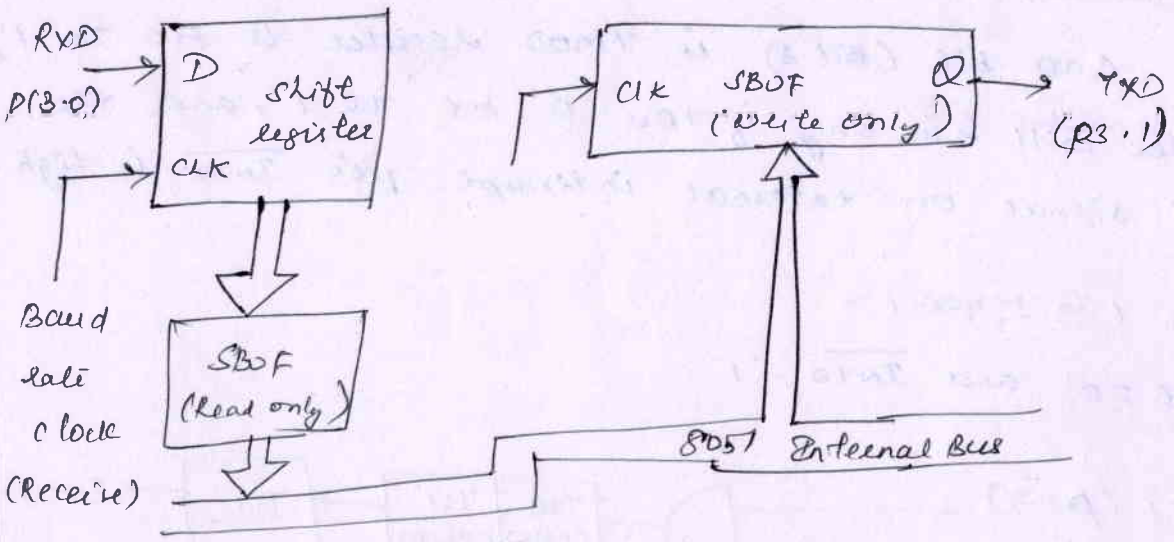
To operate following steps are performed :-

1) C/T bit in TMOD register (Bit 2) is set to 1 to allow counter mode operation.

2) M1 : M0 bits (bits 1 : 0) in TMOD register are set to 01 to select mode 1.

* 8051 Serial port :-

DAY-9



The serial port of 8051 is full duplex, means it can transit and receive simultaneously. It uses register SBUF to hold data. Register SCON controls data communication, register PCON control data rates and pin RXD (P3.0) and TXD (P3.1) do the data transfer.

SBUF is an 8-bit register dedicated for serial communication in 8051. Its address is 99H. It can be addressed like any other register in 8051.

The SBUF loads data to be transmitted and reading SBUF accesses received data. There are two separate and distinct registers, the transmit write-only register, and the receive read only register.

* Transmission :-

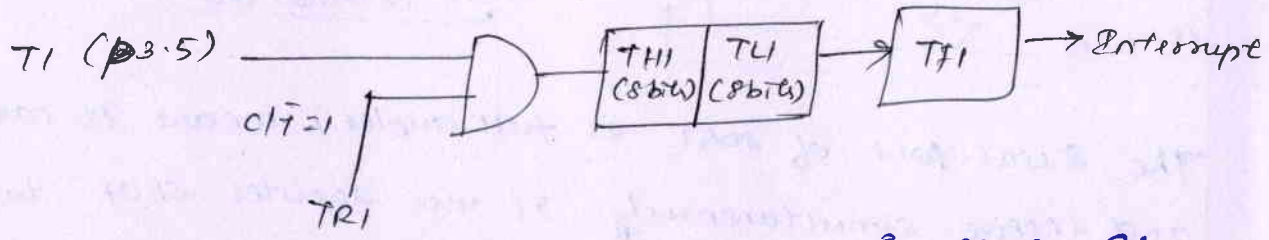
When a byte of data is to be transmitted via the TXD pin, the SBUF is loaded with this data byte. As soon as a data byte is written into SBUF, it is framed with the start and stop bits and transmitted.

3) When Gate bit (Bit 3) in TMOD register is cleared to 0, TR0 bit is set to 1 to start the counter.

4) When Gate bit (Bit 3) in TMOD register is set to 1, counter will run only if TR0 is set to 1, and the logic signal on external interrupt pin $\overline{INT0}$ is high.

Counter 1 in mode 1:-

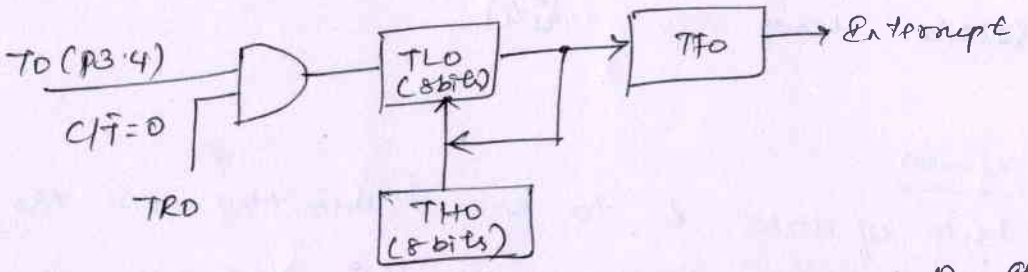
Gate = 0 and $\overline{INT0}$ = 1



The operation of counter 1 in mode 1 is same as counter 0 operation. The only difference is that here registers for counter 1 are programmed instead of counter 0.

Counter in mode 2:-

In this mode, counter is used in auto-reload mode instead of 16-bit counter. Rest is same as mode 1.



Counter 0 control logic when GATE = 0 and $\overline{INT0}$ = 1

Reception :-

When 8051 receives data serially via RXD pin of it, the 8051 deframes it. the start and stop bits are seperated out from a byte of data. This byte is placed in SBUF register.

Bit patteen of SCON register :-

The 8051 provides four programmable modes for serial data communication. A particular mode can be selected by setting the SM0 and SM1 bits in SCON. The mode selection also decides the baud rate.

(MSB) 7	6	5	4	3	2	1	0 (LSB)
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Operating modes for serial port :-

Mode 0 :- In this mode, serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted / received. 8 data bits (LSB first). the baud rate is fixed at 1/12 the oscillator frequency.

Mode 1 :- In this mode, 10 bits are transmitted or received. A start bit (0), 8 data bits (LSB) & stop bit (1).

Mode 2 :- In this mode, 11 bits are transmitted or received. A start bit (0), 8 data bits (LSB), programmable 9th data bit and a stop bit (1).

mode 3 :- In this mode, 11 bits are transmitted or received. A start bit (0), 8 data bits, 9th data bit and a stop bit (1).

Generating Baud rate :-

$$\text{Baud rate} = \frac{\text{Oscillator frequency}}{12}$$

Using Timer/counter 1 to Generate Baud rate.

$$\text{Baud rate} = \frac{K \times \text{oscillator frequency}}{32 \times 12 \times [(256 - TH1)]}$$

Using Timer/counter 2 to Generate Baud rate.

$$\text{Baud rate} = \frac{\text{Timer 2 overflow rate}}{16}$$

Serial port in mode 2.

The baud rate is fixed in this mode and is $\frac{1}{32}$ or $\frac{1}{64}$ of the oscillator frequency.

SMOD = 1, Baud rate = $\frac{1}{32}$ oscillator frequency.

SMOD = 0, Baud rate = $\frac{1}{64}$ oscillator frequency.

Programming :- DAY-10

1) 8051 uses 11.0592 MHz Crystal. To get 9600 Hz baud rate how will you program it for serial transmission.

Sol:-

$$TH1 = 256 - \frac{k \times \text{oscillator frequency}}{384 \times \text{Baud rate}}$$

$$= 256 - \frac{1 \times 11.0592 \times 10^6}{384 \times 9600} = 253 = FDH.$$

Program:-

```
MOV TMOD, #20    ; Initialize timer 1 in mode 2
MOV SCON, #4CH   ; Initialize serial mode 1
ORL PCON, #80H   ; Make SMOD=1
MOV TH1, #FDH    ; Load count.
```

2) Write an 8051 assembly language program to transfer letter "A" serially at 9600 baud rate continuously.

```
        MOV TMOD, #20H   ; timer 1, mode 2
        MOV TH1, #FDH    ; 9600 baud rate
        MOV SCON, #50H   ; 8-bit, 1 stop REN enabled
        SETB TR1         ; start timer 1
START:  MOV SBUF, #"A"   ; Letter 'A' to be transferred
HERE:   JNB TI, HERE     ; wait for last bit to transfer
        CLR TI           ; clear TI for next character
        SJMP START       ; Go to send the character again.
```

Programming 8051 for Receiving Data serially.

(3) Write an 8051 assembly language program to receive bytes serially with baud rate 9600, 8 bit data and 1 stop bit. simultaneously send received bytes to port 2.

Sol:-

```
        MOV TMOD, #20H      ; timer 1, mode 2
        MOV TH1, #FDH       ; 9600 baud rate
        MOV SCON, #50H      ; 8-bit, 1 stop, REN enabled
        SETB TR1            ; start timer 1
HERE:   JNB RI, HERE        ; wait for character receive completely.
        MOV A, SBUF         ; save the received character
        MOV P2, A           ; send character to port 2
        CLR RI              ; Get ready to receive next byte
        SJMP HERE           ; Go to receive next character.
```

(4) Write a C program to transfer letter "c" serially at 9600 baud continuously. Use 8-bit data and 1 stop bit.

Sol:-

```
#include <reg51.h>
void main (void)
{
    TMOD = 0X20;
    TH1  = 0XFD;
    SCON = 0X50;
    TR1 = 1;
    while(1)
    {
        SBUF = 'c';
        while (TI==0);
        TI=0;
    }
}
```

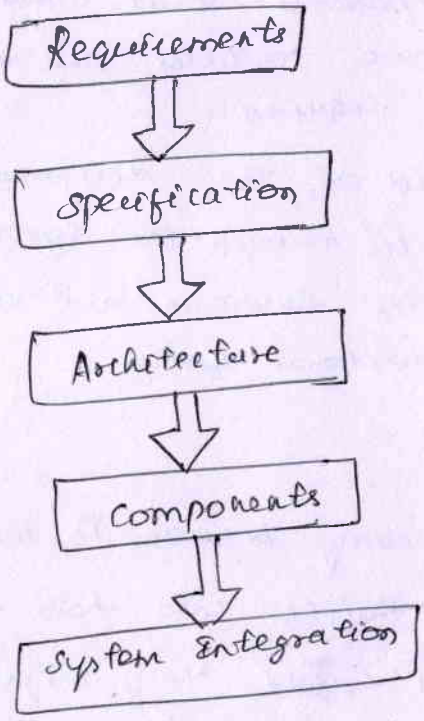
Unit-II - Embedded systems

Embedded system Design process

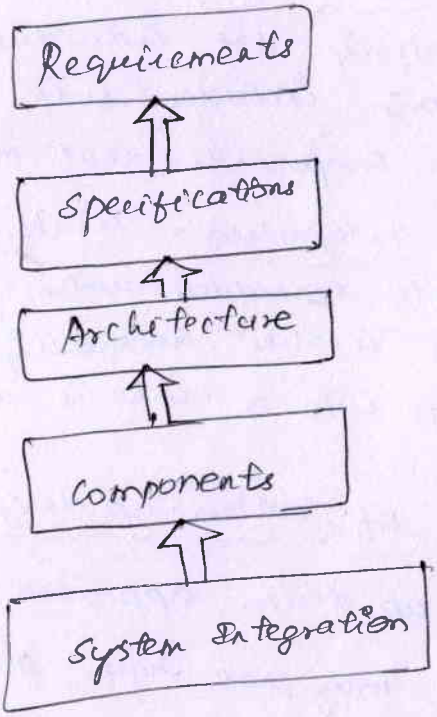
Design methodology refers to the set of principles, guidelines and best practices guiding the entire design process. It involves decisions about the choice of hardware components, software development techniques, testing procedures and more.

DAY-11

* **Major steps in embedded system design process:-**



Top down design



Bottom up design

* **System requirements:-** The process begins with understanding and documenting the system requirements. This phase sets the goals and functionalities the embedded system must achieve. It provides a broad overview of what the system needs to do.

* Specifications :- The specification outlines how the system should behave and what specific functions it needs to perform.

* Architecture development :- Once the specification is in place, the design process moves on to developing the system architecture. The architecture defines the system's structure in terms of large components or modules. It outlines how these components interact and work together to meet the specified behaviors and functions.

* Component design :- With the architecture in place, the next stage involves designing the individual components that make up the system. This includes both software modules and any specialized hardware components that may be required.

* System Integration :- Finally, based on the designed components, the complete embedded system is built through the system integration. This phase involves assembling all the hardware and software components into a cohesive and functional system.

Necessity of Bottom-up design :-

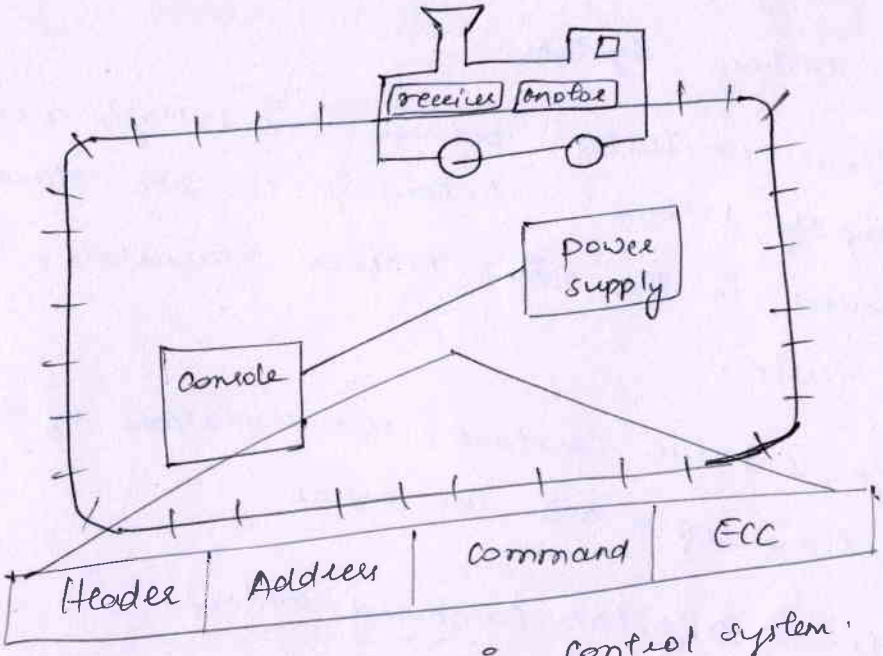
Bottom up design approach is necessary because, in many cases, designers may not have perfect insight into how later stages of the design process will unfold. They might face uncertainties or challenges that could impact decisions made at earlier stages.

Goals :-

* Manufacturing cost
* Performance
* Power consumption.

DAY-12

* Model train Controller :-



Model train control system.

Requirements :-

a) Console to control 8 trains on 1 track.
b) Throttle with at-least 63 levels in both directions.
c) Inertia control with atleast 8 level to adjust responsiveness of the train to commanded changes in speed.
d) Emergency stop button.
e) Error detection scheme for sending error-free messages.

Digital command Control (DCC)

↓ Digital command control is a mechanism for controlling locomotives and therefore trains, in a more realistic fashion. The National Model Railroad Associated developed the Digital command control (DCC) standard to support interoperable digitally controlled model trains.

* DCC does not specify the control panel, CPU type, programming language & many other features of an actual model railway system.

* the data signal oscillates between two voltages around the power supply voltage. According to DCC standards, bits are encoded in the time between transitions, not by voltage levels.

* To keep the DC value constant, the durations of the high and low sections of a bits are equal.

DCC components :- Digital command control.

* Throttle

This is also known as cab, It is the device used to control model train locomotives.

* Command station :-

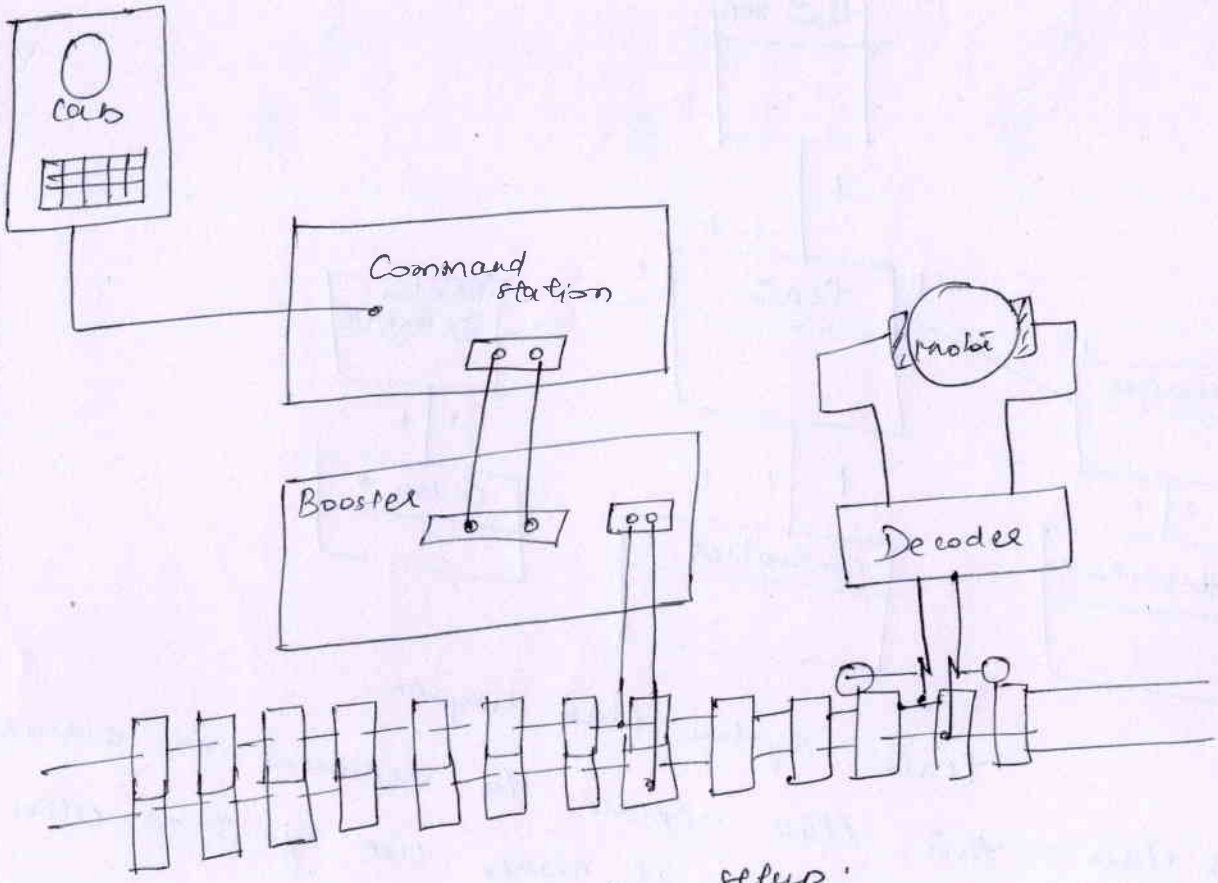
The command station is the real intelligence of the DCC system.

* Booster :-

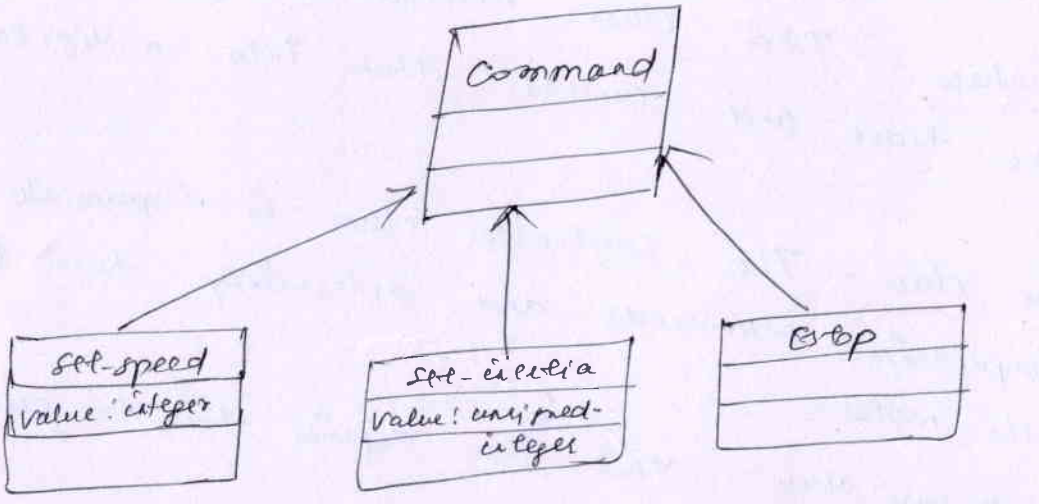
the booster receives commands from the command station, amplifies them and delivers them to the track.

* Decoder :-

A small electronic device installed inside a locomotive responsible for responding to commands from the command station.

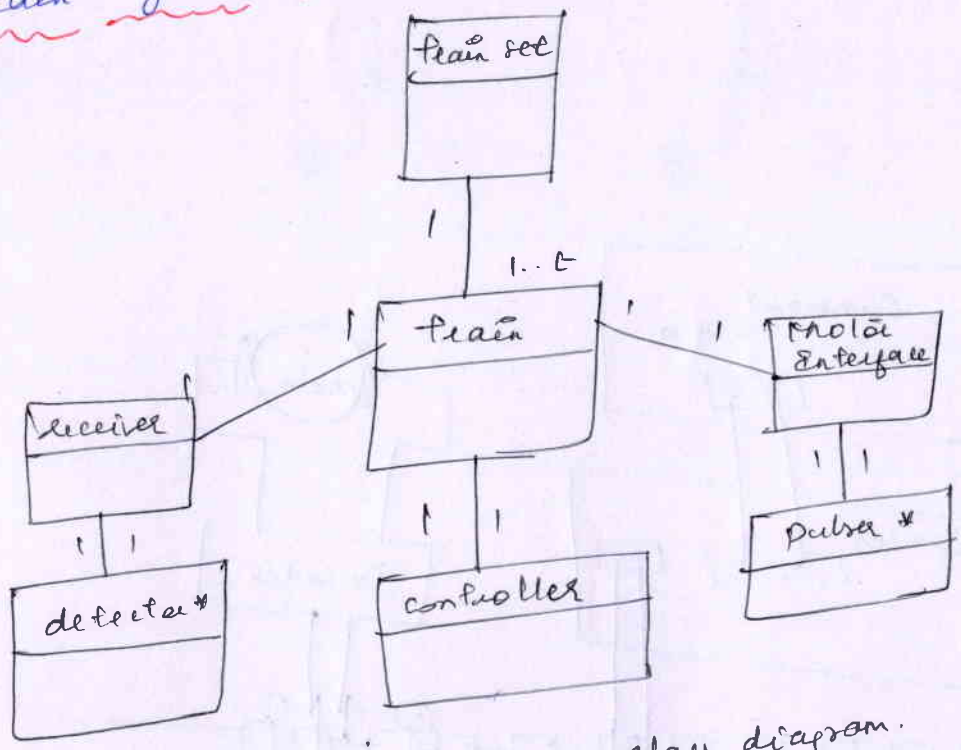


DCC system setup.



Class diagram for the train controller messages.

Train system class diagram:-



Train system class diagram.

Train class:- This class appears to represent the overall functionality of a train. It makes use of three other classes to manage its operation.

Receiver class:- This class receives analog signals from the track and converts them into a digital format.

Controller class:- The controller class is responsible for interpreting commands and determining how to control the motor.

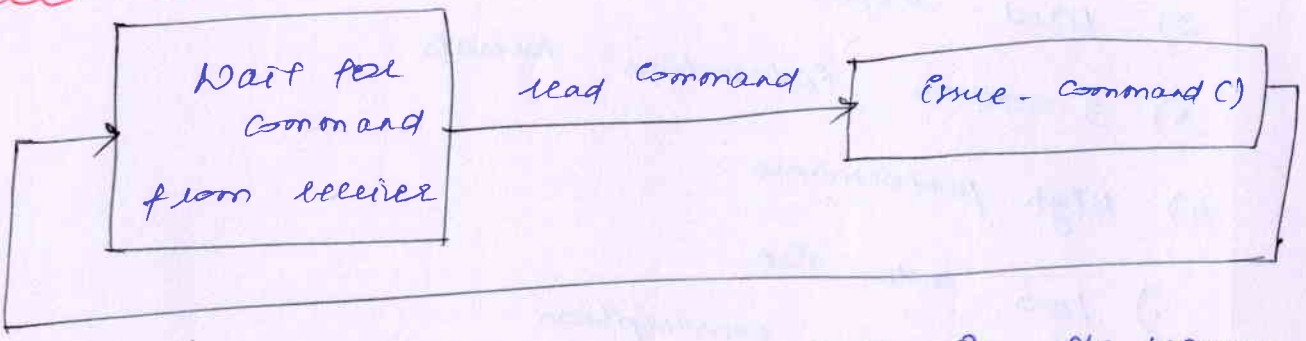
Motor interface class:- This class defines how to generate analog signals to control the motor.

Detector class:- It is responsible for detecting analog signals on the track and converting them into a digital format.

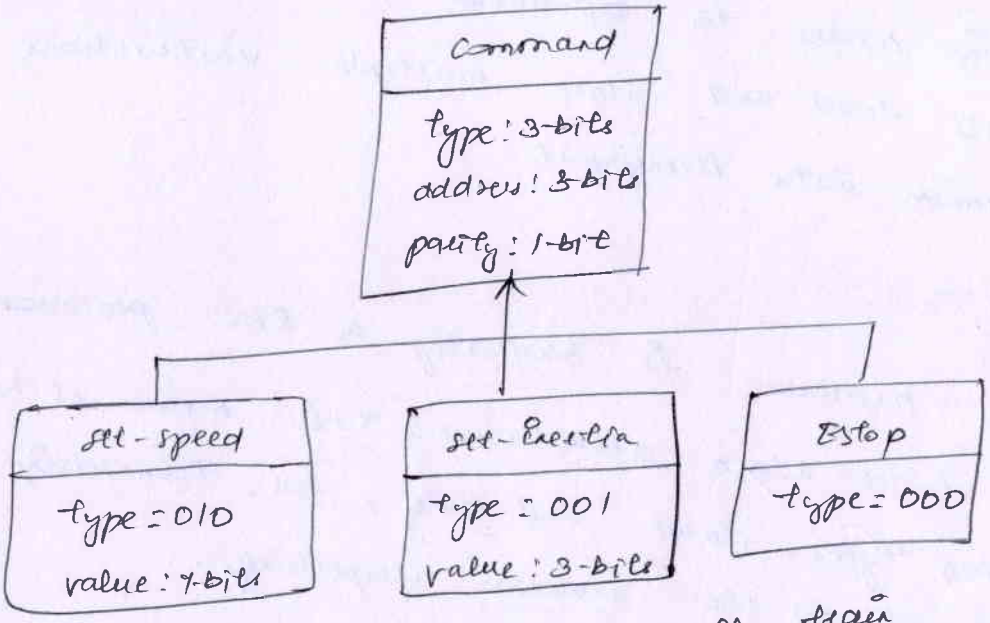
+ Pulses Class:-

It translates digital commands into analog motor control signals to control the motor speed.

State-diagram for controller operate behaviour:-



* the number of bits used to determine the message type. Here, 3 bits are used to determine message types, allowing for eight possibilities, with five codes unused.



Class diagram for the train controller commands.

* the number of bits used to determine data field lengths. This depends on the resolution requirements for speed & inertia.
* Number of parity bits. A single-parity bit is used for basic error detection.

ARM processor

DAY 13

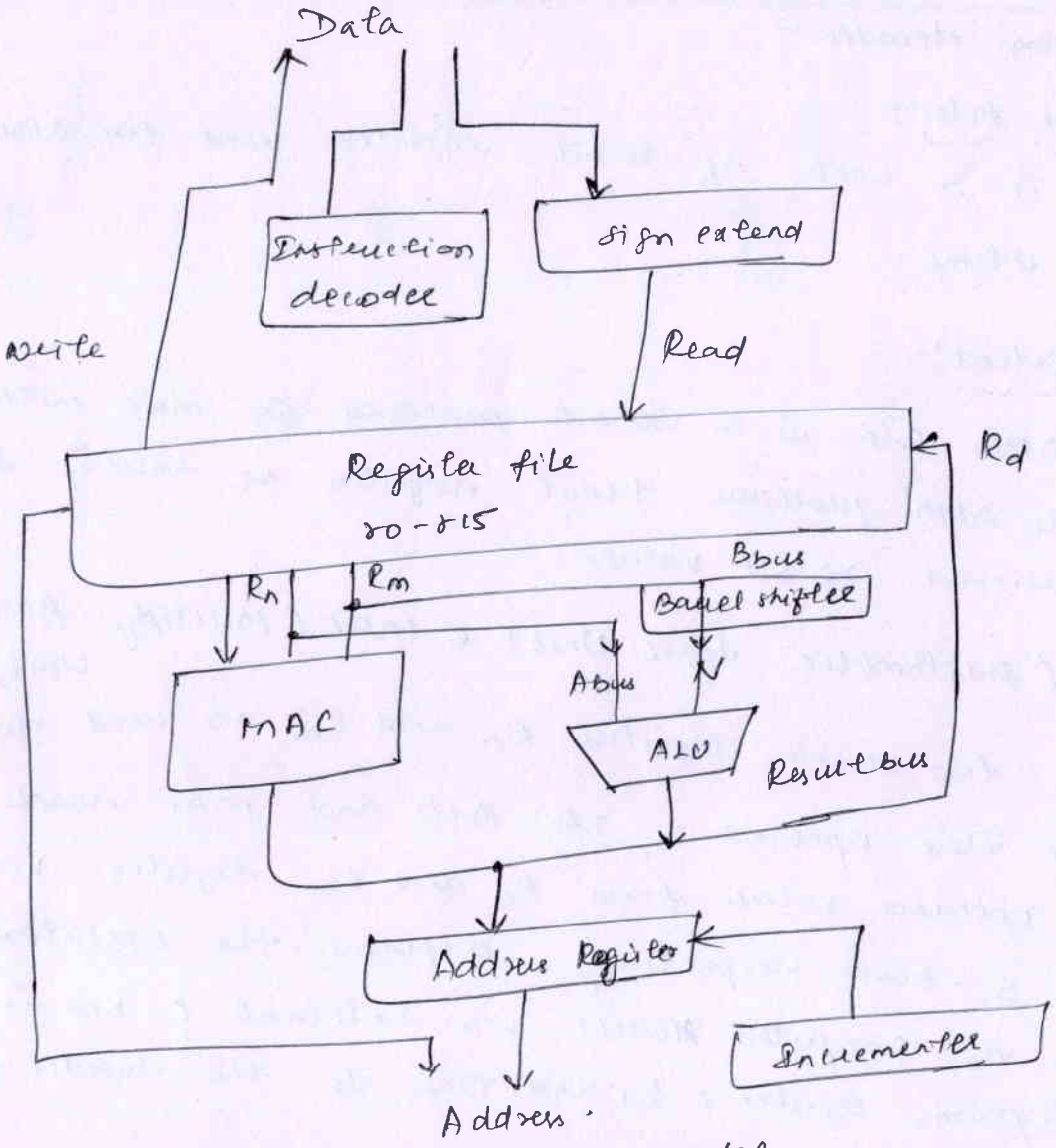
Features :-

1) Load - store architecture
2) Fixed - length 32-bit instruction.
3) 3-address instruction formats
4) High performance
5) Low code size
6) Low power consumption
7) Low silicon area
8) It has control over both ALU.
9) supports auto-increment and auto-decrement addressing modes to optimize program loops.
10) Supports load and store multiple instructions to maximum data throughput.

Architecture :-

The ARM processor is basically a RISC processor, it uses a load-store architecture. This means it has two instruction types, load and store, for transferring data in and out of the processor respectively.

* Load :- This instruction copies data from memory to registers in the processor core.
* Store :- This instruction copies data from registers in the processor core to memory.



ARM core dataflow model.

* Data bus :-

The data enters the ARM core through the data bus. The data is either in the form of an instruction opcode or a data item.

* Instruction decoder :-

This unit decodes the instruction opcode read from the memory and then the instruction is executed.

~~Instruction~~ decoder :-

Register file :-

This is a bank of 32-bit registers used for storing data items.

Sign extend :-

The ARM core is a 32-bit processor. So most instructions of the ARM processor treat registers as holding signed or unsigned 32-bit values.

ALU (Arithmetic logic Unit) & MAC (Multiply Accumulate Unit) :-

The two source registers Rn and Rm are used to store these operands. The ALU and MAC reads the operand values from Rn and Rm registers via A and B buses respectively, performs the operation and store the computed result via internal C bus in destination register, Rd and then to the register file.

The load and store instructions generate address using ALU and stores it in the address register.

Address register :-

This holds the address generated by the load and store instructions and places it on the address bus.

Barrel shifter :-

The contents of the Rm register alternatively can be preprocessed in the barrel shifter before applying as an input to the ALU.

→ Instruction set :-

Load and store instructions :-

LDR Rd, [Rx]

LDR R2, [R1]

LDRB R2, [R1]

LDRH Rd, [Rx]

LDRH R2, [R1]

STR Rx, [Rd]

STR R4, [R3]

STRB Rx, [Rd]

STRB R4, [R3]

STRH Rx, [Rd]

STRH R4, [R3]

Arithmetic Instructions :-

ADD - Add

ADC - Add with carry

SUB - SUBS

SBC - subtract with carry

MUL - multiply

UMULL - multiply long

RSB - Reverse subtract

RSC - Reverse subtract with carry.

Addition:-

ADD

ADDS

ADC

Subtraction:-

SUB

SUBS

SBC

Multiply:-

MLA (multiply & Accumulate)

UMLAL (Unsigned multiply & Accumulate long)

Division:-

UDIV (Unsigned Divide).

Logical Instruction:-

AND

ANDS

ORR

ORS

EOR

EORS

BIC

BICS

Rotate & shift:-

LSL = logical shift left

LSR = logical shift right

ASR = Arithmetic Right shift

ROR = Rotate Right

RRX = Rotate Right Extended.

Looping, Branch, Conditional Execution:-

3 types of branching:-

1) Branch
2) Branch link (BL)
3) Branch exchange (BX) and Branch link exchange (BLX)

DAY 14

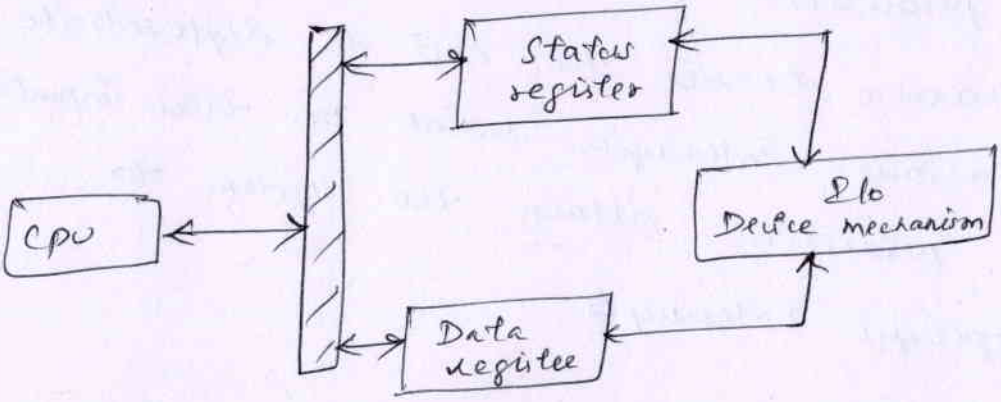
Central processing Unit:-

Programming I/O:-

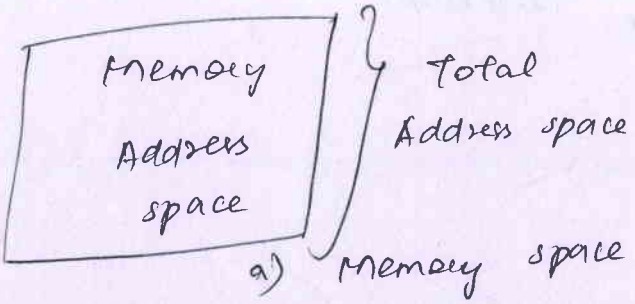
The interface between CPU and the device primarily relies on registers. These registers communicate between the CPU and device's internal components.

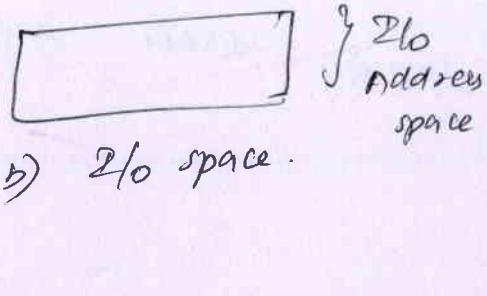
Data register:- Data registers store values that the device interprets as data.

Status register:- Status register provide information about the device's ongoing or completed operation.



Input & output primitives:-



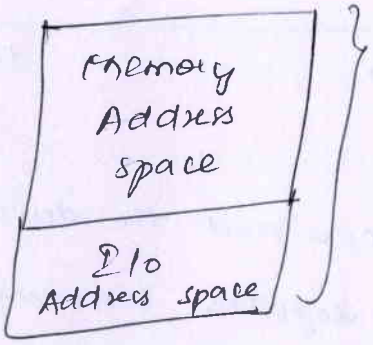


I/o mapped I/o :-

Different I/o address spaces for memory and I/o.

memory mapped I/o :-

The total memory address space is partitioned and part of this space is allotted to I/o addressing.



Interrupts :-

When an interrupt occurs, the program counter (PC) is updated to point to an interrupt handler routine, called as Device Driver or Interrupt service Routine.

priority and vectors :-

Interrupt priorities :-

This mechanism enables the CPU to differentiate between various Interrupts based on their importance. Assigning priorities allows the system to precede specific Interrupts.

Interrupt vectors :-

Interrupt vectors allow interrupting devices to specify their respective interrupt handlers.

DAY 15

* Supervisor mode, Exceptions and Traps:-

→ Supervisor mode, often referred to as privileged mode or Kernel mode, is introduced in ARM processors, and in many other CPU architectures to provide a higher level of control and privilege to the operating system or trusted system software.

→ ARM architecture provides a supervisor mode. The ARM7 enters this mode after reset. However, we can activate supervisor mode using the ARM instruction known as SWI when ARM is in user mode.

→ When the processor switches to supervisor mode, the bottom 5 bits of the current program status register (CPSR) are set to 1.

→ Before executing a SWI instruction or any other instruction that causes a mode switch, the processor typically saves the old CPSR into a special register called the saved program status register (SPSR).

Exceptions:-

Exception occurs as a direct result of executing an instruction, such as:-

1) Software Interrupt instruction
2) Undefined or illegal instruction

③ memory error during fetching an instruction
④ memory fault during read/write from memory
⑤ Arithmetic error.

→ Interrupts and exceptions are both mechanism used in computer systems to handle changes in the flow of control of a program, but they serve different purposes and are triggered in different ways.

→ Interrupts are signals external hardware devices send to the CPU to request attention.

→ They are asynchronous events that can occur at any time

→ Exceptions on other hand, are events that are generated internally by the CPU or program itself during its execution.

→ When an exception occurs, the CPU interrupts the normal flow of program execution and transfers control to an exception handler specified in the program or the operating system.

Traps :-

A trap exception is a specific type of exception that occurs when a program or the operating system intentionally generates an exception condition using a software interrupt

→ Trap exceptions are used to change the normal flow of program execution to a predefined exception handler associated with that specific trap. This controlled diversion allows for the execution of privileged or system-level operations.

→ One of the primary uses of trap execution is to facilitate the transition from user mode to supervisor mode. The entry into supervisor mode is a critical operation from a security perspective.

DAY 16

Models for programs:-

This contains the model both data operations (eg, arithmetic calculations), and control operations (eg, conditionals). The strength of CDFA lies in its ability to combine both control and data elements.

Data flow graph:-

In high level language, a sequence of statements with a single entry and exit point without control statements, is called a basic block.

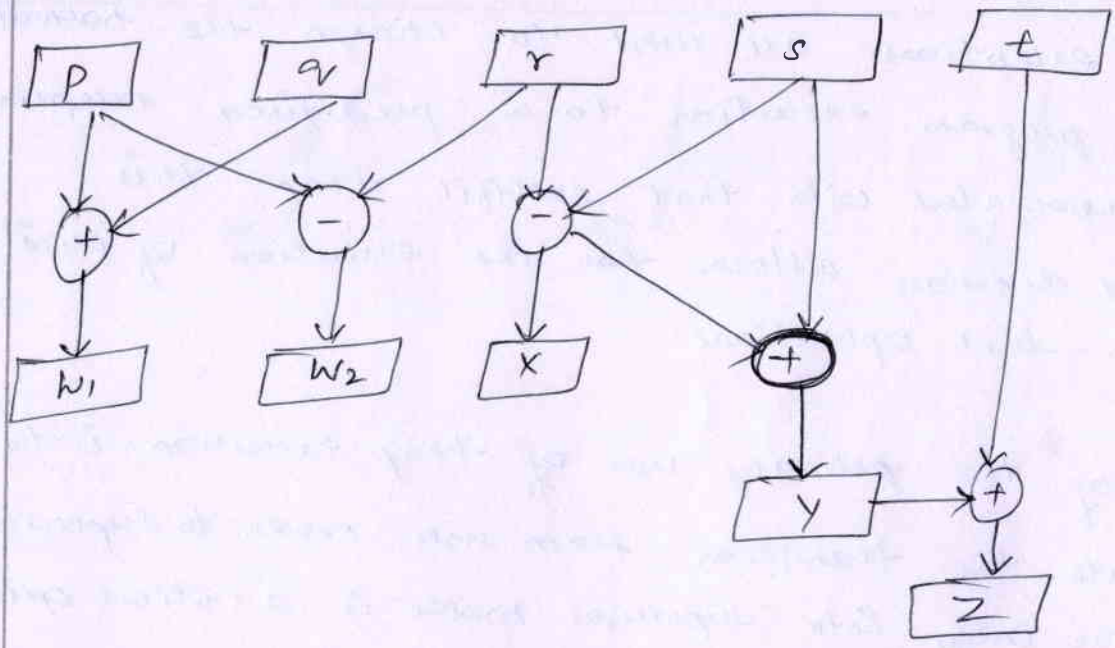
$N = p + q$

$X = r - s$

$N = p - r$

$y = X + s$

$2 = y + t$



This shows the data flow graph for a single assignment form. Each variable is defined at most once and data flow goes from the definition point to its uses without any loops.

The value nodes can be either inputs to the basic block, such as p and q or variables alloted within the block, such as w1 and y.

Controls / Data flow graphs :-

→ A control / Data flow graph (CDFG) adds constructs to describe control in the data flow graph.

In a basic CDFG, there are two types of nodes :-

* Decision nodes :- Decision nodes are used in CDFGs to represent control flow decisions, such as conditional branches and loops. These nodes typically have two or more outgoing edges, each corresponding to a

possible control flow path depending on the outcome of the decision. A single type of decision node can be used to represent all types of control in a sequential program.

Data flow nodes :-

Data flow nodes in a CDFG encapsulate a complete data flow graph for a basic code block.

DAY 17

* Assembly, Linking and Loading :-

The final compilation steps are assembly and linking, which convert instructions into an image of the program's bits in memory.

Compiler :-

This produce an intermediate representation of the program in the form of human-readable assembly language. This choice allows compiler writers to focus on the high-level aspects of compilation without dealing with low-level details like instruction formats and memory addresses.

Assembler :-

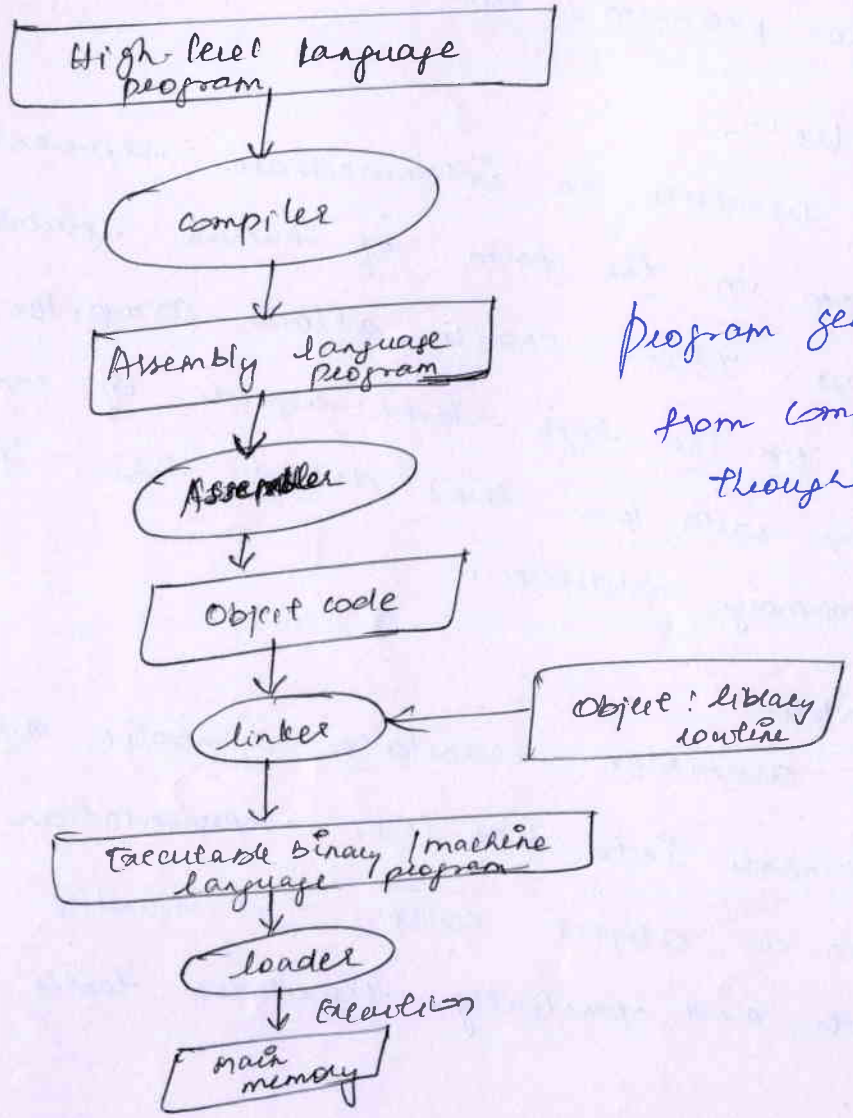
The assembler translates symbolic assembly language statements into bit-level representations of instructions known as object code. It handles instruction formats and partially translates labels into addresses.

* Linker :-

Linking is the step that follows assembly. It involves combining multiple object files generated from different source files and resolving references between them. The linker determines the final addresses of instructions and data, producing an executable binary file.

* Loader :-

The program responsible for bringing the executable program into memory for execution is called a loader. It ensures the program is correctly placed in memory and ready to execute.



Program generation from compilation though loading.

* Compilation Techniques!- DAY 18

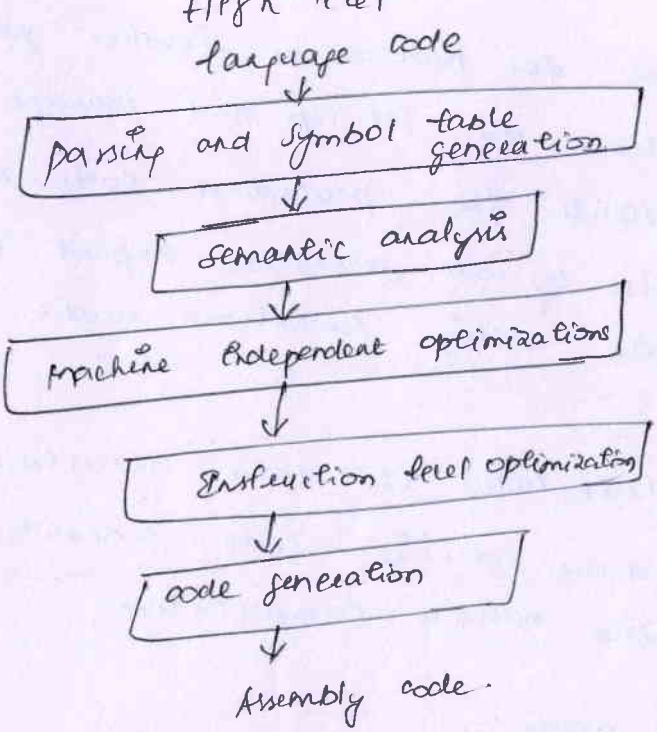
Understanding the compilation process helps meet your performance goals by writing high-level code that gets compiled into the instructions you want or by identifying when to write assembly code and how to integrate it into your high-level language program.

Compilation is not just about translation!-

It also involves optimization.

+ The translation process converts high-level language programs into lower-level instructions.

+ The optimization techniques improve the instruction sequences. This process is more effective than independently translating source code statements.



Steps in Compilation process :-

1) parsing :- In this initial step, the compiler breaks the code into individual statements and expressions.

2) Symbol table generation :- During parsing, the compiler identifies and records all named objects, such as variables, functions and data types, in a symbol table.

3) semantic analysis :- this phase focuses on checking the code for semantic correctness.

4) Machine - independent optimization :- It tries to make the intermediate code more efficient by transforming a section of code that doesn't involve hardware components like CPU registers or any absolute memory location.

Procedures :-

Code generation for procedures involves generating the necessary instructions to set up and manage procedure calls, pass parameters, execute the procedure code and handle the return. The specifics of code generation depend on the CPU's architecture and the calling conventions used.

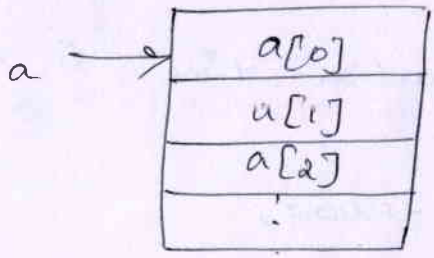
Data structures :-

The compiler translates the data structure references in to memory addresses during code generation. This translation requires address computation.

One - dimensional array :-

The first element of the array is accessed using

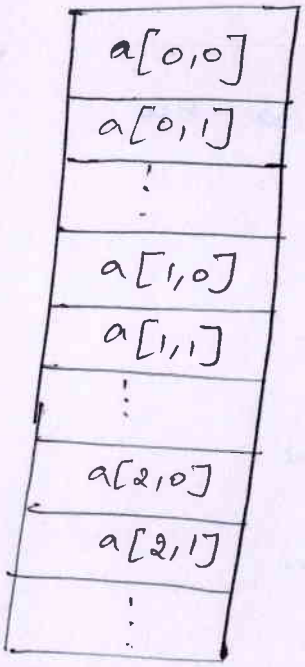
a[0], the second element with a[1], the third with a[2] and soon.



Layout of a one-dimensional array in memory.

Two-dimensional array :-

The inner variable various most, each row is stored as a continuous memory block and the elements within that row are stored sequentially.



Memory Layout for two-dimensional array.

* Program level performance Analysis :- DAY-19

Assessing cpu performance differs from evaluating program performance the cpu clock speed is not a reliable indicator of how a program will perform.

It is also essential to note that a cpu's ability to execute specific program segments quickly does not

guarantee that it will run the entire program at the rate we desire.

Determining program execution time is a challenging task:-

Because of following reason,

1) Input data values
2) Cache Behaviour
3) Instruction-level variability.

Various ways to measure program performance:-

1) Microprocessor simulators
2) Timer on the microprocessor BW
3) Logic analyzer.

Various performance measures:-

1) Average-Case Execution Time
2) Worst-Case Execution Time
3) Best-Case Execution Time

Elements of program performance:-

Program execution time is given as

Execution time = program path + Instruction timing

Program path:- It is the sequence of instructions executed by the program either in its low-level representation or high level language.

Instruction Timing:-

It is determined based on the sequence of instructions traced by the program path.

Challenges in estimating program:-

1) Indirect correspondence
2) Memory & variable Estimations.
3) Compiler optimization.

Handling challenges in Estimating Instruction Execution Time:-

1) Using look up tables
2) Considering Nearby Instructions
3) Operand variations.

Measurement - Driven performance Analysis:-

→ Determining the worst-case execution path of a program requires providing specific inputs.

→ Identifying the exact inputs that guarantee worst-case execution is often infeasible.

→ To accurately measure a program's performance on a particular CPU type, you need access to that CPU or its simulator.

Example:-

Identify data-dependent paths in the nested if statements given below:-

```
if (p)          /* Test 1 */
  {
    if (q)      /* Test 2 */
      {
        e = a * b      /* Assignment 1 */
      }
    else
      {
        e = a + b;     /* Assignment 2 */
      }
    f = a - b;         /* Assignment 3 */
  }
else
  {
    if (r)      /* Test 3 */
      {
        g = c - a;     /* Assignment 4 */
      }
  }
```

Unit- III

Processes and operating system.

DAY-20

* Structure of a Real-time System:-

Time constraints are the key parameter in real time systems. It controls autonomous system such as robots, satellites, air traffic control and hydroelectric dams.

When user gives an input to the system, it must process within the time limit and result is sent back.

Real time system fails if it does not give results within the time limits.

A real-time system is any information processing system which has to respond to externally generated input stimuli within a finite and specified period.

Real time systems are those systems in which the overall correctness of the system depends on both the functional correctness and the timing correctness. Real time systems also have a substantial knowledge of the system it controls and the applications running on it.

Rtas kernel services:-

1. Timers
2. Device I/o supervisor
3. Dynamic memory allocation
4. Intertask communication and synchronization
5. Task management.

Characteristics of RTOS:-

* Determinism
* Responsiveness
* User control
* Reliability
* Fail-soft operation.

Real time systems are of clock based or event based.
Interactive is one more category of real time system.

1) **Clock based tasks:-**

Clock based tasks are also called cyclic or periodic tasks.

→ All jobs of a periodic task t_i have a regular inter arrival time T_i, we call t_i the period of the periodic task t_i.

Static scheduling is used for scheduling periodic tasks

plant time constant → sampling time (T_s) → Interrupt.

2) **Event based tasks:-**

→ Event based task is called aperiodic tasks.

→ Actions are to be performed not at particular times or time intervals but in response to some event. The system must respond within a given maximum time to a particular event.

3) **Interactive systems:-**

→ The combination of clock based system and event based system which gives the importance of average execution time of the task is called interactive systems.

→ This system receives the input from the plant or operator and initiates the task and executes within the average response time.

Hard real time systems	Soft Real time systems
* Hard real-time system is one where the response time is specified as an absolute value. * This time is normally dictated by the environment. * Time granularity is in millisecond. * It support short term data integrity. * Safety is critical * Error detection is by system * Eg:- Automobile breaking system	* A soft real-time system is one where the response time is normally specified as an average value. * This time is normally dictated by the business or market. * Time granularity is in second * It support long term data integrity. * Safety is non-critical. * Error detection is by user * Airline reservation system.

DAY-21

* **Task Assignment and scheduling:-**

Tasks are units of sequential code implementing the system actions and executed concurrently by an OS.

* A task, also called a thread, is a simple program that thinks it has the CPU all to itself. The design process for a real-time application involves splitting the work to be done into tasks responsible for a portion of the problem.

Real time tasks are of two types:-

1) Periodic tasks:- This types of tasks consists of an infinite sequence of identical activity, called instances, which are invoked within regular time period.

2) Non periodic tasks:-
This invoked by the occurence of an event.

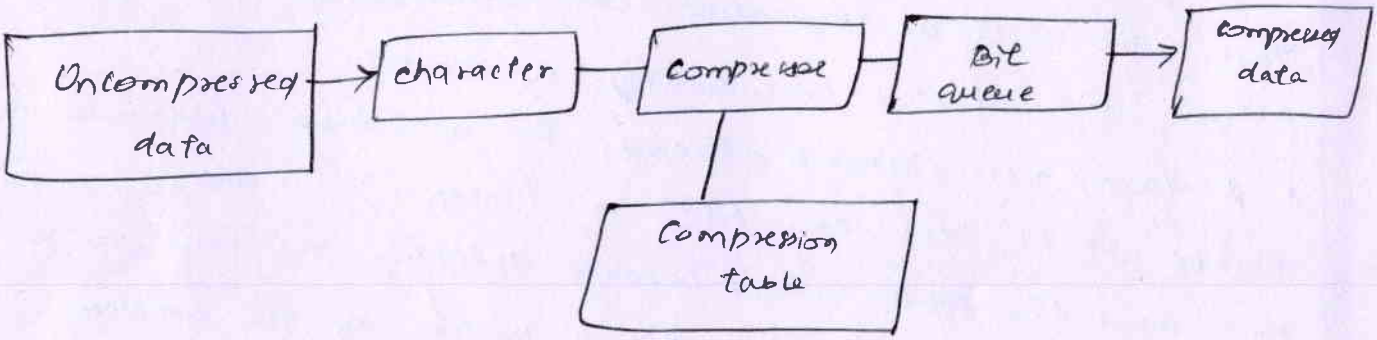
A task is a set of related jobs which jointly provides some function. A job is a unit of work that is scheduled and executed by a system.

Multiple tasks and multiple processes :-

Terms used in the real time task.

1. Release time (or) ready time
2. Deadline
3. Response time
4. Complete time
5. Laxity

A process is a sequential program in execution. Terms like job and task are also used to denote a process. Task is a single instance of an executable program.



→ Input and output of the compresse box is serial ports. It takes uncompressed data and process it. output of the box is compressed data. Given data is compressed using a predefined compression table. Modem is used such type of box.

← The program's need to receive and send data at different rates. It is an example of rate control problem. It uses asynchronous input.

DAY-22

* Multirate systems:-

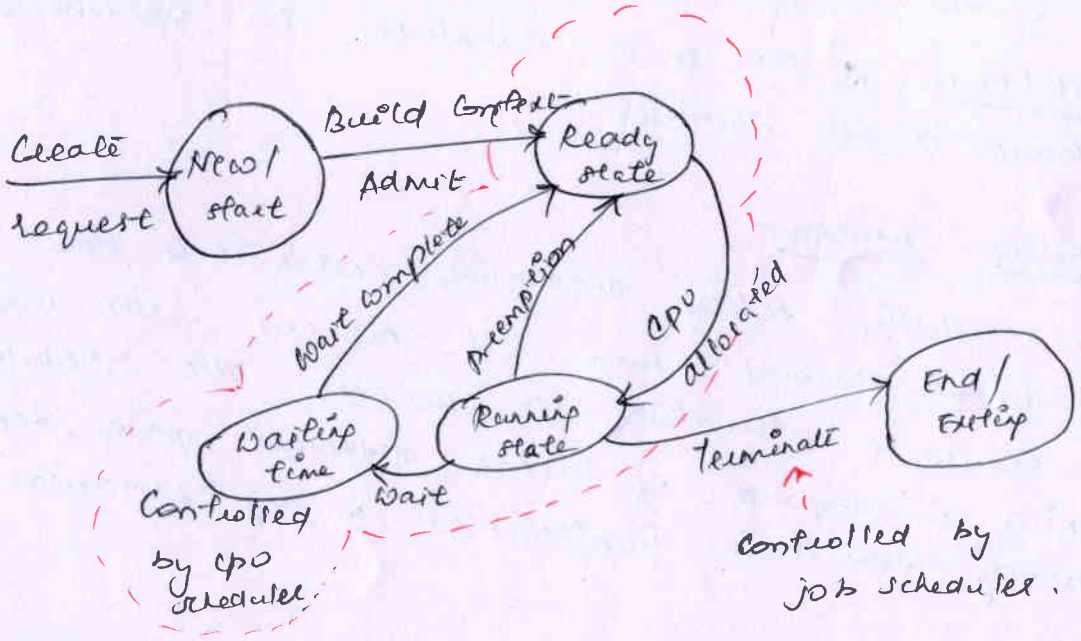
More complicated control systems have multiple sensors and actuators and must support control loops of different tasks. Multirate embedded computing system includes automobile engines, printers and cell phones.

Ex:- Automotive engine control

Tasks in automotive engine control are spark control, crankshaft sensing, fuel air mixture, oxygen sensor and kalman filter.

Process state and scheduling:-

Each process has an execution state which indicates what process is currently doing. The process descriptor is the basic data structure used to represent the specific state for each process.



State diagram is used by process manager to determine the type of service to provide to the process. The process states are as follows :- New, ready, running, waiting and end.

* New :- Operating system creates new process by using fork() system call. These process are newly created process and resources are not allocated.

* Ready :- The process is competing for the cpu. Process reaches to the head of the list (queue).

* Running :- The process that is currently being executed. Operating system allocates all the hardware and software resources to the process for execution.

* Waiting :- A process is waiting until some event occurs such as the completion of an input-output operation.

* Exit / End :- A process is completes its operation and releases it all resources.

Scheduling process :-

* The scheduling policy determines when it is time for process to be removed from the cpu and which ready process should be allocated the cpu next. The scheduling mechanism is composed of several different parts, depending on exactly how it is implemented in any particular operating system.

---

* Preemptive Real-time Operating systems :-

* Real-time operating system executes processes based upon timing constraints provided by the system designer.

Preemption:-

→ A computation or task is preemptable if it can be interrupted when another more critical task needs to be executed.

→ If a task does not complete before its quantum expires, the system preempts it and gives the processor to the next waiting task.

→ Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process. This task is known as a context switch.

→ The context switches can occur only in kernel mode (system mode). Kernel mode is a privileged mode of the CPU in which only the kernel runs and which provides access to all memory locations and all other system resources.

Preemptive:-

It allows the kernel to vary the process priority dynamically, while a process is not running, the kernel periodically increases its priority. When a process receives some CPU time, the kernel reduces its priority.

Priority - driven scheduling:-

Priority assignment can be done as follows:-

1) **per task:-** Every job in the same task has the same priority.

2) Per job : Each job in a task can have a different priority. Eg :- Earliest Deadline first (EDF)

3) Per time tick : Job priority will be reassigned after each tick. Eg :- Least slack time first (LST).

Possible implementation of preemptive priority - driven scheduling :-

1. Assign priorities to jobs.
2. Scheduling decisions are made when job becomes ready or processor becomes idle or priorities of jobs change.
3. At each scheduling decision time, choose ready task with highest priority.

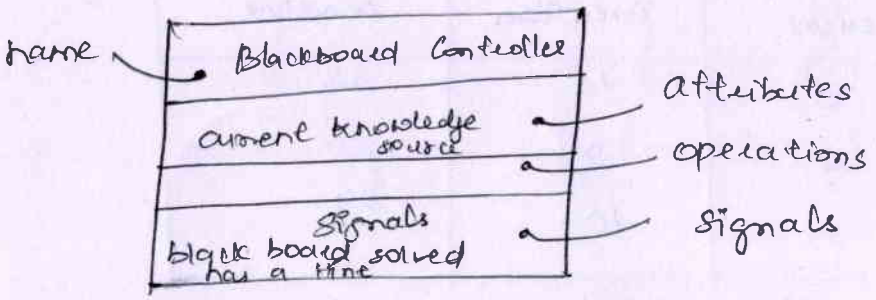
* Processes and context :-

A context switch can mean a register context switch, a task context switch, a thread context switch or a process context switch.

* A register is a small amount of very fast memory inside of a CPU that is used to speed the execution of computer programs by providing quick access to commonly used values.

Processes and object-oriented Design :-

Unified modeling language (UML) refers to processes as active objects. The objects that have independent threads of control. The class that defines an active object is known as an active class.



UML active class

Active classes are just classes which represents an independent flow of control. Active classes share the same properties as all other classes.

All the threads that lie in the context of a process are peers of one another.

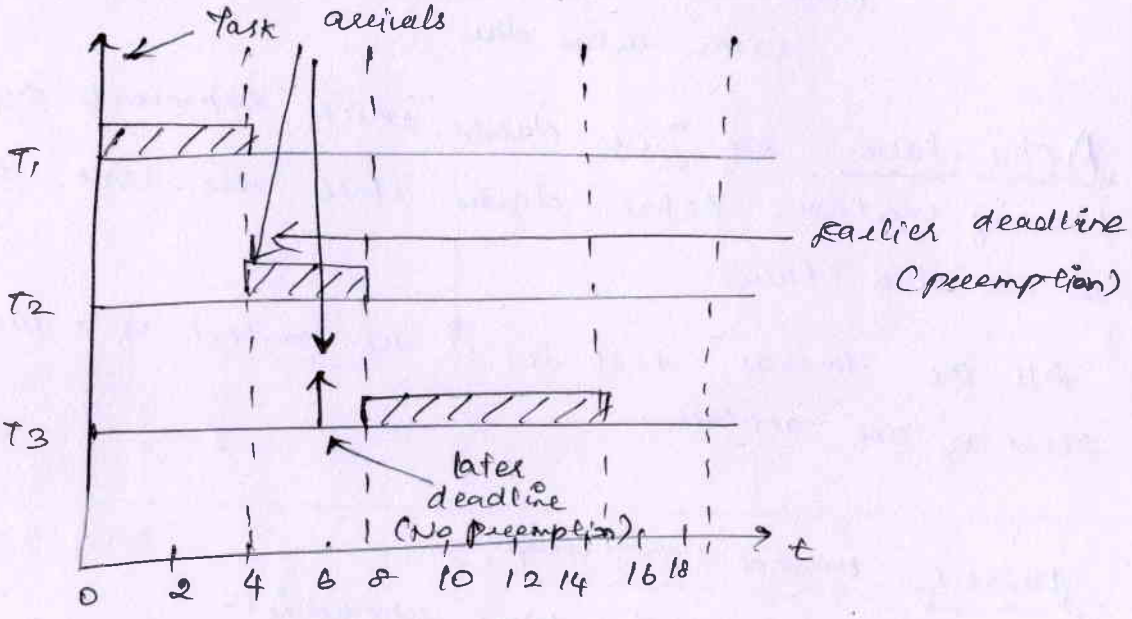
DAY-23

* Priority based scheduling :-
  * Earliest Deadline First scheduling :-

Earliest Deadline First (EDF)

* EDF is one of the best known algorithms. It is a optimal dynamic algorithm. In dynamic priority algorithms, the priority of a task can change during its execution.
  * EDF can also be applied to aperiodic task sets. Its optimality guarantees that the maximal lateness is minimized when EDF is applied.
  * the success of a real-time system depends on whether all the jobs of all the tasks can be guaranteed to complete their execution before their deadlines.
  * Implementation of EDF :- It is really not feasible to implement EDF scheduling.

Task	Arrival	Duration	Deadline
T_1	0	10	33
T_2	4	3	28
T_3	5	10	29



Problems for implementations :-

1. Absolute deadlines change for each new task instance, therefore the priority needs 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 (ie, schedulability)
2. EDF is optimal with respect to minimizing the maximum lateness.

Rate monotonic scheduling :-

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

In RM algorithms, the assigned priority is never modified during runtime of the system.

If a lower priority process is running and a higher priority process becomes available 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 for scheduling periodic tasks where the deadlines of the tasks equal their periods.

Rate monitoring example :-

Task (T)	Period (P)	CPU burst (C)
T_1	2	1
T_2	3	1.01

$$\text{Utilization} = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3}$$

$$= 0.5 + 0.3366 = 0.8366$$

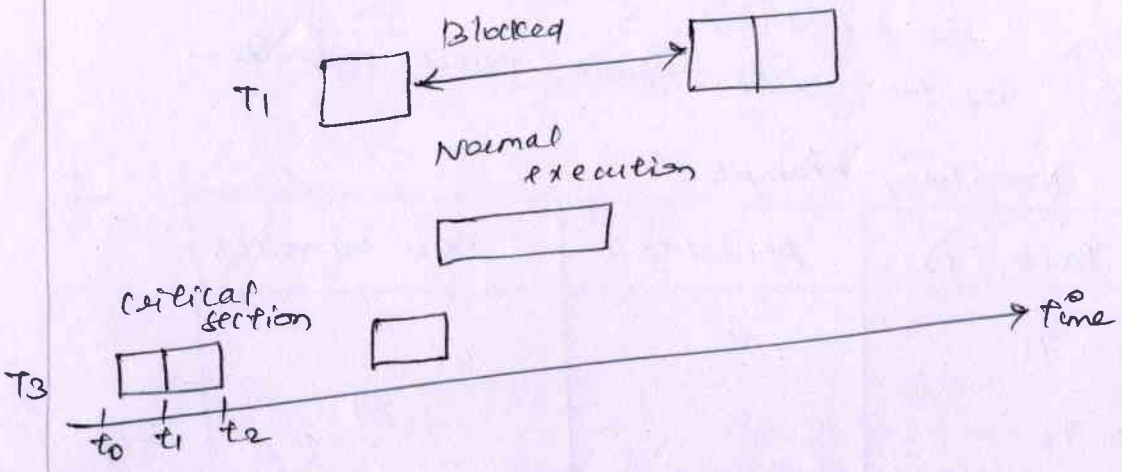
$$U = 83.66\%$$

priority Inversion:-

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

Consider three tasks T_1, T_2 and T_3 with decreasing priorities. Task T_1 and T_3 share some data or resource that requires exclusive access, while T_2 does not interact with either of the other two tasks.

Task T_3 starts at time t_0 and locks semaphore s at time t_1. At time t_2, T_1 arrives and preempts T_3 inside its critical section. After a while, T_1 requests to use the shared resource by attempting to lock s, but it gets blocked, as T_3 is currently using it. Hence at time t_3 continues to execute inside its critical section. Next when T_2 arrives at time t_4, it preempts T_3, it has a higher priority and does not interact with either T_1 or T_3.



Simple solutions:-

1. Make critical sections non-preemptable.
2. Execute critical sections at the highest priority of the task that could use it.
3. 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.

Timing anomalies:-

* Contention for resources can cause timing anomalies due to priority inversion and deadlock. Unless controlled, these anomalies can be arbitrary duration, and can seriously disrupt system timing.

→ It cannot eliminate these anomalies, but several protocols exists to control them:-

1) priority inheritance protocol
2) Basic priority ceiling protocol
3) stack-based priority ceiling protocol.

Wait for graph:-

Wait-for graph is used for representing dynamic-blocking relationship among jobs. In the wait-for graph of a system, every job that require some resource is represented by a vertex labeled by the name of the job.

Wait-for graph is used to model resource contention. Every job holding a resource is represented by a vertex pointing away from the resource and towards the job.

DAY-24

* Interprocess Communication Mechanisms :-

Exchange of data between two or more seperate / independent processes / threads is possible using IPC.

A complex programming environment often uses multiple cooperating processes to perform related operations. The kernel must provide mechanisms that make this possible. These mechanisms are collectively refered to as interprocess communication.

5 types of inter-process communication are as follows :-

1. shared memory permits processes to communication by simply reading and writing to a specified memory location.
2. mapped memory is similar to shared memory, except that it is associated with a file in the file system.
3. Pipes permit sequential communication from one process to a related process.
4. FIFOs are similar to pipes, except that unrelated processes can communicate because the pipe is given a name in the file system.
5. sockets support communication between unrelated processes even on different computers.

Purpose of IPC :-

1) Data transfer
2) sharing data
3) Event modification
4) Resource sharing
5) Process control

IPC is used for two functions!-

1) Synchronization
2) Message passing.

Features of message passing!-

1) Simplicity
2) Uniform semantics
3) Efficiency
4) Reliability
5) Correctness
6) Security
7) Portability.

IPC message format!-

The actual function of message passing is provided in form of a pair of primities!-

1) Send (destination_name, message)
2) Receive (source_name, message).

Send primitive is used for sending a message to destination. process sends information in the form of a message to another process designated by a destination.

Design characteristics of message system for IPC

1. Synchronization between the process
2. Addressing
3. Format of the message.
4. Queueing discipline.

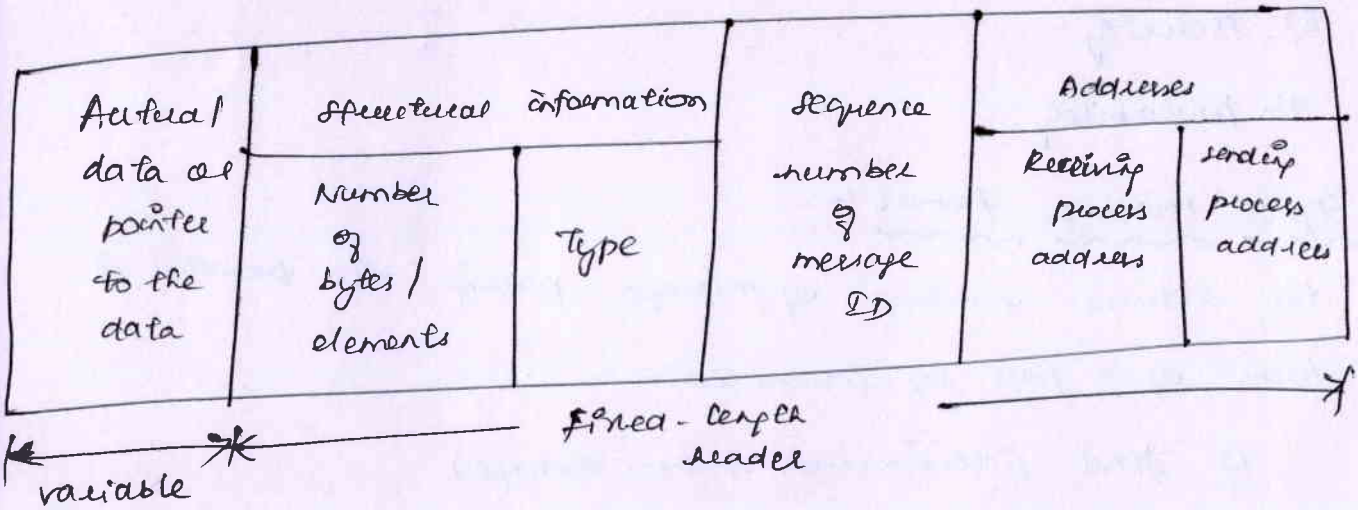
Issues in Ipc by message passing :-

Message is a block of information.

The message block consist of a fixed length header followed by a variable size collection of typed data objects.

The header block of a message may have the following elements :-

1) Address
2) Sequence number
3) Structural information.



message structure.

Some important issues to be considered for the design of an Ipc protocol based message passing system :-

1) The sender's identity
2) The receiver's identity
3) Number of receivers
4) Guaranteed acceptance of sent messages by the receiver
5) Acknowledgement by the sender
6) Handling system crashes or link failures
7) Handling of buffers
8) Order of delivery of messages.

Ipc Synchronization :-

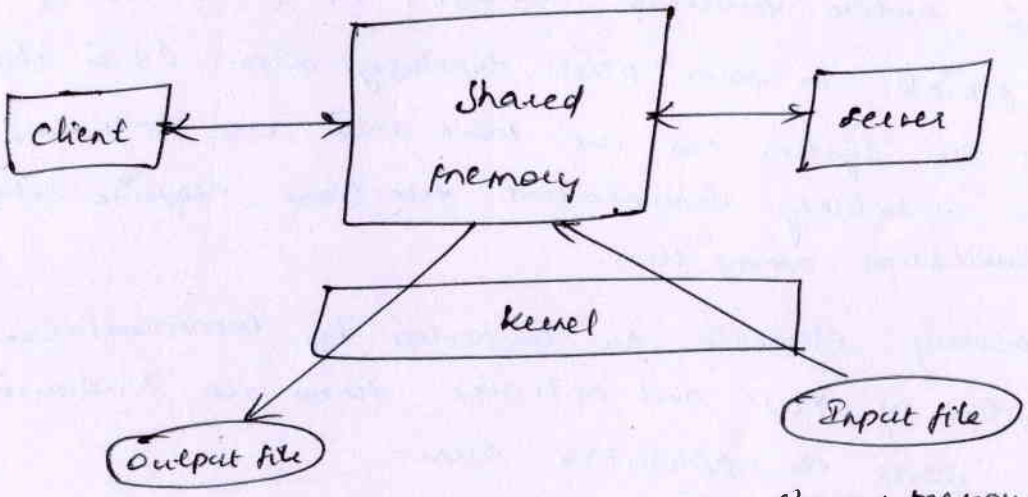
Send operation can be synchronous or asynchronous. Receive operation can be blocking or non-blocking.

Sender and receiver process can be blocking mode or nonblocking mode.

Different possibility of sender and receivers are as follows :-

1) Blocking send, blocking receive.
2) Nonblocking send, block receive.
3) Nonblocking send, Nonblocking receive.

Shared memory :-



Client Server with shared memory

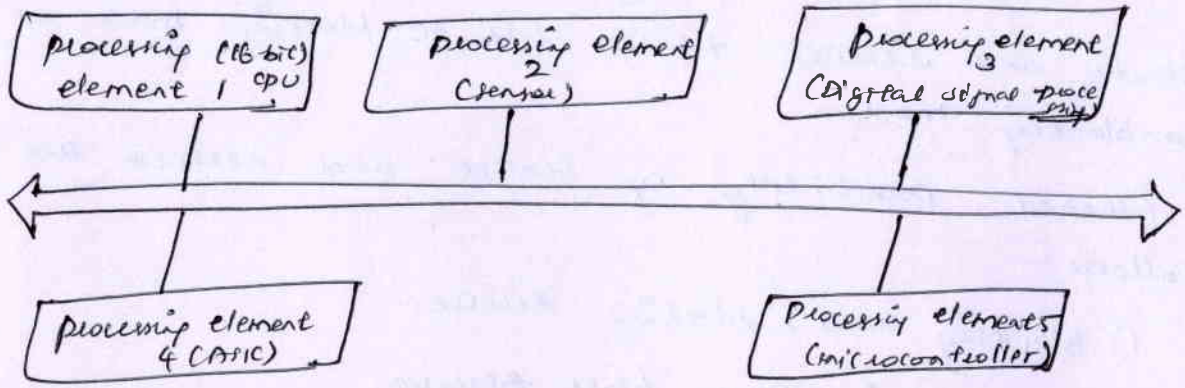
A region of memory that is shared by cooperating processes is established. processes can then exchange information by reading and writing data to the shared region.

Shared memory allows maximum speed and convenience of communication, as it can be done at memory speeds when within a computer.

* Distributed Embedded system:-

Day - 25

In a distributed embedded system, several processing Elements (PEs) are connected by a network that allows them to communication



Processing elements may includes DSP, CPU or microcontroller. Non programmable unit such as the ASICs is also used to implement as PE.

By using this entire processing element, it forms bus topology. It is also possible to form other topology also. It is also possible that the system can use more than one network, such as when relatively independent functions require relatively little communication among them.

All the processing elements are connected by communication link. The system of PEs and networks forms the hardware platform on which the application runs.

Why Distributed:-

→ Higher performance at lower cost

→ physically distributed activities, ie, time constants may not allow transmission to central site.

→ Improved debugging: use on CPU in network to debug other

→ may buy subsystems that have embedded processors.

→ Distributed systems are necessary because the devices that the pts communicate with are physically separated.

Network Abstractions :-

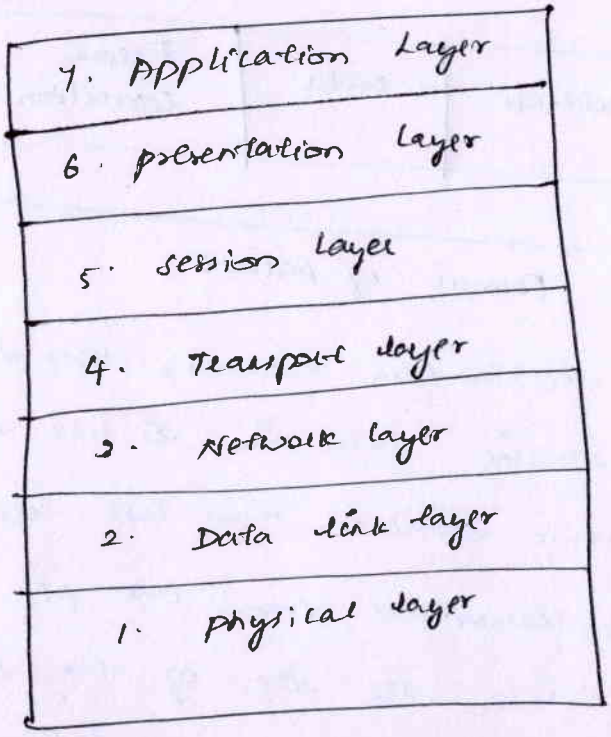
→ Networks are complex systems. They provide high-level services while hiding many of the details of data transmission from the other components in the system.

→ It is based on a common model of network architecture and a suite of protocols used in its implementation.

The International Organization for standardization (ISO) established the open standards Interconnection (OSI) Reference model.

→ Each layer deals with a particular aspect of network communication. There are seven layers in the model, hence the name the 7-layer model.

The model acts as frame of reference in the design of communications and networking products.



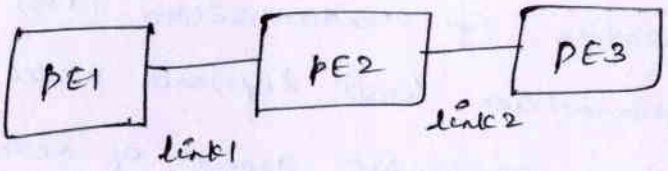
OSI layers

Hardware and software Architecture:-

Distributed embedded systems can be organized in many different ways depending upon the needs of the application and cost constraints.

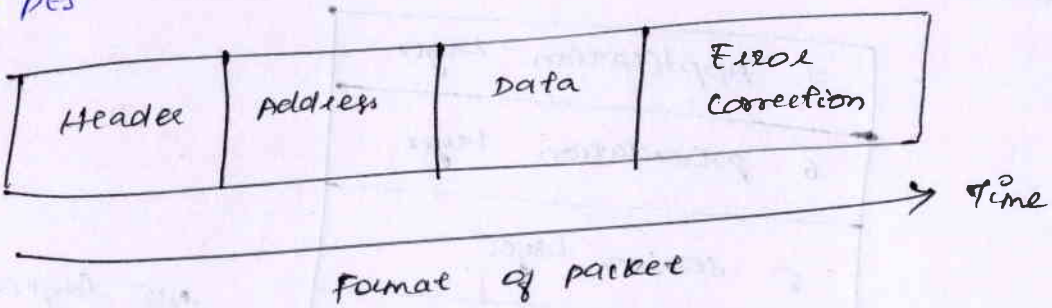
Point to point:-

→ point to point link establishes a connection between exactly two PEs. point to point links are simple to design precisely because they deal with only two components.



Input device sampled the signal and passed to the first digital filter (FI) by using point to point link.

A bus is a more general form of network since it allows multiple device to be connected to it. Like a microprocessor bus, PEs connected to the bus have addresses.



packet contains destination address, user data and error correction codes. sending data size is not exactly fit into packet but processing elements must take care of packet. The data to be transmitted from one PE to another may not fit exactly into the size of the data payload on the packet.

Arbitration scheme:-

The device that is allowed to initiate transfers on the bus at any given time is called the bus master.

1) Fixed-priority arbitration

2) Fair arbitration schemes.

Crossbar network:-

A bus topology provides limited available bandwidth since all devices connect to the bus, communications can interfere with each other. For reducing communication conflicts, other network topology can be used.

Message passing programming:-

A message passing system is a subsystem of distributed operating system that provides a set of message-based IPC protocols and does so by shielding the details of complex network protocols and multiple heterogeneous platforms from programmers.

→ It enables processes to communicate by exchanging messages and allows programs to be written by using simple communication primitives, such as send and receive.

→ Transport layer provides message-based programming interface:-

```
send_msg (addr, data1);
```

→ Data must be broken into packets at source, reassembled at destination

→ Data push programming: Receivers respond to new data.

DAY-26

* Mpsocs and shared memory multiprocessors:-

single processors may be sufficient for low-performance applications that are typical of early microcontrollers, but an increasing number of applications require multiprocessors to meet their performance goals.

Multiprocessor systems-on-chips (Mpsoc) are one of the key applications of today. Mpsoc are increasingly used to build complex integrated system.

Definition:- multiprocessor is parallel processors with a single shared address.

parallel processing program:- is a single program that runs on multiple processors simultaneously.

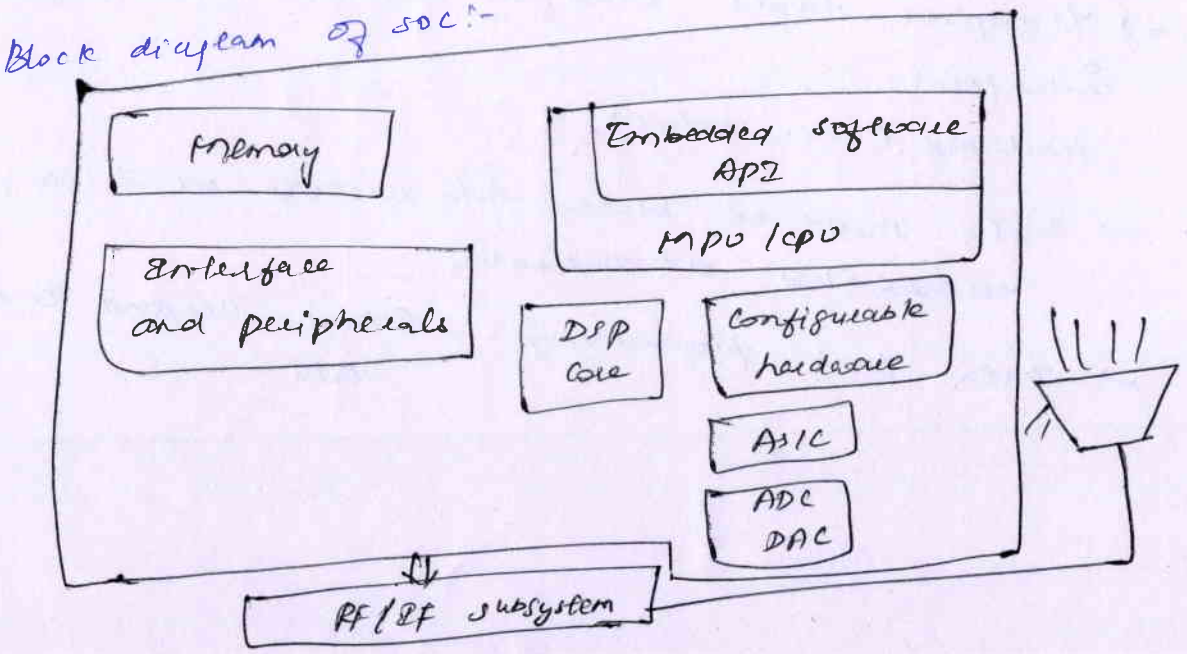
cluster is a set of computers connected over a local Area Network (LAN) that function as a single large multiprocessor.

Shared memory is a memory for a parallel processor with a single address space, implying implicit communication with loads and stores.

Mpsocs often requires large amount of memory.

The typical Mpsoc is a heterogeneous multiprocessor.

Block diagram of SOC:-



1) Memory Controller :- Interfaces with the onboard RAM.

2) DMA :- Handles the automated transfers of data between the RAM and memory-mapped hardware.

3) USB Controller :- Manages the hardware side of the device's USB connections.

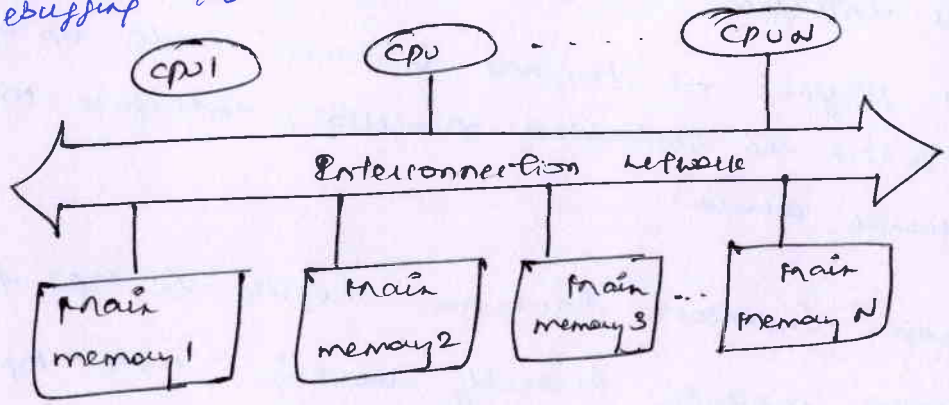
4) DSP Core :- It provides hardware acceleration for some signal processing, such as JPEG encoding.

5) Camera :- Allows the SoC to interface with a camera.

6) Display :- Enables the SoC to drive various display types.

7) Storage :- Manages I/O with the various types of storage that can be used with the SoC.

8) Debug :- Enables the SoC to be connected to hardware debugging tools through various mechanisms, such as JTAG.

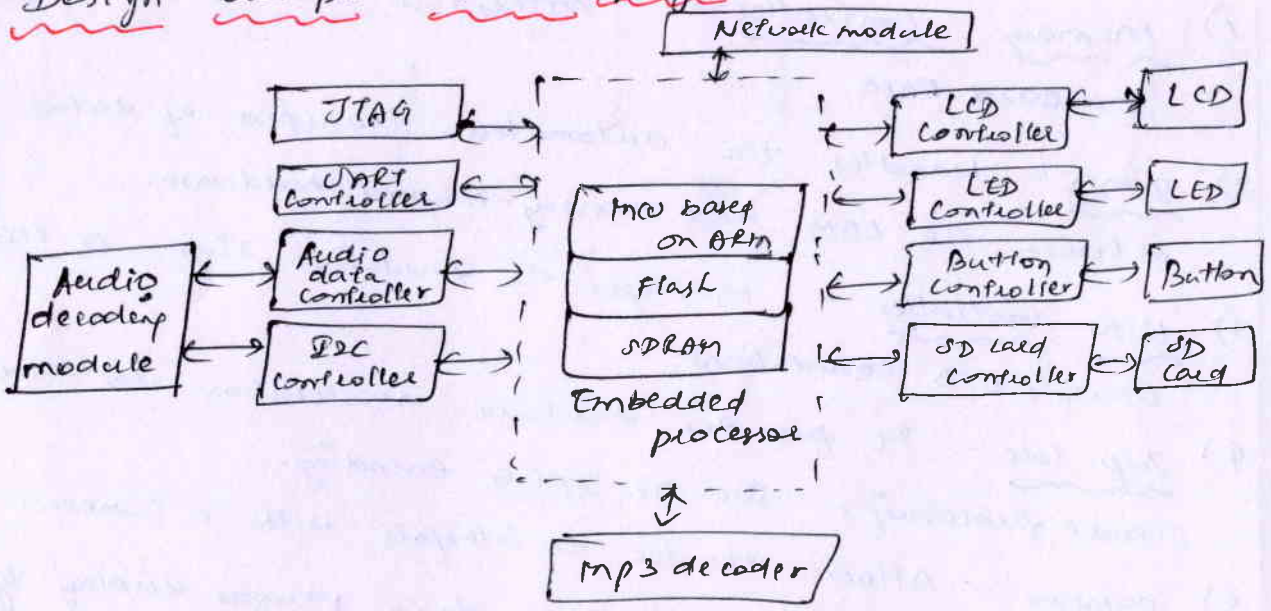


Shared memory

A shared-memory model is often preferred because it makes life simpler for the programmer.

* Signal address
* Message passing
* Heterogeneous memory systems
* Irregular memory structures.

Design Example - Audio player.



Hardware design of mp3 player

Audio players are often called mp3 players after the popular audio data format. An mp3 player performs three basic functions :- Audio storage, audio decompression and user interface.

The mp3 player we designed is main made up of four parts, which are embedded processor, interface module, storage and external devices.

The most important hardware module in mp3 player is the decoder module. Generally speaking, most mp3 players use DSP core (or) dual core which is comprised by DSP and RISC to solve the complexity of application algorithms.

RISC core can be used as both audio decoding unit and control unit, it is useful for resource constrained system on chip design.

* **Engine Control Unit:-**

Engine control unit (ECU) is the central controller and heart of the engine management system. It controls the fuel supply, air management, fuel injection and ignition.

Each ECU typically contains a dedicated chip that runs its own software or firmware and requires power and data connections to operate.

An ECU receives inputs from different parts of the vehicle, depending on its function.

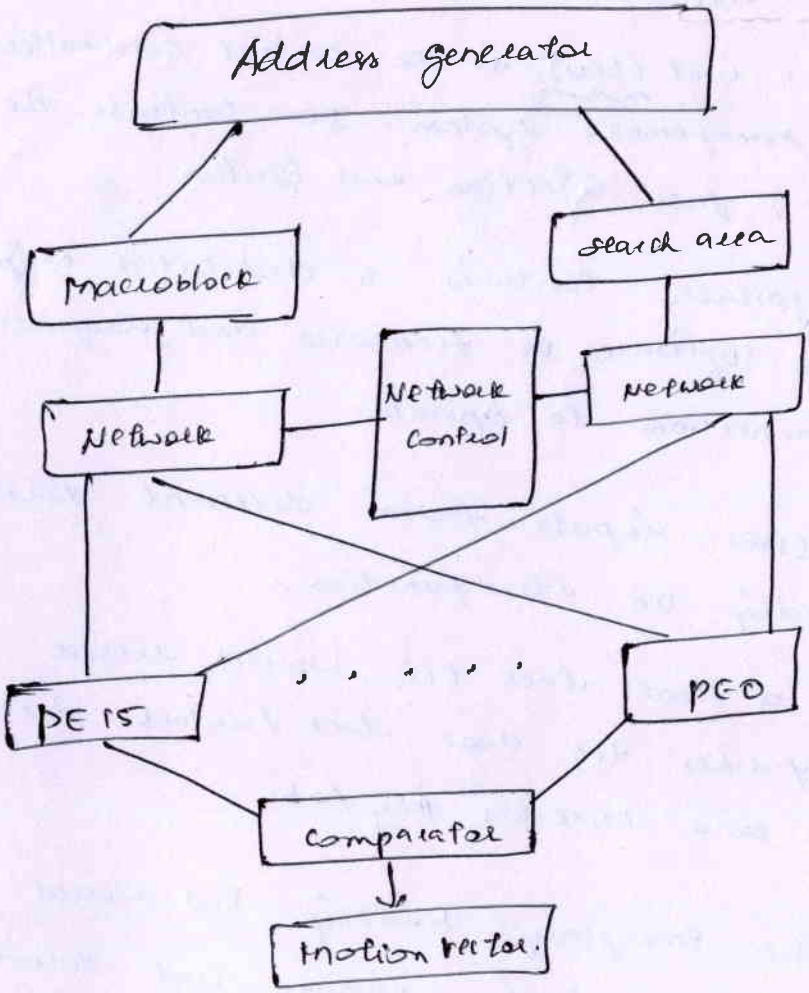
For example, a door lock ECU would receive input when a passenger pushes the door lock/unlock button on a car door or on a wireless key fob.

An automatic emergency braking ECU would receive inputs from forward-facing radars that detect when the vehicle is approaching an obstacle too quickly.

The ECU would then communicate to actuators to perform an action based on the inputs. The automatic emergency braking ECU would engage the brakes to prevent a collision.

* **Video accelerators:-**

Block motion estimation is used in digital video compression algorithms so that one frame in the video can be described in terms of the differences between it and another frame



Architecture for the motion estimation accelerator.

Frame is selected as a reference and subsequent frames are predicted from the reference. The 60% computational complexity at encoder side.

The process of video compression using motion estimation is also known as inter-frame coding.

A second compression technique is used, known as intra-frame coding.

The machine consists of two memory :- macroblock memory and search memory.

It has 16 PEs that perform the difference calculation on a pair of pixels, the comparator sums them up and selects the best value to find the motor vector.

Depending on number of different motion estimation algorithms, the networks connecting the memories to the PEs may also be simplified.

Unit-IV

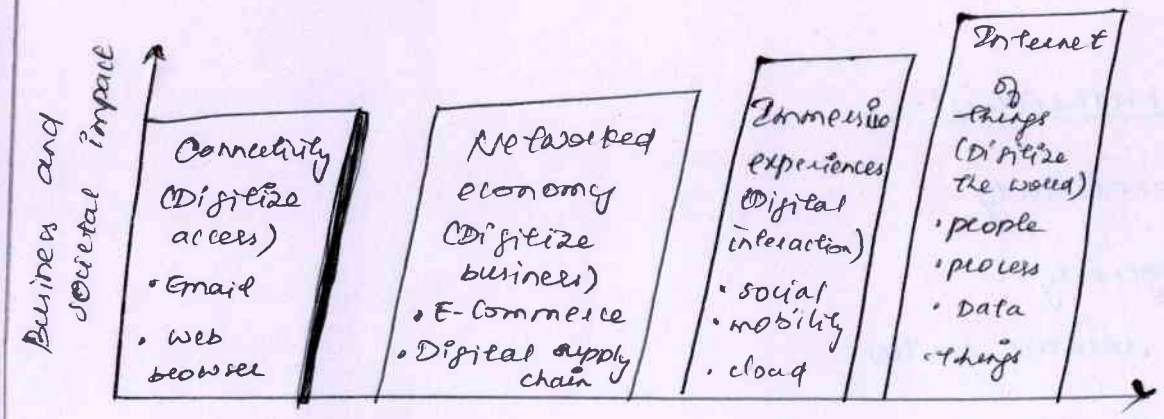
IoT Architecture and protocols.

DOJ- 29

* Introduction of IoT :-

The Internet of things (IoT) refers to the capability of everyday devices and people through the existing Internet Infrastructure. Devices connect and communicate in many ways.

Examples :- smartphones.



Evolutionary phase of internet.

Evolutionary phase of internet is connectivity, Networked Economy, Immersive experiences and IoT.

1. Connectivity :- In the phase, peoples are connected to email, Web services and searches the information.

2. Networked Economy :- this phase support e-commerce and supply chain enhancements along with collaborative engagement to drive increased efficiency in business processes.

3. Immersive Experiences :- this phase extended the internet experience to encompass widespread video and social media while always being connected through mobility.

4. Internet of things :- It adds connectivity to objects and machines in the world around us to enable new services and experiences.

Definition of IoT:-

The Internet of things (IoT) is the network of physical objects ie, devices, vehicles, buildings and other items embedded with electronics, software, sensors and network connectivity that enables these objects to collect and exchange data.

IoT characteristics:-

1) Interconnectivity
2) Heterogeneity
3) Things-related services
4) Dynamic changes
5) Integrated into information network
6) self-adapting
7) self-configuration.

Component of IoT:-

Major components of IoT devices are as follows:-

1) control units
2) sensor
   a) Temperature sensor
   b) Image sensors
   c) Light sensors
   d) Micro flow sensors
   e) Gas RFID sensors

3) Communication modules

4) power sources.

Working of IOT :-

1) collect and transmit data :- The device can sense the environment and collect information related to it and transmit it to a different device or to the internet.

2) Actuate device based on triggers :- It can be programmed to actuate other devices based on conditions set by user.

3) Receive information :- Device can also receive information from the network.

4) Communication assistance :- It provides communication between two devices of same network or different network.

Advantages of IOT :-

1) Improved customer engagement and communication

2) support for technology optimization

3) support wide range of data collection

4) Reduced waste.

Disadvantages of IOT :-

1) Loss of privacy and security

2) Flexibility

3) Complexity

4) Compatibility

5) Safe time

Applications :-

1) Home
2) offices
3) factories
4) vehicles
5) cities
6) Worksites.

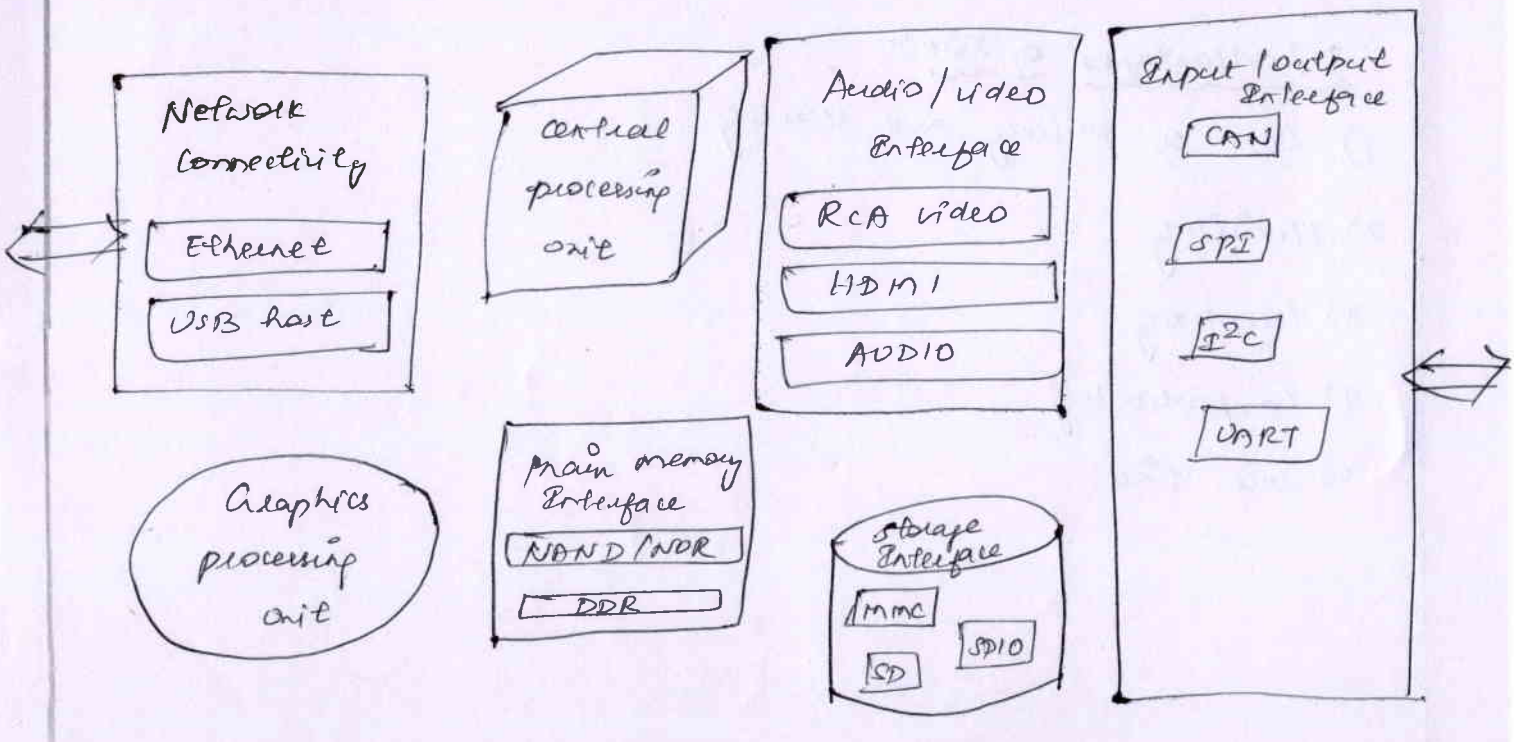
---

* Physical Design :-

Physical Design of IoT system refers to IoT devices and IoT protocols.

Things in IOT :-

IoT devices have unique identity and they are refer as "things" in IoT. Device can perform remote sensing, actuating and monitoring. IoT devices can exchange data between them and proces data or send to centralized location for processing and storage.



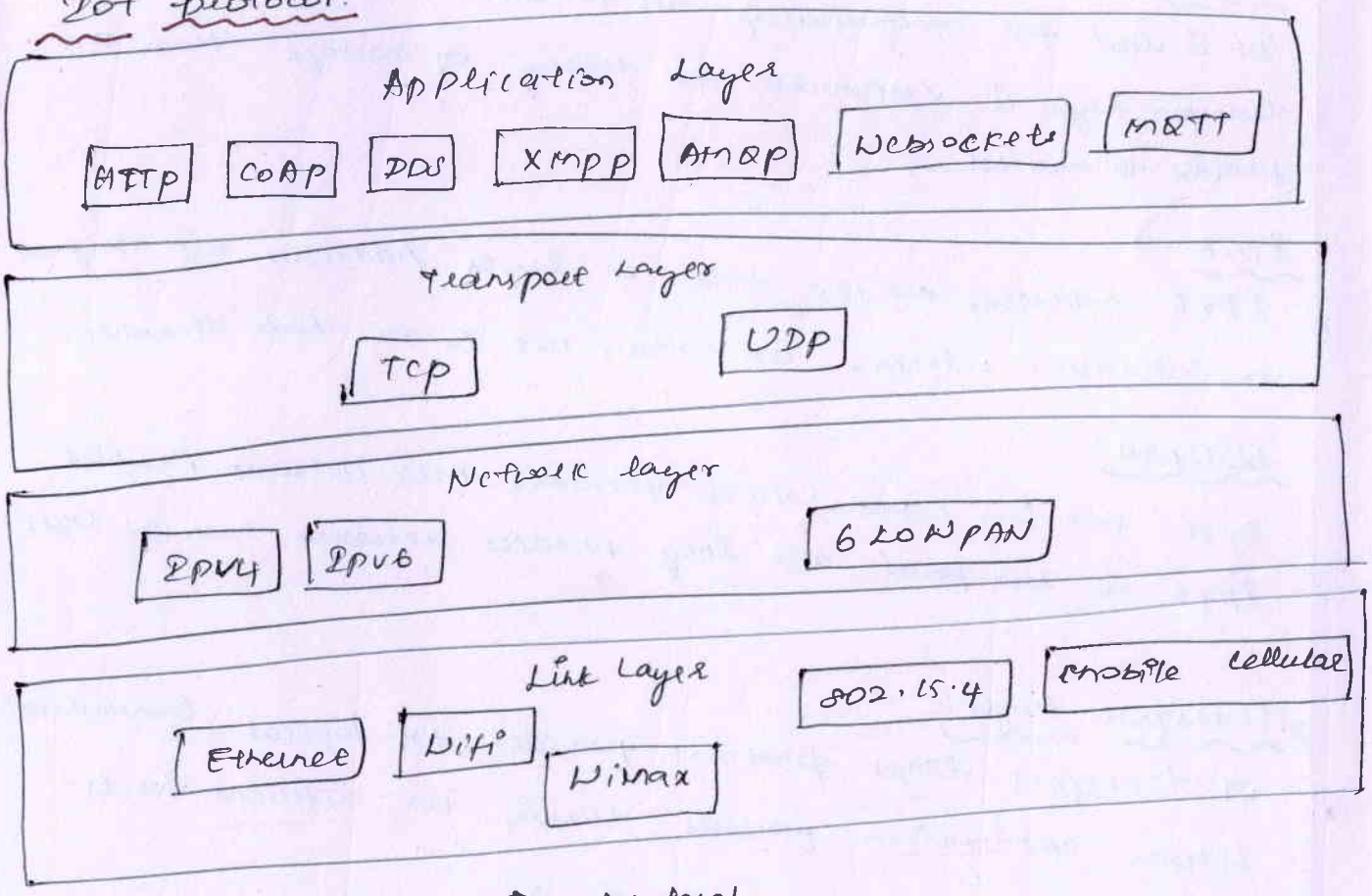
Block diagram of IOT devices

+ IoT devices provide interface to various wire and wireless devices. Interface includes memory interface, I/O interface for sensors, Internet connectivity interface, storage interface, etc.

+ Using sensors, IoT collects various information like temperature, light intensity, humidity, air pressure.

+ various types of IoT devices are smart clothing, smart watch, wearable sensors, LED lights, automobile industry, etc.

IoT protocol:-



IoT protocol

1. Link Layer:-

Link layer protocols decide how data is sent on physical medium. Link layer works on the local area network. protocol of link layer is explained below:-

a) 802.3 Ethernet.

b) 802.11 Wifi

c) 802.16 Wimax

d) 802.15.4 Zigbee

e) Mobile Communication (2G/3G/4G)

2. Network layer :-

The network layer is responsible for the delivery of packets from the source to destination.

IPV4 :-

It is used for communicating all internet enabled devices. The transport layer is responsible for delivery of message from one process to another.

IPV6 :-

IPV6 addresses are 128 bits in length. Addresses are assigned to individual interfaces on nodes, not to the node themselves.

6LOWPAN :-

IPV6 over low power wireless personal Area Network Enables IPV6 is low-power and lossy wireless networks such as WSNs.

3. Transport layer :-

A transport layer protocol provides for logical communication between application processes running on different hosts.

TCP (Transmission control protocol):

⇒ TCP provides a connection-oriented, reliable, byte stream service. The term connection oriented means the two applications using TCP must establish a TCP connection with each other before they can exchange data.

d) MQTT (Message Queue telemetry Transport)

MQTT is open connectivity for mobile, M2M and IoT.

MQTT is a lightweight broker-based publish/subscribe messaging protocol designed to be open, simple, lightweight and easy to implement.

e) XMPP (Extensible Messaging Presence Protocol)

The XMPP is targeted at delivering instant messages and presence information.

It is an open and XML-Based protocol.

f. DDS (Data Distribution Service)

DDS is an object management Group (OMG) standards for pub/sub that addresses the needs of mission and business

DDS provides a shared "global data space".

g. AMQP (Advanced message Queuing protocol)

A protocol to communicate between clients and messaging middleware servers (brokers). The Broker is the AMQP server.

---

* Logical Design:-

DAY-30

IoT Functional Blocks:-

The functional model (FM) is derived from external and external requirements. Functional view is derived from the functional model in conjuction with high-level requirements.

the Application, Virtual Entity, IoT service, and Device Fns

UDP (User Datagram protocol)

UDP is a simple, datagram-oriented, transport layer protocol. This protocol is used in place of TCP.

4. Application layer:-

Application layer is responsible for accessing the network by user. It provides user interfaces and other supporting services such as e-mail, remote file access, file transfer, sharing database.

a) HTTP (Hyper Text Transport protocol)

HTTP is an application protocol, HTTP is used to retrieve web pages from remote servers.

HTTP uses the services of TCP.

HTTP is a stateless protocol.

b) CoAP - Constrained Application protocol:-

* CoAP is a specialized web transfer protocol for use with constrained nodes and constrained networks.

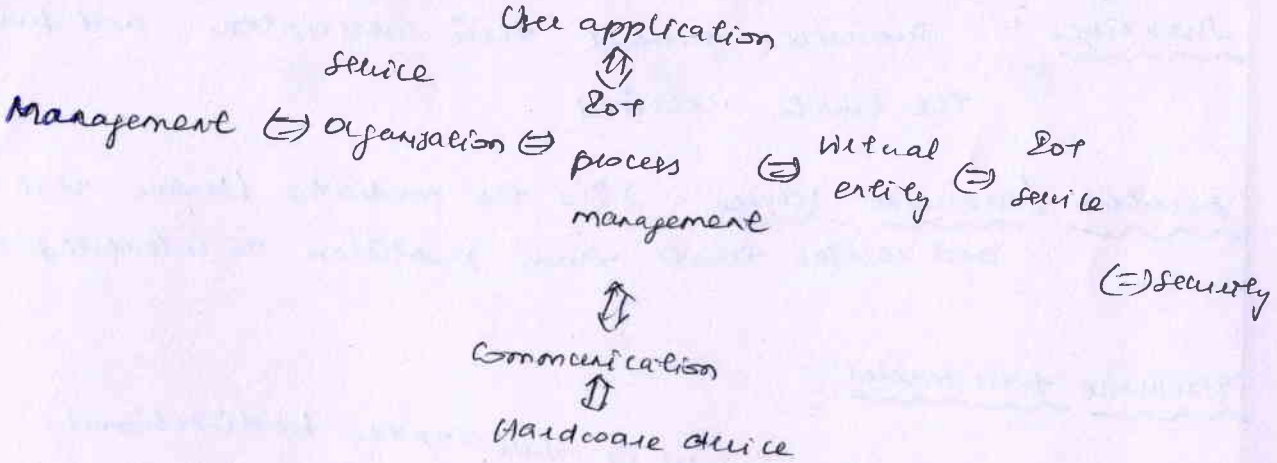
* CoAP is designed for simplicity, low overhead and multicast support in resource-constrained environments.

c) Websocket:-

Websocket is a communications protocol, providing full-duplex communication channels over a single TCP connection.

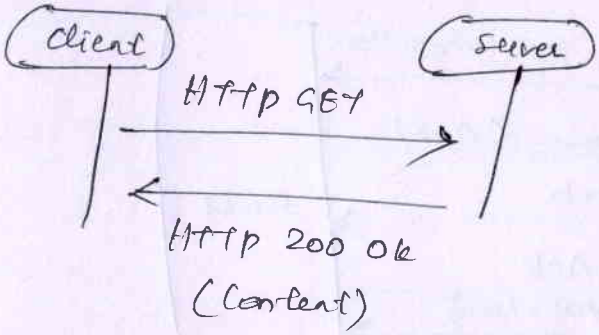
After connection is established, messages can be transmitted either client or server initiated. This means that you can make a dynamic web page where changes occur in real time

are generated by starting from the User, Virtual Entity, Resource, Service and Device class from the IoT Domain model.



IoT functional model.

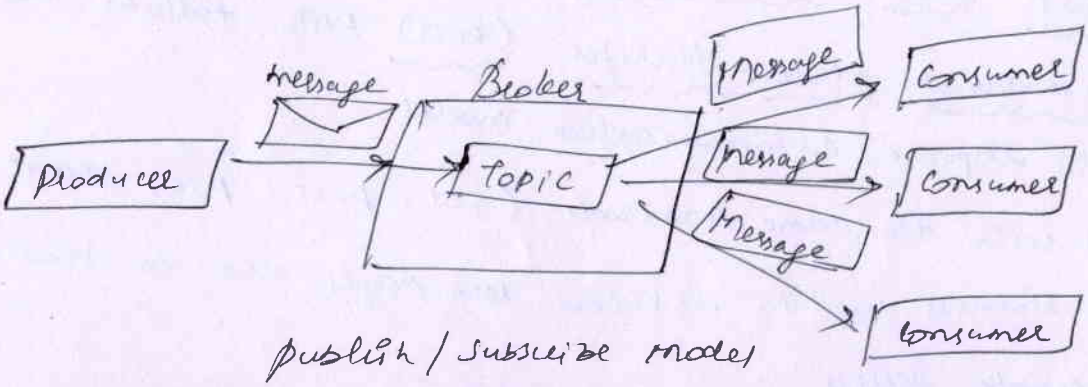
IoT communication model:-



In the request/response model, client requests information from the server and waits till the response is served from the server.

HTTP protocol is used by Request/Response model.

Publish/subscribe model:-



Publish/subscribe model

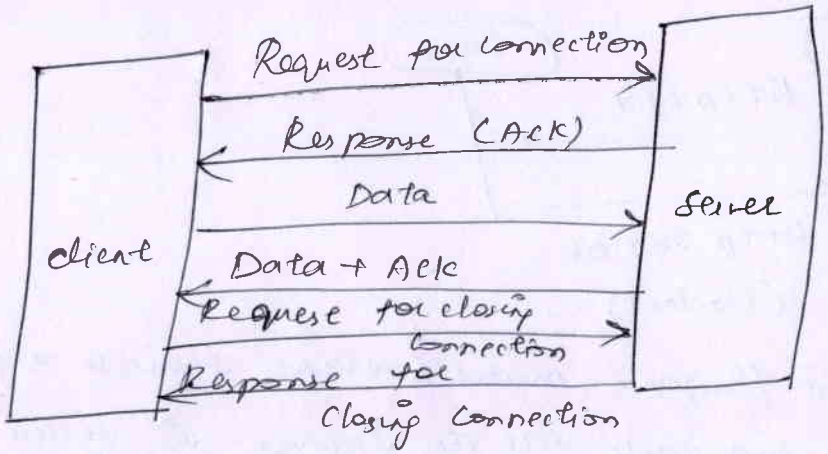
Publishers :- publishers generate event data and publishes them.

Subscribes :- Subscribes submit their subscription and process the events received.

publish /subscribe service :- It's the mediator /broker that filters and routes events from publishers to interested subscribers

Exclusive pair model :-

This communication model is full-duplex, bi-directional communication model. It uses persistent connection between client and server.



Exclusive pair model.

IOT communication API's :-

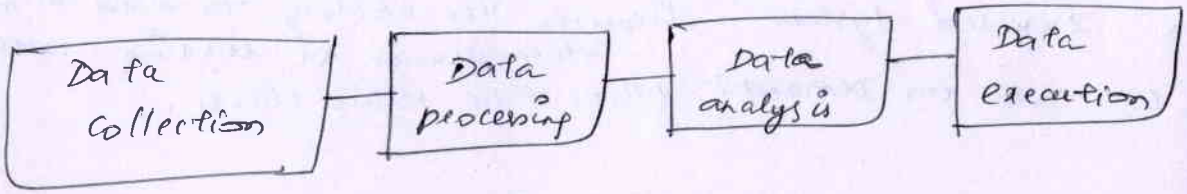
1. REST based communication API :-

Representational state Transfer (REST) APIs follow the request-response communication model.

HTTP with the same methods (GET, POST, PUT, DELETE, etc) that browsers use to retrieve web pages and to send data to remote servers.

Big data Analytic:-

A category of technologies and services where the capabilities provided to collect, store, search, share, analyze and visualize data.



Wireless sensor Network:-

A WSN is a network formed by a large number of sensor nodes where each node is equipped with a sensor to detect physical phenomena such as light, heat, pressure, etc.

Communication protocols:-

- Communication protocols are used as a backbone of IoT systems.
- Communication protocol also performs error collection and detection, flow control, etc.

Embedded System:-

A system is a set of interacting or interdependent component parts forming a complex unit.

Embedded system is an electronic system which is designed to perform one or a limited set of functions using software and hardware.

1. Client - server : Requires that a service offer one or more operations & that services wait for clients to request these operations.
2. Stateless : Requires communication between service consumer & service provider.
3. Cache : Requires responses to be clearly labeled as cacheable.
4. Uniform Interface : Requires all service providers and consumers.
5. Layered system : Requires the ability to add or move intermediaries at runtime without disrupting.
6. Code - on- Demand :- Allows logic within clients.

Websocket based communicate API :-

- Websocket support full - duplex, two-way communication between client and server.

* IoT Enabling Technologies :- DAY-31

IoT is enabled by several technologies including wireless sensor networks, cloud computing, Big data analytics, etc.

Cloud computing :-

Cloud computing is a pay - per-use model for enabling available, convenient, on-demand network access to a shared pool of configurable computing resources.

(SaaS) Software as a service :- Model in which an application is hosted as a service to customers who access it via the internet.

(PaaS) platform as a service :- Platform as a service is another application delivery model and also known as cloud - ware.

Unmanned Aerial vehicle :-

It is popularly known as Drone, is an airborne system or an aircraft operated remotely by a human operator or autonomously by an onboard computer.

It has 3 components :-

1. An autonomous or human operated control system which is usually on the ground or a ship but may be on another airborne platform.
2. An unmanned Aerial Vehicle (UAV)
3. A Command and Control (C2) system - sometimes referred to as a Communication, Command and Control (C3) system, to link the two.

---

* Domain specific IoTs :-

Day-32

IoT-enabled products like smart appliances, smart vending machine, smart lighting and smart payments system offer more security, privacy and energy efficiency to the user.

Inventory management in Retail :-

Retail involves the sale of goods from a single point directly to the consumer in small quantities for the end use.

Retail is a challenging business but the pressures of today's economic condition are resulting in even more selective consumer shopping and spending.

* this process involves huge transaction of monetary resources and hence it is imperious that a high preference is given to this in a BPM.

This can considerably improve supply chain efficiencies and enable leaner inventories.

Large retailers such as walmart are already using IOT for supply chain and inventory management.

Tracking is done using RFID readers attached to the retail store shelves.

Smart payments in Retail:-

Smart payment system uses Near Field communication (NFC) and Bluetooth communication.

Near field communication (NFC) technology is a standards-based wireless communication technology that allows data to be exchanged between devices that are a few centimeters apart.

NFC operates at 13.56 MHz and transfers data at up to 424 kbits/second.

Data transfer using smart devices are possible using NFC technology like Android beam. Two users can share documents, photos, resumes and business cards by just waving their smart phone.

Smart vending machine :-

Smart vending solution offers its customers flexible payment options and monitors the machines remotely and in real time.

Smart vending machine provide following :-

1. Achieve high levels of efficiency in the management of their assets.
2. offers its customers flexible payment options :- RFID / NFC card, mobile payments, Debit / credit cards.
3. Monitor the machines remotely and in real time.
4. Simplifies business since the vending machines contain multiple sensors that alert the owners about their location, the state inventory and eventual maintenance issues.

Route Generation and scheduling in Logistics :-

Route generation and scheduling system can generate end to end routes using combination of route patterns and transportation modes.

The Internet of vehicles (IoV) is an integration of 3 networks :-

1) An inter-vehicle networks
2) An intra-vehicle network
3) Vehicular mobile Internet.

Fleet tracking in logistics:-

It is automated vehicle routing and scheduling.

It supports driver compliance, safety and performance reporting.

Fleet maintenance and fuel consumption capabilities.

Track, schedule and route vehicles in real time.

Benefits:-

1) Accelerate delivery and dispatch rates
2) Improve customer satisfication
3) Reduce fuel consumption and vehicle maintenance costs.
4) Ensure compliance with government and industry regulations.
5) Improve fleet productivity, uptime and safety.

* IoT and M2M:- DAY-33

Machine to machine (M2M) communication is the communication among the physical things which do not need human intervention.

M2M communication is a form of data communication that involves one or more entities that do not necessarily require human interaction or intervention in the process of communication. M2M is also named as Machine Type communication (MTC) in 3GPP.

Reasons for shifting from M2M to IoT:-

1. It supports multiple application with multiple device.
2. It is information and service centric.
3. It supports open market place.
4. IoT uses horizontal enabler approach.
5. It requires generic commodity devices.
6. Used in B2B and B2C.

Key features of M2M:-

1) Low mobility
2) Time controlled
3) Time Tolerant
4) packet switched
5) Online small data transmissions
6) Low power consumptions
7) Location specific trigger.

Architecture and components of M2M:-

1. M2M Device:- A device that runs application(s) using M2M capabilities and network domain functions.
2. M2M Area network:- A M2M area network provides connectivity between M2M devices and M2M gateways.
3. M2M gateways:- Equipments using M2M capabilities to ensure M2M devices interworking & interconnection to the network and application domain.

4. M2M application server :-
Applications that run the service logic and use service capabilities accessible via open interfaces.

3. M2M application :- The application component of the solution is a realization of the highly specific monitor and control process. The application is further integrated into overall business process system.

* IoT system management with NETCONF - YANG :-

DAY-34

Need for IoT systems management :-

a. Automating configuration
b. Monitoring operational and statistical data
c. Improved Reliability
d. System wide configurations
e. Multiple system configurations
f. Retrieving and Reusing configurations.

Simple Network management protocol (SNMP)

Simple Network management protocol (SNMP) is an application-layer protocol used to manage and monitor network devices and their functions.

SNMP has a simple architecture based on a client-server model.

Steeper of SNMP:-

1. It is simple to implement.
2. Agents are widely implement.
3. Agent level overhead is minimal.
4. It is robust and extensible.

Limitations:-

1. It is too simple and does not scale well.
2. There is no object oriented data view.
3. It has no standard control definition.

NETCONF:-

Network configuration protocol (NETCONF) is a session-based network management protocol. NETCONF allows retrieving state or configuration data and manipulating configuration data on network devices.

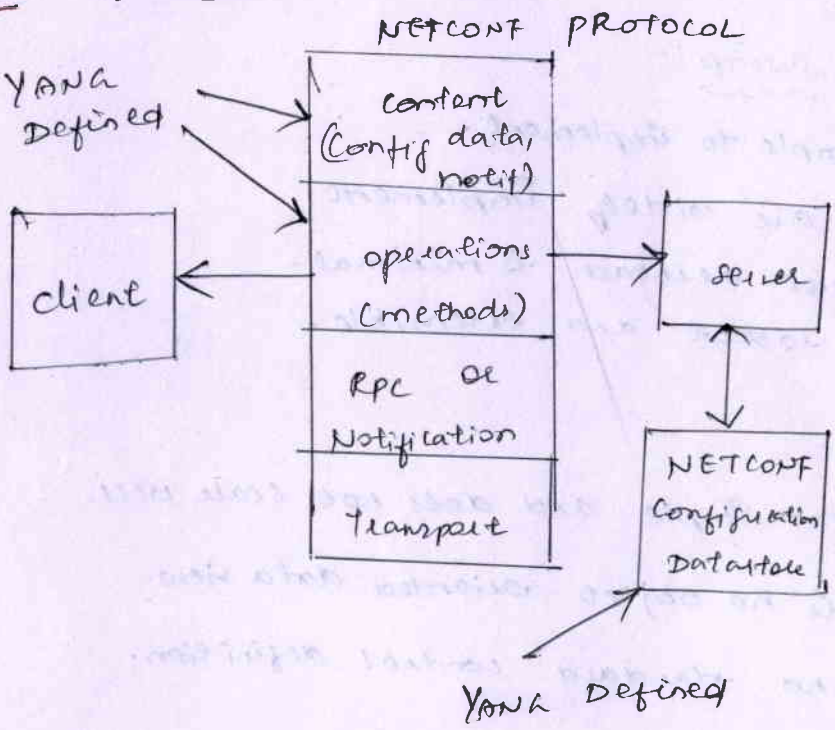
NETCONF is defined for transaction-safe configuration of devices.

It uses an Extensible Markup Language (XML) based data encoding for the configuration data as well as the protocol messages.

* It uses secure shell (SSH) as the transport layer across network devices.

* NETCONF provides various operations to retrieve and edit configuration data from network devices.

NETCONF PROTOCOL LAYERS :-



YANG :-

YANG is a data modeling language used to model configuration and state data manipulated by the NETCONF protocol.

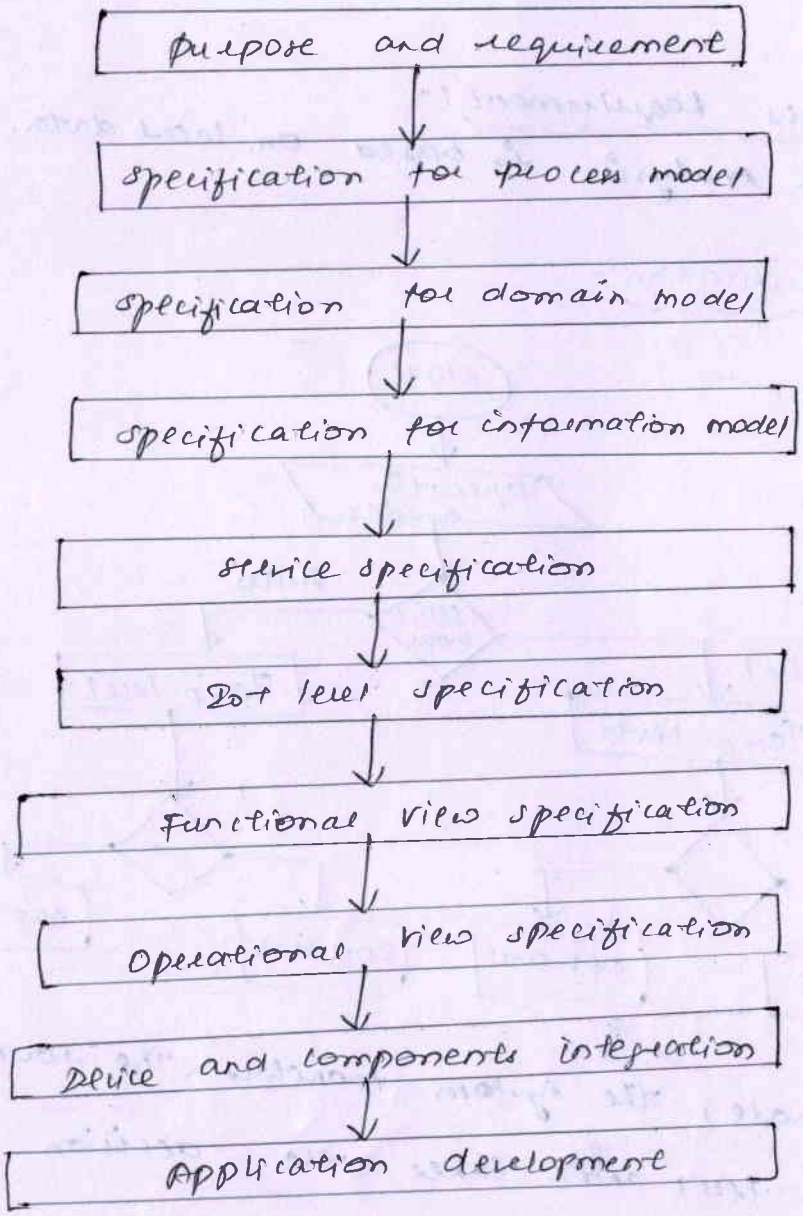
YANG is used to model both configuration and state data of network elements.

YANG modules defines the data exchanged between the NETCONF clients and server. A module comprises of a number of 'leaf' nodes which are organized into a hierarchical tree structure.

DAY-35

* IoT platform Design methodology:-

An IoT platform facilities communication, data flow, device management, and the functionality of applications. The goal is to build IoT applications within an IoT platform framework. The IoT platform allows applications to connect machines, devices, applications, and people to data and control centres.



Design methodology steps

Purpose and Requirement specification :-

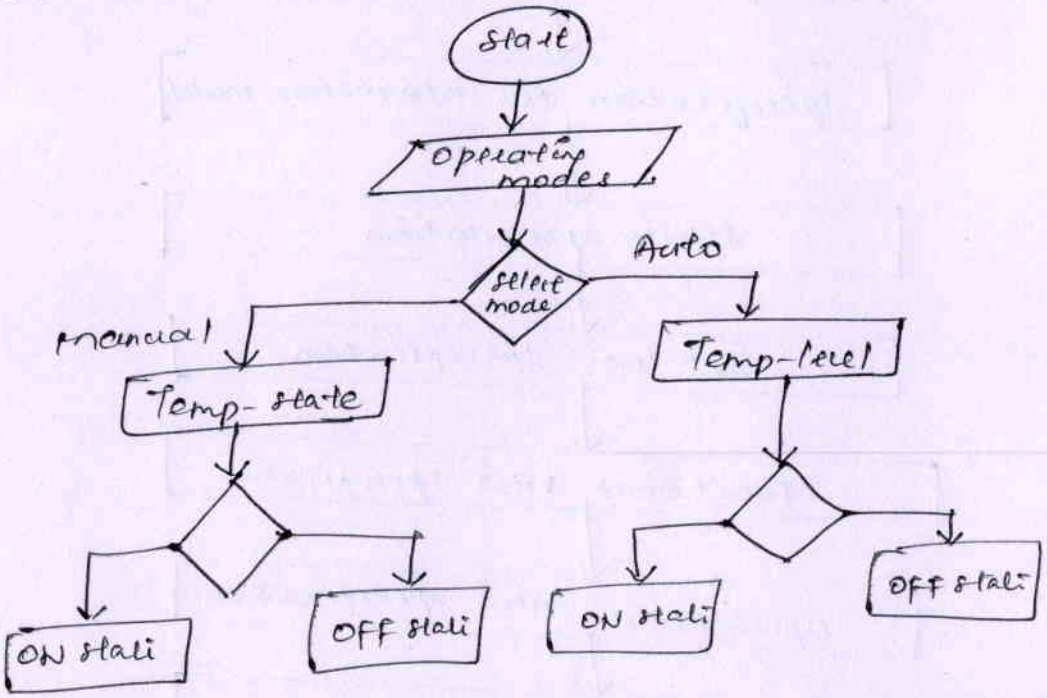
Purpose :- Using web application, user can remotely control the home electronic devices.

Behaviour :- Home automation works on two modes :- 1. manual and 2. Automatic mode.

System management Requirement :-
Remote monitoring and control function is provided by system.

Data Analysis Requirement :-
Data Analysis is based on local data.

Process specification :-

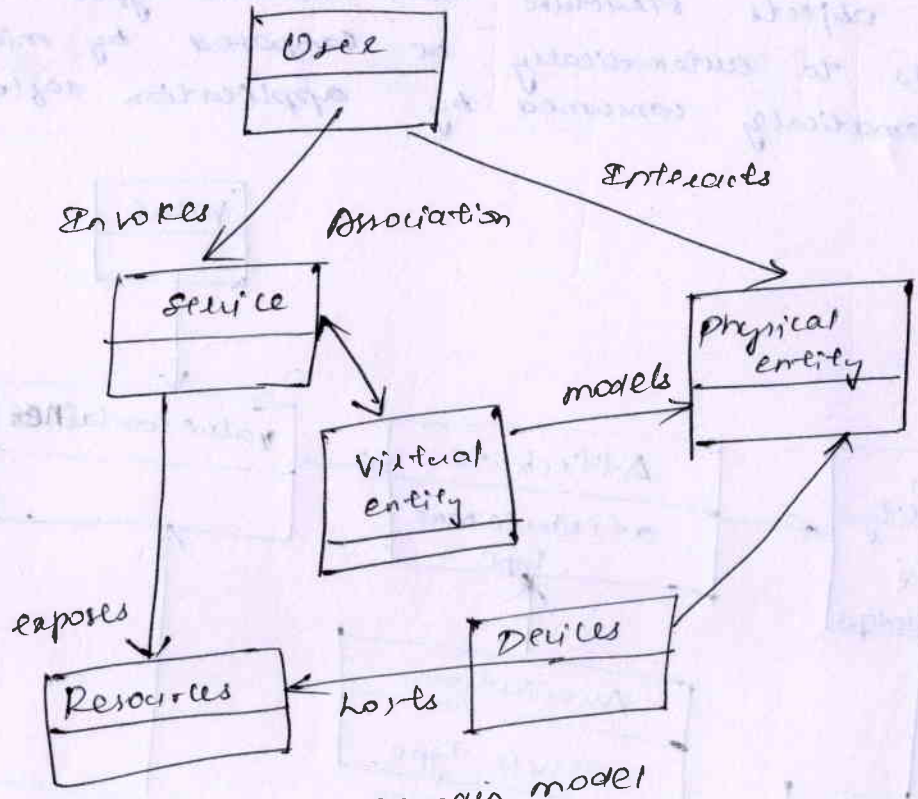


In auto mode, the system monitors the room temperature and light level and takes the decision for switching ON/OFF

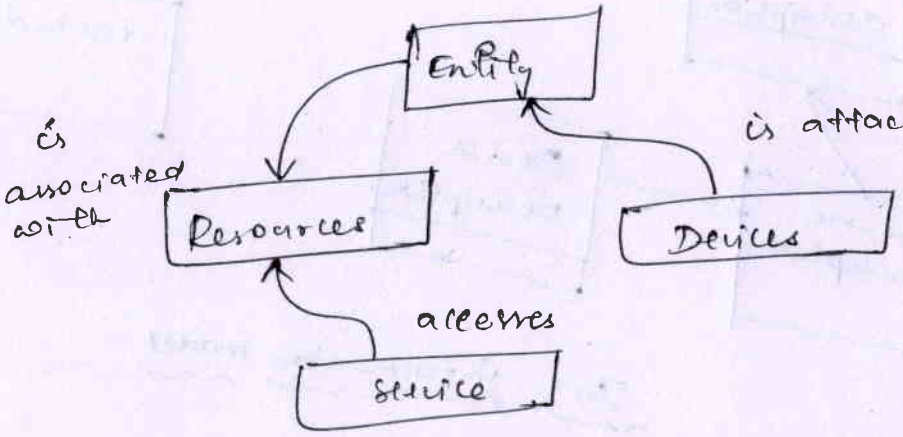
In manual, user performs the operation.

* Domain Model Specifications :-

DO4-36



IoT Domain model



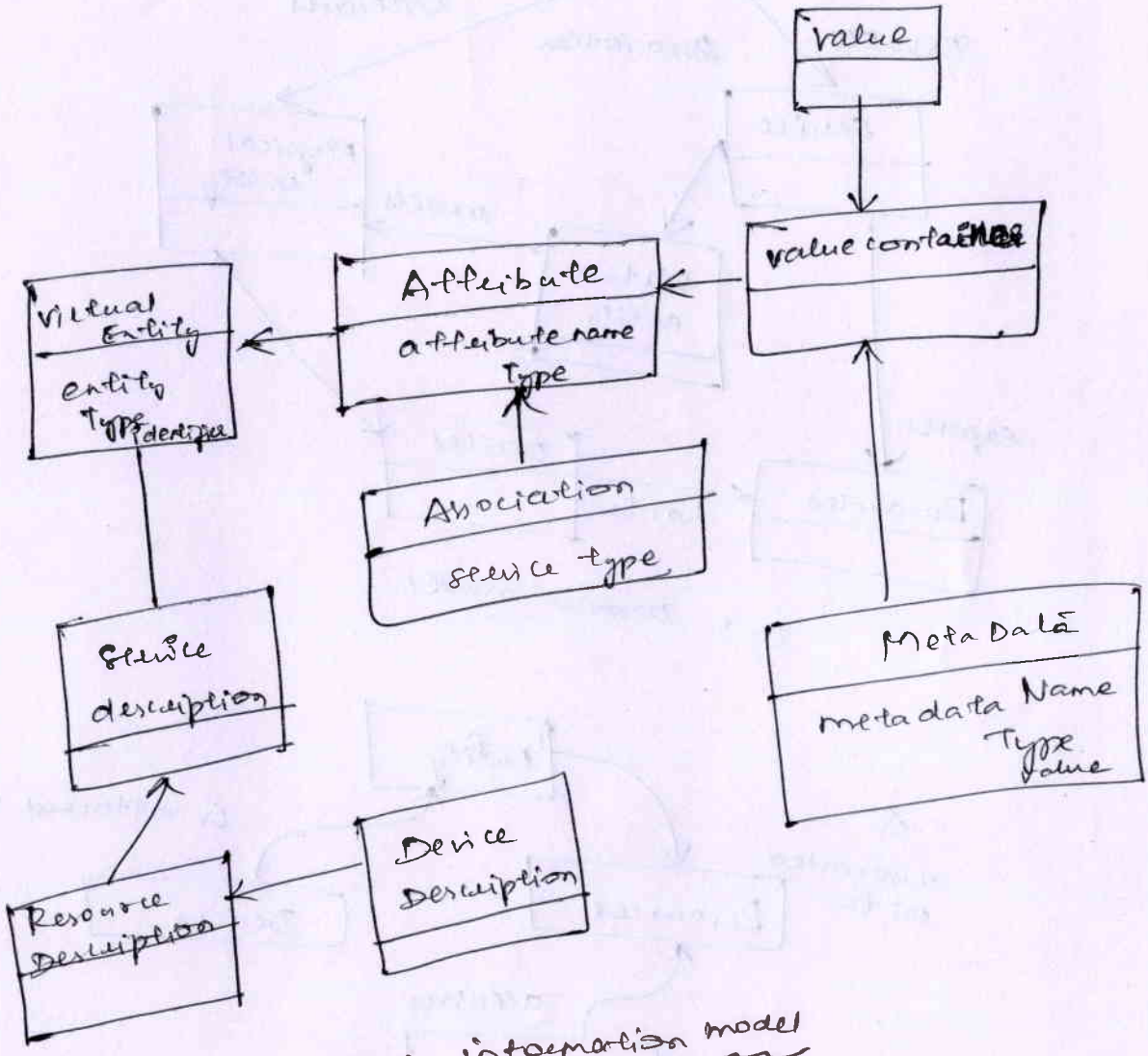
Key concepts & interaction in IoT model.

The relations between services and entities are modeled as specifications. One physical entity can be represented by multiple virtual entities, each services a different purpose.

1) Sensors
2) Actuators
3) Tags

Information model specification:

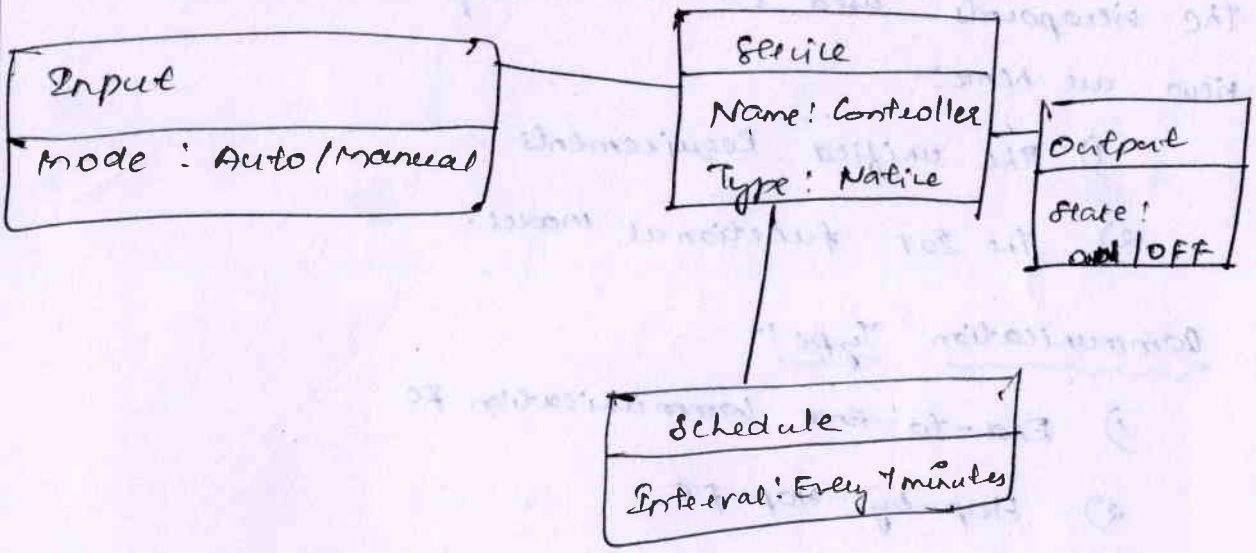
The information model for an object can obtain information about the objects structure and resource types. this can enable APIs to automatically be composed by middleware and automatically consumed by application software.



IoT information model

IoT information model is represented using Unified modeling Language (UML) diagram. The IoT information model maintains the necessary information about virtual Entities and their properties or attributes.

Service specification.



IoT Level specification :-

IoT Level 1 :- Single node, perform sensing, perform analysis and hosts the application.

IoT Level 2 :- Single node, perform sensing, perform local analysis.

IoT Level 3 :- Single node, data is analyzed in cloud and application is cloud based.

IoT Level 4 :- Multiple node, perform local analysis, application is cloud based.

IoT Level 5 :- Multiple end node and coordinator node.

IoT Level 6 :- Multiple Independent node, perform sensing, send data to cloud.

* Function view specification :- (Communication model)

Functional view describes the system's runtime functional components, their responsibilities, default functions, interfaces and primary interactions.

The viewpoints used for constructing the IoT functional view are hence:-

1) The unified Requirements
2) The IoT functional model.

Communication type:-

1) End-to-End communication FC
2) Hop-by-Hop FC
3) Network FC.

Operational view specification:-

Deployment and operational view depends on the specific actual use case and requirements. Smart object in the IoT uses different methods for communication using different technology.

The viewpoints:-

1) The IoT Domain model is used as a guideline to describe the specific application domain.
2) The functional model is used as a reference to the system definition.
3) Network connectivity diagrams can be used to plan connectivity topology.
4) Device Descriptions can be used to map actual hardware on the device.

Device and component integration:-

To integrate the devices and components. The

devices and components used in home automation examples are Raspberry pi, sensor, laser pointer, dependent ustilar, etc.

Application Development :-

It is the final stage in developing IoT application.

Two modes are provided :-

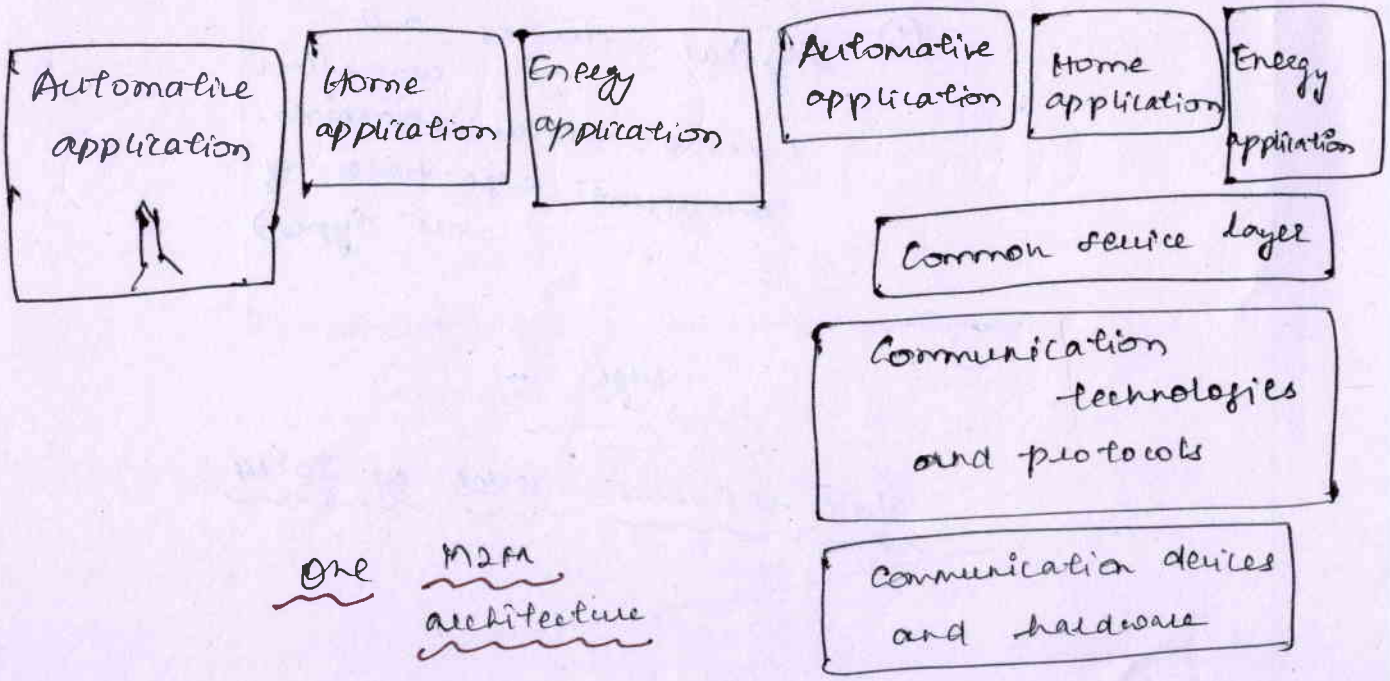
1) Auto mode :- It is enabled the light control to the web application is disabled and it reflects the current state of light.

2) Manual mode :- If auto mode disabled, the light control is enabled and it is used for manually controlling light and air conditioner.

DAY-37

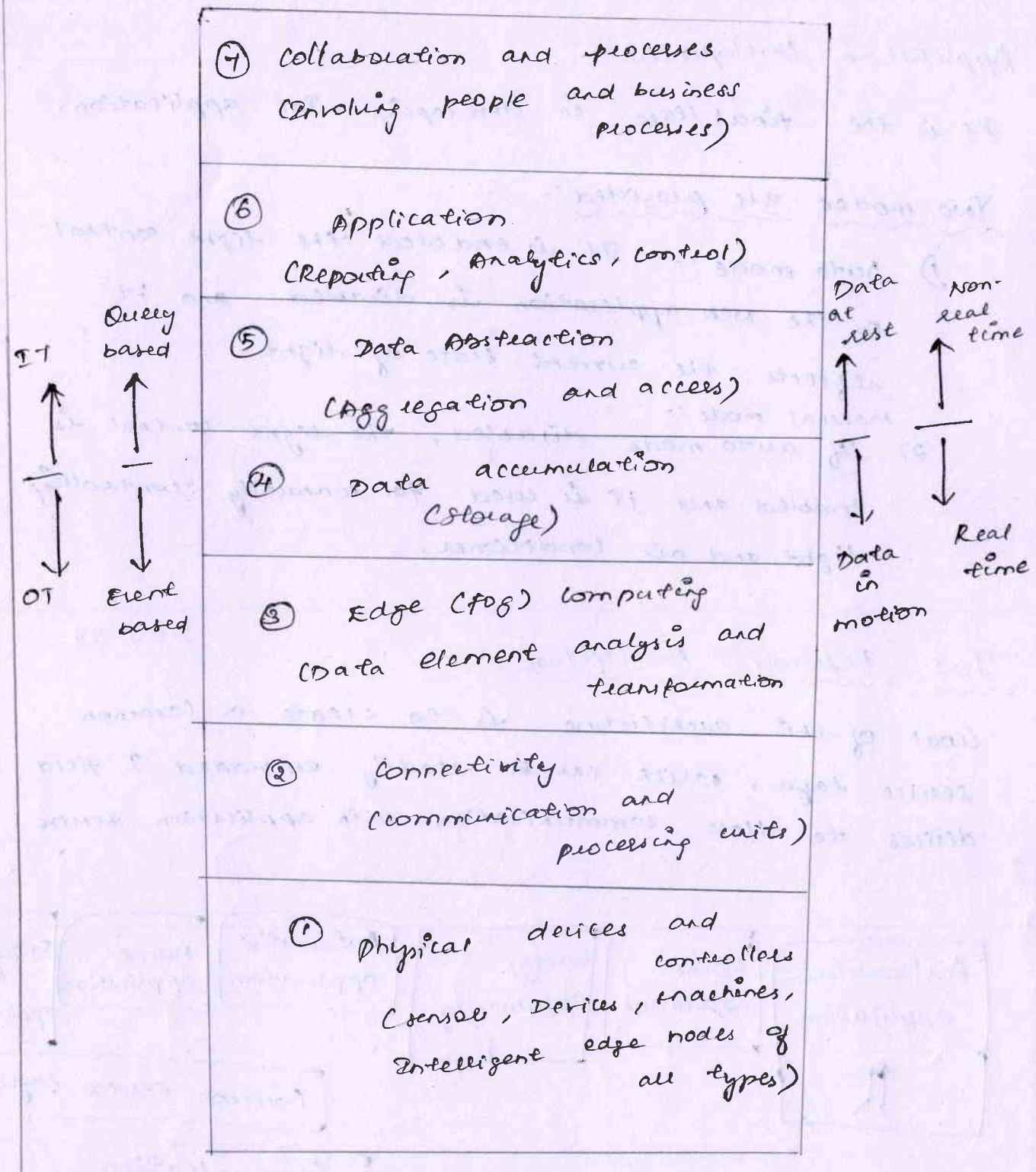
* IoT Reference Architecture :-

Goal of this architecture is to create a common service layer, which can be readily embedded I field devices to allow communication with application servers.



One M2M architecture

IoT World Forum Standardized Architecture center



IoT reference model of IoTWF

IoT reference model define a set of levels with control flowing from center to the edge which includes sensors, device, machines and other type of an intelligent nodes.

Layer 1:- Comprises physical devices and controllers that might control multiple device. This level enables devices to communicate with one another and to communicate, via the upper logic levels, with application platforms such as computers, remote-control devices, and smart-phones.

Layer 2:- The IRF model includes gateways in level 2. Because the gateway is a networking and connectivity device, its placement at level 2 seems to make more sense.

Layer 3:- It performs data element analysis and transformation.

Layer 4:- The data accumulation level, is where data coming from numerous devices, and filtered and processed by the edge computing level.

Layer 5:- Data abstraction Layer:- Confirms that data set is complete and consolidates data into one place or multiple data stores using virtualization.

Layer 6:- Application Layer:- Interprets data using software application. Applications may include monitor, control and provide reports based on the analysis of the data.

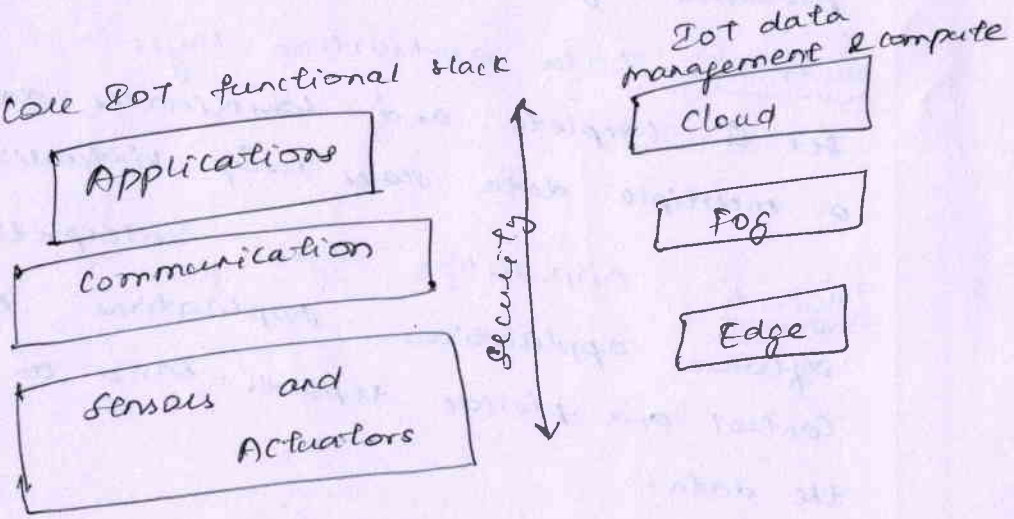
Layer 7:- Collaboration & processes Layer:- Consumes and shares the application information.

Using reference model, the following things are achieved:-

1. Decompose the problem into smaller parts.
2. Identify different technologies at each layer.
3. Define a system in which different parts can be provided by different vendors.
4. Have a process are defining interface that leads to interoperability.
5. Define a tiered security model that is enforced at one transition points between levels.

Characteristics of IOTWF model:-

1. simplifies
2. clarifies
3. Identifies
4. standardizes
5. Organizes.



It consists of core IOT functional stack group and IOT data management and compute stack.

DAY-38

* IoT protocols:-

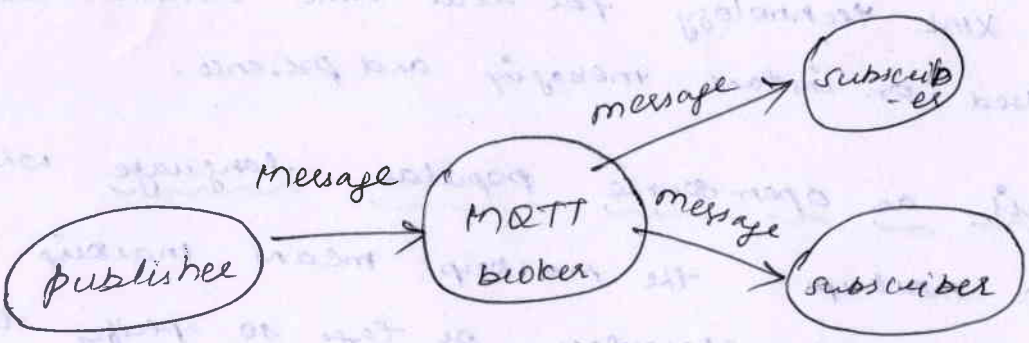
Messaging protocols are the rules, formats and function for messages sent between machines.

* MQTT:-

Message Queue Telemetry Transport (MQTT) is open connectivity for mobile, M2M and IoT.

MQTT characteristics:-

1. Lightweight message queuing and transport protocol
2. Asynchronous communication model with messages (events)
3. Low overhead (2 bytes header) for low network bandwidth applications.
4. Publish/subscribe (pubsub) model.
5. Decoupling of data producer and data consumer through message queues.
6. Runs on connection-oriented transport (TCP).



MQTT publish/subscribe framework

* A producer publishes a message (publication) on a topic (subject). A consumer subscribes (makes a subscription) for messages on a topic (subject).

- A message server (called BROKER) matches publications to subscriptions.
- If none of them match the message is discarded after modifying the topic. If one or more matches the message is delivered to each matching consumer after modifying the topic.

Publish / subscribe has three important characteristics:-

1. It decouples message senders and receivers, allowing for more flexible applications.
2. It can take a single message and distribute it to many consumers.
3. This collection of consumers can change over time, and vary based on the nature of the messages.

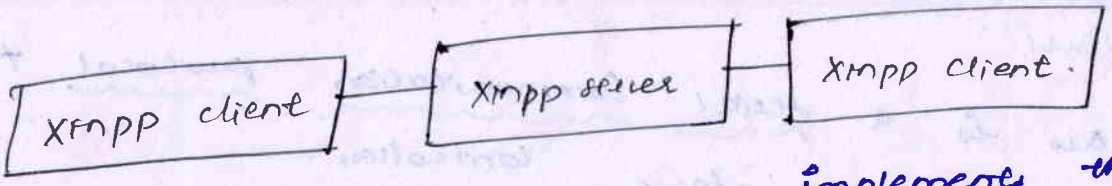
---

* XMPP:-

Extensible messaging and presence protocol (XMPP) is an open XML technology for real-time communication. It is based on instant messaging and presence.

XMPP is an open-source popular language which uses the markups. The markup means marking by some signs and characters or tags to specify the contents between the markups.

It allows the exchange of data between two or more systems and supports presence and contact list maintenance.



Each client on the XMPP network implements the clients the protocol with the XMPP server providing routing capability.

XMPP uses the transmission control protocol as its original and "native" transport protocol for web applications and firewalls.

Advantages:-

1. Supports HTTP transport protocol
2. It offers persistent connection.
3. It is decentralized in nature
4. It allows servers with different architecture to communicate.
5. Utilizes a decentralized client-server architecture.
6. It uses TLS and SASL to provide secured end to end connection.

Disadvantages:-

1. It does not have QoS mechanism as used by MQTT protocol
2. Streaming XML has overhead due to text-based communication compare to binary based communication
3. XML content transports asynchronously.
4. Server may overload with presence and instant messaging.

* Modbus :-

Modbus is a serial communication protocol for use with programmable logic controllers.

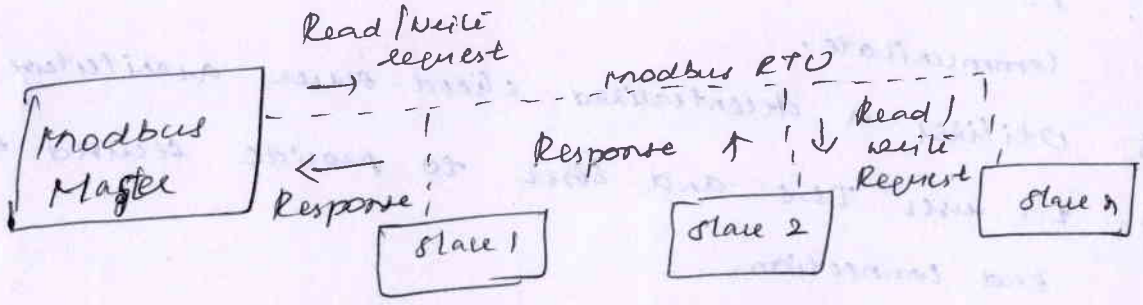
Modbus is an open protocol developed by Modicon.

There are 3 versions :-

1. Modbus
2. Modbus RTU
3. Modbus ASCII
4. Modbus TCP.

Modbus is a data communication protocol that is based on a request-response model.

Modbus is an application-layer messaging protocol, positioned at level 7 of the OSI layer.



Modbus serial architecture.

Modbus devices communicate using a master-slave (client-server) technique in which only one device (master/client) can initiate transactions (called queries). The other devices (slave/server) respond by supplying the requested data to the master, or by taking the action requested in the query.

A slave is any peripheral device which processes information and sends its output to the master using modbus

DAY-39

* CANBUS :-

Control Area Network (CAN) bus is a serial communication protocol that allows devices to exchange data in a reliable and efficient way.

CANBus was originally designed for automotive applications by Bosch in 1980s.

It is a multi-master, multi-slave, half-duplex and fault-tolerant protocol that fits well with the requirements of automobile applications.

CAN bus is a set of 2 electrical wires (CAN-LOW and CAN-HIGH) in the car network where information can be sent to and from ECUs.

The CAN network is divided into subnetworks connected together using a gateway module ECU.

The theoretical bit rate the CAN bus could support goes up to 1Mbps.

CAN bus is a broadcasting bus. The sender send its frame on the bus without any target specified.

CAN is a CSMA/CD protocol, meaning each node on the bus can detect collisions and back off for a certain amount of time before trying to retransmit.

CAN logic and arbitration :-

1. CAN 2.0A messages begin with an 11-bit message ID which identifies the message type and also establishes the message priority.
2. As with many computer interfaces, the CAN transceiver inlert the microcontroller signal. It uses logic "0" and logic "1".
3. CAN uses the message ID to perform bus access arbitration between nodes.
4. Each node waits for an idle bus state then begin to transmit its message ID.
5. Each node also listens to the bus to see if the bus state match its transmission.
6. If a node detects a dominant bus state while transmitting a recessive a message ID bit ("1" logic). it drops out of the current arbitration round and will try again the next time the bus is idle.

Advantages :-

1. Low cost
2. Large market segment
3. Light weight, Low latency
4. Reliable
5. Flexible

* BACNET :-

BACNet stands for Data Communication protocol for Building Automation and control Networks.

This data communication protocol is both an ISO and ANSI standard used for interoperability between cooperating building automation devices.

BACNet uses an object-oriented model for abstracting and representing information. BACNet includes 54 standard objects that cover many common and generally useful application.

A BACNet device is often comprised of a microprocessor based controller and software combination & use of BACNet protocol.

A Bacnet device object instance number must be field-configurable to be unique across the entire BACNet network where the device is installed.

BACNet divides the task of device interoperability into 3 distinct areas :-

1) Objects (information)

2) services (Action requests)

3) Transport systems (Internetworking, electronic messages)

1. Objects :- All information within a interoperable

BACnet device is modeled in terms of one or more information objects.

Each object is identified with an object identifier. An object identifier is a 32-bit binary number containing a code for the object type & object instance number.

Properties:-

BACnet property conveys information about a BACnet object. A property identifier and the property value. property Identifiers are numbers that uniquely identify a given property is the context of the object type. properties purpose is to allow other BACnet devices to read information about the object containing the property and potentially write (change) a different value to the property.

Services:-

It is formal requests that one BACnet device sends to another BACnet device to ask it to do something.

Services are grouped into 5 categories:-

1) Object access (read, write, create, delete)
2) Device management (discover, time synchronization, initialize, backup and restore database).
3) Alarm and event (alarms & changes of state)
4) File transfer (send data, program transfer)
5) Virtual terminal (human machine interface menus)

The device defines each request and any parameters that needs to be conveyed in the request and its reply.

UNIT- V

IOT SYSTEM DESIGN

DAY- 40

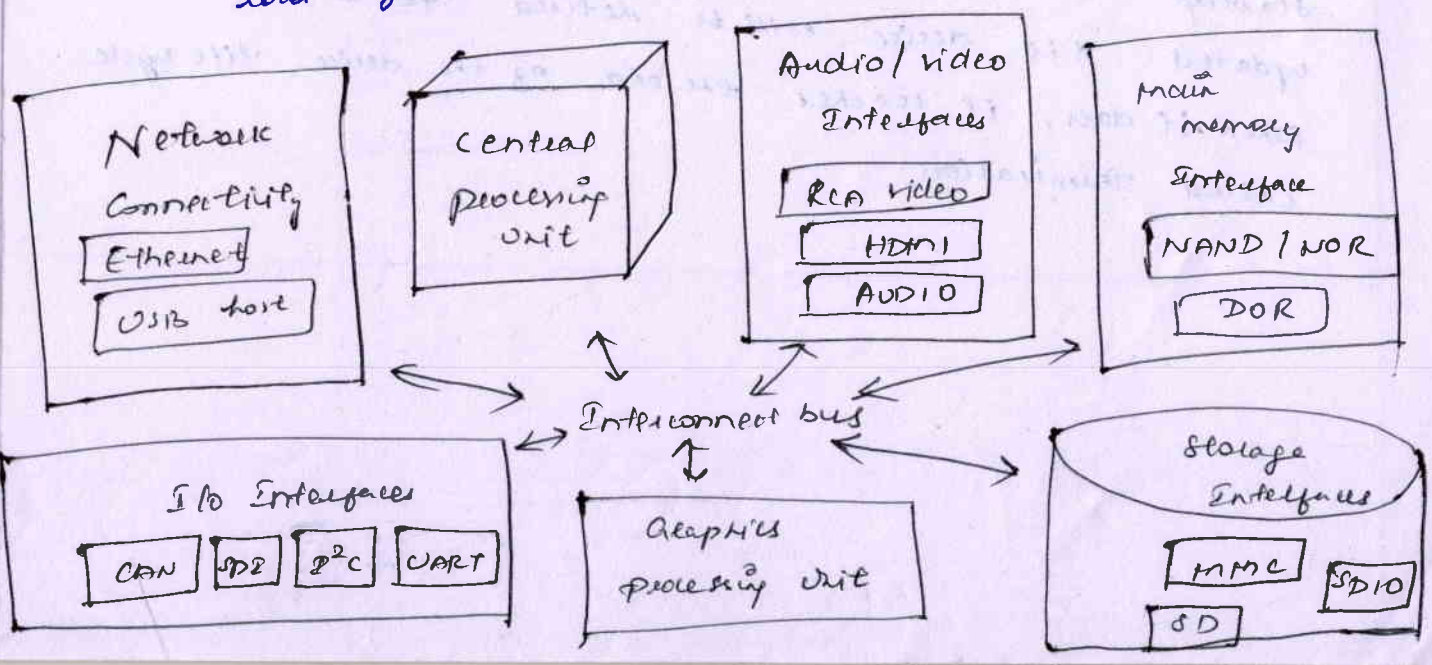
* Basic building blocks of an IOT device :-

* IOT device are connected to the Internet and send information about themselves or about their surroundings over a network.

→ The Internet of things is the network of physical objects ie, devices, vehicles, buildings and other items embedded with electronics, software, sensors and network connectivity.

* the Internet of things refers to the set of devices and systems that interconnect real-world sensors and actuators to the Internet. this includes many different types of systems, such as :-

1) Mobile devices
2) Smart meters and objects
3) Wearable devices including clothing, health care implants, Smart watches & fitness devices.
4) Internet-connected automobiles
5) Home automation systems including thermostats, lighting and home security.
6) other measuring sensors for weather, traffic, ocean tides, road signals & more.



IoT devices provide interface to various wire and wireless devices. Interface includes memory interface, I/O interface for sensors, Internet connectivity interface, storage interface, etc.

Sensors :- Devices that can measure a physical quantity and convert it into a signal, which can be read and interpreted by the microcontroller unit.

Actuation :- IoT devices can have various types of actuators attached that allow taking actions upon the physical entities in the vicinity of the device.

Communication :- communication modules are responsible for sending collected data to other devices or cloud based servers and receiving data from other devices.

IoT device Life cycle :-

1. Boot-up :- The device loads the firmware & start to work as defined.
2. Initialization :- once boot-up is completed, the system reads the configuration, established connection, etc.
3. Operation :- The device performs its desired purpose continually.
4. Update :- New firmware is installed, the device reboots and then starts to load the new firmware.

The device should complete its previous life cycle before starting the next life cycle every time the firmware is updated. The device will be retired for whatever reason. When it does, it reaches the end of the device life cycle called termination.

* Raspberry Pi :-

A Raspberry pi is a credit card-sized computer originally designed for education, inspired by the 1981 BBC micro.

Creator Eben Upton's goal was to create a low-cost device that would improve programming skills and hardware understanding at the pre-university level.

To get the Raspberry Pi working an SD card needs to be prepared with the linux operating system installed.

Raspberry pi models have the following features!-

1. Operating system!- Raspbian RaspBmc, Arch Linux, Risc OS, OpenELEC Pidora.
2. video output :- HDMI composite RCA.
3. supported resolutions!- 640 x 350 to 1920 x 1200, including 1080p, PAL and NTSC standards.
4. power source !- micro USB.

The Raspberry pi comes with a set of 26 exposed vertical pins on the board. these pins are a general purpose Input/output interface that is purposely not linked to any specific native function on the Raspberry pi board.

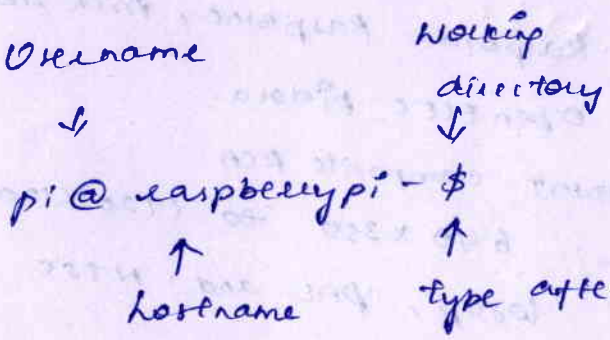
The Raspberry pi draws its power from a microUSB port and requires a microUSB-to-AC adapter. Because the pi is a micro computer and not simply a cell phone getting a battery topped off, you need to use a high quality charger with stable power delivery that provides a constant 5V with atleast 700mA minimum output for older model units and 2.5A for the pi3.

* Linux on Raspberry pi :-

There are several unix like operating systems for the RPI and there is an operating system called RISC OS that has its origin at the developers of the first ARM chips.

The Raspberry pi foundation recommends the use of the following Linux Distributions :-

1. Debian 7
2. Raspbian
3. Arch Linux ARM
4. Qtonpi.



command prompt.

Raspbian is the desired operating system for the Raspberry pi. In order to download and install the operating system onto our Raspberry pi.

1. Download both Raspbian and Win32 Disk Imager and save somewhere easily accessible.
2. Plug the USB memory card reader into your computer
3. Open Win32 Disk Imager
4. Find the location of the image file and the memory card
5. Click "write".

Logging In:-

Now it is time to turn on our Raspberry pi. When the memory card, HDMI lead, Ethernet cable, mouse and keyboard are plugged in, plug in the power lead.

Wait until your screen reads "raspberrypi login".

Username = pi [Enter]

password = raspberry [Enter]

Starting the Raspbian GUI:-

GUI stands for Graphical User Interface and is a type of operating system. It is the most common type of user interface as it is very friendly way for people to interact with the computer. It makes use of pictures, graphics, icons and pointers hence the name "Graphical User Interface".

DAY-43

* Raspberry Pi Interfaces:-

Three types of interface is supported by Raspberry pi.

1. Serial:-

* It uses serial peripherals for serial communication
* Transmit (TX) and Receive (RX) pin is used for serial communication.

2. Serial peripheral Interface (SPI):-

SPI is a communication protocol used to transfer data between micro-computers like the Raspberry Pi and peripheral devices. These peripheral devices may be either sensors or actuators.

* SPI uses 4 seperate connections to communicate with the target device. these connections are the serial clock (CLK), master Input slave output (MISO), master output slave Input (MOSI) and chip select (CS).

* the clock pin sense pulses at a regular frequency, the speed at which the Raspberry pi and SPI device agree to transfer data to each other.

* for the ADC, clock pulses are sampled on their rising edge, on the transition from low to high.

* the MISO pin is a data pin used for the master to receive data from the ADC. Data is read from the bus after every clock pulses.

* the MOSI pin sends data from the Raspberry pi to the ADC. The ADC will take the value of the bus on the rising edge of the clock. This means the value must be set before the clock is pulsed.

* the chip select line chooses which particular SPI device is in use. If there are multiple SPI devices, they can all share the same CLK, MOSI and MISO.

The SPI has the following features!-

1) 16-bit shift register.

2) 16-bit Receive buffer register (SPIBUF) and 16-bit Receive buffer emulation alias register (SPIEMU)

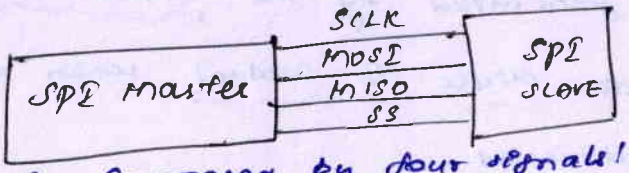
3) 16-bit transmit data register (SPIDAT0) and 16-bit transmit data and format selection register (SPIDAT1)

4) 8-bit baud clock generator

5) serial clock (SPICLK) I/o pin.

6. Slave In, master out (SPISIMO) I/o pin
7. slave out, master in (SPISOMI) I/o pin
8. Multiple slave chip select (SPISCS[n]) I/o pin
9. programmable SPI clock frequency range
10. programmable character length (2 to 16 bits)
11. programmable clock phase (delay or no delay)
12. programmable clock polarity (high or low)
13. Interrupt capability
14. DMA support (read/write synchronization events)
15. Up to 66 MHz operation.

Master - slave Configuration of SPI :-



SPI bus is composed by four signals :-
MOSI, MISO, SCLK and /SS

[Master out slave in (MISO)
Master in slave out (MISO)
Serial clock (SCLK)
Active low slave select (SS)]

- MOSI :- This pin is used to transmit data out of the SPI module when it is configured as a master and receive data when it is configured as slave.
- MISO :- This pin is used to transmit data out of the SPI module when it is configured as a slave and receive data when it is configured as master.
- /SS :- This pin is used to output the select signal from the SPI module to another peripheral with which a data transfer is to take place when its configured as a master and its used as an input to receive the slave signal when the SPI is configured as slave.

* SCLK :- This pin is used to output the clock with respect to which the SPI transfers data or receive clock in case of slave.

* SCK master device will generate a pulse and the data will be synchronized in both master and slave devices.

* SPI is a synchronous protocol.

* SPI creates a data loop between 2 devices. Data leaving the master exits on the SDO (Serial data output) line. Data entering the master enters on the serial data input, SDI line.

* A clock (SCK), is generated by the master device.

* SS allows a master device to control when a particular slave is being addressed.

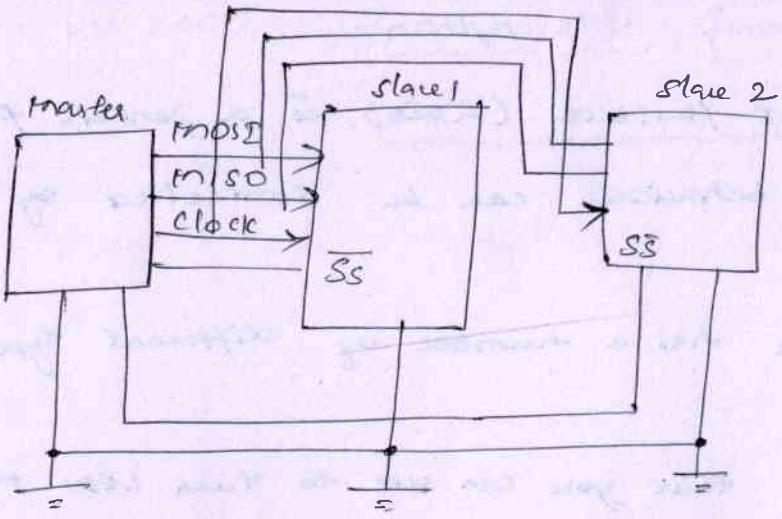
* For SPI, there are serial clocks (SCLK), chip select line (CS), Serial Data in (SDI) and Serial Data out (SDO).

* Synchronous operation, latch on rising or falling edge of clock, SDI on rising edge, SDO on falling edge. It operates in 1 to 2 MHz range.

* Master sends out clocks and chip selects. Activates the slaves it wants to communicate with.

* SPI data transmit and data receive register are the main elements of the SPI.

* The shift register in the master of (8,16,32) width and the shift registers in the slave are linked by MOSI and MISO pins to form a distributed 16, 32, 64 bit register respectively.



multiple slave Interfaces

* Data on the master SPI data transmit register becomes the input data for the slave read from the MOSI and the data read from the master SPI data receive register was the data send from the slave from MISO.
* Data on the shift registers are transferred into data receive register when the transfer completes and this data may be read from data receive register any time before next transfer has completed.

I2C :- Inter-Integrated communication :-

* I2C is a communication protocol that the Raspberry pi can use to speak to other embedded devices (temperature sensors, displays, accelerometers, etc).
* I2C is a useful bus that allows data exchange between microcontrollers and peripherals with a minimum of wiring.
* I2C is a two wire bus, the connections are called as SDA (Serial Data) and SCL (Serial clock).
* As same data & clock lines are shared between multiple slaves, & devices to communicate with each other.

Raspberry Pi Programming with python:- DAY-44

General purpose Input/output (GPIO) is a generic pin on a chip whose behaviour can be controlled by the user at run time.

The GPIO connector has a number of different types of connection:-

1. True GPIO pins that you can use to turn LEDs on and off etc.
2. I2C Interface pins that allow you to connect hardware modules with just two control pins.
3. SPI Interface with SPI devices, a similar concept to I2C but uses different standard.
4. Serial Rx and Tx pins for communication with serial peripherals.

Controlling LED with Raspberry Pi:-

Install python 2 library RPi.GPIO. A library that will let us control the GPIO pins.

Install commands:-

```
Import RPi.GPIO as GPIO
Import time
GPIO.setmode (GPIO.BOARD)
GPIO.setup (7, GPIO.OUT)
GPIO.output (7, True)
time.sleep (1)
GPIO.output (7, False)
time.sleep (1)
GPIO.output (7, True)
time.sleep (1)
```

```
GPIO.output (7,False)

print "Done"
GPIO.cleanup()
```

DAY-45

Task 1:-

Turn LED on for 2 seconds and off for 1 seconds, loop forever.

```
import Rpi.GPIO as GPIO
import time
def main():
GPIO.cleanup()
GPIO.setmode (GPIO.BOARD) # to use Raspberry Pi board pin numbers
GPIO.setup (11, GPIO.OUT) # set up GPIO output channel
while True:
GPIO.output (11, GPIO.LOW) # see Rpi board pin 11 to 0. Turn off LED.
time.sleep()
GPIO.output (11, GPIO.HIGH)
time.sleep (2)
main().
```

Interfacing an LED and switch with Raspberry Pi

```
Import RPi.GPIO as GPIO
# use the pin numbers from the ribbon cable board
GPIO.setmode (GPIO.BCM)
# setup this pin as input
GPIO.setup (17, GPIO.IN)
# check the value of the input pin
GPIO.input (17)
# Hold down the button, run the command again. the output should be "True".
GPIO.input (17)
```

Interfacing light sensor :-

```
# Tell the GPIO library to use
# Broadcom GPIO references
GPIO.setmode(GPIO.BCM)

# Define function to measure charge time
def RCtime(PiPin):
measurement = 0

# Discharge capacitor
GPIO.setup(PiPin, GPIO.OUT)
GPIO.output(PiPin, GPIO.LOW)
time.sleep(0.1)

GPIO.setup(PiPin, GPIO.IN)
# count loops until voltage across
# capacitor reads high on GPIO
while(GPIO.input(PiPin) == GPIO.LOW):
measurement += 1

return measurement

# Main program loop
while True:
    print RCtime(4) # measure timing using GPIO4.
```

DAY-16

* Home Automation :-

Home automation is the automatic control of electronic devices in your home. These devices are connected to the Internet, which allows them to be controlled remotely.

Interconnected devices enable to intelligently monitor and control smart homes in a future Internet of things.

Home Automation works on 3 levels :-

1) monitoring
2) control
3) Automatic.

Smart Lighting

Smart control the lights with automation signal system to save energy.

Smart, connected lighting is the next-generation energy-efficient LED products with additional sensors to sense things such as occupancy and temperature.

In automatic light control system, Light Dependent Resistor (LDR) sensor is used to detect bright (medium) dim / dark conditions.

Smart lighting is considered the one of the main solutions for energy reduction by means of controlling lighting level according to desired need with minimum energy consumption.

Smart - lighting systems utilize motion and light sensors for performing the control algorithms.

Smart appliances :-

Connecting everyday objects to the internet is an essential element of the IoT. The ability of an object to respond to remote commands and change its behaviour makes it an active device, such as the new wifi heating thermostat or a sky+ box.

Smart refrigerators can keep track of the items stored and send updates to the users when an item is low on stock.

OpenRemote is the professional open source middleware for an Internet of things.

Open Remote is a state of the art open source software platform for building control and automation.

Intrusion Detection :-

Intrusion Detection system (IDS) includes both hardware and software mechanisms and IDS is responsible for identifying malicious activities by monitoring network environment and system.

To detect any form of intrusion in restricted areas and report it immediately, following concept is used.

1. A PIR sensor is required to detect the presence of any human being in the room.

2. An RFID is required to validate the presence of the person in the room by tallying his identity with those in the database.

3. A camera is required to click the picture of the room and send it via email as an alarm.

4. An internet connection is required to register all these movements on a website so that it can be accessed from any place and any device.

Smoke for Gas Detection:-

Smoke or gas detector sensor which detects the smoke and turns on the buzzer alarm and all these are update on the Web page.

MQ2 is a semiconductor type sensor, which can appropriately sense the presence of smoke, LPG, methane, butane and other hydrocarbon.

The MQ-2 smoke sensor reports smoke by the voltage level as output. The more smoke is there, the greater the voltage output. The MQ-2 also has a built-in potentiometer to adjust the sensitivity to smoke.

By adjusting the potentiometer, one can change how sensitive it is to smoke, so there is a form of calibrating it to adjust how much voltage it will give in relation to the smoke it is exposed to.

DAY-47

* Smart cities!-

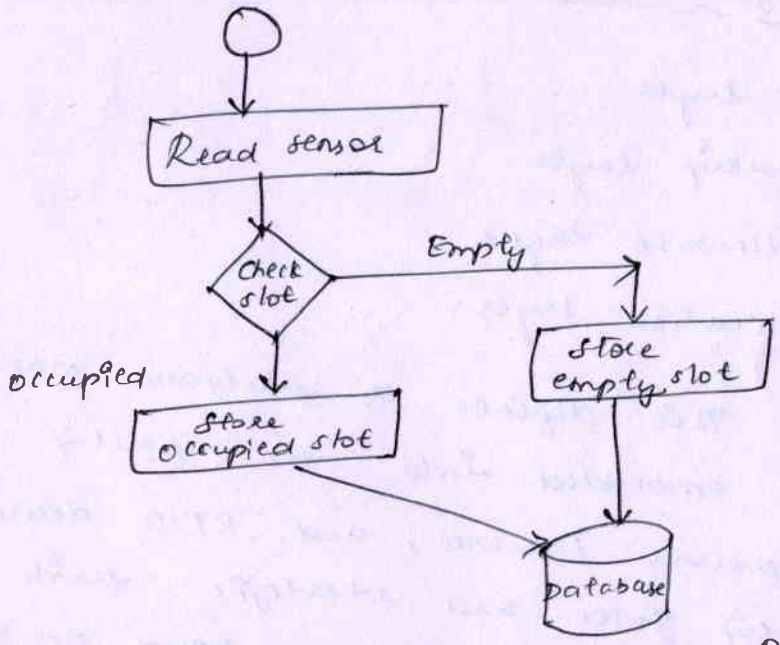
Smart city includes!-

1. Smarter management of city infrastructure using big data analytics.
2. Collaboration across multiple and dispirate agencies using cloud technologies
3. Real-time data collection, enabling quick response using mobile technologies.
4. Enhanced security!- Improved public safety and law enforcement and more efficient emergency response.
5. Better city planning improved schematics, project management and delivery.
6. Networked utilities smart metering and grid management.
7. Building developments more automation, and better management and security.

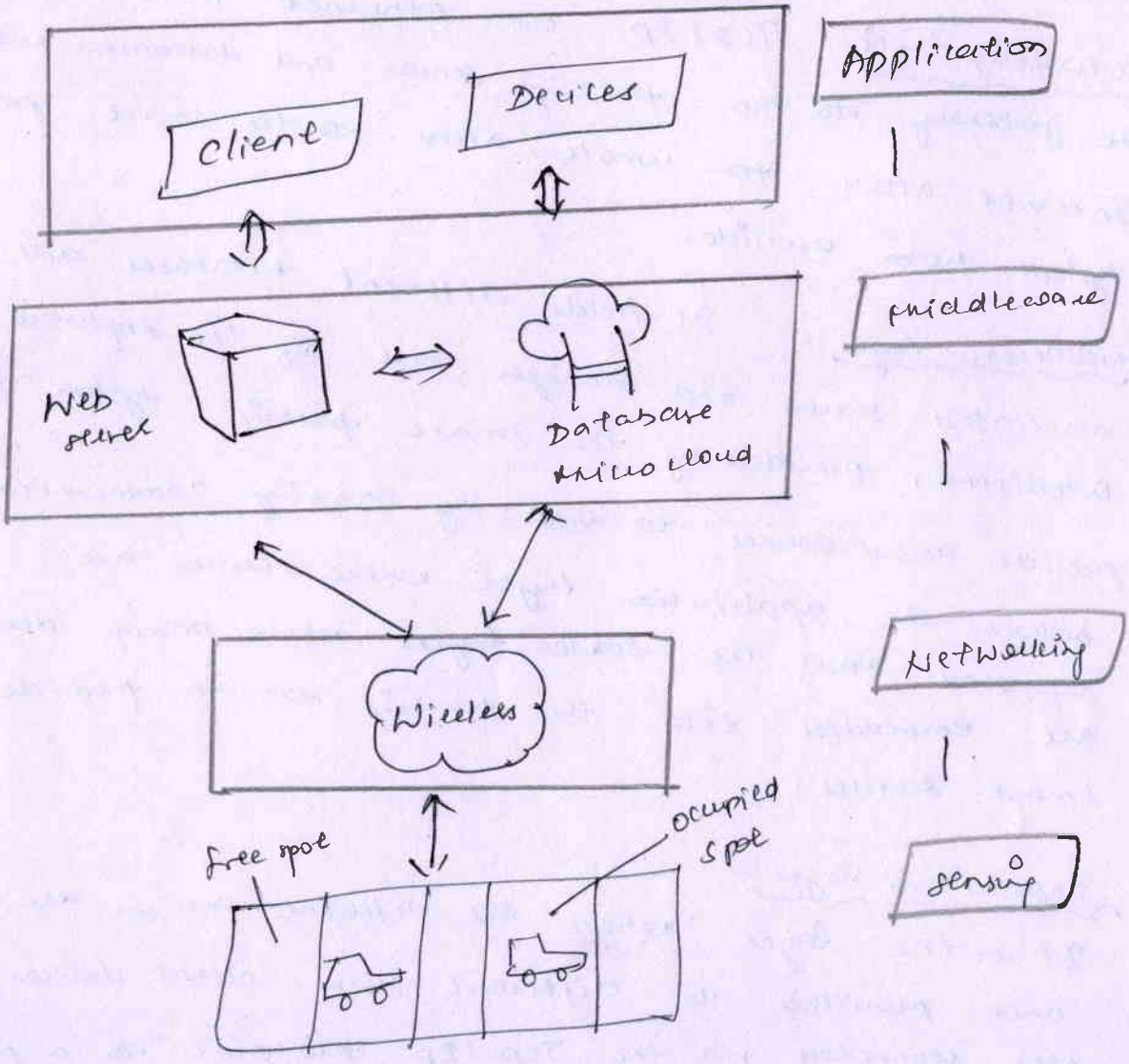
Smart parking!-

Smart parking systems typically obtains information about available parking spaces in a particular geographic area and process is real-time to place vehicles at available positions.

Smart parking application can be accessed by drivers from smart phones, tablets, sensor is used for each parking slot, to detect whether the slot is empty or occupied.



Process specification for smart parking IoT system.



This technology includes four layers:-

1) Sensing layer
2) Networking layer
3) middleware layer
4) Application layer.

Sensing Layer:- This defines a platform where sensor devices are embedded into the parking lot to detect car presence (absence, and RFID devices located at the parking gates and strategic parts of the parking are used to identify cars based on a unique mapping between RFID tags and car.

Networking layer:- TCP/IP over Ethernet for connecting the gateway to the parking sensor and database and Internet access for remote access to the smart parking system from outside

Middleware layer:- It holds different databases and associated services and manages all of the software intelligence, provided by the smart parking system to provide smart services to users by enabling communication between the application layer where services are required and the lower layers where smart devices are embedded into the parking lot to provide smart services.

Application layer:-
It is the layer where the different services are defined and provided to different users. Client devices have been connected via the TCP/IP protocol to a parking database.

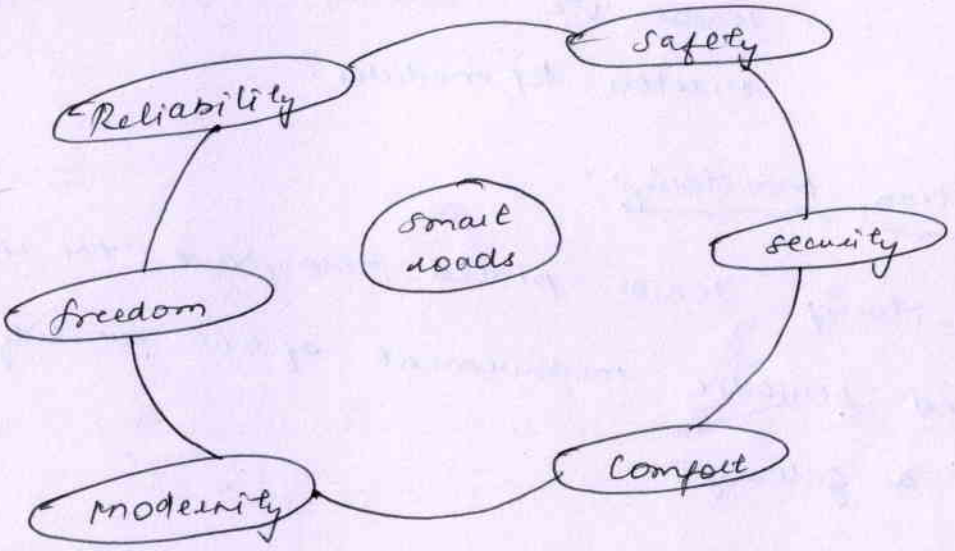
Smart Lighting :-

Smart light infrastructure is the backbone of the IOT in smart cities. Smart and wireless street light luminaries can act as service gateways for other street level IOT devices.

Smart street lights are intelligent lights that gather dynamic data i.e, data that keep changing dynamically by time, through some sensors and generate required information for the request claimed by a citizen on road.

Smart Roads :-

Sensor is installed on road to provides road traffic condition, travel time estimation, congestion and accident.



Smart roads characteristics

* Sensor collect this information and stored on the central database using cloud. This information helps for solving traffic congestion, making safe driving, keeping road condition upto date.

+ User can access the information from the cloud.

+ User also get real time information.

* Environment :-

DAY-48

Weather monitoring :-

Remote weather monitoring system which measures the following weather parameters :-

1. Daylight :- Using a photodiode as a wired binary switch sensor.

2. Humidity :- Using a wired analog humidity sensor

3. Temperature :- Using a digital bit-stream temperature sensor via wireless medium employing wireless RF modules.

Air pollution monitoring :-

Air monitoring sensor placed throughout the city. Sensor send periodic measurement of air quality data to a gateway.

Gateway sends information to network where the data is analyzed by the application server which can identify zones of concern and provide recommendations.

Application server provides information regarding air quality level throughout the city, including alert and pollution pattern via computer or mobile device.

Noise pollution monitoring:-

Nowadays assessments of environmental noise in urban areas is mainly carried out by officials who collect data at a sparse set of locations, by setting up sound level meters during a short period of time

The sensors interact with microcontroller which processes this data and transmits it over internet. Also allows authorities to monitor air pollution in different areas and take action against it.

Forest Fire Detection:-

The forecast forest fires cannot be detected by the satellites fire spreads uncontrollable. the wireless sensor network can detect and forecast forest fire more perfectly and accurately than the traditional satellite-based detection approach

This requires constant surveillance of the forest area. Wireless sensor networks (WSN) can potentially provide solution for these detection proces.

River floods Detection:-

IOT based river monitoring uses sensor to monitor water level and flow rate.

various sensors collect the information and send to the local processing center. processing sensor process the data and stored on the cloud.

* Agriculture :-

Machine Diagnosis and prognosis :-

IoT plays an important role in both diagnosis and prognosis.

Machine fault diagnostic and prognostic techniques have been the considerable subjects of condition-based maintenance system in the recent time due the potential advantage that could be gained from reducing downtime, decreasing maintenance costs and increasing machine availability.

Indoor Air Quality Monitoring :-

To Notice in healthy environment in factory, indoor air quality is monitored. Factory provides safety environment to workers.

Generating harmful gases (CO, NO and NO_2) in the factory, creates problem for workers. To avoid this, sensor is used to monitor sense the air quality of room.

Air quality is measured at different locations If serious happened, at threshold level of air quality is changed, sensor send alert to authority and it is also give the alarm.