Task 1

1. Reactive systems: Responsive under varying workload and when faced by failures.
   Ex. computers

2. Real time systems: Requires high accuracy in time synchronization.
   Ex. Video conferencing, airplane sensor and autopilot systems are hard real-time

3. Continuous/discrete/hybrid systems:
   Ex. Sine wave as continuous and sampling as discrete.

4. Embedded systems: Has a dedicated function in a mechanical or electronic system.
   Ex. GPS systems, central heating systems

5. Dependable systems: is the trustworthiness of a computer system such that reliance can justifiably be placed on the services it delivers.
   Ex. Autonomous cars

6. Distributed systems: A collection of autonomous computers linked by a computer network, and communicate and coordinate their actions only by message passing.
   Ex. Network in a company

Attributes of dependability

1. **Reliability:** is the probability of a component, or system, functioning correctly over a given period of time, under a given set of operating conditions.

. Dependability with respect to continuity of service
. The ability of the system to deliver services as specified

2. **Availability:** is the probability that the system will be functioning correctly at any given time.

. Dependability with respect to readiness for usage
. The ability of the system to deliver services when requested

3. **Safety:** property of a system that it will not endanger human life or the environment.

. Dependability with respect to avoidance of catastrophic consequences
. The ability of the system to operate without catastrophic failure

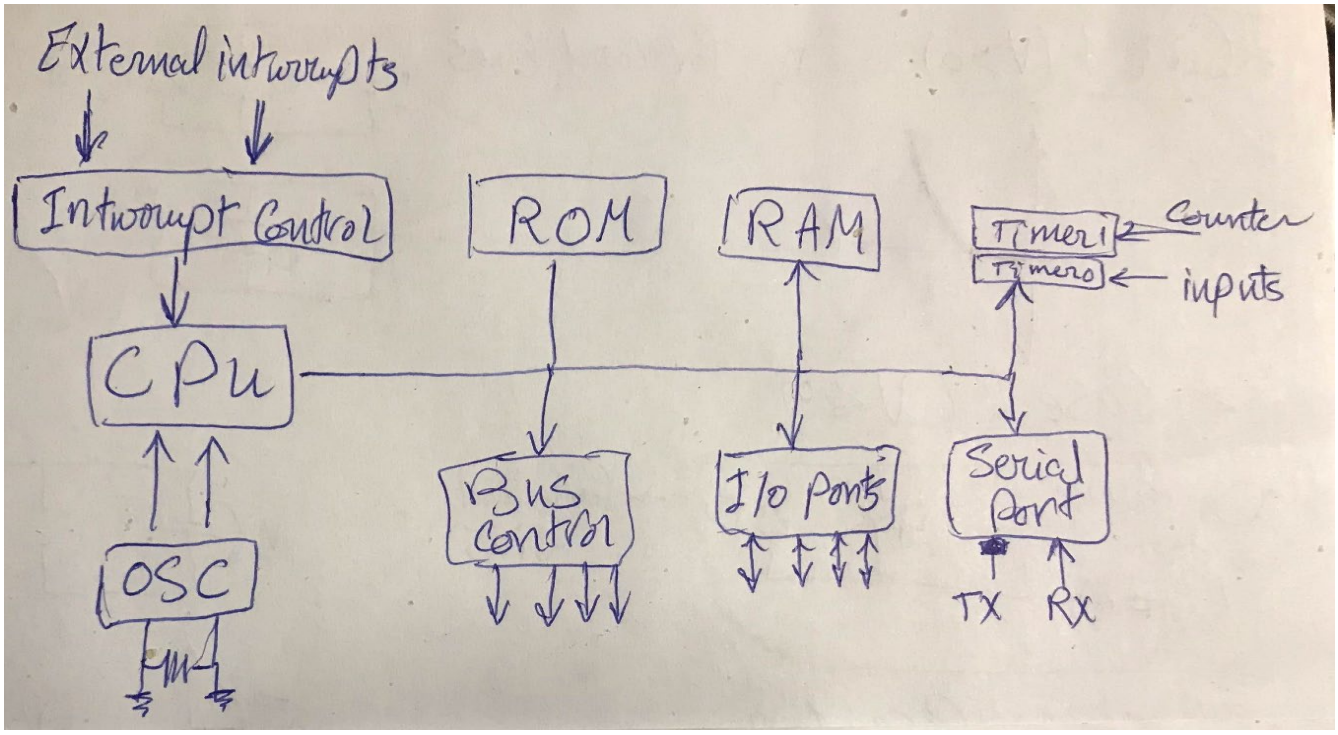
4. **Security:** Prevention of or protection against (a) access to information by unauthorized recipients or (b) intentional but unauthorized destruction or alteration of that information.

. Dependability with respect to prevention of unauthorized access and/or handling of information
. The ability of the system to protect itself against accidental or deliberate intrusion

Main elements of a Microcontroller

- ***CPU (Central Processing Unit):*** controls and monitors all the processes taking place inside the MCU. It is responsible for the reading and execution of all logical and mathematical functions being performed.

- ***RAM (Random Access Memory).*** This is temporary storage used only when powered on, for helping to run and calculate the programs the MCU is told to execute. It is continually overwritten while in use.
- ***ROM (Read-Only Memory).*** A pre-written permanent memory that persists even without power. It essentially instructs the MCU on how to execute its programs when asked.
- ***Internal Oscillator (the main timer of the MCU):*** The microcontroller's core clock and controls the execution rhythms of its internal processes. They keep track of time as it elapses during a given process, and help the MCU to begin and end specific functions at specified intervals.
- ***I/O (Input/Output) Ports.*** This consists of one or more communications ports, typically in the form of connective pins. They allow the MCU to be linked to other components and circuits for the flow of input/output data signals and power supply.



Which processors are typically used for microcontrollers?

.	AVR	ARM
1.	AVR micro controller refers to Advanced Virtual RISC (AVR).	ARM micro controller refers to Advanced RISC Micro-controller (ARM).
2.	It has bus width of 8 bit or 32 bit.	It has bus width of 32 bit and also available in 64 bit.
3.	Its speed is 1 clock per instruction cycle.	Its speed is also 1 clock per instruction cycle.
4.	It uses Flash, SRAM, EEPROM memory.	It uses Flash, SDRAM, EEPROM memory.

- ***2-bit (branch) predictor***

```c
#include <stdio.h>

int main()
{
    wTaken=0;
    wn_Taken=0;
    sTaken=0;
    sn_Taken=0;
    max=0;
    if (wTaken==max){
        wTaken = wTaken+1;
        max=max+1;
    }
    else if (sTaken==max){
        sTaken = sTaken+1;
        max=max+1;
    }
    else if (wn_Taken==max){
        wn_Taken = wn_Taken+1;
        max=max+1;
    }
    else if (sn_Taken==max){
        sn_Taken = sn_Taken+1;
        max=max+1;
    }
    return 0;
}
```