Competency 11

Explores IoT and identify the building blocks of embedded systems to develop simple applications

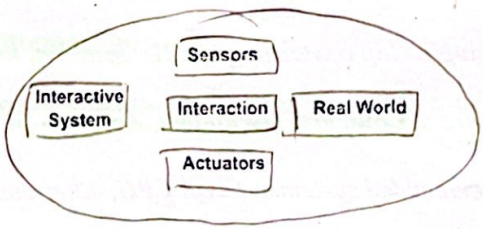
11.1: Acquires the knowledge of basic building blocks of embedded

Embedded System (microcontrollers)

- Embedded system is a computer system embedded into some other system such as a refrigerator, washing machine, car, etc. computer system ≠ embedded system
- It also follows Input, Process and Output (IPO) model.
- Sensors capture the state of the physical world like heat, speed and light as inputs. Processor processes them according to a program and produces outputs. Outputs drive actuators and change the state like heat, speed and light of the physical world.
- **Embedded systems, therefore, do physical computing.**

Physical computing

Physical computing involves interactive systems that can sense and respond to the world around them.



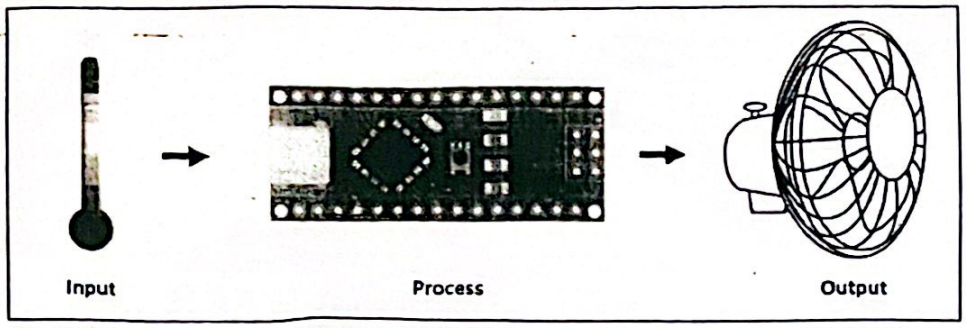
Eg:- **Smart automotive traffic control systems, factory automation processes, washing machines, fitness equipment found in homes, offices and industry.**

There are many **applications of physical computing**, for example in creative arts, museums, ubiquitous and embedded computing, scientific sensing, robotics, engineering control systems and robotics. detecting through sensors.

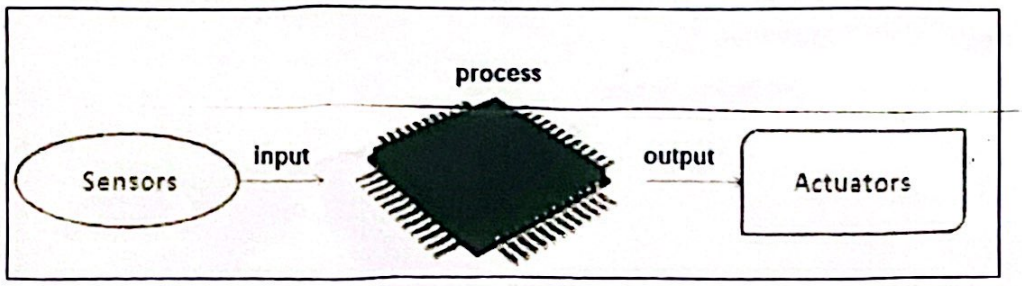
IEEE Definition for Embedded System

"A computer system that is part of a larger system and performs some of the requirements of that system; for example, a computer system used in an aircraft or rapid transit system." [IEEE,1992]

The components of an embedded system.

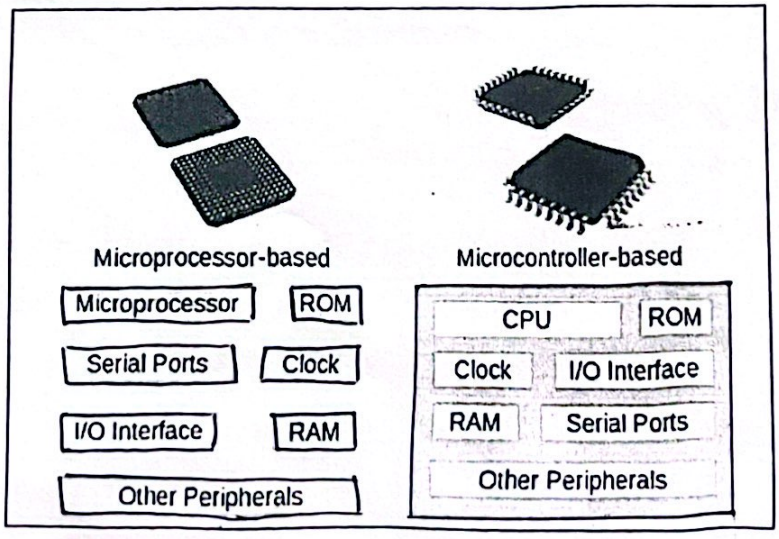


Embedded system model of physical world



Components of an Embedded System

- ***Embedded systems are developed using either microprocessors optimized for the purpose called embedded processors or microcontrollers.***
- ***The microcontroller is a single chip containing a CPU, memory, I/O ports, other peripherals & hardware components (Timers, Counters, Oscillators, Analog - Digital Converter etc.)***
- ***In microprocessor-based embedded systems, except CPU, other components are external to the microprocessor chip.***
- ***A majority of embedded systems are microcontroller based because they do not require expensive, powerful microprocessors to implement their basic functionalities.***



- Microcontrollers are used in multiple industries and applications, including in the home and enterprise, building automation, manufacturing, robotics, automotive, lighting, smart energy, industrial automation, communications and **internet of things (IoT) deployments.**
- Microcontrollers can use random access memory (RAM), flash memory, EPROM or EEPROM.
- Microcontroller architecture can be based on the Harvard architecture or von Neumann architecture, both offering different methods of exchanging data between the processor and memory. With a Harvard architecture, the data bus and instruction are separate, allowing for simultaneous transfers. With **Von Neumann architecture, one bus is used for both data and instructions**.

control bus, Address bus, Data bus ← von neumann.

Hardware and Software for Embedded System

- Embedded system development requires the knowledge of both hardware and software components.

Software's used in embedded system

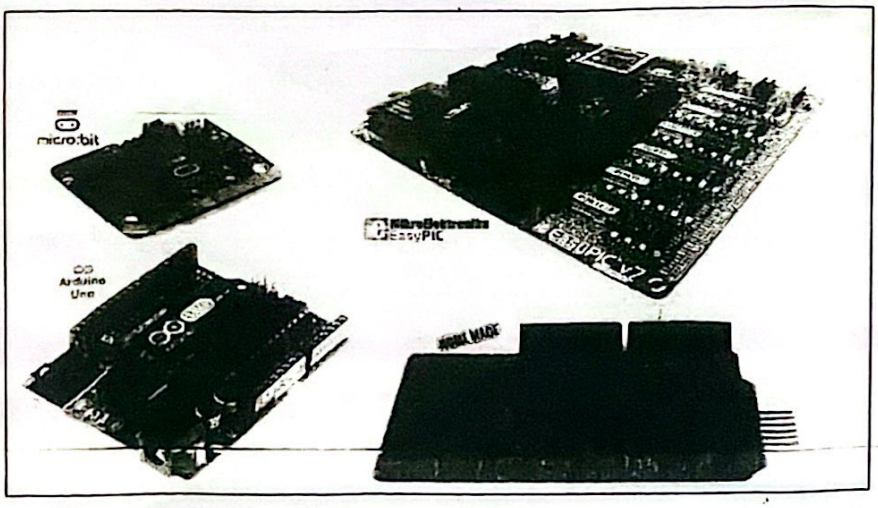
- OpenWrt, PikeOS, eCos, Fusion RTOS, Nucleus RTOS, RTEMS, INTEGRITY, uC/OS, QNX, FreeBSD and OSE.
- Code is typically written in ***C or C++,*** but various high-level programming languages, such as ***Python and JavaScript,*** are now also in common use to target microcontrollers and embedded systems.

➢ **They include a microcontroller-based development board, sensors, actuators, and an Integrated Development Environment (IDE).**

Microcontroller based development boards

- A **development board** is a printed circuit **board** with circuitry and hardware designed to facilitate experimentation with a certain **microcontroller**.
- Also there are certain hardware circuits which greatly aid testing and debugging such as pushbuttons and LEDs .It also served users of the microprocessor as a method to prototype applications in products.
- Unlike a general-purpose system such as a home computer, usually a development board contains little or no hardware dedicated to a user interface. It will have some provision to accept and run a user-supplied program, such as downloading a program through a serial port to flash memory. or some form of programmable memory in a socket in earlier systems.

- Microcontroller based development boards (**Arduino, Micro:bit, Raspberry pi)**



There are different microcontroller-based embedded system development platforms. Among them, we use **Arduino platform** due the following reasons:

- Free and open source hardware designs and software tool chain
- Cross platform support
- Extensive official and community support
- Extensive availability of software libraries
- Extensive hardware extendibility with extension shields
- Extreme user friendliness due to the use of wrapper functions (basic programming knowledge with any programming language is sufficient)

Arduino

Arduino is an open-source, low cost, easy-to-use hardware, and software platform. There are different kinds of Arduino boards available. They can be chosen according to our requirements and affordability.

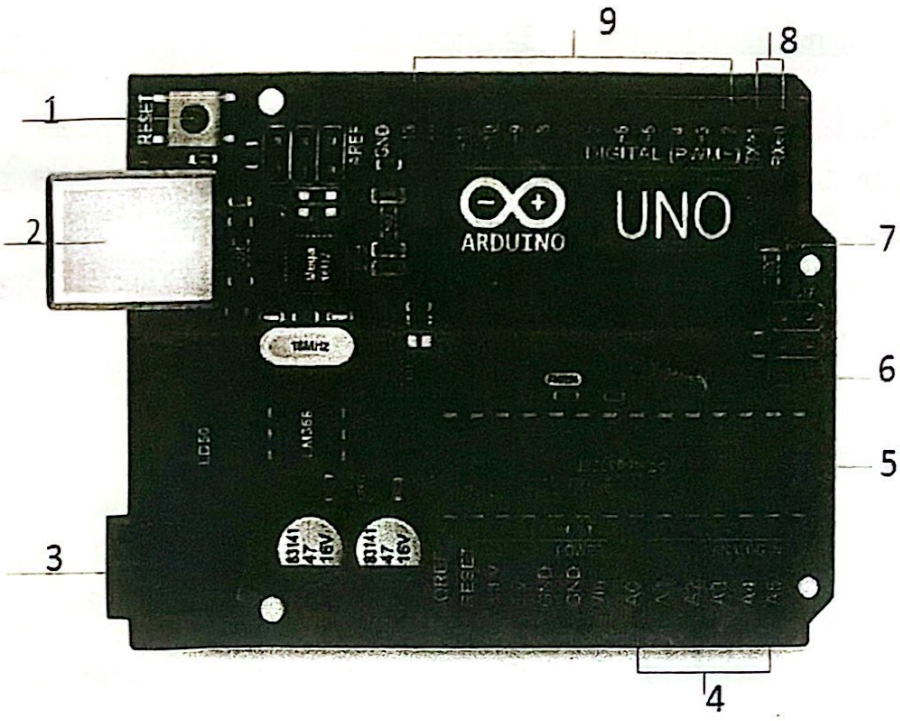
Eg:- **Uno** , Meg , 101 , Zero ,Yun , Lilypad , Nano MRK Zero

Among the different Arduino boards available, we use Arduino Uno due to the following reasons:

- Inexpensive
- Most widely used, heavily documented, extensively library supported development board
- 493 extension shield support
- 5V compatible I/O ports (most sensors and actuators use 5V)
- Do It Yourself (DIY) supported hardware design (using through hole electronic components)

Arduino Uno Board

The features of the Arduino Uno microcontroller-based development board.



1	Reset button	6	Oscillator
2	USB port	7	Power indicator
3	Power supply jack	8	Transmit & receive pins
4	Analog input pins	9	Digital input / output pins
5	Microcontroller		

Analog input pins - feed analog inputs to the microcontroller.

Digital I/O pins – feed digital inputs as well as deliver digital outputs.

Transmit and receive pins- transmit and receive data over serial communication to and from an externally connected device.

Arduino Uno comes with an **ATmega328P microcontroller**.

The **oscillator** provides clock pulses for the microcontroller to operate.

The **USB port** connects a computer to the development board. It uploads the firmware into the microcontroller, sends and receives data between the computer and the board and also supplies DC 5 volts to the board.

The **power supply jack** provides supplementary power when the board is not connected to a USB Port.

The **power indicator** indicates the status of power. Reset button resets the microcontroller.

➢ IDE Software (code editor, compiler and programmer)

- An IDE normally consists of at least a source code editor, build automation tools, and a debugger.
- Some IDEs, such as NetBeans and Eclipse, contain the necessary compiler, interpreter, or both.

Arduino Integrated Development Environment (IDE)

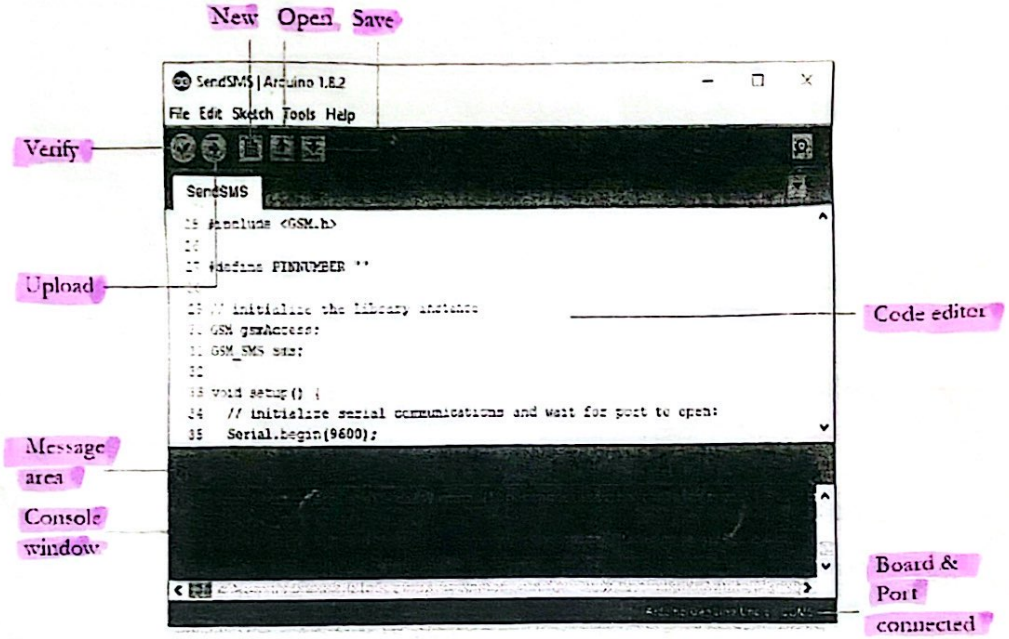
An Integrated Development Environment is essential for firmware development.

An IDE for embedded system consists of an uploader as well as a code editor and a compiler.

The uploader is used to upload the machine code into a microcontroller.

Arduino provides a free and open-source IDE called Arduino IDE available at : https://www.arduino.cc/en/main/software.

Key components of Arduino IDE



Programs written using Arduino IDE are called **sketches.**

- ♣ New button creates a new sketch in the Code editor.
- ♣ Open button loads an existing sketch from secondary storage.
- ♣ Save button saves the current sketch in the code editor.
- ♣ Verify button checks the source code for any syntax errors. If any errors found, they are reported in the console window.
- ♣ Upload button compiles source code into machine code and uploads it into the microcontroller.
- ♣ The message area displays the status of IDE.
- ♣ The console window shows error messages and other information.
- ♣ Board and port connected show the board in use and the connected port.

Embedded System Development

Develop the following embedded systems:

1. Blinker System
2. Auto Light System
3. Auto Fan System
4. Door Alarm System

To construct an embedded system, it is essential to follow some steps as given below:

- Construct the schematic diagram and assemble hardware
- Design firmware
- Develop firmware
- Compile firmware and Uploade machine code

SYSTEM 1: Blinker

This system is to turn on and off a Light Emitting Diode (LED) periodically.

Required components

1 × Arduino Uno microcontroller-based development board (to control the Blinker system)

1 × LED (to emit the light periodically)

1 × 220Ω Resistor (to drop the voltage and limit the current to LED as required)

```
// blinks an LED every ½ a second
const int ledPin = 8;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  digitalWrite(ledPin, HIGH);
  delay(500);
  digitalWrite(ledPin, LOW);
  delay(500);
}
```

Compile firmware and Uploade machine code.

The source code of firmware can be verified for any syntax errors using the verify button on IDE.
After verification, the development board is connected to the computer via a USB port.
Then source code is compiled into machine code and it is uploaded into the microcontroller using the upload button. Then the LED would start to blink every ½ a second drawing power via USB port.

SYSTEM 2: Auto Light

In system 1, we simply blinked the LED periodically. In this system, we are going to turn on and off a LED depending on the ambient light intensity using a light sensor.

Required components

1 × Arduino Uno microcontroller-based development board (to control the Auto Light system)

1 × LED (to emit the light)

1 × 220Ω Resistor (to drop the voltage and limit the current to LED as required)

1 × Light Dependent Resistor (LDR) (to capture the ambient intensity as an input)

1 × 10Ω Resistor (to drop the voltage and limit current flow to ground line)

```
// switches an LED on and off depending on light intensity
const int ldrPin = A0;
const int ledPin = 8;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  int sensorValue = analogRead(ldrPin);
  if (sensorValue < 150)
    digitalWrite(ledPin, HIGH);
  else
    digitalWrite(ledPin, LOW);
}
```

The analogRead(pin)function **reads voltage at the specified analog pin and returns a number between 0 and 1023.** This number represents the amount of light intensity that falls upon the LDR. It is then compared with a pre-determined value and the LED is turned on/off depending on the light intensity. In practice, a suitable pre-determined value is used to turn on the LED at the nightfall.

Compile firmware and Uploading machine code.

As in system 1, compiling firmware and uploading the machine code have to be done.

SYSTEM 3: Auto Fan

We have developed a simple embedded system to turn on and off a light periodically and another embedded system to sense the state of physical world and to activate an actuator according to that input. an Auto Fan system to turn on and off a motor of a fan depending on the room temperature.

Required components

1 × Arduino Uno microcontroller-based development board (to control the AutoFan system)

1 × 9 Volts DC Motor (to function the fan)

1 × LM35 Temperature Sensor (to capture temperature as an input)

1 × BC547 Transistor (to supply sufficient current to drive the motor)

1 × 1kΩ Resistor (to limit the current flow to transistor)

1 × 1N4001Rectifier Diode (to protect transistor from flyback current from motor)

```
// switches a motor of a fan on and off depending on room temperature
const int sensorPin = A0;
const int motorPin = 9;

void setup()
{
  pinMode(motorPin, OUTPUT);
}

void loop()
{
  int sensorValue = analogRead(sensorPin);
  float voltage = value * 5.0 / 1024;
  float temp = voltage * 100;
  if (temp > 30)
    digitalWrite(motorPin, HIGH);
  else
    digitalWrite(motorPin, LOW);
}
```

The analogRead(pin) function reads voltage at the specified analog pin and returns a number between 0 and 1024.
First, this number is converted into a voltage between 0 and 5V and then into a temperature in Celsius. The determined temperature is then compared with a pre-defined value and the motor is turned on/off depending on the room temperature.

Compile firmware and Uploading machine code.

As in the previous systems, compiling firmware and uploading the machine code have to be done.

SYSTEM 4: Door –Alarm

In this system, we would construct a door alarm system to trigger an alarm when a door is opened. The system uses a reed switch as a sensor to capture the input.

Required components

1 × Arduino Uno microcontroller-based development board (to control door alarm system)

1 × Piezo Buzzer (to generate alarm as output)

1 × Reed Switch (to detect whether door is open)

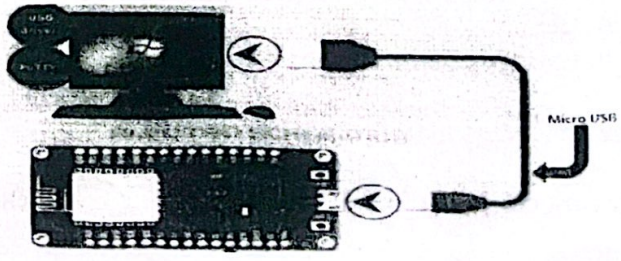
1 × 10kΩ Resistor (to drop the voltage and limit current flow to ground line)

```
// triggers an alarm when a door is opened
const int switchPin = 9;
const int buzzerPin = 8;

void setup()
{
  pinMode(switchPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
}

void loop()
{
  int switchState = digitalRead(switchPin);
  if (switchState == LOW)
    tone(buzzerPin, 262);
  else
    noTone(buzzerPin);
}
```

The tone(pin, frequency)function generates a square wave with the specified frequency on the given digital I/O pin. The generated tone plays continuously until the noTone(pin)function is called.

Compile firmware and Uploading machine code.

As in the previous systems, compiling firmware and uploading the machine code have been done.



11.2: Explores the Internet of Things (IoT) to create a simple application

Smart World

- IoT can create a "Smart World" where life is full of autonomous interconnected smart systems.

For example, what if your smart alarm clock not only wakes up you in the morning but also notifies your electric kettle to boil water for your morning tea.

How about smart refrigerator monitors the level of food items in it and send you a shopping list to your smartphone in the weekend. **The whole idea is to make our life more convenient and comfortable**.

- Things are everyday objects from small wristwatches to large cars and buildings.
- When added computing power, things become smart and are called embedded systems.
- Embedded systems can sense the state of the physical world and change the state of the physical world.
- When ***we connect the embedded systems to the Internet, they start interacting with themselves and users and create an Internet of Things.***

Enabling Technologies

Many technologies have enabled the Internet of Things.

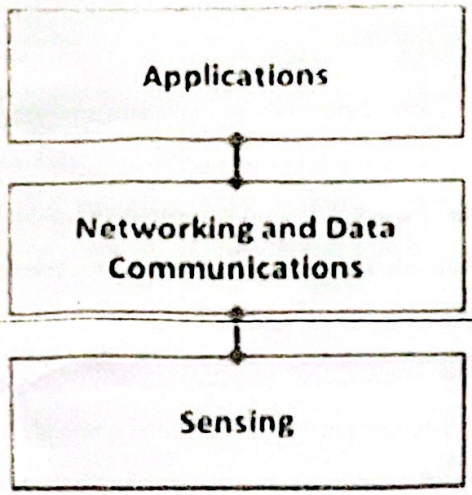
They include the Internet Protocol(IPv6) with an extremely large address space, increasing bandwidth and decreasing the cost of networking technology, increasing functionality and decreasing size and cost of sensor technology and increasing performance and capacity and decreasing power consumption, size and cost of processor and storage technologies, etc.

Iot Standards

Though many organizations work on the standardization process, we focus here on those that work on IoT and provide a definition for it. Accordingly, we considered IoT definitions from the European Telecommunications Standards Institute (ETSI), ITU, IEEE, the Internet Engineering Task Force (IETF), the National Institute of Standards and Technology (NIST), the Organization for the Advancement of Structured Information Standards (OASIS) and the World Wide Web Consortium (W3C). This list may be expanded in the future.

01) IEEE (Institute of Electrical and Electronic Engineers)

"A network of items—each embedded with sensors—which are connected to the Internet."



02) ETSI (European Telecommunications Standards Institute)

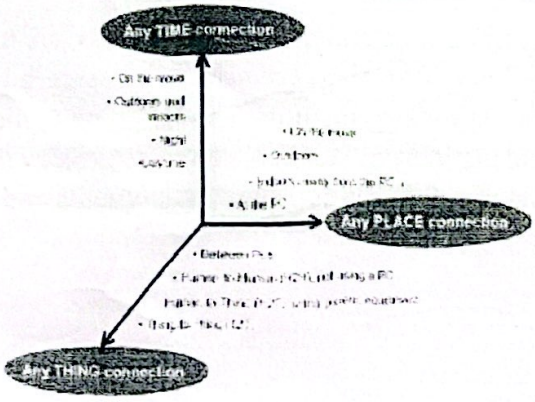
"Machine-to-Machine (M2M) communications is the communication between two or more entities that do not necessarily need any direct human intervention. M2M services intend to automate decision and communication processes."

03) IETF (Internet Engineering Task Force)

"The basic idea is that IoT will connect objects around us (electronic, electrical, non-electrical) to provide seamless communication and contextual services provided by them. Development of RFID tags, sensors, actuators, mobile phones make it possible to materialize IoT which interact and co-operate each other to make the service better and accessible anytime, from anywhere."

04) ITU (International Telecommunication Union)

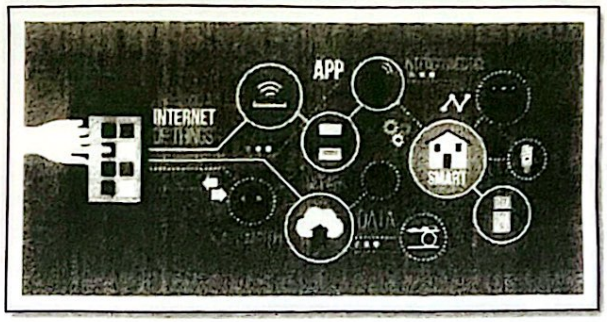
In its 2005 IoT report, ITU describes the IoT as a "ubiquitous network," in which the concept of ubiquitous networks is founded upon the all-inclusive use of networks and networked devices (ITU, SERIES Y, 2005). Literally, a ubiquitous networked environment is one in which networks and connectivity are available everywhere and anytime. Early forms of ubiquitous information and communication networks are evident in the widespread use of mobile phones.



connectivity will take on an entirely new dimension. Today, users can connect at any time and at any location. Tomorrow's global network will not only consist of humans and electronic devices, but all sorts of inanimate things as well. These things will be able to communicate with other things, e.g., fridges with grocery stores, laundry machines with clothing, implanted tags with medical equipment and vehicles with stationary and moving objects.

Additionally, ITU described the enabling technologies for the realization of the IoT. These technologies are: RFID for tagging things, sensor technologies for "feeling" things, smart technologies for making things "think" and nanotechnology for shrinking things.

Applications of IoT



- IoT can be applied to any physical system such as wearables, appliances, homes, transport, and agriculture, health, etc.
- Consumer IoT applications includes smart wearables with monitoring and home appliances with remote accessing capabilities.
- Commercial IoT applications include smart medical and health care systems with remote monitoring, emergency notification and assistive capabilities.
- Smart building and home automation applications with capabilities to monitor and control lighting, climate, entertainment, security and other systems in a building.
- Transportation applications such as smart traffic lights, smart parking, vehicle control, and safety and road side assistance systems etc.
- Industrial IoT applications include smart manufacturing applications such as process controls and predictive maintenance systems.
- Agriculture applications such as rainfall, humidity, pest infestation, and soil content monitoring systems and smart irrigation and fertilization systems, etc.