# 1...

# Introduction to Embedded Systems

## Learning Objectives...

- To understand fundamentals of embedded system.
- To understand what is a microcontroller, microcomputer, embedded system.
- To understand key concepts of embedded systems like IO, timers, interrupts, interaction with peripheral devices.
- To understand different components of a micro-controller and their interactions.

## 1.1 DEFINITION, CHARACTERISTICS OF EMBEDDED SYSTEM

**Embedded System:**

- Embedded is something attached with the other things
- Systems that are designed to perform specific set of functionalities or tasks.

**Definition :**

- An **Embedded system** is a microcontroller-based computer hardware system having software embedded in it.
- Some popular applications of embedded systems are mobile phones, dishwasher, washing machine, micro-ovens, smart watches etc.
- An embedded system can be an independent system.

**Components:**

- An embedded system has three components:
  1. Hardware
  2. Application Software
  3. Real Time Operating System

**Characteristics of Embedded System:**

1. **Single-Function:** An embedded system always performs a specialized operation and does the same again and again.
2. **Microprocessor Based:** Embedded system must be microprocessor or microcontroller based.
3. **Memory:** Embedded system software embeds in ROM; it does not need secondary memory.
4. **Hardware:** Software based system-Embedded systems software helps to provide more features and flexibility whereas hardware provides security and flexibility.
5. **Sophisticated Functionality:** Every embedded system application has specific set of functionalities. For example, Functionalities of washing machine is different from that of a microwave.
6. **Real Time Operation:** Operations performed by embedded system are always time-bound fashion. Embedded system must meet various timing and other constraints. They are imposed on it by the real-time natural behaviour of the external world.
7. **Portability**: Portability is a measure of the ease of using the same embedded software in various environments. It requires generalized abstractions between the application program logic itself and the low-level system interfaces.

## 1.2 REAL TIME SYSTEMS

- Real time systems are computer systems that used for real-time computing applications which processes data and events based on real time constraints. This systems perform a task in real-time and also generate quick responses under critical situations.
- Real time system is an information processing system based on hardware & software components which performs real-time application functions and can respond to events within predictable and specific time constraints.
- Examples of real-time systems include air traffic control systems, process control systems and autonomous driving systems etc.
- Processing in this type of system must occur within the specified constraints. Otherwise, this will lead to system failure.
- For real time computing systems must have to satisfy two requirements:
  - **Timeliness**: The ability to produce the expected result by a specific deadline.
  - **Time synchronization**: The capability of agents to coordinate independent clocks and operate together in unison.

**Types of Real-Time Systems:**

1. Hard Real-Time Operating System
2. Soft Real-Time Operating System
3. Firm Real-Time Operating System

1. **Hard Real-time operating system:**

- These Systems are Time-bounded where strictly time bounded operations are necessary for a successful output. For example, traffic signal.
- These operating systems guarantee that critical tasks be completed within a range of time. For example, a robot is hired to weld a car body. If the robot welds too early or too late, the car cannot be sold, so it is a hard real-time system that requires complete car welding by robot hardly on the time., scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

2. **Soft Real-time operating system:**

- This operating system provides some relaxation in the time limit hence no time-bound operation is required. For example, in a Microwave oven, there is no strict cooking time instruction. User can customize time delays as per the requirement.
- For example, Multimedia systems, Digital Audio Systems etc. Explicit, Programmer-defined and Controlled processes are encountered in real-time systems.

3. **Firm Real-time operating system:**

- This type of real-time system has to follow deadlines as well. In spite of its small impact, missing a deadline can have unintended consequences, including a reduction in the quality of the product. For example, Multimedia applications.

**Real-Time Tasks:**

- There are four types of tasks handled by real-time operating systems are as mentioned below:

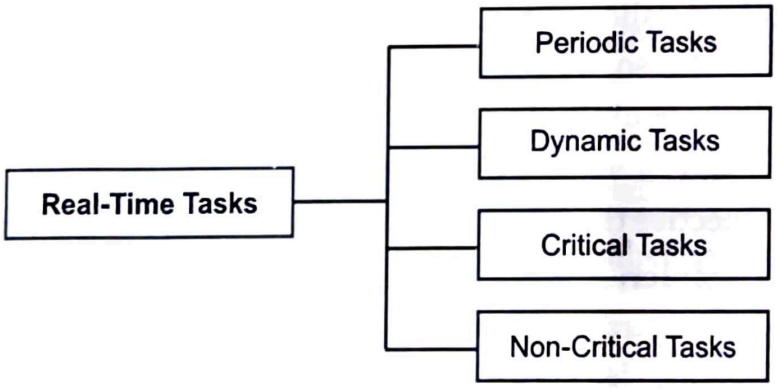


**Fig 1.1: Real-time Tasks**

1. **Periodic Tasks:** Tasks that takes place after a fixed interval of time known as periodic tasks. Tasks are released after a regular interval.
2. **Dynamic Tasks:** Tasks that are performed according to the invocation of certain events. An events may be generated by an external or internal process in the system.
3. **Critical Tasks:** Meeting the timeline is very much crucial in these tasks. If the timelines are missed then it can cause great difficulty. For example, in case of life support system, if system lags to meet the execution as per timeline, failure for that can cost human life.
4. **Non-Critical Tasks:** These types of tasks are not critical to the application.

**Classification of Real-Time task Scheduling:**

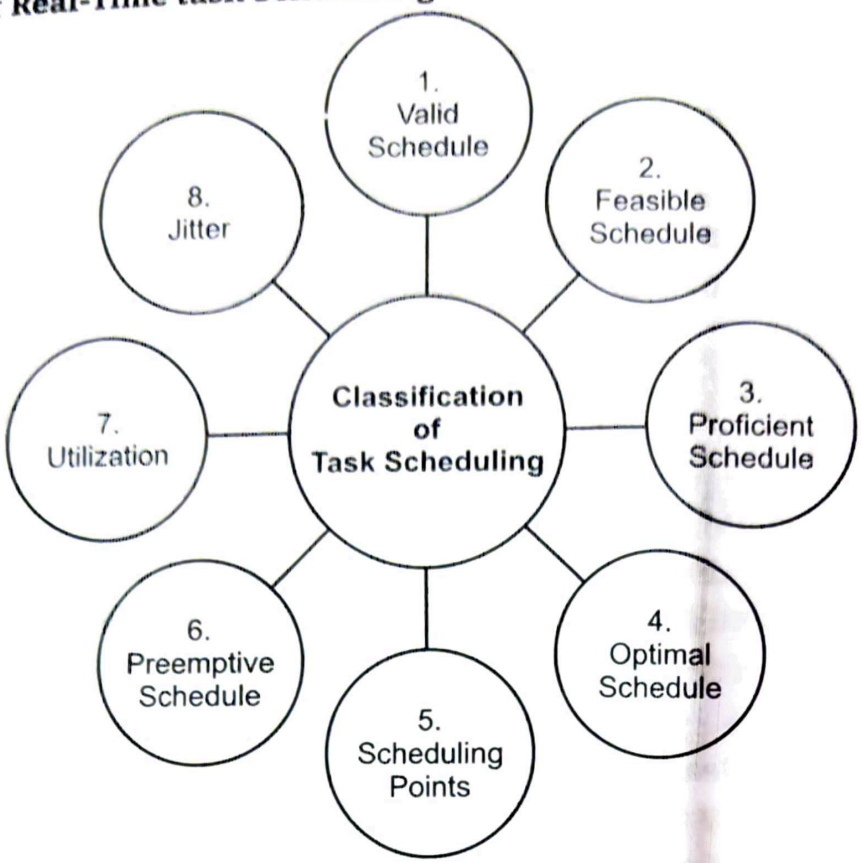


**Fig. 1.2 : Classification of Task Scheduling**

- Task scheduling is classified as shown below:

1. **Valid Schedule:** A valid schedule for a set of tasks is one where at most one task is assigned to a processor at a time, no task is scheduled before its arrival time, and all the resources are utilized to execute that one specific task only.
2. **Feasible Schedule:** A valid schedule is called a feasible schedule only if all tasks meet their respective time constraints in the schedule as per the timeline.
3. **Proficient Scheduler:** A proficient scheduler is one that can schedule the entire task feasibly in regards to any other schedule but the other schedulers are not able to schedule at least one task feasibly among all the tasks that is being schedule by the first scheduler.
4. **Optimal Scheduler:** A real-time task scheduler is called optimal if it can feasibly schedule any task set that any other scheduler can feasibly schedule.
5. **Scheduling Points:** The scheduling points of a scheduler are the points on a timeline at which the scheduler makes decisions regarding which task is to be run next. It is important to note that a task scheduler does not need to run continuously, and the operating system activates it only at the scheduling points to decide which task to run next. The scheduling points are defined as instants marked by interrupts generated by a periodic timer in a clock-driven scheduler. The occurrence of certain events determines the scheduling points in an event-driven scheduler.

6. **Preemptive Scheduler:** A preemptive scheduler is one that, when a higher priority task arrives, suspends any lower priority task that may be executing and takes up the higher priority task for execution. A preempted lower priority task can resume its execution only when no higher priority task is ready.
7. **Utilization:** The processor utilization (or simply utilization) of a task is the average time for which it executes per unit time interval.
8. **Jitter:** Jitter is the deviation of a periodic task from its strict periodic behavior. The arrival time jitter is the deviation of the task from the precise periodic time of arrival. It may be caused by imprecise clocks or other factors such as network congestions.

## 1.3 PROCESSOR BASICS: GENERAL PROCESSORS IN COMPUTER VS EMBEDDED PROCESSORS

- General Processors are the processors that power desktop computers and are the center of the computer revolution. They are of much higher cost than microcontroller.
- Embedded Processor is a microprocessor designed especially for handling the needs of an embedded system. They require less power, so these processors are very small and draw less power from the source.
- General Processors can perform multiple tasks and are more robust to development purpose. For example, Desktop/Laptop Processors.
- Embedded processors are dedicated, high-efficiency processors that can perform a set of limited tasks and are specifically designed to do that only. For example, Microprocessors in PCBs.

**Table 1.1: General Purpose Processor Vs Embedded Processor**

| Sr. No. | General Purpose Processor | Embedded Processor |
|---|---|---|
| 1. | It is designed using a microprocessor as the main processing unit. | It is mostly designed using a microcontroller as the main processing unit. |
| 2. | It contains large memory semiconductor memories like cache and RAM. It also contains secondary storage like hard disks etc. | It uses semiconductor memories, but does not require secondary memories like hard disk, CD. It sometime has special memory called flash memory. |
| 3. | It is designed such that it can cater to multiple tasks as per requirement. | It is designed such that it can cater to a particular predefined task. |

| Sr. No. | General Purpose Processor | Embedded Processor |
|---|---|---|
| 4. | It is mostly costlier as compared to the embedded systems. | It is cheaper as compared to a computer. |
| 5. | It requires huge number of peripheral devices and their controllers. | It is cheaper as it requires less no of peripheral devices and their controllers are microcontroller chip itself. |
| 6. | The Operating system and other software for the general purpose computers, are normally complicated and occupy more memory space. | The operating system (mostly RTOS i.e. Real Time Operating System) and other software occupy less memory space. |

**Microcontrollers and Properties:**

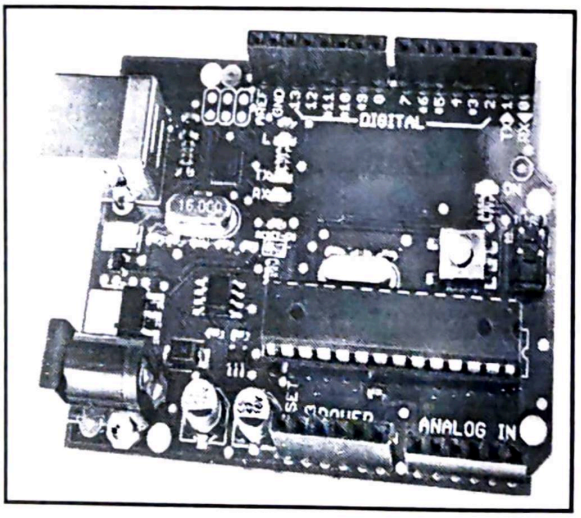


**Fig. 1.3: Microcontroller**

- A Microcontroller is an Integrated Circuit (IC) chip that can execute programs to control other devices. A Microcontroller has RAM, ROM and Input Output (IO) ports.
- A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.
- Microcontroller sometimes referred to as an embedded controller or microcontroller unit (MCU), microcontrollers are found in vehicles, robots, office machines, medical devices, mobile radio transceivers, vending machines and home appliances, among other devices. They are essentially simple miniature personal computers (PCs) designed to control small features of a larger component, without a complex front-end operating system (OS).
- A microcontroller is embedded inside of a system to control a singular function in a device.
- Microcontrollers are used in a wide array of systems and devices.

**Properties of a Microcontroller:**

1. It is a small computer. It has processor and some other components.
2. Used in automatically controlled devices.
3. Used in Embedded systems.
4. It has less computational capacity than microprocessor. So it is used for simpler tasks only.
5. Do not have math coprocessors.
6. Perform tasks such as Fetch, Decode and Execute.
7. It has memory, both RAM and ROM with some other I/O devices too.
8. Power consumption is less in microcontroller.
9. Used to handle real time tasks and they are single programmed, self-sufficient and task oriented.
10. Microcontrollers can use volatile memory types such as Random Access Memory (RAM) and non-volatile memory types such as Flash Memory, Erasable Programmable Read-only Memory (EPROM) and Electrically Erasable Programmable Read-only Memory (EEPROM).
11. Microcontrollers are designed to be readily usable without additional computing components because they are designed with sufficient on-board memory as well as offering pins for general I/O operations, so they can directly interface with sensors and other components.
12. 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.
13. Microcontroller processors can be based on Complex Instruction Set Computing (CISC) or Reduced Instruction Set Computing (RISC). CISC generally has around 80 instructions while RISC has about 30, as well as more addressing modes.

**Components of Microcontroller:**

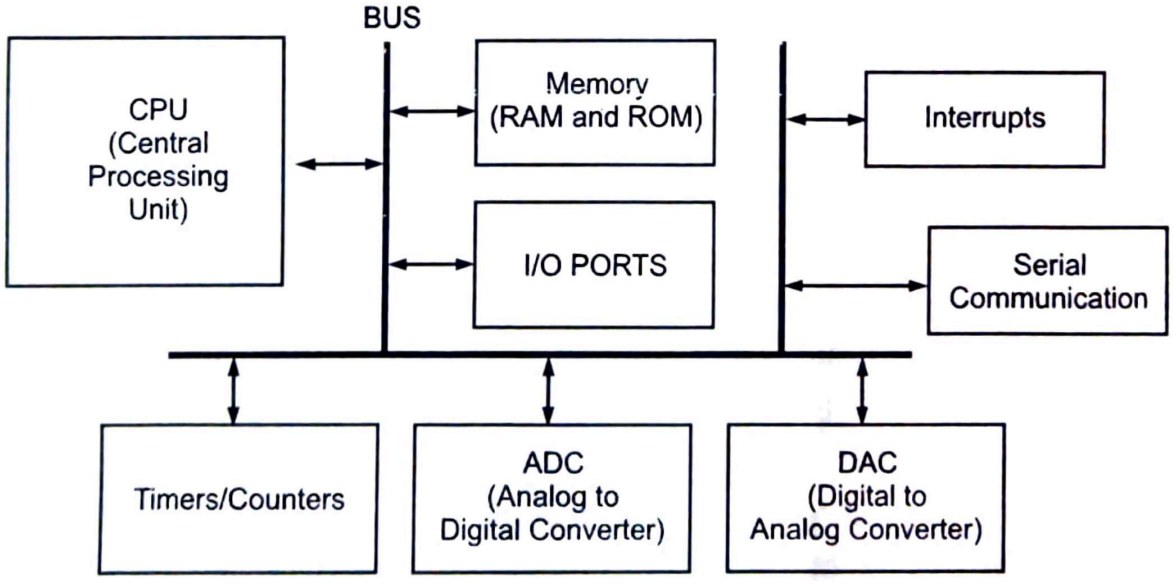


**Fig. 1.4: Components of Microcontroller**

- The core elements of a Microcontroller are:
  1. The processor (CPU)
  2. Memory
  3. I/O Peripherals

1. **The Processor (CPU):** A processor can be thought of as the brain of the device. It processes and responds to various instructions that direct the microcontroller's function. This involves performing basic arithmetic, logic and I/O operations. It also performs data transfer operations, which communicate commands to other components in the larger embedded system.
2. **Memory:** A microcontroller's memory is used to store the data that the processor receives and uses to respond to instructions that it has been programmed to carry out. A microcontroller has two main memory types:
   (a) **Program Memory**, which stores long-term information about the instructions that the CPU carries out. Program memory is non-volatile memory, meaning it holds information over time without needing a power source.
   (b) **Data Memory**, which is required for temporary data storage while the instructions are being executed. Data memory is volatile, meaning the data it holds is temporary and is only maintained if the device is connected to a power source.
3. **I/O Peripherals:** The input and output devices are the interface for the processor to the outside world. The input ports receive information and send it to the processor in the form of binary data. The processor receives that data and sends the necessary instructions to output devices that execute tasks external to the microcontroller.

- While the processor, memory and I/O peripherals are the defining elements of the microprocessor, there are other elements that are frequently included. There are many supporting components that can be classified as peripherals.

**Other supporting elements of a Microcontroller include:**

- **Analog to Digital Converter (ADC):** An ADC is a circuit that converts analog signals to digital signals. It allows the processor at the centre of the microcontroller to interface with external analog devices, such as sensors.
- **Digital to Analog Converter (DAC):** A DAC performs the inverse function of an ADC and allows the processor at the centre of the microcontroller to communicate its outgoing signals to external analog components.
- **System bus:** The system bus is the connective wire that links all components of the microcontroller together.
- **Serial Port:** The serial port is one example of an I/O port that allows the microcontroller to connect to external components. It has a similar function to a USB or a parallel port but differs in the way it exchanges bits.

**System-On-Chip:**

- System on a Chip or SoC is an integrated circuit that holds many components like CPU, microprocessor, microcontroller, memory, input/output ports, secondary storage etc.
- It is a complete computer system on a chip. They are small, self-contained, energy efficient and have low heat output.
- A SoC potentially includes all the core capabilities of a server such as software, a microprocessor, graphics processing unit, networking chip, memory and data storage.
- A microcontroller is a processor that has its program and data memory built in.
- A system-on-a-chip (SoC) is a microchip with all the necessary electronic circuits and parts for a given system, such as **a smartphone or wearable computer**, on a single integrated circuit (IC).

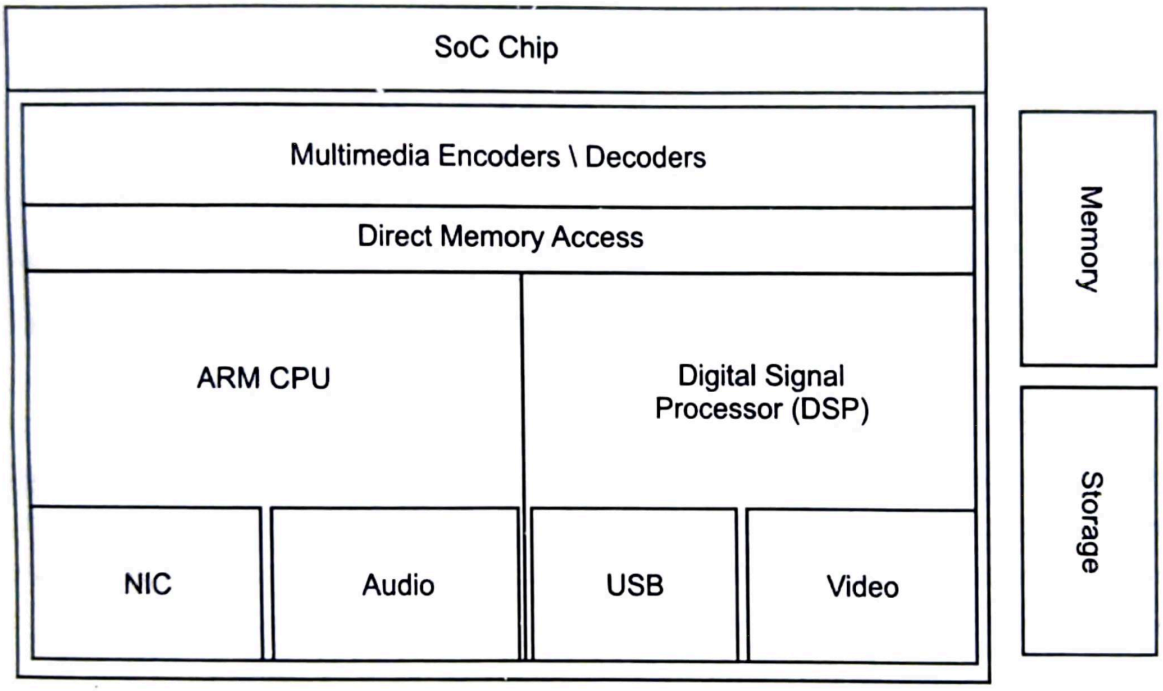


**Fig. 1.5: System-on-Chip (SoC)**

**Applications of SoC:**

1. **Mobile devices/Smart phones:** Mobile devices such as smart phones, SoC may integrate analog, 3G, Wi-Fi capabilities. Also now a day's smartphones are capable of enabling virtual reality experiences and streaming real-time 4K video.
2. **Cloud:** Using microservers as a pool of computing resources for a cloud platform.
3. **Supercomputing:** Constructing high performance computers using a large number of microservers as a component part.
4. **Smart Tablets:** SoC enabled fast connectivity, high-quality graphics, and efficient battery use for users of the tablet.
5. **Smartwatches:** SoCs that has enabled the development of a variety of smartwatches. Late last year, Fossil released their Fossil Sport Smartwatch which used the Wear 3100 platform.

## 1.4 COMPONENTS OF EMBEDDED SYSTEM

- Embedded systems that are made up of hardware and software components and which is use for performing specific functions.
- The embedded systems can be used in different sectors like industries, medical devices, agricultural sectors, automobile industry etc.
- There are multiple components involved in the design of an embedded system. The components used are software components and hardware components.

**A. Hardware Components:**

1. Power Supply
2. Processor
3. Memory
4. Timers/Counters
5. Communication Port
6. Input and Output
7. Circuits used in application

**1. Power Supply:**

- A power supply is a crucial component of the embedded system design. It is an electrical device mainly used to power up the electrical load. Normally, a 5V power supply is required for the system, however, it can also range from 1.8 to 3.3V.User can pick either one based on your requirements and application. To work the embedded system properly, a smooth and efficient power supply is needed. Both wall adopter and battery can be used as a power supply. Some power supplies work as independent equipment while others are incorporated into the embedded technology they power.

**2. Processor:**

- For any embedded system the processor acts as the brain of the system. The processor is responsible for deciding the performance of the embedded system. In the market there are multiple types of processors available and can be selected as per user requirement. The embedded system can act as a microcontroller and microprocessor. The processor can be an 8-bit processor, a 16-bit processor, and a 32-bit processor. The lesser the bit the smaller the application is for embedded systems. When large applications are used the higher bit processor is needed in the embedded system.

**3. Memory:**

- As there are different microcontrollers is used in the embedded system the memory is present in the microcontroller itself. There are basically two types of memory RAM (Random access memory) and ROM (Read-only memory). As the

RAM is volatile type memory the data can be stored temporarily in the memory and when system is switch off the data is lost from the memory. Read-only memory is classified as code memory. The ROM is used for storing the program and when the system is switch on the embedded system fetch code from ROM memory.

4. **Timers/Counters:**

- In some of the applications there is always a requirement of delay that needed to provide in the application. For example, in LED display applications there is a requirement of some delay so that LED can be continuing blink. And for that timer and counter can be used in the embedded system. The programming can be done in such a way so that delay can be generating the embedded system. The delay time span can be decided by using the crystal oscillator and system frequency so that delay can be generated as per user requirement.

5. **Communication Port:**

- Communication ports are used in embedded systems to establish communication with other embedded systems. There are several communication ports including USB, UART, USB, I2C, SPI, and RS-485. For simple applications, communications ports are utilized from the microcontroller, and for complex and advanced applications these ports are externally installed inside the embedded systems.

6. **Input and Output:**

- Input is required to interact with the embedded system. A sensor can be used to provide input to the system. The microcontroller used in the system can be configured as an input or output port. In the microcontroller, there are a fixed number of inputs and output ports that you can utilize as per your requirement.

7. **Circuits used in Application:**

- When the embedded system is design there are several hardware components that can be used for design purposes. The selection of the circuit is completely dependent on the application used for the embedded systems. For example, in temperature sensor applications there is a requirement of temperature sensors for measuring the temperature.
- Following are the **basic circuit components** that can be used in an electrical circuit.
  1. Printed Circuit Boards
  2. Resistor
  3. Capacitor
  4. Transistors
  5. Diodes
  6. Integrated circuit
  7. LED (light-emitting diode)
  8. Inductor

**B. Software Components:**

- Along with Hardware components there are also some software components are there used in Embedded System:

1. Editor
2. Compiler
3. Assembler
4. Emulator
5. Linker
6. Debugger

## 1.5 INTRODUCTION TO EMBEDDED PROCESSOR

- An embedded processor is a microprocessor designed especially for handling the needs of an embedded system. Embedded systems require less power, so these processors are very small and draw less power from the source. An ordinary microprocessor only comes with the processor in the chip. The peripherals are separate from the main chip, resulting in more power consumption.
- There are two main types of embedded processors: ordinary microprocessors and microcontrollers. Embedded processors are used for those systems which do not require the processing power of standard devices such as desktops, laptops or workstations.
- Processor is the heart of an embedded system. It is the basic unit that takes inputs and produces an output after processing the data. For an embedded system designer, it is necessary to have the knowledge of both microprocessors and microcontrollers.

**Units of Embedded Processor:**

- A processor has two essential units:
  1. **Program Flow Control Unit (CU):** The CU includes a fetch unit for fetching instructions from the memory. The EU has circuits that implement the instructions pertaining to data transfer operation and data conversion from one form to another
  2. **Execution Unit (EU):** The EU includes the Arithmetic and Logical Unit (ALU) and also the circuits that execute instructions for a program control task such as interrupt, or jump to another set of instructions.

**Types of Processors:**

Processors can be of the following categories:

- General Purpose Processor (GPP)
  - Microprocessor
  - Microcontroller
  - Embedded Processor
  - Digital Signal Processor
  - Media Processor
- Application Specific System Processor (ASSP)
- Application Specific Instruction Processors (ASIPs)
- GPP core(s) or ASIP core(s) on either an Application Specific Integrated Circuit (ASIC) or a Very Large Scale Integration (VLSI) circuit.

**Microprocessor:**

- A microprocessor is a single VLSI chip having a CPU. In addition, it may also have other units such as coaches, floating point processing arithmetic unit, and pipelining units that help in faster processing of instructions.
- Earlier generation microprocessors' fetch-and-execute cycle was guided by a clock frequency of order of ~1 MHz. Processors now operate at a clock frequency of 2GHz.

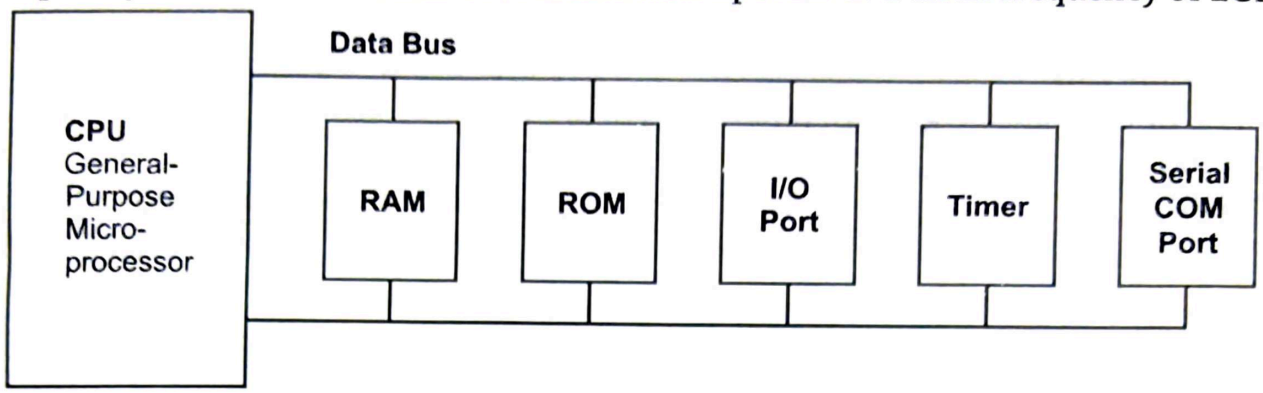


**Fig. 1.6: A Simple Block Diagram of a Microprocessor**

**Microcontroller:**

- A Microcontroller is a single-chip VLSI unit (also called **Microcomputer**) which, although having limited computational capabilities, possesses enhanced input/output capability and a number of on-chip functional units.

| CPU | RAM | ROM |
|---|---|---|
| I/O Port | Timer | Serial COM Port |

- Microcontrollers are particularly used in embedded systems for real-time control applications with on-chip program memory and devices.

## Summary

- Specific set of functionalities or tasks are fulfilled by Embedded Systems.
- An Embedded system is a microcontroller-based computer hardware system having software embedded in it.
- An Embedded system characteristics such as Single-Function, Microprocessors based, Memory, Hardware, Sophisticated Functionality, Real time operation, Portability
- There are two main types of embedded processors: ordinary microprocessors and microcontrollers.
- Real Time Tasks are of different types such as Periodic Tasks, Dynamic Tasks, Critical Tasks, on-Critical Tasks etc.
- The core elements of a microcontroller are: The processor (CPU), Memory and I/O Peripherals etc.
- A microcontroller is embedded inside of a system to control a singular function in a device.
- Microcontrollers are used in a wide array of systems and devices.

## Check Your Understanding

1. Which memory storage is widely used in Embedded Systems?
   (a) Flash Memory (b) EEPROM
   (c) SRAM (d) DRAM
2. How an embedded system communicates with the outside world?
   (a) Memory (b) Output
   (c) Peripherals (d) Input
3. Which of the following is/are type/types of Microprocessor?
   (a) CISC (b) RISC
   (c) EPIC (d) All of the mentioned
4. The process in which the processor constantly checks the status flag is called as _____.
   (a) Polling (b) Inspection
   (c) Reviewing (d) Echoing
5. ANSI stands for _____.
   (a) American National Standards Institute
   (b) American National Standard Interface
   (c) American Network Standard Interfacing
   (d) American Network Security Interrupt
6. There are ____ major types of ports in computers.
   (a) 1 (b) 2
   (c) 3 (d) 4
7. _____ converts digital bits to the analog output.
   (a) PWM (b) DAC
   (c) DEC (d) PMW
8. What is the name of the first microcontroller?
   (a) 8051 (b) 8085
   (c) 8086 (d) 8052
9. What does API stand for?
   (a) Algorithm Program Interface (b) Analog Program Interface
   (c) Application Programming Interface (d) None of the mentioned
10. Embedded System must be _____.
    (a) Microprocessor based (b) Microcontroller based
    (c) Both (a) and (b) (d) None of the above

**Answers**

| | | | | |
|---|---|---|---|---|
| 1. (d) | 2. (c) | 3. (d) | 4. (a) | 5. (a) |
| 6. (b) | 7. (b) | 8. (a) | 9. (c) | 10. (c) |

## Practice Questions

**Q.I Answer the following questions in short.**

1. What is Embedded System?
2. List types of Real time System.
3. What is SoC (System on Chip)?
4. List different Embedded System Components.
5. List different applications of embedded system.
6. What are different applications of SoC?

**Q.II Answer the following questions.**

1. List and explain characteristics of embedded system.
2. Differentiate between general purpose processor and embedded processor.
3. List and explain properties of microcontroller.
4. Explain in details core elements of microcontroller with the help of block diagram.
5. Write short note on Hardware components of embedded system.

**Q.III Define the terms.**

1. Real Time System
2. System-On-Chip
3. Microprocessor
4. Microcontroller
5. Embedded Processor

***

# 2...

# Internet of Things: Concepts

## Learning Objectives...

- To understand the definition and significance of the Internet of Things.
- To understand the Physical and Logical Design of the Internet of Things.
- To discuss the applications of an IoT.

## 2.1 INTRODUCTION

- IoT means Internet of Things, where a word 'thing' can be sensors, software, and other technologies and the purpose of this thing is connecting and exchanging data with other devices and systems over the internet.

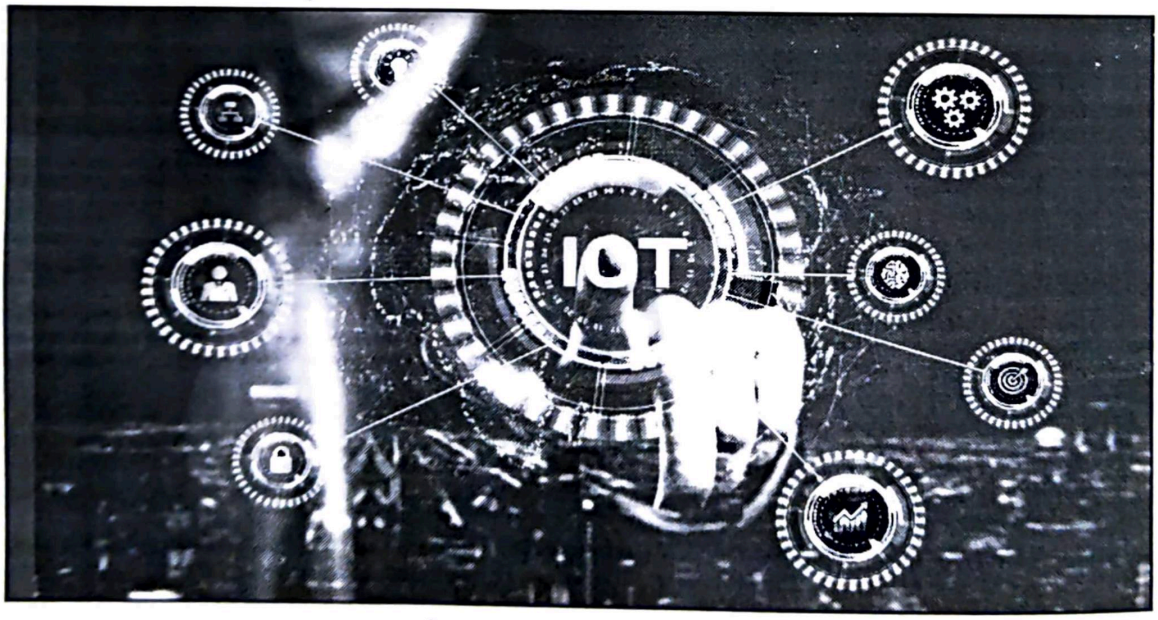


**Fig. 2.1: Concept of IoT**

### 2.1.1 Definition

- The internet of things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people which are provided with unique identifiers UIDs and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

## 2.1.2 Characteristics of the IoT

- The Internet of Things (IoT) is characterized by the following key features:

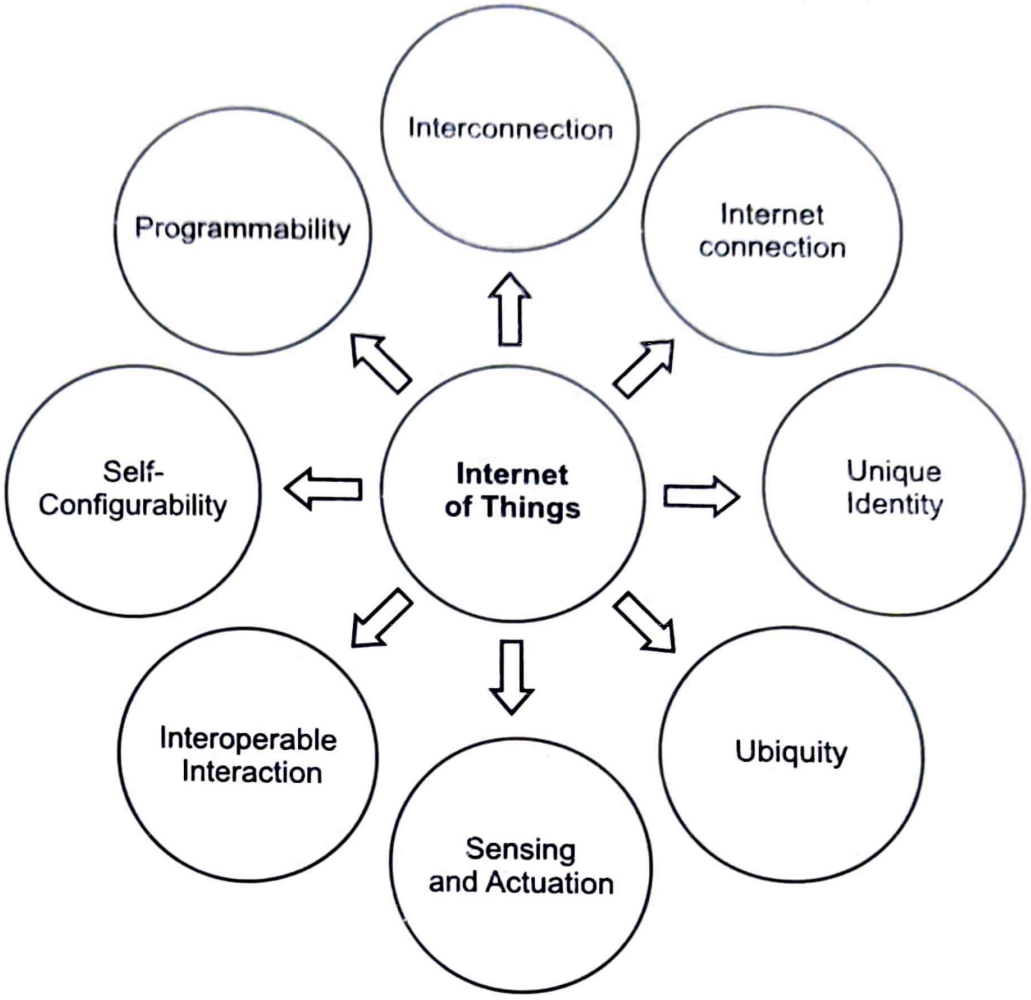


**Fig. 2.2: Characteristics of IoT**

1. **Connectivity:**
   Connectivity is an important requirement of the IoT infrastructure. Things of IoT should be connected to the IoT infrastructure. Anyone, anywhere, anytime can connect, this should be guaranteed at all times. For example, connection between people through internet devices such as mobile phones, and other gadgets, also connection between Internet devices such as routers, gateways, sensors, etc.
2. **Intelligence and Identity:**
   The extraction of knowledge from the generated data is very important. For example, a sensor generates data, but that data will only be useful if it is interpreted properly. Each IoT device has a unique identity. This identification is helpful in tracking the equipment and at times for querying its status.
3. **Scalability:**
   The number of elements connected to the IoT zone is increasing day by day. Hence, an IoT setup should be capable of handling the massive expansion. The data generated as an outcome is enormous, and it should be handled appropriately.

4. **Dynamic and Self-Adapting (Complexity):**

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

5. **Architecture :**

   IoT architecture cannot be homogeneous in nature. It should be hybrid, supporting different manufacturers 'products to function in the IoT network. IoT is not owned by anyone engineering branch. IoT is a reality when multiple domains come together.

6. **Safety:**

   There is a danger of the sensitive personal details of the users getting compromised when all his/her devices are connected to the internet. This can cause a loss to the user. Hence, data security is the major challenge. Besides, the equipment involved is huge. IoT networks may also be at the risk. Therefore, equipment safety is also critical.

7. **Self-Configuring:**

   This is one of the most important characteristics of IoT. IoT devices are able to upgrade their software in accordance with requirements with a minimum of user participation. Additionally, they can set up the network, allowing for the addition of new devices to an already-existing network.

8. **Interoperability:**

   IoT devices use standardized protocols and technologies to ensure that they can communicate with each other and with other systems. Interoperability is one of the key characteristics of the Internet of Things (IoT). It refers to the ability of different IoT devices and systems to communicate and exchange data with each other, regardless of the underlying technology or manufacturer.

## 2.1.3 Trends in Adoption of IoT

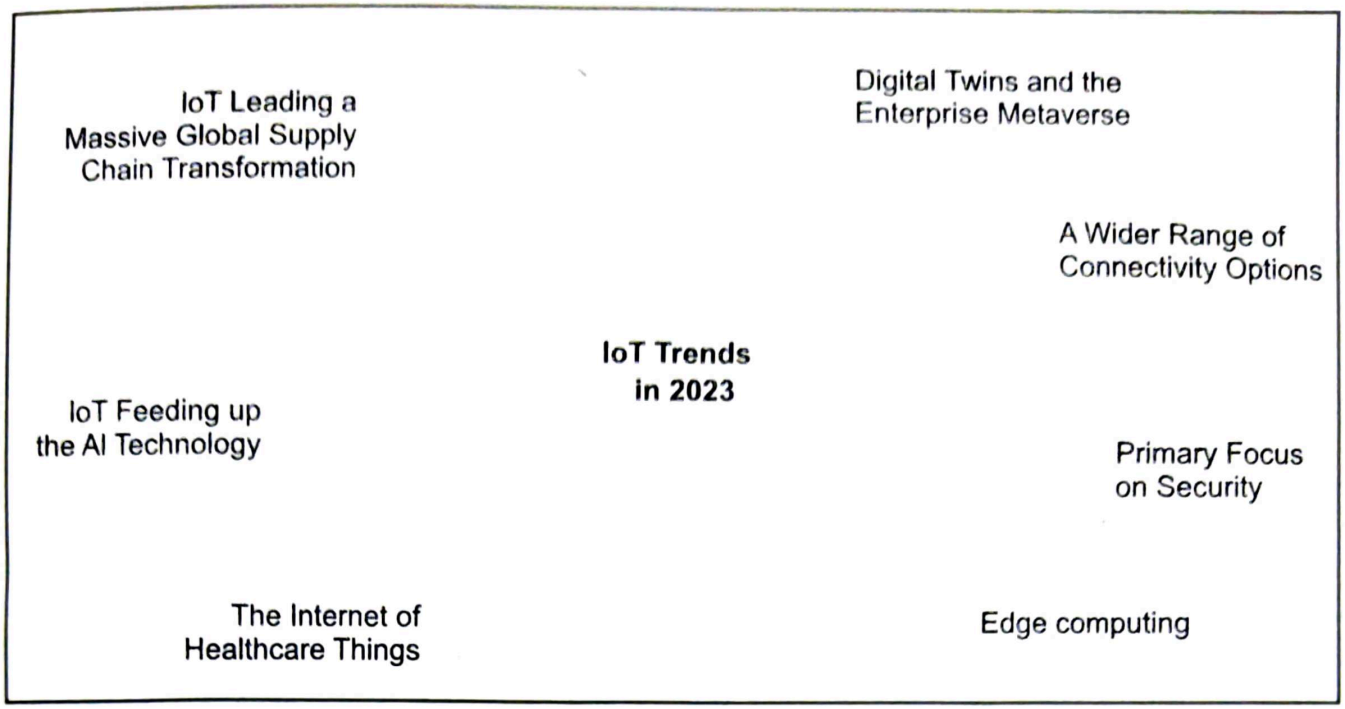


**Fig. 2.3: IoT Trends**

1. **IoT in the Healthcare Sector:**

- The healthcare industry is always at the frontline of IoT adoption. IoT covers most active areas of the healthcare sector – telemedicine, the use of cameras in hospitals and healthcare centres, and specialized medical equipment like insulin pumps, oxygen pumps, heart rate monitors, or fitness bands to monitor lifestyle are enabling doctors to understand the health conditions and lifestyles of the patients better.
- The technology allows medical professionals to diagnose and treat several patients, thus helping in expanding healthcare facilities to remote regions where physical access to hospitals and doctors is difficult.

2. **IoT in the Manufacturing Sector:**

- Manufacturers are gaining a competitive advantage with IoT technology. The role of IoT in the manufacturing industry is no more confined to supervisory control and data acquisition for operating industrial machines. It has gone beyond the boundaries.
- The access to a large volume of data enables manufacturers to take timely decisions regarding asset maintenance. The sensor alerts help them check equipment accuracy or remove it from the manufacturing process until repaired. It allows manufacturing companies to improve asset performance and minimize operating costs.

3. **IoT in Security Management:**

- With the increasing innovation of IoT devices, the risk associated with them is also increasing. IoT devices provide access points to personal networks that can be hacked and exploited by unauthorized users. However, things have started to change.

- The IoT security tools help avoid breaches, risks, and fix network vulnerabilities. Implementation of block chain technologies for protecting networks in banks and financial institutions or using cyber security tools to address security issues are needs of the hour. This IoT trend is a necessity, not just a requirement.

4. **IoT in the Retail Sector:**

- It is a fantastic experience when you choose to entrust a robot to be your customer service representative. A robot rolls up to you and intercepts and asking whether you need help to find things. So, if walked into a store searching for a specific sunglass that you have seen online, your smartphone can open a map to show the counter where you can find that sunglass in that store. You walk to that location, try the sunglass, and put it in your bag.
- IoT technology allows retail companies to manage inventory, optimize the supply chain, reduce operational costs and improve customer experience.

## 2.2 IoT DEVICES

### 2.2.1 Definition

- IoT devices are hardware devices, such as sensors, gadgets, appliances and other machines that collect and exchange data over the Internet. They are programmed for certain applications and can be embedded into other IoT devices.
- For example, an IoT device in your car can identify the traffic ahead and send out a message automatically to the person you are about to meet of your impending delay.

### 2.2.2 Examples of IoT Devices

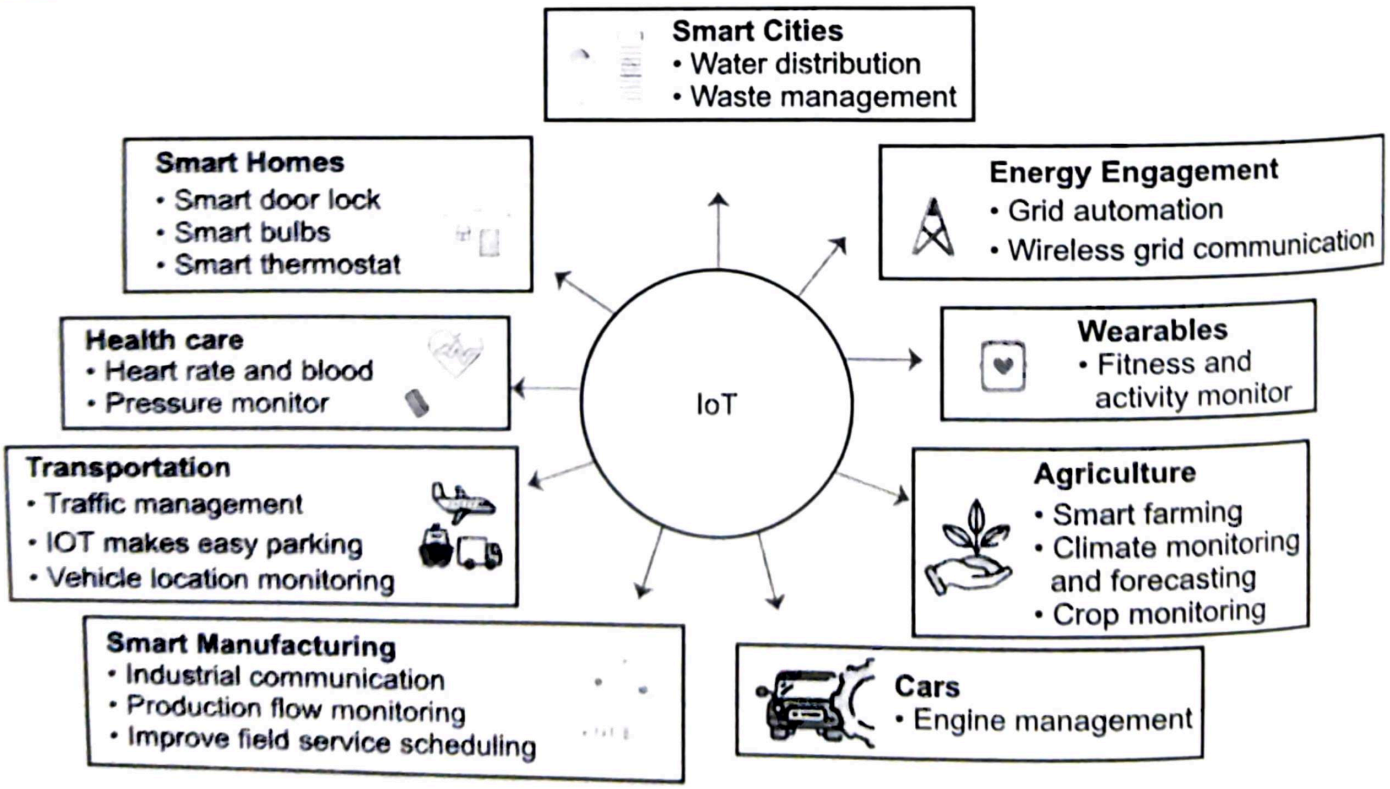


**Fig. 2.4: Examples of IoT Devices**

- Some examples of IoT devices are shown below:

1. **Smart Homes:**

- The key driver behind smart and secure homes is IoT. A variety of sensors, lights, alarms and cameras (all of which can be controlled from a smartphone) are connected via IoT to provide 24x7 security. Smart home security cameras provide alerts and peace of mind

2. **Health Care:**

- Activity trackers are sensor devices that can monitor and transmit key health indicators in real-time. You can track and manage your blood pressure, appetite, physical movement and oxygen levels.

3. **Transportation:**

- IoT solutions for transportation meet a growing range of needs, in a variety of operating conditions. Some examples are given below:
  (i) **Traffic management**: IoT applications for urban traffic management improve both safety and traffic flow, and help cities get the maximum value from their infrastructure spending.
  (ii) **Public Transportation**: Transit IoT applications enable transit agencies to operate more efficiently while improving the passenger experience with amenities such as informational signage and high-speed Internet connectivity.
  (iii) **Electric Vehicles and EV Charging:** The number of electric cars and EV charging stations are growing rapidly. The entire EV infrastructure will rely on IoT connectivity for system maintenance, payment processing and more.
  (iv) **Railways**: IoT solutions support both light rail and heavy commercial rail systems, and this is leading the way with high-performance 5G mobile access routers for reliable and secure high-speed communications and geo-positioning — even in tunnels and urban canyons.
  (v) **Trucking/Logistics:** Fleet managers can track vehicle analytics, reduce the need for truck rolls, and automate processes to save operational costs - including the monitoring and reporting of truck refrigeration.

4. **Smart Manufacturing:**

- Smart manufacturing is the process of using IoT technology and automation in manufacturing processes to ensure that productivity is continually being optimized. Using Indoor Intelligence to accurately track assets and help ensure worker safety, increase productivity, and improve security on your production floor is key to activating all the benefits that come from transforming your manufacturing and warehousing facilities into smart spaces

5. **Smart Cities:**

- IoT has the potential to tame the pressure of urbanization, create new experience for city residents, and make day-to-day living more comfortable and secure. Following are some cases which are helpful to make smart cities.

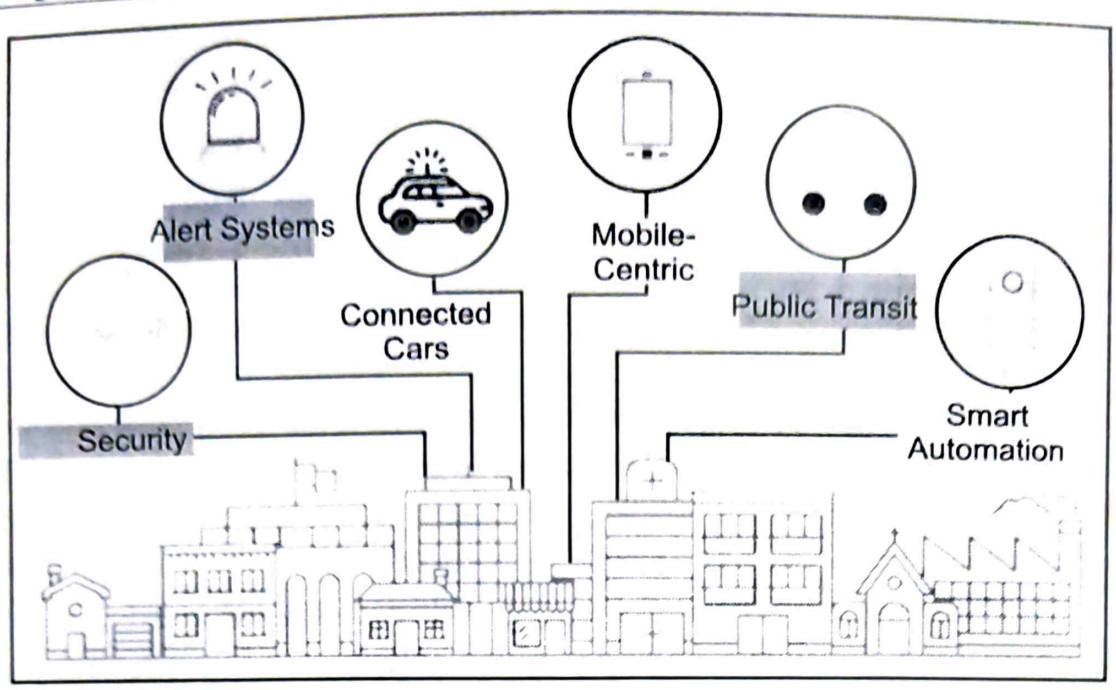


**Fig. 2.5: Smart City**

(i) **Road Traffic:** Smart cities ensure that their citizens get from point A to point B as safely and efficiently as possible. To achieve this, municipalities turn to IoT development and implement smart traffic solutions.

(ii) **Smart Parking:** With the help of GPS data from drivers' smartphones (or road-surface sensors embedded in the ground on parking spots), smart parking solutions determine whether the parking spots are occupied or available and create a real-time parking map.

(iii) **Public Transport:** The data from IoT sensors can help to reveal patterns of how citizens use transport. Public transportation operators can use this data to enhance traveling experience, achieve a higher level of safety and punctuality. To carry out a more sophisticated analysis, smart public transport solutions can combine multiple sources, such as ticket sales and traffic information.

6. **Energy Engagements:**

- IoT-based energy management systems use real-time power consumption data to help optimize the use of electricity, dynamically switch towards more cost and resource-efficient regimes, and work out effective and sustainable energy consumption strategy based on usage patterns.

7. **Wearable:**

- The devices are hands-free gadgets with practical uses, powered by microprocessors and enhanced with the ability to send and receive data via the Internet. The rapid adoption of such devices has placed wearable technology at the forefront of the Internet of things (IoT).

8. **Agriculture:**

- IoT in agriculture uses robots, drones, remote sensors, and computer imaging combined with continuously progressing machine learning and analytical tools for monitoring crops, surveying, and mapping the fields, and providing data to farmers for rational farm management plans to save both time and money.

9. **Cars:**

- Automotive IoT refers to a complex system of devices (e.g., sensors, cameras, GPS trackers) that are connected to the cloud and provide real-time data that enables optimization of the car manufacturing process as well as more efficient transport management.

## 2.2.2 IoT Devices vs. Computers

- One big difference between IoT devices and computers is that the main function of IoT devices is not to compute (not to be a computer) and the main function of a computer is to compute functions and to run programs.

**Difference between IoT Devices and Computers:**

**Table 2.1: IoT Devices and Computers**

| Sr. No. | IoT Devices | Computers |
|---|---|---|
| 1. | IoT devices are special-purpose devices. | Computers are general-purpose devices. |
| 2. | IoT devices can do only a particular task for which it is designed. | Computers can do so many tasks. |
| 3. | The hardware and software built-in in the IoT devices are streamlined for that particular task. | The hardware and software built-in in the computers are streamlined to do many tasks (such as calculation, gaming, music player, etc.) |
| 4. | IoT devices can be cheaper and faster at a particular task than computers, as IoT devices are made to do that particular task. | A computer can be expensive and slower at a particular task than an IoT device. |

## 2.3 BASIC BUILDING BLOCKS

- Four things form basic building blocks of the IoT system – Sensors, Processors, Gateways, and Applications. Each of these nodes has to have its own characteristics in order to form a useful IoT system.

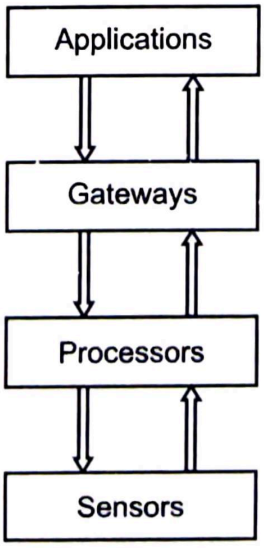


**Fig. 2.6: Basic Building Blocks of IoT System**

1. **Sensors:**

- Sensors form the front end of the IoT devices. These are the so-called "Things" of th system. Their main purpose is to collect data from its surroundings (sensors) or giv out data to its surrounding (actuators).
- These have to be uniquely identifiable devices with a unique IP address so that the can be easily identifiable over a large network.
- These have to be active in nature which means that they should be able to collect rea time data. These can either work on their own (autonomous in nature) or can be mad to work by the user depending on their needs (user-controlled).
- Examples of sensors are Gas sensor, Water quality sensor, Moisture sensor, etc.

2. **Processors:**

- Processors are the brain of the IoT system. Their main function is to process the dat captured by the sensors and process them so as to extract the valuable data from th enormous amount of raw data collected. In a word, we can say that it give intelligence to the data.
- Processors mostly work on real-time basis and can be easily controlled by applications. These are also responsible for securing the data – that is performing encryption and decryption of data.
- Embedded hardware devices, microcontroller, etc. are the ones that process the dat because they have processors attached to it.

3. **Gateways:**

- Gateways are responsible for routing the processed data and send it to proper locations for its (data) proper utilization.
- In other words, we can say that gateway helps in to and fro communication of the data. It provides network connectivity to the data. Network connectivity is essential for any IoT system to communicate.
- LAN, WAN, PAN, etc. are examples of network gateways.

4. **Applications:**

- Applications form another end of an IoT system. Applications are essential for proper utilization of all the data collected.
- These cloud-based applications which are responsible for rendering the effective meaning to the data collected. Applications are controlled by users and are a delivery point of particular services.
- Examples of applications are home automation apps, security systems, industrial control hub, etc.

## 2.4 PHYSICAL DESIGN OF IoT

- The physical design of an IoT system is referred to as the Things/Devices and protocols that are used to build an IoT system. All these things/Devices are called Node Devices and every device has a unique identity that performs remote sensing, actuating and monitoring work and the protocols that are used to establish communication between the Node devices and servers over the internet

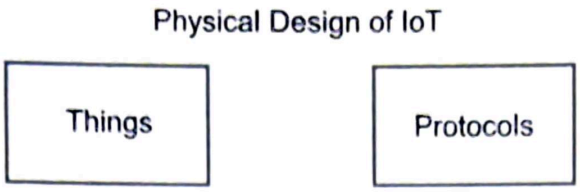


**Fig. 2.7: Physical Design of IoT**

## 2.4.1 Things/Devices in IoT

- Things/Devices are used to build a connection, process data, provide interfaces, provide storage, and provide graphics interfaces in an IoT system. All these generate data in a form that can be analyzed by an analytical system and program to perform operations and used to improve the system.

**IoT Protocols:**

- IoT protocols help to establish communication between IoT Device (Node Device) and Cloud based Server over the Internet. It helps to send commands to IoT Device and received data from an IoT device over the Internet. By the figure given below, you can understand which protocols used.

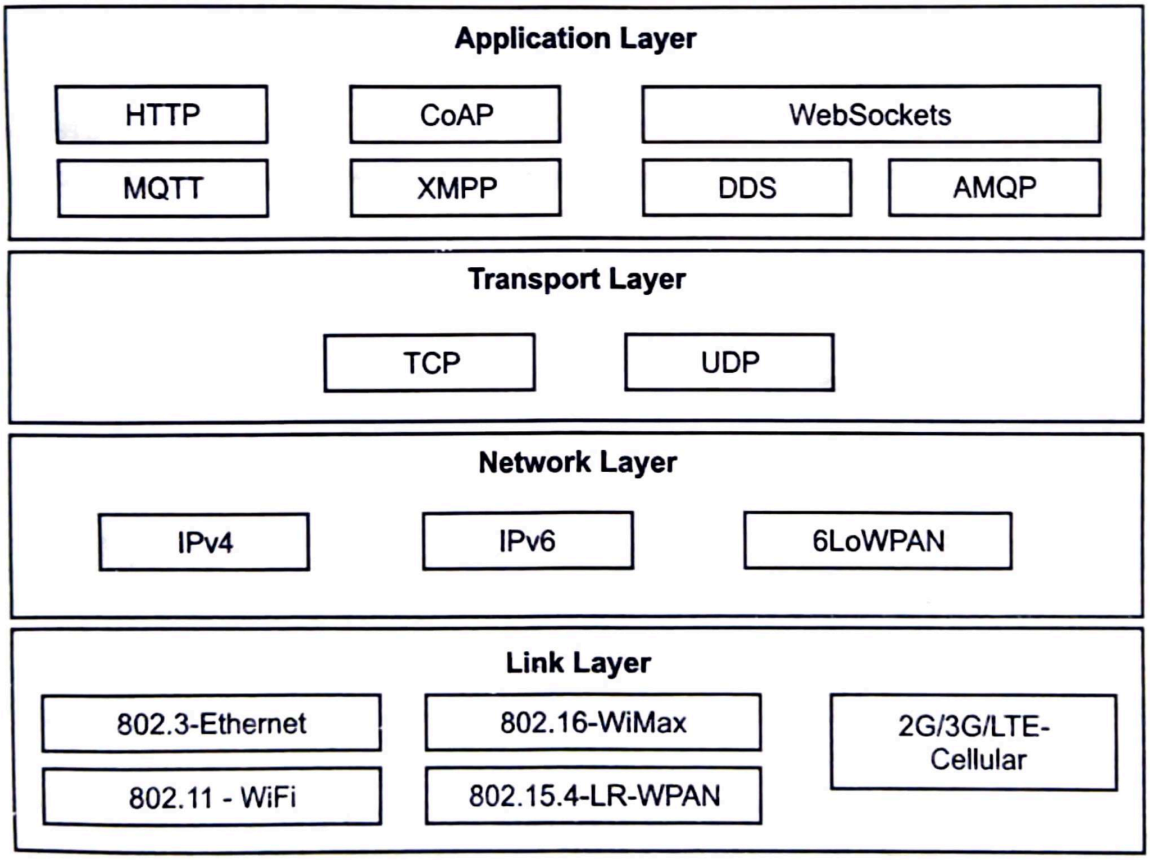


**Fig. 2.8: IoT Protocols used in Physical Design**

## 2.4.2 Interoperability of IoT Devices

- **Interoperability** is the ability of two or more devices, systems, platforms or networks to work in conjunction. Interoperability enables communication between heterogeneous devices or system in order to achieve a common goal. However, the

current devices and systems are fragmented with respect to the communication technologies, protocols, and data formats. This diversity makes it difficult for devices and systems in the IoT network to communicate and share their data with one another. The utility of IoT network is limited by the lack of interoperability.

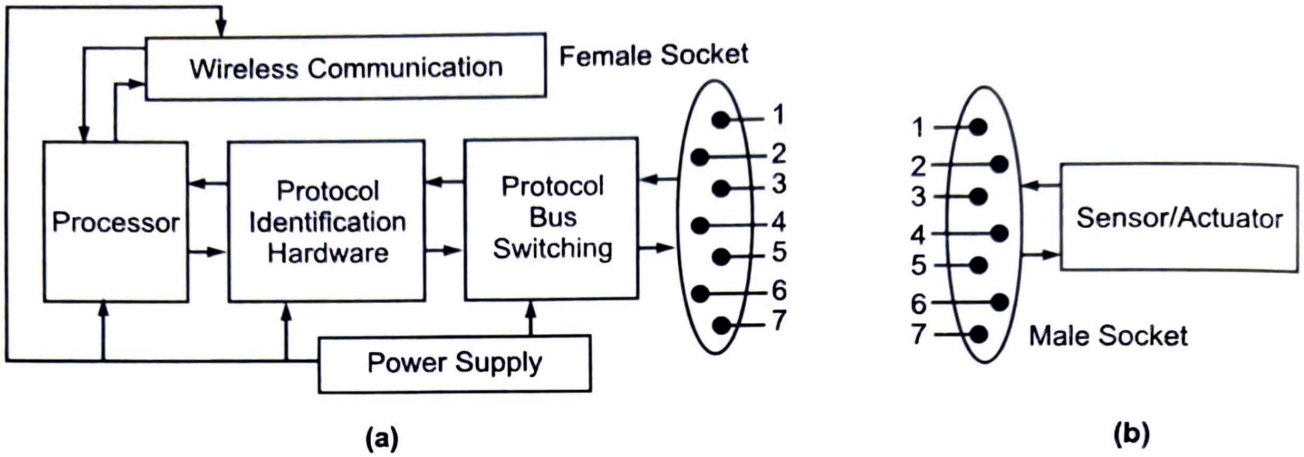


**Fig. 2.9: Interoperability of IoT Devices**

- This problem is sorted with the Leveraging Reliable Cross-domain Communication across IoT Networks. e.g. plug-and-play (PnP) solution. The proposed PnP solution, named SensPnP, is the combination of embedded hardware and firmware that has the capability of integrating third-party embedded sensors with the IoT devices without any prior information about the sensors and the Internet. The architecture of a PnP-enabled IoT device, which supports heterogeneous embedded peripheral communication protocols.

## 2.4.3 Sensors and Actuators

### (a) Sensors:

- Sensors are such devices which are used to convert physical quantities, events or characteristics into the electrical signals for the purpose of monitoring and controlling. So sensor takes input from environment and converts into electrical form then fed to the system or controller. Sensor works as an input device. For example, Thermocouple, Photo cell, RTD, LVDT, Strain gauge, Load cell etc. A block diagram for sensor is shown below:

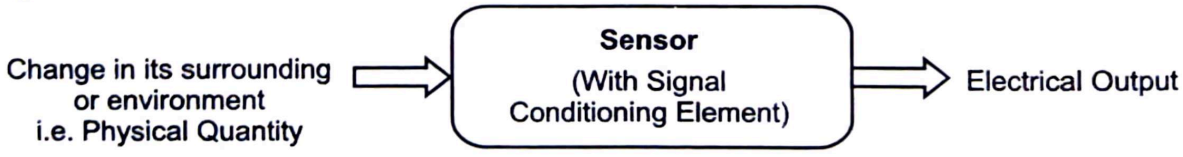


**Fig. 2.10: Block Diagram of a Sensor**

### (b) Actuators:

- Actuators are such devices which deliver physical quantity (like force or motion) to the environment by converting source energy according to control signal received

that can be in electrical form. Here source energy can be pneumatic, hydraulic or electric type and motion produced (by actuator) can be either linear or rotary. Actuator acts as output device. For example, different types of electric motor actuator, heaters, electro pneumatic actuator, electro-hydraulic actuator, magnetic actuator etc. A block diagram of actuator is shown below:

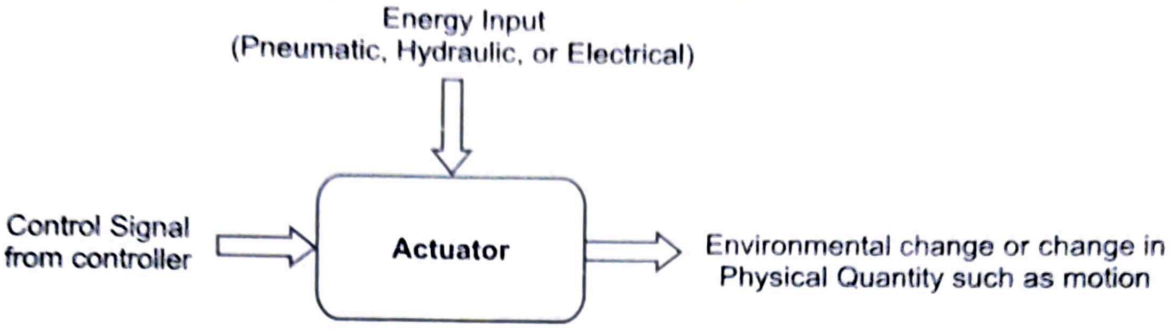


**Fig. 2.11: Block Diagram of an Actuator**

**Difference between Sensor and Actuator:**

- A comparison between sensor and actuators is shown in the Table 2.2.

**Table 2.2: Difference between Sensor and Actuator**

| Sr. No. | Sensor | Actuator |
|---|---|---|
| 1. | Sensor converts physical quantities and characteristics into electrical signals. | Actuator converts electrical signals into physical action such as force and motion. |
| 2. | It acts as an input device in any control system and placed in input port. | It acts as an output device in a control system and placed in output port. |
| 3. | Sensor takes input from environment and senses surroundings condition. | Actuator takes input from output signal conditioning unit of system. |
| 4. | Sensor gives output to input signal conditioning unit of system to convert into electrical form. | It gives output to environment and makes impact on load to control parameters. |
| 5. | It gives information to the system about environment condition to monitor and control. | It accepts command from system to deliver physical action. |
| 6. | Sensors are often used to measure process pressure, temperature, fluid levels, flow, vibration, speed etc. | Actuators are often used to operate control valves, dampers, guide vanes, and to move objects from one place to another, to move conveyor belts in robotic arms movement etc. |
| 7. | Examples: Thermocouple, Photo Cell, RTD, LVDT, Strain Gauge, Load Cell, Hall Sensors, Differential Flow Meters, Speed Probes, PH Meter etc. | Examples: Motor Actuator, Servo Motor, Stepper Motor, Heaters, Electro Pneumatic Actuator, Electro-Hydraulic Actuator, Magnetic Actuator etc. |

- An actuator is a machine component or system that moves or controls the mechanism of the system. Sensors in the device sense the environment, and then control signals are generated for the actuators according to the actions needed to perform.
- The following figure shows what actuators do; the controller directs the actuator based on the sensor data to do the work.

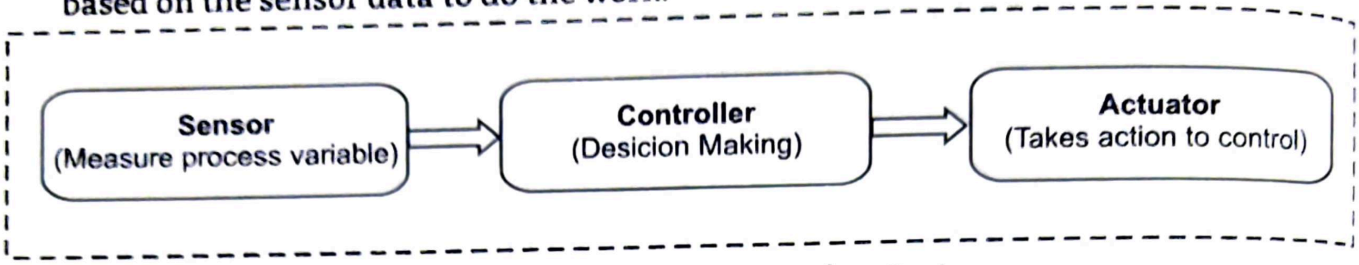


**Fig. 2.12: Sensor and Actuator in a System**

- Sensors in the device sense the environment and fed to the controller, then based on set value, control signal is generated for the actuator to perform actions required to maintain set value. Such control signal is sent to actuator that moves or controls the mechanism or the system. It is to be noted that an actuator needs external energy to perform action.

## 2.4.4 Need of Analog/Digital Conversion

- Analog-to-digital conversion (ADC) is an electronic process in which a continuously variable, or analog, signal is changed into a multilevel digital signal without altering its essential content. An analog-to-digital converter changes an analog signal that's continuous in terms of both time and amplitude to a digital signal that's discrete in terms of both time and amplitude. The analog input to a converter consists of a voltage that varies among a theoretically infinite number of values. Examples are sine waves, the waveforms representing human speech and the signals from a conventional television camera.

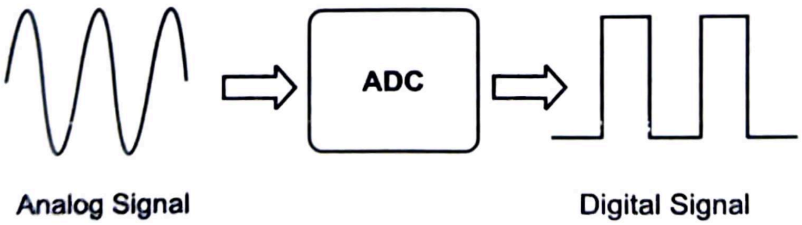


**Fig. 2.13: Analog - to -Digital Conversion**

- Digital signals propagate more efficiently than analog signals, largely because digital impulses are well defined and orderly. They're also easier for electronic circuits to distinguish from noise, which is chaotic. That is the main advantage of digital communication modes.
- The nature of audible sounds generating sine waves that ADC units can sample is not going to change unless there is a quantum change in physics. As such, ADC technology is likely to be embedded in all types of computing devices long into the future. ADC is maybe one of the most significant technological advancements of the past century.

## 2.5 LOGICAL DESIGN OF IoT

- Logical Design of Internet of Things(IoT) includes the following blocks:
  (i) IoT Functional Blocks
  (ii) IoT Communication Models
  (iii) IoT Communication APIs

**(i) IoT Functional Blocks:**

- IoT systems include several functional blocks such as Devices, Communication, Security, Services, and Application. The functional blocks provide sensing, identification, actuation, management, and communication capability. These functional blocks consist of devices that handle the communication between the server and the host, enables monitoring control functions, manage the data transfer, secure the IoT system using authentication and different functions, and provide an interface for controlling and monitoring various terms.

**(ii) IoT Communication Models:**

- In the Internet of Things system, there are multiple kinds of models available that is used for communicating between the system and server such as:
  - Request-response Model
  - Push-pull Model
  - Publish-subscribe Model
  - Exclusive Pair Model

**(iii) IoT Communication APIs:**

- In the IoT, APIs are used to communicate between the server and system. Some API's include:
  - REST-based communication APIs
  - Client-server
  - Stateless
  - Cacheable
  - Web socket based communication API
- An IoT system includes a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication and management.

### 2.5.1 IoT Functional Blocks

- **Device**: An IoT system includes of devices that provide sensing, actuation and monitoring and control functions.
- **Communication**: Handles the communication for the IoT system.
- **Services**: Services for device monitoring, device control service, data publishing services and services for device discovery.

- **Management**: This block provides various functions to manage the IoT system.
- **Security**: This block secures the IoT system and by providing functions such as authentication, authorization, message and content integrity, and data security.
- **Application**: This is an interface that the users can use to control and monitor various aspects of the IoT system. Application also allows users to view the system status and view or analyze the processed data.

## 2.5.2 IoT Enabling Technologies

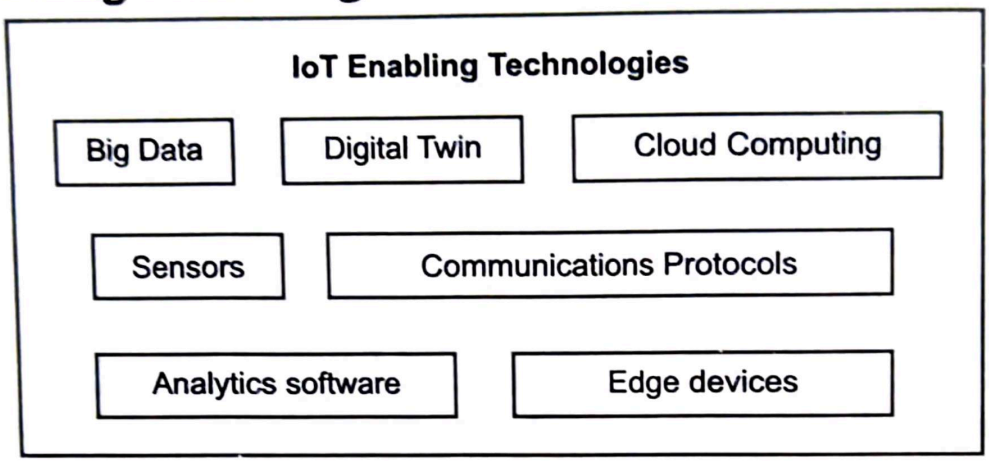


**Fig. 2.14: IoT Enabling Technologies**

Some of the technologies are enlisted below:

1. **Big Data:**

- Big Data analytics is the process of collecting, organizing and analyzing large sets of data (called Big Data) to discover patterns and other useful information. Big Data analytics can help organizations to better understand the information contained within the data and will also help identify the data that is most important to the business and future business decisions. Analysts working with Big Data typically want the knowledge that comes from analyzing the data.
- Some examples of Big Data generated by IoT systems are described as follows:
  - Sensor data generated by IoT system such as weather monitoring stations.
  - Machine sensor data collected from sensors embedded in industrial and energy systems for monitoring their health and detecting Failures.
  - Health and fitness data generated by IoT devices such as wearable fitness bands
  - Data generated by IoT systems for location and tracking of vehicles
  - Data generated by retail inventory monitoring systems

2. **Digital Twin:**

- John Vickers, manager of NASA's National Center for Advanced Manufacturing introduced the concept of Digital TWIN in 2003. This concept defines the digital copy of a physical asset that grows in a virtual environment over the physical asset's lifetime. We know that sensors within the object collect real-time data, and a set of

models forming the digital twin is updated with all of the same information. Hence, an inspection of the digital twin would tell the same information as a physical inspection of the smart object itself, remotely. The digital twin of the smart object is used to not only optimize operations of the smart object through reduced maintenance costs and downtime but to improve the next generation of its design

**3. Cloud Computing:**

- Cloud computing is a trans-formative computing model that involves delivering applications and services over the Internet Cloud computing involves provisioning of computing, networking and storage resources on demand and providing these resources as metered services to the users, in a "pay as you go" model. Cloud computing resources can be provisioned on demand by the users, without requiring interactions with the Cloud Service Provider. The process of provisioning resources is automated. Cloud computing resources can be accessed over the network using standard access mechanisms that provide platform independent access through the use of heterogeneous client platforms such as the workstations, laptops, tablets and smartphones.
- Cloud computing provides three service models. These are very essential for the IoT because they allow any user with a browser and an internet connection to transform smart object data into actionable intelligence.

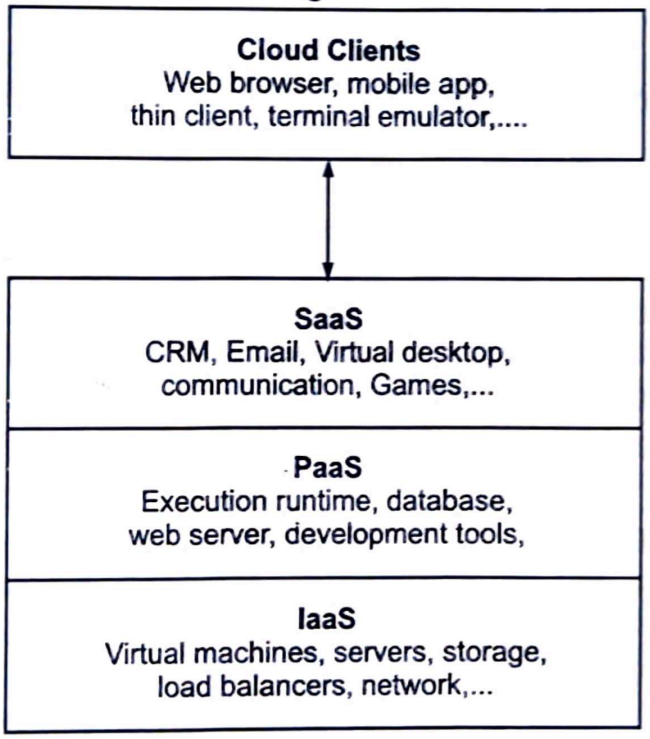


**Fig. 2.15: Models of Cloud Computing**

**4. Sensors:**

- Sensors are capable in detecting events or changes in a specific quantity (e.g. pressure) communicating the event or change data to the cloud (directly or via a gateway) and, in some circumstances, receiving data back from the cloud (e.g. a control command) or communicating with other smart objects.

5. **Communication Protocols:**

- Communication protocols form the backbone of IoT systems and enable network connectivity and coupling to applications. Communication protocols allow devices to exchange data over the network. Multiple protocols often describe different aspects of a single communication. A group of protocols designed to work together are known as a protocol suite; when implemented in software they are a protocol stack.
- Internet communication protocols are published by the Internet Engineering Task Force (IETF). The IEEE handles Wired and Wireless networking, and the International Organization for Standardization (ISO) handles other types. The ITU-T handles telecommunication protocols and formats for the public switched telephone network (PSTN). As the PSTN and Internet converge, the standards are also being driven towards convergence.

## 2.5.3 IoT Levels and Deployment Templates

- Developing an IoT Level Template system consists of the following components:
  1. **Device**: These may be sensors or actuators capable of identifying, remote sensing or monitoring.
  2. **Resources**: These are software components on IoT devices for accessing and processing, storing software components or controlling actuators connected to the device. Resources also include software components that enable network access.
  3. **Controller Service**: It is a service that runs on the device and interacts with web services. The controller service sends data from the device to the web service and receives commands from the application via web services for controlling the device.
  4. **Database**: Stores data generated from the device.
  5. **Web Service**: It provides a link between IoT devices, applications, databases, and analysis components.
  6. **Analysis Component**: It performs an analysis of the data generated by the IoT device and generates results in a form which are easy for the user to understand.
  7. **Application**: It provides a system for the user to view the system status and view product data. It also allows users to control and monitor various aspects of the IoT system.

## 2.5.4 Applications of IoT

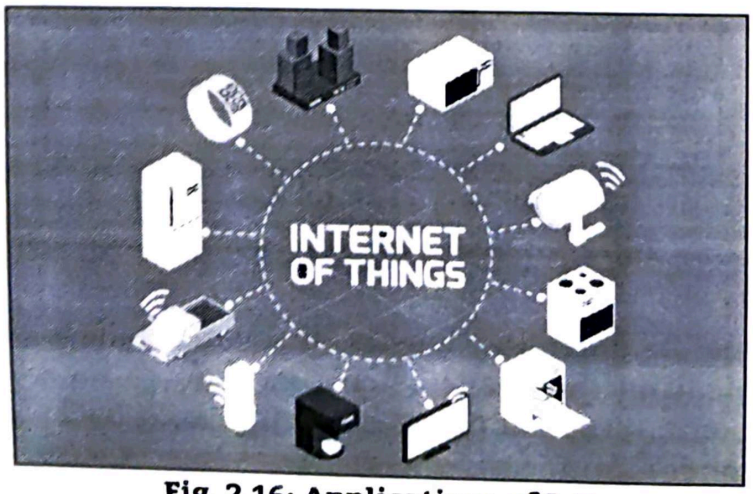


**Fig. 2.16: Applications of IoT**

1. **IoT Applications in Agriculture:**

- IoT makes monitoring and management of micro-climate conditions in a reality for indoor planting which in turn increases production. For outside planting, devices using IoT technology can sense soil moisture and nutrients, in combination with weather data, better control smart irrigation and fertilizer systems. If the sprinkler systems distribute water only when needed. This avoids wasting a precious resource.

2. **IoT Applications in Health Sector:**

- The use of wearables or sensors connected to patients, allows doctors to monitor a patient's condition outside the hospital and in real-time. Through continuously monitoring certain metrics and automatic alerts on their vital signs, the Internet of Things helps to improve the care for patients and the prevention of fatal events in high-risk patients.

3. **IoT Applications in Traffic Monitoring:**

- The Internet of things can be very useful in the management of vehicular traffic in large cities, contributing to the concept of smart cities. When we use our mobile phones as sensors, which collect and share data from our vehicles through applications such as Google Maps, we are using the Internet of Things to inform us and at the same time contribute to traffic monitoring, showing the conditions of the different routes, and feeding and improving the information on the different routes to the same destination, distance, estimated time of arrival.

4. **IoT Applications in Home Automation:**

- Home Automation is one of the best examples of IoT. Smart homes or IoT-based home automation systems are becoming popular day by day. In a smart home, consumer electronic gadgets such as Lights, Fans, Air-conditioners, etc. can be connected to each other through the internet. This interconnection enables the user to operate these devices from a distance. A smart home is capable of lighting control, energy management, expansion, and remote access. Currently, this application of IoT is not utilized at a large scale because the installation cost is too high, which makes it difficult for a majority of people to afford it. However, home automation holds quite a favorable future.

5. **IoT Applications in Process Automation:**

- In the manufacturing industry, performing reoccurring tasks manually such as label wrapping, packaging, etc., is difficult and is prone to human errors; therefore, automation comes into play. For instance, take the example of a Cold Drink Manufacturing Industry. Here, manufacturing machines and conveyor belts are required to be interconnected in order to share information, status, and data. This interconnection is IoT dependent. The status of the manufactured product and the machine health report is sent to the manufacturer at regular intervals in order to

identify the faults in advance. An IoT equipped industry is beneficial as it elevates the production speed and maintains the uniform quality of the product throughout the production. It also helps to make the workplace more efficient and safe by reducing human error.

6. **IoT Applications in Disaster Management:**

- IoT helps in the prediction and management of natural disasters. For instance, take the example of forest fires. Various sensors can be installed around the boundaries of the forests to avoid the chaos and destruction caused by a forest fire. These sensors continuously monitor the temperature and carbon content in the region. A detailed report is regularly sent to a common monitoring hub. In case of a forest fire, an alert is sent to the control room, police station, and fire brigade. Therefore, IoT helps in staying prepared and respond quickly in case of emergency.

7. **IoT Applications in Water and Waste Management:**

- Water treatment units for water recycling have helped immensely in water management. The use of IoT makes them more efficient as one can see how much wastewater is being produced, the consumption of water in a specific area, and changes in waste production over time.
- With a smart waste management system, it gets handy to predict waste quantities in a specific location. Accordingly, authorities can plan how to process it, when to clear it, and how to interpret the data for future planning. City upgrade projects will pick up the pace when analytics solutions and data obtained are combined, which would bring about transformations, etc.

## Summary

- The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people which are provided with unique identifiers UIDs and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.
- Four things form basic building blocks of the IoT system – Sensors, Processors, Gateways, and Applications.
- Logical Design of Internet of Things (IoT) Comprises with the following blocks: IoT Functional Blocks,
- IoT Communication Models and IoT Communication APIs.
- Sensors are used to convert physical quantities, events or characteristics into the electrical signals for the purpose of monitoring and controlling.
- Actuators are such devices which deliver physical quantity (like force or motion) to the environment by converting source energy according to control signal received that can be in electrical form.

## Check Your Understanding

1. What is IoT?
   (a) Network of physical objects embedded with sensors
   (b) Network of virtual objects
   (c) Network of objects in the ring structure
   (d) Network of sensors
2. Which of the following is not an application of IoT?
   (a) BMP280 (b) Smart home
   (c) Smart city (d) Self-driven cars
3. Which of the following is not a fundamental component of an IoT system?
   (a) Sensors (b) Connectivity and data processing
   (c) User interface (d) Transformer
4. What is the full form of IIOT?
   (a) Index Internet of Things (b) Incorporate Internet of Things
   (c) Industrial Internet of Things (d) Intense Internet of Things
5. Which layer is used for wireless connection in IoT devices?
   (a) Application layer (b) Network layer
   (c) Data link layer (d) Transport layer
6. Which of the following is used to capture data from the physical world in IoT devices?
   (a) Sensors (b) Actuators
   (c) Microprocessors (d) Microcontrollers
7. Which of the following protocol is used to link all the devices in the IoT?
   (a) HTTP (b) UDP
   (c) Network (d) TCP/IP
8. Which of the following is not an actuator in IoT?
   (a) Stepper motor (b) A fan
   (c) An LED (d) Arduino
9. Identify the incorrect advantage of IoT.
   (a) Reduced Waste (b) Enhanced Data collection
   (c) Improve customer engagement (d) Security
10. In which of the following terms in resolution expressed?
    (a) bits (b) bytes
    (c) nibble (d) words

**Answers**

| | | | | |
|---|---|---|---|---|
| 1. (a) | 2. (a) | 3. (d) | 4. (c) | 5. (c) |
| 6. (a) | 7. (d) | 8. (d) | 9. (d) | 10. (a) |

## Practice Questions

**Q. I Answer the following question in short.**

1. Define IoT devices.
2. Write any two characteristics of IoT.
3. Write a basic building Block of IoT.
4. Write the need of Analog to Digital Conversion.
5. Which components are required in deployments of IoT?

**Q.II Answer the following questions.**

1. Write difference between IoT Devices and Computers.
2. Write difference between Sensors and Actuators.
3. Write a short note on Interoperability of IoT Devices.
4. Describe any four applications of IoT.
5. Explain trends in Adoption of IoT.

**Q.III Define the terms.**

1. IoT
2. Big Data
3. Sensors
4. Actuators
5. Smart City

***

# 3...

# Introduction to IoT Design Methodology

## Learning Objectives...

- To understand the design stage in perspective of security of the IoT.
- To understand the challenges to Secure IoT.
- To understand key elements of IoT Security for the designing of tamper-proof product.

## 3.1 INTRODUCTION

- Designing IoT systems can be a complex and challenging task as these systems involve interactions between various components such as IoT devices and network resources, web services, analytics components, application and database servers.
- IoT system designers often tend to design IoT systems keeping specific products/services in mind. So that designs are tied to specific product/service choices made. But it make updating the system design to add new features or replacing a particular product/service choice for a component becomes very complex, and in many cases may require complete redesign of the system.
- Here we discuss a generic design methodology for IoT system design which is independent of specific product, service or programming language.
- IoT systems designed with the proposed methodology have reduced design, testing and maintenance time, better interoperability and reduced complexity.

## 3.2 IoT PLATFORMS DESIGN METHODOLOGY

- IoT Design Methodology includes:
  - Purpose and Requirements Specification
  - Process Specification
  - Domain Model Specification

- Information Model Specification
- Service Specification
- IoT Level Specification
- Functional View Specification
- Operational View Specification
- Device and Component Integration
- Application Development

**Step 1: Purpose and Requirements Specification:**

- The first step in IoT system design methodology is to define the purpose and requirements of the system. In this step, the system purpose, behaviour and requirements (such as data collection requirements, data analysis requirements, system management requirements, data privacy and security requirements, user interface requirements) are captured.
- Applying this to the example of a Smart Home Automation System, the purpose and requirements for the system may be described as follows:
  - **Purpose:** A Home Automation System that allows controlling of the lights in a home remotely using a web application.
  - **Behaviour:** The Home Automation System should have auto and manual modes. In the auto mode, the system measures the light level in the room and switches on the light when it gets dark. In the manual mode, the system provides the option of manually and remotely switching on/off the light.
  - **System Management Requirement:** The system should provide remote monitoring and control functions.
  - **Data Analysis Requirement:** The system should perform local data analysis.
  - **Application Deployment Requirement:** The application should be deployed locally 18PCSC41-Internet of Things 2020-2021(EVEN) 3 M.Sc.(CS) CS Department-MTNC on the device, but should be accessible remotely.
  - **Security Requirement:** The system should have basic user authentication capability.

**Step 2: Process Specification:**

- The use cases of the IoT system are formally described based on or derived from the purpose and requirements specifications.

**Step 3: Domain Model Specification:**

- The third step in the IoT design methodology is to define the Domain Model. The domain model describes the main concepts, entities and objects in the domain of IoT system to be designed. Domain model defines the attributes of the objects and relationships between objects. Domain model provides an abstract representation of the concepts, objects and entities in the IoT domain, independent of any specific technology or platform.

- The entities, objects and concepts defined in the domain model include:
  - **Physical Entity** : Physical Entity is a discrete and identifiable entity in the physical environment (e.g. a room, a light, an appliance, a car, etc.).
  - **Virtual Entity** : Virtual Entity is a representation of the Physical Entity in the digital world.
- Device provides a medium for interactions between Physical Entities and Virtual Entities. Devices are either attached to Physical Entities or placed near Physical Entities.

**Step 4: Information Model Specification:**

- The fourth step in the IoT design methodology is to define the Information Model. Information Model defines the structure of all the information in the IoT system, for example, attributes of Virtual Entities, relations, etc. Information model does not describe the specifics of how the information is represented or stored.
- To define the information model, we first list the Virtual Entities defined in the Domain Model Information model adds more details to the Virtual Entities by defining their attributes and relations.

**Step 5: Service Specification:**

- The fifth step in the IoT design methodology is to define the service specifications. Service specifications define the services in the IoT system, service types, service inputs/output, service endpoints, service schedules, service preconditions and service effects.
- From the process specification and information model, we identify the states and attributes. For each state and attribute we define a service. These services either change the state or attribute values or retrieve the current values.

**Step 6: IoT Level Specification :**

- The sixth step in the IoT design methodology is to define the IoT level for the system. Based on the requirements, we will select the IoT application deployment level.

**Step 7: Functional View Specification:**

- The Functional View (FV) defines the functions of the IoT systems grouped into various Functional Groups (FGs). Each Functional Group either provides functionalities for interacting with instances of concepts defined in the Domain Model or provides information related to these concepts.
- The Functional Groups (FG) included in a Functional View include:
  - **Device** : The device FG contains devices for monitoring and control. In the home automation example. The device FG includes a single board mini-computer, a light sensor and relay switch(actuator).
  - **Communication** : The communication FG handles the communication for the IoT system. The communication FG includes the communication protocols that form the backbone of IoT systems and enable network connectivity. The

communication FG also includes the communication APIs (such as REST and WebSocket) that are used by the services and applications to exchange data over the network.

**Step 8: Operational View Specification:**

- In this step, various options relating to the IoT system deployment and operation are defined, such as, service hosting options, storage options, device options, application hosting options, etc. Operational View specifications for the home automation example are as follows:
  - **Devices**: Computing device (Raspberry Pi), light dependent resistor (sensor), relay switch (actuator).
  - **Communication APIs (Application Programming Interface)**: REST APIs
  - **Communication Protocols**: Link Layer - 802.11, Network Layer - 1Pv4/1Pv6, Transport TCP,
  - Application - HTTP.

**Step 9: Device and Component Integration:**

- In this step, different devices and components are decided. According to the application, part components are needed to be selected. For example, in Home Automation system components such as LDR Sensor, Relay Switch Actuator, Controller Raspberry Pi Minicomputer is used.

**Step 10: Application Development:**

- Using all the information from previous steps, we will develop the application (code) for the IoT system.
- The application has controls for the mode (auto on or auto off) and the light (on or off). In the auto mode, the IoT system controls the light appliance automatically based on the lighting conditions in the room. When auto mode is enabled the light control in the application is disabled and it reflects the current state of the light. When the auto mode is disabled, the light control is enabled and it is used for manually controlling the light.

## 3.2.1 Basics of IoT Networking

- Internet of Things (IoT) is a network of physical objects or people called "things" that are embedded with software, electronics, network, and sensors that allows these objects to collect and exchange data. The goal of IoT is to extend to internet connectivity from standard devices like computer, mobile, tablet to relatively dumb devices like a toaster.
- IoT makes virtually everything "smart," by improving aspects of our life with the power of data collection, AI algorithm, and networks. The thing in IoT can also be a person with a diabetes monitor implant, an animal with tracking devices, etc.
- The entire IoT process starts with the devices themselves like smartphones, smartwatches, electronic appliances like TV, Washing Machine which helps you to communicate with the IoT platform.

## 3.2.2 Networking Components

In this section, we will learn about four fundamental components of an IoT system:

1. **Sensors/Devices**:

- Sensors or devices are a key component that helps to collect live data from the surrounding environment. All this data may have various levels of complexities. It could be a simple temperature monitoring sensor, or it may be in the form of the video feed.
- A device may have various types of sensors which performs multiple tasks apart from sensing. For example, a mobile phone is a device which has multiple sensors like GPS, Camera but your smartphone is not able to sense these things.

2. **Connectivity:**

- All the collected data is sent to a cloud infrastructure. The sensors should be connected to the cloud using various mediums of communications. These communication mediums include Mobile or Satellite Networks, Bluetooth, Wi-Fi, WAN, etc.

3. **Data Processing**:

- Once that data is collected, and it gets to the cloud, the software performs processing on the gathered data. This process can be just checking the temperature, reading on devices like AC or heaters. However, it can be sometimes also very complex like identifying objects, using computer vision on video.

4. **User Interface**:

- The information needs to be available to the end-user in some way which can be achieved by triggering alarms on their phones or sending them notification through email or text message. The user sometimes might need an interface which actively checks their IoT system. For example, the user has a camera installed in his home. He wants to access video recording and all the feeds with the help of a web server.
- However, it is not always one-way communication. Depending on the IoT application and complexity of the system, the user may also be able to perform an action which may create cascading effects.For example, if a user detects any changes in the temperature of the refrigerator, with the help of IoT technology the user should able to adjust the temperature with the help of their mobile phone.

## 3.2.3 Internet Structure

- The internet is the largest structure of interconnected networks set up to allow computers to communicate with each other globally using standardized communication protocols.

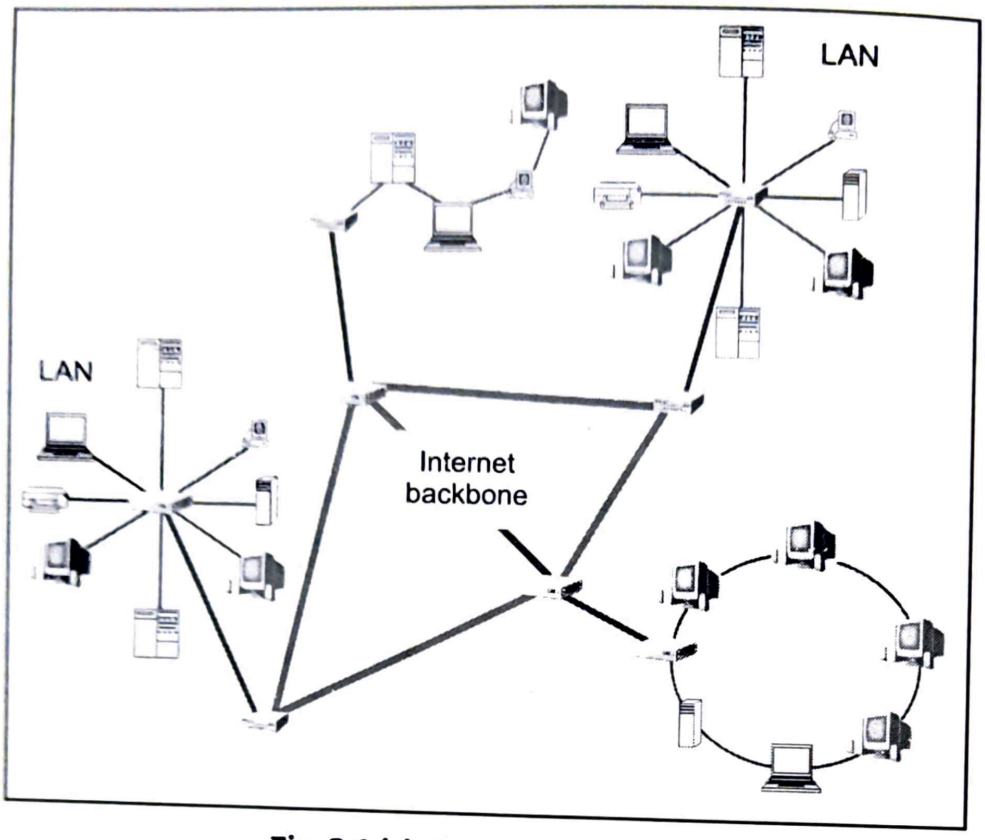


**Fig. 3.1 (a): Internet Structure**

**Internet Structure includes two main parameters:**

1. Internet Address
2. Protocol Stacks and Packets

**1. Internet Address:**

- Computers connected to the internet means that the systems are connected to computers' worldwide network. Therefore, each machine/device has its own or unique address. Addresses of the internet are in the form **"kkk.kkk.kkk.kkk,"** where each **"kkk"** ranges from 0-256. This structure of the internet address is known as an IP address (Internet Protocol). Fig. 3.1(b) describes the connection between two computers using the internet. Both systems have unique IP addresses. However, the internet is a unique object between both systems.

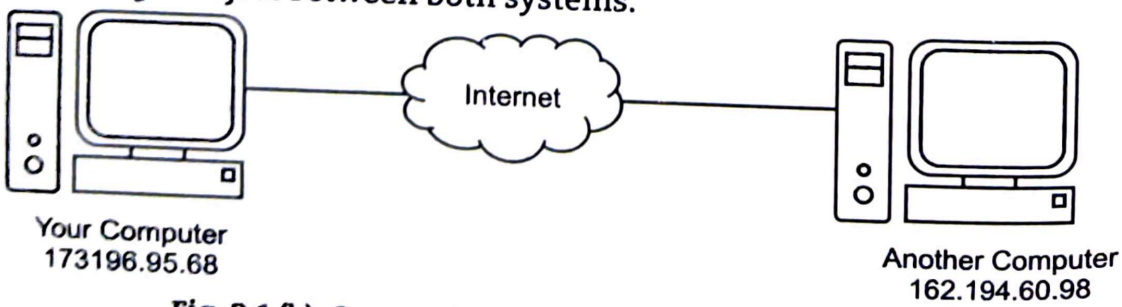


**Fig. 3.1 (b): Connection between two computers**

## 2. Protocol Stacks and Packets:

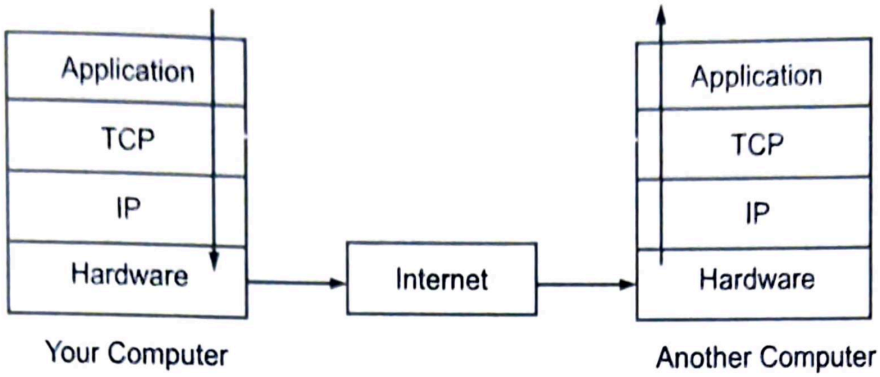


**Fig. 3.1 (c): The path framework related to that message**

- The above figure briefly describes the path framework related to that message from "One computer" to another computer.

**Procedure to send message on the internet:**

1. The message that needs to be sent is written in an application on "First computer" it starts from the top using the protocol stack and moves downward.
2. If the message is large, the stack layer breaks the message into smaller chunks so that data management remains stable. The chunks of data are known as **Packets.**
3. The data from the application layer move towards the TCP/IP layer. The packet of the data is assigned with a port number. In computers, various types of message applications are working at a time. Therefore, it is essential to know which application is sending the message so that it needs to be synced at the reception level (another computer) with the same application. Hence, the message will listen on the same port.
4. After necessary processing at the TCP level, the packets move towards the IP layer. The IP layer provides the destination layer where the message should be received. At this level, message packets retain port number as well as IP address.
5. The hardware layer is responsible for converting alpha/numeric messages into a digital signal and sending the message through the telephone's path.
6. Internet Services Provider (ISP) is also attached to the internet, where the ISP router examines the recipient's address. The next stop of the packet is another router.
7. Eventually, the packets reach another computer. This time packets start from the bottom.
8. As the packets move upwards, the packets' unnecessary data is removed that was helping to reach the destination; this includes IP address and port number.
9. On reaching the stack's top, all the packets are reassembled to form the original message sent by "first computer".

## 3.3 IoT COMMUNICATION MODELS AND IoT COMMUNICATION APIs

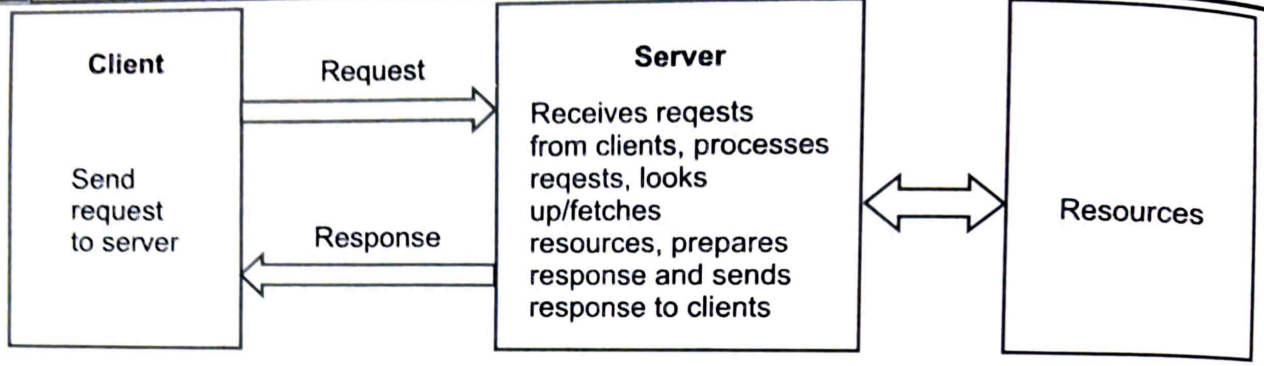


**Fig. 3.2: Request-Response Communication Model**

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

**Request and Response Model :**

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

**IoT Communication Apis**

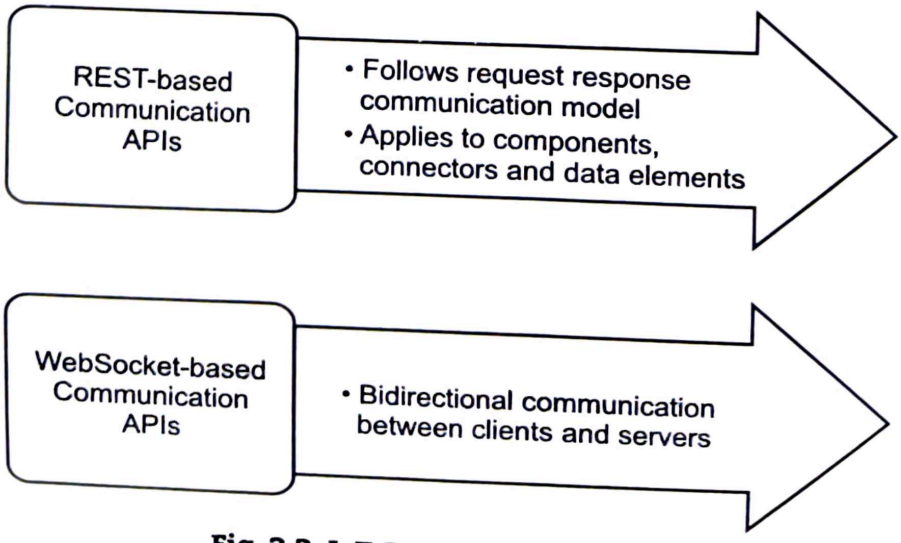


**Fig. 3.3: IoT Communication APIs**

## 3.4 SENSOR NETWORKS

- Today sensors are in all places. For example, sensors are in our phones, workplaces, vehicles, and the environment.
- A sensor network comprises a group of small, powered devices, and a wireless or wired networked infrastructure. They record conditions in any number of environments including industrial facilities, farms, and hospitals. The sensor network connects to the internet or computer networks to transfer data for analysis and use.
- Sensor network nodes co-operatively sense and control the environment. They enable interaction between persons or computers and the surrounding environment.

**Wired vs. Wireless Sensor Networks:**

- Sensor networks can be wired or wireless. Wired sensor networks use Ethernet cables to connect sensors. Wireless Sensor Networks (WSNs) use technologies such as Bluetooth, Cellular, Wi-Fi or Near Field Communication (NFC) to connect sensors.
- WSNs are easier to deploy and maintain and offer better flexibility of devices. With the rapid development of sensors and wireless technologies, WSNs have become a key technology of the IoT. WSNs don't need the physical network infrastructure to be modified.

**Operation of a Sensor Network:**

- Sensor networks typically include sensor nodes, actuator nodes, gateways, and clients. Sensor nodes group inside the sensor field and form networks of different topologies. The following process describes how sensor networks operate:
  - A Sensor node monitors the data collected by the sensor and transmits this to other sensor nodes. During the transmission process, data may be handled by multiple nodes as it reaches a gateway node. The data is then transferred to the Management node.
  - The Management node is managed by the user and determines the monitoring required and collects the monitored data.

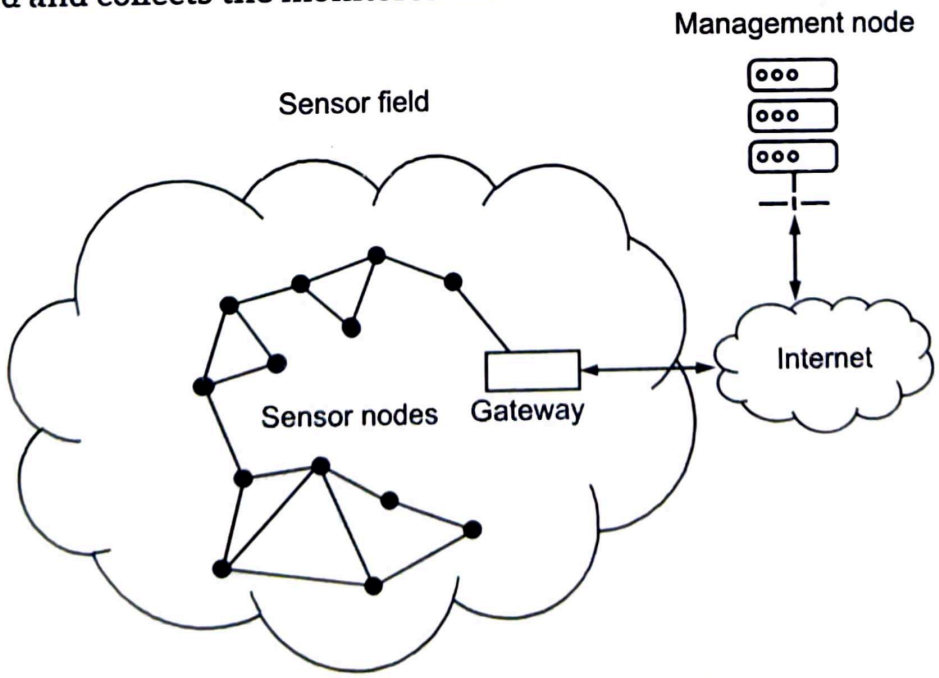


**Fig. 3.4: Working of Sensor Network**

**Sensor Nodes:**

- There are many nodes in a sensor network. These nodes are the detection stations. There is a sensor/transducer, microcontroller, transceiver, and power source:
  - A sensor senses the physical condition, and if there is any change, it generates electrical signals.
  - The signals go to the microcontroller for processing. A central processor sends commands to the transceiver and data is transmitted to a computer.

**Sensors:**

- The sensor is the bond of a sensor network node. Examples of sensors include temperature sensors, accelerometers, infrared detectors, proximity sensors, and motion detectors.

**Sensor Network Topologies:**

- There are four types of Sensor Network Topologies:
  1. Point to Point Network
  2. Star Network
  3. Tree Network
  4. Mesh Network

## 3.5 FOUR PILLARS OF IoT: M2M, SCADA, WSN, AND RFID

### 3.5.1 M2M (Machine to Machine)

- M2M is the connection of two or more than two devices with the internet for data sharing and analytics. IoT and M2M provide remote access for exchanging information among machines without human intervention. The main difference between IoT and M2M is that IoT connects any device to the Internet for better performance, and M2M is the connection of two or more than two devices with the Internet for data sharing and analytics.
- M2M is a system allows machines to communicate with each other and with humans. It is an extension of the Internet of Things (IoT) that allows for easier and more direct access to the data and functionality of connected devices. M2M is a standardized technology for managing and sharing the data and functionality of connected devices. It is used in applications such as metering, automation, tracking and tracing, healthcare, and many others.
- Machine-to-Machine (M2M) communication is the exchange of information between machines. M2M is a network of physical objects capable of capturing information about their state, communicating that information over a network without requiring human interference, and using that information to control their operational behavior. The most widespread form of M2M communication is data exchange between objects in a utility metering context, such as smart electric meters in homes and businesses.

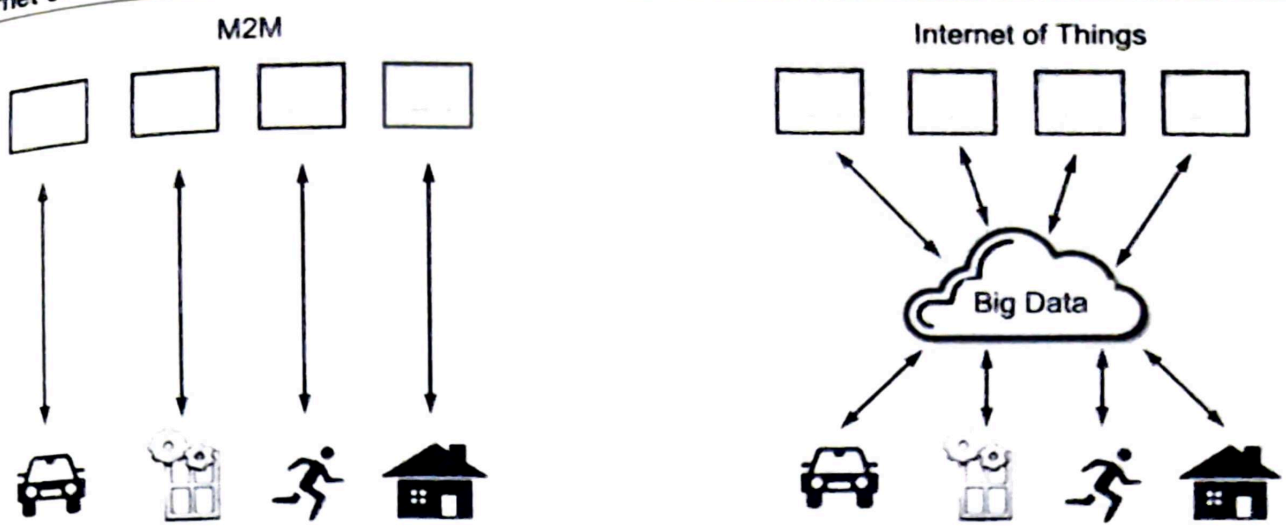


**Fig. 3.5: Pillars of IoT**

**Table 3.1 Differences between IoT and M2M**

| Sr. No. | Basis | IoT | M2M |
|---|---|---|---|
| 1. | *Abbreviation* | Internet of Things. | Machine to Machine. |
| 2. | *Communication* | IoT sensors automation | Communicates directly between machines. |
| 3. | *Connection* | The connection is via using various communication types. | Point to Point Connection. |
| 4. | *Communication protocols* | HTTP, FTP, Telnet, etc. are used. | Communication technology techniques and traditional protocols are used. |
| 5. | *Intelligence* | Objects are responsible for decision making. | Observation of some degree of intelligence. |
| 6. | *Technology* | Hardware and Software based technology. | Hardware-based technology. |
| 7. | *Data Delivery* | Depending on the Internet protocol. | Devices can be connected through mobile or other networks. |
| 8. | *Internet Connection* | Active Internet connection is required. | Devices do not rely on an internet connection. |
| 9. | *Scope* | Many users can connect at a time over the Internet. | Communicate with a single machine at a time. |
| 10. | *Business Type* | B2C (Business to Customers) and B2B(Business to Business). | Only B2B (Business to Business) is used. |
| 11. | *Open API support* | IoT supports open API Integrations. | M2M does not support open API. |
| 12. | *Data Sharing* | Data is shared with applications that tend to improve the end-user experience. | Data is shared with the communication parties themselves. |

## 3.5.2 SCADA (Supervisory Control and Data Acquisition)

- Supervisory Control and Data Acquisition (SCADA) is a system of software and hardware elements that allows industrial organizations to:
  - Control industrial processes locally or at remote locations.
  - Monitor, gather, and process real-time data.
  - Directly interact with devices such as sensors, valves, pumps, motors, and more through human-machine interface (HMI) software.
  - Record events into a log file.
- The basic SCADA architecture begins with Programmable Logic Controllers (PLCs) or Remote Terminal Units (RTUs). PLCs and RTUs are microcomputers that communicate with an array of objects such as factory machines, sensors, HMIs, and end devices, and then route the information from those objects to computers with SCADA software. The SCADA software processes, distributes, and displays the data, helping operators and other employees analyses the data and make important decisions.
- For example, the SCADA system quickly informs an operator that a batch of product is showing a high occurrence of errors. The operator pause the operation and views the SCADA system data via an HMI to determine the cause of the issue. The operator reviews the data and discovers that Machine 4 was malfunctioning. The SCADA system's ability to inform the operator of an issue helps him to resolve it and prevent further loss of product.

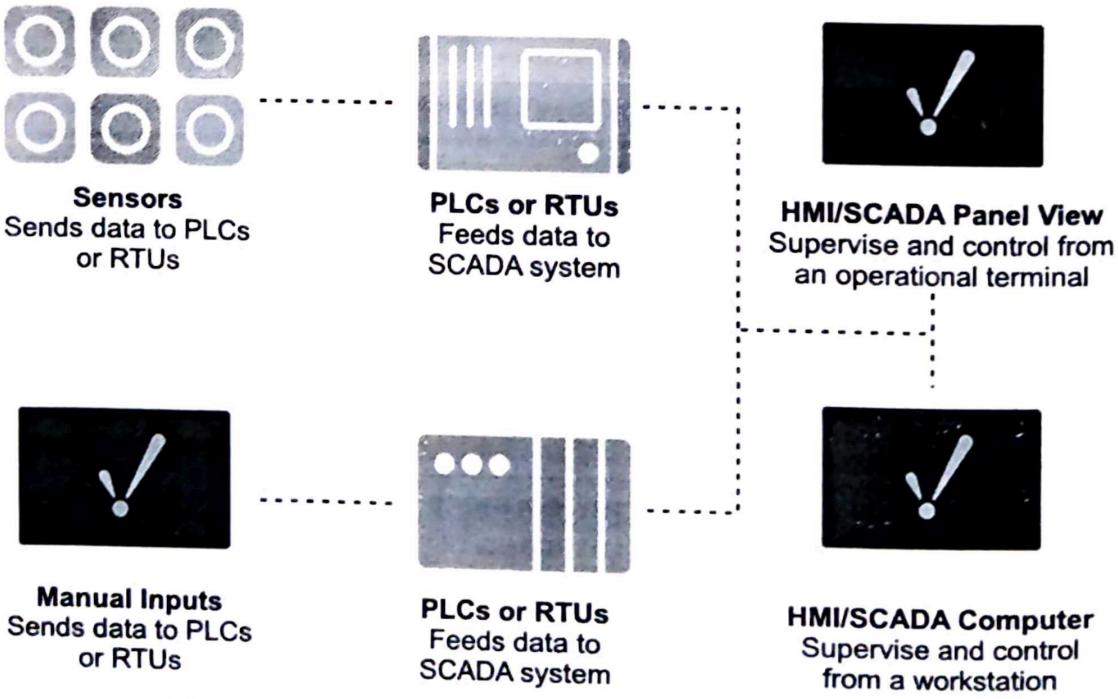


**Fig. 3.6: Supervisory Control and Data Acquisition**

## 3.5.3 WSN

- Wireless Sensor Network in IoT is an infrastructure-less wireless network that is used for deploying a large number of wireless sensors that monitor the system, physical and environmental conditions.

- A communication protocol is used to connect Sensors embedded in IoT devices. A Low-Power Wide-Area Network, LPWAN, is a type of wireless network designed to allow long-range communications between these IoT devices. Lora based Wireless Sensor Network is widely used. Sub-1 GHz, ZigBee, Thread etc. are also used to connect sensor networks and gateway and data collected from this sensor network can be sent to cloud using cellular networks such as NBIoT, LTE-M or Wi-Fi etc.

**Components of WSN in IoT:**

1. **Sensor Nodes:** Sensors play the vital role of capturing environmental variables.
2. **Radio Nodes:** Radio nodes or Master nodes in a Wireless sensor network receive data from the sensors and forward it to the gateway.
3. **Access Point or Gateway:** It is used to receive the data sent by the radio nodes wirelessly typically through the internet and send it over the cloud.
4. **Edge Computing and Data Analysis:** The data received by the gateway is analyzed. This data is further analyzed on the cloud and displayed on IoT mobile application or IoT dashboard.

**IoT and Wireless Sensor Networks:**

- WSN protocols in IoT are used to provide a connectivity medium between IoT sensor nodes and a central gateway. IoT consists of different tech stacks; WSN is just one and is a subset of IoT. It is a part where data is transmitted among several IoT devices mostly without internet.

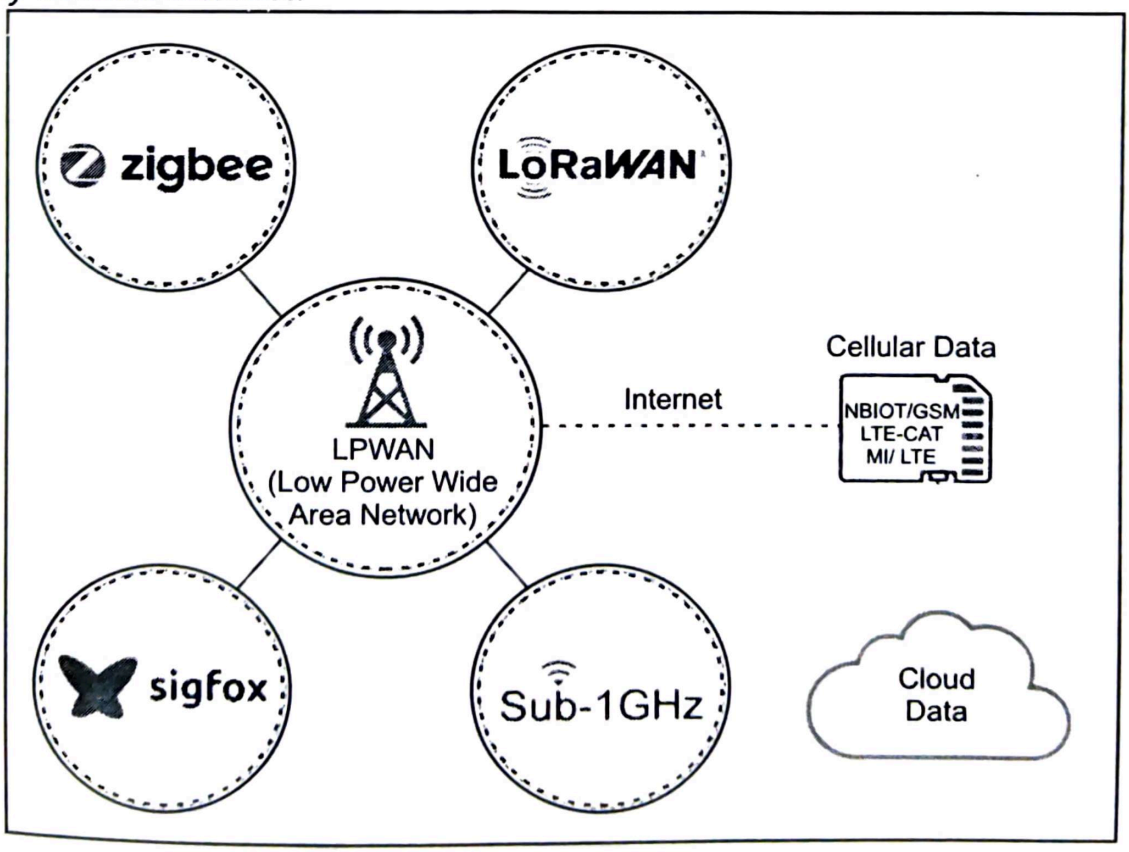


**Fig. 3.7: Wireless Sensor Network**

**Wireless Sensor Network Applications:**

1. Patient monitoring in hospitals, Home security, Military applications, Livestock monitoring, Server Room monitoring.
2. Wireless sensor network for smart agriculture.
3. Wireless sensor network for forest fire detection.
4. Wireless sensor network for water quality monitoring.
5. Wireless sensor network for office monitoring.
6. Wireless sensor network for environmental monitoring.
7. Wireless sensor network for landslide detection.
8. Wireless sensor network for IoT security.

## 3.5.4 RFID

- RFID is an acronym for "radio-frequency identification" and refers to a technology whereby digital data encoded in RFID tags or smart labels (defined below) are captured by a reader via radio waves. RFID is similar to barcoding in that data from a tag or label are captured by a device that stores the data in a database. RFID has several advantages over systems that use barcode asset tracking software. The most notable is that RFID tag data can be read outside the line-of-sight, whereas barcodes must be aligned with an optical scanner.

**Working:**

- RFID belongs to a group of technologies referred to as Automatic Identification and Data Capture (AIDC).
- AIDC methods automatically identify objects, collect data about them, and enter those data directly into computer systems with little or no human intervention.
- RFID methods utilize radio waves to accomplish this. At a simple level, RFID systems consist of three components: an RFID tag or smart label, an RFID reader, and an antenna.
  - RFID tags contain an integrated circuit and antennas which are used to transmit data to the RFID reader. RFID reader is also called an interrogator.
  - Then the RFID reader converts the radio waves to a more usable form of data. After that the information collected from the tags is transferred through a communications interface to a host computer system. In this system, the data can be stored in the database and analyzed at a later time.

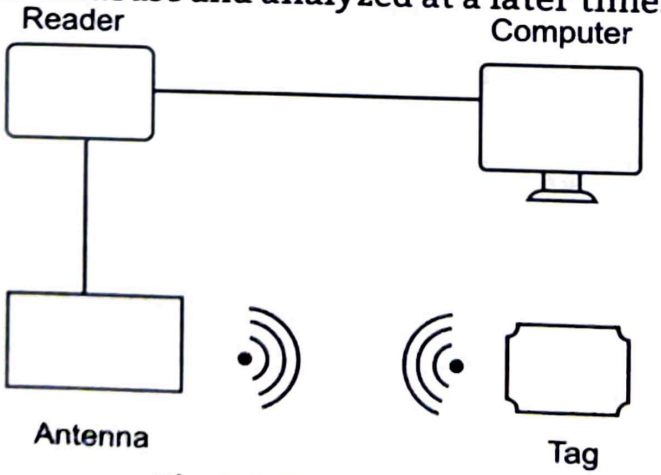


**Fig. 3.8: RFID System**

## Summary

- Internet of Things (IoT) is a network of physical objects or people called "things" that are embedded with software, electronics, network, and sensors that allows these objects to collect and exchange data.
- The first step in IoT system design methodology is to define the purpose and requirements of the system. In this step, the system purpose, behaviour and requirements (such as data collection requirements, data analysis requirements, system management requirements, data privacy and security requirements, user interface requirements) are captured.
- A sensor network comprises a group of small, powered devices, and a wireless or wired networked infrastructure.
- Request and Response Model follows Client-Server architecture. The client, when required, requests the information from the server.
- Four pillars of IoT are M2M, SCADA, WSN, and RFID.
- M2M is the connection of two or more than two devices with the internet for data sharing and analytics.
- Wireless Sensor Network(WSN) in IoT is an infrastructure-less wireless network that is used for deploying a large number of wireless sensors that monitor the system, physical and environmental conditions.
- RFID is an acronym for "radio-frequency identification" and refers to a technology whereby digital data encoded in RFID tags or smart labels are captured by a reader via radio waves.

## Check Your Understanding

1. IoT stands for _____.
   (a) Internet of Things (b) Internet of Technology
   (c) Incorporate Of Things (d) Incorporate of Technology
2. Identify among the following which is not a fundamental component of IoT System.
   (a) User Interface (b) Sensors
   (c) Transformers (d) Connectivity & Data Processing
3. WSN stands for _____.
   (a) Wireless sensor network (b) Wired sensor network
   (c) Wired sensor node (d) Wireless sensor node
4. WSN measures ____ physical parameter.
   (a) Sound (b) Temperature
   (c) Pollution (d) All the above

5. The development of WSN was motivated from _____.
   (a) Military
   (b) Hospitals
   (c) Schools
   (d) Offices
6. The components of a sensor node is _____.
   (a) Radio Trans receiver
   (b) Microcontroller
   (c) An electronic circuit
   (d) All the above
7. How many components do the RFID system consists of?
   (a) One
   (b) Two
   (c) Three
   (d) Four
8. What is the range of RFID tags which are using the ultra-high frequency?
   (a) Up to 10 cm
   (b) Up to 60 cm
   (c) 10 to 15 meter
   (d) Up to 100 cm
9. What are the strengths of RFID?
   (a) Advanced technology
   (b) Small in size and easy to use
   (c) Has high memory capacity
   (d) All of the above
10. The SCADA systems used to _____.
   (a) Monitor
   (b) Control
   (c) Both (a) and (b)
   (d) None of the above

**Answers**

| | | | | |
|---|---|---|---|---|
| 1. (a) | 2. (c) | 3. (a) | 4. (d) | 5. (a) |
| 6. (b) | 7. (b) | 8. (c) | 9. (d) | 10. (d) |

## Practice Questions

**Q.I Answer the following questions in short.**

1. How many steps are involved in IoT Platforms Design Methodology?
2. What is the purpose of IoT Platform?
3. Write any two applications of IoT networking.
4. Enlist different devices which are help us to communicate with IoT Platform.
5. Enlist IoT Networking components.
6. Draw the block diagram of IoT Communication Model.
7. Which technologies are used in WSNs?
8. What is M2M?
9. What is the full form of SCADA?

10. Mention any two applications of WSNs.
11. What are the components of WSN in IoT?
12. What is access point or gateway?
13. What is the full form of RFID?
14. How many components are included in RFID?
15. Mention the four pillars of IoT.

**Q.II Answer the following questions.**

1. Write a short note on SCADA.
2. Explain RFID system in detail.
3. Write a short note on WSNs.
4. Explain the designing steps of IoT with example.
5. Explain internet structure using Request-Response model.
6. Difference between M2M and IoT.
7. Describe SCADA.
8. Write down applications of WSNs.
10. Enlist the components of WSN in IoT. Explain in detail.

**Q.III Define the terms.**

1. SCADA.
2. WSNs.
3. RFID.

***

# 4...

# Introduction to IoT Protocols

## Learning Objectives...

- To analyze protocols for communication among IoT devices.
- To understand the key components that make up an IoT system.

## 4.1 INTRODUCTION

- The IoT system can function and transfer information only when devices are online and safely connected to a communications network. This is where IoT standards and protocols make an entry. IoT devices can be connected either using an IP (Internet Protocol) or a non-IP network. IP network connections are relatively complex and require increased memory and power from IoT devices, although range is not a problem. On the other hand, non-IP networks demand relatively less power and memory but have a range limitation.

## 4.2 PROTOCOL STANDARDIZATION FOR IoT

- IoT communication protocols are modes of communication that protect and ensure optimum security to the data being exchanged between connected devices.
- Networking standards are the most important in the IoT. Standard protocols define rules and formats for setting up and managing IoT networks, along with how data are transmitted across these networks. Networking protocols can be categorized into multiple layers accordingly to the communication stack (i.e. OSI or TCP/IP model).
- The IoT devices are typically connected to the Internet via an IP (Internet Protocol) network. However, devices such as Bluetooth and RFID allow IoT devices to connect locally. In these cases, there is a difference in power, range, and memory used. Connection through IP networks are comparatively complex, requires increased memory and power from the IoT devices while the range is not a problem. On the

other hand, non-IP networks demand comparatively less power and memory but have a range limitation.

- As far as the IoT communication protocols or technologies are concerned, a mix of both IP and non-IP networks can be considered depending on usage.
- In the next section, we are going to discuss all internet protocols in details.

## 4.3 M2M AND WSN PROTOCOLS

### M2M: Machine-to-Machine Communications

- One of the new technologies that are part of the Internet of Things is Machine-to-Machine (M2M) communications. M2M, though not well-defined, is a set of methods and protocols to allow devices to communicate and interact over the Internet (or other network) without human intervention.
- M2M is sometimes considered to be low-overhead short-range wireless communication between machines, utilizing protocols with much less overhead than full-blown TCP/IP. Many M2M applications involve low power wireless devices with limited computing power and narrowly-defined functionality.
- Low-overhead protocols have been devised for them, including Message Queue Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), and Open Mobile Alliance Light Weight M2M (OMA LWM2M).
- CoAP is actually a specialized web transfer protocol designed for applications such as smart energy and building automation. There is no reason why IoT devices cannot use high-powered CPUs and wide bandwidth, and in many applications this is clearly necessary, such as smart cars interacting with external servers. So IoT spans a huge range from very simple low-powered specialized devices and sensors with low bandwidth needs to complex, high-powered devices in large high-bandwidth environments.

### WSNs: Wireless Sensor Networks

- IoT configurations often involve sensors, which can be connected by wireless networks. Such sensor networks are termed "Wireless Sensor Networks" or WSNs. A WSN comprises spatially distributed autonomous devices equipped with sensors, connected through a wireless network to some type of gateway. The sensors typically monitor physical or environmental conditions. The gateway communicates with another set of devices that can act on the information from the sensors.
- A WSN can be generally described as a network of nodes that cooperatively sense and may control the environment enabling interaction between persons or computers and the surrounding environment.
- WSNs typically consist of small, inexpensive, resource constrained devices that communicate among each other using a multi-hop wireless communication. Each node of WSN is called as a SN which contains one sensor, embedded processors,

limited memory, low-power radio, and is normally operated with battery. Each SN is responsible for sensing a desired event locally and for relaying a remote event sensed by other SNs so that the event is reported to the destination through BS. As SNs have limited energy so applications and protocols for WSNs should be carefully designed for optimized consumption of energy for prolonging the network lifetime.

- Low Energy Adaptive Clustering Hierarchy (LEACH) is the one of the most common protocols used in this WSN. LEACH is a self-organizing, adaptive clustering protocol which uses steady energy load distribution among the SNs in the WSN.
- The operation of LEACH is divided into rounds and each round is divided into two phases namely as: Set-up and Steady-state phase. Steady-state phase is always long compared to the Set-up phase to minimize the overhead. In LEACH protocol, the SNs organize themselves into local clusters, with one node acting as the leader and known as cluster head (CH) and rest of the nodes act as ordinary nodes. To extend the lifetime of the network, LEACH includes randomized rotation of the high-energy CH and performs local data fusion to transmit the amount of data being sent from the CHs to the BS. If BS is far away from the network then the energy of CHs will be affected as only CHs are directly communicating with the BS. Set of clusters will be different for different time interval and the decision to become a CH depends on the amount of energy left at the SN.

**Application of WSNs:**

1. It is used in patient monitoring.
2. It is used in environmental monitoring of air, water, and soil.
3. It is used in structural monitoring for buildings and bridge.
4. It is also used in industrial machine monitoring and process monitoring. The wireless network could be Wi-Fi or Bluetooth.

## 4.4 RFID PROTOCOL

- A communications protocol is a way of organizing the conversation between devices -- in the case of RFID, between tags and a reader to ensure that information actually gets transferred.
- A protocol defines:
  - **An Air Interface:** What sort of modulation of the reader signal is used to define a binary one? What is a zero? What kind of signal does the tag send? How fast does everything go? Is information sent in discrete packets, and if so how are they formed?
  - **Medium Access Control**: Who gets to talk when? How are collisions between contending users (tags in this context) resolved?
  - **Data Definitions:** What sort of data is associated with a tag? What does it mean?


**Fig 4.1: RFID Technology**

- The current standard Protocol used in RFID is the EPCglobal Class 1 Gen 2 protocol (EPC C1G2). This is an arbitration oriented protocol used in every commercial reader, also included in ISO 18000-6C. The main purpose of this protocol is the reading the sensor data from a RFID tag.
- In order to improve the performance of the standard EPC C1G2 when streaming sensor data from passive sensors, the sensor Frame Slotted Aloha protocol (sFSA) is recommended. This work analyses the times consumed by the identification and the read phase and, taking into account this analysis, proposes the sFSA protocol. Furthermore, it presents an experimental performance evaluation of an RFID sensor network using CRFID tags using the proposed sFSA protocol in comparison with the standard EPC C1G2. These measurements are carried out using a physical platform which consists of a Software Defined Radio (SDR) reader and several Wireless Identification and Sensing Platform (WISP) tags.
- The WISP is a battery-free RFID sensor device powered via the RF energy transmitted by the reader. It carries a programmable micro-controller that allows the use of different identification and sensing protocols. Furthermore, these tags contain an accelerometer, whose measurement data can be written into the user memory and read via the RFID communication. The analysis of the implementations performed within this work show that the proposed protocol increases the Sensor Read Rate (SRR), defined as the number of sensor data reads per second, compared to EPC C1G2 by more than five times on average.

## 4.4.1 Modbus Protocol

- Modbus Protocol is a request-response protocol implemented using a master-slave relationship. In a master-slave relationship, communication always occurs in pairs one device must initiate a request and then wait for a response and the initiating device (the master) is responsible for initiating every interaction. Typically, the master is a Human Machine Interface (HMI) or Supervisory Control and Data Acquisition (SCADA) system and the slave is a sensor, Programmable Logic Controller (PLC), or Programmable Automation Controller (PAC). Since Modbus protocol is just a messaging structure, it is independent of the underlying physical layer. It is traditionally implemented using RS232, RS422, or RS485.

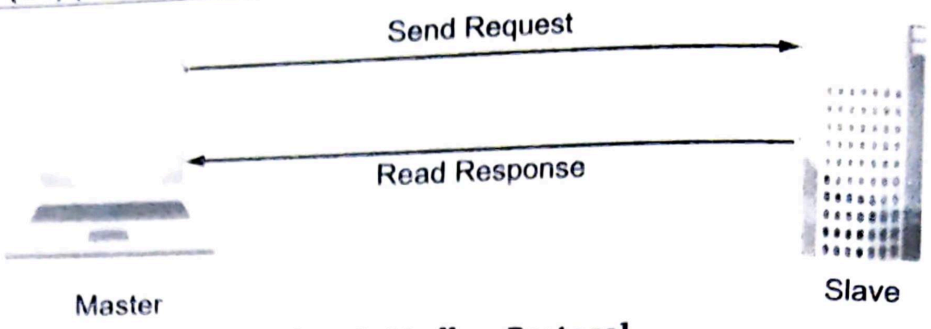


**Fig 4.2: Modbus Protocol**

- The function code in the request tells the addressed slave device what kind of action to perform. The data bytes contains any additional information that the slave will need to perform the function. For example, function code 03 will request the slave to read holding registers and respond with their contents. The data field must contain the information telling the slave which register to start at and how many registers to read. The error check field provides a method for the slave to validate the integrity of the message contents.

**The Response:**

- If the slave makes a normal response, the function code in the response is an echo of the function code in the request. The data bytes contain the data collected by the slave, such as register values or status. If an error occurs, the function code is modified to indicate that the response is an error response, and the data bytes contain a code that describes the error. The Controllers can be setup to communicate on standard Modbus networks using either of two transmission modes: ASCII or RTU.

1. **ASCII Mode:**
   - When controllers are setup to communicate on a Modbus network using ASCII (American Standard Code for Information Interchange) mode, each eight-bit byte in a message is sent as two ASCII characters. The main advantage of this mode is that it allows time intervals of up to one second to occur between characters without causing an error.
2. **RTU Mode:**
   - When Controllers are setup to communicate on a Modbus network using RTU (Remote Terminal Unit) mode, each eight-bit byte in a message contains two four-bit hexadecimal characters. The main advantage of this mode is that its greater character density allows better data throughput than ASCII for the same baud rate. Each message must be transmitted in a continuous stream.

## 4.4.2 ZigBee Architecture

**What Is ZigBee?**

- ZigBee communication is specially built for control and sensor networks on IEEE 802.15.4 standard for Wireless Personal Area Networks (WPANs), and it is the product from ZigBee alliance. This communication standard defines physical and Media Access Control (MAC) layers to handle many devices at low-data rates. These ZigBee's WPANs operate at 868 MHz, 902-928MHz, and 2.4 GHz frequencies. The data rate of 250 kbps is best suited for periodic as well as intermediate two-way transmission of data between sensors and controllers.

- ZigBee is a low-cost and low-powered mesh network widely deployed for controlling and monitoring applications where it covers 10-100 meters within the range. This communication system is less expensive and simpler than the other proprietary short-range wireless sensor networks as Bluetooth and Wi-Fi.

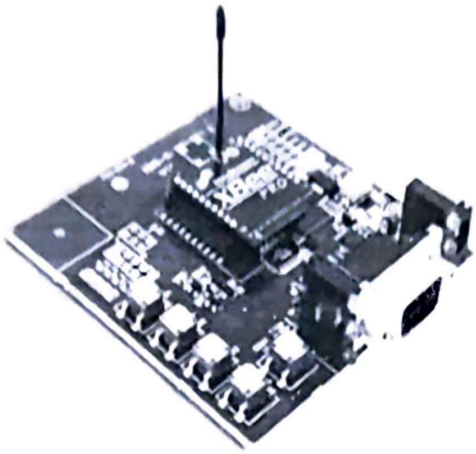

**Fig. 4.3 (a): ZigBee Modem**

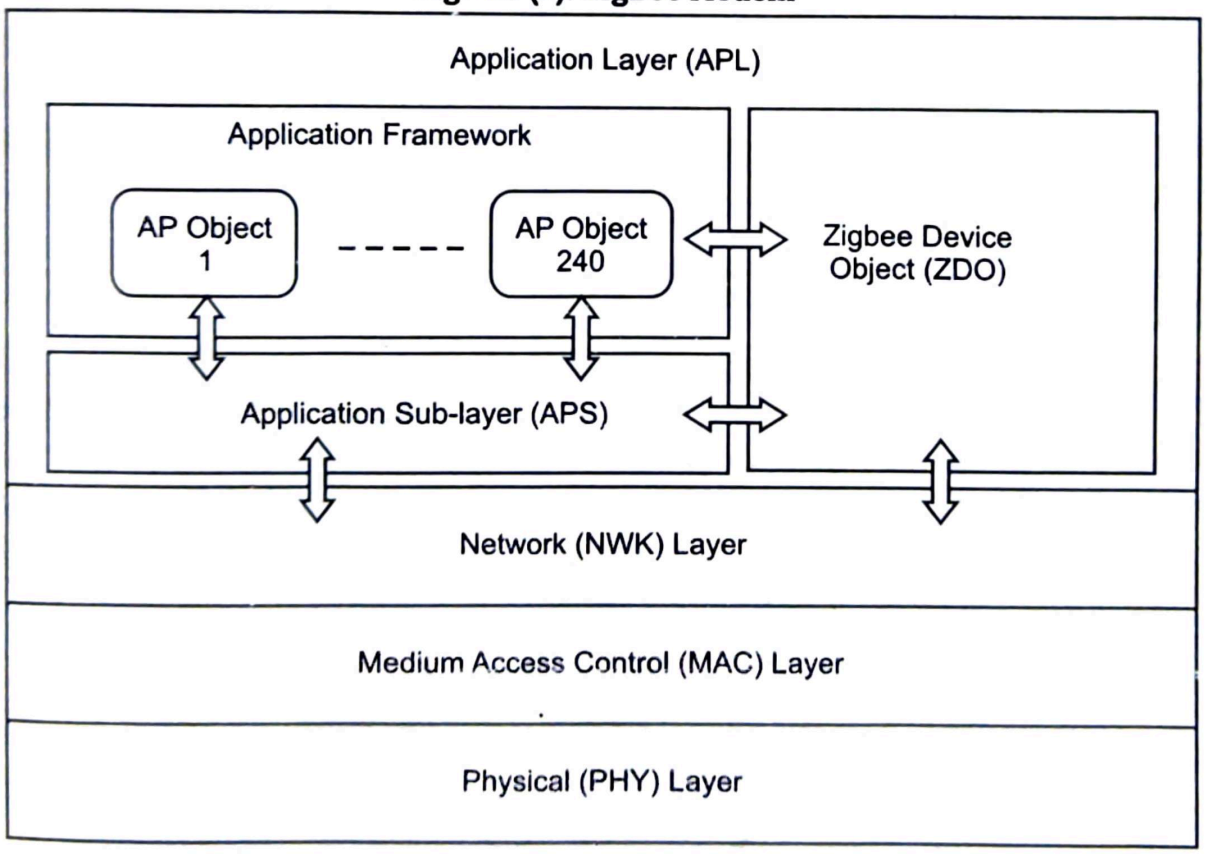


**Fig. 4.3 (b): ZigBee Architecture**

- ZigBee system structure consists of three different types of devices as ZigBee Coordinator, Router, and End device. Every ZigBee network must consist of at least one co-ordinator which acts as a root and bridge of the network. The co-ordinator is responsible for handling and storing the information while performing receiving and transmitting data operations.

- **Physical Layer:** This layer does modulation and demodulation operations upon transmitting and receiving signals respectively. Mapping bits of information and permits them to travel through the air by modulation and spreading techniques which is the basic task of physical layer.
- **MAC Layer:** This layer is responsible for reliable transmission of data by accessing different networks with the Carrier Sense Multiple Access Collision Avoidances (CSMA). This also transmits the beacon frames for synchronizing communication.
- **Network Layer:** This layer takes care of all network-related operations such as network setup, end device connection, and disconnection to network, routing, device configurations, etc.
- **Application Support Sub-Layer:** This layer enables the services necessary for ZigBee device objects and application objects to interface with the network layers for data managing services. This layer is responsible for matching two devices according to their services and needs.
- **Application Framework:** It provides two types of data services as key-value pair and generic message services. The generic message is a developer-defined structure, whereas the key-value pair is used for getting attributes within the application objects. ZDO provides an interface between application objects and the APS layer in ZigBee devices. It is responsible for detecting, initiating, and binding other devices to the network.

## 4.5 IP BASED PROTOCOLS

### 4.5.1 MQTT (Secure)

- In 1999, IBM and Eurotech developers established the first version of Message Queuing Telemetry Transport, or MQTT. It is an open TCP/IP-based protocol that provides data exchange within a network of devices.
- Message Queuing Telemetry Transport (MQTT) protocol is used to connect and control smart home devices. It works as a messaging protocol and enables communication between devices in an IoT ecosystem.

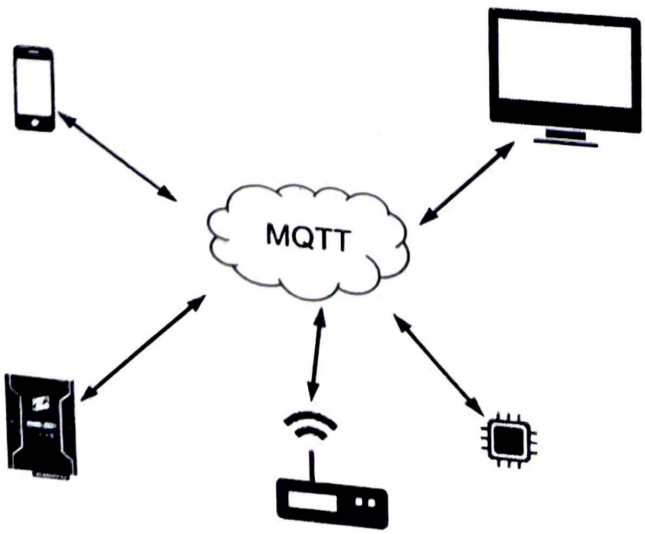


Fig. 4.4: MQTT Protocol

- The protocol uses a publish-subscribe pattern and includes the following components:
  - **A Server, or Broker**, which communicates with the clients (publishers and subscribers) via an internet connection or a local network.
  - **A Publisher** creates messages and publishes them to a certain topic.
  - **A Subscriber** receives the messages relevant to the topic it is subscribed to.
- Publishers and Subscribers may switch roles, and sometimes MQTT clients take on the role of both. The number of publishers and subscribers can be unlimited. This will only depend on your server's capacity. The clients have IDs by which the server identifies them.
- Each topic comprises different subtopics, or levels, that form a hierarchy. By publishing messages to the exact subtopics, the publisher will be able to reach the intended recipient that has a subscription for the same topic levels.
- Like any other binary protocol, MQTT has an edge over text protocols in machine-to-machine communication. Devices can send and receive the binary data without extra processing, which speeds up the messaging within the network.
- MQTT is a high-usage technology that was initially used to build connections within a satellite-based network. The lightweight protocol allowed for low bandwidth and power consumption.
- The resource efficiency of MQTT played a major role in the system, working via the costly satellite link. Later on, MQTT has come into use with more accessible and low-cost communication channels and in various application areas, including the Internet of Things (IoT).

**Application of MQTT:**

1. MQTT is a flexible and easy-to-use technology that provides effective communication within an IoT system.
2. Amazon Web Services used MQTT as a basis for AWS IoT services. AWS IoT Core has an MQTT-based message broker, and it supports two levels of MQTT.

## 4.5.2 6LoWPAN

- 6LoWPAN stands for IPv6 over Low-power Wireless Personal Area Networks. It is a standard protocol for realizing IPv6 communication on wireless networks composed of low-power wireless modules.
- 6LoWPAN specification contains packet compression and other optimization mechanisms to enable the efficient transmission of IPv6 packets on a network with limited power resources and reliability, which makes efficient IPv6 communication over low-power wireless networks possible.

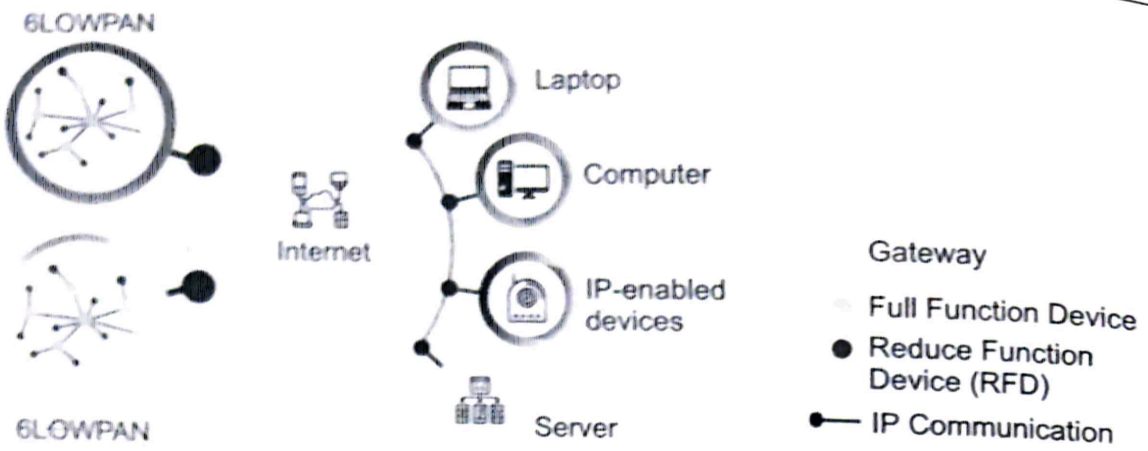


**Fig. 4.5 : 6LoWPAN Architecture**

## 4.5.3 LoRa

- LoRa encodes information on radio waves using chirp pulses similar to the way dolphins and bats communicate. LoRa modulated transmission is robust against disturbances and can be received across great distances.
- LoRa is ideal for applications that transmit small chunks of data with low bit rates. Data can be transmitted at a longer range compared to technologies like Wi-Fi, Bluetooth or ZigBee. These features make LoRa well-suited for sensors and actuators that operate in low power mode.
- LoRa can be operated on the license free sub-gigahertz bands, for example, 915 MHz, 868 MHz, and 433 MHz. It also can be operated on 2.4 GHz to achieve higher data rates compared to sub-gigahertz bands, at the cost of range. These frequencies fall into ISM bands that are reserved internationally for industrial, scientific, and medical purposes.

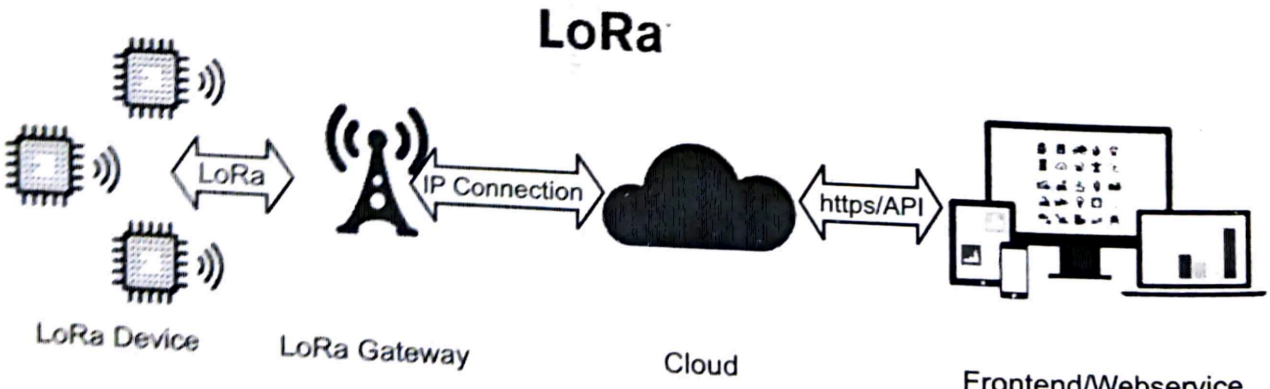


**Fig. 4.6: LoRa Protocol**

## Summary

- IoT communication protocols are modes of communication that protect and ensure optimum security to the data being exchanged between connected devices.

- Machine-to-Machine (M2M) is one of the new technologies that are part of the Internet of Things communications. M2M is a set of methods and protocols to allow devices to communicate and interact over the Internet (or other network) without human intervention.
- IoT configurations often involve sensors, which can be connected by wireless networks. Such sensor networks are termed "Wireless Sensor Networks" or WSNs.
- Modbus Protocol is a request-response protocol implemented using a master-slave relationship. In a master-slave relationship, communication always occurs in pairs one device must initiate a request and then wait for a response and the initiating device (the master) is responsible for initiating every interaction.
- RFID protocol is the way of organizing the conversation between tags and a reader to ensure that information actually gets transferred.
- ZigBee communication is specially built for control and sensor networks on IEEE 802.15.4 standard for Wireless Personal Area Networks (WPANs), and it is the product from ZigBee alliance. This communication standard defines physical and Media Access Control (MAC) layers to handle many devices at low-data rates.
- Message Queuing Telemetry Transport (MQTT) protocol is used to connect and control smart home devices. It works as a messaging protocol and enables communication between devices in an IoT ecosystem.
- 6LoWPAN stands for IPv6 over Low-power Wireless Personal Area Networks. It is a standard protocol for realizing IPv6 communication on wireless networks composed of low-power wireless modules.
- LoRa encodes information on radio waves using chirp pulses. LoRa is ideal for applications that transmit small chunks of data with low bit rates. Data can be transmitted at a longer range compared to technologies like Wi-Fi, Bluetooth or ZigBee.

## Check Your Understanding

1. Among the following layers, identify the one which is used for wireless connection in IoT devices?
   (a) Datalink layer (b) Transport layer
   (c) Network layer (d) Application layer
2. Which of the following IoT gateway must provide?
   (a) Protocol abstraction (b) Security with hardware
   (c) Simple and fast installation (d) Data storage
3. Which of the following protocol used to link all devices in IoT?
   (a) UDP (b) TCP/IP
   (c) HTTP (d) Network

4. What is the full form of the LPWAN?
   (a) Low Protocol Wide Area Network
   (b) Low Power Wide Area Network
   (c) Long Protocol Wide Area Network
   (d) Long Power Wide Area Network
5. What is the full form of the MQTT?
   (a) Multi-Queue Telemetry Things
   (b) Multiple Queue Telemetry Things
   (c) Message Queue Telemetry Things
   (d) Message Queue Telemetry Transport
6. How many components do the RFID system consists of?
   (a) One
   (b) Two
   (c) Three
   (d) Four
7. WSNs are spatially distributed _____ sensors.
   (a) Autonomous sensors
   (b) Dependent sensors
   (c) Bi-directional sensors
   (d) None of the above
8. Which of the following are the applications of WSN?
   (a) Health Monitoring
   (b) Industrial Process Monitoring and Control
   (c) Military
   (d) All of the above
9. 'ZigBee is a technological standard created for controlling and sensing the network'. True/False?
   (a) True
   (b) False
   (c) Can be True or False
   (d) Cannot say
10. Which of the following is true according to ZigBee Standard?
   (a) Low Power Consumption
   (b) Low Data Rate
   (c) Short-Range
   (d) All of the above

**Answers**

| | | | | |
|---|---|---|---|---|
| 1. (a) | 2. (a) | 3. (b) | 4. (b) | 5. (d) |
| 6. (b) | 7. (a) | 8. (d) | 9. (a) | 10. (d) |

## Practice Questions

**Q. I Answer the following question in short.**

1. What is the full form of MQTT in IoT?
2. Which protocol is used to connect IoT devices?
3. What is M2M?

4. Which protocol is used in WSNs?
5. Which protocol is used in RFID?
6. What is Modbus protocol?
7. What is ZigBee?
8. How many layers are there in ZigBee architecture?
9. Which are applications of MQTT?
10. What is the full form of 6LoWPAN?

**Q.II Answer the following questions.**

1. What is RFID protocol?
2. Explain WSN protocol.
3. Explain M2M and WSN protocol with example.
4. Explain ZigBee architecture with Modbus protocol.
5. Draw the architecture of ZigBee and explain in detail.
6. Mention IP based protocol. Explain any one of them in detail.
7. Explain MQTT protocol in detail.
8. Explain 6LoWPAN in detail.
9. Explain LoRa in detail.

**Q.III Define the terms.**

1. Zigbee
2. LoRa.

***

# 5...

# Cloud Platforms for IoT

## Learning Objectives...

- To get information about clod computing environment.
- To aware about messaging platforms.
- To get knowledge regarding AWS and other technologies of IoT Cloud platform.
- To know the fundamental of SOAP Architecture.

## 5.1 INTRODUCTION TO CLOUD STORAGE MODELS

- Cloud storage allows user to save data and files in an off-site location that user can access either through the public internet or a dedicated private network connection. Data that user transfer off-site for storage becomes the responsibility of a third-party cloud provider.
- Cloud storage delivers a cost-effective, scalable alternative to storing files on on-premise hard drives or storage networks. Computer hard drives can only store a finite amount of data. When users run out of storage, they need to transfer files to an external storage device.
- Cloud storage services provide elasticity, which means you can scale capacity as your data volumes increase or dial down capacity if necessary. By storing data in a cloud, your organization save by paying for storage technology and capacity as a service, rather than investing in the capital costs of building and maintaining in-house storage networks.

**Communication API :**

- Cloud Models are relied on Communication API.
- Communication API facilitates data transfer, control information transfer from application to cloud, one service to another.
- It also exists in the form of Communication Protocols.
- It supports RPC, PUBSUB and WAMP.
- Django web framework is used to implement Communication API.

- A cloud storage API is a specialized type of API set that facilitates the access, addition, editing and removal of data stored on a remote cloud storage server. It allows Web applications and services to access cloud storage from a cloud storage service provider in a programmatic manner.
- Web applications request services and operations from a remote cloud storage server through an integrated cloud storage API. Typically, these APIs are designed over REST and SOAP architectures that facilitate the retrieval of data stored over distributed database systems. For example, the Google Cloud Storage API allows developers to program and manage data through its REST-oriented interface on Google Cloud Storage. Cloud storage APIs also allows remote data management services, session initiation/termination and other storage management functionality.
- A cloud storage API is an application programming interface that connects a locally based application to a cloud-based storage system so that a user can send data to it and access and work with data stored in it. To the application, the cloud storage system is just another target device, like disk-based storage.
- A cloud API is specific to the storage service it targets. For example, a cloud object storage service might offer an API that can create, fetch and delete objects on that platform, as well as perform other object-related tasks. An API for a file storage service operates at the file and folder level, supporting operations such as uploading and downloading files and sharing folders with multiple users.

**Challenges of Cloud Storage API:**

- Cloud Storage APIs provide a range of capabilities that developers can implement in their applications, but each is specific to a cloud storage service.
- This adds complexity to the development process when multiple cloud services are involved. Even if those services are with the same provider, individual APIs are often still required. The process becomes even more complex when storage services and their APIs are updated or changed in any way.
- The lack of a common cloud API can make it difficult to move data from one provider's service to another's.

**Examples of Cloud Storage APIs:**

1. **Amazon Web Service (AWS)**: This one is a Hypertext Transfer Protocol (HTTP) interface to Amazon S3 that is programming language-neutral. The API lets developers build applications that include HTTP requests and use standard HTTP headers and status codes. In this way, developers can work with any toolkit that supports HTTP. However, Amazon added functionality to HTTP in some areas, so developers should be aware of these changes when working with the API.
2. **Google Drive API:** This API lets developers create applications that integrate with Google Drive. It supports file upload and download, search for files and folders, and file and folder sharing. Developers can also create third-party shortcuts that are external links to data in a different data store or cloud storage system.

## 5.2 CLOUD FOR IoT

**Why IoT Cloud?**

- An IoT cloud is a massive network that supports IoT devices and applications. This includes the underlying infrastructure, servers and storage needed for real-time operations and processing.
- IoT cloud offer an efficient, flexible and scalable model for delivering the infrastructure and services needed to power IoT devices and applications for businesses with limited resources. IoT clouds offer on-demand, cost-efficient hyperscale; so organizations can get benefitted the significant potential of IoT without having to build the underlying infrastructure and services from scratch.
- Cloud computing enables companies to store, manage and process data over cloud-enabled platforms providing flexibility, scalability and connectivity. Different models of cloud computing when implemented correctly help businesses with digital transformation, efficiency and growth. However, when connected with IoT, the cloud enables a thing that's never seen before making business thrive at a faster rate.
- There are several cloud services & solutions playing numerous roles in an IoT environment. Some of the cloud computing services have inbuilt capabilities of machine learning, business intelligence tools and SQL engines to perform complex tasks required of IoT. But, let us understand how these cloud services benefit the IoT ecosystem.

**Roles of Cloud Computing in IoT:**

1. **Enables Remote Computing Capabilities:**
   - With a large storage capacity, IoT eliminates the dependencies on on-site infrastructure. With continued development and internet-based tech development such as the internet and devices supporting advanced cloud solutions, cloud technology has become main stream. Packed with IoT, cloud solutions provide enterprises with the capability to access remote computing services with a single click or command.
2. **Security and Privacy:**
   - Tasks can be handled automatically with cloud tech and IoT, organizations are able to reduce security threats by a considerable amount. A cloud tech-enabled with IoT is a solution that provides preventive, detective and corrective control. With effective authentication and encryption protocols, it also provides users with strong security measures. Protocols such as biometrics in IoT products help manage as well as safeguard user identities along with data.
3. **Data Integration:**
   - Current technology developments have not only integrated IoT and cloud smoothly but also provide real-time connectivity and communication. This in turn makes the extraction of real-time information about key business processes

and performing on-spot data integration with 24/7 connectivity easy. Cloud-based solutions with powerful data integration capabilities are able to handle a large amount of data generated from multiple sources along with its centralized storage, processing and analysis.

4. **Minimal Hardware Dependency:**
   - Presently, several IoT solutions offer plug-and-play hosting services that are enabled by integrating the cloud with the IoT. With cloud-enabled, IoT hosting providers need not rely on any kind of hardware or equipment to support the agility required by IoT devices. It has become easy for organizations to implement large scale IoT strategies seamlessly across platforms and move to Omni-channel communication.
5. **Communication between Multiple Devices:**
   - IoT devices and services need to connect with each other and communicate to perform tasks that are enabled using cloud solutions. By supporting several robust APIs, Cloud & IoT is able to interact amongst themselves and connected devices. Having a cloud supported communication helps fasten the interaction happen seamlessly.

## 5.3 INTRODUCTION TO AMAZON WEB SERVICES FOR IoT And SkyNet IoT

**What is Amazon Web Services for IoT?**

- AWS IoT provides the cloud services that connect your IoT devices to other devices and AWS cloud services. AWS IoT provides device software that can helps user to integrate IoT devices into AWS IoT-based solutions. If user devices can connect to AWS IoT, it can connect them to the cloud services that AWS provides.

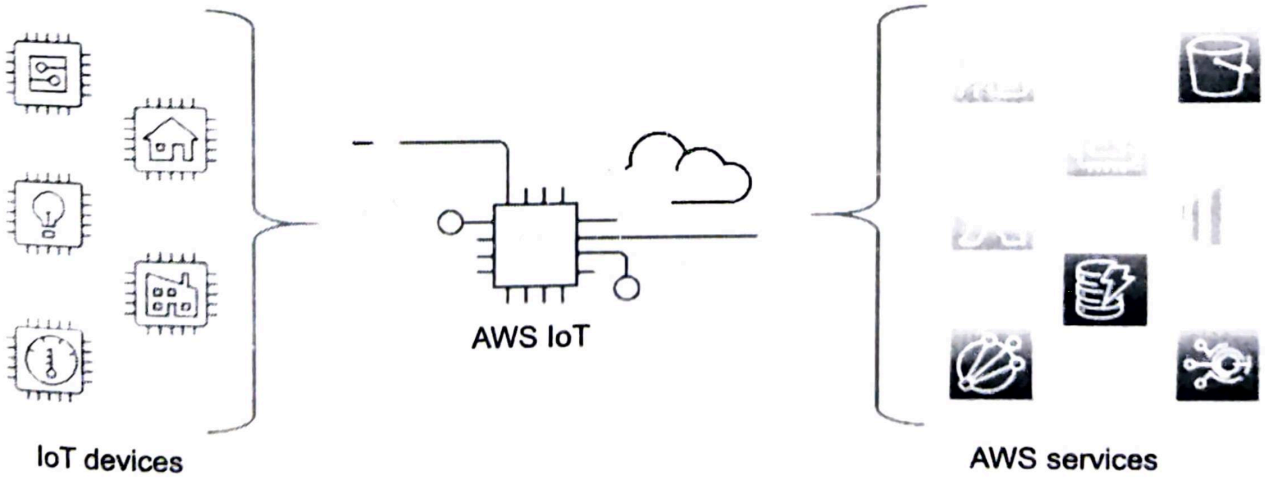


**Fig. 5.1: AWS IoT**

- AWS IoT lets user select the most appropriate and up-to-date technologies for user solution. To manage and support users IoT devices in the field, AWS IoT Core supports these protocols:

- MQTT (Message Queuing and Telemetry Transport)
- MQTT over WSS (Websockets Secure)
- HTTPS (Hypertext Transfer Protocol - Secure)
- LoRaWAN (Long Range Wide Area Network)

**How devices and apps access AWS IoT?**

- AWS IoT provides the following interfaces for AWS IoT:
  - **AWS IoT Device SDKs:** Build applications on user devices that send messages to and receive messages from AWS IoT.
  - **AWS IoT Core for LoRaWAN:** Connect and manage your long range WAN (LoRaWAN) devices and gateways by using AWS IoT Core for LoRaWAN.
  - **AWS Command Line Interface (AWS CLI):** Run commands for AWS IoT on Windows, Mac OS, and Linux. These commands allow user to create and manage thing objects, certificates, rules, jobs, and policies.
  - **AWS IoT API:** Build IoT applications using HTTP or HTTPS requests. These API actions allow you to programmatically create and manage thing objects, certificates, rules, and policies.
  - **AWS SDKs:** Build IoT applications using language-specific APIs.

**Categories of IoT products:**

- There are 3 key categories of IoT products on AWS as mentioned below:
  1. **Devices Software:** It includes services such as the FreeRTOS and AWS IoT Greengrass etc.
  2. **Connectivity and Control Services:** It includes services like AWS IoT Core, AWS IoT Device Defender and AWS IoT Device Management, etc.
  3. **Analytics Services:** It includes services such as AWS IoT Events, AWS IoT Analytics, AWS IoT SiteWise and AWS IoT ThingsGraph, etc.
- AWS IoT cloud services enable users to connect IoT devices with other IoT devices and AWS Cloud services. User may connect IoT devices with AWS IoT-based applications with the aid of the device software that AWS IoT offers. Devices can connect to AWS IoT. They can use AWS Cloud services.

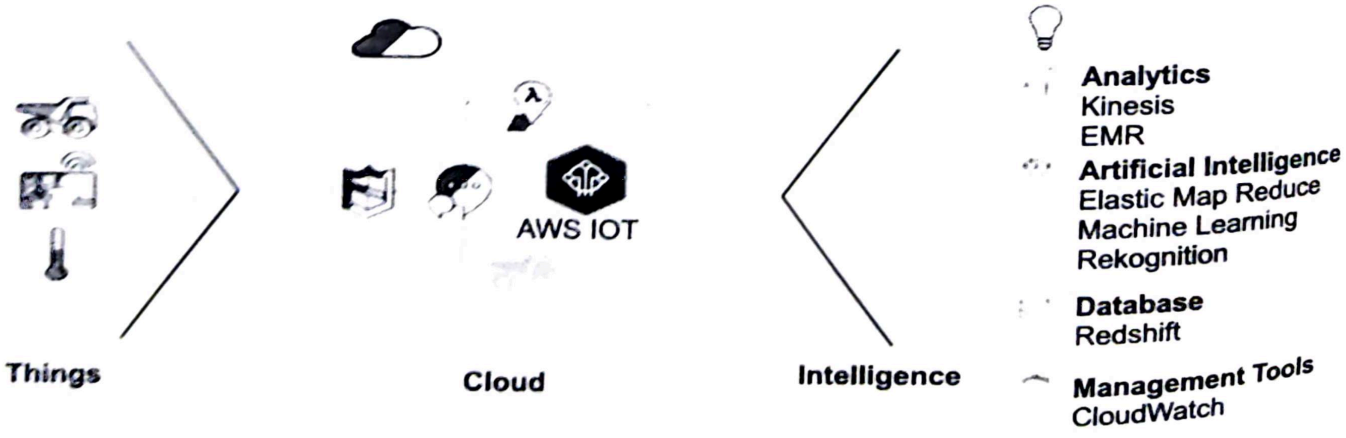


**Fig. 5.2: Connecting of IoT Devices**

**Introduction to SkyNet IoT:**

- SkyNet is an open source instant messaging platform for Internet of Things (IoT).
- The SkyNet API supports both HTTP REST and real time WebSockets.
- SkyNet allows users to register devices on the network.
- A device can be anything including sensors, smart home devices, cloud resources, drones etc.

**What is SkyNet?**

- SkyNet is an open protocol for hosting data and web applications on the decentralized web.
- SkyNet decentralizes "the cloud" so that user and application data is not stored by (and only accessible to) a single, central authority. Instead, data is held in a decentralized manner, allowing it to be:
  - Available across the globe on any device.
  - Accessible to any application and controlled by the user.

**SkyNet:**

- A Cloud-based MQTT (Message Queue Telemetry Transport) powered network.
- It transports message with any connected nodes such as smart home devices, sensors, cloud resources, drones, Arduinos, RaspberryPis, among others.
- Powered by Node.JS, known for fast, event-driven operations, ideal for nodes and devices such as RaspberryPi, Arduino, and Tessel.
- When nodes and devices register with SkyNet, they are assigned a unique id known as a UUID along with a security token.
- When node or device is connecting to Skynet, it can query and update devices on the network and send machine-to-machine (M2M) messages in an RPC-style.
- SkyNet aim is real time M2M communication.
- SkyNet is designed to run on a single network or mesh of IoT networks that share a common API or communications protocol.
- Devices can discover, query, and message other devices on the network.
- SkyNet recently released its IoT Hub which allows the user to connect smart devices with and without IP addresses directly to SkyNet including: Nest, Phillips Hue lightbulbs, Belkin Wemos, Insteons, and other not-so-smart devices such as Serial port devices and RF (Radio Frequency) devices.

**Why does Skynet matter?**

- Skynet matters because information matters. Skynet allows for a decentralized web that is censorship-resistant, has highly redundant storage and applications, and is available around the globe.

**For developers:**

- You do not pay for your application's storage.
- You can launch an app with access to a user's data on day one.
- You will not have to worry about corporations pulling access to their resources.
- If you are invested in centralized provider, you can maintain a failover site for when they go down.

**For users:**

- You take your data with you, not worrying about corporate oversight.
- You support developers and content creators by simply accessing their work.
- You experience a web free of targeted ads.
- You never have to put your privacy or security at risk.

- With SkyNet platform, we are able to save a ton of work by not having to develop our own communication tools, authentication scheme, or backend service to handle the streams of data (our devices simply turn on, authenticate and connect with SkyNet automatically, and we are able to start passing data securely).

**Python Client for SkyNet:**

```
from socketIO_client import SocketIO

HOST = '<Enter Host IP>'
PORT = 3000
UUID = '<enter UUID>'
TOKEN = '<enter Token>'

def on_status_response(*args):
    print 'Status: ' args

def on_ready_response(*args):
    print 'Ready: ' args

def not_ready_response(*args):
    print 'Not Ready: ' args

def on_sub_response(*args):
    print 'Subscribed: ' args

def on_msg_response(*args):
    print 'Message Received: ' args

def on_whoami_response(*args):
    print 'Who Am I: ' args
```

①

```
def on_id_response(*args):
    print 'Web Socket connecting to SkyNet with socket ID: ' +args[0]['socketid']
    print 'Sending Arguments: ' , type(args), args

    socketIO.emit('identity',{'uuid':UUID,'Socket Id': args[0]['socketid'], 'token':TOKEN})

def on_connect(*args):
    print 'Requesting WebSocket connection to SkyNet '

    socketIO.on('identity',on_id_response)
    socketIO.on('ready',on_ready_response)
    socketIO.on('not ready',on_noready_response)

socketIO = SocketIO(HOST,PORT)
socketIO.on('connect', on_connect)

socketIO.emit('status':on_status_response)
socketIO.emit('subscribe',{'uuid':UUID,'token':TOKEN},on_sub_respons

socketIO.emit('whoami',{'uuid':UUID,'token':TOKEN},on_whoami_respons

socketIO.emit('message',{devices': UUID, 'message': 'hello'},{pay

socketIO.emit('message',on_msg_response)

socketIO.wait()
```

②

**Fig. 5.3**

## 5.4 MESSAGING PLATFORM

**What is an IoT Messaging Protocol?**

- Internet of Things (IoT) messaging protocols are used to transmit telemetry (also known as Messages) from IoT devices to an IoT Messaging Hub. These protocols can operate over TCP, or even a higher level abstraction such as HTTPS.

- Connecting IoT devices to a network, the Internet, or even each other may use one of a number of methods. IoT devices may be connected over Wi-Fi, Cellular, Bluetooth, ZigBee, LoRaWAN, or some other connectivity method. Once the components of an IoT solution are connected to each other, there will be a messaging protocol used to transmit device telemetry (also known as messages) to and from devices.

**Most popular IoT Messaging Protocols:**

- The most popular IoT Messaging Protocols used by devices are listed below:

**1. MQTT (Message Queue Telemetry Transport):**

- The MQTT, or Message Queue Telemetry Transport, protocol is a lightweight, publish/subscribe network protocol for transporting telemetry messages between IoT devices. This protocol typically runs over TCP/IP, however, it can operate on top of other networking protocols so long as they provide ordered, lossless, bi-directional connections.
- MQTT is designed to be lightweight, and is ideal for connection scenarios where IoT devices may have limited bandwidth or other constraints requiring remote devices with small code footprints.

**Feature Highlights:**

- Lightweight protocol (great for constrained networks).
- Supports Publish / Subscribe messaging.
- Low power usage.
- Minimized data packet size.
- OASIS standard protocol.

**2. AMQP (Advanced Message Queue Protocol):**

- AMQP (Advanced Message Queue Protocol) is an open standard application layer protocol. It is a binary protocol designed to support a wide array of messaging applications and communications patterns. It is not specifically built for Internet of Things (IoT) solutions, but it works very well for message communications which include many IoT scenarios.

**Feature Highlights:**

- Binary application layer protocol.
- Can be used for Point-to-Point and Publish / Subscribe messaging.
- Broad compatibility with messaging scenarios.
- Supports end-to-end encryption of messaging.

**3. DDS (Data Distributed Service):**

- DDS (Data Distributed Service) protocol is designed to be used with real-time systems and is an Object Management Group (OMG) machine-to-machine standard. The goals of DDS are to enable dependable, high-performance, interoperable, real-time, scalable data exchanges using a Publish / Subscribe messaging pattern.

- The DDS protocol is designed to address the unique needs of application scenarios such as aerospace, defense, air-traffic control, autonomous vehicles, medical devices, robotics, power generation, transportation systems, and other real-time data exchange systems.

**Feature Highlights:**

- Designed for Real-Time systems.
- Provides Publish / Subscribe messaging.
- Connects devices directly to each other.
- Low overhead.

**4. XMPP (Extensible Messaging and Presence Protocol):**

- XMPP (Extensible Messaging and Presence Protocol) is a communications protocol based on XML (Extensible Markup Language). It's designed to provide near real-time exchange of structured XML data between two or more devices. More recently XMPP has been used for Publish / Subscribe messaging, VoIP (Voice over IP), Internet of Things (IoT), gaming, and other systems.

**Feature Highlights:**

- Designed to be extensible.
- Open Standard.
- Client / Server architecture.

**5. CoAP (Constrained Application Protocol):**

- The CoAP, or Constrained Application Protocol, is a specialized application protocol designed for constrained devices. It is designed to require low power, work across lossy networks, and can be used to connect devices to each other or other general nodes on the Internet. CoAP is not just used for IoT scenarios, but is also in use on other systems such as SMS on mobile communication networks.

**Feature Highlights:**

- Low Power usage.
- Used on constrained devices.

## 5.5 INTRODUCTION TO RESTFUL WEB SERVICES - gRPC, SOAP

### 5.5.1 gRPC

**History of the gRPC:**

- In 2015, Google developed gRPC as an extension of the RPC framework to link many microservices created with different technologies. Initially, it was closely associated with Google's internal infrastructure, but later, it was made open-source and standardized for community use. During the first year of its release, it was leveraged

by top organizations to power use cases from microservices to web, mobile, and IoT. And in 2017, it became the Cloud Native Computing Foundation (CNCF) incubation project due to its increasing popularity.

- **gRPC** is a framework for implementing RPC APIs via HTTP/2. To understand what this means in comparison to other API building styles, let's look at API design's timeline first.
- Traditionally, there have been two distinct ways to build APIs: RPC and REST. There's also SOAP, an ancient protocol that is mostly used in strictly standardized operations like banking and aviation.
- **RPC (Remote Procedure Call)** is classic and the oldest API style currently in use. It uses procedure calls to request a service from a remote server the same way you would request a local system via direct actions to the server (like SendUserMessages, Locate Vehicle, add Entry). RPC is an efficient way to build APIs; RPC messages are lightweight and the interactions are straightforward.
- When **REST (Representational State Transfer)** was introduced in 2000, it was meant to solve this problem and make APIs more accessible. Namely, it provided a uniform way to access data indirectly, via resources, using generic HTTP methods such as GET, POST, PUT, DELETE, and so on. REST APIs are basically self-descriptive. This was the main difference between RPC and REST since with RPC, you address the procedures and there's little predictability of what procedures in different systems may be.
- Although REST presented an improved format for interacting with many systems, it returned a lot of metadata, which was the main reason it couldn't replace simple and lightweight RPC. This was also one of the reasons two new approaches emerged later: Facebook's GraphQL and gRPC from Google.

**gRPC Main Concepts:**

- gRPC owes its success to the employment of two techniques: using HTTP/2 instead of HTTP/1.1 and protocol buffers as an alternative to XML and JSON. Most of the gRPC benefits stem from using these technologies.
- gRPC is a robust open-source RPC (Remote Procedure Call) framework used to build scalable and fast APIs. It allows the client and server applications to communicate transparently and develop connected systems. Many leading tech firms have adopted gRPC, such as Google, Netflix, Square, IBM, Cisco, & Dropbox. This framework relies on HTTP/2, protocol buffers, and other modern technology stacks to ensure maximum API security, performance, and scalability.

**Protocol Buffers for defining Schema:**

- Protocol buffers are a popular technology for structuring messages developed and used in nearly all inter-machine communication at Google. In gRPC, protocol buffers (or protobufs) are used instead of XML or JSON in REST. Let us see how they work.

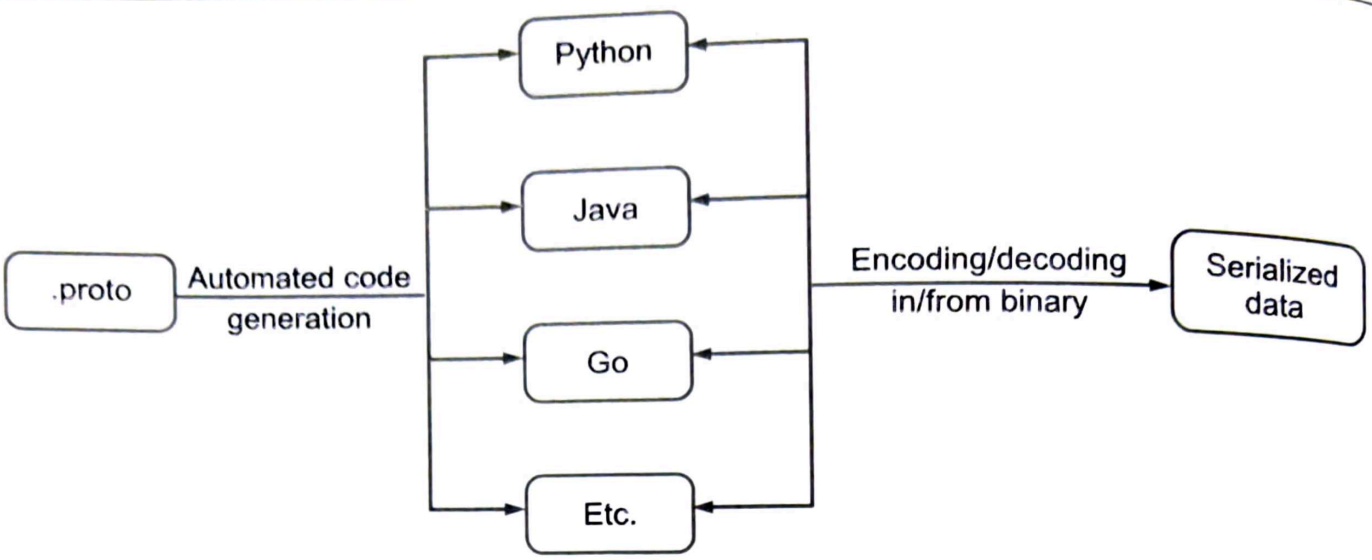


**Fig. 5.4: Data serialization with Protocol Buffers**

1. In a .proto text file, a programmer defines a schema — how they want data to be structured. They use numbers, not field names, to save storage. They can also embed documentation in the schema.
2. Using a protoc compiler, this file is then automatically compiled into any of the numerous supported languages like Java, C++, Python, Go, Dart, Objective-C, Ruby, and more.
3. At runtime, messages are compressed and serialized in binary format.

**Benefits of gRPC:**

- gRPC offers a refreshed take on the old RPC design method by making it interoperable, modern, and efficient using such technologies as Protocol Buffers and HTTP/2.

Following are the benefits of gRPC:

1. **Lightweight Messages:** Depending on the type of call, gRPC-specific messages can be up to 30 percent smaller in size than JSON messages.
2. **High performance:** By different evaluations, gRPC is 5, 7, and even 8 times faster than REST+JSON communication.
3. **Built-in code generation:** gRPC has automated code generation in different programming languages including Java, C++, Python, Go, Dart, Objective-C, Ruby, and more.
4. **More Connection Options**: While REST focuses on request-response architecture, gRPC provides support for data streaming with event-driven architectures: server-side streaming, client-side streaming, and bidirectional streaming.

**Drawbacks of gRPC:**

- It is fair to hope that gRPC disadvantages will be overcome with time. Unfortunately, they hamper its mainstream adoption in a big way.

1. **Lack of Maturity:**

- The maturity of specific technology can be a big obstacle to its adoption. That's apparent with gRPC too. Compared to its peer GraphQL With minimal developer support outside of Google and not many tools created for HTTP/2 and protocol buffers, the community lacks information about best practices, workarounds, and success stories. But this is likely to change soon. As with many open-source technologies backed by big corporations, the community will keep growing and attracting more developers.

2. **Limited Browser Support:**

- Since gRPC heavily relies on HTTP/2, user can't call a gRPC service from a web browser directly, because no modern browsers can access HTTP/2 frames. So, user needs to use a proxy, which has its limitations. There are currently two implementations:
  - gRPC web client by Google implemented in JavaScript with an upcoming support for Python, Java, and more languages.
  - gRPC web client by the team from the improbable for TypeScript and Go.

3. **Not Human-readable Format:**

- By compressing data to a binary format, *protobuf* files become non-human readable, unlike XML and JSON. To analyze payloads, perform debugging, and write manual requests, required extra tools like gRPC command line tool and server reflection protocol. Although *protobuf* files can be converted into JSON out of the box.

**When to use gRPC?**

- Conditions for using gRPC instead of REST:
  - Real-time communication services where you deal with streaming calls.
  - When efficient communication is a goal.
  - In multi-language environments.
  - For internal APIs where you do not have to force technology choices on clients.
  - New builds as part of transforming the existing RPC API might not be worth it.

**gRPC Architecture:**

- In the following gRPC architecture diagram, we have the gRPC client and server sides. In gRPC, every client service includes a stub (auto-generated files), similar to an interface containing the current remote procedures. The gRPC client makes the local procedure call to the stub with parameters to be sent to the server. The client stub then serializes the parameters with the marshalling process using *Protobuf* and forwards the request to the local client-time library in the local machine.

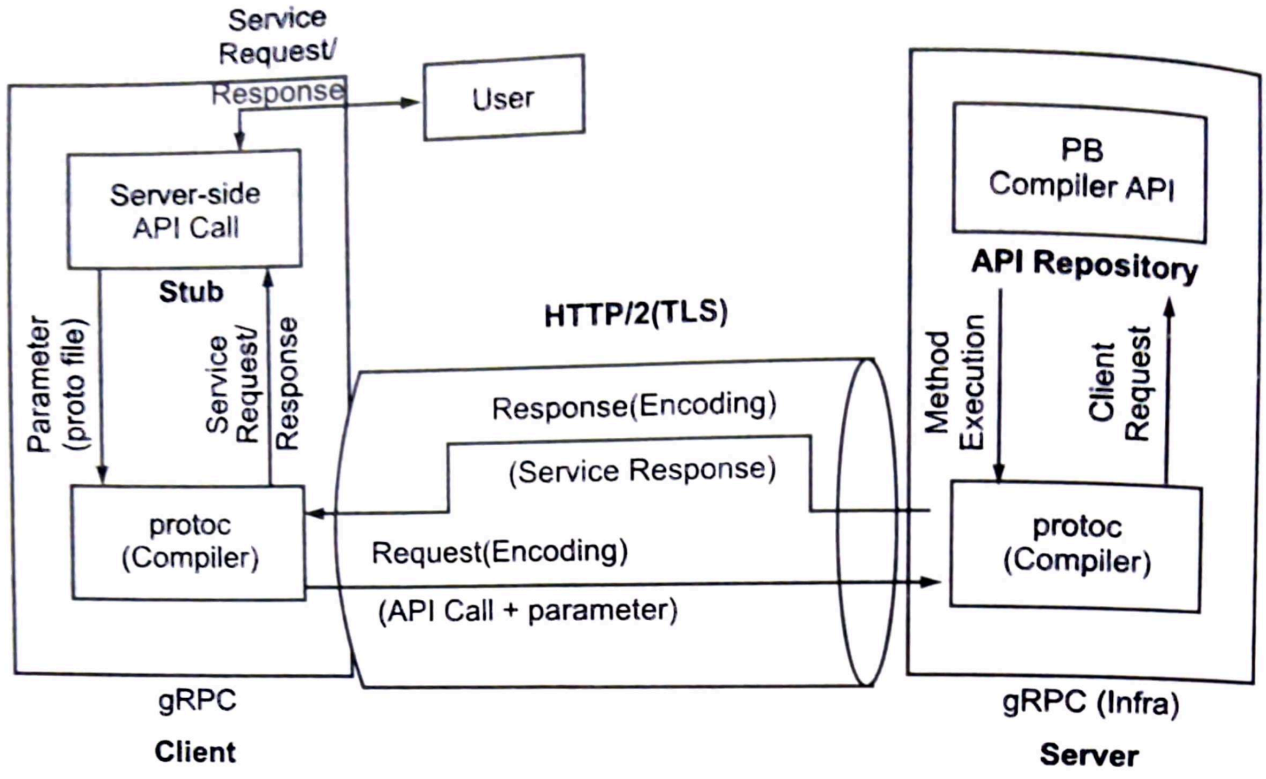


**Fig. 5.5: gRPC Architecture**

- The OS makes a call to the remote server machine via HTTP/2 protocol. The server's OS receives the packets and calls the server stub procedure, which decodes the received parameters and executes the respective procedure invocation using *Protobuf*. The server stub then sends back the encoded response to the client transport layer. The client stub gets back the result message and unpacks the returned parameters, and the execution returns to the caller.

## 5.5.2 SOAP

**Introduction to SOAP:**

- SOAP (Simple Object Access Protocol) is an XML-based messaging protocol for exchanging information among computers. SOAP is an application of the XML specification.
- SOAP is a communication protocol designed to communicate via Internet and can extend HTTP for XML messaging. It provides data transport for Web services and used for broadcasting a message and platform, language-independent.
- SOAP enables client applications to easily connect to remote services and invoke remote methods.
- Although SOAP can be used in a variety of messaging systems and can be delivered via a variety of transport protocols, the initial focus of SOAP is remote procedure calls transported via HTTP.
- Other frameworks including CORBA, DCOM, and Java RMI provide similar functionality to SOAP, but SOAP messages are written entirely in XML and are therefore uniquely platform- and language-independent.

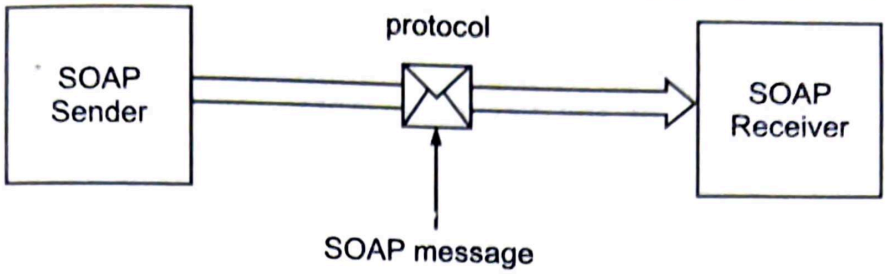


**Fig. 5.6: SOAP**

**What is SOAP (Simple Object Access Protocol)?**

- SOAP (Simple Object Access Protocol) is a message protocol that enables the distributed elements of an application to communicate. SOAP can be carried over a variety of standard protocols, including the web-related Hypertext Transfer Protocol (HTTP).
- SOAP was developed as an intermediate language for applications that have different programming languages, enabling these applications to communicate with each other over the internet. SOAP is flexible and independent, which enables developers to write SOAP application programming interfaces (APIs) in different languages while also adding features and functionality.
- SOAP is a lightweight protocol used to create web APIs, usually with Extensible Markup Language (XML). It supports a wide range of communication protocols across the internet, HTTP, Simple Mail Transfer Protocol (SMTP) and Transmission Control Protocol. The SOAP approach defines how a SOAP message is processed, the features and modules included, the communication protocols supported and the construction of SOAP messages. SOAP uses the XML Information Set as a message format and relies on application layer protocols, like HTTP, for message transmission and negotiation.

**SOAP Building Blocks and Message Structure:**

- Simple Object Access Protocol, as a specification, defines SOAP messages that are sent to web services and client applications. SOAP messages are XML documents that are comprised of the following three basic building blocks:

1. The SOAP **Envelope** encapsulates all the data in a message and identifies the XML document as a SOAP message.
2. The **Header** element contains additional information about the SOAP message. This information could be authentication credentials, for example, which are used by the calling application.
3. The **Body** element includes the details of the actual message that need to be sent from the web service to the calling application. This data includes call and response information.

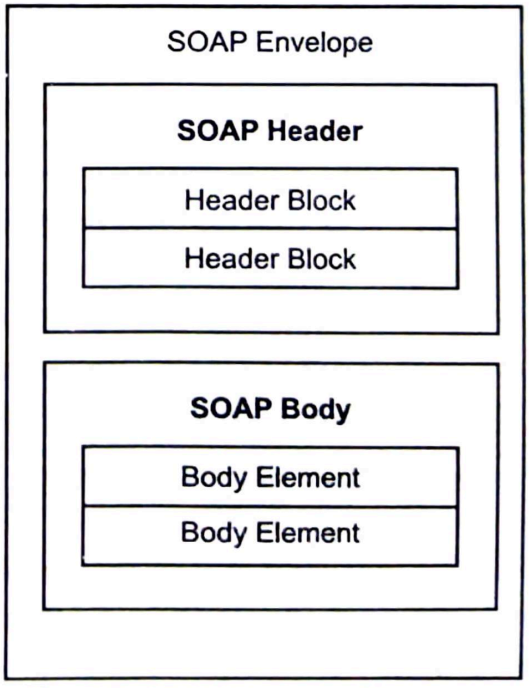


**Fig. 5.7: SOAP Message Structure**

**SOAP Advantages and Disadvantages:**

- SOAP is an integral part of the Service-oriented Architecture (SOA) and the web services specifications.

**Advantages of SOAP:**

- **Platform- and Operating System-Independent:** SOAP can be carried over a variety of protocols, enabling communication between applications with different programming languages on both Windows and Linux.
- **Works on the HTTP Protocol:** Even though SOAP works with many different protocols, HTTP is the default protocol used by web applications.
- **Can be transmitted through different Network and Security devices:** SOAP can be easily passed through firewalls, where other protocols might require a special accommodation.

**Disadvantages:**

- **No provision for passing data by reference:** This can cause synchronization issues if multiple copies of the same object are passed simultaneously.
- **Speed:** The data structure of SOAP is based on XML. XML is largely human-readable, which makes it fairly easy to understand a SOAP message. However, that also makes the messages relatively large compared to the Common Object Request Broker Architecture (CORBA) and its remote procedure call (RPC) protocol that will accommodate binary data. Because of this, CORBA and RPC are faster.
- **Not as flexible as other methods:** Although SOAP is flexible, newer methods such as RESTful architecture, use XML, JavaScript Object Notation, YAML or any parser needed, which makes them more flexible than SOAP.

## Summary

- Cloud storage allows user to save data and files in an off-site location that user can access either through the public internet or a dedicated private network connection.
- Communication API facilitates data transfer, control information transfer from application to cloud, one service to another.
- There are some challenges of cloud storage API.
- Some popular examples of Cloud storage API are Amazon Web Service (AWS), Google Drive API.
- An IoT cloud is a massive network that supports IoT devices and applications. This includes the underlying infrastructure, servers and storage needed for real-time operations and processing.

- AWS IoT lets user select the most appropriate and up-to-date technologies for user solution. To manage and support users IoT devices in the field, AWS IoT Core supports these protocols:
  - MQTT (Message Queuing and Telemetry Transport)
  - MQTT over WSS (Websockets Secure)
  - HTTPS (Hypertext Transfer Protocol - Secure)
  - LoRaWAN (Long Range Wide Area Network)
- Key categories of IoT products on AWS are Devices Software, Connectivity & Control Services, Analytics Services etc.
- SkyNet is an open protocol for hosting data and web applications on the decentralized web.
- SkyNet decentralizes "the cloud" so that user and application data is not stored by (and only accessible to) a single, central authority.
- In 2015, Google developed gRPC as an extension of the RPC framework to link many microservices created with different technologies.
- SOAP is a lightweight protocol used to create web APIs, usually with Extensible Markup Language (XML).

## Check Your Understanding

1. Which of the following is the simplest unmanaged cloud storage device?
   (a) File transfer utility (b) Antivirus utility
   (c) Online image utility (d) None of the mentioned
2. How many categories of storage devices broadly exist in the Cloud?
   (a) 1 (b) 2
   (c) 3 (d) None of the mentioned
3. Point out the wrong statement.
   (a) Some APIs are both exposed as SOAP and REST.
   (b) The role of a cloud vendor specific API has impact on porting an application.
   (c) There are mainly three types of cloud storage.
   (d) None of the mentioned
4. In which year, Amazon Web Services founded?
   (a) 2005 (b) 2006
   (c) 2007 (d) 2008
5. API enables services portability between _____.
   (a) Systems (b) Devices
   (c) Networks (d) Services

6. Identify the java extension file in IoT.
   (a) .c (b) .py
   (c) .exe (d) .jar
7. Which of the following is not an HTTP method?
   (a) CREATE (b) POST
   (c) PUT (d) OPTION
8. Which of the following is a components of a Web Service Architecture?
   (a) WSDL (b) UDDI
   (c) SOAP (d) All of the mentioned
9. SOAP is a format for sending messages and is called as _____.
   (a) Network protocol (b) Communication protocol
   (c) Data transfer protocol (d) None of the mentioned
10. _____ is used to convert user application into web application.
   (a) Web services (b) Java service
   (c) Browser Action (d) Struts Services

**Answers**

| | | | | |
|---|---|---|---|---|
| 1. (a) | 2. (b) | 3. (c) | 4. (b) | 5. (a) |
| 6. (d) | 7. (a) | 8. (d) | 9. (b) | 10. (a) |

## Practice Questions

**Q.I Answer the following questions in short.**

1. What is cloud storage?
2. List examples of cloud storage API.
3. What is meant by SkyNet?
4. List popular IoT messaging protocol used by devices.
5. Explain the message structure of SOAP.

**Q.II Answer the following questions.**

1. Explain role of Cloud Computing in IoT.
2. Explain in detail Communication API.
3. Write a short note on Massaging Platform- MQTT and AMQP.
4. Explain SOAP with its advantages and disadvantages.
5. Explain AWS for IoT in detail.

**Q.III Define the terms.**

1. IoT Cloud
2. SkyNet
3. Remote Procedure Call(RPC)
4. SOAP (Simple Object Access Protocol)
5. MQTT (Message Queue Telemetry Transport)

***

# 6...

# Security in IoT

## Learning Objectives...

- To understand the design stage in perspective of security of the IoT.
- To understand the challenges to secure IoT.
- To understand key elements of IoT Security for the designing of tamper-proof product.

## 6.1 INTRODUCTION

- Companies are increasingly taking benefit of IoT devices to improve productivity and increase visibility into their operations. As a result, growing numbers of networked devices organized on corporate networks have access to sensitive data and critical systems. Often, these devices have security issues that make them vulnerable to attack and place the rest of the organization at risk. For example, cyber threat actors commonly target unprotected printers, smart lighting, IP cameras, and other networked devices to gain access to an organization's network. From there, they can move laterally through the network to access more critical devices and sensitive data and create ransomware and double extortion cyber-attacks that can render a business network useless.
- Securing the company against cyber threats requires securing all devices connected to the corporate network. IoT security is a vital component of a corporate cyber security strategy because it limits the risks posed by these insecure, networked devices.

## 6.2 VULNERABILITIES OF IoT

- IoT has brought a fundamental shift and benefits to how we interact with devices and how those devices interact with each other. Though, it is important to be aware of IoT cyber security vulnerabilities.
- Let us see vulnerabilities by implementing simple strategies and tools which can help you secure devices and detect attacks ahead of time.

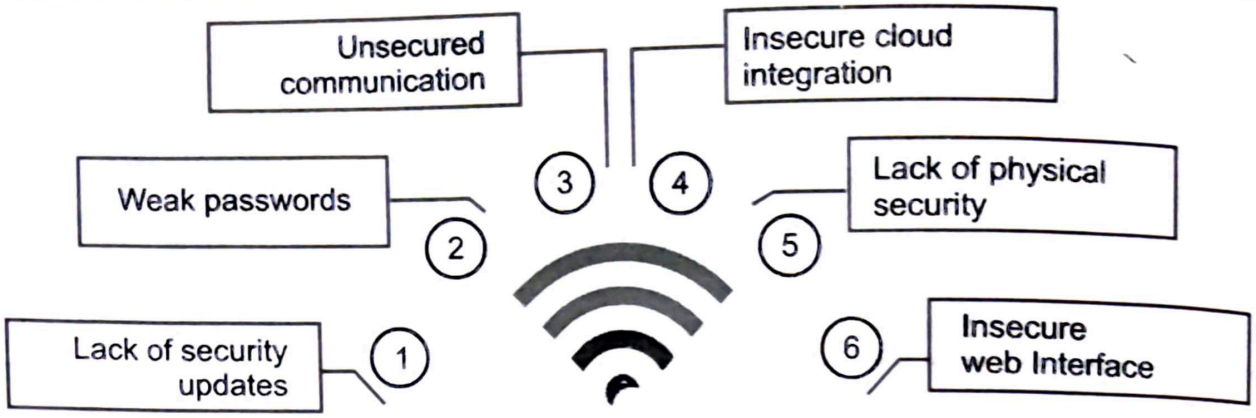


**Fig. 6.1: Vulnerabilities of IoT Devices**

**(i) Lack of Security Updates:**

- Many IoT devices are not designed with security updates in mind. This means once a device is released; it is rarely updated with the latest security patches. This leaves devices vulnerable to attacks that exploit known vulnerabilities.

**(ii) Weak Passwords :**

- One of the biggest IoT security challenge is weak password security. It is important to create a strong password while creating new online accounts. Unfortunately, not many IoT devices are password-protected. Users generally tend to go with default passwords or options that are easy to guess, leaving these devices vulnerable to breaches. Additionally, many IoT devices such as smart watches rely on ID verification using biometric systems. While these can be more secure than using passwords that are easy to guess, the verification data needs to be stored and managed more securely.

**(iii) Unsecured Communication:**

- Many IoT devices communicate using unencrypted protocols. This can make it vulnerable to interception and tampering such as SSL (Secure Sockets Layer) downgrade attack by malicious actors.

**(iv) Insecure Cloud Integration:**

- Many IoT devices depend on cloud-based services to store and process data. If these services are not secured properly, it can leave the device vulnerable to attacks. Attacks such as Directory Listening can also expose sensitive data to threat actors.

**(v) Lack of Physical Security:**

- Some IoT devices have poor physical security measures, such as weak or non-existent passwords on the device itself. It makes easy for hackers to gain access to them. This is especially regarding for devices that are located in public places, such as Smart city sensors or Parking meters.

**(vi) Insecure Web Interface:**

- Insecure web interfaces on Internet of Things (IoT) devices can pose a serious security risk to individuals and organizations. These interfaces are often used to

access and control various functions of the device, such as changing settings, monitoring performance. However, if these interfaces are not properly secured, they can be exploited by malicious actors to gain unauthorized access to the device and potentially compromise its functionality or data.

## 6.2.1 Security Requirements

- Addressing some vulnerability by implementing simple strategies and tools can help you secure your devices and detect attacks ahead of time.

1. **Weak Passwords, Settings, and ID Verification:**

- Weak password security is one of enterprises' biggest IoT security challenges.

**Solution:** It is highly recommended that all IoT devices have at least 8-character long passwords using a combination of special characters, numbers, and upper and lowercase alphabets. Commonly used passwords like "12345", "password," etc., are discouraged. You must also install reliable firewalls in your IoT devices to protect data stored or transferred between devices.

2. **Outdated Software:**

- Using deprecated software is one of the most significant IoT security vulnerabilities, making the devices and the entire IoT ecosystem easy to get compromised.

**Solution:** Using outdated software is a weak link in overall IoT security, and it is imperative to ensure that the IoT devices do not run on outdated software and are regularly updated and upgraded (when necessary) to keep the data safe.

3. **Mismanagement of IoT Connectivity:**

- IoT enterprises do not always have a single view of all their connected devices for visualization, monitoring and operational control purposes, especially when devices are spread across multiple platforms.

**Solution:** One central connectivity management platform to manage and protect all their cellular connected assets, in one place.

4. **Lack of IoT Security Knowledge and Protocols:**

- Lack of awareness and sufficient IoT security knowledge can greatly hinder securing IoT devices.

**Solution:** Businesses must follow the pre-established cyber security frameworks like ISO and NIST cyber security framework and refer to them as authentic sources of information. Training the employees on all security best practices is another great undertaking that can help secure the ecosystem in the long run.

5. **Poor Data Protection:**

- IoT devices thrive on data, and vast amounts of it include confidential and sensitive information. This data is collected and exchanged between devices at any given time and is often stored in the cloud.

**Solution:** It is required that all externally sourced components are built securely and follow industry-standard encryption paradigms. A connectivity management platform also helps protect and manage all cellular data from a single platform while being transferred to the network, making sure it reaches the right hands.

## 6.2.2 Challenges for Secure IoT

- People are surrounded by IoT devices, such as IoT cars, Smart refrigerators, and other smart home products. These devices exchanging data through connected sensors. There are even smart machines now, networking in every industry right from manufacturing to retail. This large magnitude of growth comes with its share of cyber threats. No matter if it is as large as a manufacturing robot or as small as an electronic chip – without security, every IoT device can be hacked. The more amazing fact is that it can only take five minutes for an IoT device to be attacked.

**Top IoT Security Challenges and Solutions:**

1. **Poor Compliance from Manufacturers:**

- This is one of the main security issues with IoT devices. Even a small device as a fitness tracker can expose Gmail Login credentials if it lacks compliance from the manufacturer's side. Since there are no universal IoT security standards, there is not security policy to stop manufacturers from working on the next product instead of providing updates for the old one. Even if they provide updates, it lasts for a short time.

**Solution:** Thus to protect consumers against attacks, each device should be tested before launching and updated regularly. In fact, companies should make security a crucial element in their product design process. Other measures that manufacturer can take are:

- Use quality hardware and firmware.
- Use updated operating systems and software.
- Secure data transfer and storage.
- Encourage users to use strong coded passwords.
- Provide them with regular automatic updates.

2. **Lack of Awareness among Users:**

- IoT is a growing technology and people are still new to it. They might have learned to protect their emails and personal computers with strong passwords, but securing a connected device is still alienating for most users. Insufficient knowledge and awareness regarding IoT functionality is the reason for this. This allows attackers to collect personal or confidential information and then exploit it for dark web profits.

**Solution:** Since IoT involves consumer-oriented applications, it is important to provide them with adequate knowledge on how to use these products. A few of the principles that users can follow are:

- Changing default passwords and user names.
- Changing them regularly every 30 to 90 days.
- Using multi-factor authentication to increase the level of security.
- Keeping software updated.
- Using secure internet connections.

3. **Lack of Physical Hardening:**

- Physical hardening is one of the important aspects of IoT security. Since IoT devices operate autonomously without any human intervention, they need to be secured physically from threats. Such attacks may not be intended to damage the device but rather extract information from it. Even a microSD card can be a goldmine for the attacker revealing all the private data and passwords.

**Solution:** Ensuring the physical hardening of an IoT device begins with the manufacturers. However, users are equally responsible for keeping IoT devices physically safe. One way to do so is to keep keys in Trusted Platform Modules (TPMs) and Trusted Execution Environments (TEE). TPM is a chip installed near the CPU on an IoT device. It is mostly used for cryptographic operations like generating a security key, saving it, storing data, and so on.

4. **Botnet Attacks:**

- A botnet is a network of connected devices hijacked by malware that allows hackers to carry out various scams. These bots serve as a tool to automate mass attacks such as unauthorized access, server crashing, data theft, and DDoS (Distributed Denial of Services) attacks.

**Solution:** The following steps are suggested to protect IoT devices against botnet attacks right from production to retirement,

- Make sure your IoT device has built-in security features – into architecture, interfaces, and designs.
- Create a separate network solely for your IoT devices. Also, use intrusion prevention systems as a third-party firewall.
- Use the router's built-in security features to protect all devices in that network.
- Disable unused features.
- Use comprehensive security software.

5. **Ransomware Attack:**

- Ransomware is one of the worst malware types practiced by hackers. They do not destroy sensitive files or data but instead holds the victim's information at ransom to demand money. This practice is evolving and IoT devices with poor security can be easily targeted. Unprotected IoT devices such as wearables, healthcare gadgets, smart home products, and other smart equipment are at a risk in such attacks.

**Solution:** The best practices to prevent such attacks are:

- Conduct mini risk assessment.
- Carefully read the privacy statements.
- Have a recovery plan.
- Use different passwords for every internet-connected device.
- Deactivate your device when not in use.

**6. Rogue IoT Devices:**

- Employees usually bring their own IoT-connected devices to the workplace. If these personal devices are not secured they can risk the entire organization, exposing it to cyber attacks.

**Solution:** To protect against such rogue gadgets, the following precautions are recommended,

- Restricting who can physically connect to your network infrastructure should be the first line of defense.
- Identify the authorized devices that connect to the network and ensure it is free from any suspicious or unknown web traffic.
- Create a whitelist of devices. Only these devices will be able to connect to the network.
- Monitor network access for performance, capacity, and compliance.

**7. Crypto Mining with IoT Bots:**

- IoT botnets infect home routers and other Internet of Things as well as attempt to mine for crypto currency. These hijacked devices allow criminals to rake in crypto-cash while the device owners remain unaware that their gadgets are being used to produce crypto coins.
- In February 2019, more than half a million computing devices were hijacked by a crypto currency miner botnet called Smominru, forcing the various devices to mine nearly 9,000 Monero crypto coins without the knowledge of the owners of the devices, according to technology portal ZDNet.

**Solution:** These botnets are generally released on a private network of interconnected computers as it enhances their computational process to mine cryptocurrencies. Because of its distributed nature, taking down botnet is very difficult. The best protection from infection is strong patching schedules and layered security.

- Another possible solution is Blockchain IoT security, which is most familiar for bitcoin. Blockchains contain strong protections against data tampering, locking access to the IoT devices, and only allowing whitelisted devices into the network. This way, it creates a permission private network, specifically designed for IoT security.

## 6.2.3 Threat Modeling

- A threat could be a statement to a person, thing, or device that could impose dangerous or unlikely situations on them. A threat poses no danger on its own, for a threat to matter it must have some sort of impact.
- IoT threat modelling is the activity of identifying and quantifying the security risks associated with an IoT product and its surrounding ecosystem. Ideally, threat modelling should be done during the product design phase in what experts call "security by design". It is very common to find companies that elaborate their risk assessments and threat modelling once the product is developed.

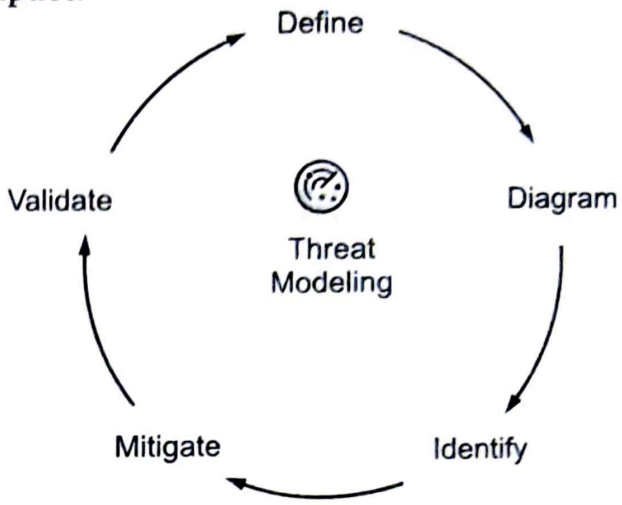


**Fig. 6.2: Threat Modeling**

- Threat Modelling is a process using which one can systematically identify any possible attacks against any device and then prioritize certain issues against their severity.
- The most common way to use threat modelling in your security assessments is to follow a FRAMEWORK. There are multiple frameworks out there among which most common and effective are STRIDE and DREAD.

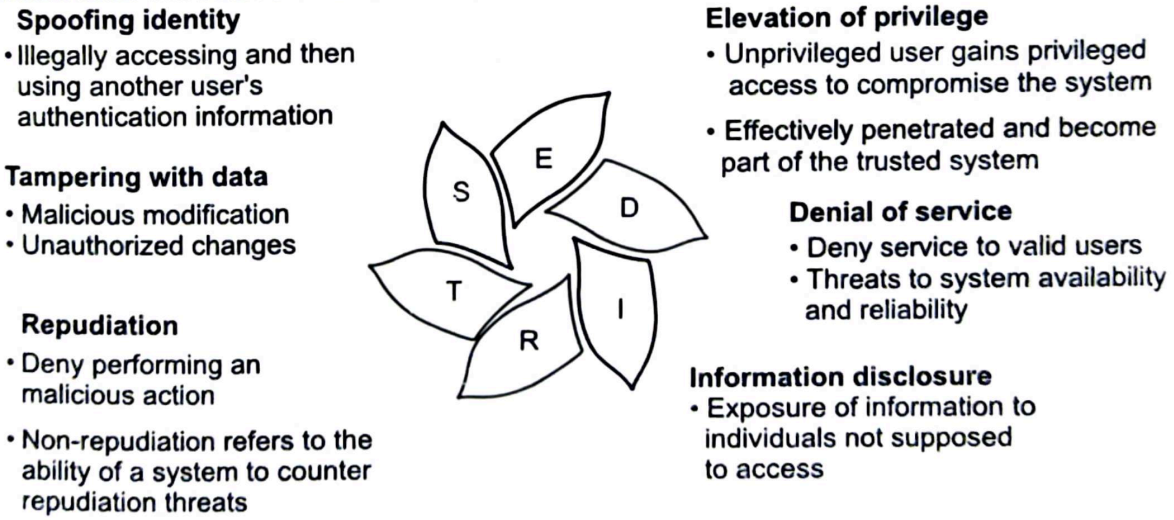


**Fig. 6.3: STRIDE Threat Model**

- STRIDE is a threat classification model and only focuses on identifying weaknesses in the technology and not on vulnerable assets or possible attackers. The STRIDE acronym represents the following threats:
  - **Spoofing:** When an actor pretends to play the role of a system component.
  - **Tampering:** When an actor violates the integrity of data or a system.
  - **Repudiation:** When users can deny they took certain actions on the system.
  - **Information Disclosure:** When an actor violates the confidentiality of the system's data.

- **Denial of Service:** When an actor disrupts the availability of a system's component or the system as a whole.
- **Elevation of Privilege:** When users or system components can elevate themselves to a privilege level they shouldn't have access to.

**The DREAD classification scheme:**

- DREAD is a risk rating system and it helps evaluate the risk posed by each threat discovered.

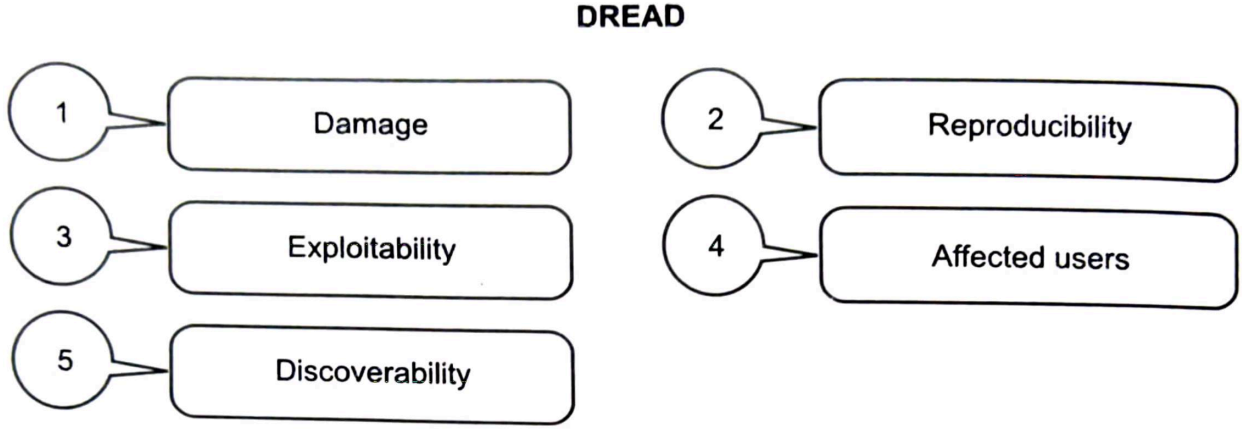


**Fig. 6.4: DREAD Acronym**

- The DREAD acronym represents the following criteria:
  - **Damage**: How damaging the exploitation of this threat would be.
  - **Reproducibility**: How easy the exploit is to reproduce.
  - **Exploitability**: How easy the threat is to exploit.
  - **Affected Users**: How many users would be affected?
  - **Discoverability**: How easy it is to identify the threat.

## 6.3 KEY ELEMENTS OF IoT SECURITY

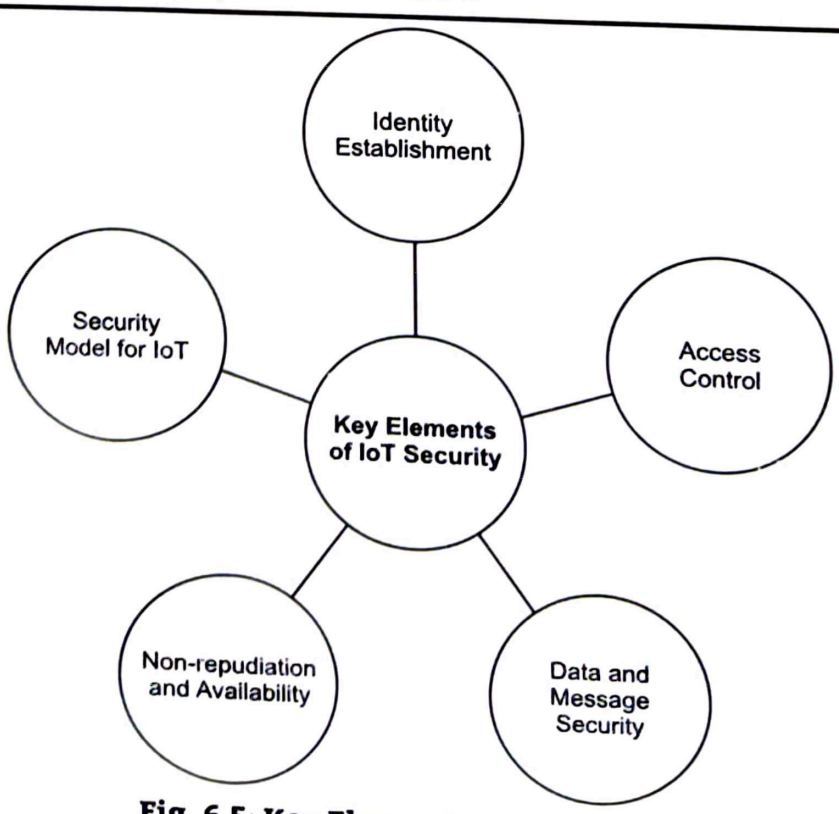


**Fig. 6.5: Key Elements of IoT Security**

- The main elements of IoT security are:

1. **Identity Establishment:** An IoT device needs to prove its identity to other networked devices and to verify the identity of all other networked devices.
2. **Access Control:** Access control is an essential element of security that determines who is allowed to access certain data, apps, and resources and in what circumstances. In the same way that keys and pre-approved guest lists protect physical spaces, access control policies protect digital spaces.
3. **Data and Message Security:** Data security generally suffers from packet sniffing. Sniffing attack begin when a computer is compromised to sharing some data or program.

- Threats to message security fall into three categories:
  (a) Confidentiality
  (b) Integrity
  (c) Authentication

4. **Non-repudiation and Availability:**

- Non-repudiation assurances that a sender and receiver of digital information or a message cannot reject the message and their involvement during the communication. The proof of delivery assurances the sender that the user has received the message.
- Non-repudiation is a legal concept that is widely used in information security and refers to a service, which provides proof of the origin of data and the integrity of the data. In other words, non-repudiation makes it very difficult to successfully deny who/where a message came from as well as the authenticity and integrity of that message.
- Digital signatures (combined with other measures) can offer non-repudiation when it comes to online transactions, where it is crucial to ensure that a party to a contract or a communication can't deny the authenticity of their signature on a document or sending the communication in the first place.

## 6.4 SECURITY MODEL FOR IoT

- Since the Internet of Things (IoT) is not a standard, there is no single standardized approach to security. There are multiple IoT reference models defined by various stakeholders including ITU-T, Cisco, Intel, IBM, Microsoft, Symantec, and others. Security is often considered in these reference models.
- An IoT security model can be seen in two viewpoints:
  (a) **In a layered architecture**, there is a security layer that spans the entire stack, from the connectivity layer at the bottom to the application layer at the top.
  (b) **In an end-to-end solution**, security is implemented at all points, from end devices to network to cloud.

- Models provide a formal framework to implement security and evaluate the maturity of those implementations.

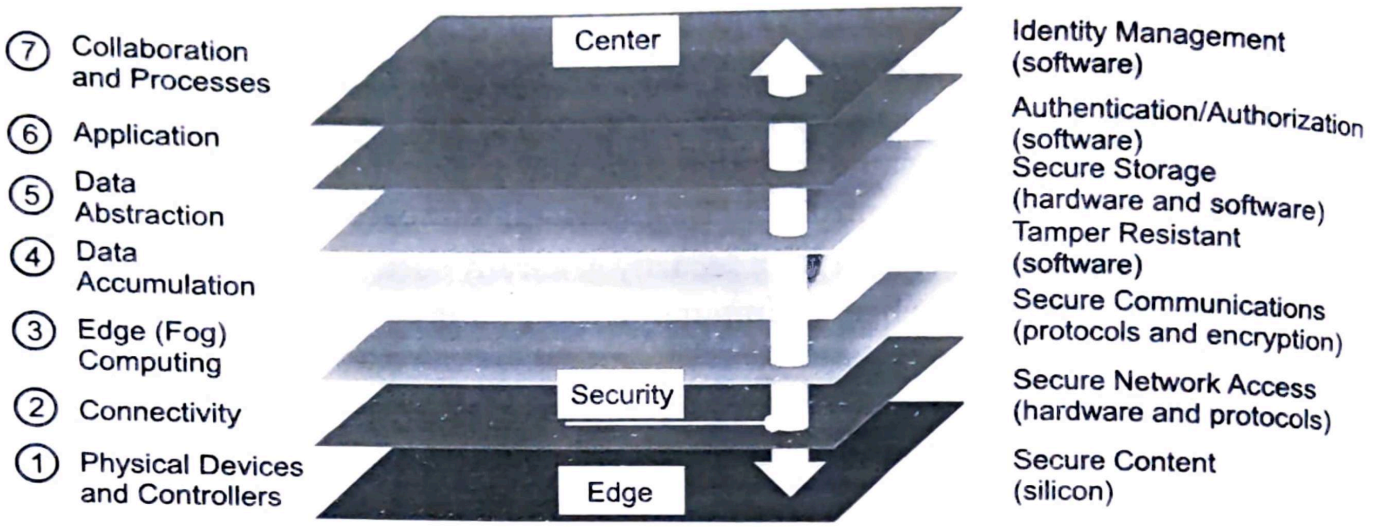


**Fig. 6.6: IoT Reference Model : Security**

- IoT security model is a layered architecture with the lowest layers being device centric and the highest layers being more cloud centric.
- For control, information flows top to bottom. For monitoring, the flow is opposite. Security is considered at each layer.
- The lower layers focus on giving secure access to physical hardware while also providing a trusted environment for code execution.
- At higher layers, the focus is on identity management, authentication, analytics, and so on.

## 6.5 CHALLENGES IN DESIGNING IOT APPLICATIONS

- Design challenges in IoT (Internet of Things) refer to the technical difficulties and trade-offs involved in creating connected devices that are both functional and secure. Some of the key design challenges in IoT include:

1. **Interoperability:** Interoperability refers to the ability of different systems, devices, or components to work together seamlessly and exchange data effectively. In the context of the Internet of Things (IoT), interoperability is a critical challenge, as a large number of various devices are being connected to the internet. The lack of standardization in the IoT can lead to difficulties in communication and data exchange between devices, resulting in the fragmented and inefficient system.
   - To overcome this challenge, organizations and industry groups are working to establish standards and protocols to ensure interoperability between IoT devices. This includes the development of common communication protocols, data formats, and security standards. Interoperability is important for enabling the full potential of the IoT and allowing connected devices to work together effectively and efficiently. Ensuring that different IoT devices can work together seamlessly and exchange data effectively.

2. **Security:** Security is a critical concern in the Internet of Things (IoT) as it involves the protection of sensitive data and systems from unauthorized access, theft, or damage. IoT devices are often vulnerable to cyber-attacks due to their increased exposure to the internet and their limited computing resources. Some of the security challenges in IoT include:
   (i) **Device security**: Ensuring that IoT devices are protected from malware and unauthorized access.
   (ii) **Network security**: Protecting the communication between IoT devices and the network from cyber-attacks.
   (iii) **Data security**: Securing the data collected and transmitted by IoT devices from unauthorized access or tampering.
   (iv) **Privacy**: Protecting the privacy of individuals whose personal information is collected and transmitted by IoT devices.

- To address these security challenges, organizations should implement robust security measures such as encryption, firewalls, and regular software updates.
- Additionally, they should conduct regular security audits and assessments to identify and address potential security risks. By prioritizing security, organizations can help to protect the sensitive data and systems involved in IoT and reduce the risk of cyber-attacks. Protecting IoT devices and the sensitive data they collect and transmit from cyber threats and unauthorized access.

3. **Scalability:** Scalability refers to the ability of a system to handle increasing workloads or numbers of users without a significant decline in performance. In the context of the Internet of Things (IoT), scalability is a major challenge as the number of connected devices is rapidly growing, leading to an increased volume of data and communication.

- Scalability challenges in IoT include:
  1. **Data management:** Effectively managing and storing the large amounts of data generated by IoT devices.
  2. **Network Capacity:** Ensuring that networks have sufficient capacity to handle the increased volume of data and communication.
  3. **Device Management:** Efficiently managing the growing number of IoT devices and ensuring that they can be easily configured and maintained.
- To address these scalability challenges, organizations should adopt scalable architectures, such as cloud computing, that can accommodate the growing number of IoT devices and the data they generate. Additionally, they should implement efficient data management and storage solutions, such as distributed databases and data lakes, to handle the increased volume of data. By

prioritizing scalability, organizations can ensure that their IoT systems can handle the growing number of connected devices and continue to deliver high performance and efficiency. Designing systems that can accommodate large numbers of connected devices and manage the resulting data flow effectively.

4. **Reliability:** Reliability refers to the ability of a system to perform its intended function consistently and without failure over time. In the context of the Internet of Things (IoT), reliability is a critical concern, as the failure of even a single IoT device can have significant consequences. Some of the reliability challenges in IoT include:
   1. **Device Failure:** Ensuring that IoT devices are designed and built to be reliable and function correctly even in harsh environments.
   2. **Network Connectivity:** Maintaining stable and reliable connections between IoT devices and the network, even in the face of hardware or software failures.
   3. **Data Accuracy:** Ensuring that the data collected and transmitted by IoT devices is accurate and reliable.

   - To address these reliability challenges, organizations should implement robust and reliable hardware and software designs for IoT devices, and conduct regular testing and maintenance to identify and resolve any issues. They should also implement redundant systems and failover mechanisms to ensure that the system continues to function in the event of a failure. By prioritizing reliability, organizations can help ensure that their IoT systems perform consistently and without failure, delivering the intended benefits and results. Ensuring that IoT systems remain functional and accessible even in the face of hardware or software failures.

5. **Power Consumption:** Power consumption refers to the amount of energy that a system or device uses. In the context of the Internet of Things (IoT), power consumption is a critical challenge, as many IoT devices are designed to be small, low-power, and operate using batteries. Some of the power consumption challenges in IoT include:
   (i) **Battery Life:** Ensuring that IoT devices have sufficient battery life to operate without frequent recharging or replacement.
   (ii) **Energy Efficiency:** Making sure that IoT devices are designed to use energy efficiently and reduce the overall power consumption of the system.
   (iii) **Power Management:** Implementing effective power management techniques, such as sleep modes, to reduce the power consumption of IoT devices when they are not in use.

   - To address these power consumption challenges, organizations should adopt low-power technologies and energy-efficient designs for IoT devices. They

should also implement effective power management techniques, such as sleep modes, to reduce the power consumption of IoT devices when they are not in use. By prioritizing power consumption, organizations can help ensure that their IoT systems are energy efficient, reducing costs and environmental impact. Minimizing the power consumption of IoT devices to extend battery life and reduce costs.

6. **Privacy:** Privacy is a critical concern in the Internet of Things (IoT), as IoT devices collect, store, and transmit large amounts of personal and sensitive information. Some of the privacy challenges in IoT include:
   (i) **Data Collection:** Ensuring that only the necessary data is collected and that it is collected in a way that respects individuals' privacy rights.
   (ii) **Data Storage:** Ensuring that the data collected by IoT devices is stored securely and that access to it is strictly controlled.
   (iii) **Data Sharing:** Controlling who has access to the data collected by IoT devices and ensuring that it is not shared without proper authorization.
   - To address these privacy challenges, organizations should implement robust privacy policies and procedures, such as data protection, data minimization, and data retention. They should also educate users on the privacy implications of using IoT devices and encourage them to take steps to protect their privacy.
   - Additionally, organizations should adopt privacy-enhancing technologies, such as encryption and anonymization, to protect the privacy of individuals whose information is collected by IoT devices. By prioritizing privacy, organizations can help to ensure that individuals' rights and freedoms are respected, and that sensitive information is protected from unauthorized access or misuse. Protecting the privacy of individuals whose personal information is collected and transmitted by IoT devices.
7. **Battery life is a limitation:** There are issues in packaging and integration of small-sized chip with low weight and less power consumption. Although helpful, the bigger monitors are not always only for convenience, rather, instead, display screen sizes are growing to accommodate larger batteries. Computers have getting slimmer, but battery energy stays the same.
8. **Increased cost and time to market:** Embedded systems are lightly controlled by cost. The need originates to drive better approaches when designing the IoT devices in order to handle the cost modelling or cost optimally with digital electronic components.

   Designers also need to solve the design time problem and bring the embedded device at the right time to the market.
9. **Security of the System:** Systems have to be designed and implemented to be robust and reliable and have to be secure with cryptographic algorithms and

6. True or False: 'It's important that the data stored on IoT drives is encrypted'.
   (a) False (b) True
7. What technology is not used to implement confidentiality?
   (a) Encryption (b) Auditing
   (c) Access control (d) Authentication
8. Which of the following is not a type or source of threat?
   (a) Operational threat (b) Cultural threat
   (c) Technical threat (d) Social threat
9. Which of the following makes sure that data is not changed when it not supposed to be?
   (a) Integrity (b) Availability
   (c) Confidentiality (d) Accounting
10. Which one out of these is in a format which is not readable by the user?
   (a) Encryption (b) Passwords
   (c) Text (d) None of these

**Answers**

| | | | | |
|---|---|---|---|---|
| 1. (c) | 2. (b) | 3. (d) | 4. (c) | 5. (b) |
| 6. (b) | 7. (b) | 8. (b) | 9. (a) | 10. (a) |

## Practice Questions

**Q.I Answer the following questions in short.**

1. What are the activities involved in the security testing of IoT products?
2. What are some ways to prevent security breaches when using IoT applications?
3. Why is IoT security more complex than other networks?
4. Enlist the few Vulnerabilities of IoT.

**Q. II Answer the following questions.**

1. What is authentication? Why do we need it in IoT systems?
2. What is Threat modeling? Why is it necessary?
3. Explain types of threat modelling.
4. Explain the key elements of Security in IoT.
5. Enlist the challenges come to secure an IoT system.

**Q.III Define the terms.**

1. DREAD Classification System
2. Botnet
3. Ransomware
4. Lightweight Cryptography
5. IoT Security

***

# April 2022

## T.Y. B.C.A. (Science)

## BCA 365 : SEC - IV : INTERNET OF THINGS (IOT)

## (2019 Pattern) (Semester - VI)

**Time : 2 Hours** **Max. Marks : 35**

**Instructions to the candidates:**

1. Figures to the right indicate full marks.
2. Draw Diagram wherever necessary.

**Q. 1 Attempt any EIGHT of the following (out of TEN)** **[8 × 1 = 8]**

(a) Which of the following offers external chips for memory & peripheral interface circuits?
(i) Embedded System (ii) Peripheral system
(iii) Microcontroller (iv) Microprocesor

(b) In real time operating system _____.
(i) All processes have the same priority (ii) A task must serviced by its deadline
(iii) Process scheduling can done (iv) Kernel is not required

(c) _____ is not application of IoT?
(i) BMP 280 (ii) Smart home
(iii) Smart city (iv) Self driven cars

(d) _____ is IoT?
(i) Network of physical objects embedded
(ii) Network of virtual objects
(iii) Network of objects in the ring
(iv) Network of sensors

(e) "Internet of things" coined in year _____.
(i) 1998 (ii) 1999
(iii) 2000 (iv) 2002

(f) Using _____ an embedded system communicates with outside world.
(i) Memory (ii) Output
(iii) Peripherals (iv) Input

(g) _____ of the following IoT networks has a very short range.
(i) Short network (ii) LPWAN
(iii) Sigfox (iv) Short range WN

(h) _____ of the following is the way in which an IoT device is associated with data.
(i) Internet (ii) Cloud
(iii) Automata (iv) Network

(i) The protection and security for an embedded system made by _____.
(i) Security chip (ii) Memory disk
(iii) IPR (iv) OTP

(j) _____ numbers of element in the open IoT architecture?
(i) Two (ii) Three
(iii) Four (iv) Seven

**Answer Key**

| | | | | |
|---|---|---|---|---|
| (a) - (iii) | (b) - (ii) | (c) - (i) | (d) - (iv) | (e) - (ii) |
| (f) - (iii) | (g) - (iv) | (h) - (ii) | (i) - (iii) | (j) - (iv) |

**Q. 2 Attempt any FOUR of the following (out of FIVE):** [4 × 2 = 8]

(a) Enlist the characteristics of Embedded system.

**Ans.** Refer Section 1.1

(b) Explain any two pillars of IoT.

**Ans.** Refer Section 3.5.

(c) Write Need of Analog/Digital conversion.

**Ans.** Refer Section 2.4.2.

(d) What is RFID protocol?

**Ans.** Refer Section 4.4.

(e) What are the challenges for secure IoT?

**Ans.** Refer Section 6.2.2.

**Q. 3 Attempt any TWO of the following. (out of THREE)** [2 × 4 = 8]

(a) Define IoT? Write trends in Adoption of IoT.

**Ans.** Refer Sections 2.1.1 and 2.1.3.

(b) Difference between General processors in computer and Embedded processors.

**Ans.** Refer Section 1.3.

(c) M2M and WSN protocols with example.

**Ans.** Refer Sections 3.5.1 and 3.5.3.

**Q. 4 Attempt any TWO of the following. (out of THREE)** [2 × 4 = 8]

(a) Difference between Real time system and Embedded system.

**Ans.** Refer Sections 1.1 and 1.2

(b) Write a basic building Block of IoT.

**Ans.** Refer Section 2.3.

(c) Explain key elements of IoT security.

**Ans.** Refer Section 6.3.

**Q. 5 Attempt any ONE of the following. (out of TWO)** [1 × 3 = 3]

(a) Explain the ZigBee Architecture with Modbus protocol.

**Ans.** Refer Sections 4.4.1 and 4.4.2.

(b) What is RESTful web services? GRPC or SOAP explain.

**Ans.** Refer Section 5.5.

***