



Unit-1

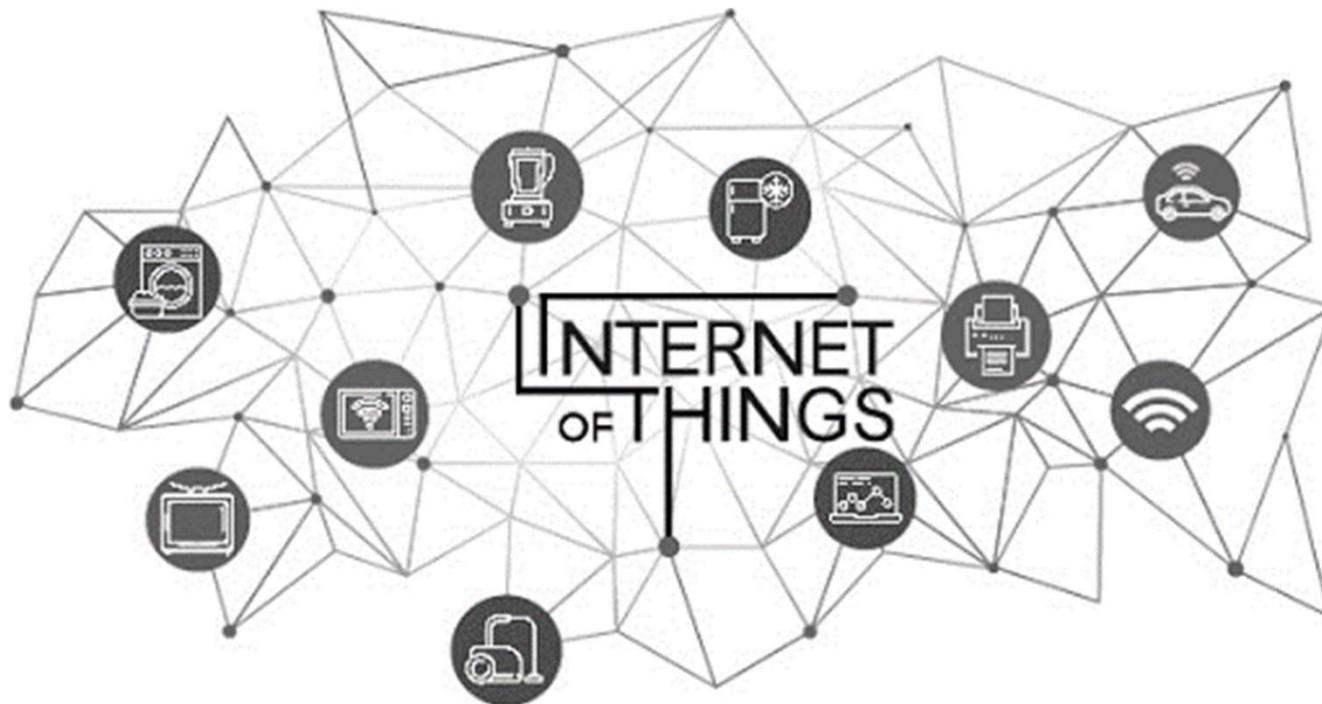
# Introduction to Internet of Things



  
**Prof. Kalpesh H Surati**  
Computer Engineering Department  
Darshan Institute of Engineering & Technology, Rajkot  
✉ Kalpesh.surati@darshan.ac.in  
📞 +91 99250 10033

# Introduction to Internet of Things (IoT)

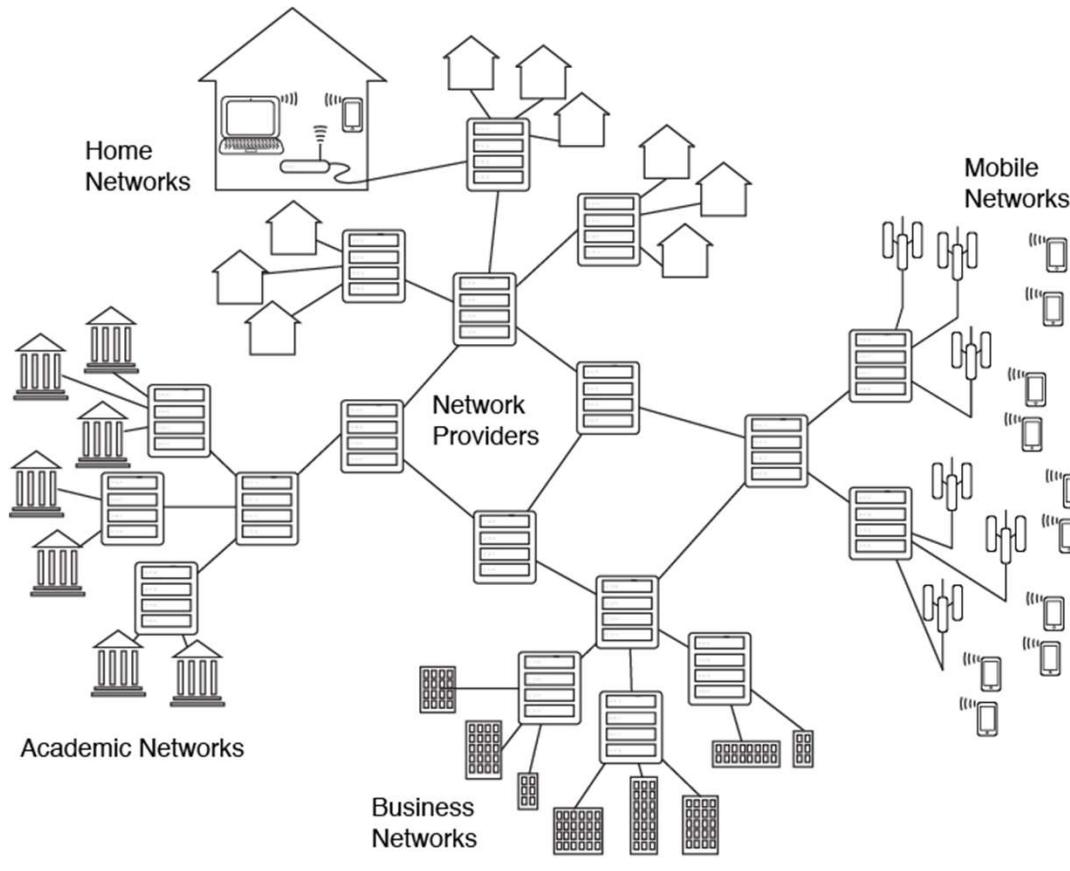
- First of all we should discuss about the name “IoT – Internet of Things” in detail.



- ▶ So we have to discuss about the first word “Internet” and then everything about the “Things”.

# Introduction to Internet of Things (IoT)

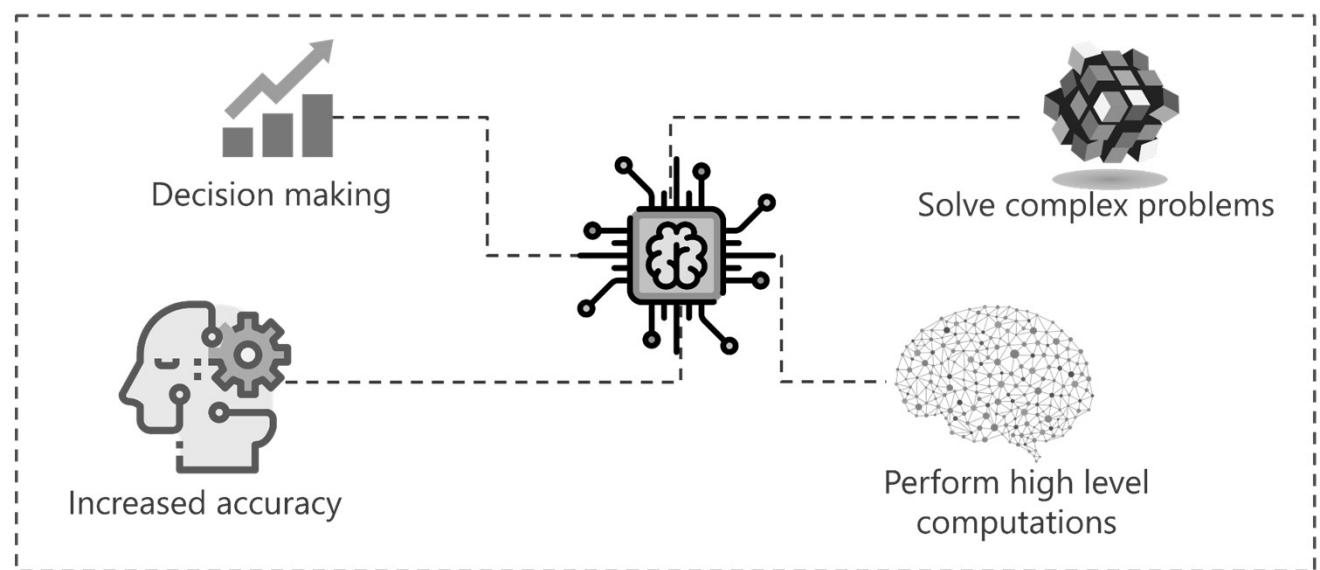
## ► What is Internet?



- In simple word, it's a Network of Networks or Interconnected LANs.
- So here we have to discuss about the Network, and we already learned everything about network in the last semester, Right?
- What is Network?, Requirement for networking

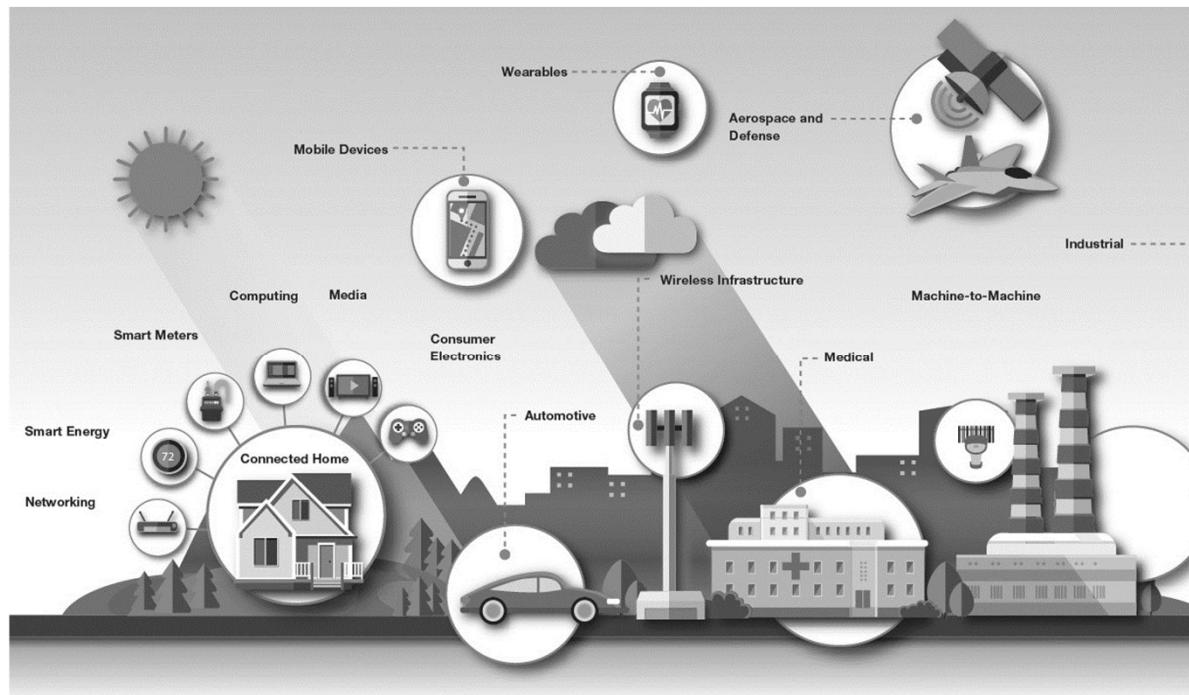
# Introduction to Internet of Things (IoT)

- ▶ Nowadays The term Internet of Things (IoT) is an emerged the popular terms.
- ▶ There are multiple ways to define IoT, but the basic of all the definitions remains the same.
- ▶ IoT is network of interconnected computing devices which are embedded in everyday objects, enabling them to send and receive data.
- ▶ The IoT is not just limited to the connected or networked devices, but in a broad way IoT devices exchange meaningful information from one device to another to get desire result.



# Introduction to Internet of Things (IoT)

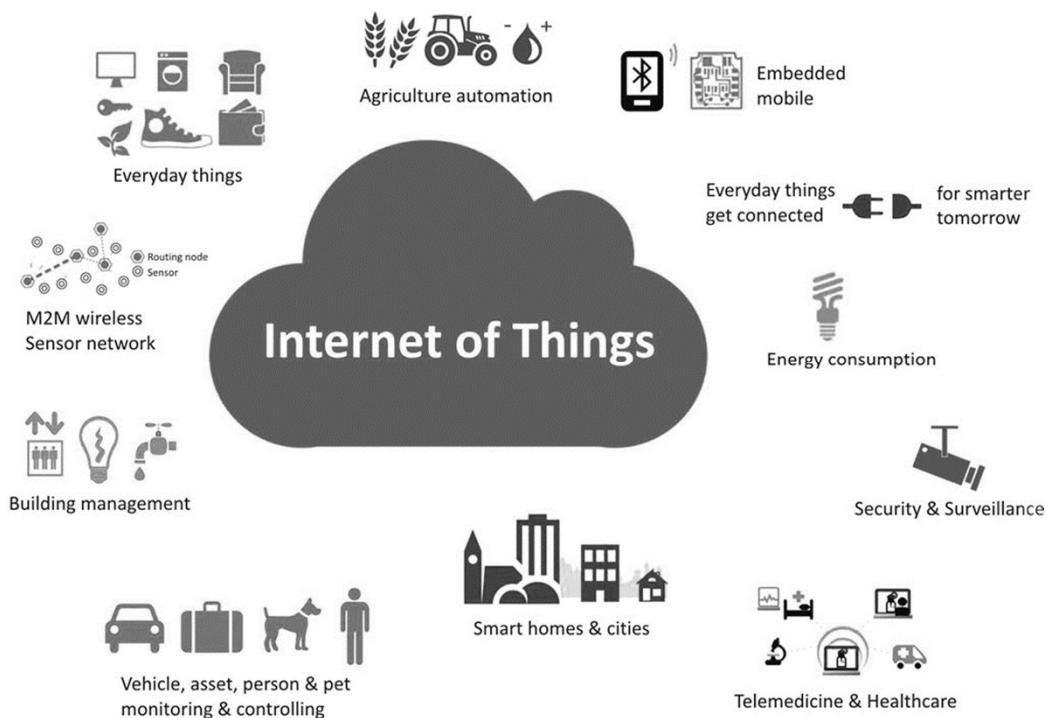
- ▶ IoT is not a single technology, it's a combination of technologies and domain knowledge.
- ▶ As a result, engineers from different domains have to work together for building a complete IoT product.



- ▶ Life would be governed entirely by Internet and IoT in the near future.

# Application areas of IoT

- ▶ The scope and application areas of IoT is very huge.
- ▶ IoT can be used to build applications for...

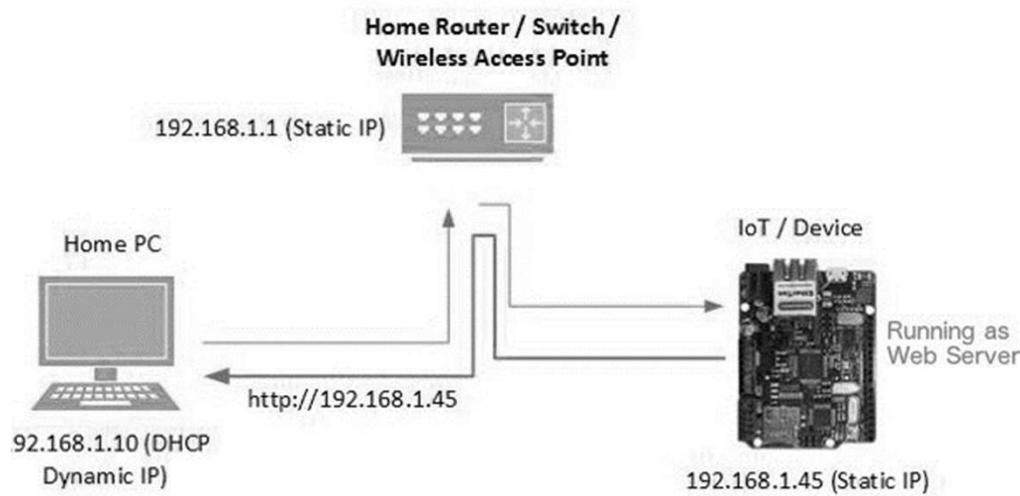


- Agriculture
- Assets Tracking
- Energy Sector
- Defense
- Embedded Applications
- Education
- Waste Management
- Healthcare Products
- Telemedicine
- Safety And Security Sector
- Smart City Applications etc.

# Characteristics of IoT

## ▶ Connectivity:

- Connectivity is an important and first requirement of IoT infrastructure.
- Every Things in IoT should be connected to the IoT infrastructure.
- Connectivity should be guaranteed at anywhere and anytime.



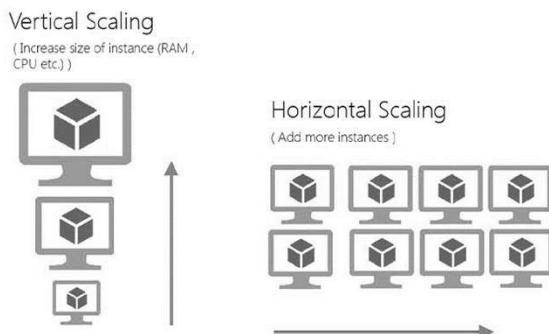
## ▶ Identity:

- Each IoT device has a unique identity (e.g., an IP address).
- This identity is helpful in communication, tracking and to know status of the things.

# Characteristics of IoT

## ▶ Intelligence:

- Just data collection is not enough in IoT, extraction of knowledge from the generated data is very important.
- For example, sensors generate data, but that data will only be useful if it is interpreted properly.
- So intelligence is one of the key characteristics in IoT.



## ▶ Scalability:

- The number of elements (devices) connected to IoT zone is increasing day by day.
- Therefore, an IoT setup should be capable of handling the expansion.
- It can be either expand capability in terms of processing power, Storage, etc as vertical scaling or horizontal scaling by multiplying with easy cloning

# Characteristics of IoT

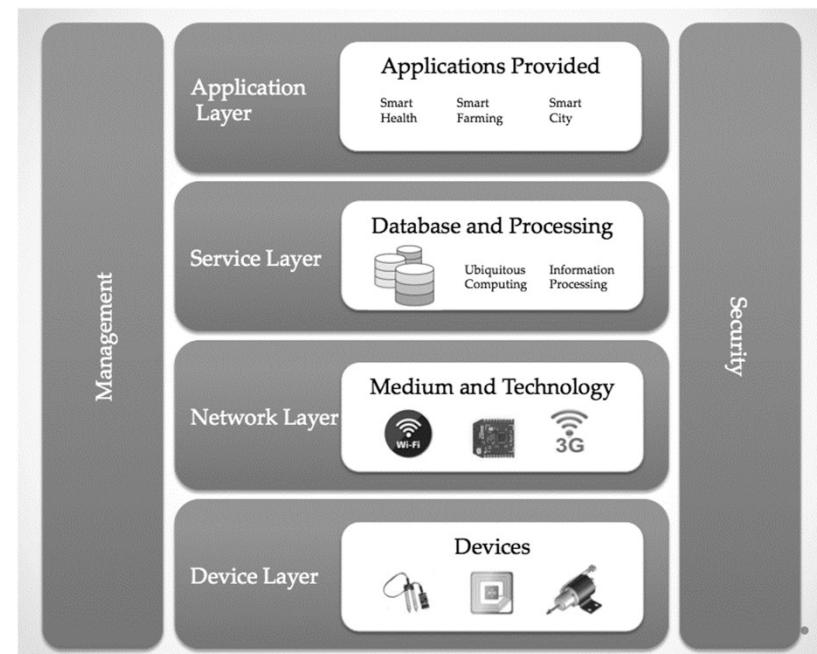


## ► Dynamic and self-adapting (complexity):

- IoT devices should dynamically adapt themselves to the changing surroundings.
- For example surveillance camera. It should be flexible to work in different weather conditions and different light situations (morning, afternoon, or night).

## ► Architecture:

- IoT architecture is yet not uniformed and standardized.
- It should be hybrid, supporting different manufacturer's products to function in the IoT network.



# Characteristics of IoT

## ▶ Safety:

- Sensitive personal details of a user might be compromised when the devices are connected to the Internet.
- So data security is a major challenge.
- This could cause a loss to the user.
- Equipment in the huge IoT network may also be at risk.
- Therefore, equipment safety is also critical.

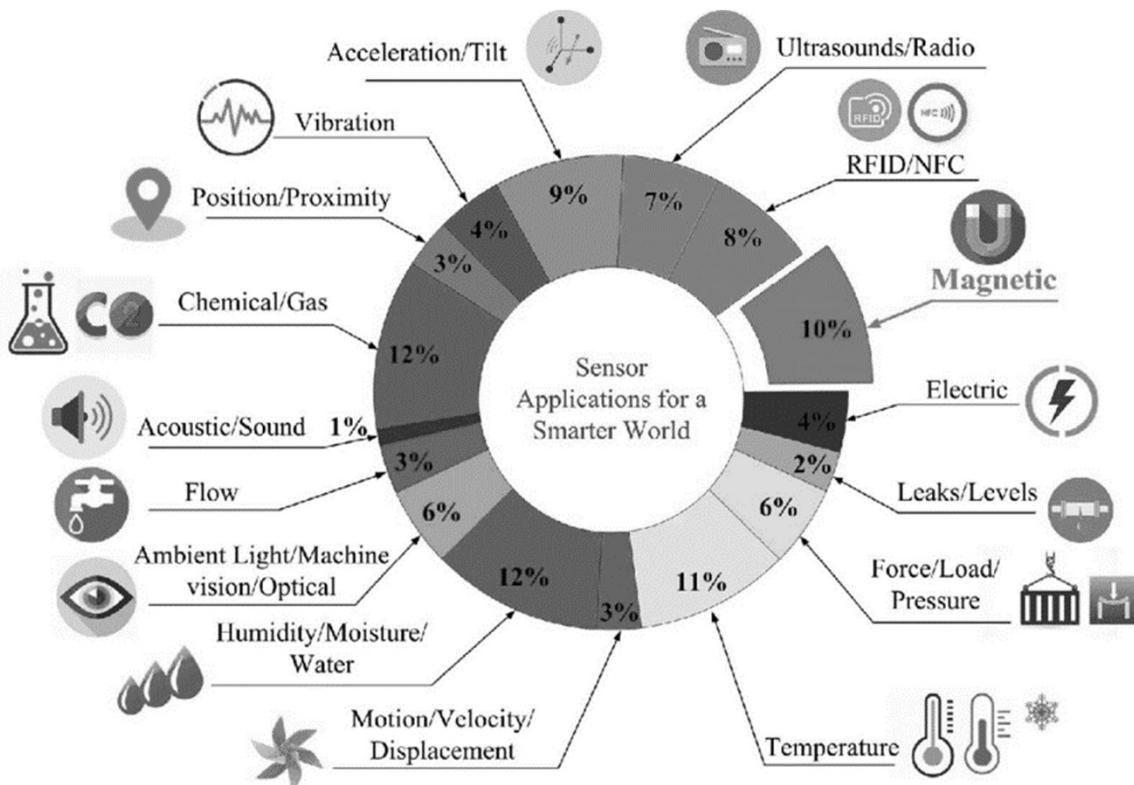


# Things in IoT

- ▶ In the IoT, things refer to a variety of devices. It can be anything even humans in it become a thing.
- ▶ For something to qualify as a “thing”, it requires identity of its existence.
- ▶ The “thing” in a network can be monitored/measure. For example, a temperature sensor could be a thing.
- ▶ Things are capable of exchanging data with other connected devices in the system.
- ▶ The data could be stored in a centralized server (or cloud), processed there and a control action could be initiated.
- ▶ The devices having all the above characteristics are known as things.



# Things in IoT



- ▶ Some of the famous “things” are temperature sensors, pressure sensors, humidity sensors, etc.
- ▶ The data from these sensors are collected and sent it to the cloud or stored it in local server for data analysis.
- ▶ Based on the data analysis, the control action would be taken.
- ▶ For example, switching off the water heater remotely when the water is heated as per requirement.

# Things in IoT

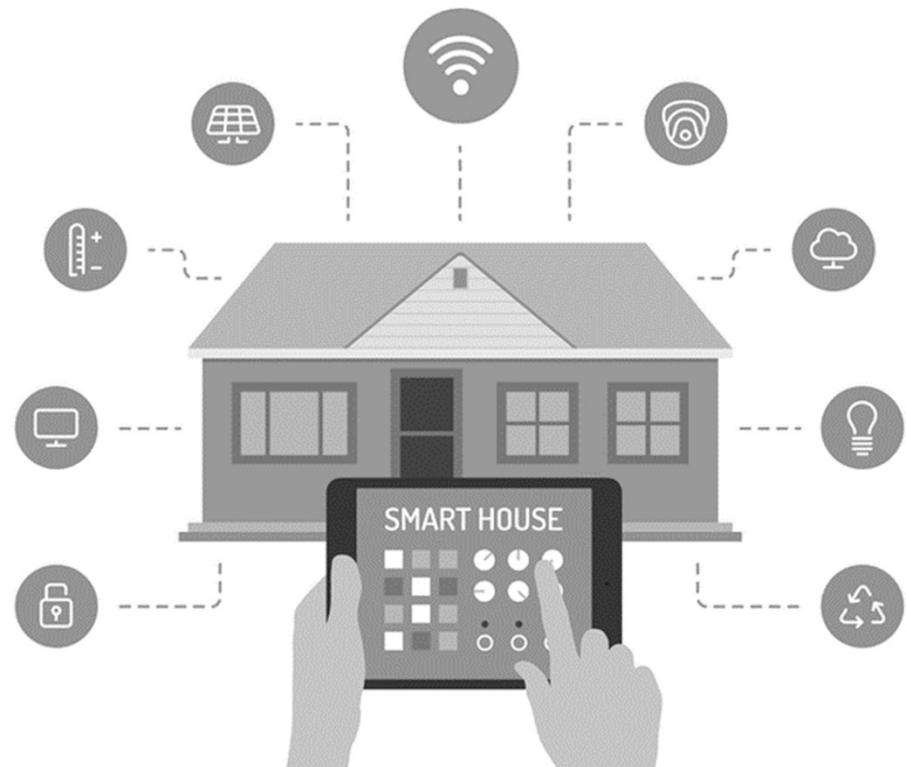
► Not just sensors, the following can also be called as things:



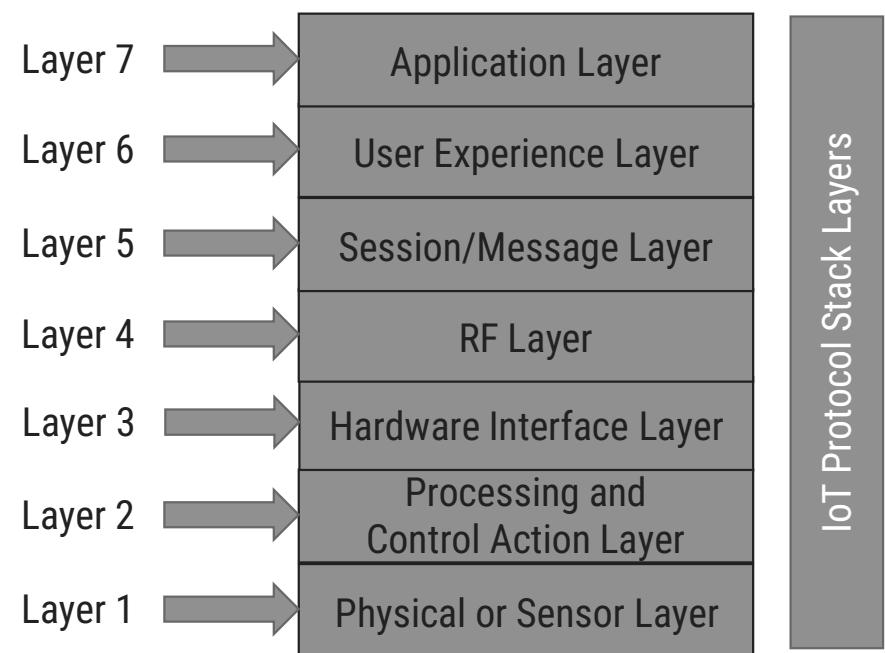
- Industrial motors
- Wearables (e.g., watch)
- Vehicles
- Shoes
- Heart monitoring implants (e.g., pacemaker, ECG real-time tracking)
- Biochip transponders (for animals in farms)
- Automobiles with built-in sensors (automobile feature real-time monitoring)
- Food/perishables quality measuring

# Things in IoT

- ▶ In IoT-based home automation, the “things” could be the following
  - Lighting control and automation devices
  - Ventilation devices
  - Air conditioning [heating, ventilation and air conditioning (HVAC)] systems
  - Appliances such as washer/dryer
  - Air purifiers
  - Ovens or refrigerators/freezers that use Wi-Fi for remote monitoring
  - Security cameras
  - Smart phones



- ▶ Like other digital technology IoT has stack layers.
- ▶ Following are the identified seven layers in IoT stack.
  - Layer 1 (Physical or Sensor Layer)
  - Layer 2 (Processing and Control Action layer)
  - Layer 3 (Hardware Interface Layer)
  - Layer 4 (RF Layer)
  - Layer 5 (Session/Message Layer)
  - Layer 6 (User Experience Layer)
  - Layer 7 (Application Layer)



## Slide 15

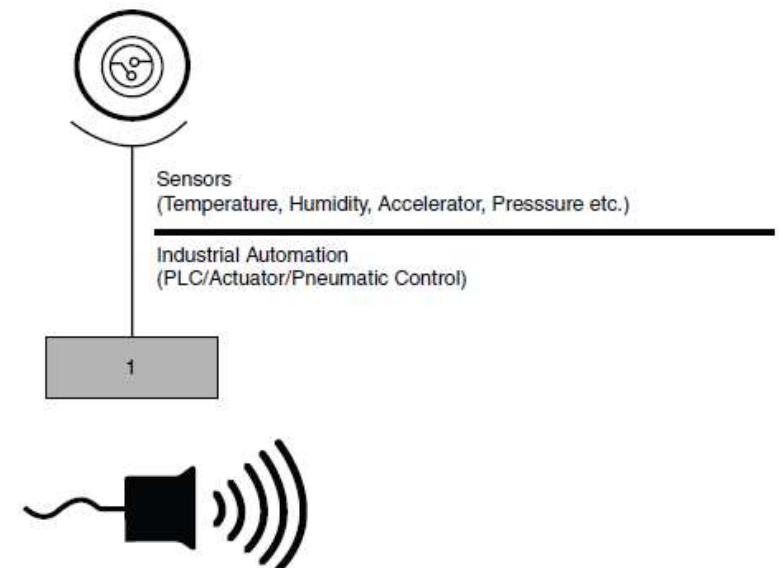
---

**A2** <https://www.rfwireless-world.com/IoT/IoT-Protocol-Stack-layers.html>  
Admin, 01-02-2021

**WU1** Theory for the stack  
Windows User, 08-02-2021

# IoT Stack - Layer 1 (Physical or Sensor Layer)

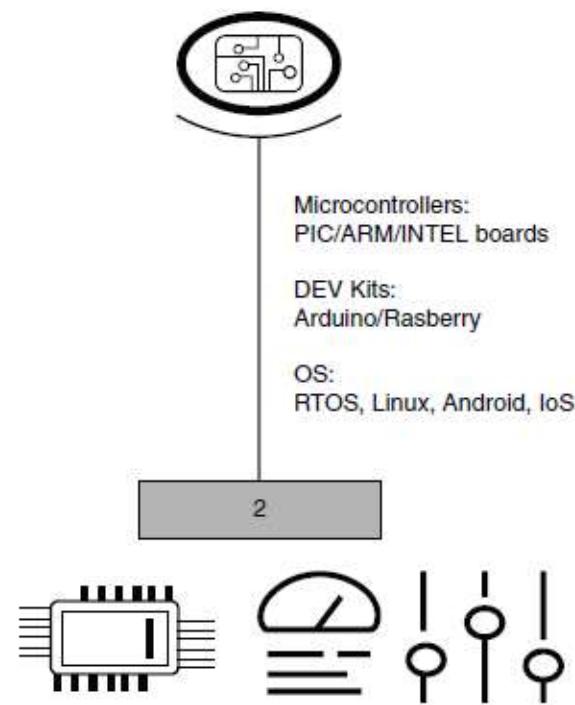
- ▶ This layer is concerned about the physical components, which mainly includes sensors.
- ▶ In this layer, the sensors are the core component.
- ▶ Temperature sensor, pressure sensor, humidity sensor, etc. can all be referred as physical layer components.
- ▶ In industrial automation, PLC, actuator, etc. are considered as physical layer components.
- ▶ This layer is responsible for data collection and action execution.
- ▶ Selection of sensors is important and choosing an appropriate sensor is the challenge in this layer.
- ▶ Action execution, sensing and data collection happens here.



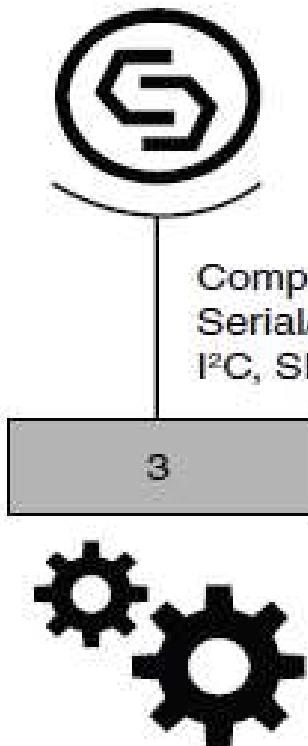
Layer 1: Sensor layer (physical layer); data collection happens here.

# IoT Stack - Layer 2 (Processing and Control Action Layer)

- ▶ This important layer contains core components of IoT system.
- ▶ The microcontrollers or processors are found in this layer.
- ▶ The data is received by the microcontrollers from the sensors.
- ▶ A variety of development kits are available in the market; like Arduino, Raspberry Pi, Node MCU, PIC, ARM development boards, etc.
- ▶ Microcontroller/Processor and operating system play vital role at this layer
- ▶ Data collected from the sensors is processed in this layer.

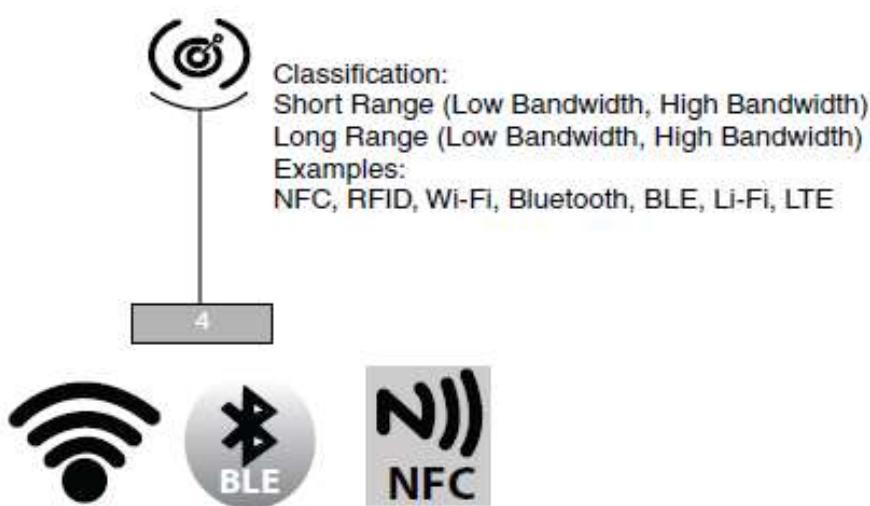


# IoT Stack - Layer 3 (Hardware Interface Layer)



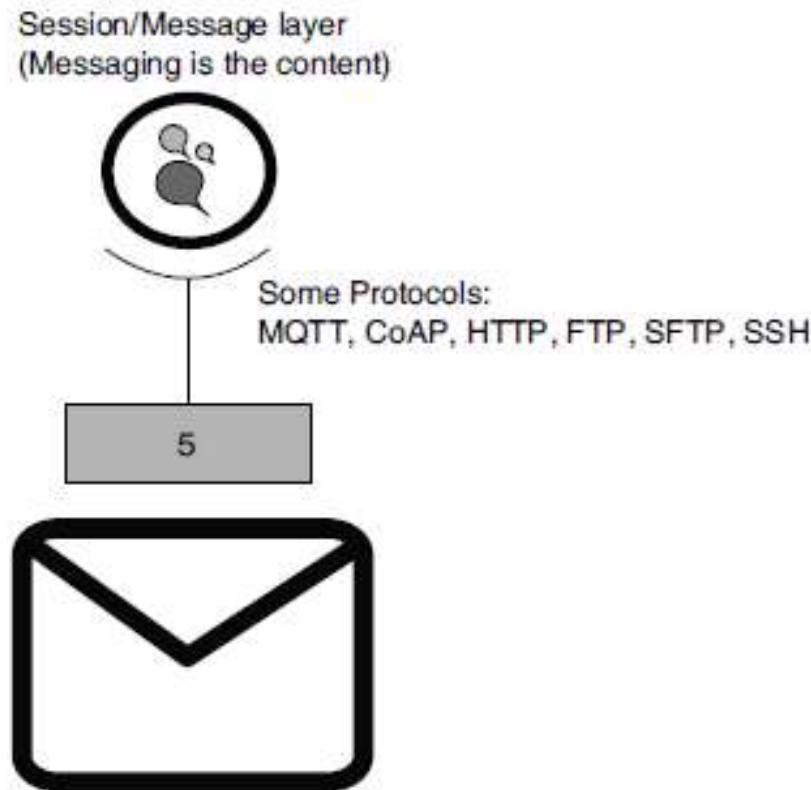
- ▶ The 3<sup>rd</sup> layer in the stack is the Hardware Interface Layer.
- ▶ Hardware components and communication standards such as RS232, CAN, SPI, SCI, I<sup>2</sup>C, etc. occupy this layer.
- ▶ All these components ensure flawless communication
- ▶ Handshake happens here.

# IoT Stack - Layer 4 (RF Layer)



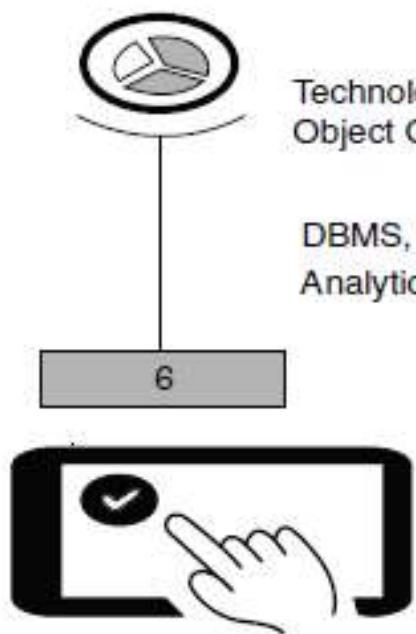
- ▶ Whenever one talks about IoT, RF is discussed and comes in picture.
- ▶ It plays a major role in the communication channel – whether it is short range or long range.
- ▶ Protocols used for communication and transport of data based on RF are listed in this layer.
- ▶ Some famous and common protocols are Wi-Fi, NFC, RFID, Bluetooth, Zigbee, etc.
- ▶ RF layer does communication of data using radio frequency based Electromagnetic (EM) waves
- ▶ This layer can also include Li-Fi; which are effective alternates for RF protocols.

# IoT Stack - Layer 5 (Session/Message Layer)



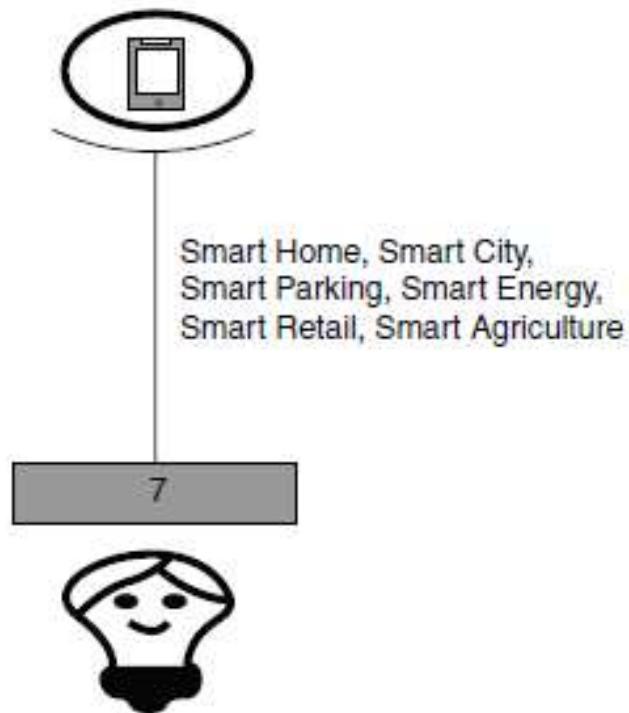
- ▶ Like computer network session management is also important in IoT.
- ▶ There are many protocols which manage how messages or data are broadcasted to the cloud.
- ▶ Layer 5 (session layer) deals with the various messaging protocols as MQTT, CoAP, etc. and also other protocols such as SSH and FTP.

# IoT Stack - Layer 6 (User Experience Layer)



- ▶ This layer deals with providing best experience to the end users of IoT products.
- ▶ The 6th layer takes care of rich UI designs with lots of features, which provide a pleasing experience while using the service/system or product.
- ▶ Object-oriented programming languages, scripting languages, analytics tools, etc. all should be included in this layer.
- ▶ This is also known as User Experience and Visualization Layer.

# IoT Stack - Layer 7 (Application Layer)



- ▶ Everything comes to perfection at this layer.
- ▶ This layer utilizes the rest six layers in order to develop desired application.
- ▶ It can range from a simple automation application to smart city application.
- ▶ After learning about the layers, it is now easier to relate them with an application, for example, vegetable quality monitoring during transport from source to the destination using IoT.

# Enabling Technologies

- ▶ IoT is a collection or group of many technologies and devices.
- ▶ The simplest of sensors, embedded systems, data analytics, communication protocols, security aspects and cloud computing with storage have all become enabling technologies.
- ▶ Enabling technologies/devices fall under one of the following categories:
  - Technologies that help in acquiring/sensing data.
  - Technologies that help in analyzing/processing data.
  - Technologies that help in taking control action.
  - Technologies that help in enhancing security/privacy.
- ▶ Let's discuss some of the enabling technologies.

## IoT Enabling Technologies

- **Wireless Sensor Network**



- **Cloud Computing**



- **Big Data Analytics**



- **Communication Protocols**



- **Embedded Systems**



# Enabling Technologies - Sensors



**Temperature Sensor**  
Precision  $\pm 0.3^\circ\text{C}$   
Range -40 to 85°C  
-40 to 185°F



**Humidity Sensor**  
Precision  $\pm 3\%$ RH  
Range 0 to 100%



**Ambient Light Sensor**  
Precision  $\pm 3\%$   
Range 0.01 to 83K lux



**Vibration Index Sensor**  
Precision 4mg  
Range -16 to 16g



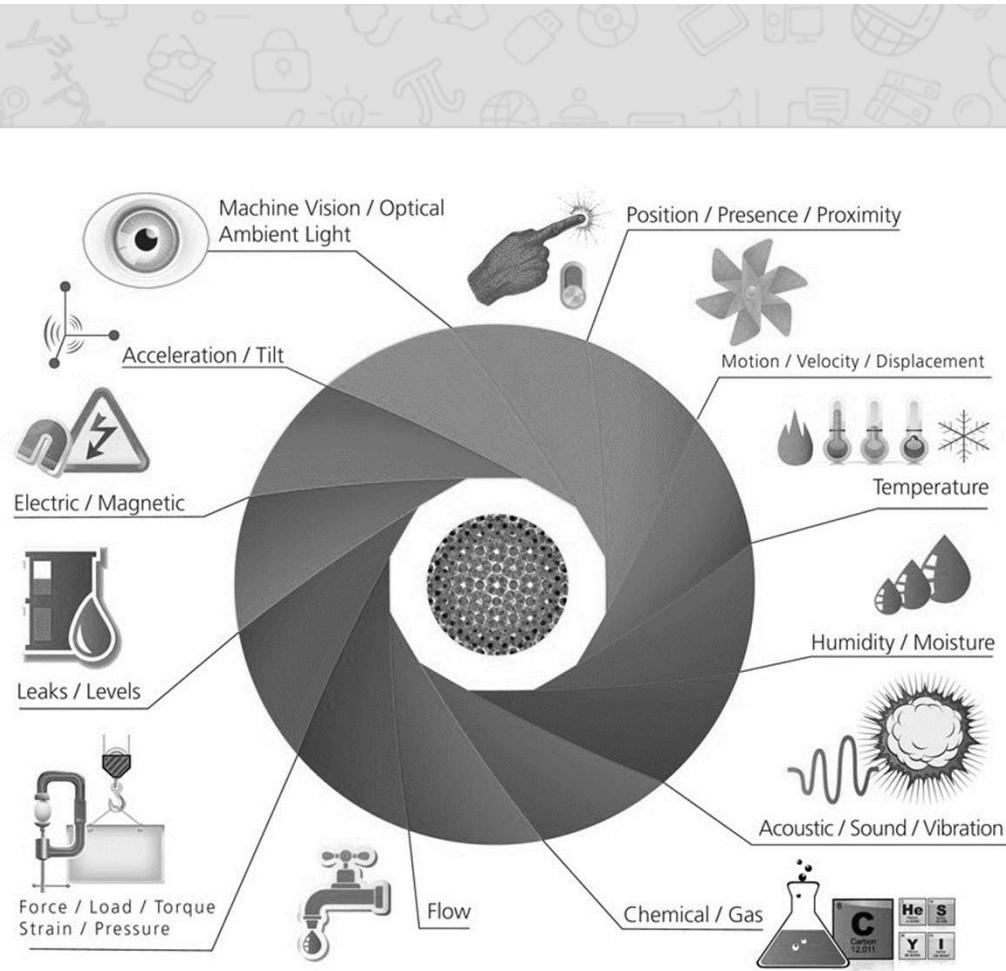
**External Temperature Sensor**  
Precision  $\pm 0.5^\circ\text{C}$   
Range -55 to 125°C  
-67 to 257°F

- ▶ Sensors are at the heart of any IoT application.
- ▶ As the name suggests, they sense the environment and retrieve data.
- ▶ Sensors are the starting point of any IoT application.
- ▶ It fetches data for us to operate on.
- ▶ Sensors could be analog or digital.
- ▶ Temperature sensor in a thermometer is an example of it. It is used to build temperature monitoring application.

# Enabling Technologies - Sensors

► Some examples of sensors that could be regarded as enabling technologies are as follows.

- Weather tracking system uses temperature/humidity/moisture sensors.
- Vehicle health monitoring sensors keep track of speed, tyre pressure, etc.
- On Board Diagnostics (OBDs) used for collecting all critical information from an automobile to detect error.
- Vibration sensors are used to track the quality of buildings/structures.
- Water quality is monitored through sensors that measure PH, chloride level, etc.
- PIR sensor is used in pedestrian signal operation with human presence detection.



# Enabling Technologies - Cloud Computing

- ▶ The next technology that is highly significant in IoT is cloud computing.
- ▶ Cloud has grown much more popular because it serves as an affordable, effective and efficient medium for data storage.
- ▶ Data storage plays a major role in IoT.
- ▶ Cloud services are categorized as follows:

- IaaS (Infrastructure-as-a-Service):

- In this cloud service, one can choose virtual machines over physical machines.
    - It is a form of cloud computing that provides virtualized computing resources over the Internet.
    - The users manage the machines, select the OS and underlying applications, and pay per their use.



# Enabling Technologies - Cloud Computing

## → PaaS (Platform-as-a-Service):

- This is a cloud computing model in which the cloud service provider delivers hardware and software tools needed for application development to users over the Internet.
- A PaaS provider hosts the hardware and software on its own infrastructure. Users have to build, manage and maintain the applications as per their requirement.



## → SaaS (Software-as-a-Service):

- In this model, a complete software application is provided to the user.
- It can also be called application as a service. This service can be availed by paying a monthly, yearly, etc., subscription.
- Some well-known service providers in the market are Amazon web services, Azure and Adafruit.

# Enabling Technologies - Big Data Analytics

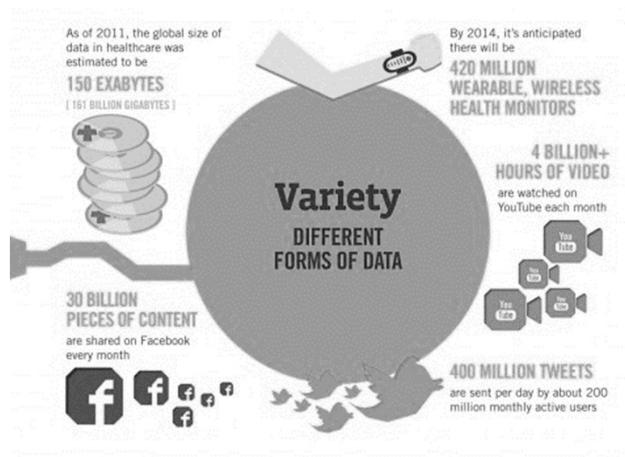
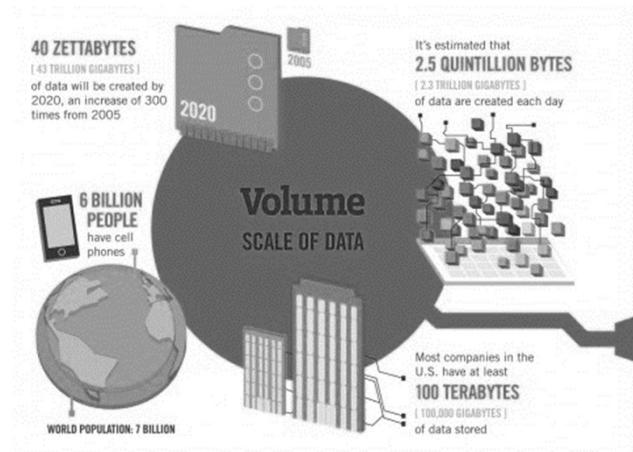


- ▶ Data is everywhere, and from every function or operation we get more data.
- ▶ IoT is all about collecting data from various sensory nodes.
- ▶ Handling the huge data is fundamental to make the application a success.
- ▶ The biggest challenge with big data is 4Vs. Volume, Variety, Speed (Velocity) at which it comes and its Veracity.

# Enabling Technologies - Big Data Analytics

## ► Scale (Volume):

- Huge volume of data is generated every minute.
- Storage has become inexpensive and hence, cost-related challenges have reduced.
- Cloud storage and hardware storage both have become affordable because of the tremendous growth in the semiconductor industry.



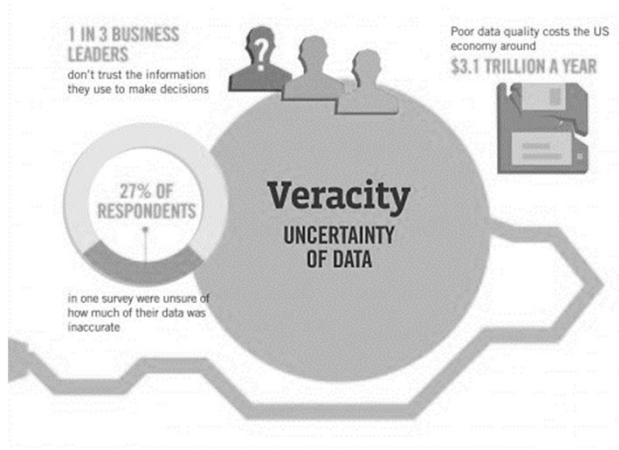
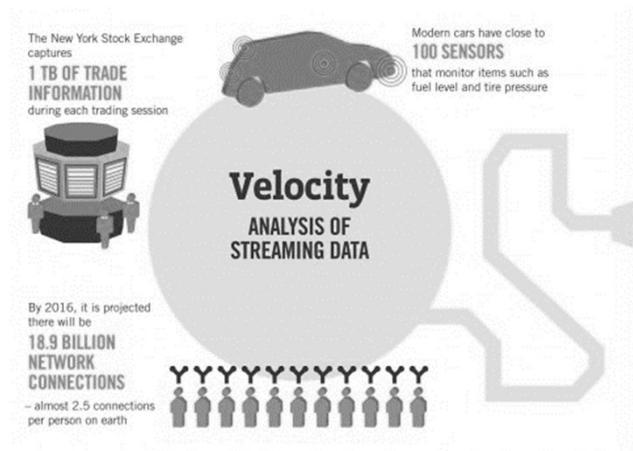
## ► Complexity (Variety):

- Data no longer comes from one single source.
- It also comes in different formats (e.g., audio, video, text and image) and has to be interpreted systematically.
- Varieties of data becomes a huge challenge.

# Enabling Technologies - Big Data Analytics

## ▶ Speed (Velocity):

- The rate at which data is generated very fast.
- Also, data dynamics changes very frequently.
- Nowadays, data comes from anywhere – from fit bit watches to refrigerators.
- All the data pours in at a very high speed, which makes it very challenging.



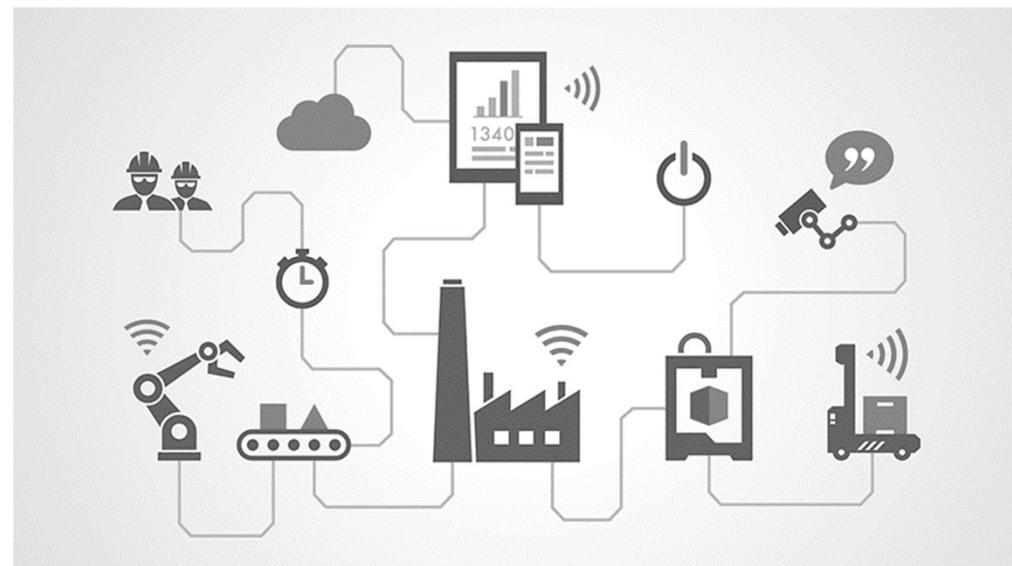
## ▶ Data in doubt (Veracity):

- How accurate is all this data anyway?
- Because we are now rely on it
- The data's nature alters dynamically and uncertainty is often seen.
- So, it would be challenging to process this unstable data.

# Enabling Technologies - Big Data Analytics

► So the question is: "Who is generating all this data?" A partial list to answer this is as follows:

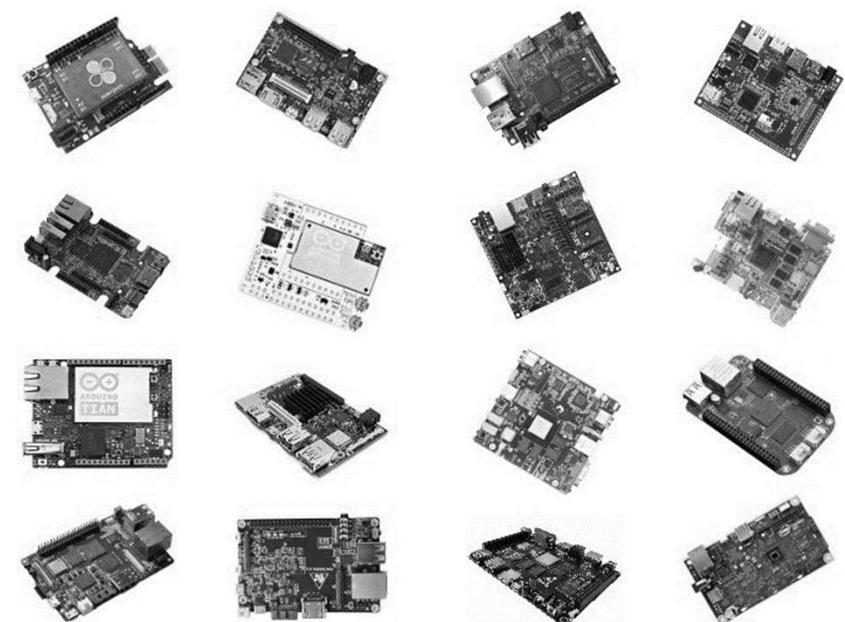
- Sensors from security systems.
- Sensors from weather monitoring systems.
- Sensors from car/navigation systems.
- Sensors from water quality monitoring systems.
- Data from wearables (e.g., bands).
- Data from industrial equipment (e.g., motor health).
- Sensors from bridges/roads about traffic density and other factors.
- Social media (e.g., tweets, photo uploads, etc.).



► In IoT data is everything. so, data analytics is one of the enabling technologies for building a complete and IoT application.

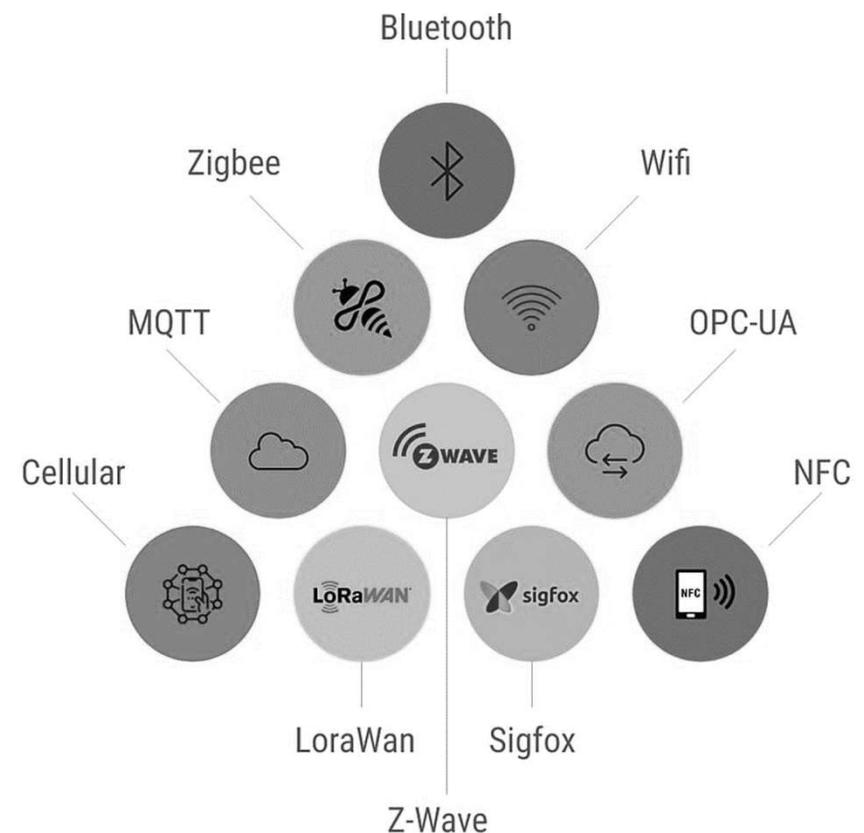
# Enabling Technologies - Embedded Computing Boards

- ▶ An embedded computing board a very important component to bring IoT design to reality.
- ▶ For making the prototype the computing boards play vital role.
- ▶ The computing boards available in the market are driven by microcontrollers or processors.
  - Some of the boards are as follows:
    - Raspberry Pi.
    - Arduino (many variants).
    - NodeMCU.
    - Intel Edison.
- ▶ All these boards are small, yet smart.
- ▶ Also, the cost involved is very minimal and one can get these boards at cheap rate.



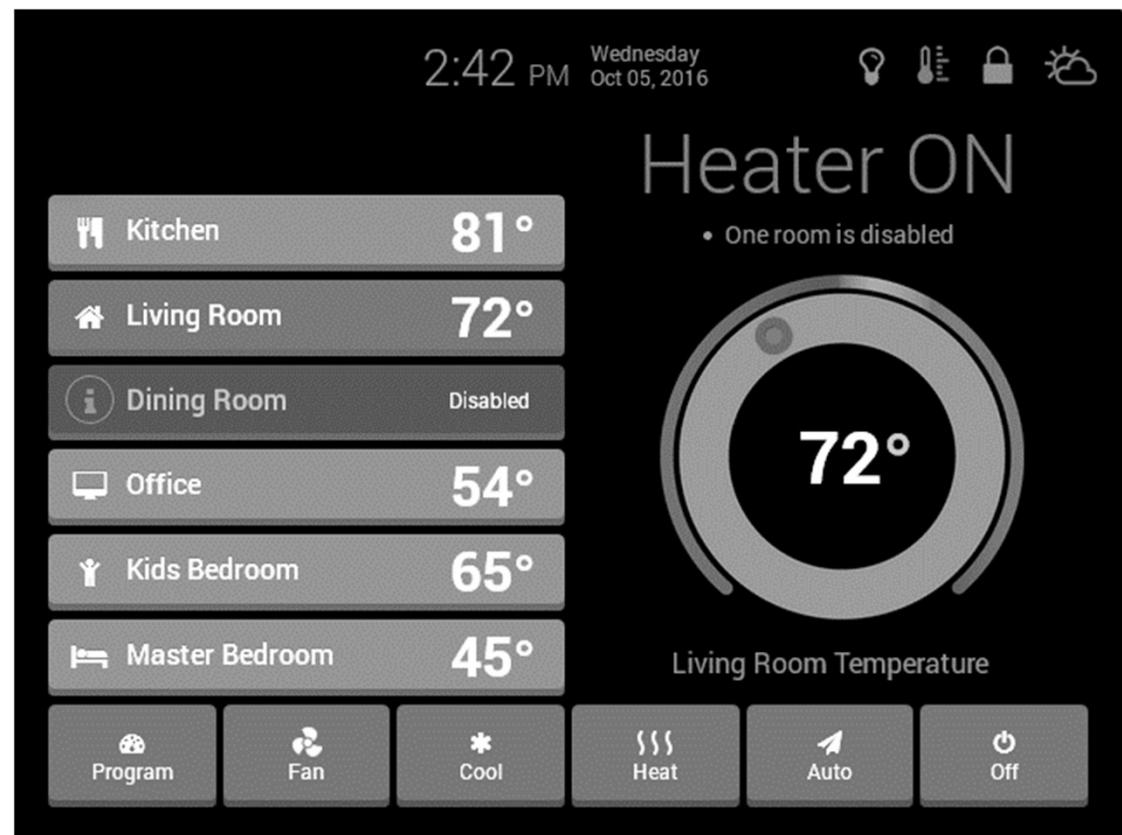
# Enabling Technologies - Communication Protocols

- ▶ Protocols are the pillars for good IoT infrastructure and hence are very important in communication.
- ▶ Data exchange happens through these protocols, which take care of the following:
  - Addressing.
  - Format of the messages.
  - Message security (encryption and decryption).
  - Routing.
  - Flow control.
  - Error monitoring.
  - Sequencing.
  - Retransmission guidelines.
  - Segmentation of the data packets.



# Enabling Technologies - User Interfaces

- ▶ All devices should have an intuitive user interface.
- ▶ IoT devices/services should be designed in such a way that accessing and handling the services are easier and comfortable for the end user.
- ▶ Generally , the end user shall be provided “mobile application or web application”.
- ▶ The application should be stable and elegant.



# IoT Challenges

- ▶ The following are some of the challenges - technical and non-technical during building an IoT application .

## 1. Security/Personnel safety:

- Security is one of the most significant challenges.
- Number of IoT devices are gradually increasing, so user data becomes more vulnerable to theft.
- Poor security features can let attackers damage the whole network and the rest of the devices could also become vulnerable.
- People's personal safety is also a concern and challenge.
- The implants and wearable used by people should be safe even from physical harm.



# IoT Challenges

## 2. Privacy:

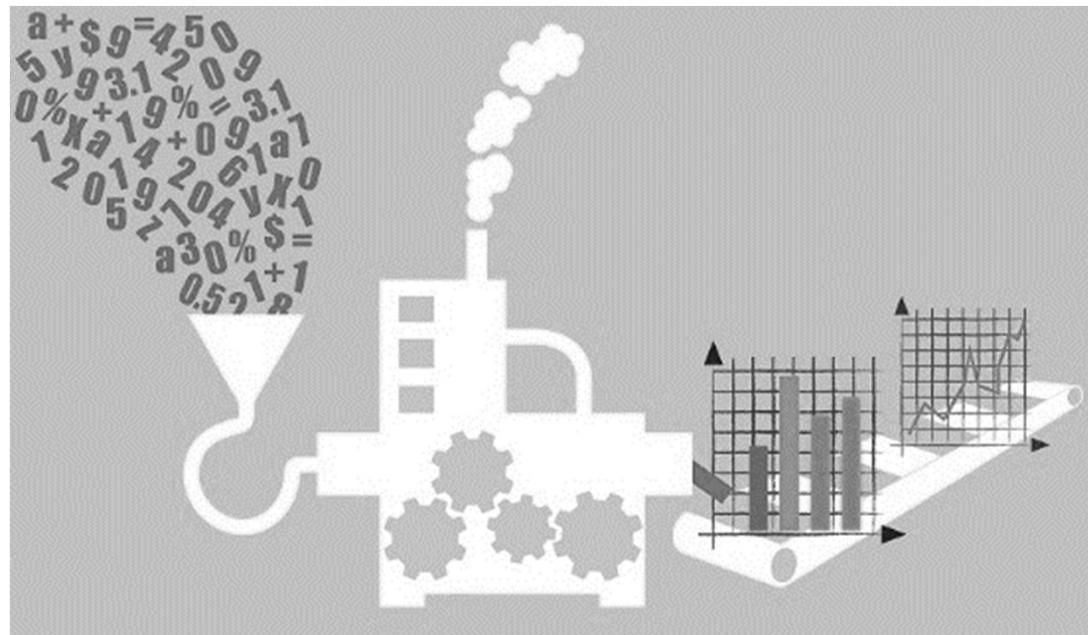
- One could be tracked/monitored by anyone, as we are connected  $24 \times 7$  to the Internet.
- So, there is a threat on user data and raises a question on user privacy.
- "How do we ensure that the data that is sensed and collected from the user is with their permission?"



# IoT Challenges

### 3. Data extraction with consistency from complex environments:

- It is a huge challenge to sense and extract data from complex environments.
- For Example, variation in temperature could also damage the products being transported.
- Maintained the temperature is very critical and should be accurately monitored.
- If the temperature is about to change, then the corrective action has to be taken.
- Data extraction and storage in the cloud could be more challenging if the internet is not available.
- Extracting data inside a room is different from extracting data from an open environment.



# IoT Challenges

## 4. Connectivity:

- This is a serious challenge that the IoT world must acknowledge.
- Since the Internet is itself a giant collection of networks and devices and IoT is a part of it. So requirement of wired and wireless connectivity is a necessity.
- The usage of frequency / spectrum is also to be noted.
- There are spectrum regulations to be followed based on the country for which the application is being developed.
- 2.4 GHz band is the ideal band everywhere.



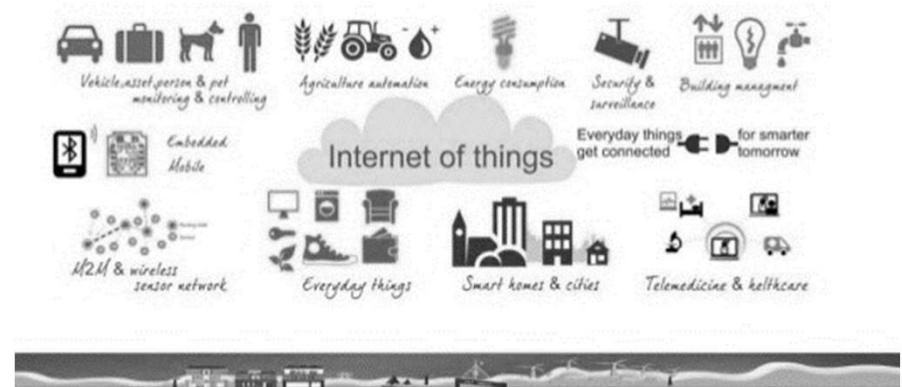
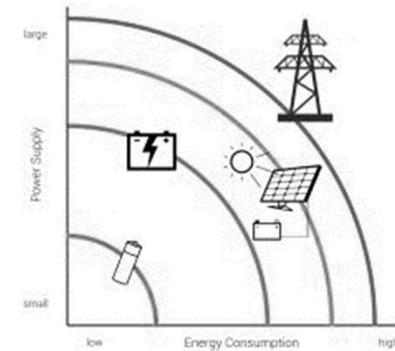
# IoT Challenges

## 5. Power requirements:

- All the IoT devices require power and most of them are battery operated.
- Even though we now have long-lasting batteries that are economical, demand for power is on the rise.
- Usage of green power sources such as solar and wind should be motivated.
- If the power requirements are met appropriately, IoT can be even more powerful.

## 6. Complexity involved:

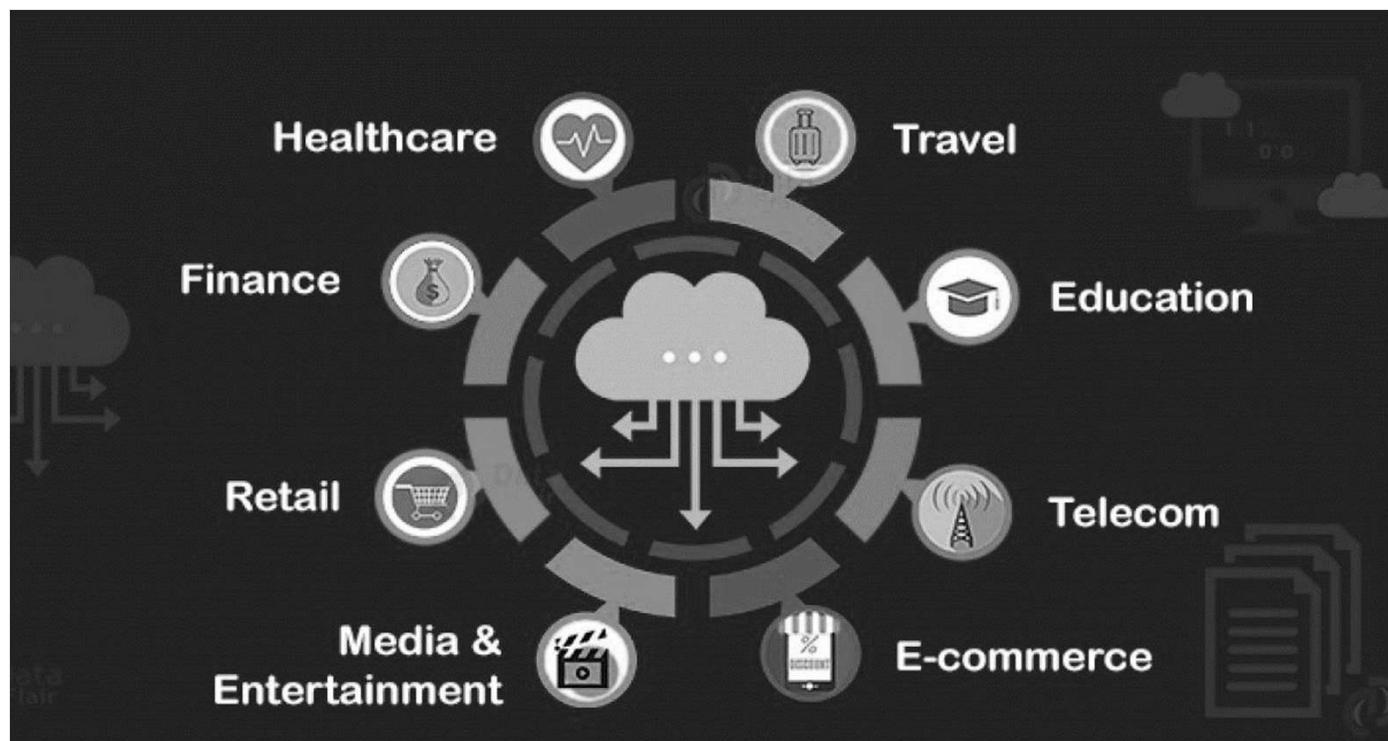
- IoT is not easy. It needs a lot of different domains to integrate into a cohesive system.
- There is very limited expertise available in the market, but the growth is very rapid.
- The toolkits, software and hardware are not abundant and real skill is required to build an application.



# IoT Challenges

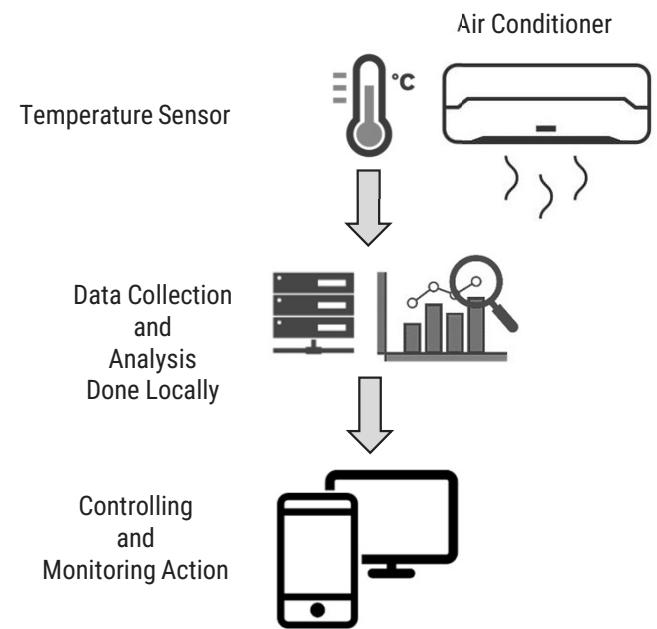
## 7. Storage:

- Cloud is becoming mandatory for the data to be stored and analyzed.
- The challenge with respect to this aspect is connected to the following points:
  - Which cloud do we use (private, public, or hybrid)?
  - How do we identify the service provider?
  - How much does it cost?
  - Do we really need cloud?



# IoT Levels

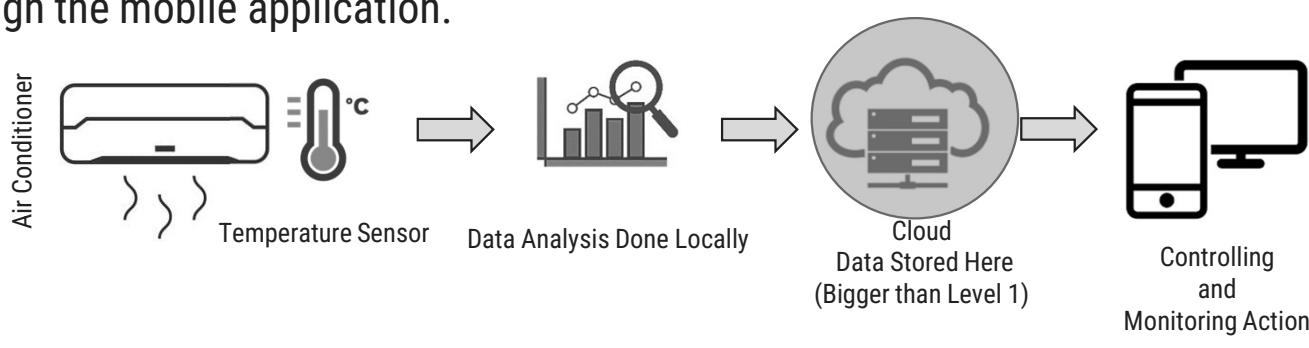
- ▶ Based on the architectural approach, IoT can be classified in five levels: Level 1 to Level 5.
- ▶ Level 1
  - It is of minimal complexity.
  - The application has one sensor (temperature sensor, pressure sensor, etc).
  - The data sensed is stored locally and the data analysis is done locally.
  - Monitoring / control is done through an application.
  - This is used for simple applications.
  - Data generated in this level application is not huge.
  - For example, a temperature sensor senses the room temperature and the data is stored and analyzed locally.
  - Based on the analysis, the control action can be triggered through mobile application or it can help in monitoring the status.



# IoT Levels

## ▶ Level 2

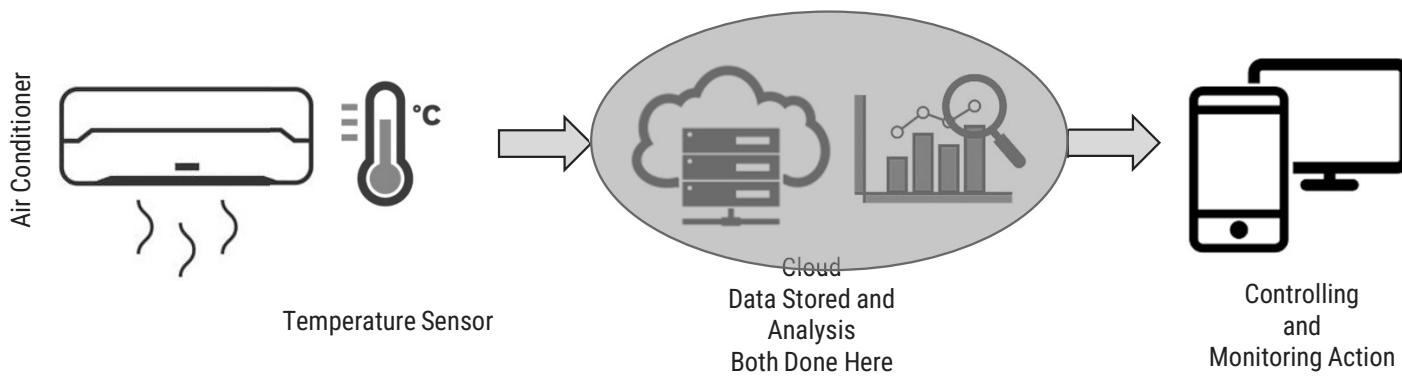
- The second level is slightly more complex than the previous level.
- The data is more voluminous and hence, cloud storage is preferred.
- The frequency of sensing done by the sensor is faster.
- The number of times sensing is done would be much more than Level 1.
- The analysis is carried out locally, while cloud is meant for storage only.
- Based on the data analysis, the control action can be triggered through the web application or mobile application.
- Some examples are agriculture applications, room freshening solutions based on odor, etc.
- IoT application of an air conditioner. The sensor reads the room temperature at a better pace and rate than Level 1; the data then goes on to the cloud for storage. Analysis is done locally and the action is triggered through the mobile application.



# IoT Levels

## ▶ Level 3

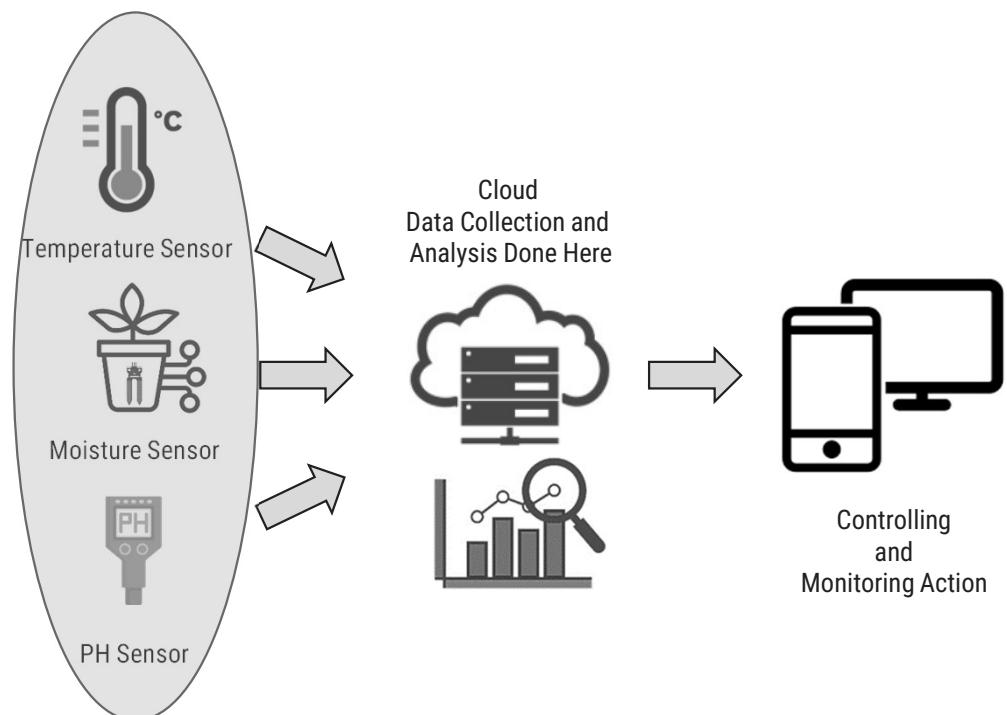
- The data is huge, frequency of sensing done by the sensor is faster and the data is stored on cloud.
- The difference is that the analysis is also carried out on cloud.
- Based on the data analysis, the control action can be triggered through the web application or mobile application.
- Some examples are agriculture applications, room freshening solutions based on odor, etc., where analysis of data occurs in the cloud.



# IoT Levels

## ▶ Level 4

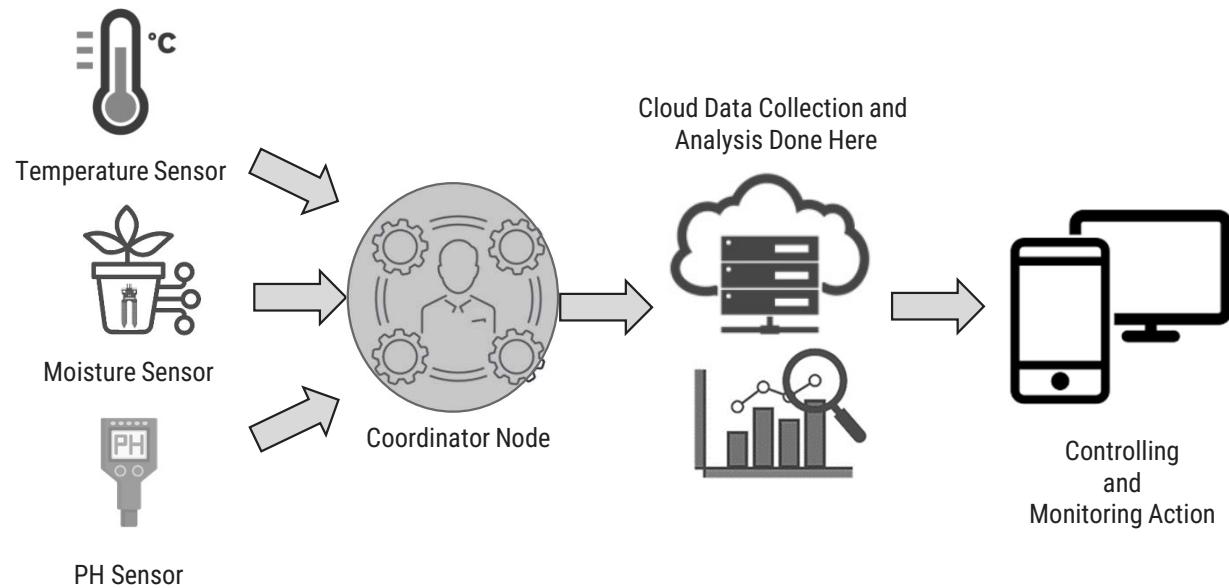
- With every passing level, the volume of data increases and hence the rate at which it is sensed also increases.
- At this level, multiple nodes are present which are independent of each other.
- These nodes upload data to the cloud.
- All the sensors upload the read sensory inputs on cloud storage.
- Analysis is also carried out on the cloud.
- Based on the analysis carried out, the control action shall be triggered through a web application or mobile application.



# IoT Levels

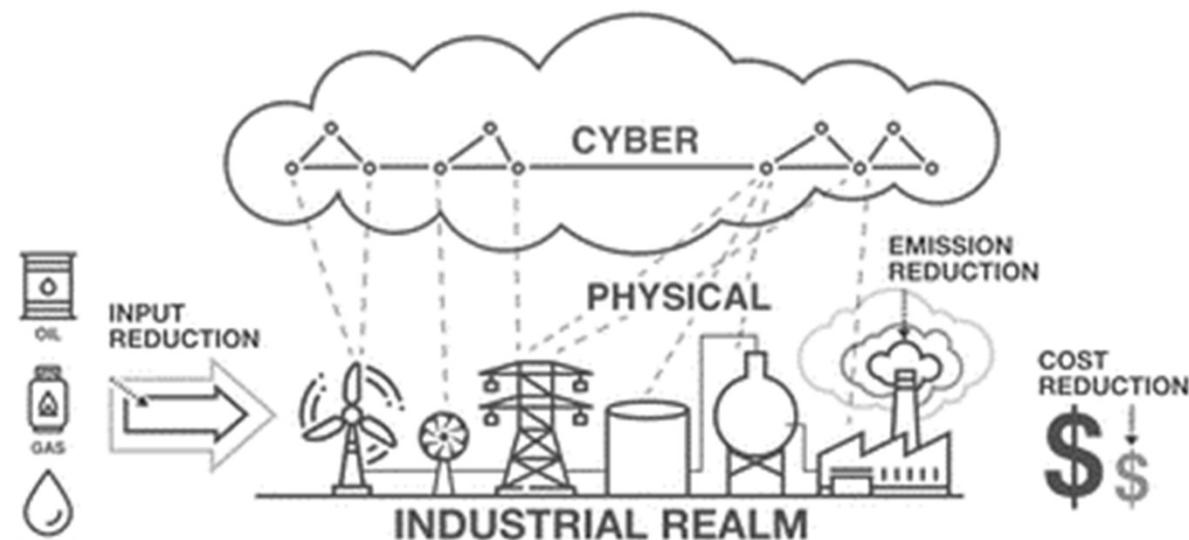
## ▶ Level 5

- At this level, the amount of data is extensive and is sensed much faster.
- Multiple nodes are involved in the applications categorized under Level 5 and these nodes are independent of each other.
- The sensing of data and its storage is the same as in all the previous levels.
- When an application is completely cloud oriented, it is computationally intensive in real time.
- Based on the data analysis, the control action can be triggered through web application or mobile application as in all other levels.



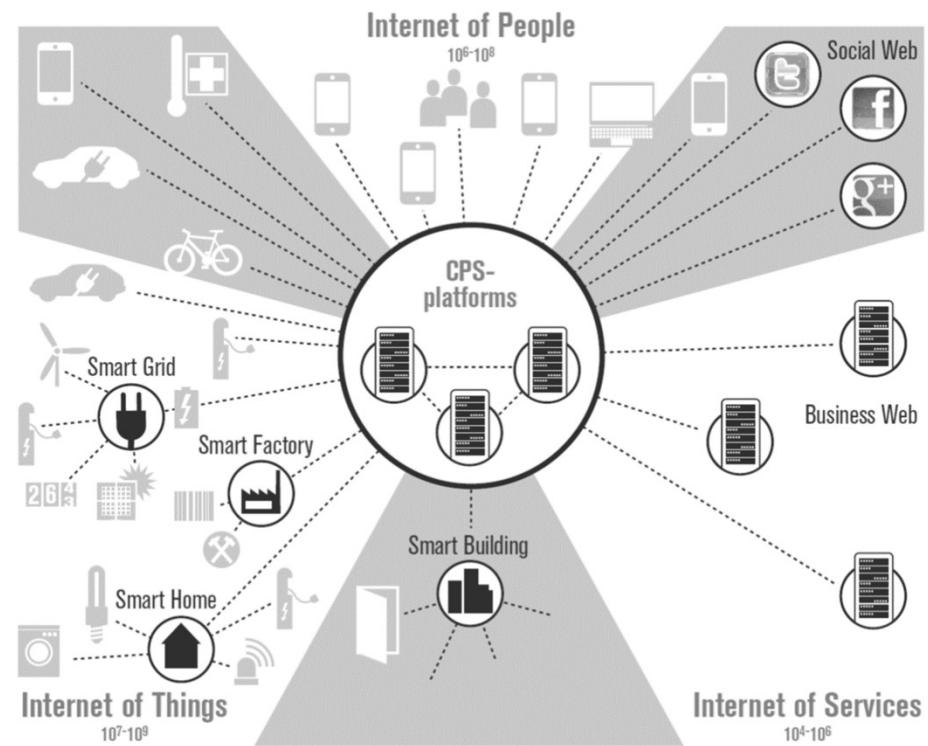
# Cyber Physical System versus IoT

- ▶ An important question is, Is IoT same as Cyber Physical System (CPS)?
  - There is a misconception that both the terms are the same.
  - We have learned the definition of IoT. The “thing” can also be accessed from anywhere, anytime by an authorized party.
  - The information or the sensed data of the things can be simple.
  - So complexity involved in the IoT applications is minimal.
  - For complex levels of operation and to address larger network of “things”, a new term called Cyber Physical System or CPS, has been introduced.



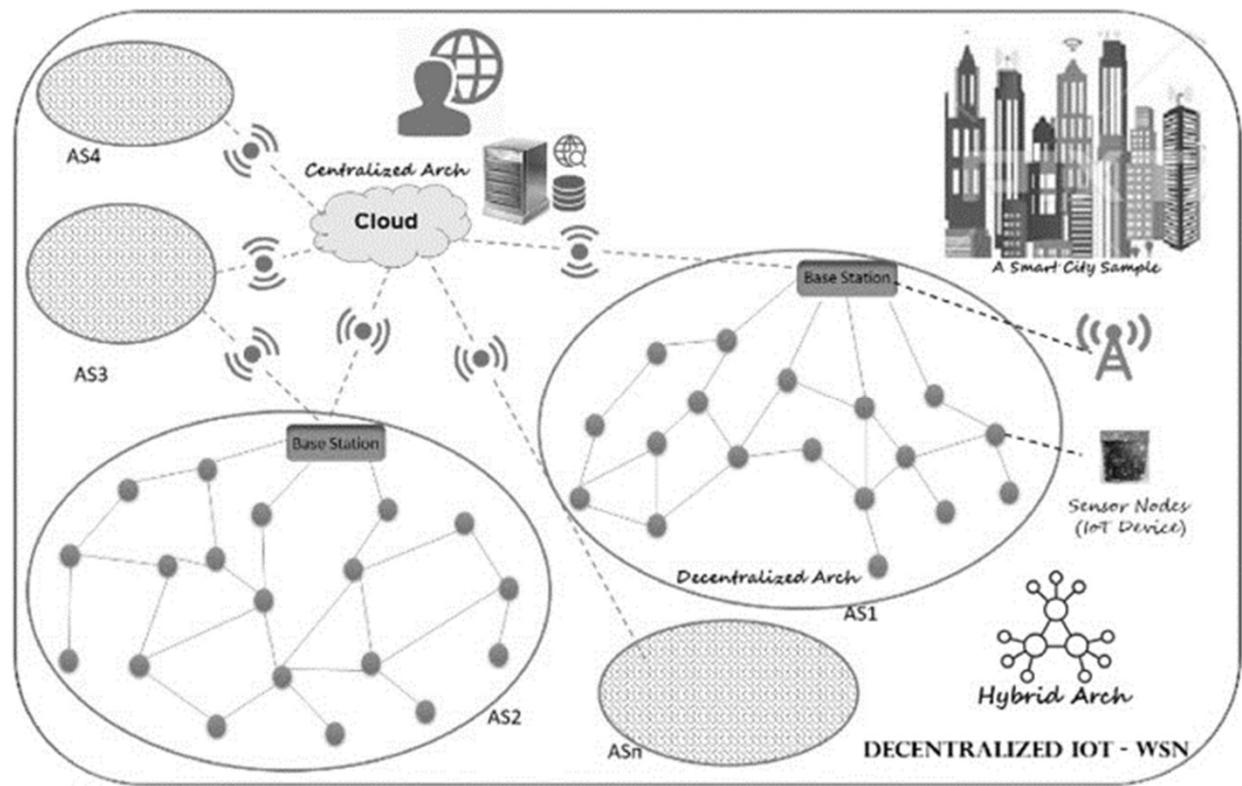
# Cyber Physical System versus IoT

- It is important to note that CPS is not IoT.
- CPS is more complex than IoT and is much more challenging.
- CPS has IoT as one of its components.
- It is a combination of multiple engineering domains coming together.
- The flight of an aero plane can be seen as a CPS which involves multiple domains of engineering.
- CPS is much more autonomous than IoT, taking appropriate decisions as and when needed.
- It is not just about identifying “things”; it is more about understanding and taking decisions in a more dynamic way.
- CPS is mainly concerned about the collaborative activity of sensors or actuators to achieve a certain goal.
- For that CPS uses an IoT system to achieve the collaborative work of the distributed systems.



# Wireless Sensor Network versus IoT

- ▶ WSN is a network of multiple autonomous sensors/nodes.
- ▶ Each node has one or more sensors.
- ▶ All the sensed data are passed to a centrally located server.
- ▶ The data passing happens in a coordinated pattern.
- ▶ We can say that WSN is all about coordinated data collection.
- ▶ On the other hand, IoT is much more than just data collection and the systems are more intelligent.



# Summary

- ▶ IoT refers to the interconnection of computing devices embedded in everyday objects via the Internet, enabling them to send and receive data.
- ▶ IoT is not owned by any one engineering branch. It is a reality when multiple domains join forces and combine efforts.
- ▶ IoT is all about providing service to any device, anywhere, anybody, and any network.
- ▶ IoT has certain characteristics which are important: a. Connectivity. b. Intelligence and identity. c. Scalability. d. Dynamic and self-adapting (complexity). e. Architecture. f. Safety.
- ▶ “Things” refer to variety of devices. At times, even humans in the loop becomes a thing. For anything to qualify as a “thing”, it requires identity. The “thing” can monitor, measure, etc.; for example, a temperature sensor could be a “thing”.
- ▶ One should understand that “THINGS” = HARDWARE + SOFTWARE + DATA + SERVICE
- ▶ IoT stack has seven layers, starting with sensor layer and ending with application layer just as OSI.

# Summary

- ▶ Security/personnel safety, privacy, data extraction with consistency from complex environments, connectivity, power requirements, complexity involved and storage are the major challenges we face while building an IoT application.
- ▶ IoT application can be classified as Level 1, 2, 3, 4 or 5 based on the complexity and architecture involved.
- ▶ IoT is all about sense, connect, store, analyze, control and sharing.

Unit-2

# Sensors, Microcontrollers and their Interfacing



**Prof. Tushar J. Mehta**  
Computer Engineering Department  
Darshan Institute of Engineering & Technology, Rajkot  
✉ tushar.mehta@darshan.ac.in  
📞 +91-8866756776

# Microcontrollers

Section - 1

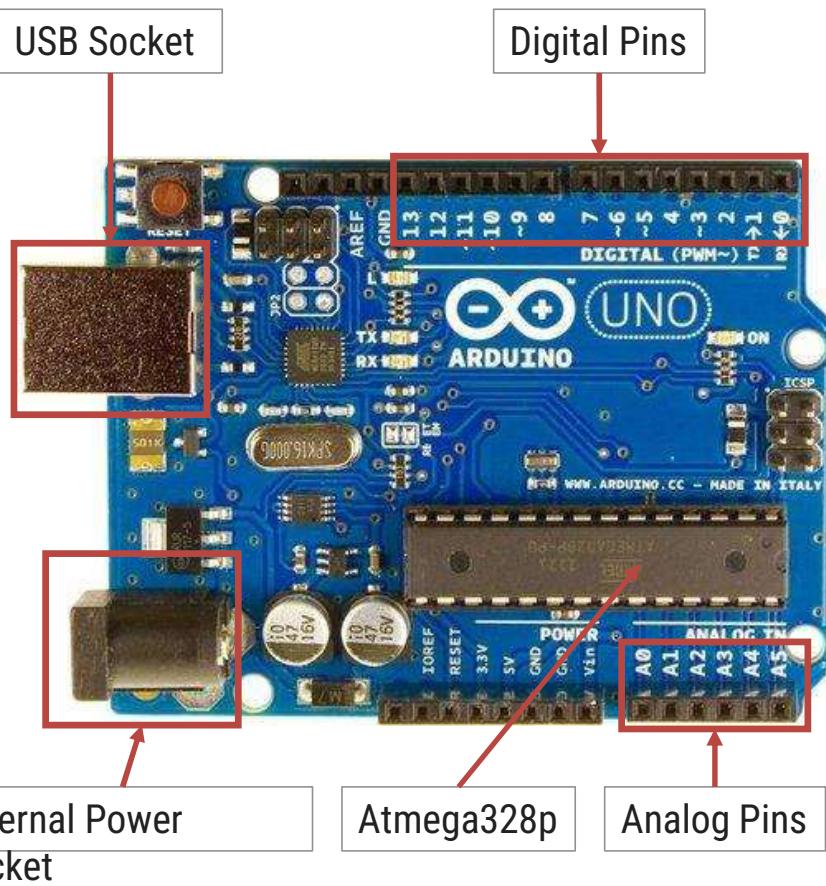
# Overview of Microcontrollers

- ▶ **Microcontroller** is a digital device consisting of **CPU, peripheral I/Os, memories, interrupts, etc.** in a single chip or IC.
- ▶ **Microcontroller** is a **compact integrated circuit** designed for **a specific operation** in an embedded system.
- ▶ The examples of various microcontrollers are given as:



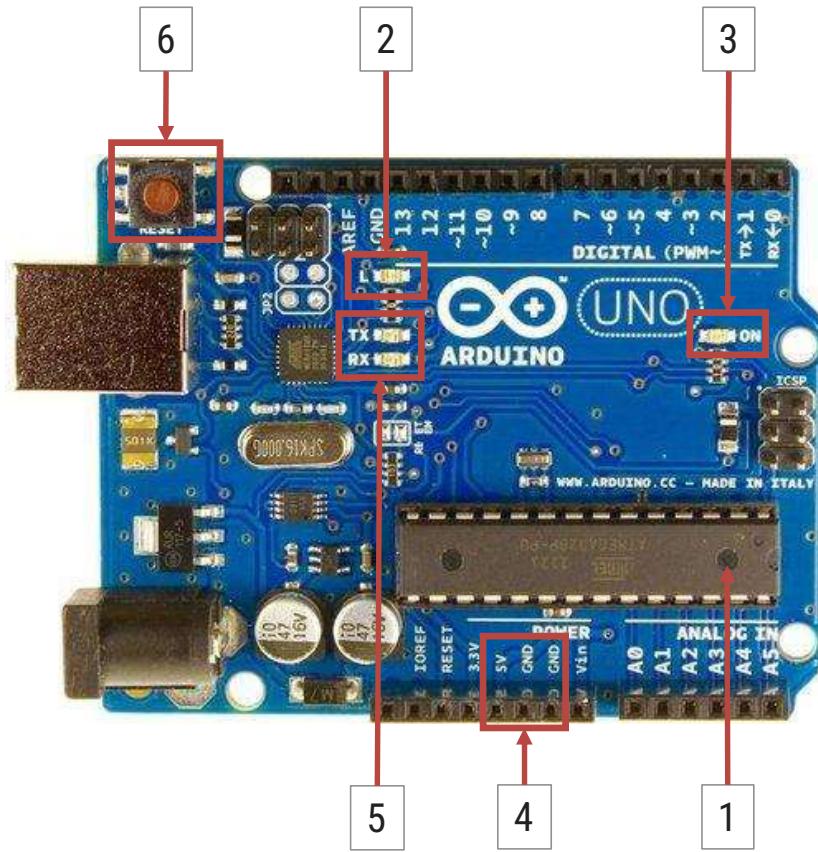
Atmel AVR	Microchip – PIC	Intel 8051
AtMega8	PIC18F452	AT89C51
AtMega328P	PIC16F877	AT89S52
AtMega16	PIC16F676	AT89c2051
AtMega2560	PIC16F72	P89V51





## Introduction to Arduino Uno

- Atmega328p Microcontroller inside
- The operating voltage is 5V which is given by
  - USB socket or
  - External power socket
- DC Current for each input/output pin is 40 mA
- Digital input/output pins are 14 to interface digital input or output devices
- There are 6 Analog i/p pins used for interfacing analog input devices.
- What is [Digital and Analog?](#)
- [Memory Components:](#)
  - 32KB flash (program) memory
  - SRAM is 2 KB
  - EEPROM is 1 KB



## Components in Arduino Uno

1. Atmega328p Microcontroller: The heart of the board.
2. On-board LED interfaced with pin 13.
3. Power LED: Indicates the status of Arduino whether it is powered or not.
4. GND and 5V pins: Used for providing +5V and ground to other components of circuits.
5. TX and RX LEDs: These LEDs indicate communication between Arduino and computer.
6. Reset button: Use to reset the ATmega328P microcontroller

# Sensors and Their Interfacing

Section - 2

# Definition of Sensor

- ▶ **Sensor** is a device that detects or measures a physical property and records, indicates, or otherwise responds to it.
- ▶ A **sensor** is a device that measures **physical input** from its environment and converts it into data that can be **interpreted by either a human or a machine**.
- ▶ Normally, the sensors are input devices and there are two types of sensors.
  - **Analog Sensors**
  - **Digital Sensors**



# List of Sensors

- The list of sensors which is to be covered in the chapter is as follows:

MQ-02/05 Gas Sensor



Obstacle Sensor



Ultrasonic Distance Sensor



LDR Sensor



Heartbeat Sensor



Gyro Sensor



GPS Sensor



Color Sensor



pH Sensor



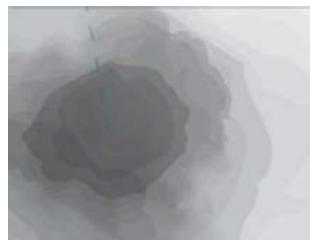
# MQ-02/05 Gas Sensor

- ▶ MQ -02 sensor is used to **detect smoke**.
- ▶ H<sub>2</sub>, LPG, CH<sub>4</sub> alcohol and smoke can be detected by MQ-02.
- ▶ MQ-05 **is not sensitive to smoke**, hence less used in the application.
- ▶ Pinout of MQ-02 sensor module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - D0 – Digital output pin
  - A0 – Analog output pin



# MQ-02/05 Gas Sensor – Working principle

- ▶ When any flammable gas passes through the coil of the sensor, the coil burns and **internal resistance decreases**.
- ▶ This results in an **increased voltage** across it. Hence we get variable voltage at A0 pin of MQ-05 gas sensor.
- ▶ If gas present -> voltage is high.
- ▶ If gas is not present -> voltage is low.



Smoke/Gas



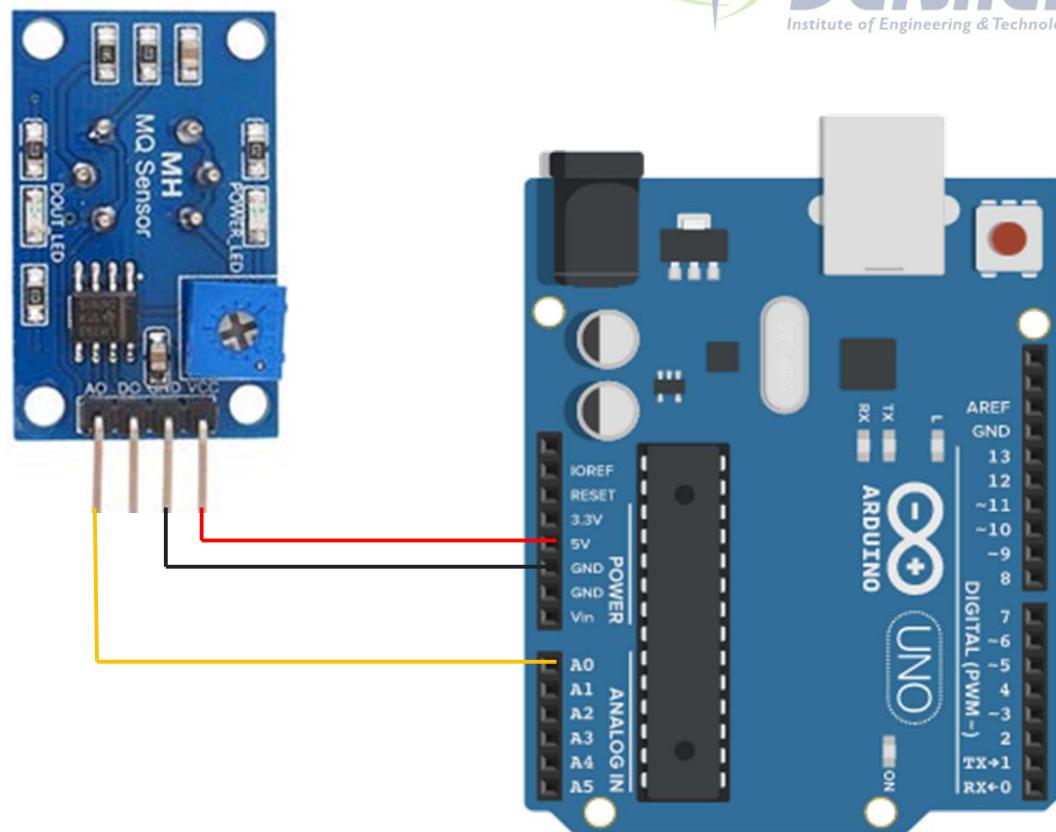
MQ-05 Gas Sensor



Variable Voltage

# MQ-02/05 Gas Sensor – Interfacing with Arduino

- ▶ The interfacing of MQ-02 with Arduino Uno is as shown in the figure.
- ▶ Here, Vcc and Gnd pins of MQ-02 are connected to 5V and GND of Arduino board.
- ▶ We want to take the input from a Gas sensor in **an analog** format so pin A0 is connected to one of the analog pins of Arduino i.e. A0.



# MQ-02/05 Gas Sensor – Code explanation

- ▶ The code in Arduino for MQ-02 gas sensor can be written as following

gas\_sensor.ino

```
1 int gas_level;
2 void setup() {
3     pinMode(A0, INPUT);
4     Serial.begin(9600);
5 }
6
7 void loop() {
8     gas_level = analogRead(A0);
9     Serial.print("Gas Level : ");
10    Serial.println(gas_level);
11    delay(500);
12 }
```

Initialize the serial communication between Arduino and computer with baud rate 9600.

Reads the analog value from specified pin and stores it in the mentioned variable.

Prints data on the serial port in ASCII text given in double quotes.

Prints the data of specified variable on the serial port and also prints new line at the end.

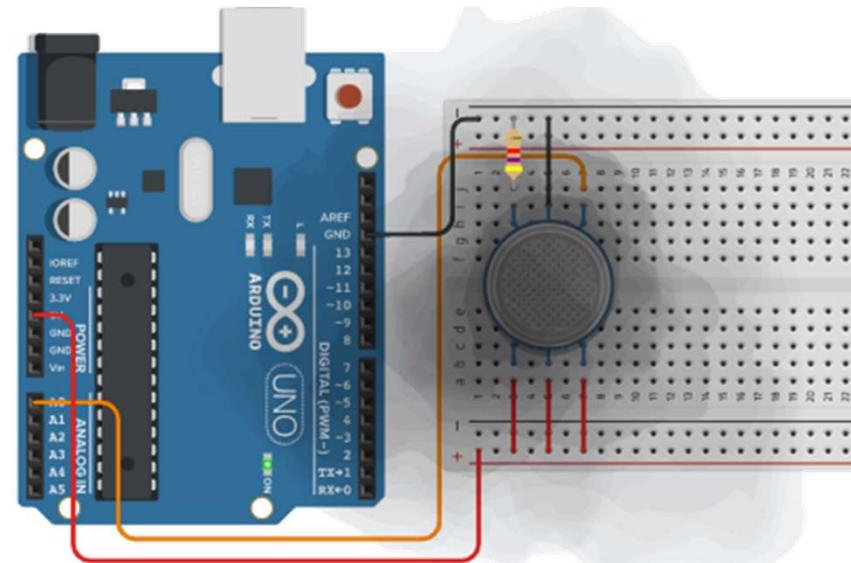
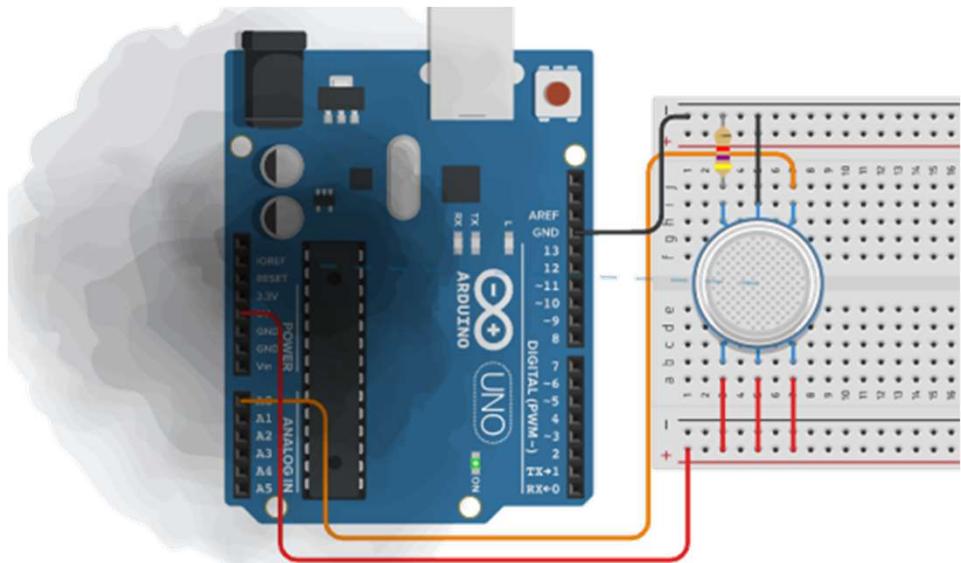
# Functions in Arduino IDE

- ▶ **Serial.begin( )**: It is used to start serial communication between Arduino board and other device. Normally, it is connected with computer for monitoring the data. Serial communication uses pin 0 and 1 also working as Rx and Tx respectively.
  - Syntax : `Serial.begin(baud_rate)`
  - Parameters :
    - `baud_rate`: It defines the speed of data transfer rate between Arduino and other device. It is compulsory to set this baud rate same at both the side for proper transfer of data.
- ▶ **Serial.print( )** : Print the data from Arduino to other device.
  - Syntax : `Serial.print("text")`
  - Parameters :
    - The text which is to be printed by Arduino board is written in double quote so as to convert it in its ASCII values.
- ▶ **Serial.println( )** : Print the data with newline from Arduino to other device.
  - Syntax : `Serial.println("text")`
  - Parameters :
    - The text which is to be printed by Arduino board is written in double quote so as to convert it in its ASCII values.

# Functions in Arduino IDE (Cont.)

- ▶ **analogRead( )**: The syntax reads analog value from specified pin and converts it into 1024 levels or in 0 to 1023 range. The values is stored into specified variable in the code statement. The value is converted into 1024 levels because Arduino has 10 bit built in A-D converter unit.
  - Syntax : `analogRead(pin)`
  - Parameters :
    - pin: The Arduino analog pin number.

# MQ-02/05 Gas Sensor – Output



 Serial Monitor

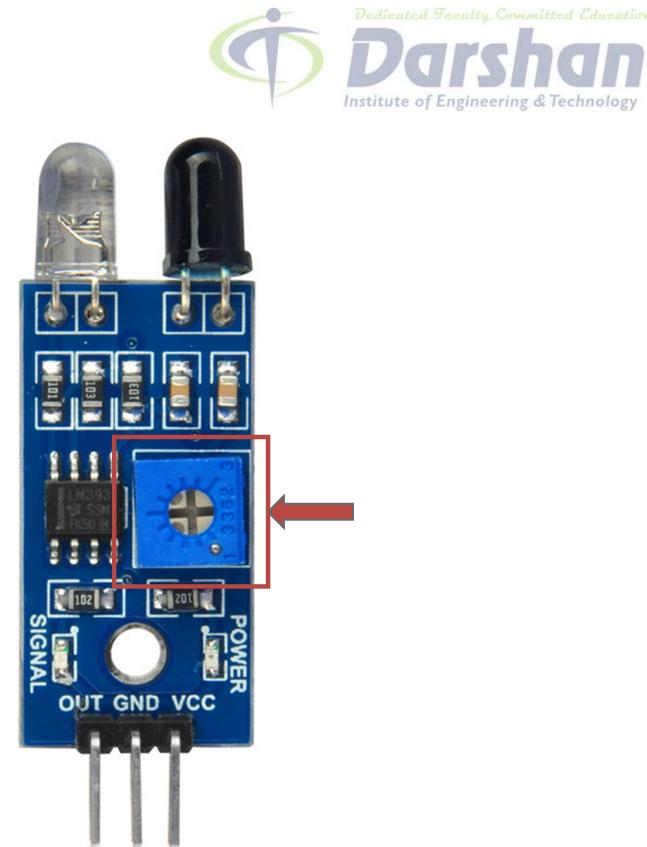
Gas Level : 306  
Gas Level : 306

 Serial Monitor

Gas Level : 745  
Gas Level : 745

# Obstacle Sensor

- ▶ Obstacle sensor is used for **detection of obstacle**.
- ▶ It is a digital sensor, hence gives binary output '1' or '0'.
- ▶ The **range** for detection of obstacle can be changed by **potentiometer** given.
- ▶ Pinout of obstacle sensor module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - Out/D0 – Digital output pin



# Obstacle Sensor – Working principle

- ▶ IR emitter transmits IR signals and IR receiver receives those signals from reflection.
- ▶ If the obstacle is present then the transmitted signals are **reflected** back by obstacle and if they have amplitude greater than threshold the output signal will be '**0**'.
- ▶ If the obstacle is not present then the transmitted signals are **not reflected** and the output signal will be '**1**'.



Obstacle in the way



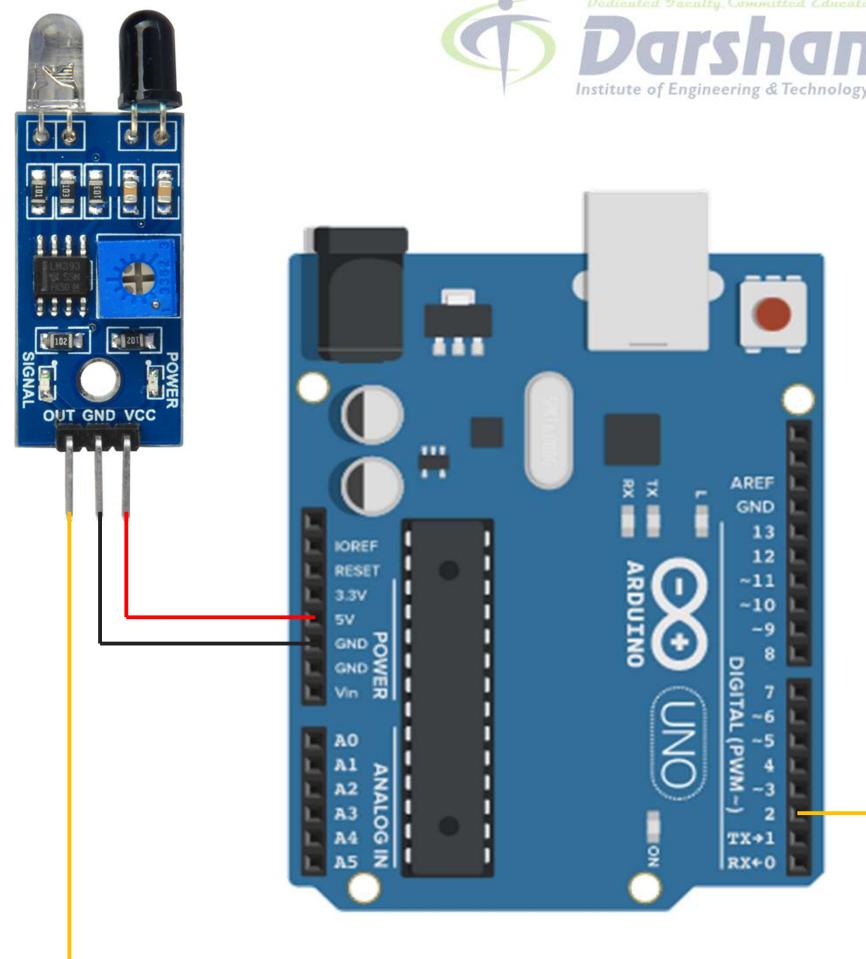
Obstacle IR Sensor



Digital Output

# Obstacle Sensor – Interfacing with Arduino

- ▶ The interfacing of obstacle sensor with Arduino Uno is as shown in figure.
- ▶ Here, Vcc and Gnd pins of obstacle sensor is connected to 5V and Gnd of Arduino board.
- ▶ The output of obstacle sensor is in **digital** form. So it is connected to digital pin 2 of Arduino.



# Obstacle Sensor – Code explanation

- ▶ The code in Arduino for Obstacle sensor can be written as following:

obstacle-detect.ino

```
1 int obstacle_pres=0;
2 void setup() {
3     pinMode(2, INPUT);
4     pinMode(13, OUTPUT);
5     Serial.begin(9600);
6 }
7
8 void loop() {
9     obstacle_pres = digitalRead(2);
10    if (obstacle_pres == LOW)
11    {
12        Serial.print("Obstacle is present");
13        digitalWrite(13,HIGH);
14    }
```

Reads the digital data from specified pin and stores it into given variable.

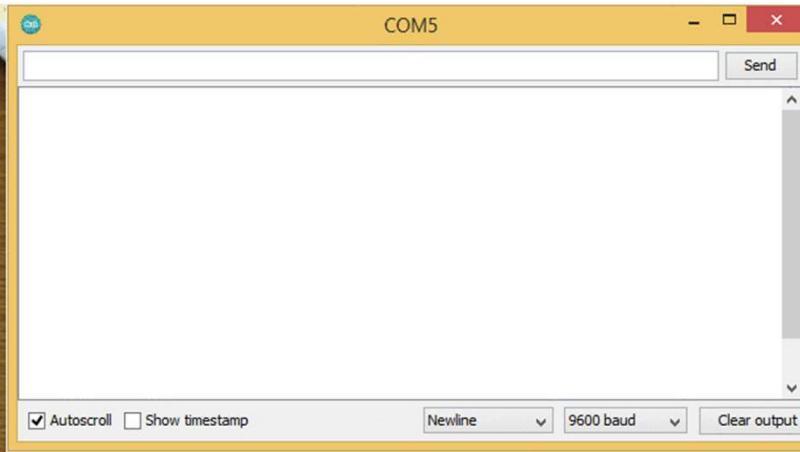
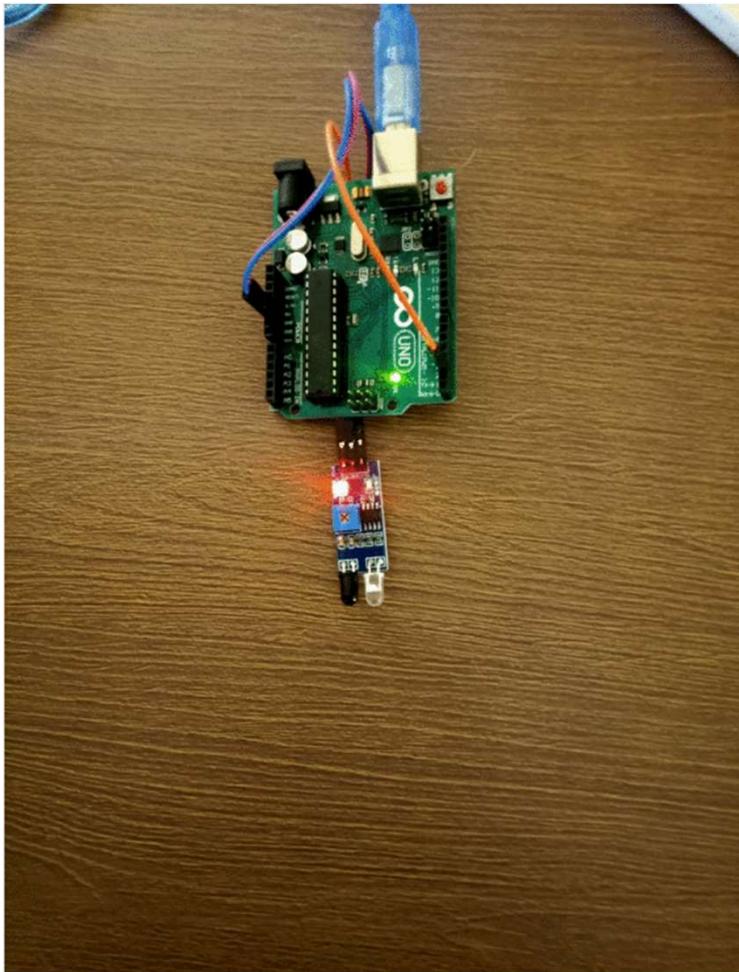
# Obstacle Sensor – Code explanation (Cont.)

obstacle-detect.ino

```
13 else
14 {
15     digitalWrite(13,LOW);
16 }
17 delay(1000);
18 }
```

- ▶ **digitalRead( )** : Reads digital data from specified pin and stores it into given variable.
  - Syntax : digitalRead(pin)
  - Parameters :
    - Pin : The Arduino digital pin number.

# Obstacle Sensor – Output



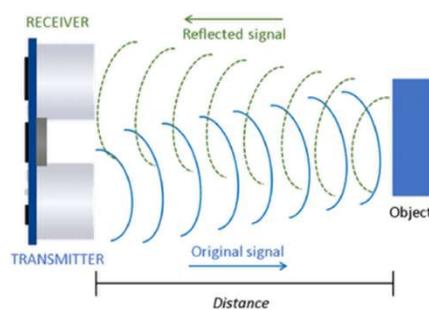
# HC-SR04 Ultrasonic Sound Sensor

- ▶ Ultrasonic Sound Sensor is used to measure the **distance of an object**.
- ▶ The sensor can only measure the distance of object. It can not identify the type of object.
- ▶ It is also known as Ultrasonic distance sensor.
- ▶ Pinout of HC-SR04 module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - Trigger – Generates the transmit pulse
  - Echo – Receives echo from obstacle

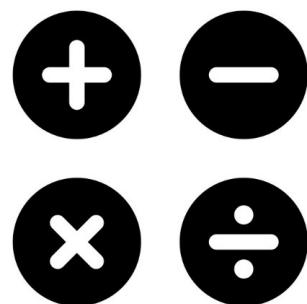


# HC-SR04 Ultrasonic Sound Sensor – Working principle

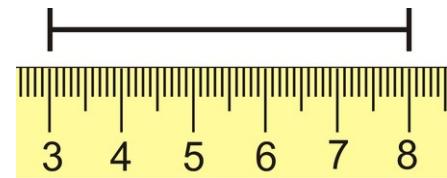
- ▶ The emitter transmits ultrasonic sound waves which are reflected by near by objects.
- ▶ The reflected pulse is received by sensor.
- ▶ Some mathematical operations are performed to obtain the distance value.
- ▶ The distance value is converted into desired unit.



Ultrasonic waves



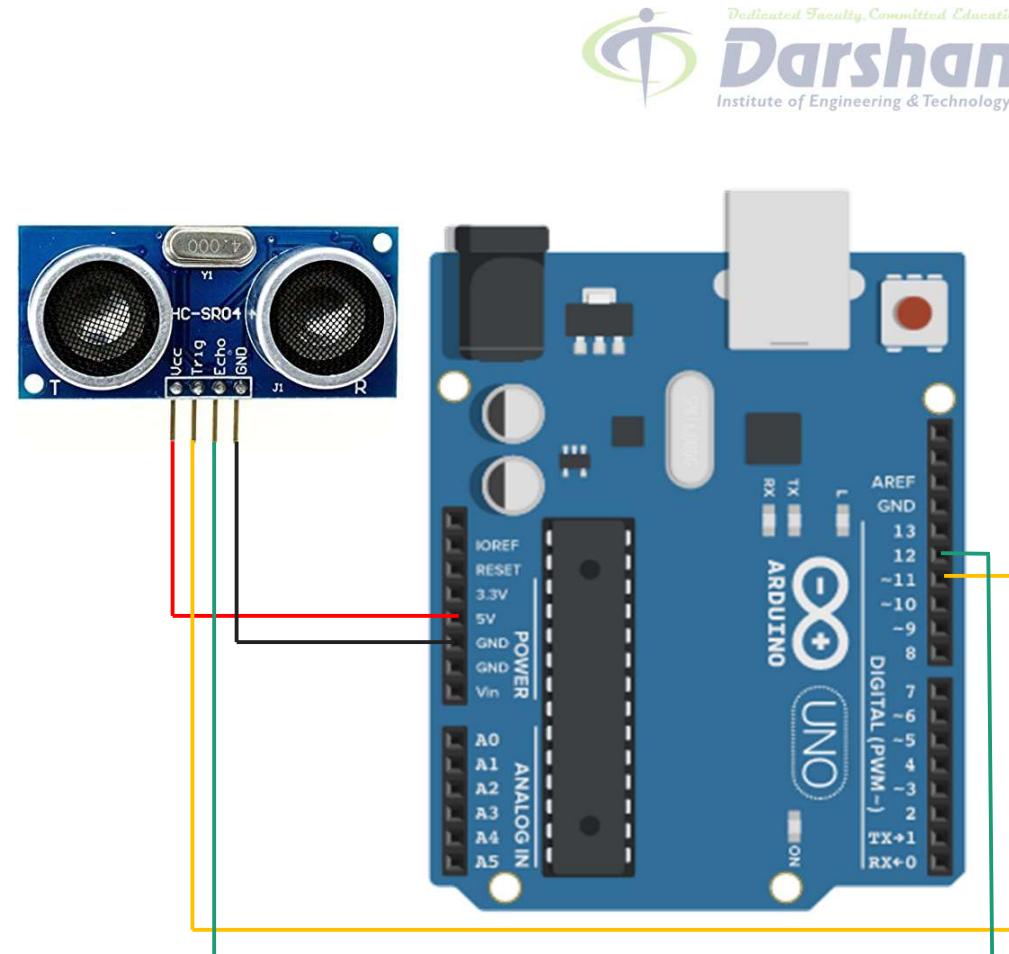
Mathematical operations



Measurement of distance

# HC-SR04 Ultrasonic Sound Sensor- Interfacing with Arduino

- ▶ The interfacing of HC-SR04 sensor with Arduino Uno is as shown in figure.
- ▶ Here, Vcc and Gnd pins of HC-SR04 sensor are connected to 5V and Gnd of Arduino board.
- ▶ The trigger and echo pin is connected to (any) digital pins of Arduino board.
- ▶ In this case, trigger is connected to 11 and echo is connected to 12.



# HC-SR04 Ultrasonic Sound Sensor – Code explanation

- ▶ The code in Arduino for HC-SR04 Ultrasonic Sound Sensor can be written as following

Distance-measure.ino

```
1 int trigPin = 11;      // Trigger
2 int echoPin = 12;      // Echo
3 long duration, cm, inches;
4
5 void setup() {
6     Serial.begin (9600);
7     pinMode(trigPin, OUTPUT);
8     pinMode(echoPin, INPUT);
9 }
10 void loop() {
11     digitalWrite(trigPin, LOW);
12     delayMicroseconds(5);
13     digitalWrite(trigPin, HIGH);
14     delayMicroseconds(10);
15     digitalWrite(trigPin, LOW);
```

Generates the delay of specified microseconds.

# HC-SR04 Ultrasonic Sound Sensor – Code explanation (Cont.)

Distance-measure.ino

```
16 pinMode(echoPin, INPUT);
17 duration = pulseIn(echoPin, HIGH);
18
19 // Convert the time into a distance
20 cm = (duration/2) / 29.1;
21 inches = (duration/2) / 74;
22 Serial.print(inches);
23 Serial.print("in, ");
24 Serial.print(cm);
25 Serial.print("cm");
26 Serial.println();
27
28 delay(250);
29 }
```

Reads a HIGH pulse on specified pin and returns the value of time in microsecond for transition from LOW to HIGH.

Distance(m) = Speed (m/s) x Time (s)

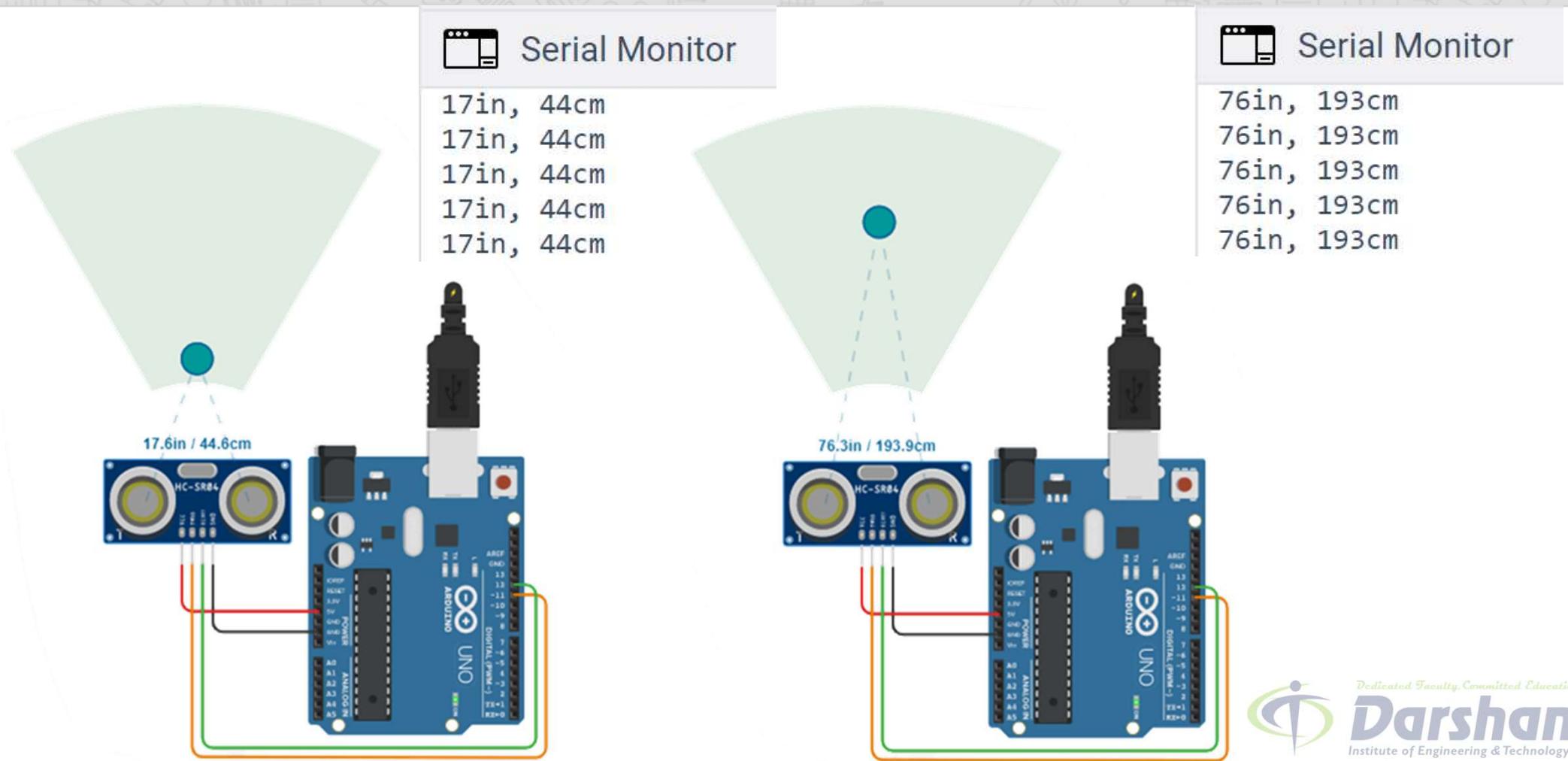
But, Time of pulse we receive is in  $\mu\text{s}$  and distance we want to find is in cm.  
Distance(cm) = Speed (cm/ $\mu\text{s}$ ) x Time ( $\mu\text{s}$ )

Speed of sound is 343m/s which is converted into 0.0343 cm/ $\mu\text{s}$ .  
So either we have to multiply 0.0343 or divide 29.1

# Functions in Arduino IDE

- ▶ **delayMicroseconds( )**: It pauses the program for amount of time in microseconds specified as the parameter.
  - Syntax : `delayMicroseconds(μs)`
  - Parameters :
    - $\mu\text{s}$  : The number of microseconds to pause.
- ▶ **pulseIn( )** : Reads a pulse HIGH or LOW from specified pin and wait for the time to go the pulse from HIGH to LOW or LOW to HIGH. It returns the time required to transit the pulse in variable.
  - Syntax : `pulseIn(pin, value)`
  - Parameters :
    - Pin: The number of the Arduino pin on which you want to read the pulse.
    - Value: The type of pulse that you want to read.

# HC-SR04 Ultrasonic Sound Sensor – Output



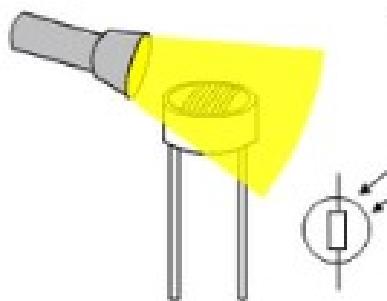
# LDR Sensor

- ▶ LDR is known as Light Dependent Resistor.
- ▶ The **internal resistance** of LDR changes with respect to light intensity.
- ▶ Normally LDR works with voltage divider network.
- ▶ The LDR is used in the application where the task is performed with light intensity as input.

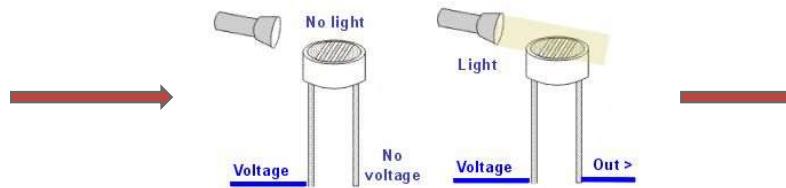


# LDR Sensor – Working principle

- ▶ The LDR works on principle of Ohm's law.
- ▶ As the light intensity on LDR is higher, the resistance of LDR decreases.
- ▶ The decreased resistance will drop more voltage across LDR according to Ohm's Law.
- ▶ The presence of light can be identified by the value of voltage across LDR.



Light Intensity to be measured



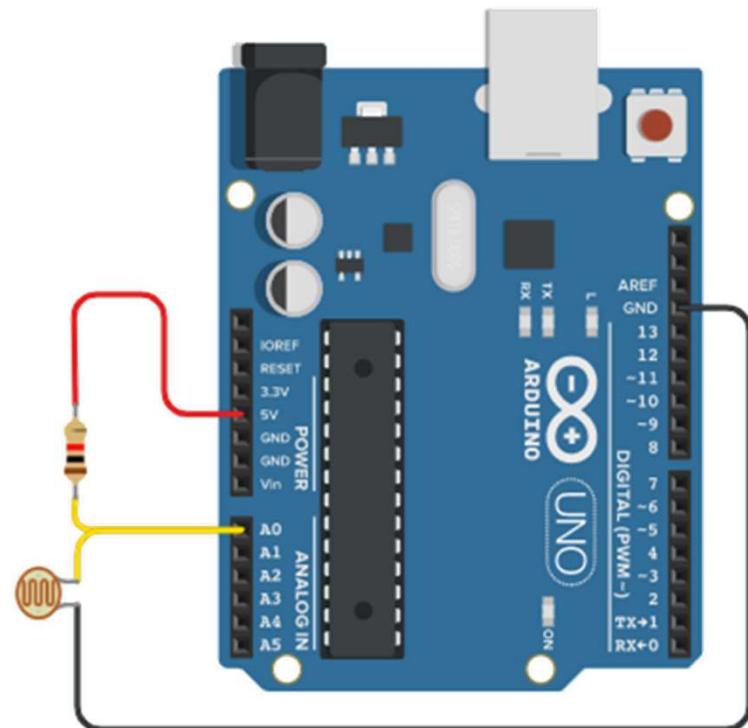
Output voltage measurement



Calculate the intensity

# LDR Sensor– Interfacing with Arduino

- ▶ The interfacing of LDR sensor with Arduino Uno is as shown in figure.
- ▶ Here, one terminal of  $10k\Omega$  resistor is connected to 5V of Arduino and second terminal is connected to A0 pin.
- ▶ One terminal of LDR is connected to A0 pin of Arduino and other is connected to Gnd.
- ▶ This creates a voltage divider network between fixed resistor and LDR.
- ▶ The voltage at A0 pin remains near to 5V when LDR is in dark (high resistance).
- ▶ The voltage at A0 pin remains near to 0V when LDR is in Bright light (low resistance).



# LDR Sensor – Code explanation

- ▶ The code in Arduino for LDR Sensor can be written as following

Light-intensity-measure.ino

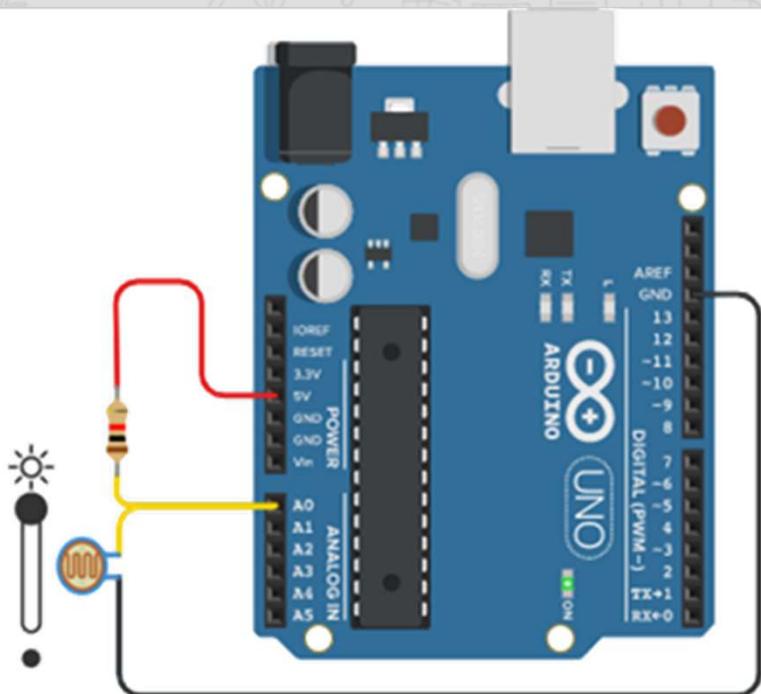
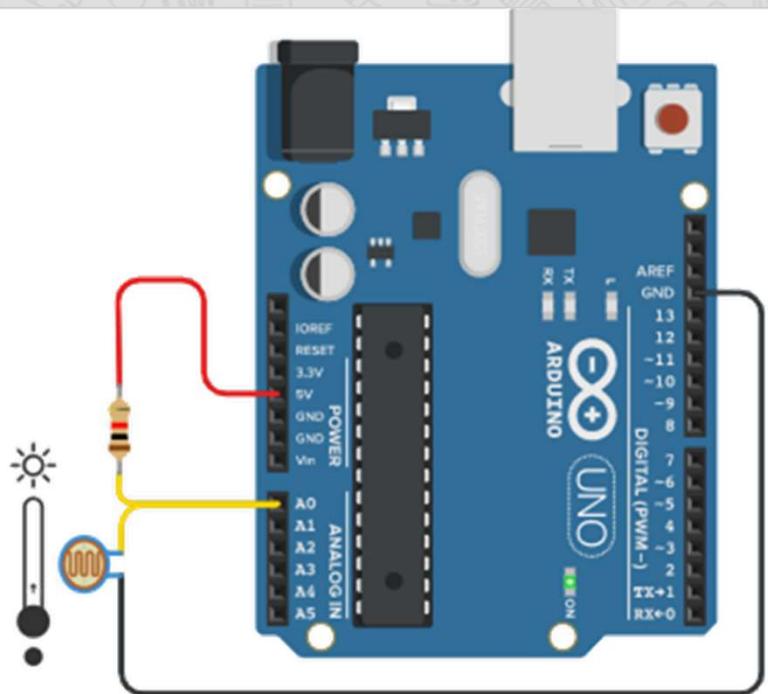
```
1 void setup()
2 {
3     pinMode(A0, INPUT);
4     Serial.begin(9600);
5 }
6
7 void loop()
8 {
9     int light_intensity = analogRead(A0);
10    if (light_intensity > 800)
11    {
12        Serial.println("Darker");
13    }
```

# LDR Sensor – Code explanation (Cont.)

Light-intensity-measure.ino

```
16  else if (light_intensity < 500)
17  {
18      Serial.println("Too Bright");
19  }
20  delay(1000);
21 }
```

# LDR Sensor – Output



Serial Monitor

Darker  
Darker  
Darker  
Darker

Serial Monitor

Too Bright  
Too Bright  
Too Bright  
Too Bright

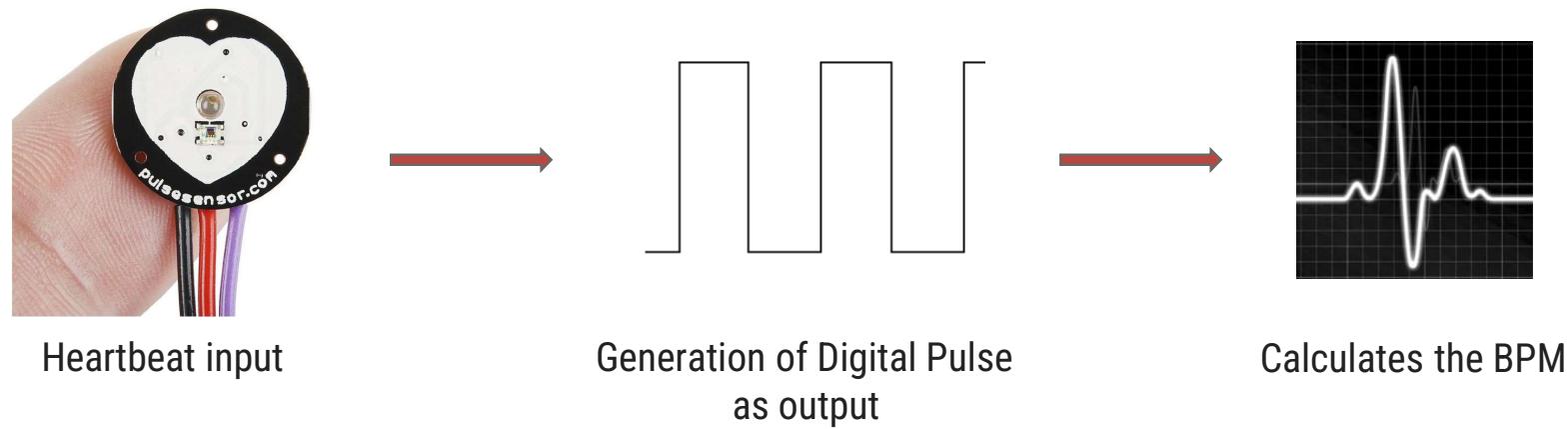
# Heartbeat Sensor

- ▶ Heartbeat sensor is having typical application in health care domain.
- ▶ This is mainly used in wearable devices to monitor the heart beat / heart rate of the person.
- ▶ The sensor is inexpensive and generates square waves for each pulse. The averaging of pulses gives analog voltage according to the heartbeat.
- ▶ Pinout of heartbeat sensor module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - A0 – Analog output pin



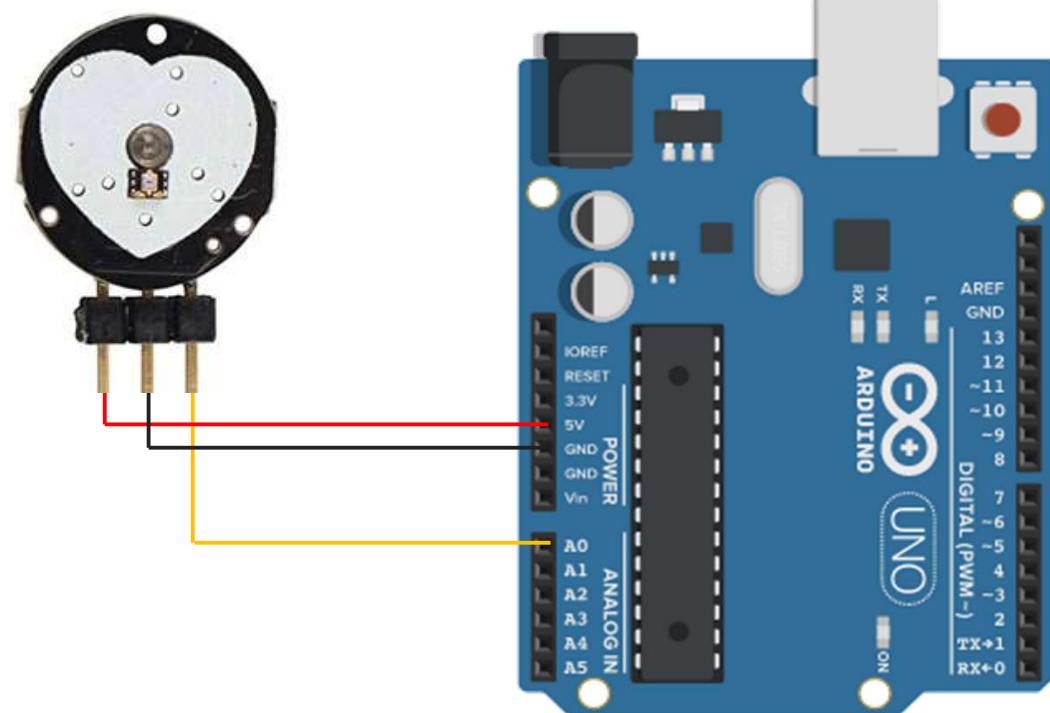
# Heartbeat Sensor – Working principle

- ▶ The sensor senses the heartbeat as per blood circulation in the body.
- ▶ It generates the output pulse signal according to the rate at which heart beats.
- ▶ The output pulses are considered as PWM waves. Hence they are converted into analog signals.
- ▶ This analog signal is converted to digital which gives the output beat rate.



# Heartbeat Sensor – Interfacing with Arduino

- ▶ The interfacing of Heartbeat sensor with Arduino Uno is as shown in the figure.
- ▶ Here, Vcc and Gnd pins of heartbeat sensor are connected to 5V and GND of Arduino board.
- ▶ We want to take the input from the sensor in **an analog** format so the signal pin of the sensor is connected to one of the analog pins of Arduino i.e. A0.



# Heartbeat Sensor – Code explanation

- ▶ The code in Arduino for Heartbeat sensor can be written as following:

Heartbeat\_sensor.ino

```
1 int UpperThreshold = 518;
2 int LowerThreshold = 490;
3 int reading = 0;
4 float BPM = 0.0;
5 bool IgnoreReading = false;
6 bool FirstPulseDetected = false;
7 unsigned long FirstPulseTime = 0;
8 unsigned long SecondPulseTime = 0;
9 unsigned long PulseInterval = 0;
10
11 void setup(){
12     Serial.begin(9600);
13 }
```

# Heartbeat Sensor – Code explanation (Cont.)

Heartbeat\_sensor.ino

```
14 void loop(){
15     reading = analogRead(0);
16     if(reading > UpperThreshold
17         && IgnoreReading == false){
18         if(FirstPulseDetected == false){
19             FirstPulseTime = millis();
20             FirstPulseDetected = true;
21         }
22     else{
23         SecondPulseTime = millis();
24         PulseInterval = SecondPulseTime -
25                         FirstPulseTime;
26         FirstPulseTime = SecondPulseTime;
27     }
28     IgnoreReading = true;
29 }
```

millis( ) function gives the time in millisecond from where the Arduino board powered or reset.

# Heartbeat Sensor – Code explanation (Cont.)

Heartbeat\_sensor.ino

```
30  if(reading < LowerThreshold){  
31      IgnoreReading = false;  
32  }  
33  
34  BPM = (1.0/PulseInterval) * 60.0 * 1000;  
35  Serial.print(BPM);  
36  Serial.println(" BPM");  
37  Serial.flush();  
38 }
```

PulseInterval is in milliseconds.

$$\therefore \text{Rate} = 1/\text{Time}$$

$$\therefore \text{Rate} = \frac{1}{\text{PulseInterval(ms)}}$$

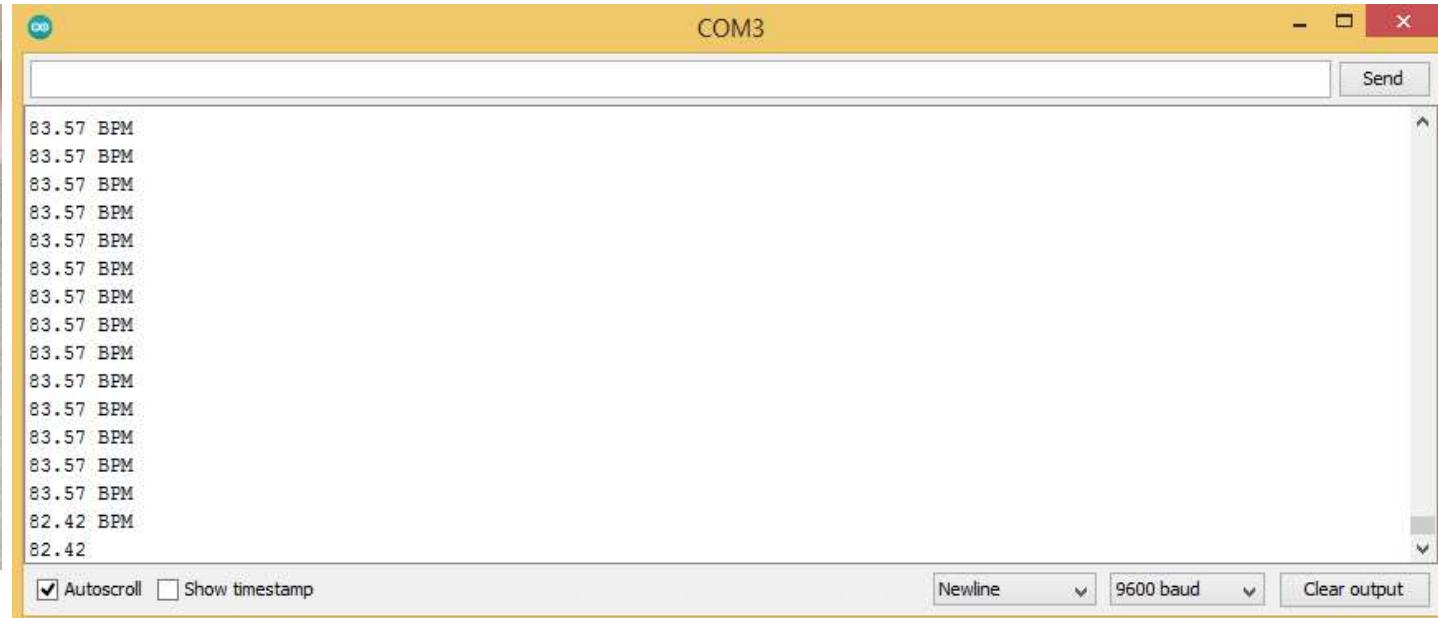
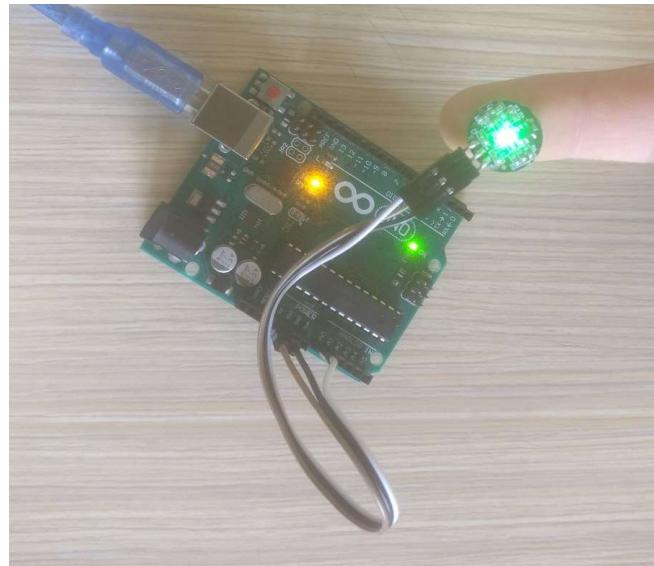
$$\therefore \text{Rate} = \frac{1}{\text{PulseInterval(s)}/1000}$$

$$\therefore \text{Rate} = \frac{1}{\text{PulseInterval(s)}} * 1000$$

$$\therefore \text{Rate in min} = \frac{1}{\text{PulseInterval}} * 1000 * 60$$

- ▶ millis( ) : Returns the number of milliseconds passed since the Arduino started running the current program.
  - Syntax : millis()

# Heartbeat Sensor – Output



# Colour Sensor

- ▶ Colour sensor is used to detect the RGB colour coordinates of a particular colour.
- ▶ The colour sensor module has TSC3200 IC that converts colour to frequency by enabling a particular colour diode turn by turn.
- ▶ The colour sensor is mainly used in the application where the objects are identified by their colour.
- ▶ Pinout of colour sensor module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - S0, S1 – Used for output frequency scaling
  - S2, S3 – Type of photodiode selected
  - (OE)' – Enable for output
  - OUT – Output frequency ( $f_o$ )



# Colour Sensor (Cont.)

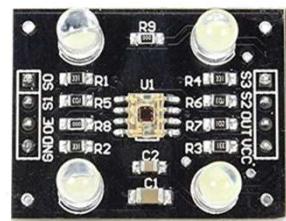
- ▶ The combination of S0 and S1 are used for output frequency scaling.
- ▶ The different microcontrollers have different counter functionality and limitations. Hence, we need to scale the frequency according to microcontroller used.
- ▶ The percentage of output scaling for different combinations of S0 and S1 are shown in the table.
- ▶ The S3 and S4 pins are used for photodiode selection. The photodiodes are associated with different colour filters.
- ▶ The table shows combinations of S3 and S4 for different photodiodes.

S0	S1	Output Frequency Scaling ( $f_0$ )	Typical full-scale Frequency
L	L	Power Down	---
L	H	2%	10 – 12 KHz
H	L	20%	100 – 120 KHz
H	H	100%	500 – 600 KHz

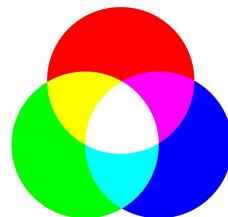
S3	S4	Photo Diode Type
L	L	Red
L	H	Blue
H	L	Clear(No Filter)
H	H	Green

# Colour Sensor – Working principle

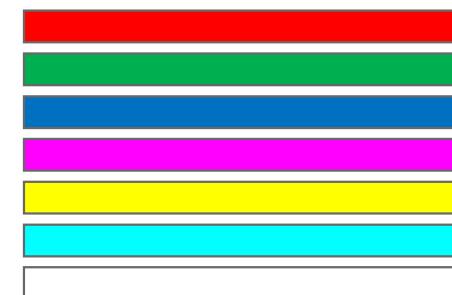
- ▶ The sensor works by imparting a bright light on an object and recording the reflected colour by the object.
- ▶ The selected photodiode for colour of red, green and blue converts the amount of light to current.
- ▶ These RGB values are further processed to identify the exact colour combination.



Input to photodiode  
of colour sensor



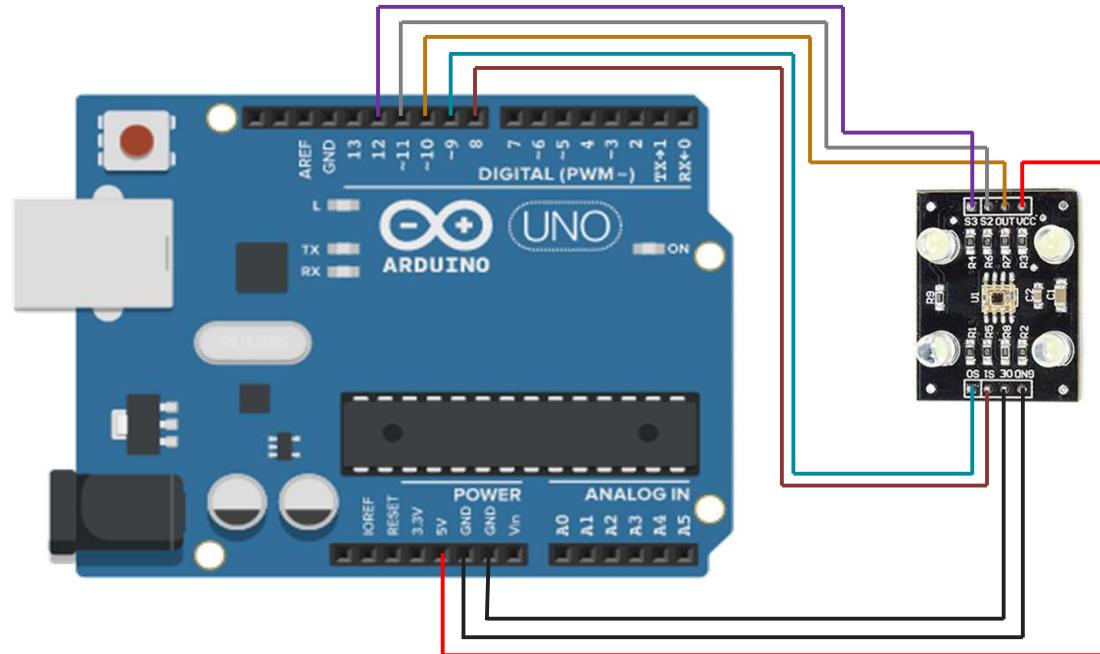
Output RGB colour values



Calculation of actual  
colour of object

# Colour Sensor – Interfacing with Arduino

- ▶ The interfacing of Colour Sensor with Arduino Uno is as shown in figure.
- ▶ Here, Vcc and Gnd pins of Colour Sensor are connected to 5V and Gnd of Arduino board.
- ▶ As we need to enable the sensor OE' is connected to GND.
- ▶ All selection pins S0, S1 S2 and S3 are connected to digital pins of Arduino that is 9,8,11 and 12 respectively.
- ▶ The output of colour sensor are taken as a pulse count. So OUT pin is also connected to digital pin. In our case it is connected to pin 10.



# Colour Sensor – Code explanation

- ▶ The code in Arduino for Colour sensor can be written as following:

Colour\_sensor.ino

```
1 #define s0 9
2 #define s1 8
3 #define s2 11
4 #define s3 12
5 #define out 10
6
7 int frequency=0;
8
9 void setup()
10 {
11     pinMode(s0,OUTPUT);
12     pinMode(s1,OUTPUT);
13     pinMode(s2,OUTPUT);
14     pinMode(s3,OUTPUT);
15     pinMode(out,INPUT);
```

# Colour Sensor – Code explanation (Cont.)

Colour\_sensor.ino

```
16  Serial.begin(9600);
17  digitalWrite(s0,HIGH);
18  digitalWrite(s1,HIGH);
19 }
20 void loop()
21 {
22  digitalWrite(s2,HIGH);
23  digitalWrite(s3,HIGH);
24  frequency = pulseIn(out,LOW);
25  frequency = map(frequency, 30,90,255,0);→
26  Serial.print("G = ");
27  Serial.print(frequency);
28  Serial.print("\t");
29  delay(100);
30 }
```

This syntax maps the value of given number or variable from its defined range to desired range.

# Colour Sensor – Code explanation (Cont.)

Colour\_sensor.ino

```
31   digitalWrite(s2,LOW);
32   digitalWrite(s3,HIGH);
33   frequency = pulseIn(out,LOW);
34   frequency = map(frequency,25,70,255,0);
35   Serial.print("B = ");
36   Serial.print(frequency);
37   Serial.print("\t");
38   delay(100);
39   digitalWrite(s2,LOW);
40   digitalWrite(s3,HIGH);
41   frequency = pulseIn(out,LOW);
42   frequency = map(frequency,10,56,255,0);
43   Serial.print("R = ");
44   Serial.print(frequency);
45   Serial.print("\t");
46   delay(100);
47 }
```

# Functions in Arduino IDE

► **map( )**: The syntax maps a number from one range to another.

→ Syntax : map(value, fromLow, fromHigh, toLow, toHigh)

→ Parameters :

- value: the number or variable to map.
- fromLow: the lower bound of the value's current range.
- fromHigh: the upper bound of the value's current range.
- toLow: the lower bound of the value's target range.
- toHigh: the upper bound of the value's target range.

# GPS Sensor

- ▶ GPS sensor is used to get the location data of the place where sensor is situated.
- ▶ The data from sensor is acquired in National Marine Electronics Association (NEMA) format. The format is used by marine department for communication.
- ▶ There are multiple variants of GPS sensors available in the market. We can select based on our requirements.
- ▶ Here, the given sensor is GY-GPS6MV2 as shown in the image.
- ▶ Pinout of GPS sensor module are
  - Vcc – Connected to 5V
  - Gnd – Connected to ground
  - Tx – Serial Data Transmit
  - Rx – Serial Data Receive



# GPS Sensor – Working principle

- ▶ The sensor is placed in the system whose location is to be tracked.
- ▶ The GPS device communicates with GPS satellite and the satellite returns its current location in form of latitude and longitude.
- ▶ The sensor transmits the data to the microcontroller in a predefined format from which we can extract the tracked location.



Latitude  
Longitude



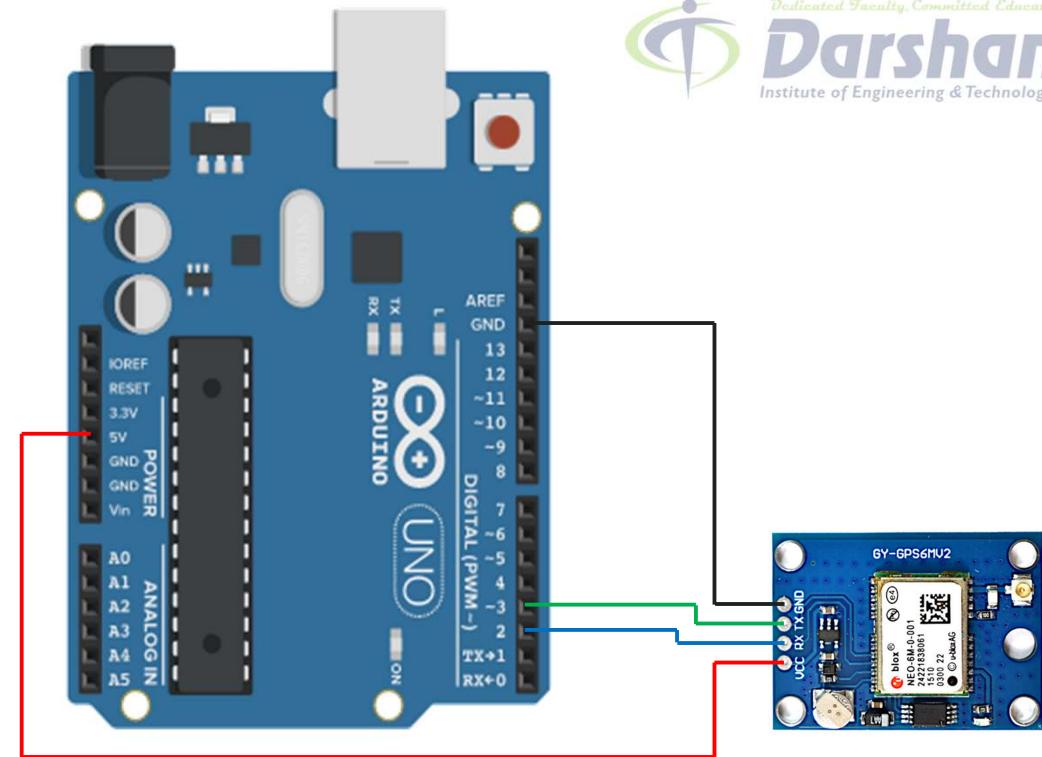
GPS device with antenna  
for communication

Returns latitude and longitude  
as output

Tracked Location

# GPS Sensor – Interfacing with Arduino

- ▶ The interfacing of GPS Sensor with Arduino Uno is as shown in figure.
- ▶ Here, Vcc and Gnd pins of GPS Sensor are connected to 5V and Gnd of Arduino board.
- ▶ As the GPS communicates with Arduino serially, we need to connect Tx and Rx pin of GPS with serial pins of Arduino.
- ▶ But, we want the data received from the GPS sensor in serial monitor also to read the current location.
- ▶ Therefore, we need to use any digital pins of Arduino as serial transmit and receive pins which is done by software serial.
- ▶ Here, the Tx and Rx pins of GPS are connected with pins 3 and 2 respectively.



# GPS Sensor – Code explanation

- ▶ The code in Arduino for GPS sensor can be written as following:

GPS\_sensor.ino

```
1 #include <TinyGPS++.h>
2 #include <SoftwareSerial.h>
3 static const int RXPin = 3, TXPin = 2;
4 static const uint32_t GPSBaud = 9600;
5 TinyGPSPlus gps;
6
7 SoftwareSerial ss(RXPin, TXPin);
8
9 void setup()
10 {
11   Serial.begin(115200);
12   ss.begin(GPSBaud);
13 }
```

Includes the TinyGPS++ library to use built in functions for reading GPS sensor.

Includes the SoftwareSerial library to use software serial communication

Defining *gps* as object of TinyGPSPlus

Defining *ss* as object of SoftwareSerial

# GPS Sensor – Code explanation (Cont.)

GPS\_sensor.ino

```
14 void loop()
15 {
16     while (ss.available() > 0)
17         gps.encode(ss.read());
18     displayInfo();
19
20     if (millis()>5000 &&
21         gps.charsProcessed()<10)
22     {
23         Serial.println("No GPS detected: Check
24 wiring.");
25         while(true);
26     }
27 }
```

The syntax reads the data serially from GPS and `gps.encode()` function encodes that data as per the format.

# GPS Sensor – Code explanation (Cont.)

GPS\_sensor.ino

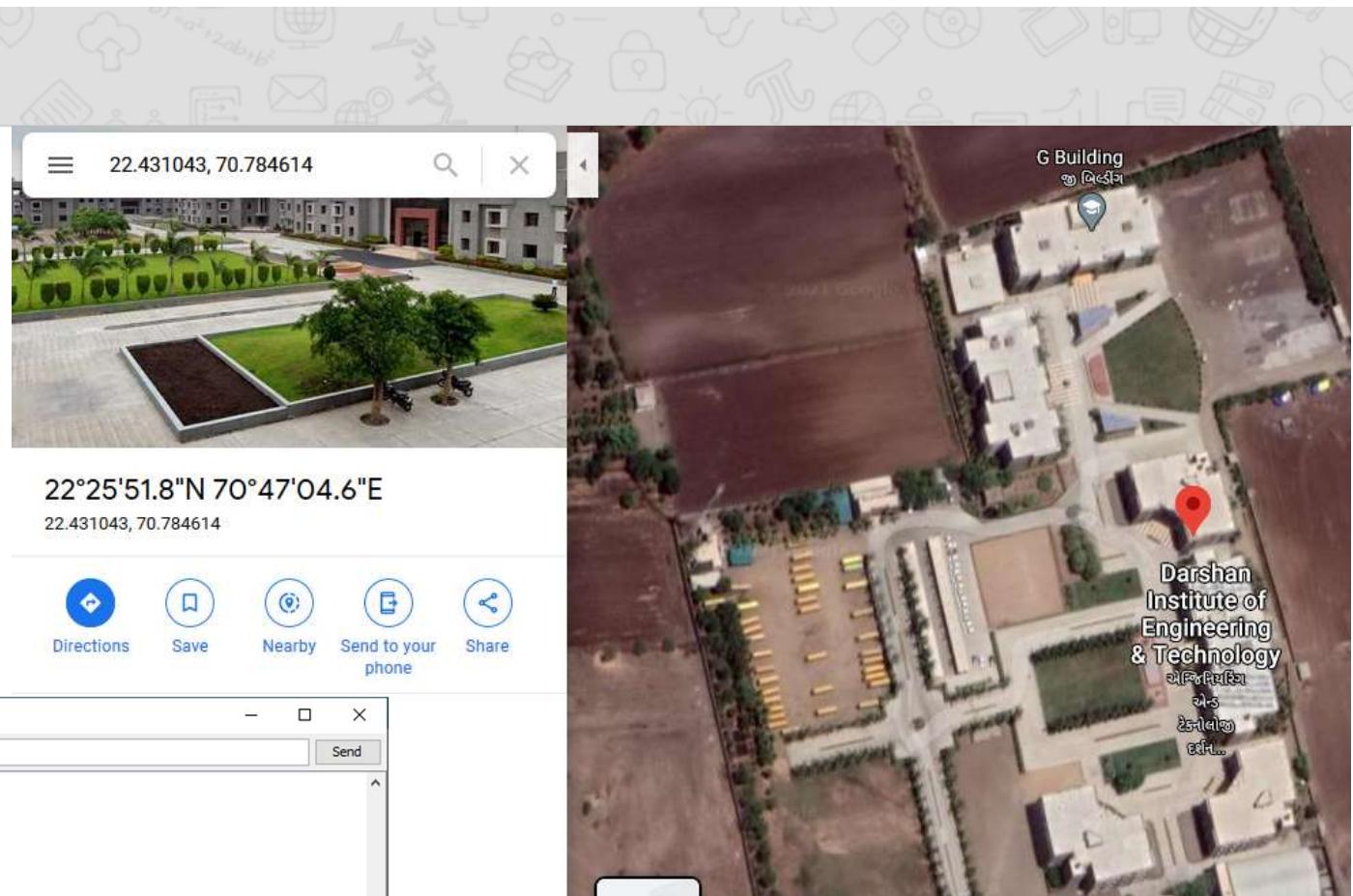
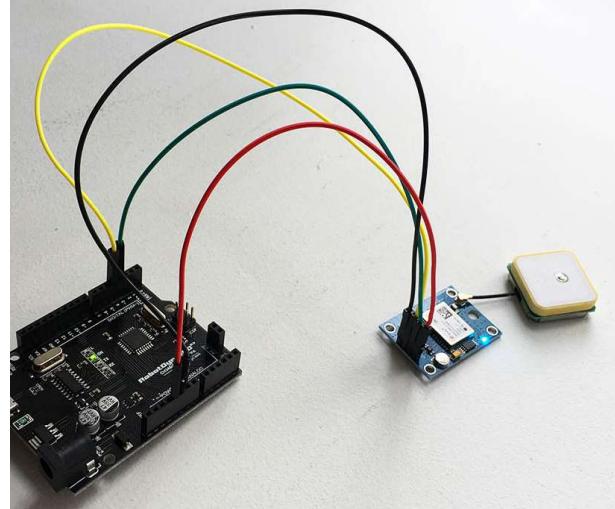
```
28 void displayInfo() {  
29     Serial.print("Location: ");  
30     if (gps.location.isValid())  
31     {  
32         Serial.print(gps.location.lat(), 6);  
33         Serial.print(",");  
34         Serial.print(gps.location.lng(), 6);  
35     }  
36     Serial.println();  
37 }
```

*gps.location.isValid()* function returns "TRUE" if the location is in valid form

*gps.location.lat()* function returns the current latitude value and it is printed with 6 decimal point precision.

*gps.location.lng()* function returns the current longitude value and it is printed with 6 decimal point precision.

# GPS Sensor – Output



```
COM4
|
Location: 22.431043, 70.784614
```

# Gyro Sensor

- ▶ The Gyro sensor is very useful in **wearable devices**.
- ▶ It is used to find the position or angle of rotation of the body.
- ▶ It mainly senses
  - Rotational motion
  - Changes in orientation
- ▶ The Gyro sensor communicates with microcontroller with I2C (Inter-Integrated Communication).
- ▶ I2C is the most sophisticated two-wire communication protocol.
- ▶ It works with the device address and its respective data frames.
- ▶ Normally, I2C is used when more numbers of devices are connected.



# Gyro Sensor (Cont.)

- ▶ The pin out of Gyro sensor MPU6050 are as following.
  - Vcc – connected to 5V
  - GND – connected to ground
  - SCL – Serial Clock Line
  - SDA – Serial Data Line
  - XDA – Auxiliary Serial Data
  - XCL – Auxiliary Serial Clock
  - AD0 – This pin can be used to vary the address incase of multiple sensors used
  - INT – Interrupt pin available for certain application

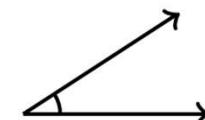


# MPU-6050 Gyro Sensor – working principle

- ▶ The MPU 6050 senses the gravitational force applied to it.
- ▶ Place the sensor in device whose inclination is to be measured.
- ▶ Obtain the values of x, y and z coordinates.
- ▶ Determine the elevation angle by performing mathematical caluculations



x,y,z : 507 506 617  
x,y,z : 507 504 616  
x,y,z : 506 505 617  
x,y,z : 506 506 618



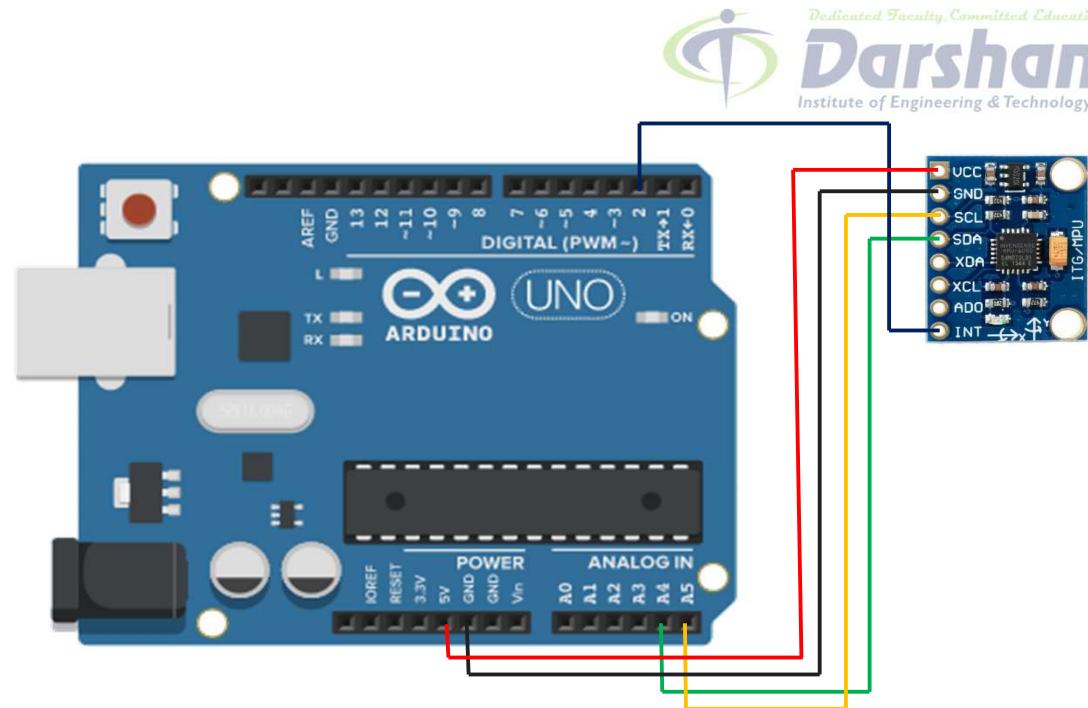
Sensor placed for  
finding Inclination

x, y, and z coordinates

Determination of Angle

# MPU-6050 Gyro Sensor – Interfacing with Arduino

- ▶ The interfacing of MPU-6050 with Arduino Uno is as shown in figure.
- ▶ Here, Vcc and Gnd pins of MPU6050 is connected to 5V and Gnd of Arduino board.
- ▶ The SCL and SDA pins are used for I2C communication. These pins are available in Arduino with A5 and A4 pins as second functionality.
- ▶ Hence, SCL and SDA are connected to A5 and A4 pins of Arduino.
- ▶ The INT pin of MPU6050 gives interrupt when angular velocity increases than defined threshold value. So interrupt pin of MPU6050 is connected to external interrupt by Arduino supported at digital pin 2.



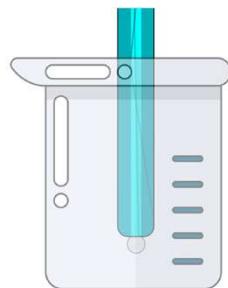
# PH Sensor

- ▶ The pH sensor is used to detect hydrogen ions concentration of a liquid.
- ▶ pH sensor are mostly used in laboratories to test the acidity of solution.
- ▶ By the use of pH sensor we can identify the pH of a solution and whether it is acid or base.



# pH Sensor – Working principle

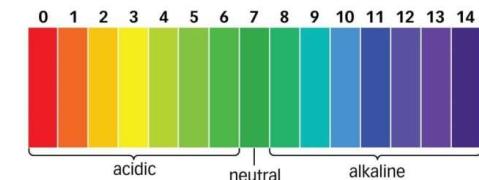
- When this sensor is placed in a solution, the smaller ions penetrate the boundary area of glass and the larger ions remain in the solution. This creates potential difference.
- The pH meter measures the difference in electrical potential between the pH electrodes.
- The potential difference generates different analog values for different liquids.
- By knowing the analog value of standard water, the pH value of other liquid can be determined.



pH Sensor with  
Electrodes



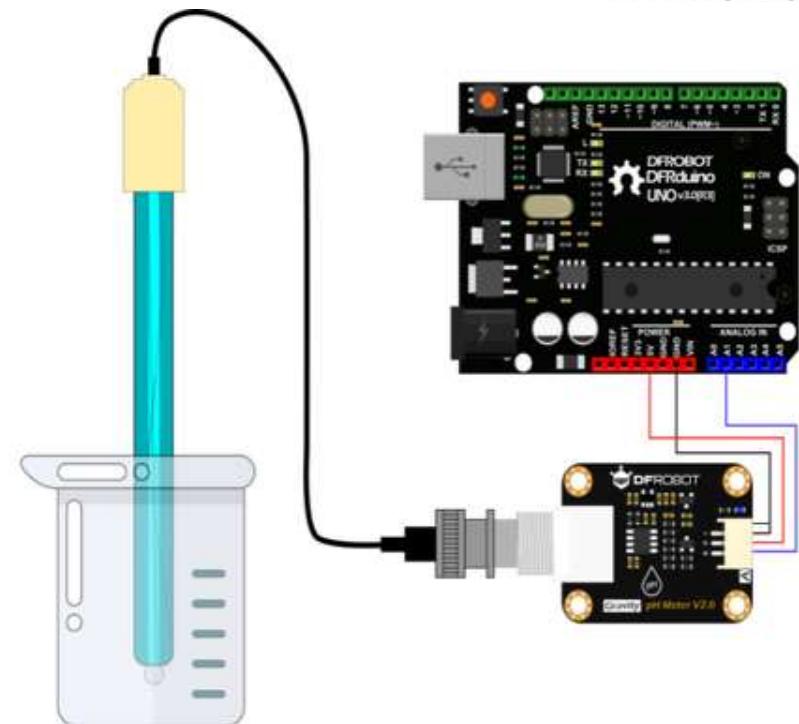
Output Voltage



Determination of alkalinity  
of the liquid

# pH Sensor – Interfacing with Arduino

- ▶ The interfacing of pH sensor with Arduino is as shown in figure.
- ▶ Here, Vcc and Gnd pins of pH Sensor are connected to 5V and Gnd of Arduino board.
- ▶ The output pin of pH sensor is connected to analog pin of Arduino.
- ▶ The connection of pH sensor is done with voltage converter via BNC pin.



# pH Sensor – Code explanation

- ▶ The code in Arduino for pH sensor can be written as following:

PH\_sensor.ino

```
1 #define SensorPin A0
2 #define Offset 0
3 unsigned long int avgValue;
4 float b;
5 int buf[10],temp;
6
7 void setup()
8 {
9   pinMode(SensorPin,INPUT);
10  Serial.begin(9600);
11 }
```

Defining *Offset* for correction of pH value. The offset value can be obtained by measuring the pH of standard water.

Defining *buf* as an integer array.

# pH Sensor – Code explanation (Cont.)

PH\_sensor.ino

```
12 void loop()
13 {
14     for(int i=0;i<10;i++) {
15         buf[i]=analogRead(SensorPin);
16         delay(10);
17     }
18     for(int i=0;i<9;i++) {
19         for(int j=i+1;j<10;j++) {
20             if(buf[i]>buf[j])
21             {
22                 temp=buf[i];
23                 buf[i]=buf[j];
24                 buf[j]=temp;
25             }
26         }
27     }
```

Taking 10 different values of analog voltage from the pin where sensor is connected

Sorting the elements of array *buf* in ascending order.

# pH Sensor – Code explanation (Cont.)

PH\_sensor.ino

```
28 avgValue=0;  
29 for(int i=2;i<8;i++)  
30     avgValue+=buf[i];  
31 avgValue = avgValue/6;  
32 float phValue=(float)((avgValue)*  
33                                5.0/1024);  
34 phValue=3.5*phValue+Offset;  
35 Serial.print("pH: ");  
36 Serial.print(phValue,2);  
37 Serial.println(" ");  
38 delay(800);  
39 }
```

Avoiding lower and upper two (total four) values of array and taking the average of elements of rest of the array.

Converting the average value into equivalent millivolts.

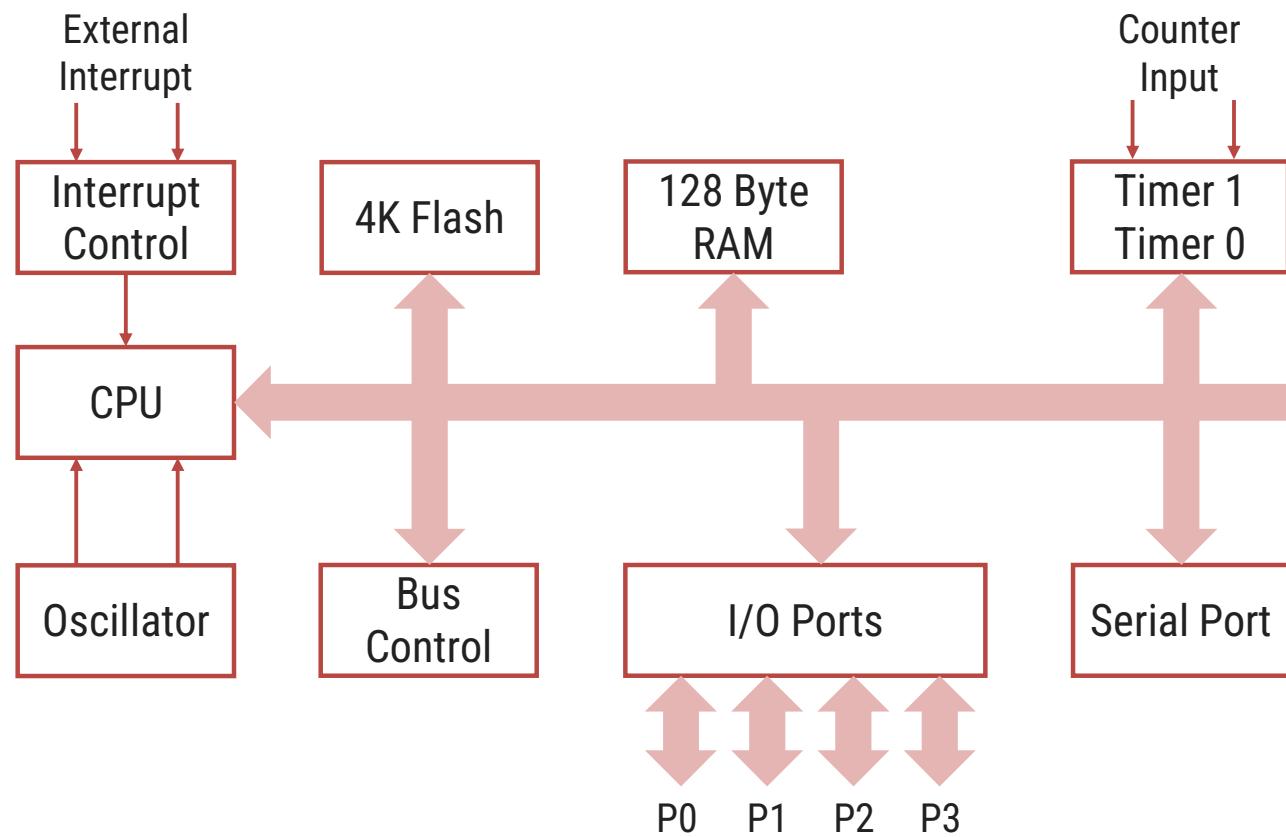
Converting millivolts to its equivalent pH values as given in [datasheet](#) and adjusting with offset.

# Other Microcontrollers

Section - 3

# 8051 Microcontroller Architecture

- ▶ 8051 was built by Intel but other companies are also permitted to make 8051 microcontrollers with same features and compatible set of instructions.
- ▶ The diagram shows Architectural representation of 8051 microcontroller.



# Features of 8051 Microcontroller

1. 8-bit microcontroller.
2. Many general purpose and few special function registers (SFRs).
  - Two major 8 bit registers. A and B. Both are used in arithmetic operations mainly.
  - A is accumulator and it addressable.
  - There are 21 SFRs among them few are bit addressable. SFRs perform various dedicated operations.
  - Also, some control registers for timer, counter and interrupt fall in category of SFRs.
3. Four register banks having 8 registers in each bank.
4. Data pointer register which is 16 bit made from combinations of two 8 bit registers – DPH and DPL.
5. 16 bit Program Counter.
6. 8 bit Program Status Word(PSW) and used as Flag register.
7. Internal ROM and EPROM.
8. Internal user accessible RAM of 128 Bytes

# Features of 8051 Microcontroller

9. There are **four ports** P0, P1, P2 and P3 that can be configured as input and output.
10. Two 16 bit timer/counter – T0 and T1.
11. Full duplex serial communication and dedicated serial buffer register SBUF.
12. Supports interrupt programming.
13. Oscillator and Clock circuits are built in.
14. Easier and simpler instruction set.

# Memory organization in 8051

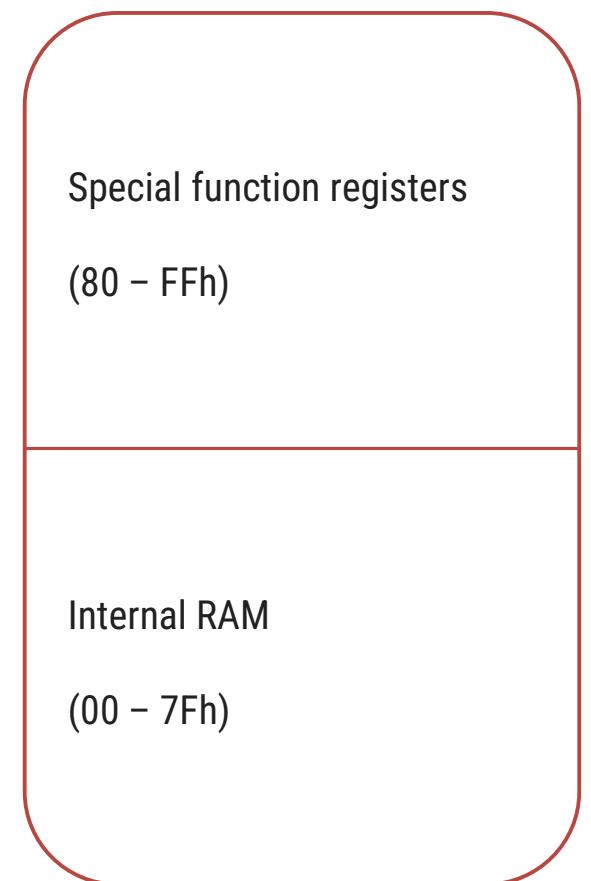
- ▶ There are two categories of memory – program memory and data memory.

## 1. Program memory

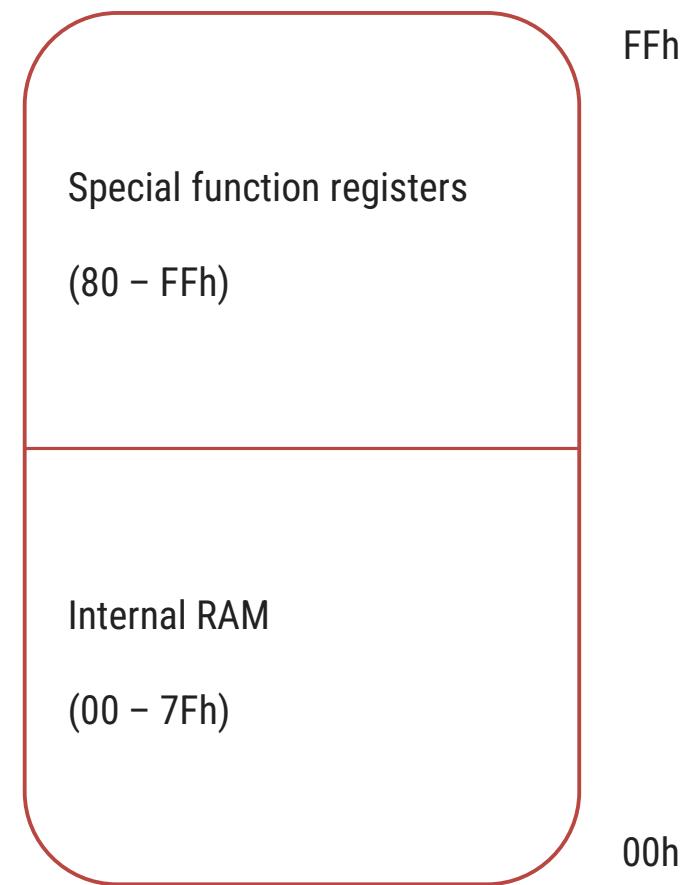
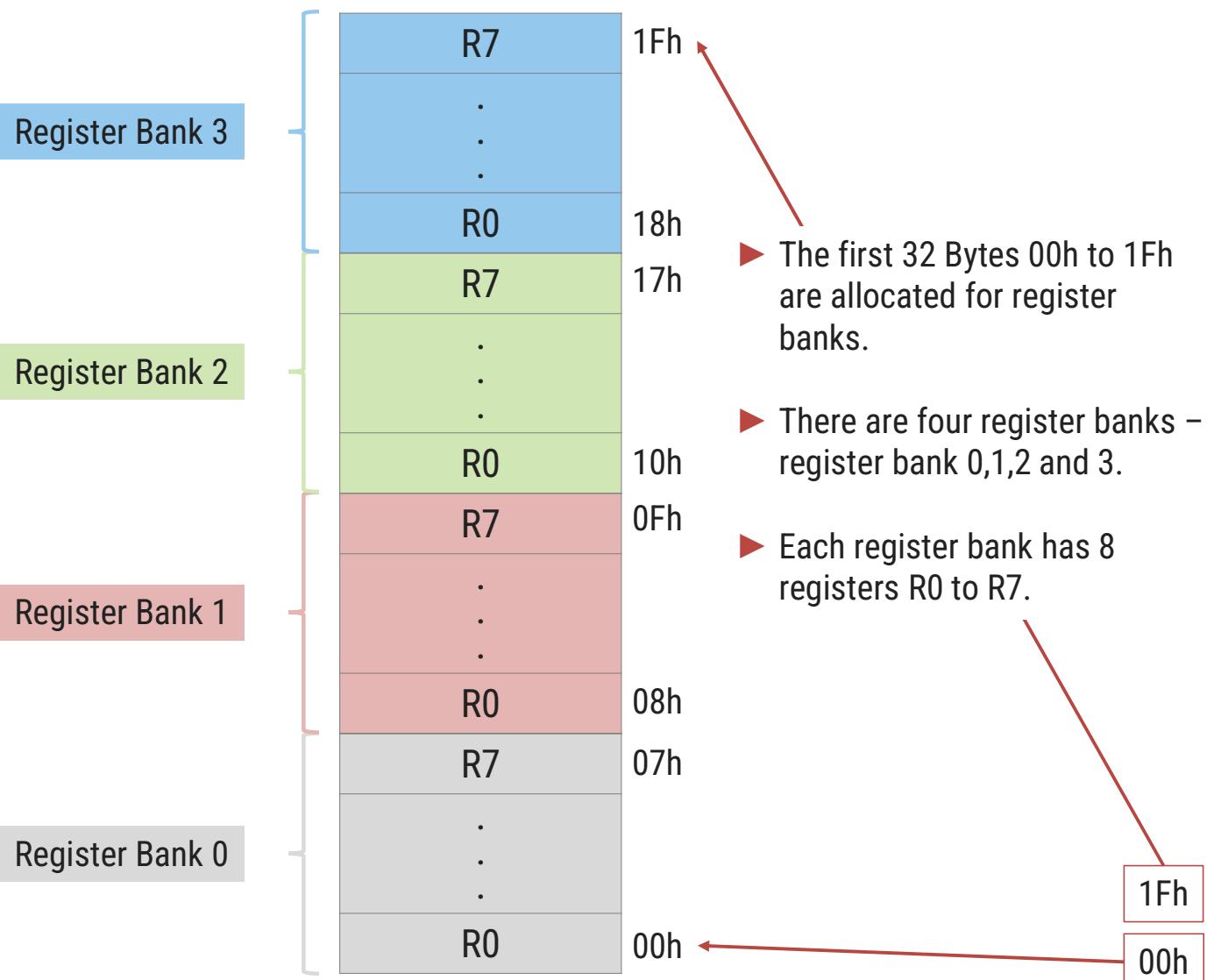
- In 8051, 4K bytes of program memory is available.
- It is normally referred as ROM and non volatile in nature.
- It is used to store following:
  1. Boot up programs
  2. Interrupt Service Routines (ISR)
  3. Macro Functions

## 2. Data memory

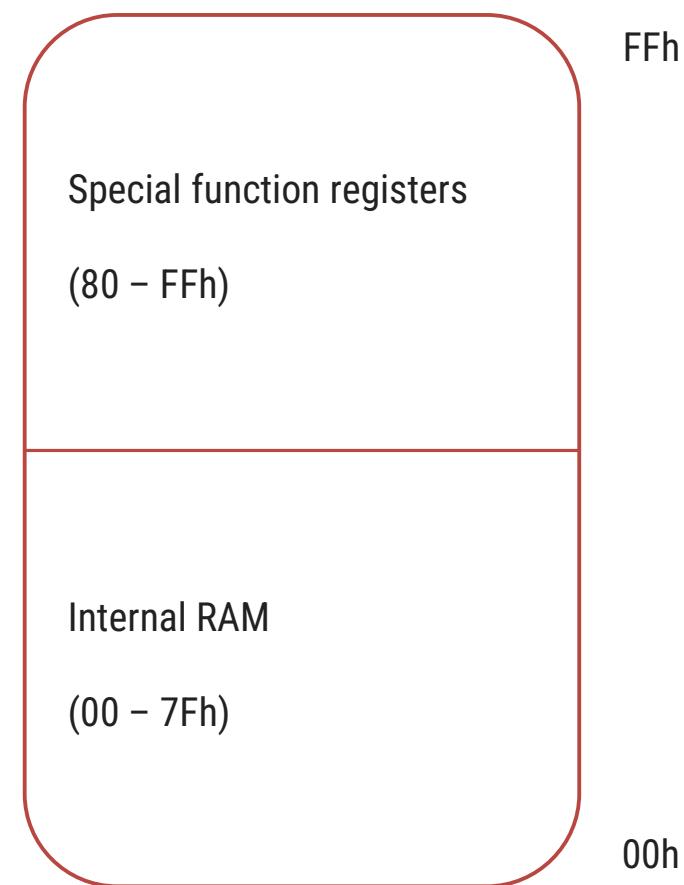
- Data memory is used to store temporary data.
- 8051 has 256 byte data memory neatly organized for various operations.
- The 256 byte data memory is divided into two parts
  1. First 128 Bytes – Internal user accessible RAM
  2. Last 128 Bytes – Special Function Registers (SFRs).



## Data Memory Organization

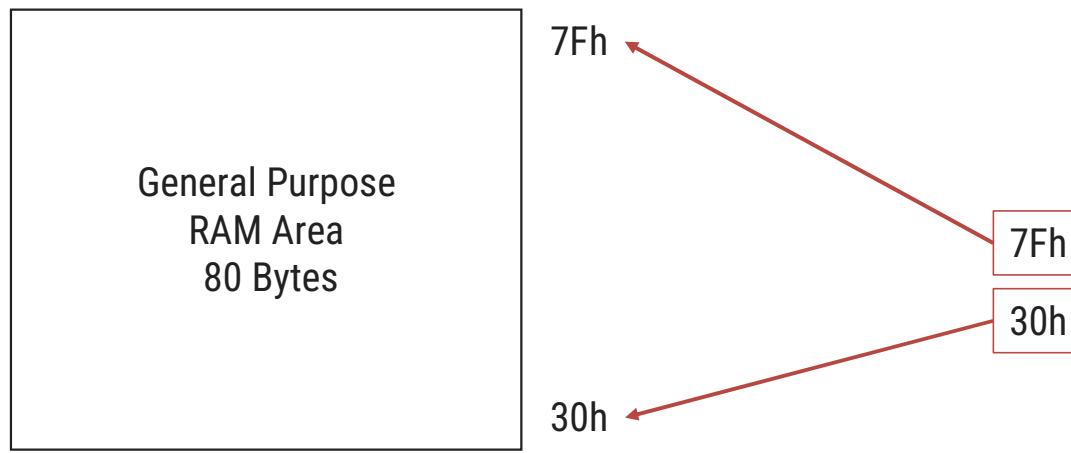


## Data Memory Organization

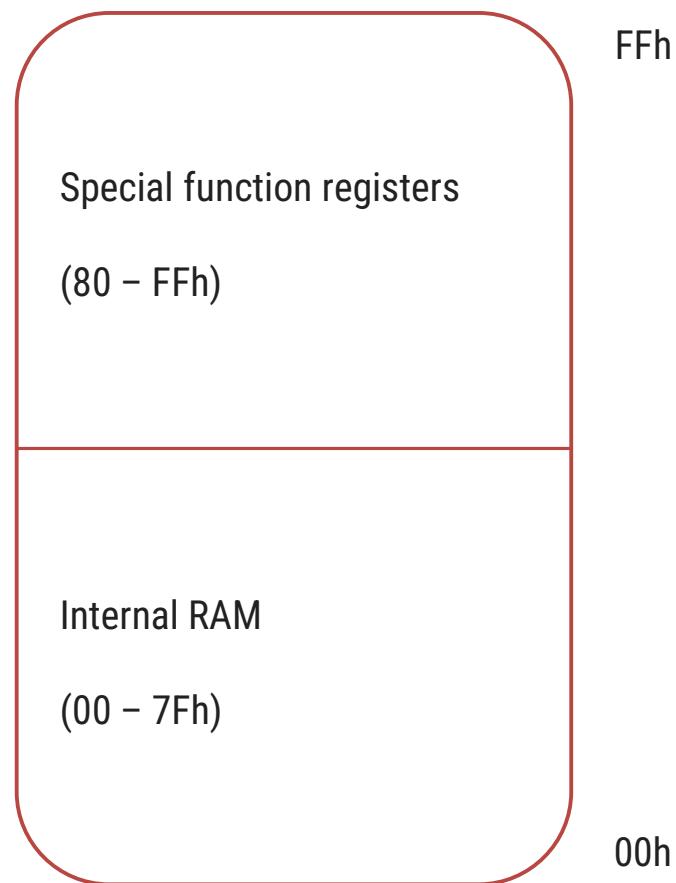


Bit Addresses	b7	b6	b5	b4	b3	b2	b1	b0	
7F								78	2Fh
77								70	2Eh
6F								68	2Dh
67								60	2Ch
5F								58	2Bh
57								50	2Ah
4F								48	29h
47								40	28h
3F								38	27h
37								30	26h
2F								28	25h
27								20	24h
1F								18	23h
17								10	22h
0F								08	21h
07								00	20h

- ▶ The last 80 bytes of Internal RAM that is 30h to 7Fh are used as General purpose RAM Area.
- ▶ Any general purpose data is stored in this area.



## Data Memory Organization



# Special Function Registers

- ▶ There are **21 SFRs** in 8051 microcontroller.
- ▶ They are important set of registers used for dedicated operations in 8051 microcontroller.
- ▶ Some of the SFRs are **bit addressable** while others are not.
- ▶ The following list explains the register and their function. The bracket is representing its address in RAM.

## 1. Accumulator – A (E0h)

- It is 8 bit register used in each Arithmetic and Logical operation.
- It is bit addressable.
- All the results of arithmetic or logical operations are stored in Accumulator
- One operand in all operations such as addition, subtraction, division and multiplication will be accumulator.

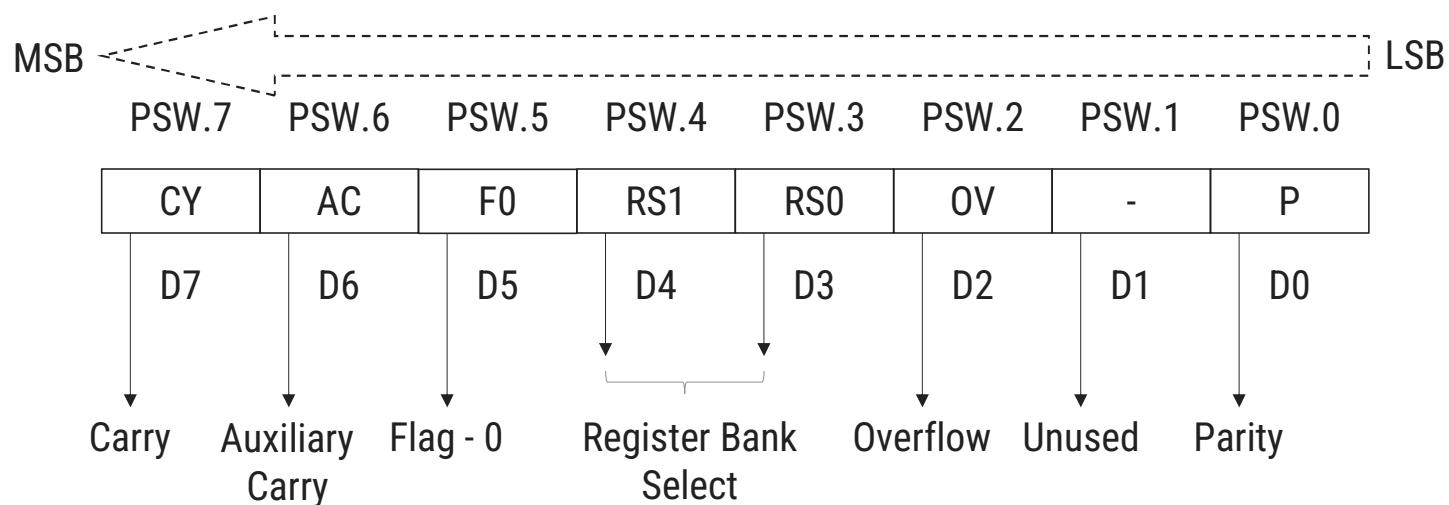
## 2. Register B (F0h)

- It is 8 bit register used in multiplication and division mainly.
- It is bit addressable.

# Special Function Registers (Cont.)

## 3. PSW Register (D0h)

- PSW (Program Status Word) is consisting various flags of 8051.
- These flags indicate whether the microcontroller works as expected or something is going wrong.
- The format of PSW register is as shown in figure.
- It is 8 bit register.
- Also PSW is bit addressable and its bits are represented by PSW.X, where X represents bit number.
- Bit 0 is LSB and bit 7 is MSB.



# Special Function Registers (Cont.)

## 3. PSW Register (D0h)

### → P – Parity Flag (PSW.0)

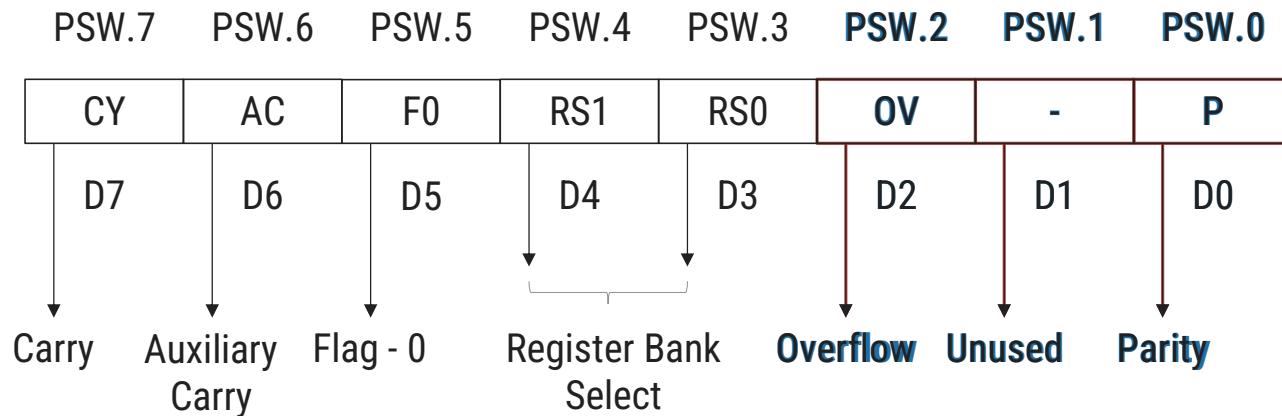
- It is D0 bit of PSW register. The parity flag is set or reset based on the result stored in accumulator. For odd number 1s in result, the flag is set otherwise it is reset.

### → PSW.1

- It is not used and reserved for future usage

### → OV – Overflow Flag (PSW.2)

- When the result of signed arithmetic operation is very large OV will be set.



# Special Function Registers (Cont.)

## 3. PSW Register (D0h)

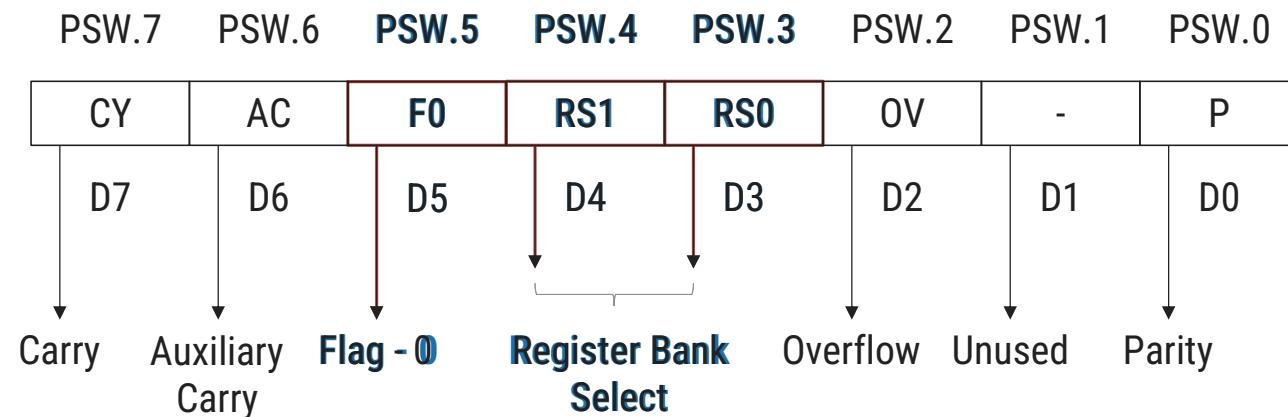
### → RS0 and RS1 – Register Bank Select (PSW.3 and PSW.4)

- The four register bank available in Internal RAM area can be selected by RS0 and RS1 bits. The table shows combination of RS1 and RS0 to select a particular register bank.

### → F0 – Flag 0 (PSW.5)

- It is a general purpose bit that can be used by a user as per requirement.

RS1	RS0	Bank Selected
0	0	Register Bank 0
0	1	Register Bank 1
1	0	Register Bank 2
1	1	Register Bank 3



# Special Function Registers (Cont.)

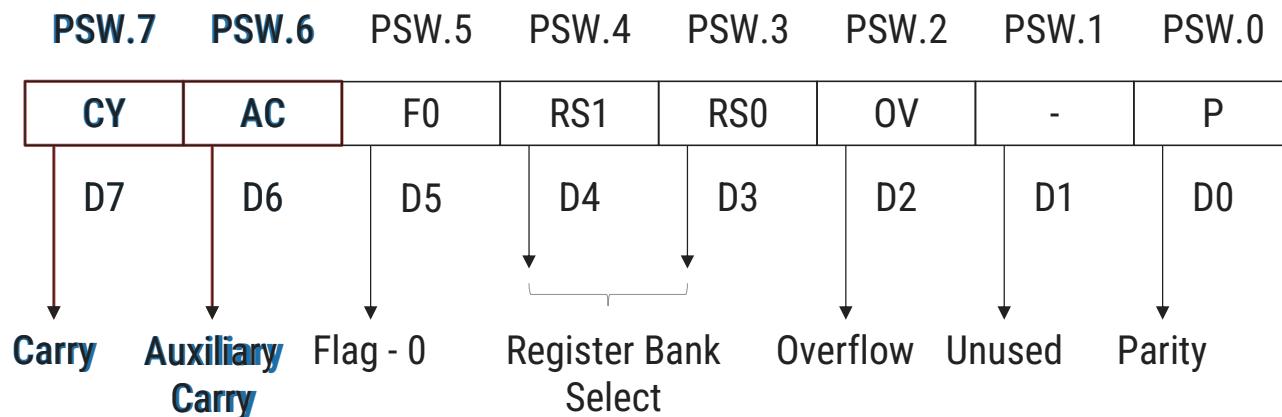
## 3. PSW Register (D0h)

### → AC – Auxiliary Carry (PSW.6)

- It is used in BCD operations. When there is a carry from D3 to D4 bit this flag is set otherwise it remains reset.

### → CY – Carry (PSW.7)

- Carry flag is set to one when the arithmetic operation has carry generated from D7 bit. It is also used in shift operations.



# Special Function Registers

► Timer and Counter registers related to timer and counter programming

4. TCON (88h) (Timer Control Register)

5. TMOD (89h) (Timer Mode Register)

6. TL0 (8Ah) (Timer0 Lower Byte)

7. TL1 (8Bh) (Timer1 Lower Byte)

8. TH0 (8Ch) (Timer0 Higher Byte)

9. TH1 (8Dh) (Timer1 Higher Byte)

► PORT register – These registers are used for PORT programming.

10. P0(80h)

11. P1(90h)

12. P2(A0h)

13. P3 (B0h)

► Interrupt programming registers

14. IE (A8h) (Interrupt Enable)

15. IP (B8h) (Interrupt Priority)

# Special Function Registers

- ▶ Power control register

16. PCON (87h) (Power Control)

- ▶ Serial Communication Registers

17. SCON (98h) (Serial Control)

18. SBUF (99h) (Serial Buffer)

- ▶ Data Pointer Registers (DPTR) – This register is combination of two 8 bit registers DPH and DPL. It is used when data is to be stored in memory pointing out a particular location.

19. DPH (83h) (Data Pointer Higher byte)

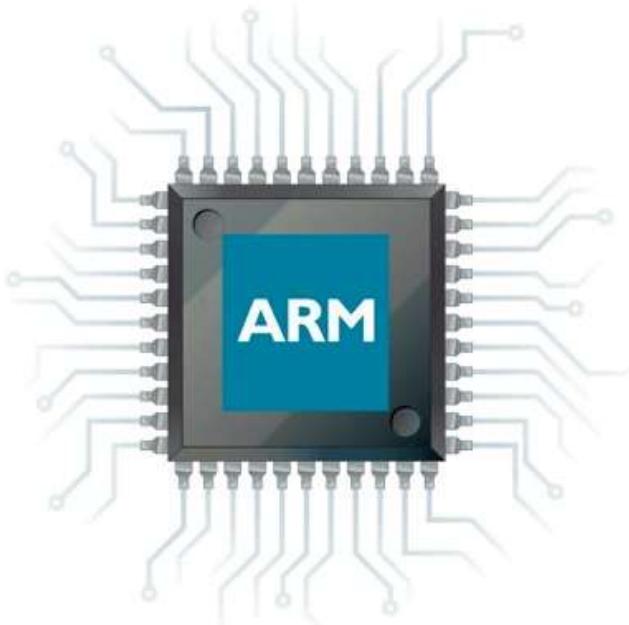
20. DPL (82h) (Data Pointer Lower byte)

- ▶ Stack Pointer – It is used while storing the data into stack memory.

21. SP (81h) (Stack Pointer)

# Advanced RISC Machine (ARM)

- ▶ An Advanced RISC Machine (ARM) is a processor which is family of CPUs based on Reduced Instruction Set Computer.
- ▶ The ARM processor is 32 bit processor developed for high-end applications that involve more complex computations and calculations.
- ▶ The ARM is based on RISC but has been enhanced for usage in embedded applications.

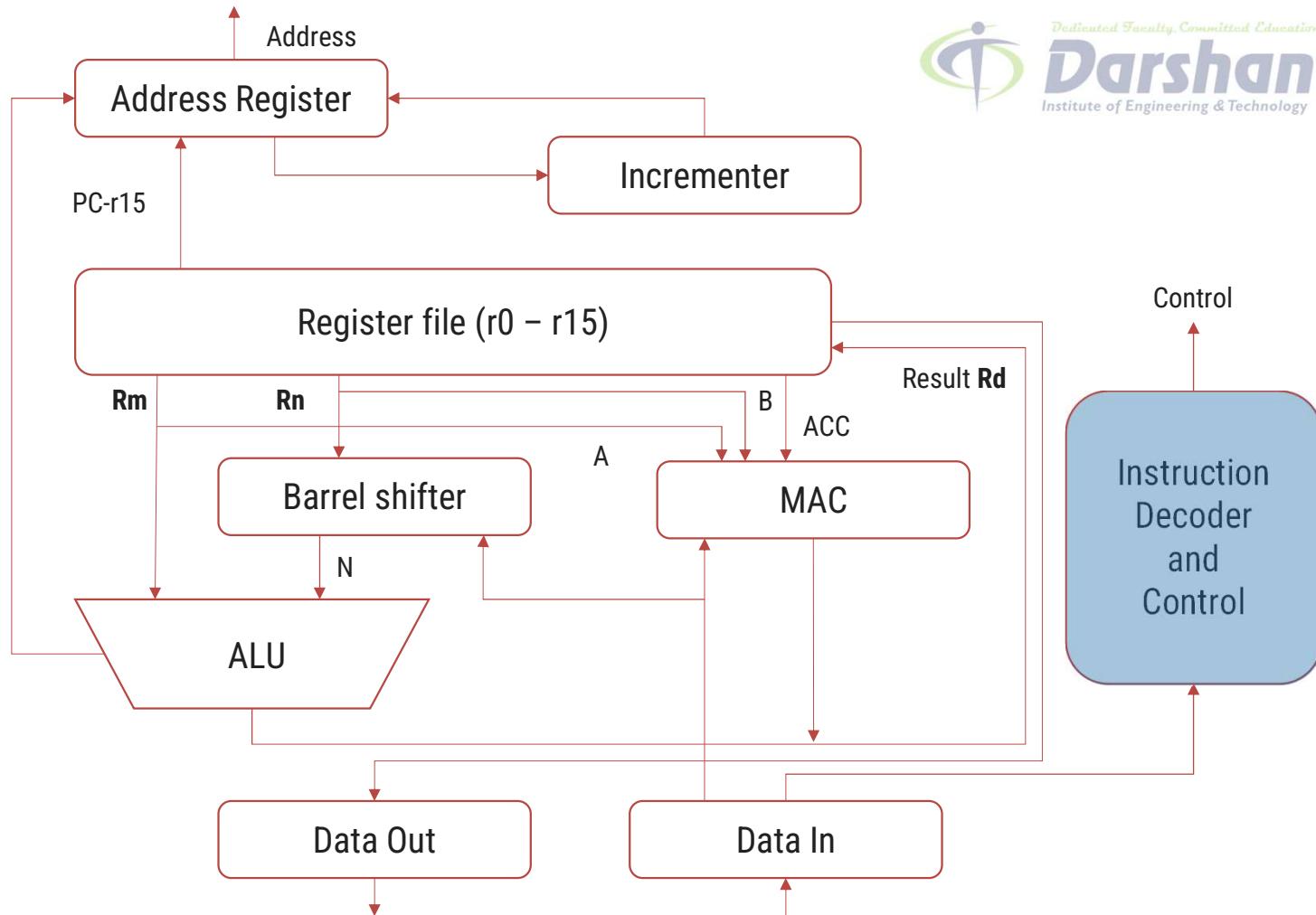


# Features of ARM

- ▶ It has large uniform register files with **load and store** architecture. The load-store architecture allows a single instruction to perform **load operation from main memory into register** and **store operation from a register into main memory**.
- ▶ ARM is 32 – bit processor with **reverse compatibility**. So it has 16-bit and 8-bit variants embedded into 32 – bit processor.
- ▶ It has very good **speed v/s power consumption ratio** and high code density.
- ▶ It has a **barrel shifter** in the data path, which can maximize hardware usage available on the chip.
- ▶ Built in **auto-increment and auto-decrement** addressing modes.
- ▶ It has **conditional execution** instructions.

# ARM Core Data Flow Model

- ▶ Figure shows ARM core flow diagram where the functional units are connected by data buses.
- ▶ The important blocks can be explained as following.
- ▶ Instruction Decoder
  - It decodes the instruction before execution.
  - There are three kinds of instruction set supported
    1. ARM Instruction set
    2. Jazelle Instruction set
    3. Thumb Instruction set



# ARM Core Data Flow Model

## ► Rm, Rn and Rd

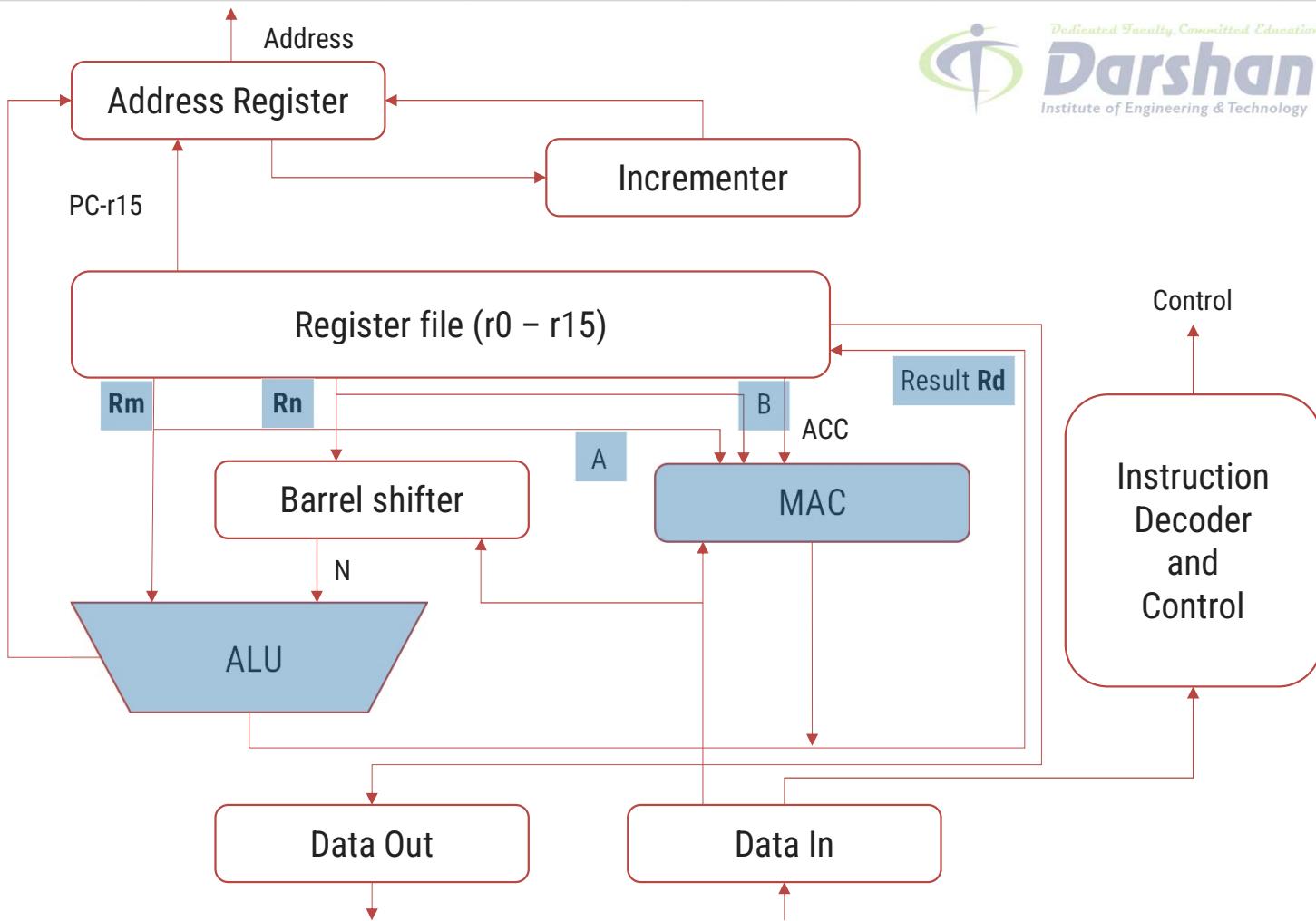
- ARM has two source registers Rm and Rn and one destination register Rd.

## ► A and B buses

- A and B buses are used to fetch the data from source register and provides for further operation.

## ► ALU and MAC

- The computation is carried out in ALU or MAC (Multiply Accumulate) unit.



# ARM Core Data Flow Model

## ▶ Barrel Shifter

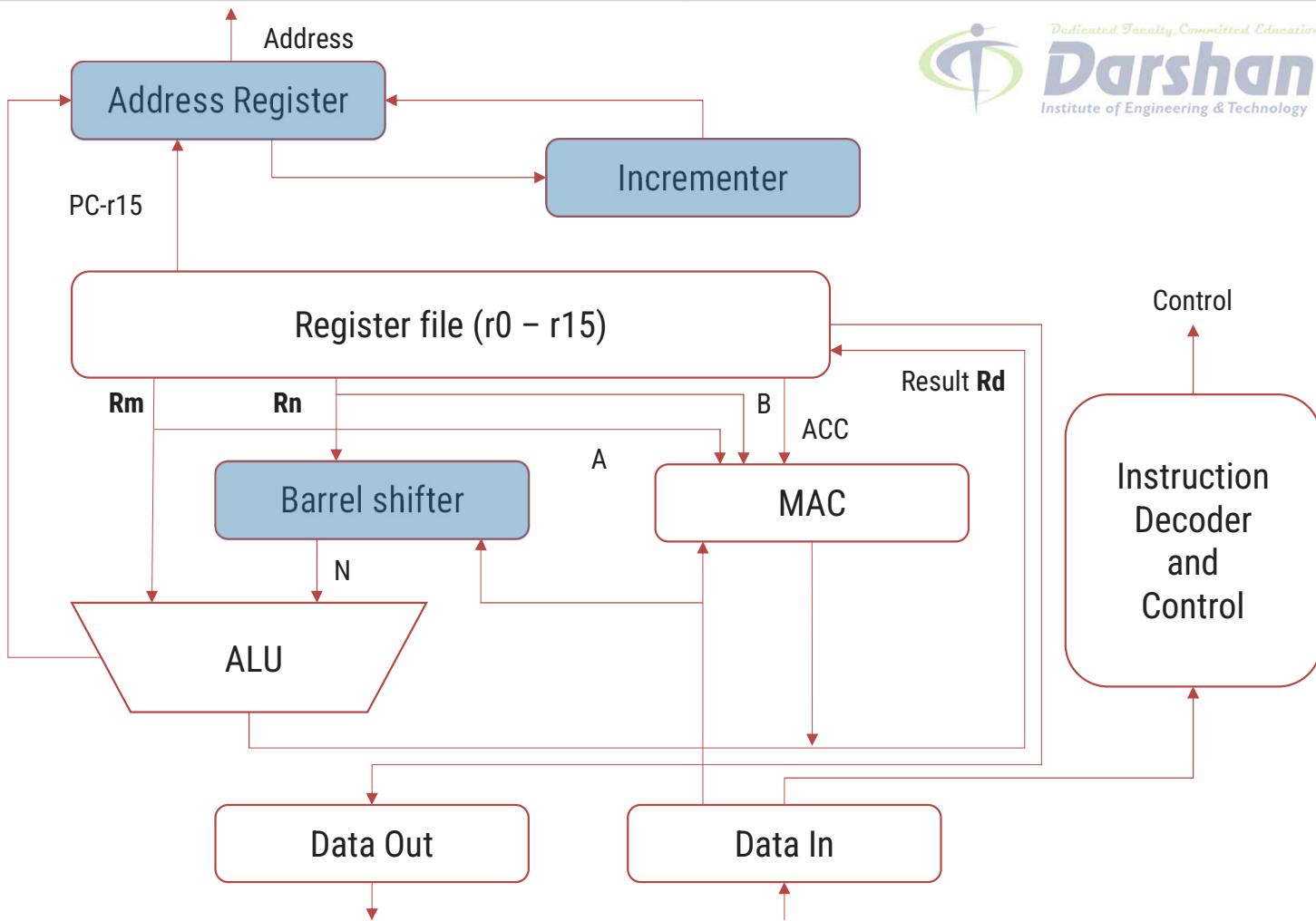
- Barrel shifter is often used for **shift** and **rotate** operation in a single clock cycle.

## ▶ Address register

- The address register is used to hold the register and the address bus will carry the current address.

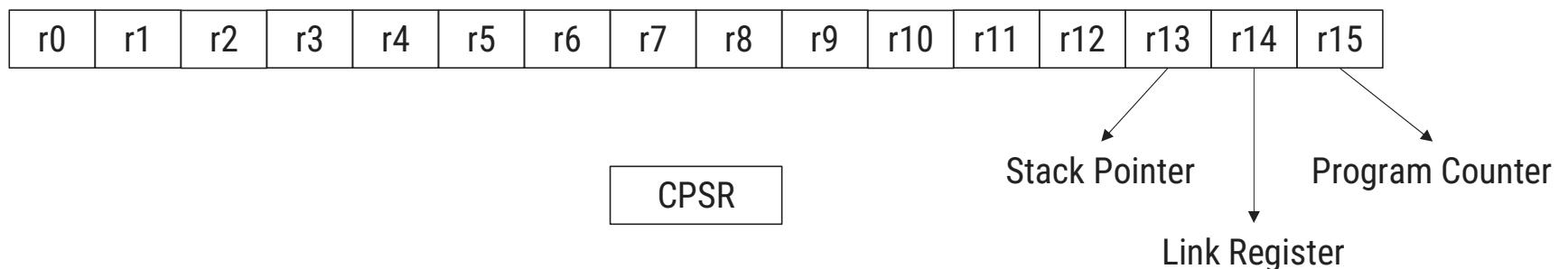
## ▶ Incrementer

- The incrementer is used for auto-increment of address and stored back in the address register.



# ARM Register Organization

- ▶ ARM has 13 general purpose registers (r0 to r12)
- ▶ All the registers are 32 bit in size.
- ▶ There are 3 SFRs in ARM
  - r13 (Stack Pointer)
  - r14 (Link Register)
  - r15 (Program Counter)
- ▶ There is one more important register in ARM – CPSR (Current Program Status Register).



# ARM Register Organization - SFRs

## 1. Stack Pointer (SP/r13)

- The stack pointer holds the address of the top of the stack.
- It is used for accessing the stack memory.

## 2. Link Register (LR/r14)

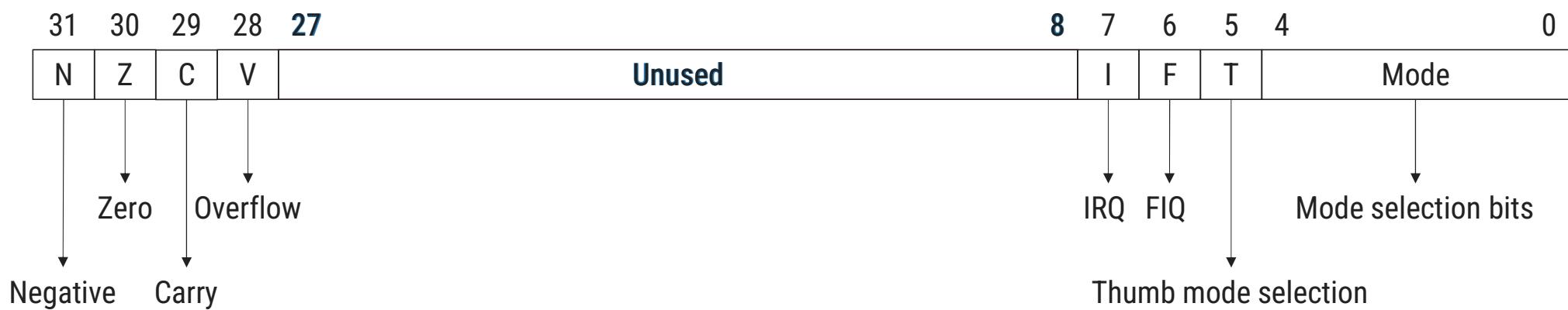
- This register stores the address of next instruction to be executed when the interrupt is generated and the processor will jump to execute the service routine or subroutine.
- After executing the subroutine, this register will return the address from where the program was interrupted.

## 3. Program Counter (PC/r15)

- The program counter is used to sequence the flow of execution of program.
- It holds the address of the next instruction to be executed.

# Current Program Status Register (CPSR)

- ▶ CPSR is the status register or flag register equivalent to PSW in 8051 but CPSR is of 32 bits.
- ▶ The format of CPSR register is shown in figure.
- ▶ Among 32 bits, 20 bits are unused or reserved for future expansion.



# Current Program Status Register (CPSR)

## ► Mode selection bits

- ARM can operate in seven different modes. Total 5 bit 0 to 4 are assigned for mode selection.

## ► T – Thumb mode selection

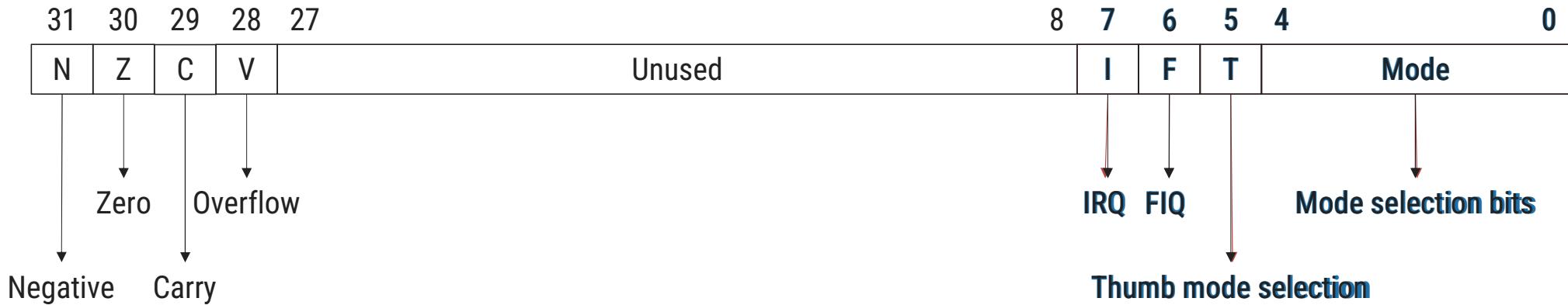
- If this bit is set to 1, ARM works in Thumb mode else ARM works in normal mode.

## ► F – FIQ Interrupt Mask

- This bit is set to zero to enable FIQ interrupts.

## ► I – IRQ Interrupt Mask

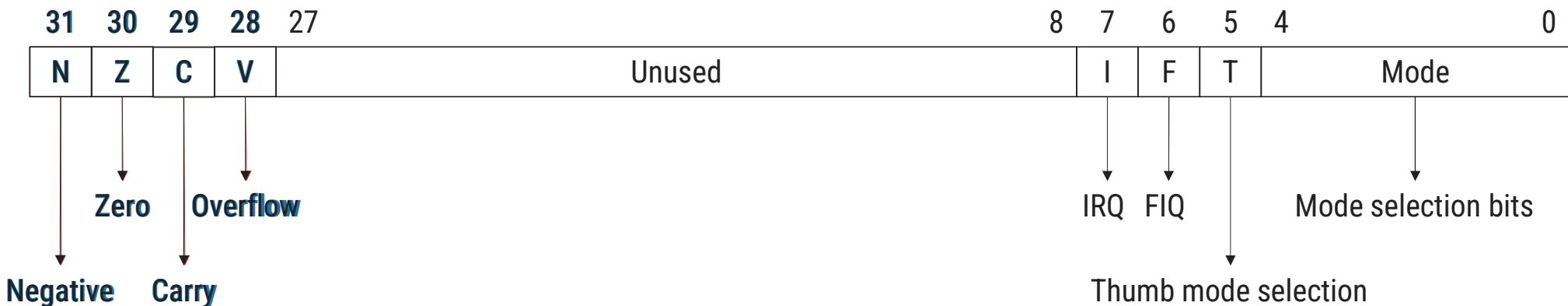
- This bit is set to zero to enable IRQ or normal interrupts.



# Current Program Status Register (CPSR)



- ▶ V – Overflow Flag
    - When result of a signed operation results in an overflow, this flag is set to 1.
  - ▶ C – Carry Flag
    - This flag is set to 1 if the result of an ALU operation generates carry.
  - ▶ Z – Zero Flag
    - This flag is set to 1 if the result of an ALU operation is zero.
  - ▶ N – Negative Flag
    - This flag is set to 1 if the result of an ALU operation is negative.



# Difference between Microprocessor and Microcontroller

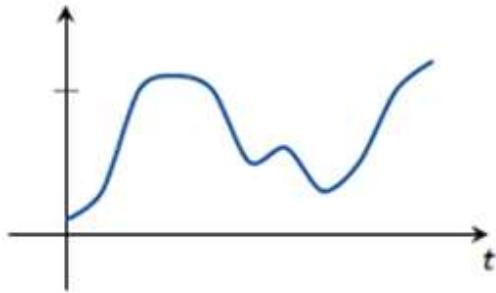
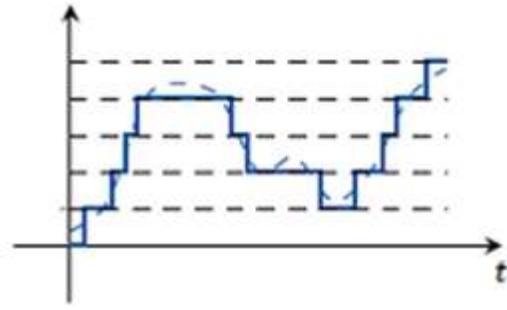
## Difference between Microprocessor and Microcontroller

	<b>Microprocessor</b>	<b>Microcontroller</b>
Application	It is used in applications of General purpose.	It is used in specific applications.
Peripherals	Peripheral devices (like memory, I/O ports, ADC/DAC etc.) are required for the operation of Microprocessor based system.	Peripheral devices are not required to operate.
Size of memory	Microprocessor based system will have large size of RAM and program memory (MB/GB).	Microcontroller based system will have smaller size of RAM and program memory (KB/MB).
Cost	The cost of the microprocessor is high compared to the microcontroller.	It is cheaper than Microprocessor.
Complexity	The structure of Microprocessor based system is complex compared to Microcontroller	The structure of Microcontroller based system is simpler.
Power Consumption	The power consumption for the microprocessor is high.	The power consumption for the microcontroller is less.



# Difference between Analog and Digital

## Difference between Analog and Digital Signal

Analog Signals	Digital Signals
Analog signals can have any possible values in the given range.	Digital signals can have only discrete values in the given range.
Examples: All physical quantities (like temperature, light intensity) measured by different analog sensors, sound waves	Examples: Count of persons entered in premises, roll numbers etc.
	



# Comparison of Flash, SRAM and EEPROM

- ▶ Flash memory is also known as program memory where the program for any microcontroller or IC is stored.
- ▶ SRAM (Static Random Access Memory) is where the program creates and manipulates variables and stores temporary data.
- ▶ EEPROM(Electronically Erasable Programmable Read Only Memory) is the space where programmer need to store long term information.



**Thank  
You**



**Prof. Tushar J. Mehta**  
Computer Engineering Department  
Darshan Institute of Engineering & Technology, Rajkot

---

✉ tushar.mehta@darshan.ac.in  
📞 +91-8866756776

Unit-3

# Protocols for IoT



**Prof. Tushar J. Mehta**  
Computer Engineering Department  
Darshan Institute of Engineering & Technology, Rajkot

---

✉ tushar.mehta@darshan.ac.in  
📞 8866756776

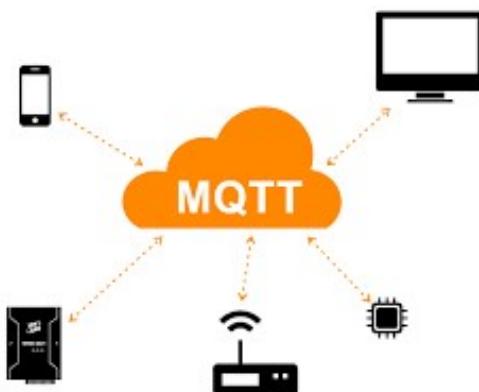


# Messaging Protocols

Section - 1

# Messaging Protocols in IoT

- ▶ Messaging protocols are very important for the transfer of data in terms of messages.
- ▶ They are useful for send/receive of a message to/from the cloud in IoT applications.
- ▶ In the section, two messaging protocols are discussed.
  1. Message Queuing Telemetry Transport (MQTT)
  2. Constrained Application Protocol (CoAP)



CoAP  
Constrained Application  
Protocol



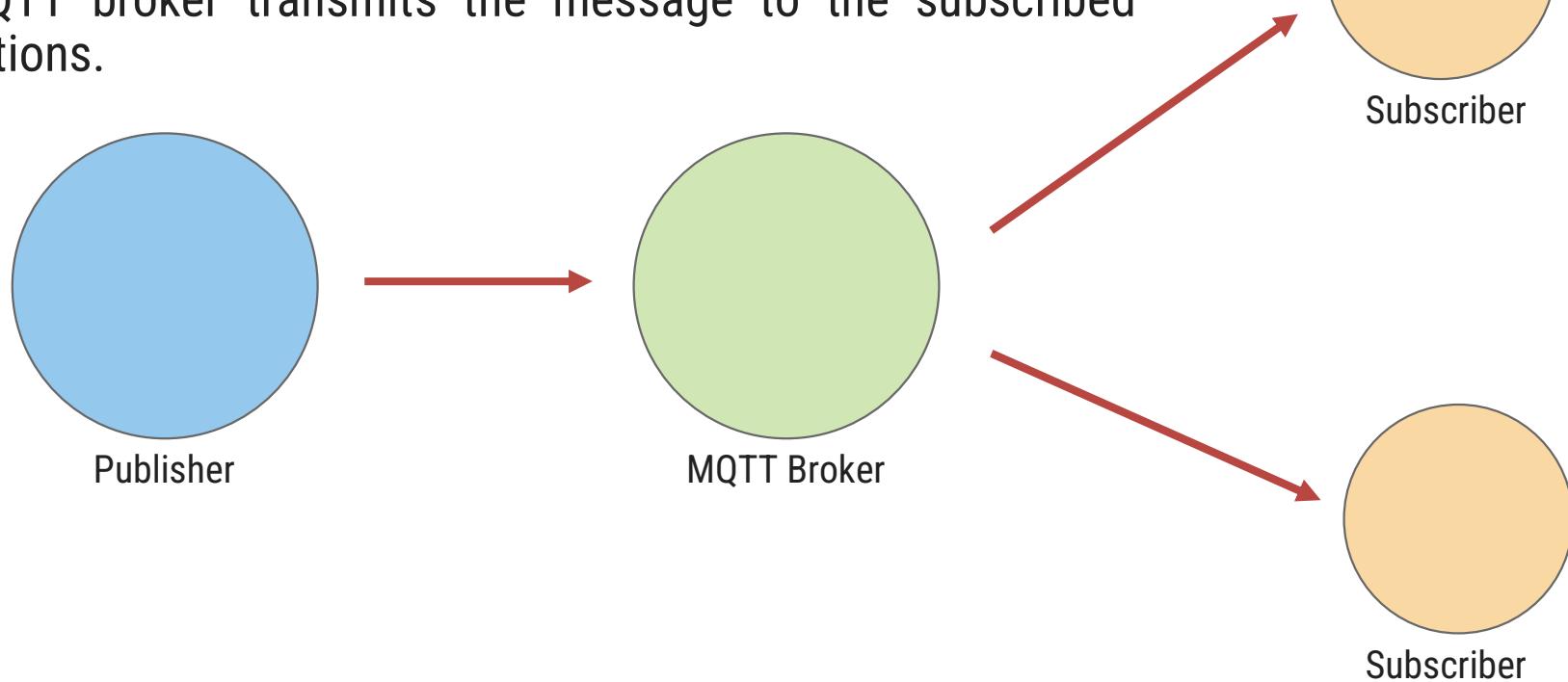
# Message Queuing Telemetry Transport (MQTT)

- ▶ As we know, IoT has one of the biggest challenges of resource constraints, the lightweight protocol is more preferred.
- ▶ MQTT is widely used in IoT applications as it is a lightweight protocol.
- ▶ Here, lightweight means, it can work with minimal resources and does not require any specific hardware architecture or additional resources.



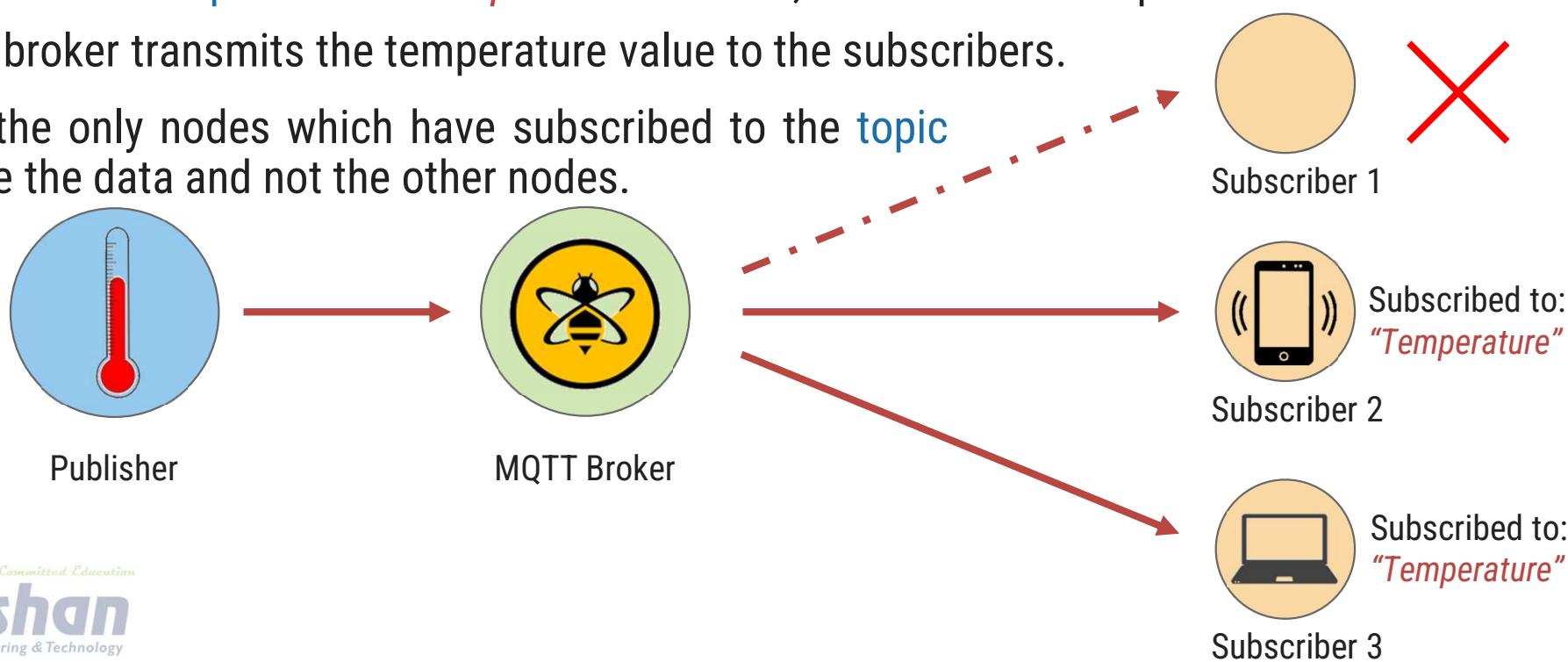
# MQTT

- ▶ MQTT works with “publish – subscribe” pattern.
  - ▶ The MQTT working can be explained as follows.
1. The publisher (node) sends a message to the MQTT broker.
  2. The MQTT broker transmits the message to the subscribed destinations.



# MQTT - Example

- ▶ Suppose a temperature sensor is connected to any system and we want to monitor that temperature.
- ▶ The controller connected to the temperature sensor publishes the temperature value to the MQTT broker with a **topic** name: "*Temperature*". Hence, it is considered a publisher.
- ▶ The MQTT broker transmits the temperature value to the subscribers.
- ▶ Note that the only nodes which have subscribed to the **topic** will capture the data and not the other nodes.



# MQTT - Example

- ▶ Can publisher and subscriber interchange their role?
- ▶ Let us take an example of a **Driverless car**.
- ▶ A GPS sensor is connected to the car which gives the current location of the car.
- ▶ The system in car, publishes the location to MQTT broker with topic name – **CurrentLocation**.
- ▶ The server/computer subscribes to the topic **CurrentLocation** and receives the current location of the car.



Publisher: **CurrentLocation**

MQTT Broker

Subscriber: **CurrentLocation**

# MQTT - Example

- ▶ Based on the currently selected destination, the server/computer computes the direction of the car.
- ▶ This data is published to an MQTT broker with a topic named – **Direction**.
- ▶ To get the command from the server, the system in the car has to subscribe to the topic named **Direction**.
- ▶ According to the command received from the server, the car will run in a particular direction.



Publisher: **CurrentLocation**  
Subscriber: **Direction**

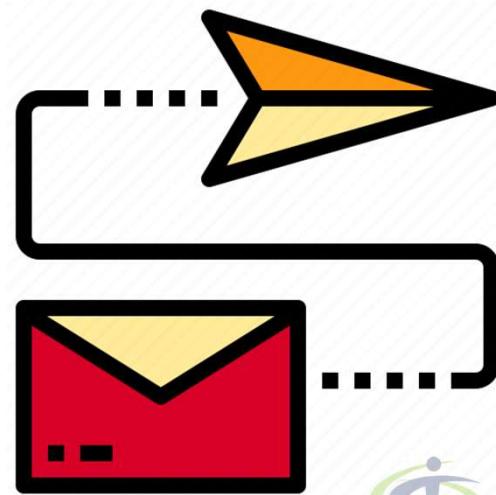
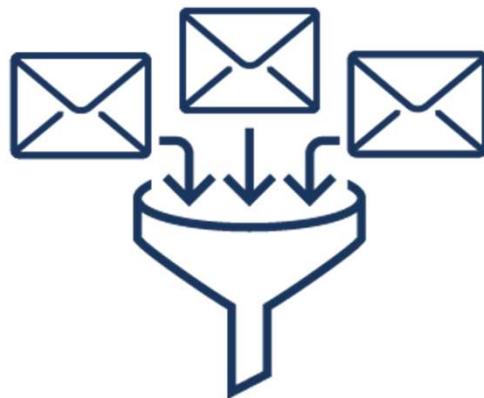
MQTT Broker

Subscriber: **CurrentLocation**  
Publisher: **Direction**

# MQTT - Working

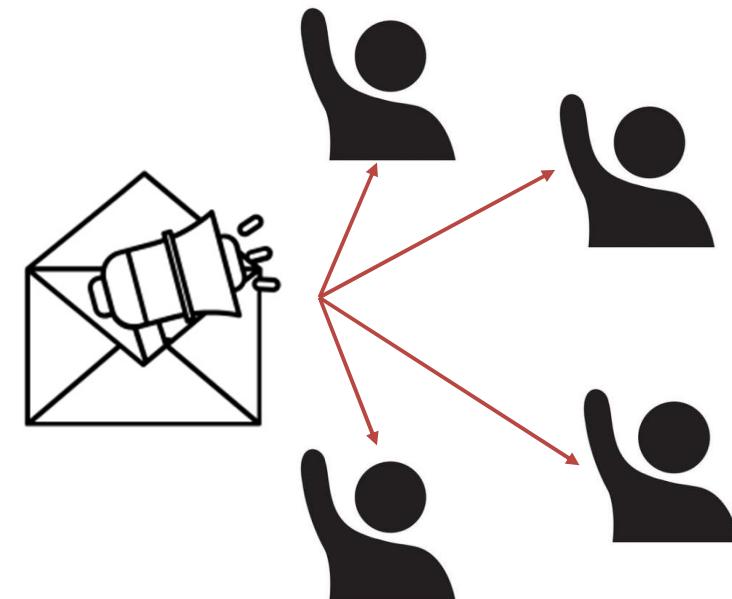
- ▶ In MQTT, clients do not have any address like a typical networking scheme.
- ▶ The broker filters the messages based on the subscribed topics.
- ▶ The messages will then be circulated to respective subscribers.
- ▶ The topic name can be anything in string format.
- ▶ The MQTT working can be explained with an example of the television broadcast.

~~Client\_Address~~



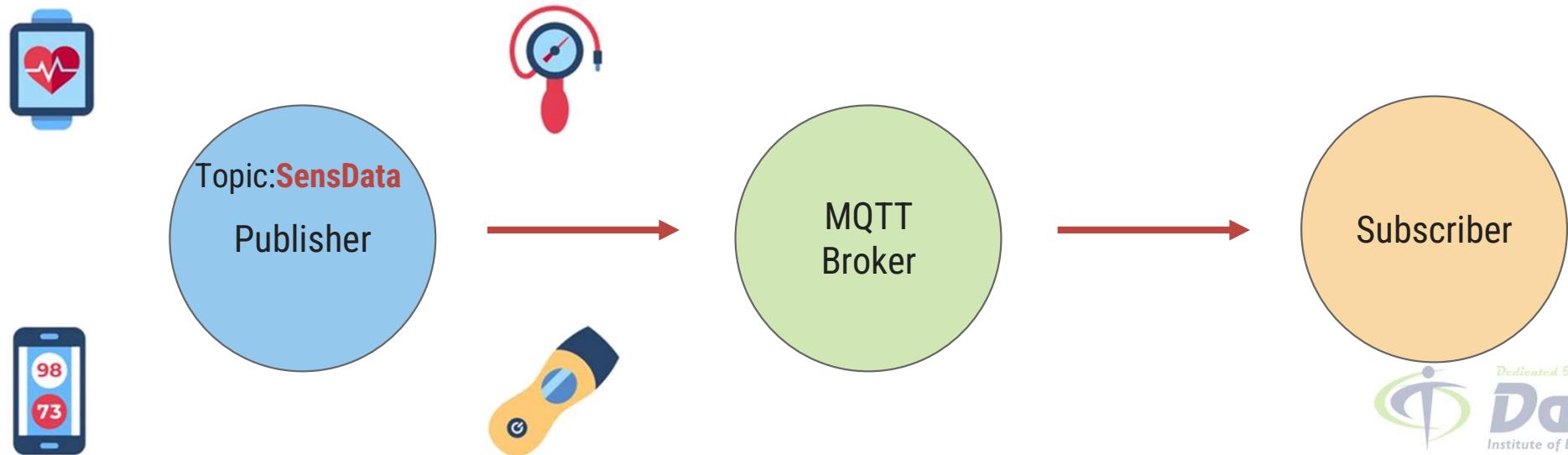
# MQTT - Working

- ▶ Television broadcaster station broadcasts all the channels.
- ▶ The viewers subscribe to their favorite channels only.
- ▶ Similarly, in MQTT, the publisher node publishes data with the topic name and interested subscribers subscribe to the topic.
- ▶ Note that there can be multiple subscribers of a same topic as well as a single subscriber can subscribe to multiple topics.



# MQTT – Node(s)

- ▶ Node(s) in MQTT collects the information from sensors or other i/p devices in case of the publisher.
- ▶ Connects it to the messaging server known as the MQTT broker.
- ▶ A specific string – Topic is used to publish the message and let other nodes understand the information. These nodes are considered subscribers.
- ▶ Any node can be publisher or subscriber and node is also referred to as client.



# QoS in MQTT

- ▶ MQTT supports different QoS (Quality of Service) levels.
- ▶ There are 3 layers of QoS supported by MQTT.
  1. Level 0 = At most once (Best effort, No Acknowledgement)
  2. Level 1 = At least once (Acknowledged, retransmitted if ack not received)
  3. Level 2 = Exactly once (Requested to send, clear to send)

# MQTT Publisher – Code Implementation

- ▶ The code for MQTT publisher is as shown in below.

MQTT\_publisher.ino

```
1 #include <ESP8266WiFi.h>
2 #include <PubSubClient.h>
3
4 const char* ssid = "OnePlusTJ";
5 const char* pwd = "Tjm12347";
6 const char* mqtt_server =
    "broker.mqtt-dashboard.com";
7
8 WiFiClient espClient;
9 PubSubClient client(espClient);
10 int value = 0;
11 unsigned long lastMsg = 0;
12 #define MSG_BUFFER_SIZE (50)
13 char msg[MSG_BUFFER_SIZE];
```

Including the “**ESP8266WiFi**” library for enabling Wi-Fi connection with protocols.

Including the “**PubSubClient**” library for use of MQTT protocol and their functions.

Defining parameters for Wi-Fi connection.

Defining MQTT server/broker for publishing the message on particular topic.

# MQTT Publisher – Code Implementation

MQTT\_publisher.ino

```
14 void setup_wifi() {  
15     delay(10);  
16     Serial.println();  
17     Serial.print("Connecting to ");  
18     Serial.println(ssid);  
19     WiFi.mode(WIFI_STA);  
20     WiFi.begin(ssid, pwd);  
21  
22     while(WiFi.status()!=WL_CONNECTED)  
23     {  
24         delay(500);  
25         Serial.print(".");  
26     }  
27     Serial.println("WiFi connected");  
28     Serial.println("IP address: ");  
29     Serial.println(WiFi.localIP());  
30 }
```

Initializing Wi-Fi module as station mode.

Connecting Wi-Fi module with given SSID and Password

Try until Wi-Fi is not connected to given SSID with delay of half second.

# MQTT Publisher – Code Implementation

MQTT\_publisher.ino

```
31 void setup() {  
32   Serial.begin(115200);  
33   setup_wifi();  
34   client.setServer(mqtt_server,1883);  
35   while (!client.connected()) {  
36     Serial.print("Attempting MQTT");  
37     String clientId="ESPClient1234";  
38     if (client.connect(  
39       clientId.c_str())) {  
40       Serial.println("connected");  
41       client.publish("darshan/6thsem/  
42         /ce", "You are in Darshan");  
43     }  
  }
```

Connects the client to the MQTT Server stored in *mqtt\_server* with port number 1883.

Execute the syntax in loop until ESP is not connected to MQTT broker.

Defining a *clientId* for MQTT server/broker.

Returns the pointer of string *clientId* to the array.

Publishing announcement message after connecting to the MQTT broker with Topic name : *darshan/6thsem/ce*

# MQTT Publisher – Code Implementation

MQTT\_publisher.ino

```
44 void loop() {  
45  
46     client.loop();  
47     unsigned long now = millis();  
48     if (now - lastMsg > 2000)  
49     {  
50         lastMsg = now;  
51         ++value;  
52         sprintf (msg, MSG_BUFFER_SIZE,  
53                    "hello world #%ld", value);  
54         Serial.print("Publish message:");  
55         Serial.println(msg);  
56         client.publish("darshan/6thsem/  
57                         ce", msg);  
58     }  
59 }
```

This function allows client to process incoming messages, publish data and refresh the connection.

Creating a string in *msg* with given buffer size and assigning the value “hello world” with concatenation of integer named *value*.

Publishes the *msg* in the given topic name: *darshan/6thsem/ce*

# MQTT Subscriber – Code Implementation

- ▶ The code for MQTT subscriber is as shown in below.

MQTT\_subscriber.ino

```
1 #include <ESP8266WiFi.h>
2 #include <PubSubClient.h>
3
4 const char* ssid = "OnePlusTJ";
5 const char* pwd = "Tjm12347";
6 const char* mqtt_server =
    "broker.mqtt-dashboard.com";
7
8 WiFiClient espClient;
9 PubSubClient client(espClient);
```

Including the “**ESP8266WiFi**” library for enabling Wi-Fi connection with protocols.

Including the “**PubSubClient**” library for use of MQTT protocol and their functions.

Defining parameters for Wi-Fi connection.

Defining MQTT server/broker for publishing the message on particular topic.

# MQTT Subscriber – Code Implementation

MQTT\_subscriber.ino

```
10 void setup_wifi() {  
11     delay(10);  
12     Serial.println();  
13     Serial.print("Connecting to ");  
14     Serial.println(ssid);  
15     WiFi.mode(WIFI_STA);  
16     WiFi.begin(ssid, pwd);  
17  
18     while(WiFi.status()!=WL_CONNECTED)  
19     {  
20         delay(500);  
21         Serial.print(".");  
22     }  
23     Serial.println("WiFi connected");  
24     Serial.println("IP address: ");  
25     Serial.println(WiFi.localIP());  
26 }
```

Initializing Wi-Fi module as station mode.

Connecting Wi-Fi module with given SSID and Password

Try until Wi-Fi is not connected to given SSID with delay of half second.

# MQTT Subscriber – Code Implementation

MQTT\_subscriber.ino

```
27 void callback(char* topic,
28   byte* payload, unsigned int length)
29 {
30   Serial.print("Message arrived [");
31   Serial.print(topic);
32   Serial.print("] ");
33   for (int i = 0; i < length; i++)
34   {
35     Serial.print((char)payload[i]);
36   }
37 }
```

A **callback** function for receiving messages every time when *client.loop()* executes.

This function accepts the arguments *char\** of *topic*, *char\** of *payload* and the *length of payload*.

Executing the for loop until all the characters of the payload are printed.

Typecasting of *payload[i]* into equivalent character and then printing on serial monitor.

# MQTT Subscriber – Code Implementation

MQTT\_subscriber.ino

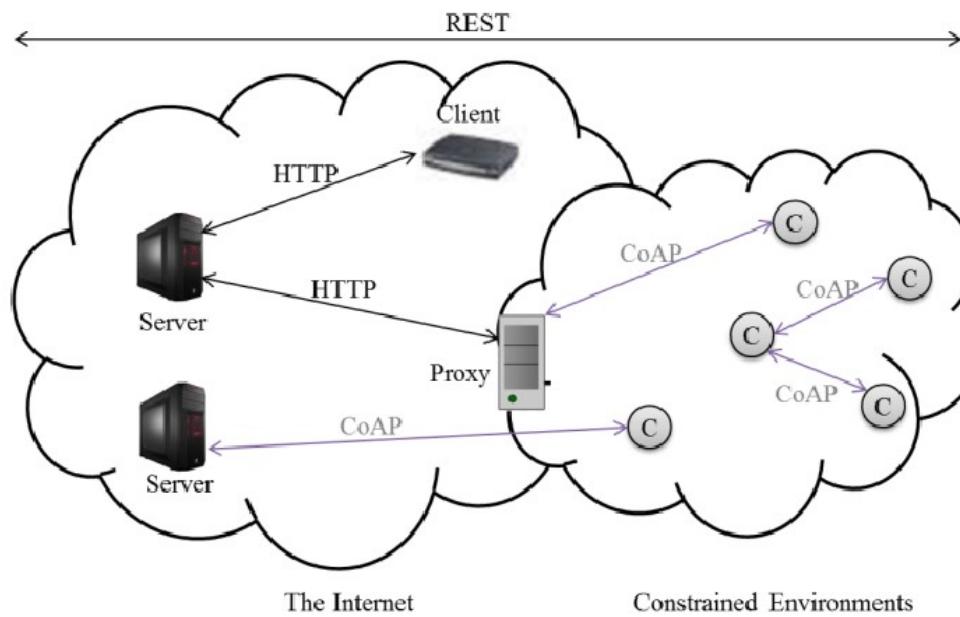
```
38 void setup() {  
39   Serial.begin(115200);  
40   setup_wifi();  
41   client.setServer(mqtt_server,1883);  
42   client.setCallback(callback); → This is built-in function client.setCallback() in  
43   while (!client.connected()) {  
44     Serial.print("Attempting MQTT");  
45     String clientId="ESPClient1234";  
46     if (client.connect(  
47           clientId.c_str())) {  
48       Serial.println("connected");  
49       client.subscribe("darshan/Msg"); → Subscribing to the Topic named "darshan/Msg"  
50     }  
51   }  
52 }  
53 void loop() {  
54   client.loop(); }
```

This is built-in function *client.setCallback()* in *PubSubClient* library and it calls a function *callback()* every time when *client.loop()* function is executed.

Subscribing to the Topic named “darshan/Msg”

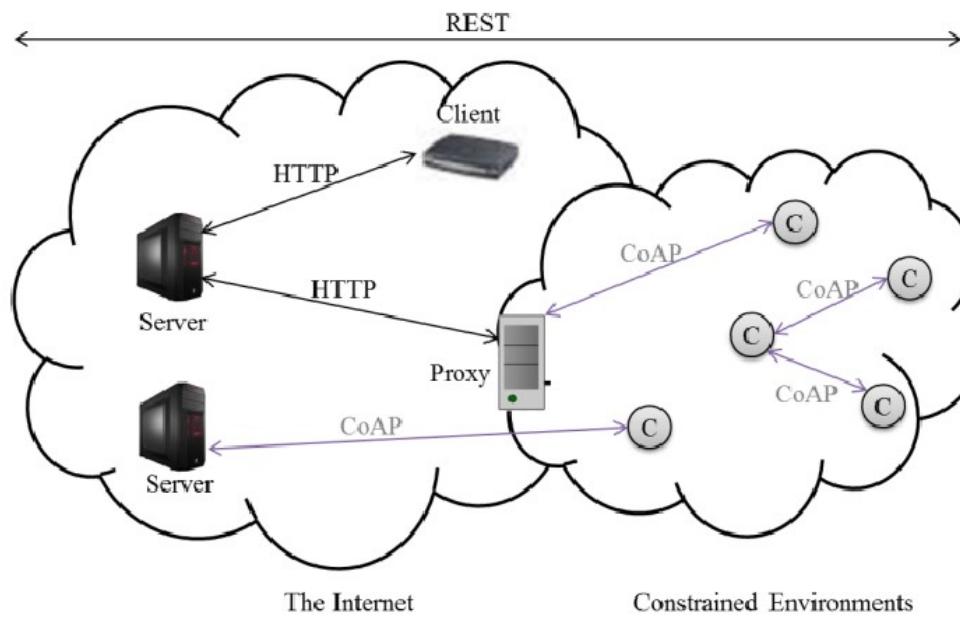
# Constrained Application Protocol (CoAP)

- ▶ CoAP is designed by IETF (Internet Engineer Task Force) to work in constrained environment.
- ▶ It is a one to one communication protocol.
- ▶ CoAP is also light weight like MQTT protocol and it uses less resources than HTTP.
- ▶ HTTP runs over TCP and is connection oriented while CoAP runs over UDP and it is connection less.



# CoAP Architecture

- ▶ CoAP is based on the RESTful architecture (**R**epresentational **S**tate **T**ransfer).
- ▶ REST approach ensures the secure, fault tolerant and scalable system.
- ▶ The CoAP optimizes the length of the datagram.
- ▶ CoAP is connectionless protocol and it requires retransmission support.

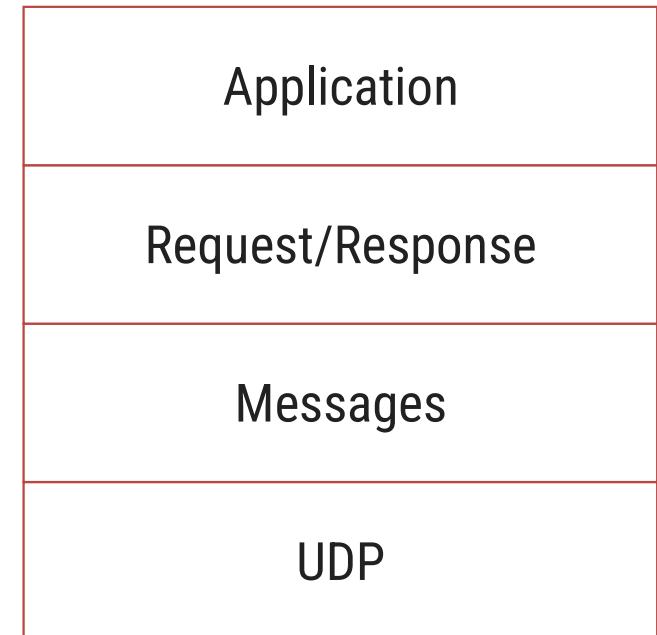


# CoAP Layers

- ▶ CoAP has four layers:

1. UDP
2. Messages
3. Request-Response
4. Application

- ▶ The message layer is designed to deal with UDP and asynchronous switching.
- ▶ The request/response layer concerns communication method and deal with request/response messages.
- ▶ In the request/response layer, clients may use GET/PUT/DELETE methods to transmit the message.

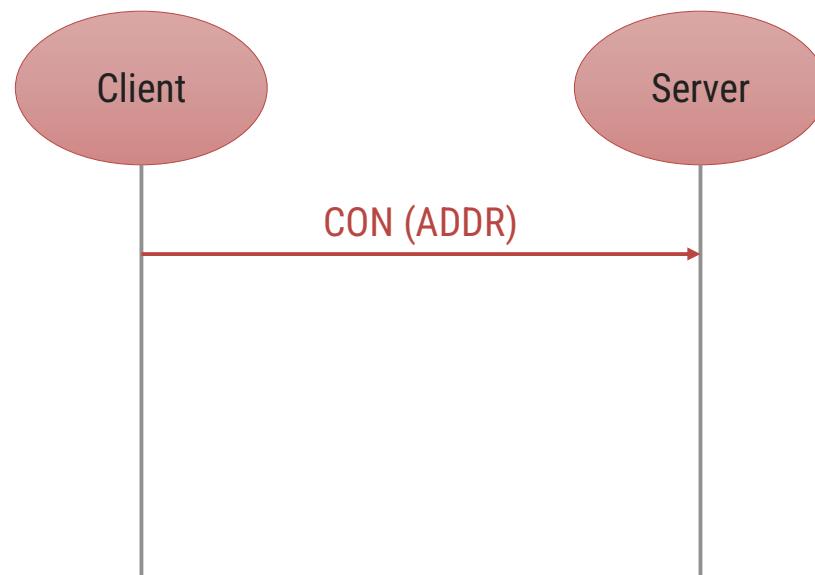


# Message Layer in CoAP

- ▶ CoAP message layer supports four types of messages:

## 1. Confirmable – Reliable Messaging (CON):

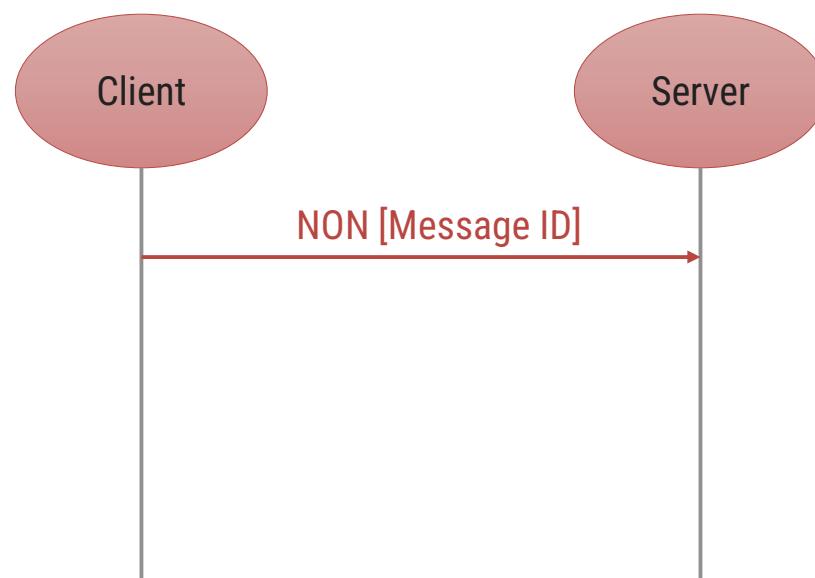
- This is reliable approach because the retransmission of message occurs until the acknowledgement of the same message ID is received.
- If there is a timeout or fail of acknowledgement, the RST message will be sent from the server as a response.
- Hence, the client will retransmit the message, this resolves the retransmission and it is considered a reliable approach.



# Message Layer in CoAP (Contd.)

## 2. Non-confirmable – Non-reliable Messaging (NON):

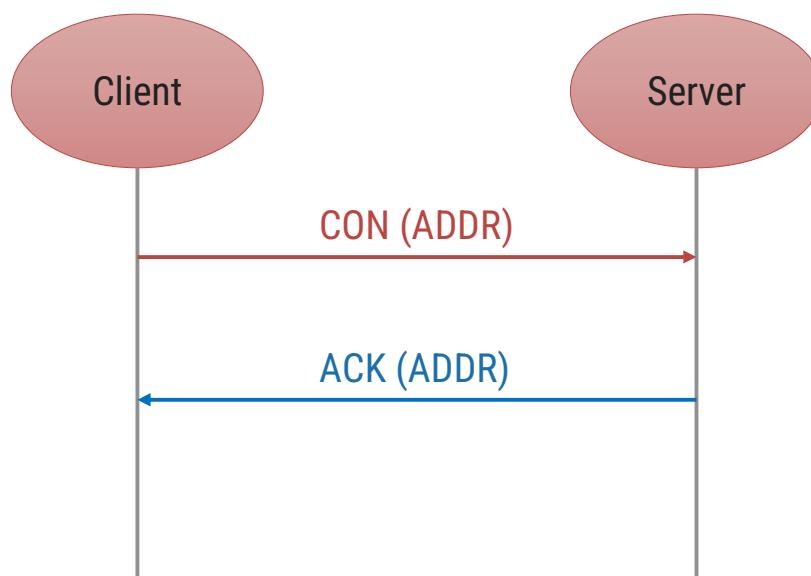
- In this message transmission style, there is no reliability ensured.
- The acknowledgement is not issued by the server.
- The message has ID for supervision purpose, if the message is not processed by the server it transmits the RST message.



# Message Layer in CoAP (Contd.)

### 3. Acknowledgement (ACK):

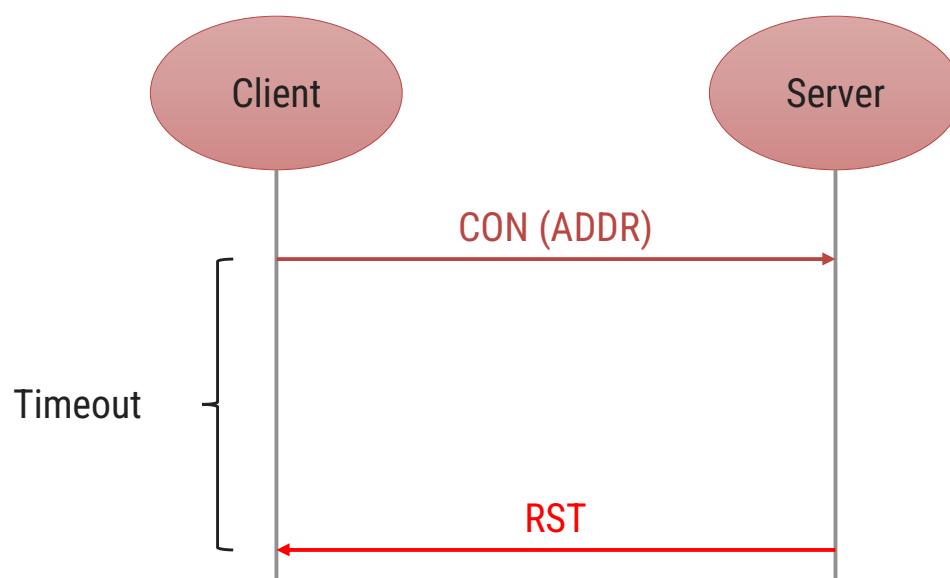
- In this messaging, the traditional acknowledgement message sent as usual protocol.
- We can compare this with regular handshaking scheme.
- The handshake is automated process that establishes a link for communication before actual data transfer begins.



# Message Layer in CoAP (Contd.)

## 4. Reset (RST):

- The receiver is expecting the message from sender of a particular message ID.
- If no message is processed or received before specific amount of time (called timeout), the message called RESET is transmitted from receiver.
- This message informs the sender that there is a trouble in transmission of messages.

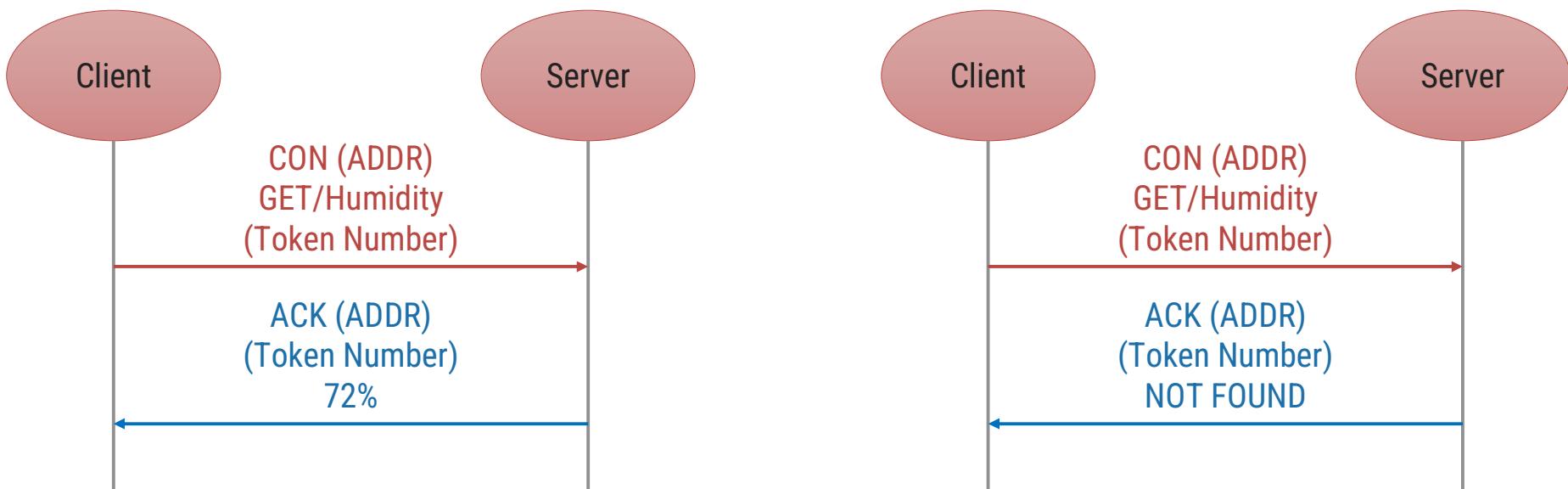


# Request-Response Layer in CoAP

- ▶ There are three modes in CoAP request-response layer.

## 1. Piggy-Backed:

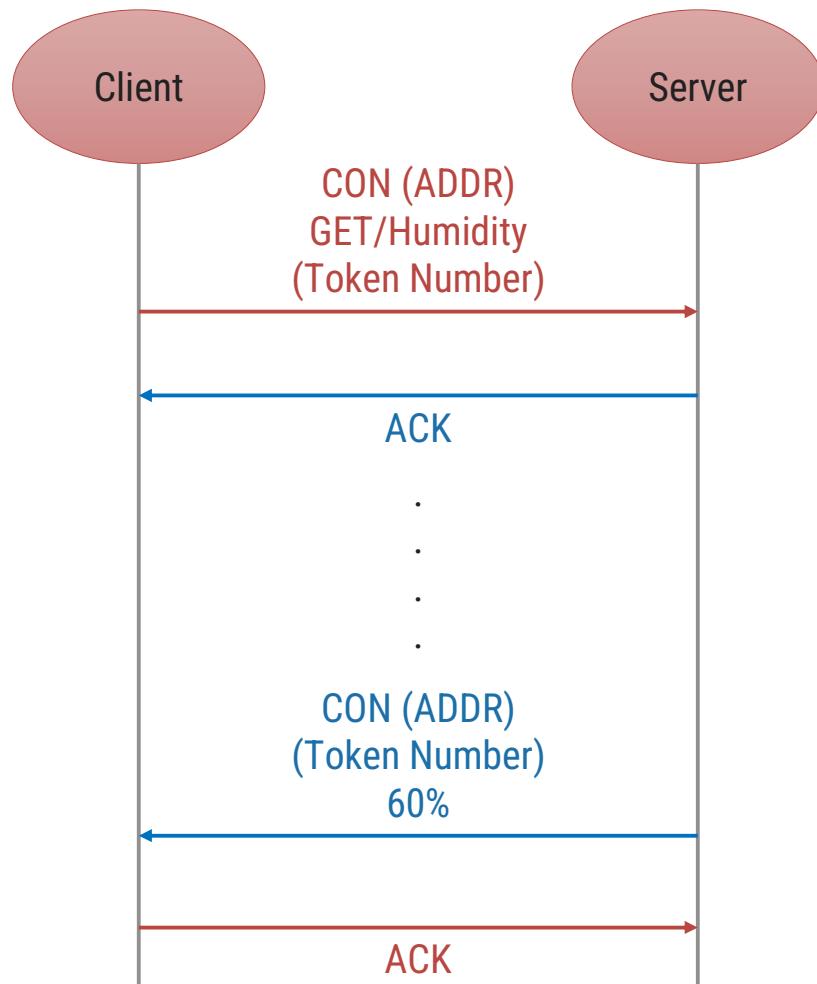
- In this mode, the client sends the data with particular method (GET, PUT etc.) with token number and CON messaging method.
- The ACK is transmitted by server immediately with corresponding token number and message.
- If the message is not received by server or data is not available then the failure code is embedded in ACK.



# Request-Response Layer in CoAP (Contd.)

## 2. Separate Response:

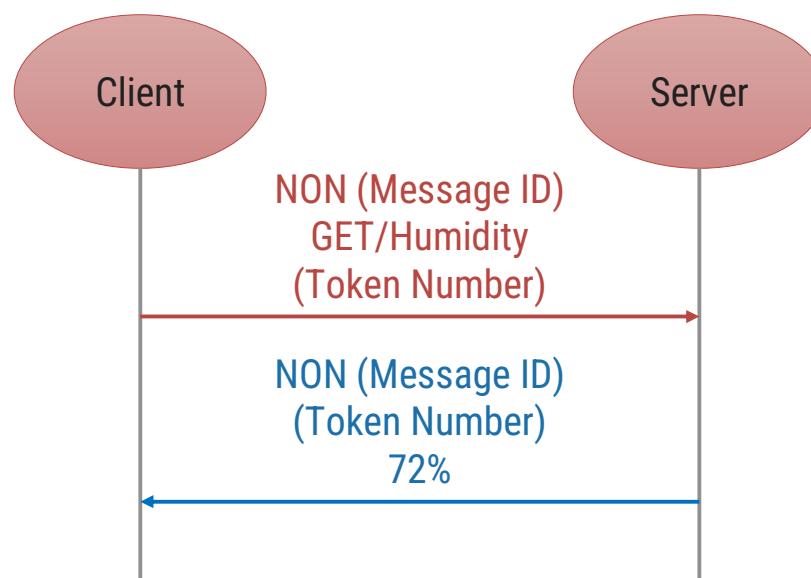
- In this mode, the client sends the data with particular method (GET, PUT etc.) with token number and CON messaging method.
- If the server is unable to respond immediately, an empty ACK will be reverted.
- After some time, when the server is able to send the response, it sends a CON message with data.
- In response to that, ACK is sent back from client to server.



# Request-Response Layer in CoAP (Contd.)

## 3. Non-Confirmable Request and Response:

- In this mode, the client sends the data with particular method (GET, PUT etc.) with token number and NON messaging method.
- The server does not give ACK in NON messaging method, so it will send a NON type response in return with token number and its data.



# CoAP Message Format

- ▶ The CoAP message format is of 16 Bytes consisting various fields.
- ▶ V : It refers to version of CoAP. It is two bit unsigned integer.
- ▶ T : It refers to message type. It is also two bit unsigned integer.
  - Confirmable (0)
  - Non-Confirmable (1)
  - ACK (2)
  - RESET (3)
- ▶ TKL : It refers to the token length and is four bit unsigned integer.
- ▶ Code: It refers to the response code.

0    1    2    3    4    5    6    7    8    9    10    11    12    13    14    15    16    17    18    19    20    21    22    23    24    25    26    27    28    29    30    31

V	T	TKL	Code	Message ID
Token (if any, with TKL bytes length)				
Options (if any)				
1    1    1    1    1    1    1    1				Payload

# CoAP Message Format (Contd.)

- ▶ Message ID: It is identifier of the message and to avoid duplication.
- ▶ Token : Optional field whose size is indicated by the Token Length field.
- ▶ Options : This field is used if any options available and having length of 4 Bytes.
- ▶ Payload : This field refers to the payload data where actual data is transmitted.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

V	T	TKL	Code	Message ID
Token (if any, with TKL bytes length)				
Options (if any)				
1	1	1	1	Payload

# XMPP Protocol

- ▶ The full form of XMPP is Extensible Messaging and Presence Protocol.
- ▶ It is designed for messaging, chat, video, voice calls and collaboration.
- ▶ The fundamental base of this protocol is XML and it is used for messaging services such as WhatsApp.
- ▶ The XMPP was originally named as Jabber and later known as XMPP.



# XMPP denotation

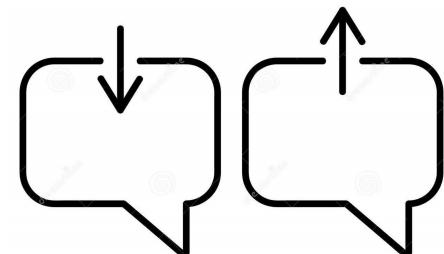
- ▶ The denotation for XMPP can be given as following:
- ▶ X – X denotes eXtensible.
  - It is considered extensible as it is designed to accept and accommodate any changes.
  - The protocol is defined in open standard and it is extensible because of open standard approach.
- ▶ M – M denotes Messaging.
  - XMPP supports sending message to the recipients in real time.
- ▶ P – P denotes Presence.
  - It is helpful to identify the status such as “Offline” or “Online” or “Busy”.
  - It also helps in understanding if the recipient is ready to receive the message or not.
- ▶ P – P denotes Protocol.
  - The set of standards together acting as a protocol.

# Messenger Requirements

► Any messenger requires the following features inbuilt :

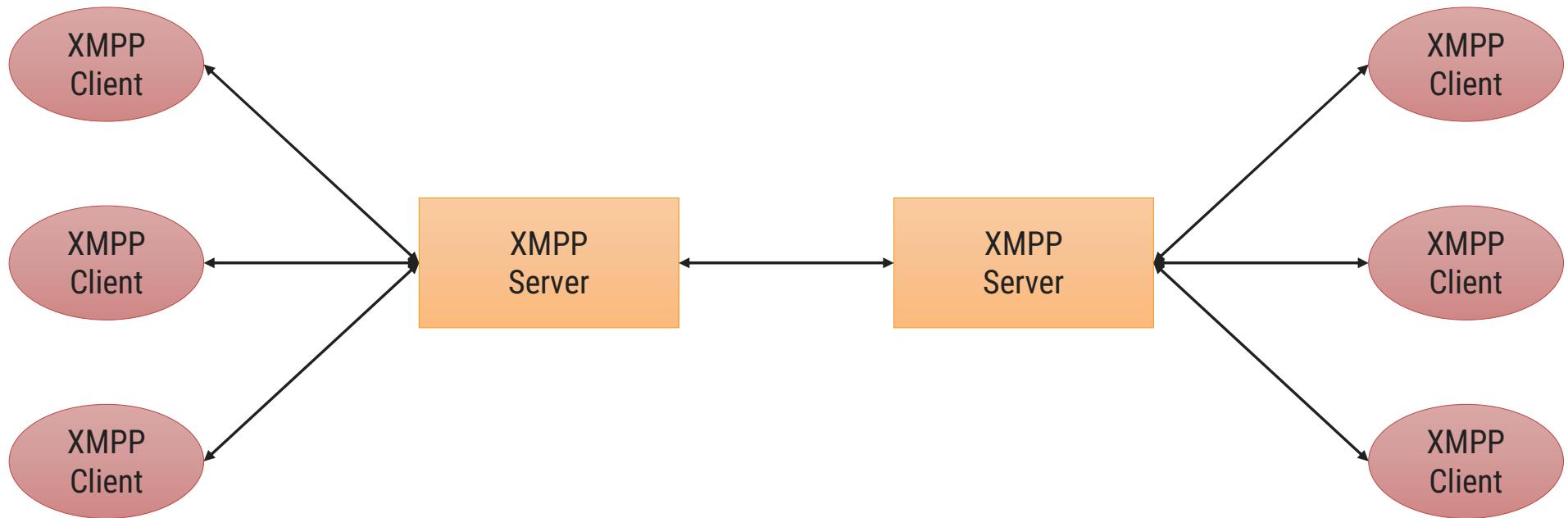
1. Message sending and receiving features.
2. Understanding the presence/absence status.
3. Managing subscription details.
4. Maintaining the contact list.
5. Message blocking services.

► Note that all the listed features are built in XMPP.



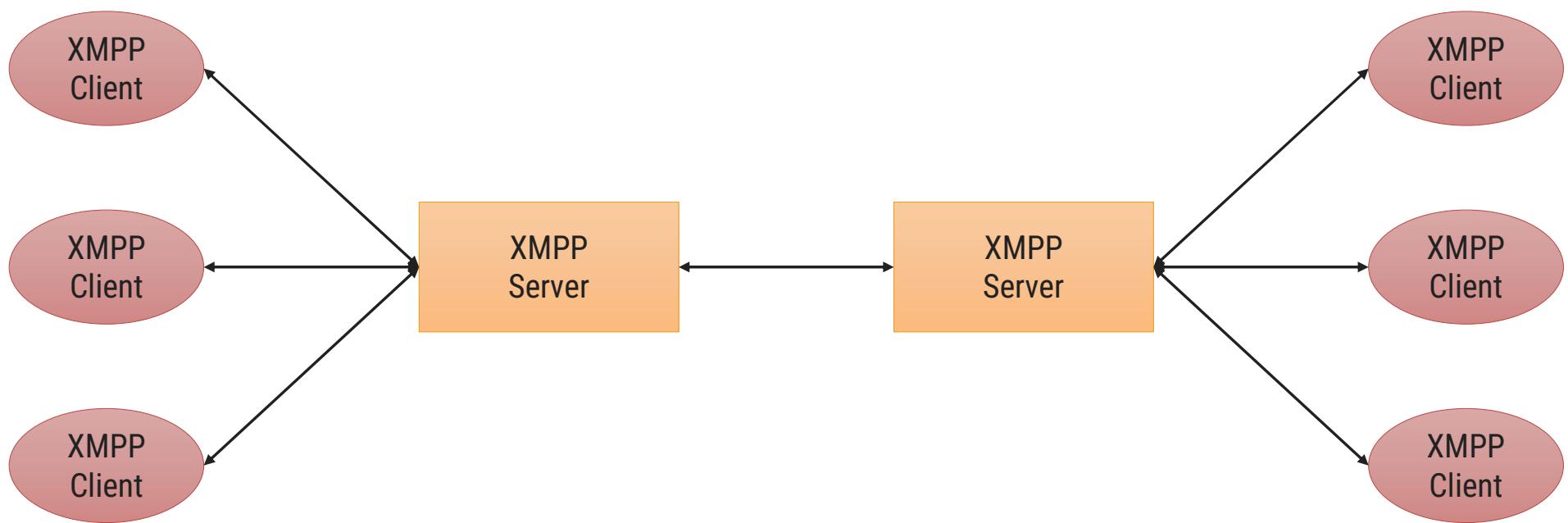
# XMPP Architecture

- ▶ The architecture of XMPP protocol is shown in the figure :
- ▶ The XMPP clients communicate with XMPP server bidirectional.
- ▶ There can be any numbers of clients connected to XMPP server.
- ▶ The XMPP servers may be connected to the other clients or other XMPP servers.



# XMPP Architecture (Contd.)

- ▶ The initial version of the XMPP was with TCP using open ended XML.
- ▶ After certain period of time, the XMPP was developed based on HTTP.
- ▶ The XMPP can work with HTTP through two different methods Polling and Binding.
- ▶ What is Polling and Binding?



## XMPP Architecture (Contd.)

- ▶ In polling method, the messages stored in the server are pulled or fetched.
- ▶ The fetching is done by XMPP client through HTTP GET and POST requests.
- ▶ In binding method, Bidirectional-streams Over Synchronous HTTP (BOSH) enables the server to push the messages to the clients when they are sent.
  
- ▶ The binding approach is more effective than polling.
- ▶ Also both method uses HTTP, hence the firewall allows the fetch and post activity without any difficulties.

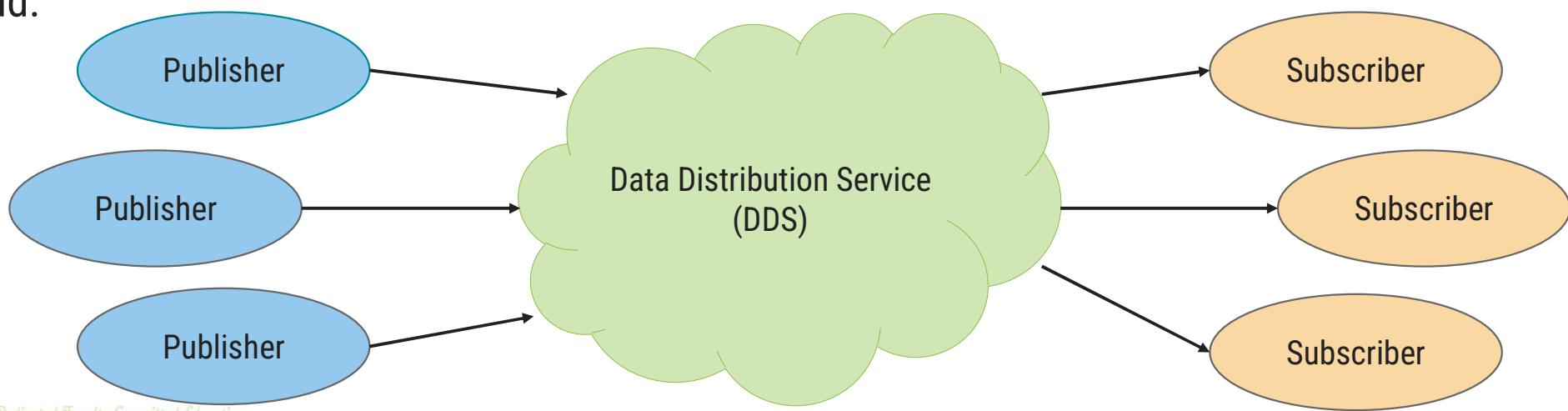
# Data Distribution Service (DDS)

- ▶ Data Distribution Service (DDS) is one of the protocol being used in the Internet of Things (IoT) applications.
- ▶ DDS is brokerless service and the protocol was developed by the Object Management Group (OMG).
- ▶ The purpose behind development of DDS is to get better machine to machine communication.
- ▶ Like MQTT, DDS is also using publish-subscribe approach.



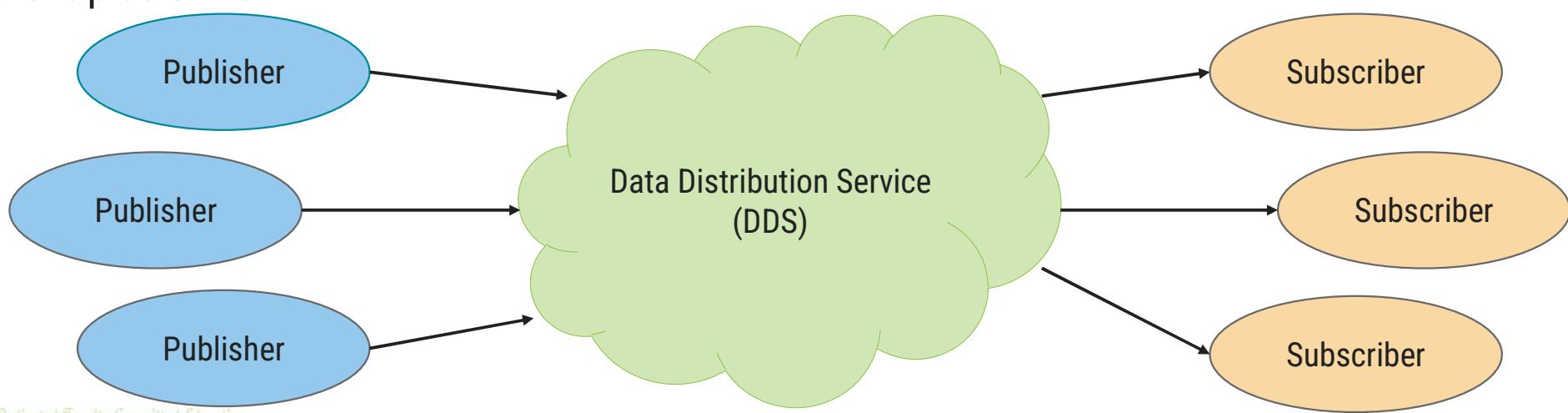
# DDS Architecture

- ▶ DDS is basically a publish – subscribe model based architecture.
- ▶ Everything including sending and receiving data, event updates, command within and among the nodes are all accomplished through the publish – subscribe approach.
- ▶ The publisher sends the data to DDS cloud with particular Topic.
- ▶ The interested nodes (Subscribers) subscribes the particular Topic and receives data from DDS Cloud.



## DDS Architecture (Contd.)

- ▶ Here, DDS Cloud has to handle all the attributes of data transfer like addressing, marshalling data, control of delivery, control of flow, etc.
- ▶ The marshalling of data means to convert any data or objects into a byte-stream, while unmarshalling is exactly the reverse process.
- ▶ This enables the communication between publisher and subscriber working on multiple different platforms.





# Transport Protocols

Section - 2



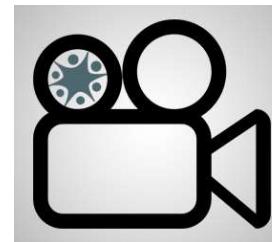
# Bluetooth 4.0 (BLE)

- ▶ BLE – Bluetooth Low Energy
- ▶ BLE is open low energy short range radio communication technology.
- ▶ The BLE is also known as Bluetooth Smart.
- ▶ This protocol is designed by Bluetooth Special Interest Group (SIG).
- ▶ BLE also works same as Bluetooth in wireless Personal Area Network (PAN).
- ▶ It can be considered the advanced version of Bluetooth.



# Bluetooth 4.0 (BLE) (Cont.)

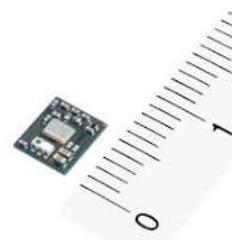
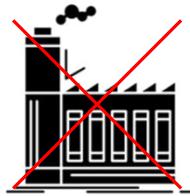
- ▶ BLE is superior to classic Bluetooth because of following reasons:
  - It is power saving.
  - It is supported by Android, IOS, Blackberry, etc.
  - It is also supported by computer OS like Windows 8/10, MAC-OS, and Linux variants.
- ▶ Because of certain advantages of BLE over classic Bluetooth, it is widely used in the areas like:
  - Healthcare
  - Entertainment
  - Fitness (in form of wearable devices)
  - Proximity related applications
  - Tracking devices



# Features of BLE

► The key features of BLE are as following:

1. It is licence free and does not add any overhead costing.
2. There is no restrictions to Manufacturer.
3. BLE modules are inexpensive and affordable.
4. The size of BLE modules are small. So it can be accommodated in any system easily.
5. The power consumption for BLE modules is very less and hence suitable for IoT applications.
6. The range of BLE has very high range compared to classic Bluetooth.



# BLE Architecture

► The BLE architecture has three components.

► Application Block

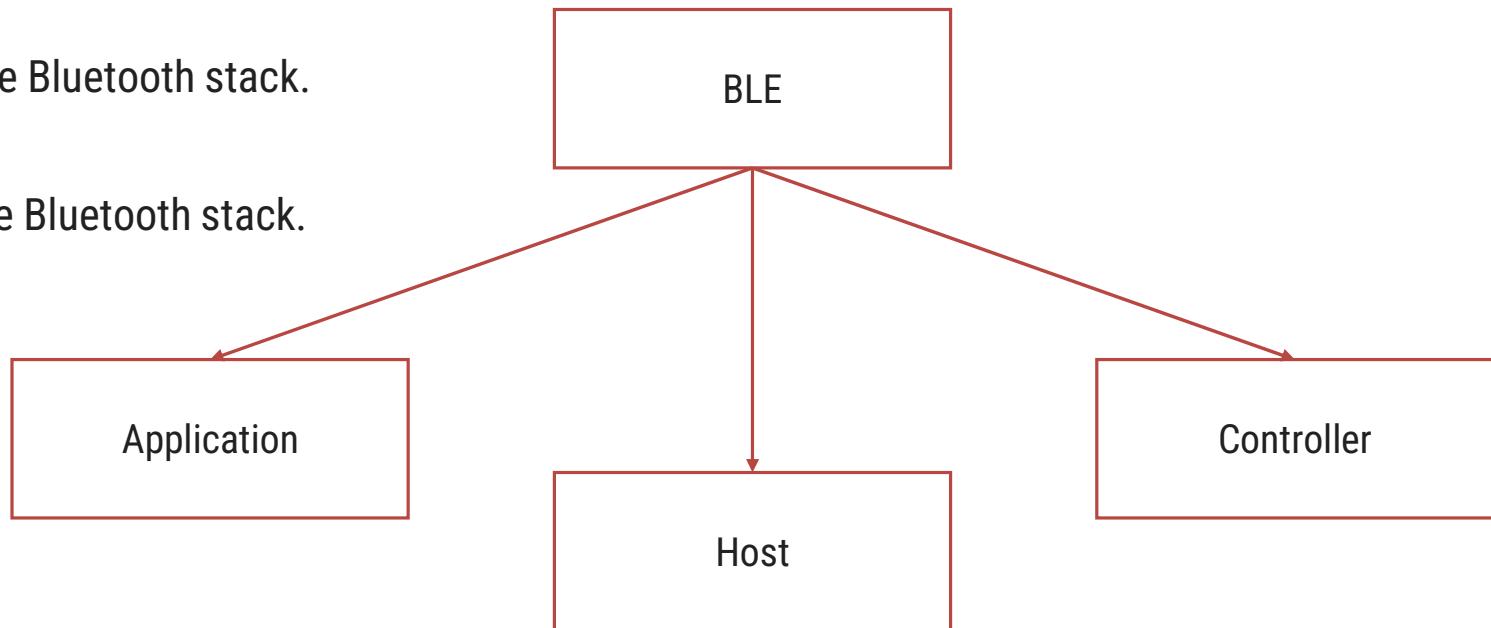
- The user application is accommodated in this block.
- It interacts directly with Bluetooth stack.

► Host Block

- It is upper layer of the Bluetooth stack.

► Controller Block

- It is lower layer of the Bluetooth stack.



Application

## Application

Host

Generic Access Profile  
(GAP)

Generic Attribute Profile  
(GATT)

Security Management Protocol  
(SMP)

Attribute Protocol  
(ATT)

Logical Link Control and Adaptation Protocol  
(L2CAP)

Controller

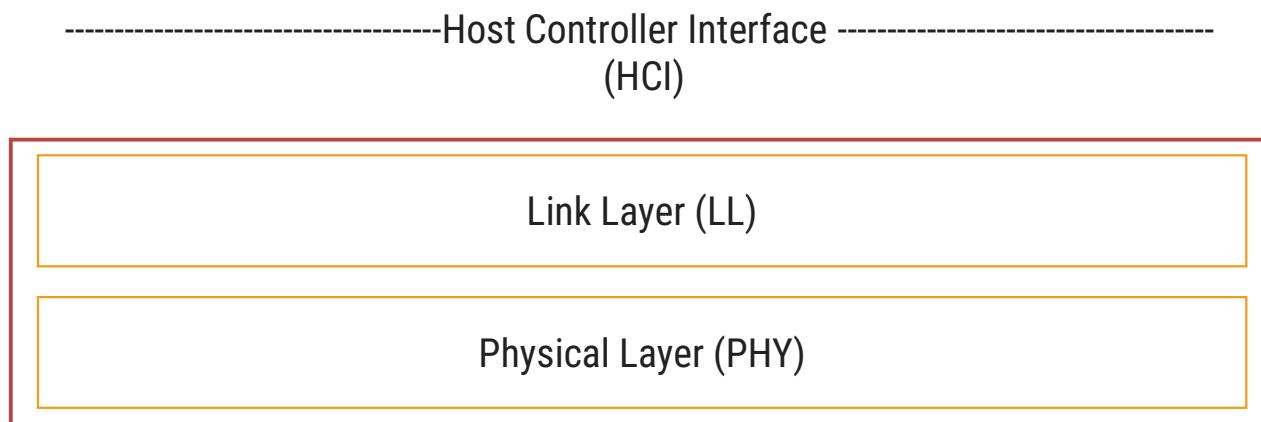
Link Layer (LL)

Physical Layer (PHY)

# BLE Architecture

1. Controller : It has three components.

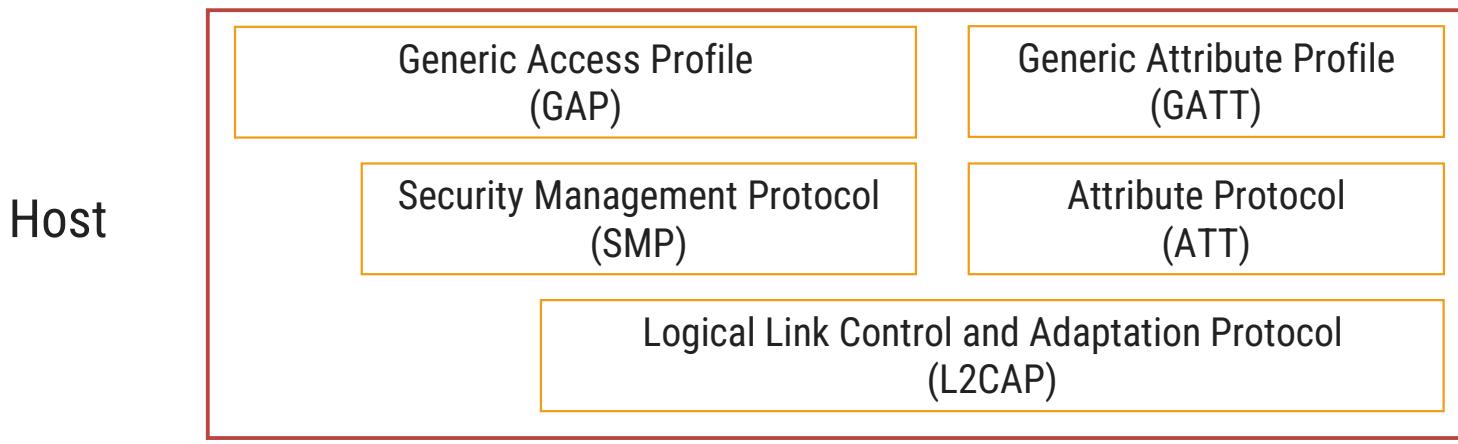
- a) Host Controller Interface (HCI) : is used to enable interoperability between hosts and controllers assembled by different manufacturers
- b) Link Layer (LL) : It defines the packet structure
- c) Physical Layer (PHY) : It takes care of the transmission/reception, modulation/demodulation and analog-to-digital conversion.



# BLE Architecture

## 2. Host : It has following components with functionalities

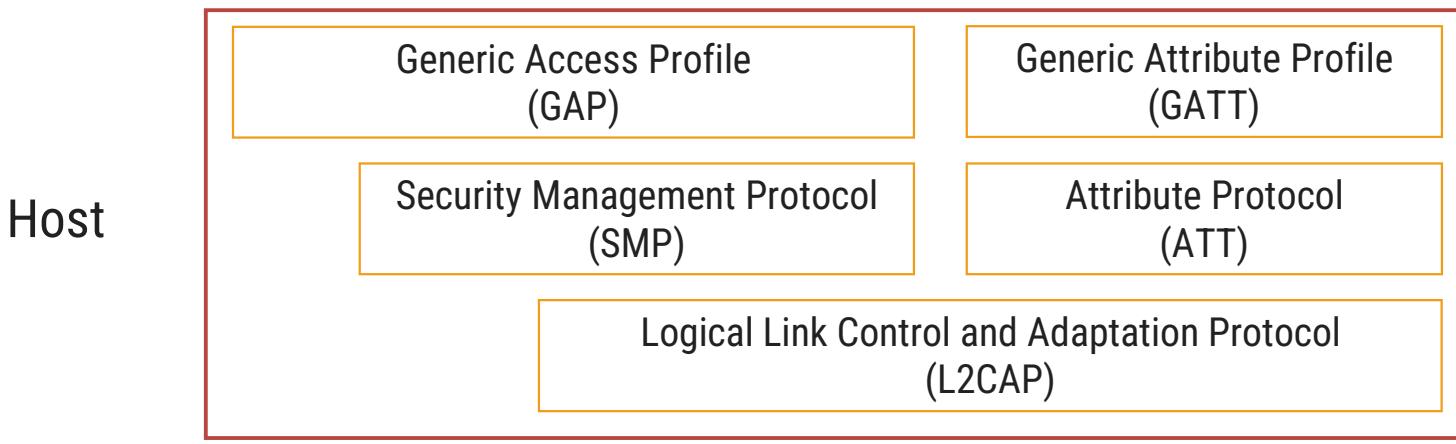
- a) Generic Access Profile (GAP) : It takes care of the device discovery, connection establishment, connection management and security.
- b) Generic Attribute Profile (GATT) : It is responsible for the data exchange process. Whenever there is a need to push data and to read or write data, the generic guidelines with respect to the process are governed by GATT.
- c) Attribute Protocol (ATT) : It is the protocol for accessing data.



# BLE Architecture

## 2. Host : It has following components with functionalities

- d) Logical Link Control and Adaptation Protocol (L2CAP) : It is responsible for fragmentation and de-fragmentation of the application data. Also, it is responsible for the multiplexing and de-multiplexing of channels over the shared logical link.
- e) Security Manager (SM) : It takes care of pairing, authentication, and encryption. Everything related to security is taken care here.
- f) Host Controller Interface (HCI) : The functionality remains the same as that in the controller. It is used to enable interoperability between hosts and controllers assembled by different manufacturers.



# BLE Architecture

## 3. Application Layer :

- The working of application layer in BLE is same as in OSI model. It is the top most layer in BLE and this layer is responsible for user interface (UI), logic and data handling.

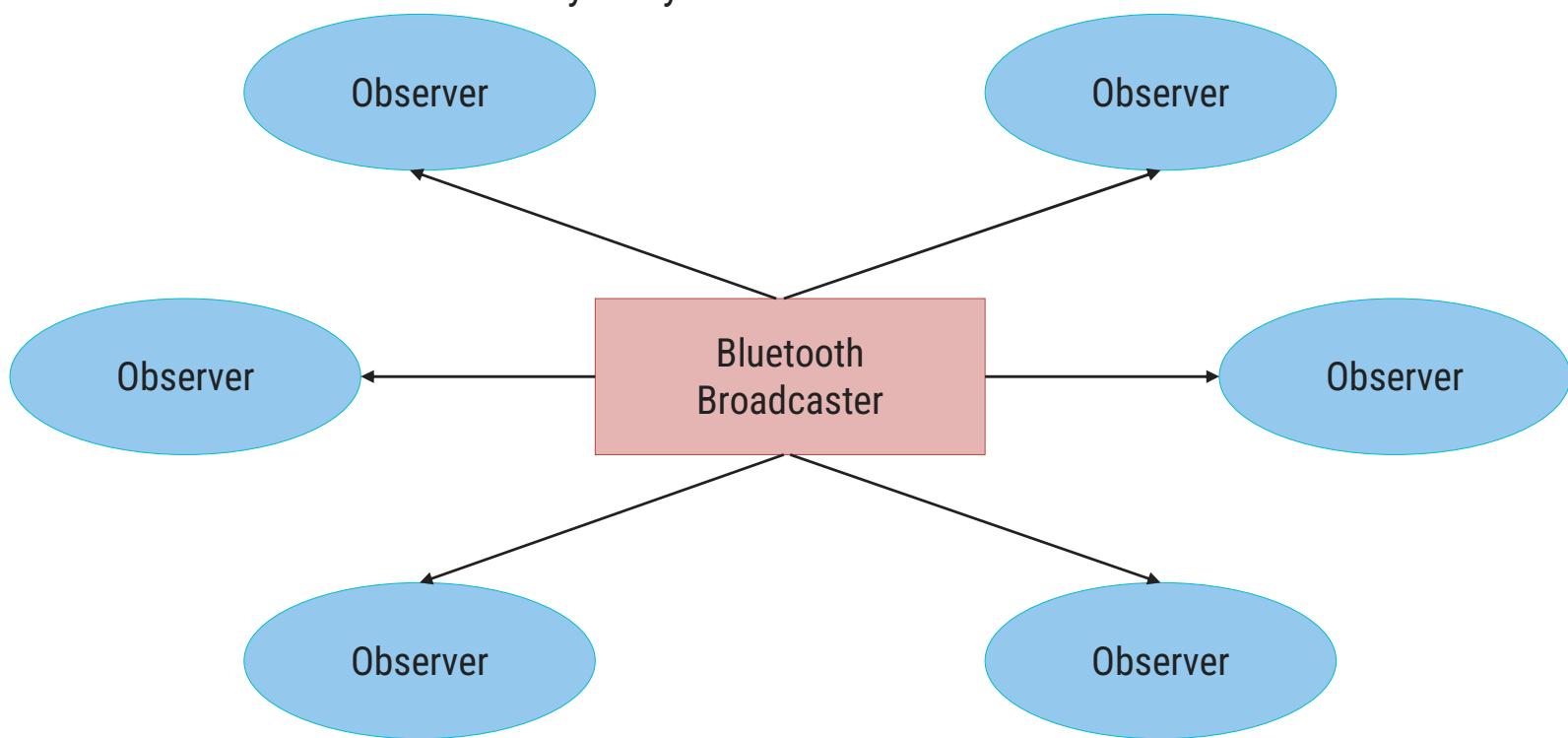
Application

Application

# BLE Topology

## Broadcasting

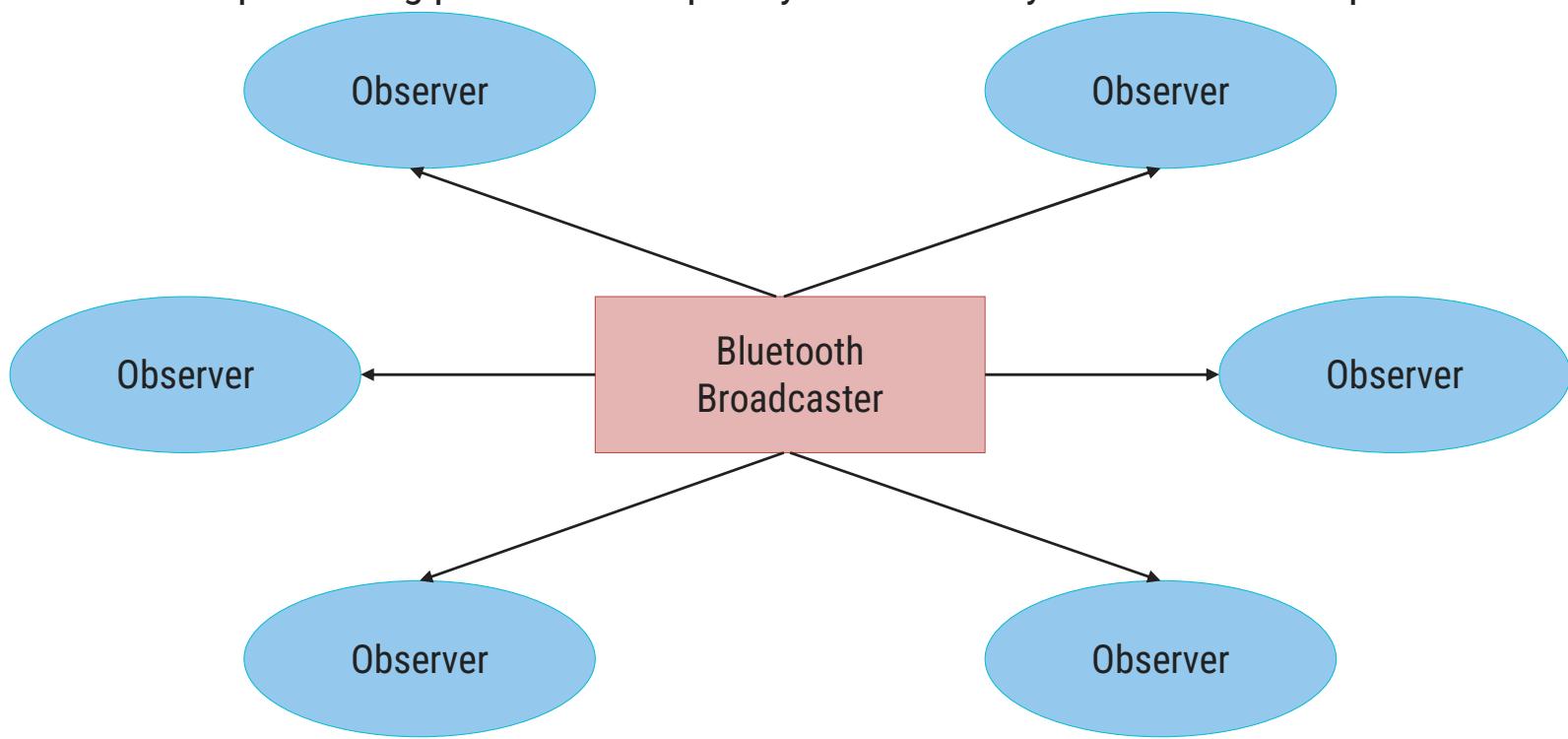
- The meaning of broadcasting is to send a message to more than one recipient. The same concept is followed in BLE.
- The broadcast method will send the data in “one way” only.
- The receiver in the vicinity of the broadcaster will receive the message or data.
- There are two parts in broadcasting topology. Broadcaster and Observer.



# BLE Topology

## ► Broadcasting

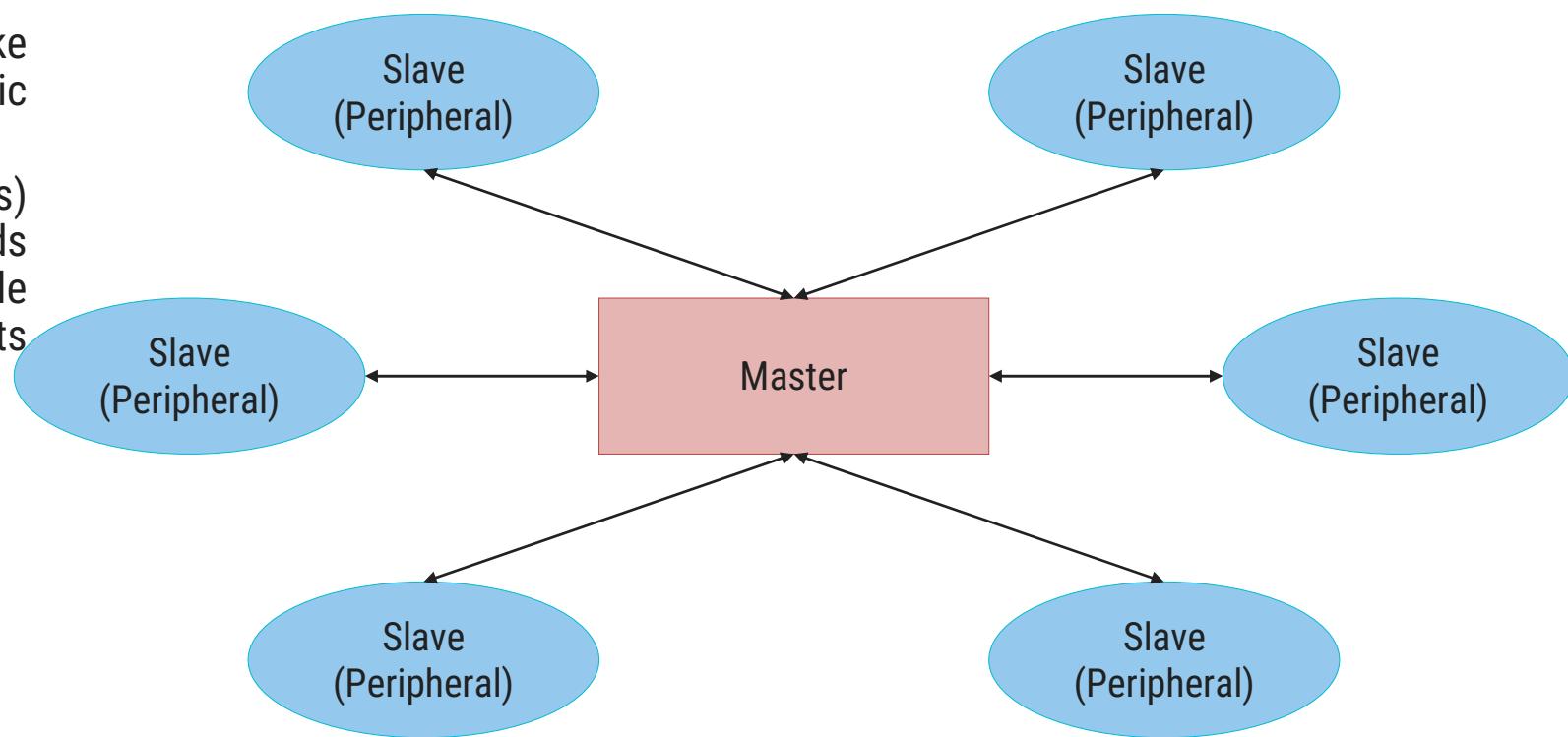
- Broadcaster : The broadcaster sends a "non-connectable" advertising packets in frequent intervals to all those who are willing to collect which is similar to a radio broadcast.
- Observer : The observer will keep scanning predefined frequency to receive any non-connectable packets..



# BLE Topology

## ▶ Connections

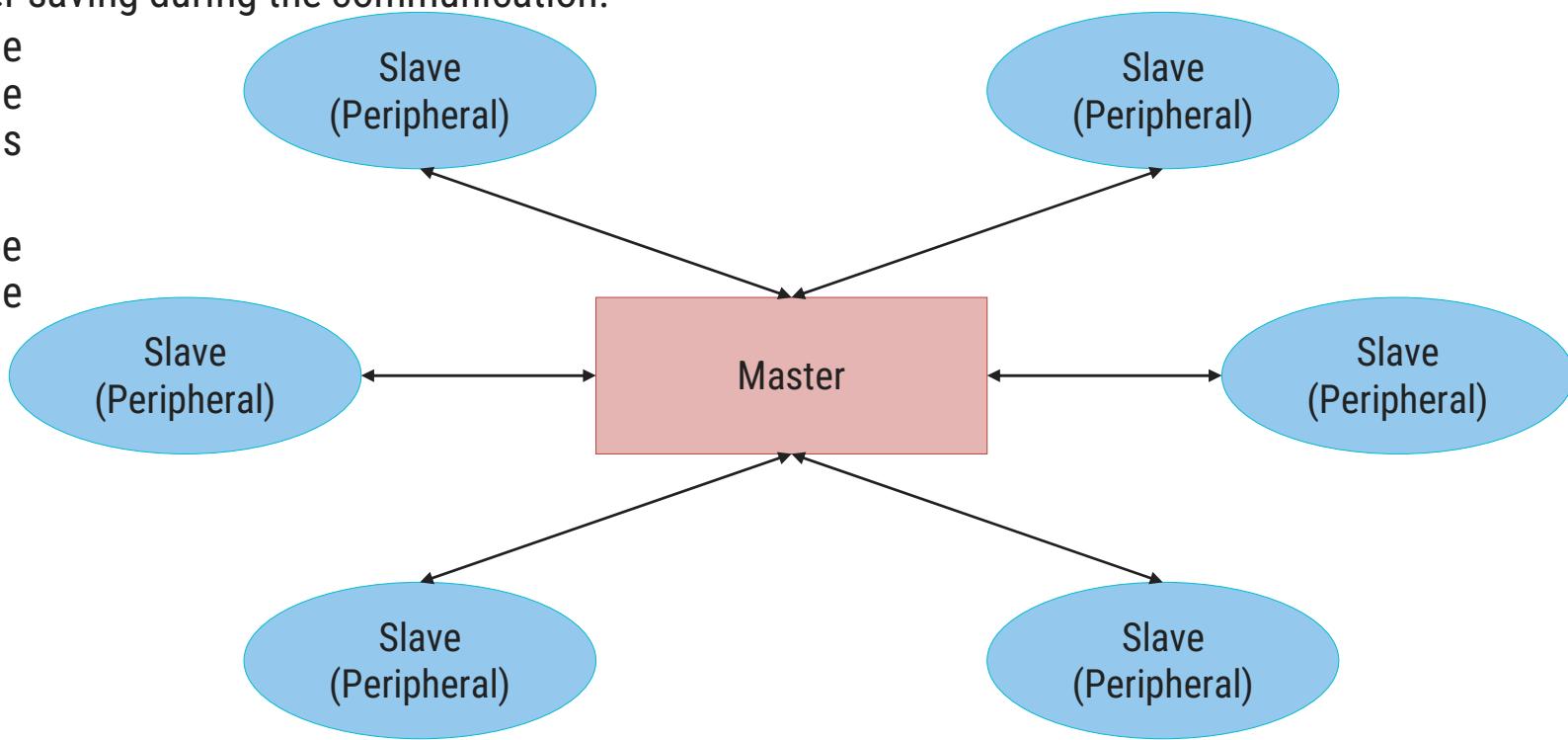
- The connections topology works on the concept of master-slave configuration.
- The master will try to find out the connectable advertising packets. Once the advertisement is found, it will initiate the connection to the slave.
- The master will take care of periodic data exchange.
- The slave(s) continuously sends the connectable advertising packets at regular interval.



# BLE Topology

## ▶ Connections

- Once the slave is connected, the event is called connection event.
- In this topology, the device will work whenever required. For the remaining time the device is in sleep mode.
- This provides power saving during the communication.
- However, during the sleep mode also the connection remains present.
- But the data will be transferred after the device invokes.



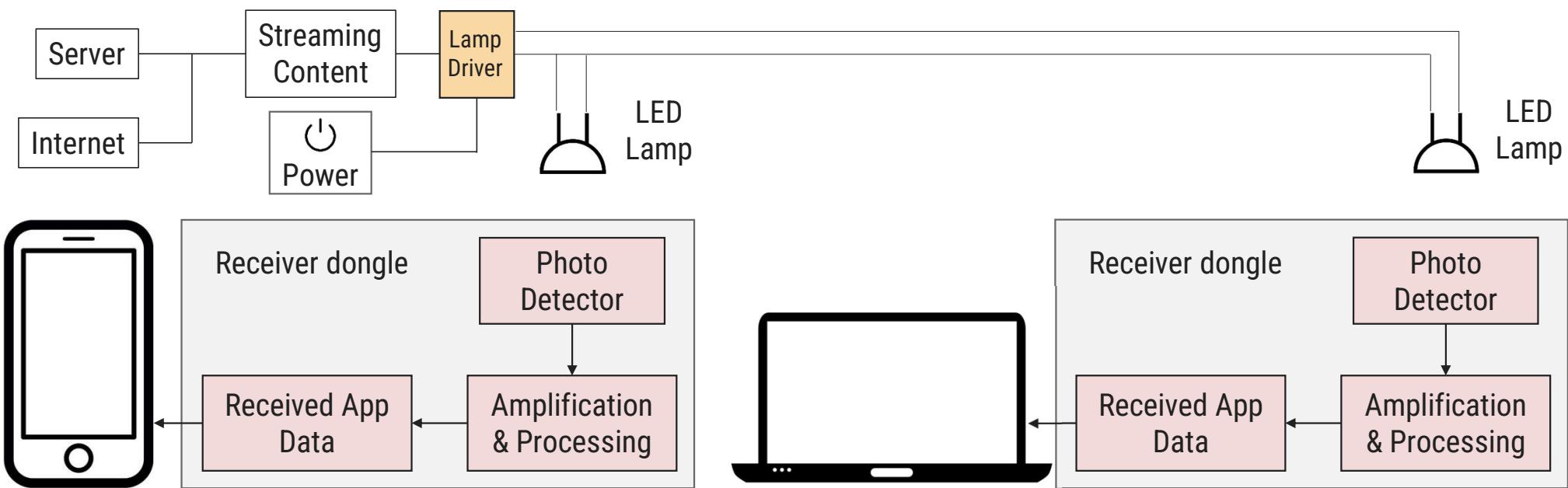
# Light Fidelity (Li-Fi)

- ▶ The Li-Fi is fastest communication protocol right now among all existing protocols.
- ▶ The symbol of Li-Fi is as shown in the figure.
- ▶ It gives the following solutions to the problems which one can face during the use of the Wi-Fi.
  - More security than Wi-Fi.
  - More available bandwidth to the connected devices or equipments.
  - Less/No congestion when more number of devices connected.
- ▶ Note that the Li-Fi has phenomenal speed of 224GBps which is fastest among all.



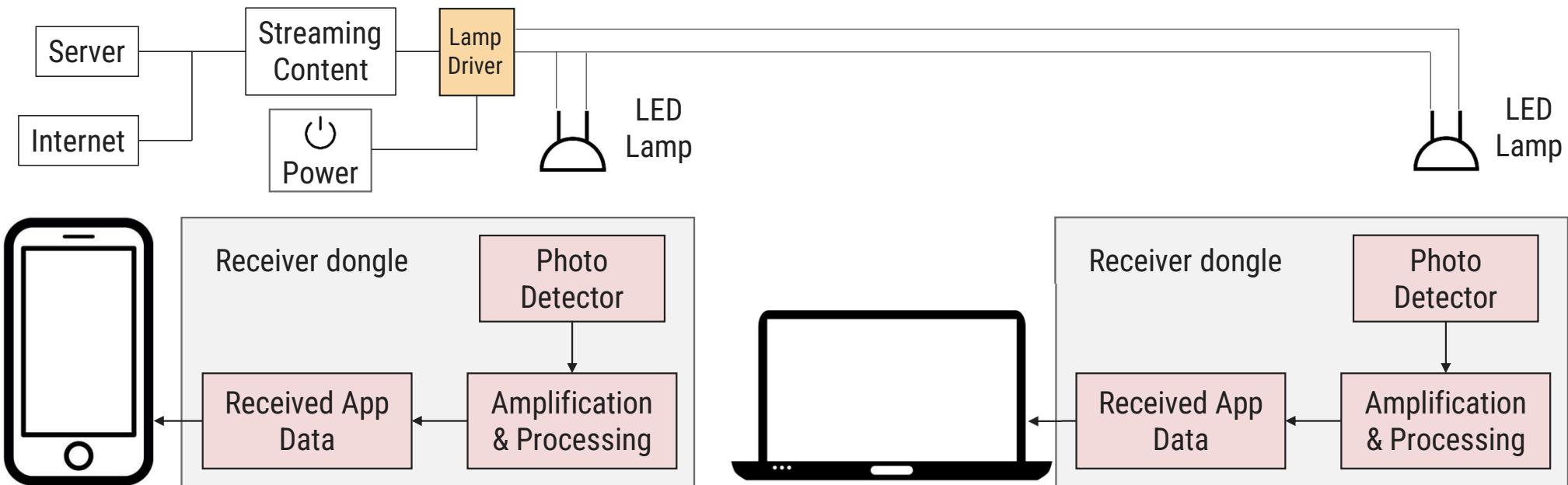
# Li-Fi Working

- ▶ The server wants to transmit the data to the receiver with very high speed. The server is either connected to internet or intranet.
- ▶ The server connected to the internet generates a data which is converted into streaming content.
- ▶ The streaming content is given to Lamp driver which is powered by external source.



# Li-Fi Working

- ▶ The LED lamp is turned on and off at very high speed that human eyes can not capture it.
- ▶ The light is received by the photo detector and it is passed for demodulation process and amplification to generate the data stream sent by transmitter.
- ▶ The received data stream is further given to receiver (e.g. mobile, PC etc.) via receiver app. All components working for receiving the data combined known as receiver dongle.



# Li-Fi – Advantages & Disadvantages

## ► Advantages

- Li-Fi is extremely secure as the light can not penetrate through walls and no data hijacking is possible.
- Li-Fi is fastest and effective.
- It is an effective alternate to RF communication.
- It is inexpensive as the light is generated through LED lamps.

## ► Disadvantages

- It can be implemented in short range as the presence of walls or any other object will become the interruption in communication.
- The required time for set up is higher.
- There is no clarity of bidirectional communication means how to send data from mobile or PC back to server.

# **Addressing & Identification Protocols**

Section - 3

# Internet Protocol Version 4 (IPv4)

- ▶ IP addresses are useful in identifying a specific host in a network.
- ▶ IP addresses are 32 bit numbers which are divided into 4 octets. Each octet represents 8 bit binary number.
- ▶ Below is an example of an IP address:

10101100	00010000	11111110	00000001
172	16	254	1

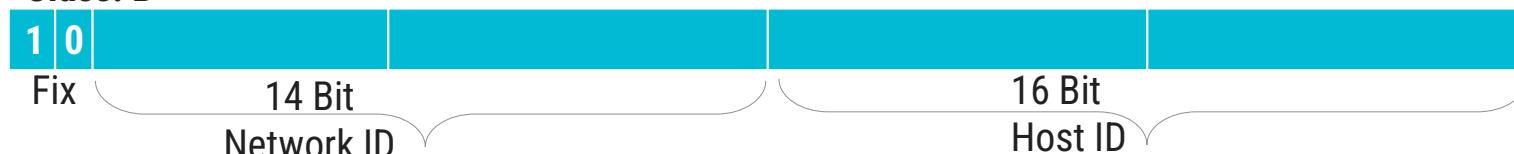
**IP addresses are divided into 2 parts:  
Network ID & Host ID  
<NID> <HID> = IP Address**

# Classification of IP Address

## Class: A



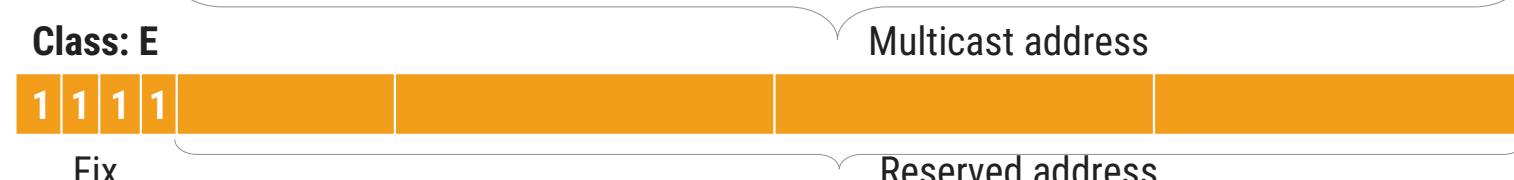
## Class: B



## Class: C



## Class: D



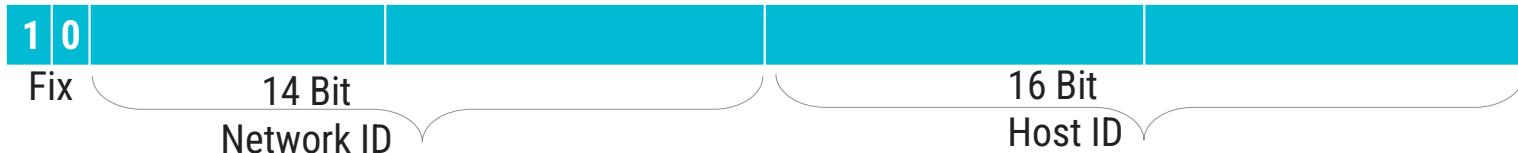
# Class A



- ▶ Only 126 addresses are used for network address.
- ▶ All 0's and 1's in Network-ID are dedicated for special IP address. So, total number of IP address in class A can be represented as following.
- ▶ The range of Class A is **0.0.0.0 to 127.255.255.255**

0.0.0.0	Special IP Address
00000001.0.0.1	
1.0.0.2	
1.0.0.3	
.	$2^{24} - 2$ are Host IP
.	
.	
126.255.255.254	
127.255.255.255	Special IP Address – Loopback

# Class B



- ▶ No special network address here. All are usable.
- ▶ The total number of hosts connected in network can be given as following:
- ▶ It is used in medium size network.
- ▶ The range of Class B is **128.0.0.0 to 191.255.255.255**

128.0.0.0	Special IP Address
10000001.0.0.1	
130.0.0.2	
130.0.0.3	
.	$2^{16} - 2$ are Host IP
.	
.	
190.255.255.254	
10111111.255.255.255	Special IP Address – Loopback

# Class C



- ▶ This class is used in small networks.
- ▶ The calculation for number of hosts can be given as following:
- ▶ Normally, the LANs are configured with Class C.
- ▶ The range for Class C is given as **192.0.0.0 to 223.255.255.255**

192.0.0.0	Special IP Address
11000001.0.0.1	
194.0.0.2	
194.0.0.3	
.	$2^8 - 2$ are Host IP
.	
.	
222.255.255.254	
11011111.255.255.255	Special IP Address – Loopback

## Class D

- ▶ Very first four bits of the first octet in Class D IP addresses are set to 1110, giving a range of:

**11100000 – 11101111**

224 – 239

- ▶ Class D has IP address rage from 224.0.0.0 to 239.255.255.255.
- ▶ Class D is reserved for Multicasting.
- ▶ In multicasting data is not destined for a particular host, that is why there is no need to extract host address from the IP address, and Class D does not have any subnet mask.

## Class E

- ▶ This IP Class is reserved for experimental purposes only for R&D or Study.
- ▶ IP addresses in this class ranges from 240.0.0.0 to 255.255.255.254.
- ▶ Like Class D, this class too is not equipped with any subnet mask.

# IPv4 Format

- ▶ Figure shows the format for IPv4.
- ▶ VER (Version):
  - It is a 4 bits field which indicates the version of IP.
- ▶ IHL (Internet Header Length):
  - It specifies the Internet Header Length in 32 bits word length and points the beginning of data.
- ▶ Types of Service:
  - It is a 8 bit field which specifies the Quality of Service (QoS). The format for these 8 bits are given here.

VER (4)	IHL (4)	Types of Service (8)	Total Length (16)				
Identification (16)		Flags(3)		Fragment Offset (13)			
Time to Live (8)	Protocol (8)	Header Checksum (16)					
Source Address (32)							
Destination Address (32)							
Options (if any)							

# IPv4 Format

## ► Types of Service:

- Precedence (3) : They are used for future options.
- Delay (1): It indicates the delay of either normal or lower.
- Throughput (1): Indicates the speed of throughput of either normal or higher rate.
- Reliability (1): It indicates normal reliability or high reliability.

Precedence	Delay	T	R	Reserved Bits
------------	-------	---	---	---------------

VER (4)	IHL (4)	Types of Service (8)	Total Length (16)			
Identification (16)		Flags(3)		Fragment Offset (13)		
Time to Live (8)	Protocol (8)		Header Checksum (16)			
Source Address (32)						
Destination Address (32)						
Options (if any)						

# IPv4 Format

## ► Total Length:

- This is a 16 bit field defining the length of IPv4 datagram. The minimum length of datagram is 20 bytes and maximum is 65535.

## ► Identification:

- This field is of 16 bits which helps in assembling the fragments and added by the senders.

## ► Flags:

- There are 3 bits in flags. First is always '1'. Second is DF (Don't Fragment) and third is MF (More Fragment).

VER (4)	IHL (4)	Types of Service (8)	Total Length (16)						
Identification (16)		Flags(3)	Fragment Offset (13)						
Time to Live (8)	Protocol (8)	Header Checksum (16)							
Source Address (32)									
Destination Address (32)									
Options (if any)									

# IPv4 Format

- ▶ **Fragment Offset:**
  - When fragmentation is done, this specifies the offset or position of overall message.
- ▶ **Time to Live:**
  - This is 8 bit field that indicates the time period of datagram to survive.
- ▶ **Protocol:**
  - This field identifies the protocol for higher layer in the IP datagram.

VER (4)	IHL (4)	Types of Service (8)	Total Length (16)	
Identification (16)		Flags(3)	Fragment Offset (13)	
Time to Live (8)	Protocol (8)	Header Checksum (16)		
Source Address (32)				
Destination Address (32)				
Options (if any)				

# IPv4 Format

- ▶ Header Checksum:
  - It is 16 bit field to provide basic protection in transmission of header via checksum.
- ▶ Source Address:
  - This is 32 bit IP address of the source of the datagram.
- ▶ Destination Address:
  - This is 32 bit IP address of the destination of the datagram.
- ▶ Options:
  - There are many optional header available for debug and test purpose.

VER (4)	IHL (4)	Types of Service (8)	Total Length (16)	
Identification (16)		Flags(3)	Fragment Offset (13)	
Time to Live (8)	Protocol (8)	Header Checksum (16)		
Source Address (32)				
Destination Address (32)				
Options (if any)				

# IPv6 Addressing

- ▶ As the number of devices grow connected to the internet, it is obvious that we need more number of IP addresses to assign all the devices.
- ▶ IPv4 IP addressing uses 32 bit IP addresses which can generate  $2^{32} = 4294967296$
- ▶ The above value is just above 4 billion. In fact the actual available IP addresses in IPv4 is even less than theoretical value.
- ▶ The actual IP addresses that can be assigned in IPv4 are 3,720,249,092.
- ▶ The IPv6 IP addressing uses 128 bit IP addresses which can generate  $2^{128} = 3.4028236692093846346337460743177 \times 10^{38}$
- ▶ The above value is very large that if we assign a unique IP to every item in the world then also it will have reserved IP addresses.,

# Features of IPv6 Addressing

- ▶ It has better and efficient routing than IPv4.
- ▶ The additional flow label is added in header of IPv6 for improvement in QoS.
- ▶ IPv6 has new application like IP telephony, video/audio, interactive games etc. with ensured QoS.
- ▶ The plug and play abilities have been improved in IPv6.
- ▶ IPv6 has eliminated the need of Network Address Translation due to the huge availability of IP addresses.

# IPv6 Header Format

- ▶ IPv6 header format has less fields than IPv4 header format.
- ▶ IPv6 header is 40 bytes which is twice than IPv4 header.
- ▶ The figure shows IPv6 header format.
- ▶ The IPv6 has simple packet handling and improved forwarding efficiency.
- ▶ The fields of IPv6 header can be explained as following:

IP Version Number (4)	Traffic Class (8)	Flow Label (20)
Payload Length (16)	Next Header (8)	Hop Limit (8)
Source Address (128)		
Destination Address (128)		

# IPv6 Header Format

- ▶ IP version :
  - It is 4 bit field to represent the IP version being used.
- ▶ Traffic Class :
  - It is 8 bit field to identify the different classes or priorities of IPv6 packets.
  - It replaces the type of services in IPv4.
  - Upper 6 bits are used for type of services lower 2 bits are used for Explicit Congestion Notification.
- ▶ Flow Label :
  - It is 20 bit field used to identify the sequence of packets.
  - It is also useful for prioritizing the delivery of packet and providing real time service.
  - The higher priority packets can be delivered ahead of lower priority packets.

IP Version Number (4)	Traffic Class (8)	Flow Label (20)
Payload Length (16)	Next Header (8)	Hop Limit (8)
Source Address (128)		
Destination Address (128)		

# IPv6 Header Format

- ▶ Payload Length :
  - It is a 16 bit field which identifies the length of payload of IPv6.

- ▶ Next Header :
  - It is a 8 bit field which is similar to protocol field in IPv4 header.
  - It represents the type of extension header that follows the primary IPv6 header.

- ▶ Hop Limit :
  - It is 8 bit field equivalent to Time to Live in IPv4 Header.
  - The value is decremented by 1 every time when it is forwarded by the host.
  - When this value reaches 0, the packet is discarded.

IP Version Number (4)	Traffic Class (8)	Flow Label (20)
Payload Length (16)	Next Header (8)	Hop Limit (8)
Source Address (128)		
Destination Address (128)		

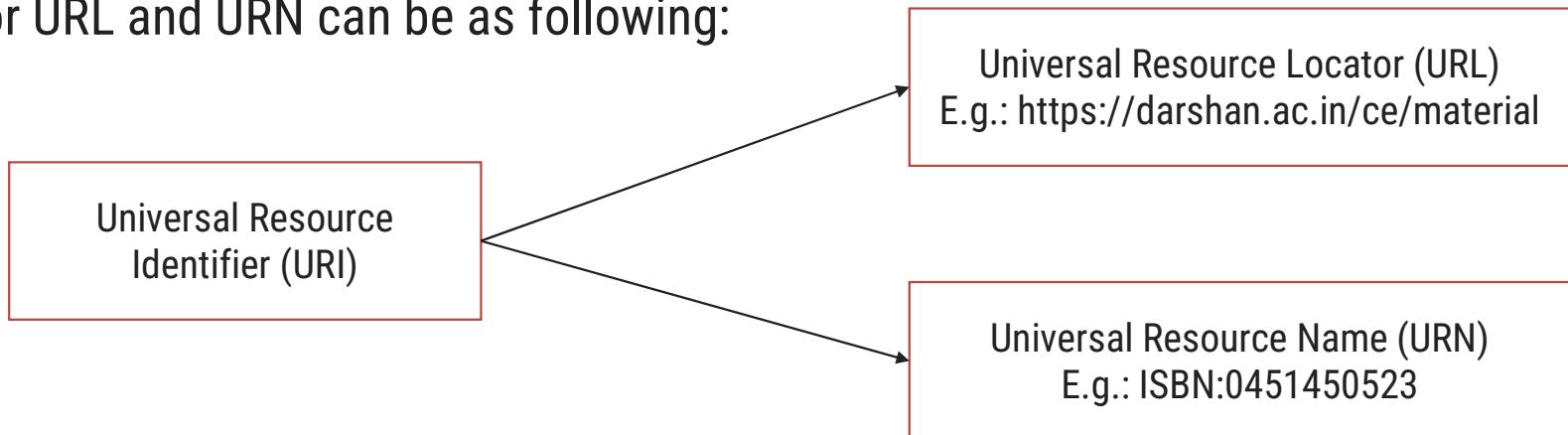
# IPv6 Header Format

- ▶ Source Address :
  - It represents the 128 bit IP address of source or sender.
- ▶ Destination Address :
  - It represents the 128 bit IP address of destination or receiver.

IP Version Number (4)	Traffic Class (8)	Flow Label (20)
Payload Length (16)	Next Header (8)	Hop Limit (8)
Source Address (128)		
Destination Address (128)		

# Uniform Resource Identifier (URI)

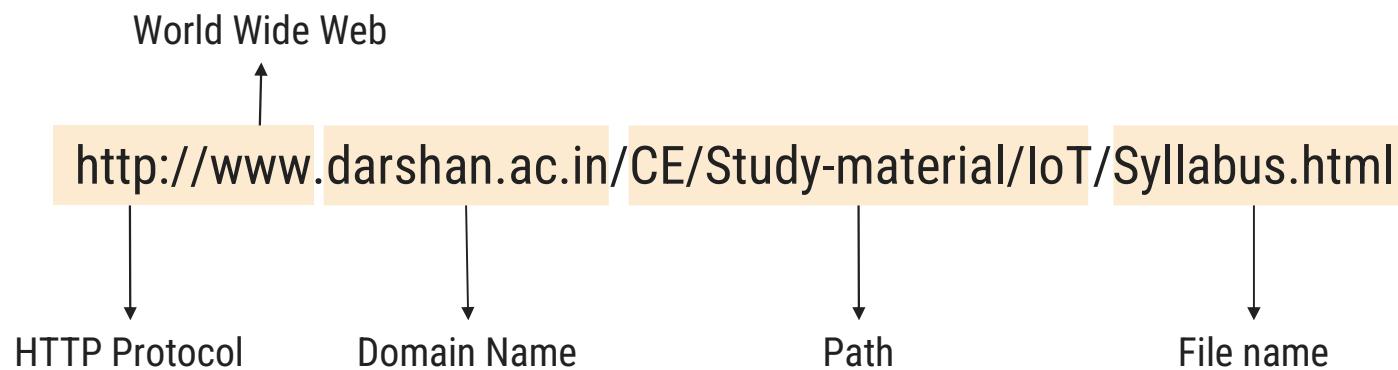
- ▶ Uniform Resource Identifier (URI) is used to identify the resources with the help of sequence of characters.
- ▶ The URI protocol was developed by IETF.
- ▶ The URI can be URL or URN.
- ▶ The sample for URL and URN can be as following:



- ▶ URN mostly provides information of unique name without locating or retrieving the resource information.
- ▶ URL provides locating and retrieving information on a network.

# Uniform Resource Locator (URL)

- ▶ The URL contains the information of how to fetch a resource information from its location.
- ▶ URL always begin with a protocol like HTTP/HTTPS
- ▶ A URL is used when client request the server for the service.
- ▶ The sample URL and its fields are explained below:





Unit-4

# Cloud for IoT



**Prof. Kalpesh H Surati**  
Computer Engineering Department  
Darshan Institute of Engineering & Technology, Rajkot  
  
✉ Kalpesh.surati@darshan.ac.in  
📞 +91 99250 10033



# Introduction to Cloud Computing

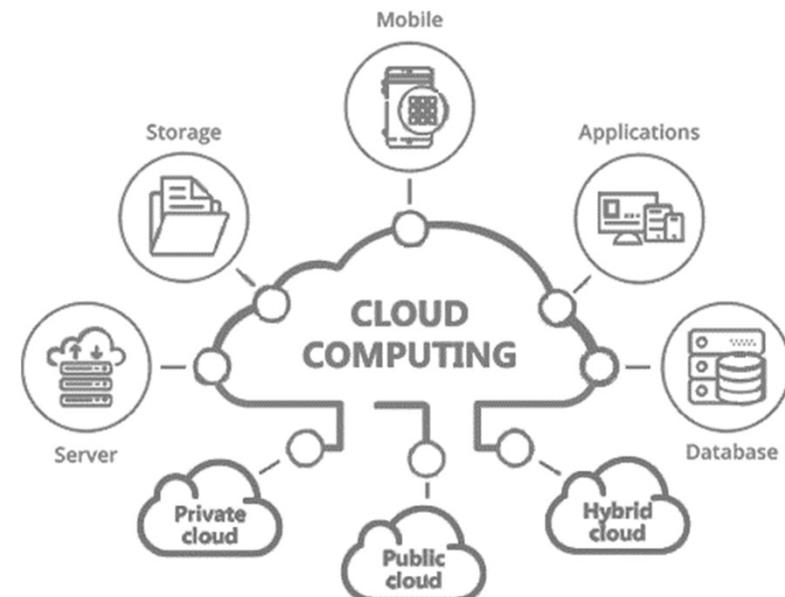
- ▶ Cloud computing is the delivery of computing services over the internet, by providing flexible, affordable, effective and efficient resources for development.
- ▶ That includes servers, storage, databases, networking, software, analytics, and intelligence.
- ▶ Cloud computing provide economical freedom along with technical strength to the application.
- ▶ In other word, it is about outsourcing of IT services and infrastructure to make them available anywhere via the Internet.



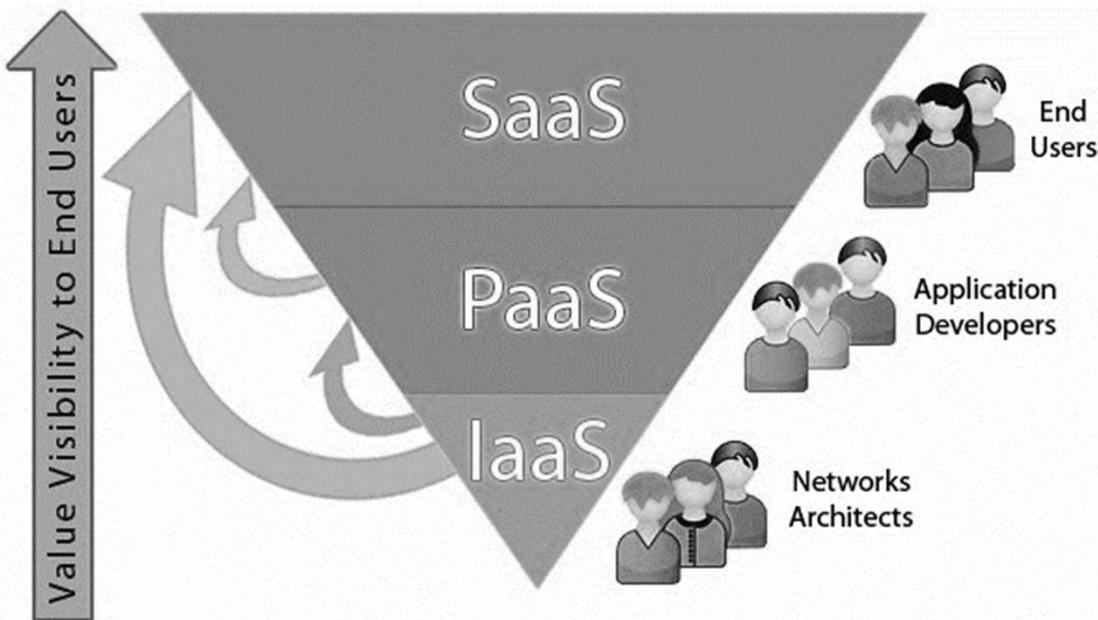
Application logic runs  
on user's computer



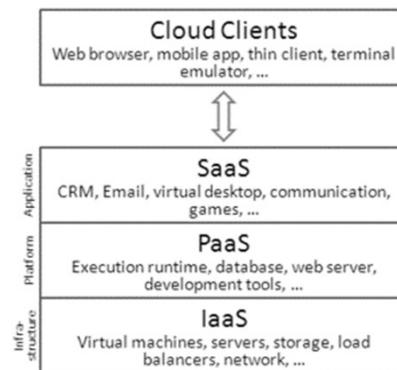
Application logic runs  
in the cloud



# Service Models of Cloud Computing

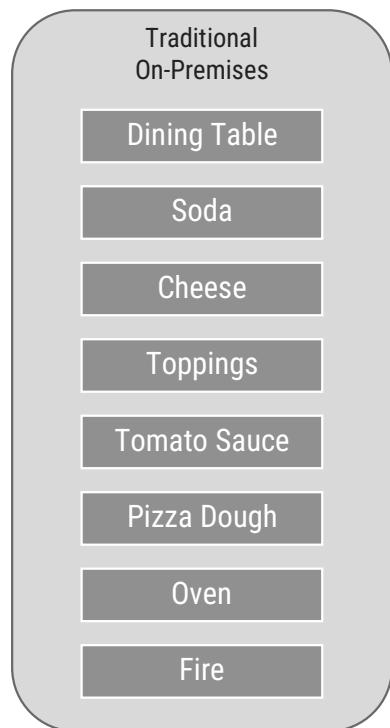


- ▶ There are **three** main service models of **cloud computing** – Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).
- ▶ Each model represents a different part of the cloud computing stack.
- ▶ Each type of cloud service provides you with different levels of control, flexibility, and management.

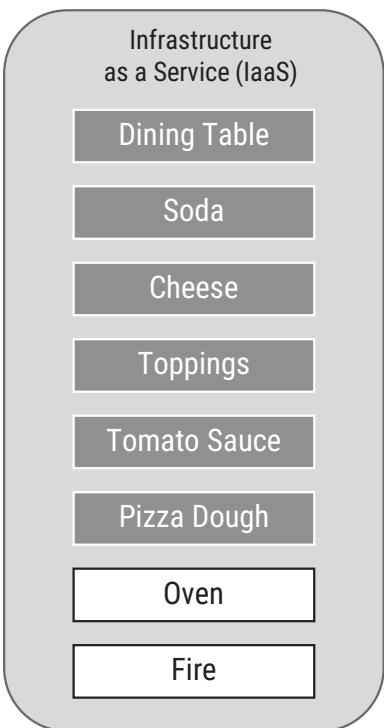


# Example of “as a Service”

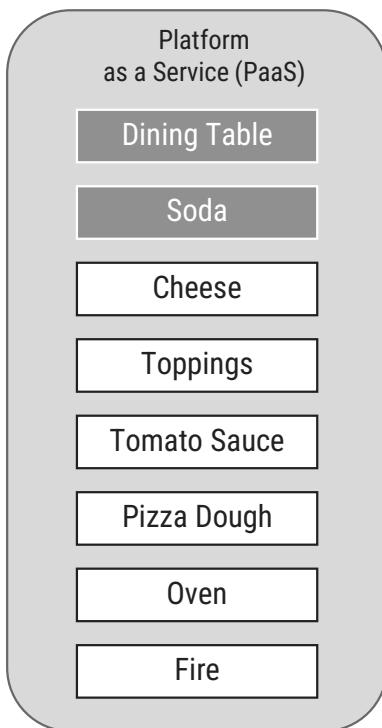
## Pizza as a Service



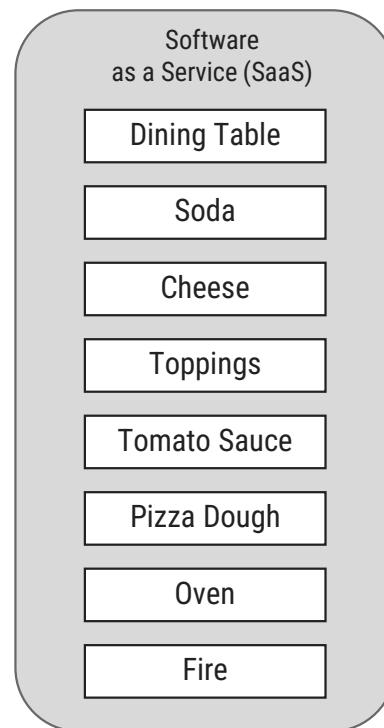
Made at Home



Take and Back



Pizza  
Delivered

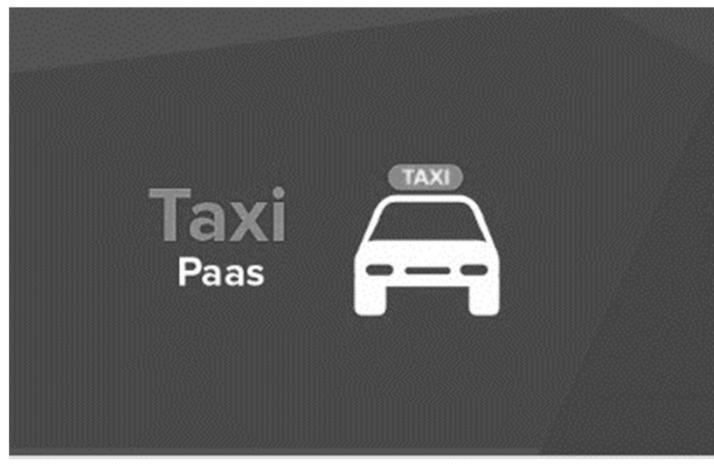
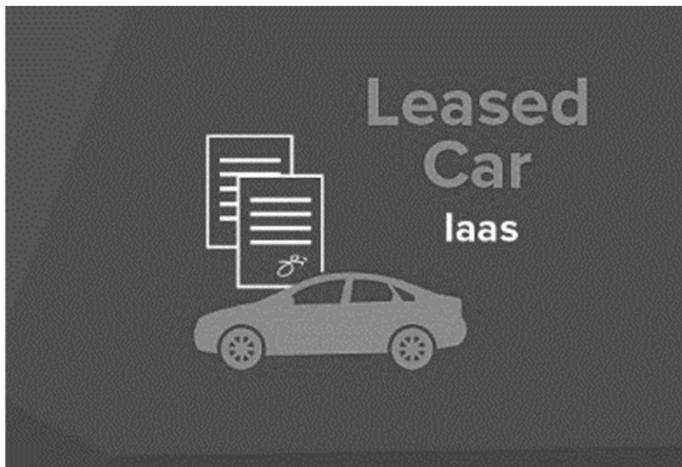


Dined at  
Restaurant

You Manage

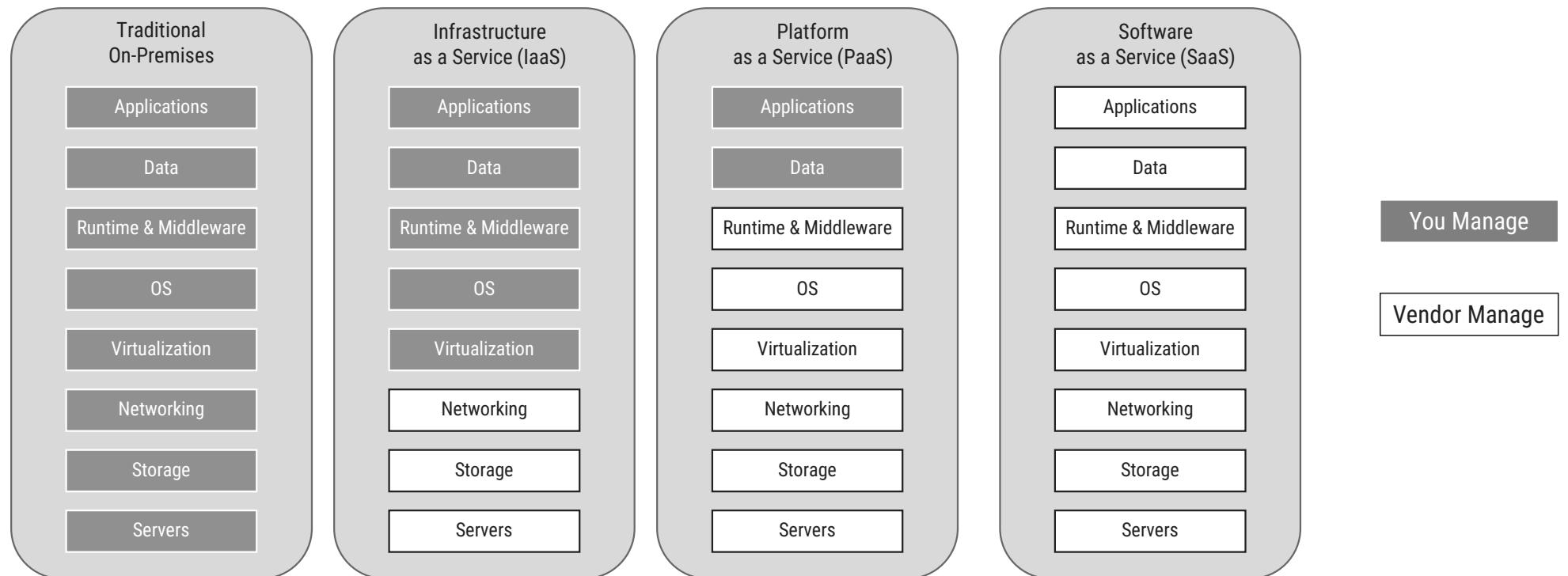
Vendor Manage

## Example of “as a Service”



**Vehicle  
as a  
Service**

# Cloud Services



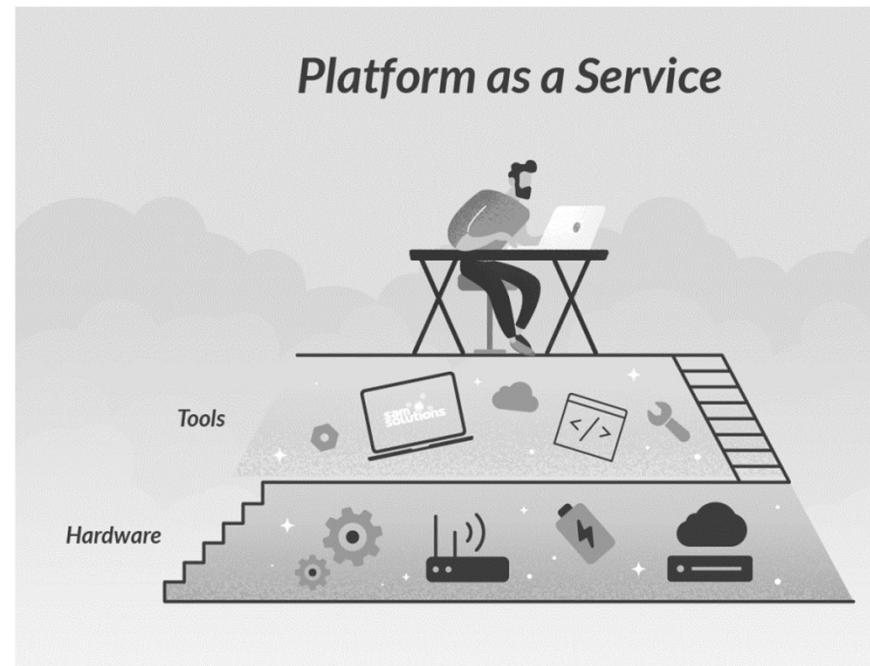
# Software-as-a-Service (SaaS)

- ▶ Complete software application as a service is provided to the user that is run and managed by the service provider.
- ▶ It can also be called application as a service as a pay monthly, yearly etc. subscription.
- ▶ In SaaS, user don't need to worry about software upgradation and management.
- ▶ A common example of a SaaS application is web-based email where you can send and receive email without having to manage the server that the email program is running.



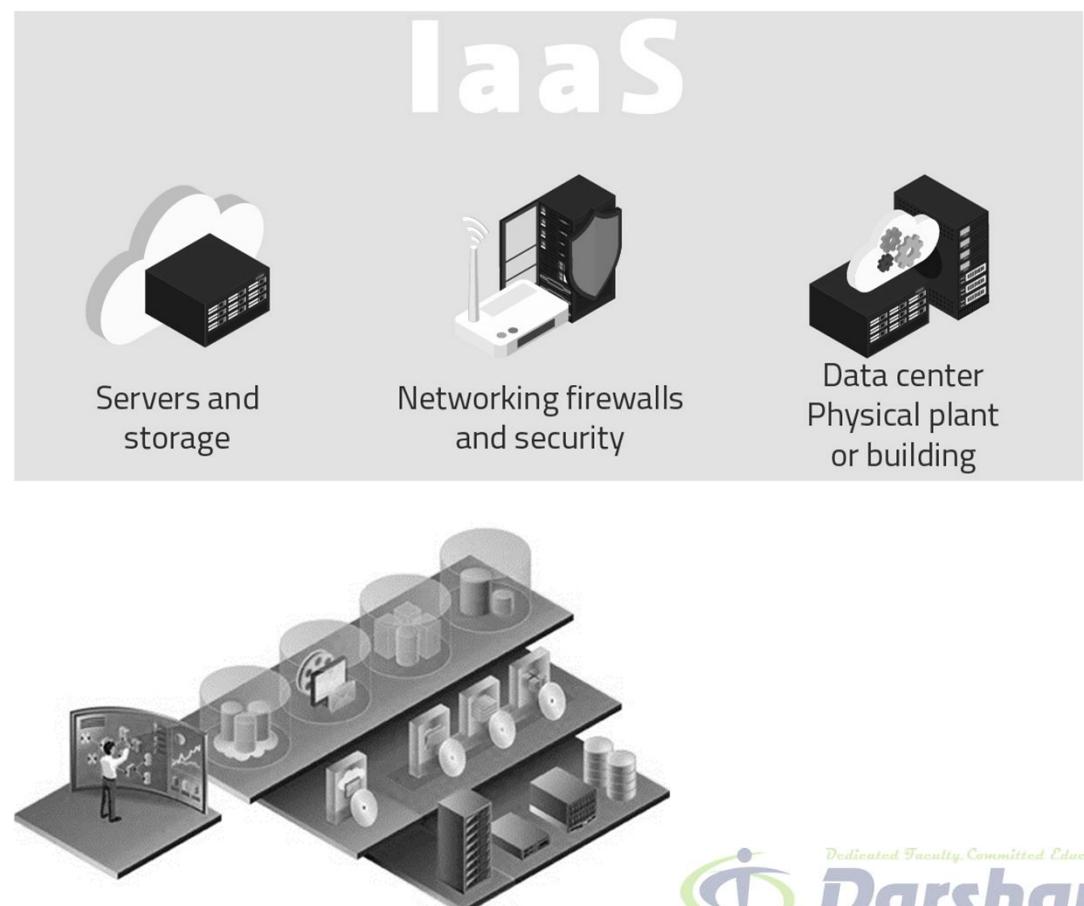
# Platform-as-a-Service (PaaS)

- ▶ It provides a development environment to application developers.
- ▶ Operating system, programming-language execution environment, database, web server, development tools, APIs, libraries, etc., will be provided by the cloud service provider.
- ▶ Users have to build, manage and maintain the applications.
- ▶ Application developers develop and run their software on a cloud platform instead of directly buying and managing the essential hardware and software layers.
- ▶ AWS Lambda, Google App Engine, IBM Cloud Foundry, Oracle Cloud Platform, Red Hat OpenShift, Zoho Creator.

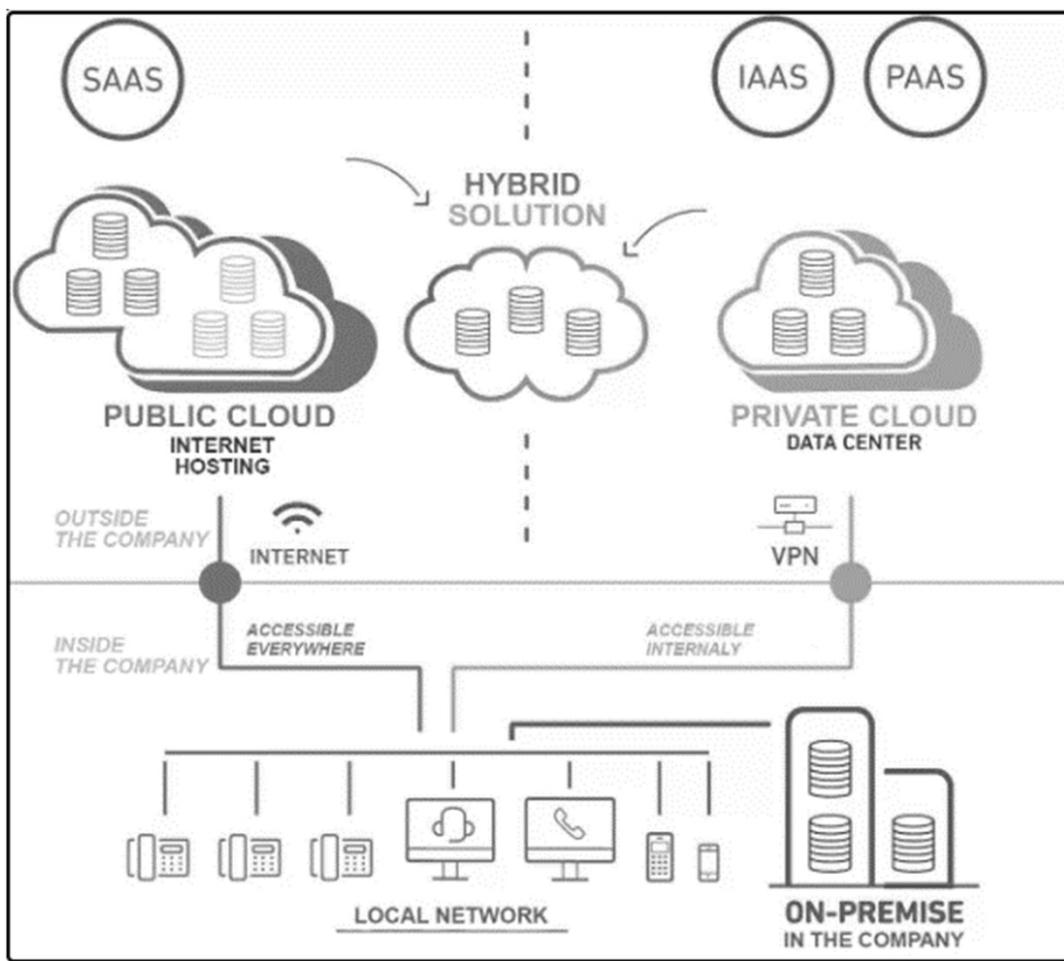


# Infrastructure-as-a-Service (IaaS)

- ▶ IaaS provides access to computing hardware, networking features, and data storage space.
- ▶ Where the consumer is able to deploy and run software, which can include operating systems and applications.
- ▶ The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications
- ▶ This service provides you with the highest level of flexibility.



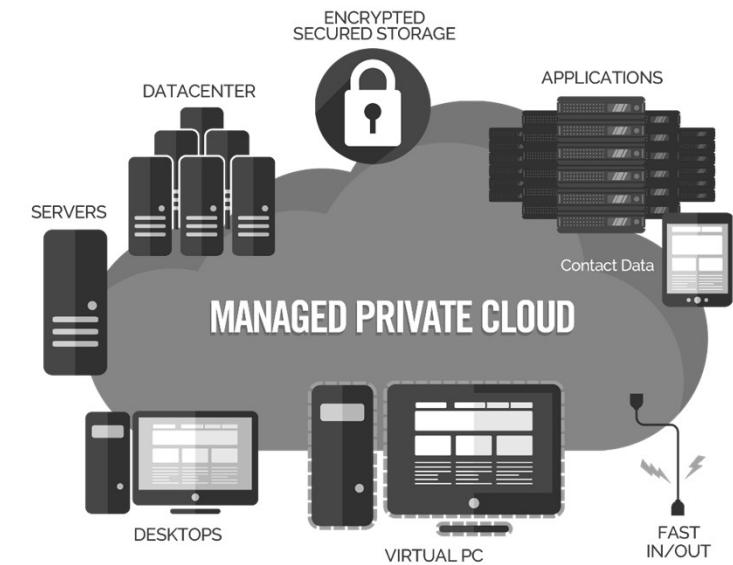
# Deployment Models of Cloud Computing



- ▶ Deployment models of cloud computing are categorized based on their location as public, private or hybrid cloud
- ▶ It is often possible to choose a geographic area to put the data “closer” to users.
- ▶ More the money you pay, better in comfort and service

# Private Cloud

- ▶ Private cloud provides computing services within the organization's private network and selected other users.
- ▶ In this cloud model all the hardware, software, datacenter, employees, infrastructure, etc., are maintained, monitored, and installed by the organization.
- ▶ This particular deployment model can be chosen wherever confidentiality matters the most.
- ▶ Advantages
  - High level of security and privacy
  - More control flexible in terms of deciding and managing the resources
- ▶ Disadvantages
  - Very Expensive
  - Need technical skill for maintaining and difficult to management
  - Policies and other related things are to be framed carefully to make sure that the data is safe



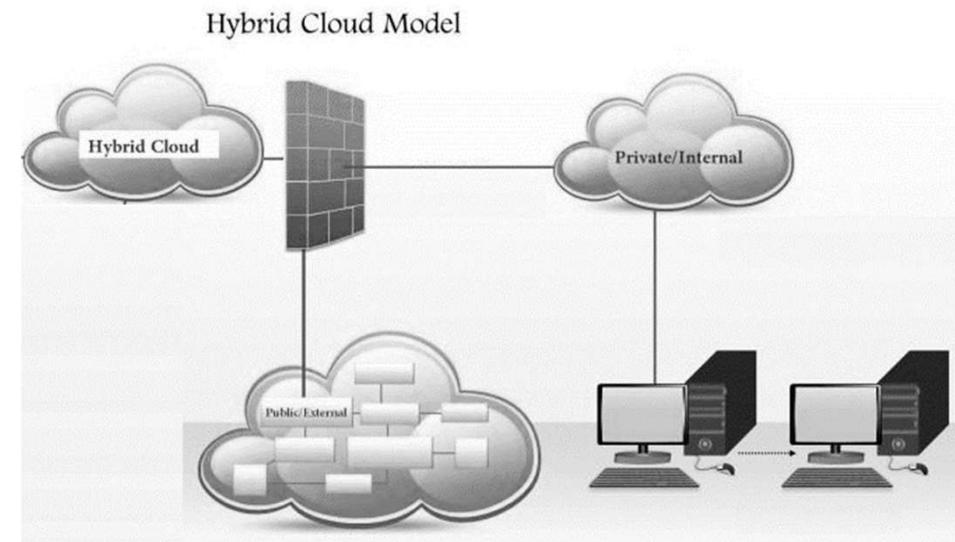
# Public Cloud

- ▶ The cloud resources are owned and managed by a third-party cloud service provider.
- ▶ Pay as per usage approach makes it most cost effective model.
- ▶ Advantages
  - Inexpensive model, no need to invest in setting up the infrastructure and maintenance
  - Less technical skill required
  - Customer support team can be reached on demand
  - Easily scaled up or scaled down based on requirements
  - High reliability—a vast network of servers ensures against failure.
- ▶ Disadvantages
  - Security and privacy issues are the major challenges
  - Less flexibility and controls

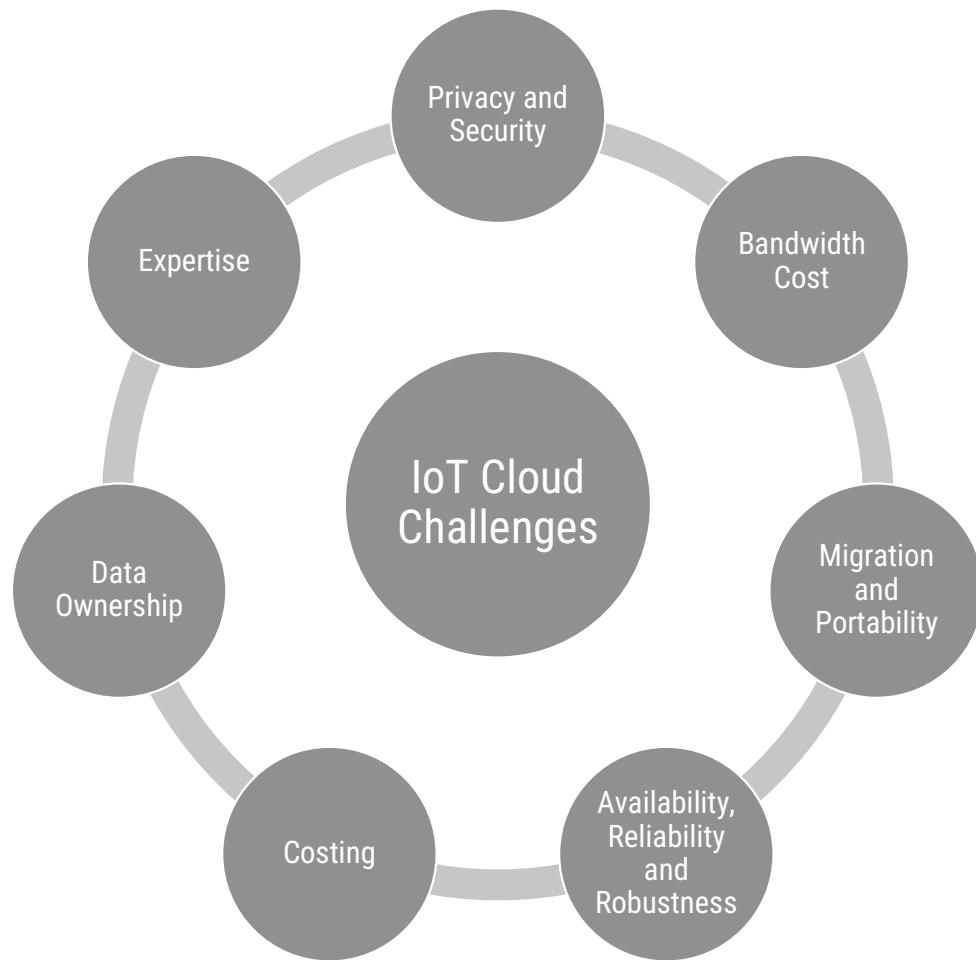


# Hybrid Cloud

- ▶ This deployment is a mix of both private and public cloud deployment
- ▶ The resources offered and managed are both in-house and third party based
- ▶ Advantages
  - Less investment needed to setup the infrastructure
  - Less technical skill required to manage and maintain the cloud
  - Customer support team can be reached on demand
  - Easily scaled up or scaled down based on requirements
- ▶ Disadvantages
  - Security and privacy issues are the major challenges
  - Less flexibility and controls compare to public cloud



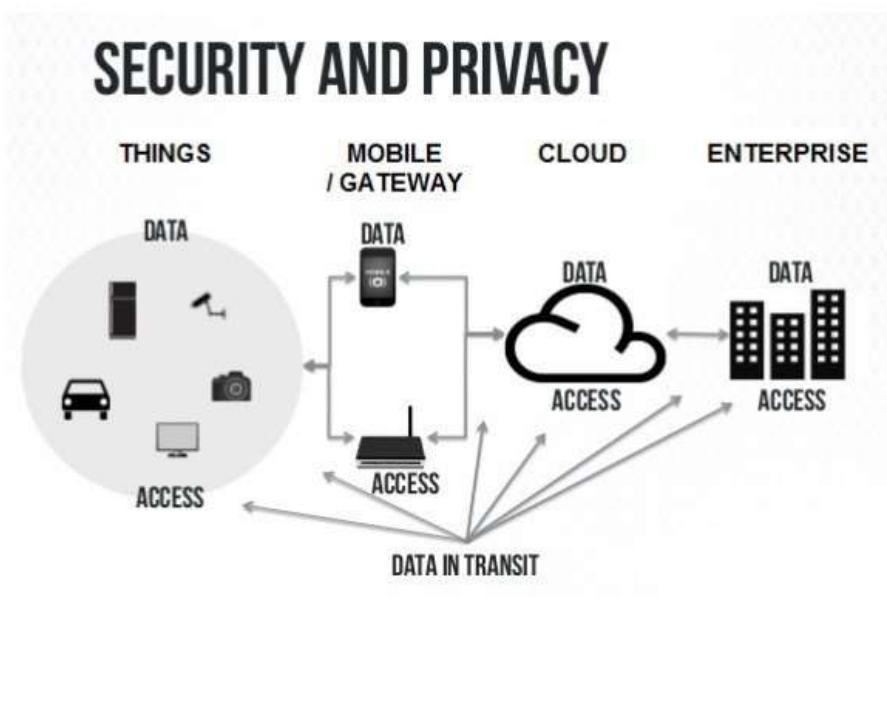
# IoT with Cloud - Challenges



- ▶ We have already discussed the challenges of IoT and cloud.
- ▶ These challenges can be increased when the IoT and cloud are integrated.
- ▶ Here we will discuss seven such challenges
  - Privacy and Security
  - Bandwidth Cost
  - Migration and Portability
  - Availability, Reliability and Robustness
  - Costing
  - Data Ownership
  - Expertise

# Privacy and Security

- ▶ Security is a major concern in the field of IoT.
- ▶ Valuable data goes into the cloud, outside the firewall this data becomes hackable.

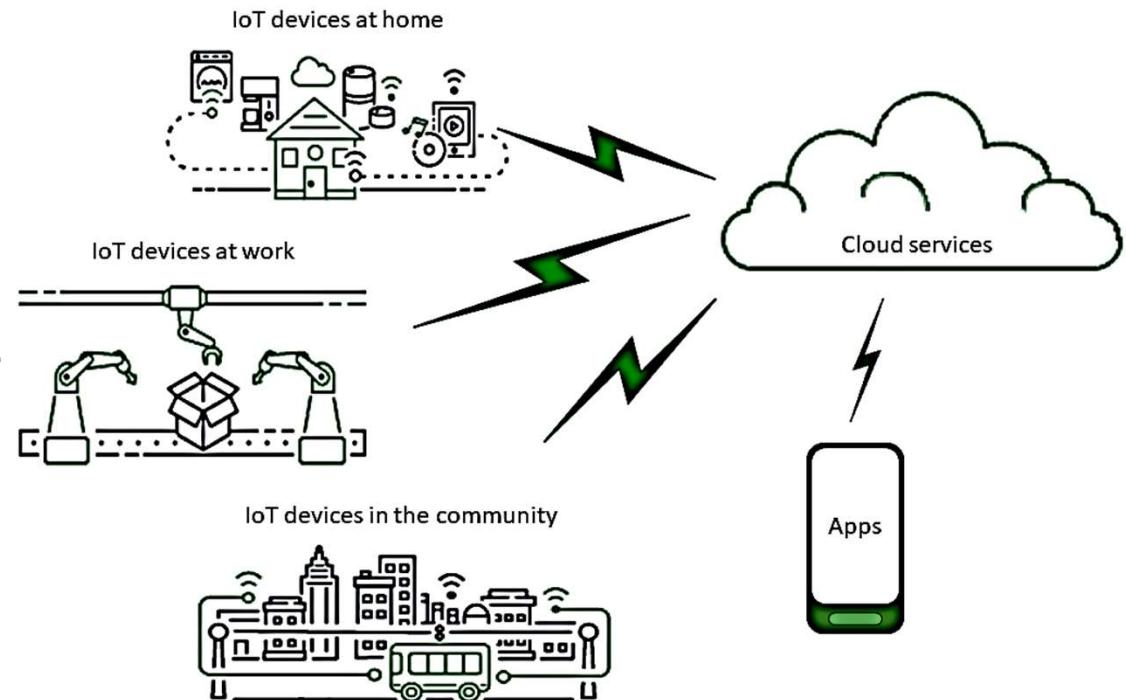


The following are solutions to this privacy and security challenge.

- Periodic monitoring of the network activities
- Select private cloud if the data is confidential
- To reduce the risk of being exposed, use recognized antivirus solutions.
- Before signing the contract with a cloud service provider, it is necessary to read and understand the regulations involved in the service being provided.

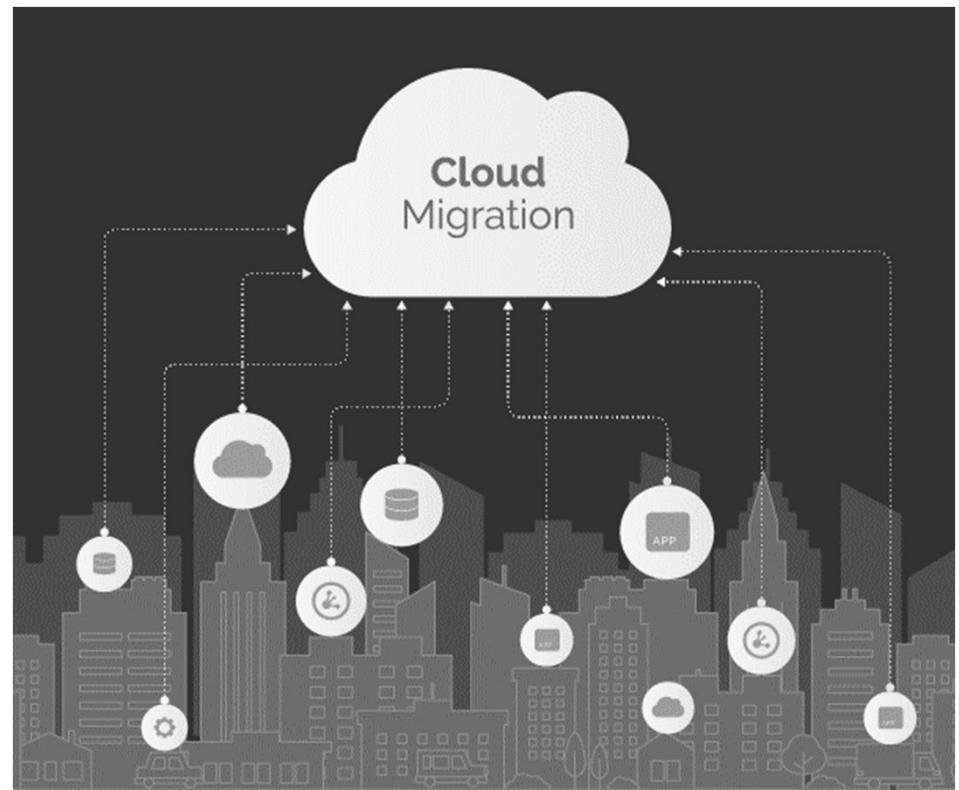
# Bandwidth Cost

- ▶ Bandwidth is one of the challenges because of continuous data transferring from the IoT devices.
- ▶ IoT is all about data, and in most cases, this would be big data. Hence, huge investment in storage is needed.
- ▶ Cloud computing is preferred for storage and processing in IoT.
- ▶ Small-scale IoT application demanding lesser resources.
- ▶ But if the application is data concentrated, then the investment in bandwidth would be considerable.



# Migration and Portability

- ▶ When data is to be moved to or migrate from the cloud, we have to take care of the followings.
  - How easy and safe is it to move the data?
  - How much downtime would this process require?
  - What is the strategy to migrate data to the cloud?
  - Will it be easy to select out of the cloud and take data back to the infrastructure?
  - How much would it cost?
  - Would there be support offered to migrate smoothly to another cloud service provider?
- ▶ All these challenges are doubled with IoT as the data comes from the various sensory nodes at a very high speed.



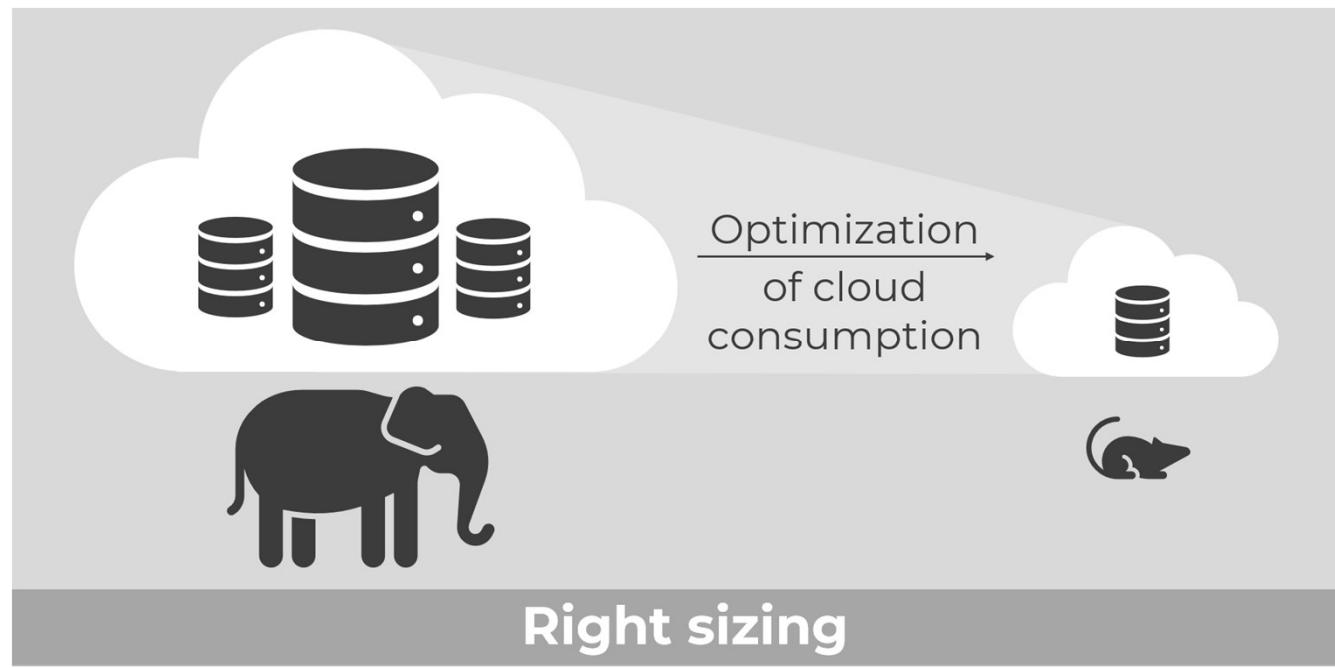
# Availability, Reliability and Robustness

- ▶ Continuous monitoring and reading of the data need to have continuous cloud service availability in IoT.
- ▶ In downtime, it would miss critical data so reliability of the process has to be monitored.
- ▶ The process should be robust towards handling data at different rates.
- ▶ Data could be flooded or slowed at anytime, in the both situations it should be handled effortlessly.



# Costing

- ▶ One of the main advantages of cloud is that it can scale up with rising demand.
- ▶ While it is scalable and flexible, an organization should plan its budget carefully.
- ▶ Wrong selection for subscription without having clear vision and planning, it may lead to unnecessary cost.



# Data Ownership

- ▶ The data stored by the user on the cloud is owned by the user.
- ▶ This means that the data is under the ownership of the person who generates it.
- ▶ However, when opting for cloud storage, the data is under the custody of the cloud service provider.
- ▶ Then, it appears that the service provider owns the data.
- ▶ When it comes to IoT, the data is generated at multiple points and ownership could lie with multiple participating parties.
- ▶ Hence, in IoT the ownership-related challenges are multiplied.



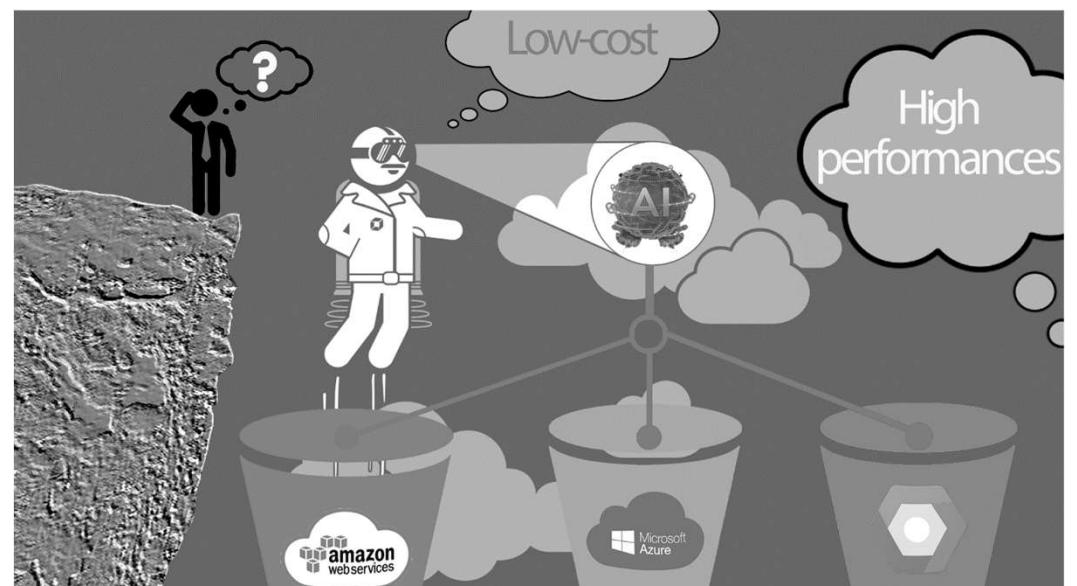
# Expertise

- ▶ To use the cloud with IoT requires a specific skill set.
- ▶ The cloud platform gets updated every now and then and so experts have to constantly upgrade themselves.
- ▶ Expertise is a definite challenge.
- ▶ When IoT and cloud comes together then it will be more challenging,
- ▶ To understand the sensors and at the same time, to be updated on cloud development is a challenging task.



# Selection of Cloud Service Provider: An Overview

- ▶ There are many parameters to select cloud service providers from the numerous service providers.
- ▶ It is advisable to consider following parameters while selecting the cloud service provider.
  - Certification and Standards Compliance
  - Financial Health of the Service Provider
  - Business and Technology Strength
  - Compliance Audit
  - Service Level Agreements
  - Reporting/Tracking
  - Costing and Billing
  - Maintenance Monitoring and Upgrade
  - Support
  - Security



# Criteria 1: Certification and Standards Compliance

- ▶ When a product adheres to the standards that are accepted widely, it is considered as a reliable product. Similarly,
- ▶ Cloud service providers (CSPs) are expected to comply with standards. Because products with standards are accepted widely, it is considered as a reliable product
- ▶ Industry accepted standards is the first criteria to select the CSP.
- ▶ Though there are many standards framed and followed by the industry, some of the main standards for cloud are ISO, Open Cloud Consortium (OCC), IEEE, SNIA (Cloud Storage Initiative).



## Criteria 2: Financial Health of the Service Provider

- ▶ The service provider should hold sufficient funding to operate business for a long period.
- ▶ If the service provider has healthy financial status and history of sustenance, then it is most unlikely to shut down.



## Criteria 3: Business and Technology Strength

- ▶ Having the technical expertise to sustain and adapt to a client's requirements is a key factor in selecting a CSP.
- ▶ Having just the technical skill and strength does not help; the CSP needs business skills as well to sustain.
- ▶ Business skills include growth planning, financial planning, and other factors that are required to sustain in the market.
- ▶ Sustenance = Technology + Business Skills

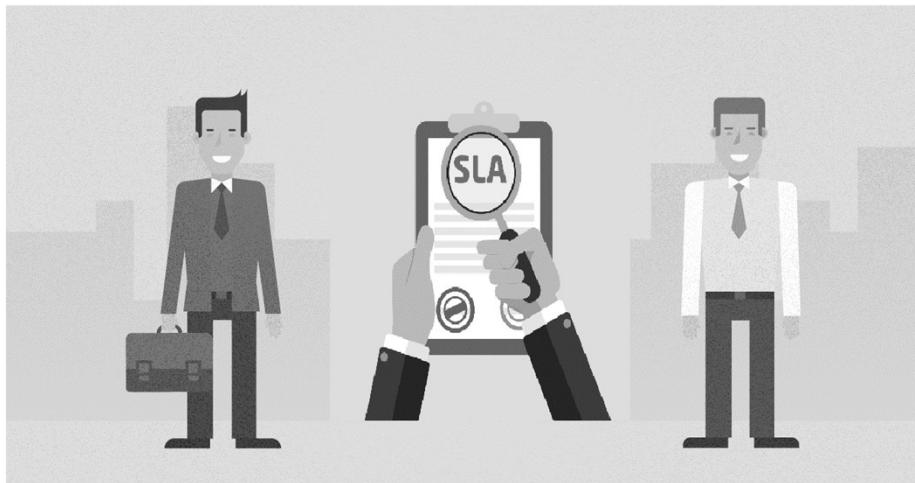


## Criteria 4: Compliance Audit

- ▶ The CSP must validate compliance with the client's requirement.
- ▶ Which should be done through a proper third party audit.
- ▶ This will enable transparency and perfect validation.



## Criteria 5: Service Level Agreements



- ▶ Service Level Agreements (SLAs) provides detail information about the services being provided.
- ▶ SLAs also indicates the real value that a customer gets out of them.
- ▶ SLAs serve as a legal contract, define the terms & conditions and the relationship between the two parties.

## Criteria 6: Reporting/Tracking

- ▶ The service provider should be capable for issuing a complete performance report, which also highlights the problems.
- ▶ This will enable the customer to understand the complete situation.



## Criteria 7: Costing and Billing

- ▶ The costing and billing should be transparent and should provide the complete details of the usage.
- ▶ It is expected to be automated with details of the complete resource utilization.
- ▶ It should come with having clarity along with the breakup.
- ▶ This means the billing should be transparent and for the usage only.
- ▶ This is a major factor in selecting the CSP.



## Criteria 8: Maintenance Monitoring and Upgrade



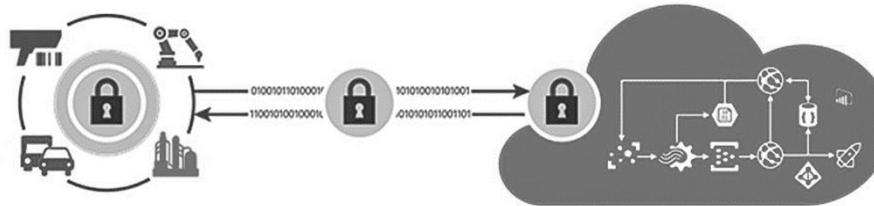
- ▶ It should be easy and less expensive to migrate to the CSP's environment.
- ▶ When there is an upgrade, it should be done with ease.
- ▶ Any maintenance should be easy and affordable.
- ▶ In short, it should be easier to install, manage, maintain, and upgrade.
- ▶ This upgrade includes migration from private to public to hybrid cloud, if needed.

# Criteria 9: Support



- ▶ Help and assistance should be provided when required.
- ▶ Support should be available based on the agreements and a dedicated resource.
- ▶ Support levels based on complexity of problem.
- ▶ Onsite support may be needed when clarifications cannot be offered over phone or online.
- ▶ Thus, support is a major deciding factor for selection of a service provider.

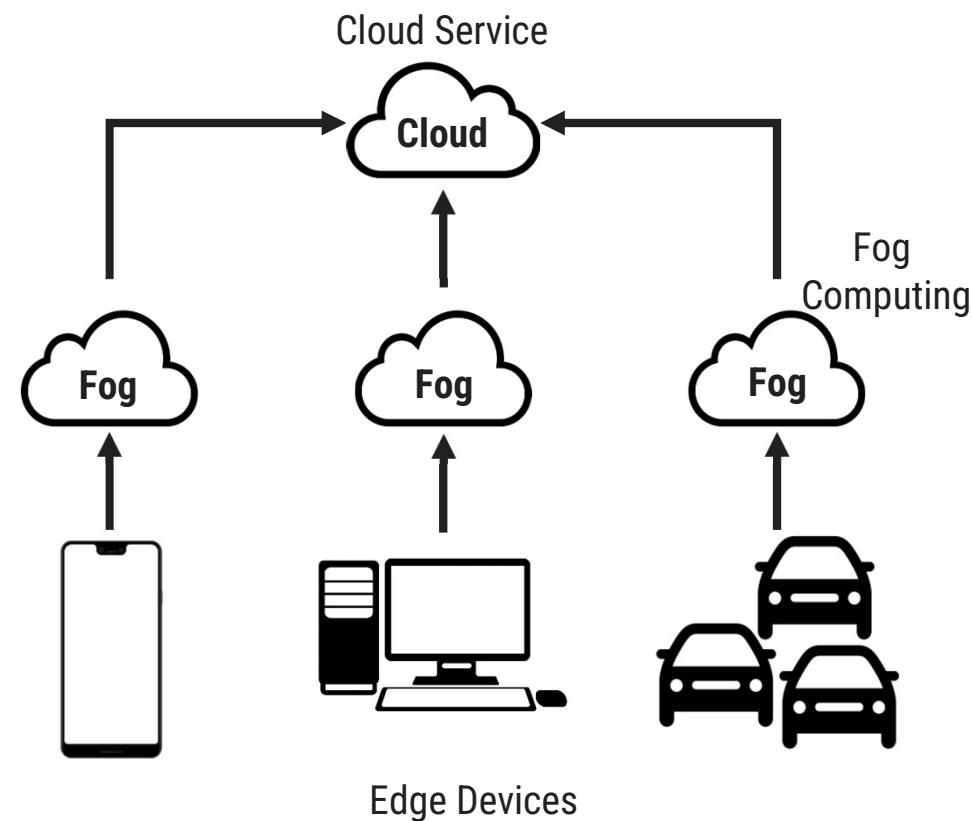
# Criteria 10: Security



- ▶ The infrastructure, both hardware and software should be secured.
- ▶ There should be defined policies about the security that should also be shared with a customer.
- ▶ This includes everything, from access restrictions to customer data.
- ▶ The data should be safe in case of a breakdown/failure.
- ▶ The recovery and backup options should be sound.
- ▶ The physical infrastructure has to be safeguarded as well.
- ▶ All these factors would require audit, which should be carried out by a third party.
- ▶ Security is the prime concern and cannot be ignored.
- ▶ Evaluation should start from this point.

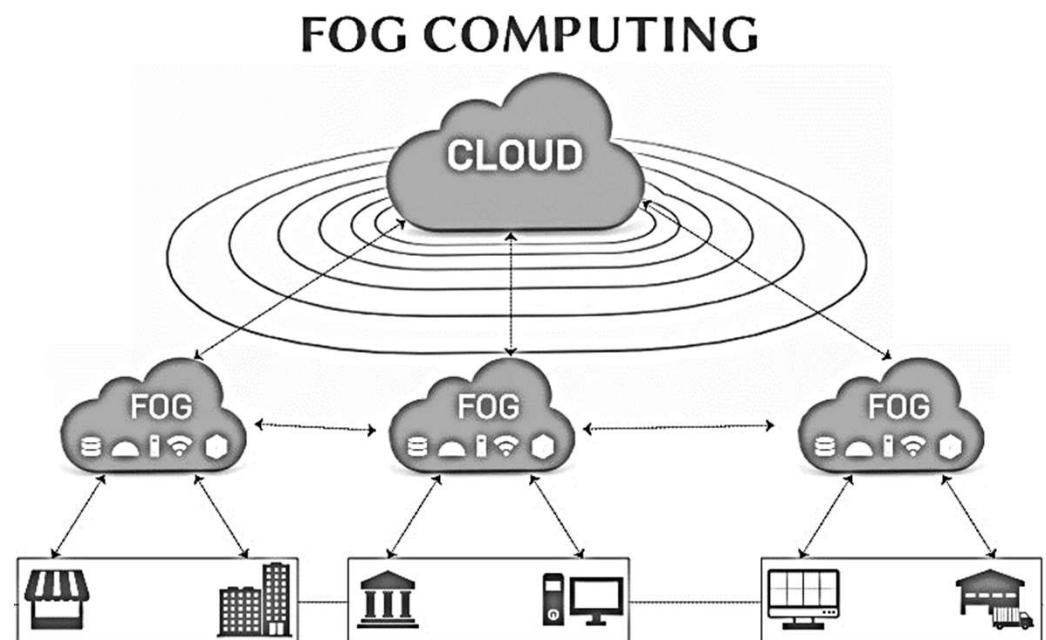
# Introduction to Fog Computing

- ▶ IoT is all about the data. The factors that affect data are the four Vs - variety, velocity, veracity and volume.
- ▶ All IoT applications require instant analysis and action.
- ▶ Most of the time, the action would be corrective in nature, it would be business critical.
- ▶ In case the data volume is high and it reaches the cloud after some delay.
- ▶ So we may lost the opportunity to use the data appropriately.
- ▶ In such cases, fog computing serves the solution.



# Introduction to Fog Computing

- ▶ The most sensitive data should be analyzed in the area closer to the place where it is generated.
- ▶ With fog computing, it is possible.
- ▶ Using fog computing we can process the data locally and to avoid the trouble by not sending the data to the cloud.
- ▶ Respond much faster because of data is moving locally so data travel is reduced considerably.
- ▶ It thus process the data in milliseconds.

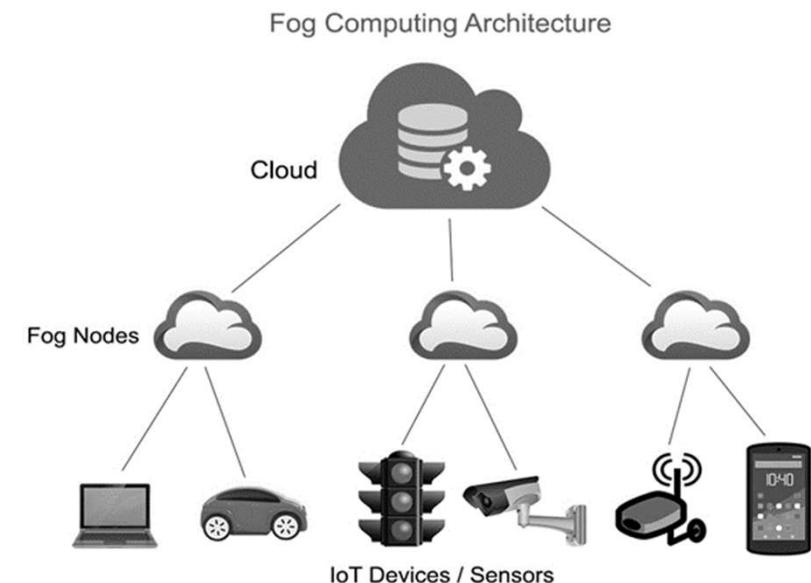


# Introduction to Fog Computing

- ▶ Only the required data will be sent to the cloud.
- ▶ This will be based on storage requirements and guidelines.
- ▶ Predictive analytics can also be carried out with the data stored in the cloud.
- ▶ The fog is below cloud, which means it is closer to the elements that generate data.
- ▶ After analysis, the data stored is pushed on to the cloud.
- ▶ Results in increased efficiency and safety both physical and asset safety.
- ▶ Some examples where faster response time is extremely important are factory or manufacturing line, oil and gas tube lines fault analysis, on-flight diagnosis, and healthcare.

# Working of Fog Computing

- ▶ Sensors/devices generate data transmit it to the middle layer, which is very close the data source.
- ▶ These nodes in the middle layer are capable of handling the data.
- ▶ This requires minimum power and lesser resources.
- ▶ All the data need not go to the cloud at the instant.
- ▶ Also, sensitive data gets processed very fast, which results in an instant response.
- ▶ Fog is not meant for hefty storage. It is still the cloud that does the task of storing big data.
- ▶ Fog is just an intermediary layer for faster data processing, and the faster response time.



# Summarize the concepts

## Concept of fog nodes

- ▶ It receives the data feed from the sensors, in real-time.
- ▶ Response time is minimal, ideally in milliseconds.
- ▶ Fog computing is transit, where data is stored for a limited time only.
- ▶ Data is then sent to cloud as a summary.
- ▶ It is important to note that not all data goes to the cloud.

## Concept of cloud computing platform

- ▶ It receives the data summary from the fog.
- ▶ Data prediction, data analytics, data storage, etc. takes place here.

# Benefits of fog computing model

- ▶ Minimal amount of data sent to the cloud.
- ▶ Reduced bandwidth consumption.
- ▶ Reduced data latency.
- ▶ Improved data security. When limited data goes to cloud, it is easier to protect it.
- ▶ Immediate processing of data in real time (this is very much needed in industrial applications).

# Difference between Edge and Fog Computing

- ▶ Both fog and edge are concerned with the computing capabilities to be executed locally, before passing it to the cloud.
- ▶ Both aim at reducing complete dependency on the cloud to perform computation.
- ▶ Analyzing data and processing it at the cloud is to be avoided.
- ▶ Both these reduce the time delay for making faster decision for real-time applications.
- ▶ The main difference between edge and fog computing is where data processing takes place.
- ▶ Edge computing is the computing carried out at the device itself, where all the sensors are-connected.
- ▶ In fog computing, data processing is moved to the processors that are connected to the local area network (LAN), making it a little farther from the sensors and actuators.
- ▶ Thus, the main difference between edge and fog computing is the distance.

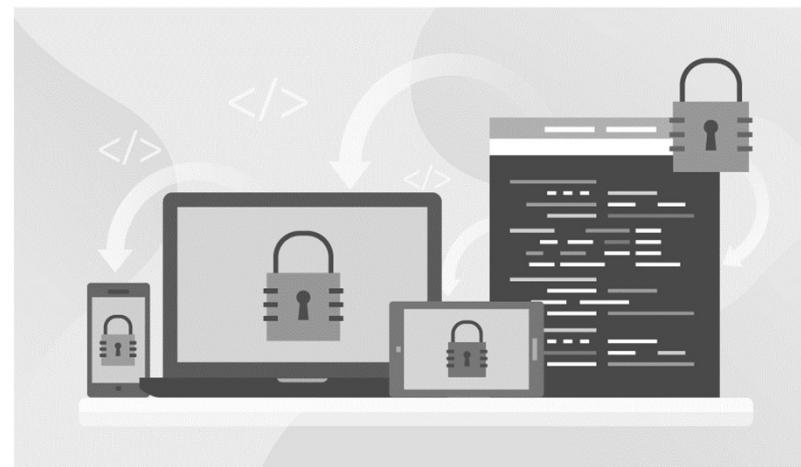
# Cloud Computing: Security Aspects

- ▶ The security of any computing platform including cloud computing depends on
  - Software security,
  - Infrastructure security,
  - Storage security, and
  - Network security,
- ▶ If any of these is compromised, it would result in security violation and could cause damages.
- ▶ Let us discuss these security aspects briefly.

# Cloud Computing: Security Aspects

## 1. Software Security:

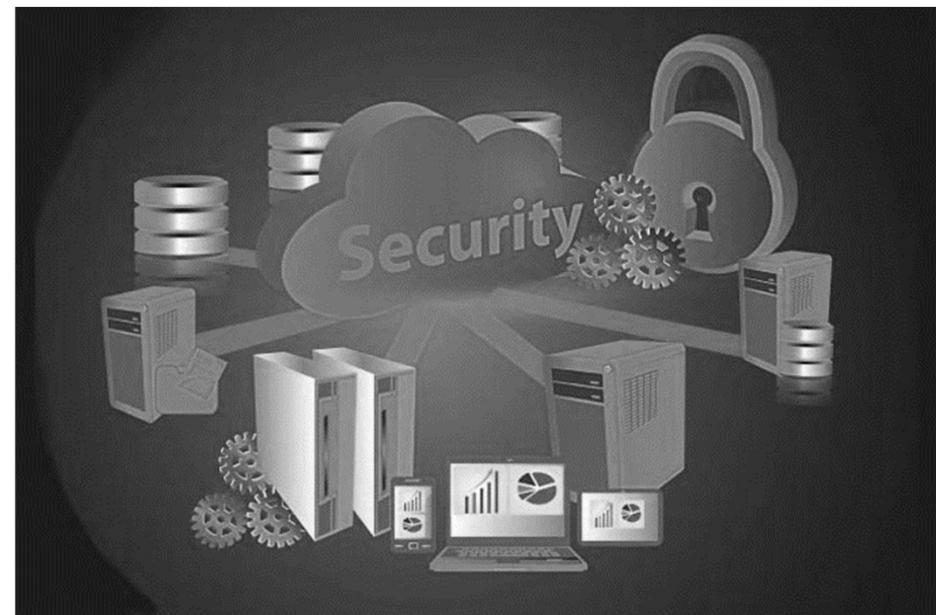
- ▶ Software is the core component and plays a vital role in presenting and ensuring a secure environment.
- ▶ If there are defects created/generated during the development phase, it is a software security threat.
- ▶ Defects such as simple software implementation defects, memory allocation, design issues, and exception handling all contribute to security issues.
- ▶ This can be ensured by complete and comprehensive testing carried out at all-stages.



# Cloud Computing: Security Aspects

## 2. Infrastructure Security:

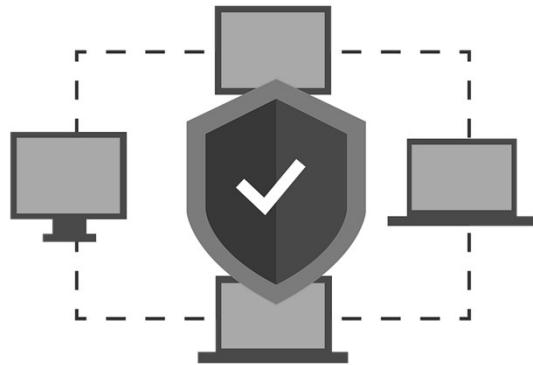
- ▶ Making sure that the infrastructure provided by the CSP is safe is a must.
- ▶ Third party could also contribute to the infrastructure.
- ▶ It is extremely important to check the security vulnerabilities with the infrastructure.
- ▶ All infrastructure related guidelines should be mentioned clearly in the agreements and should be made transparent to the customer.
- ▶ If data is damaged, everything is damaged and lost.
- ▶ Hence, care should be taken to protect the infrastructure.



# Cloud Computing: Security Aspects

## 3. Storage Security:

- ▶ It is important to be informed of who owns the data and the location where it is stored.
- ▶ Data leak, snooping, malware attacks, etc. are all threats to the stored data and can be listed under storage security.
- ▶ Appropriate antivirus software and periodic monitoring, should help protect the data.



## 4. Network Security:

- ▶ Data is stored in the cloud via the Internet, and hence all network threats become a possibility.

# Summary

- ▶ Cloud computing has become one of the most used technology components in modern day applications, which not only provides storage but also supports data analytics.
- ▶ Cloud services could be any one of the following:
  - **Software as a Service (SaaS)**: Complete software application as a service is provided to the user.
  - **Platform as a Service (PaaS)**: Development tools, APIs, libraries, etc. will be divided by the cloud service provider. User have to build, manage and maintain the applications.
  - **Infrastructure as a Service (IaaS)**: User should be provided with virtual machine support, where the user does not need to know and worry about the infrastructure. Everything should be taken care by the service provider. User will manage the machines, select the OS and underlying applications.
- ▶ The three deployment models generally used for public, private and hybrid
- ▶ Private cloud deployment model can be opted wherever confidentiality matters the most.
- ▶ When its come to public cloud deployment model, the cloud service provider owns all the resource which include hardware/infrastructure and software. Cloud service provider will take care of all resource management.

# Summary

- ▶ Hybrid development is a mix of both public and private deployment model. In this approach the resource offered and managed are both in-house and third party based.
- ▶ There are many challenges one could face while opting for cloud storage with IoT applications some of these are as follows:
  - Privacy and security
  - Bandwidth cost
  - Migration and portability
  - Reliability and availability
  - Costing
  - Data ownership
  - Expertise
- ▶ Selecting a CSP is not easy. Many parameters are to be considered before choosing the best option.
- ▶ With fog computing, it becomes possible to analyze the data at a place closer to where it is generated.

# Summary

► Fog computing provides the following advantages:

- Minimal amount of data send to cloud
- Reduce bandwidth consumption
- Reduce data latency
- Improve data security
- Immediate processing of data



Unit-5

# Application Building with IoT



**Prof. Kalpesh H Surati**  
Computer Engineering Department  
Darshan Institute of Engineering & Technology, Rajkot  

---

✉ Kalpesh.surati@darshan.ac.in  
📞 +91 99250 10033



# Applications with IoT

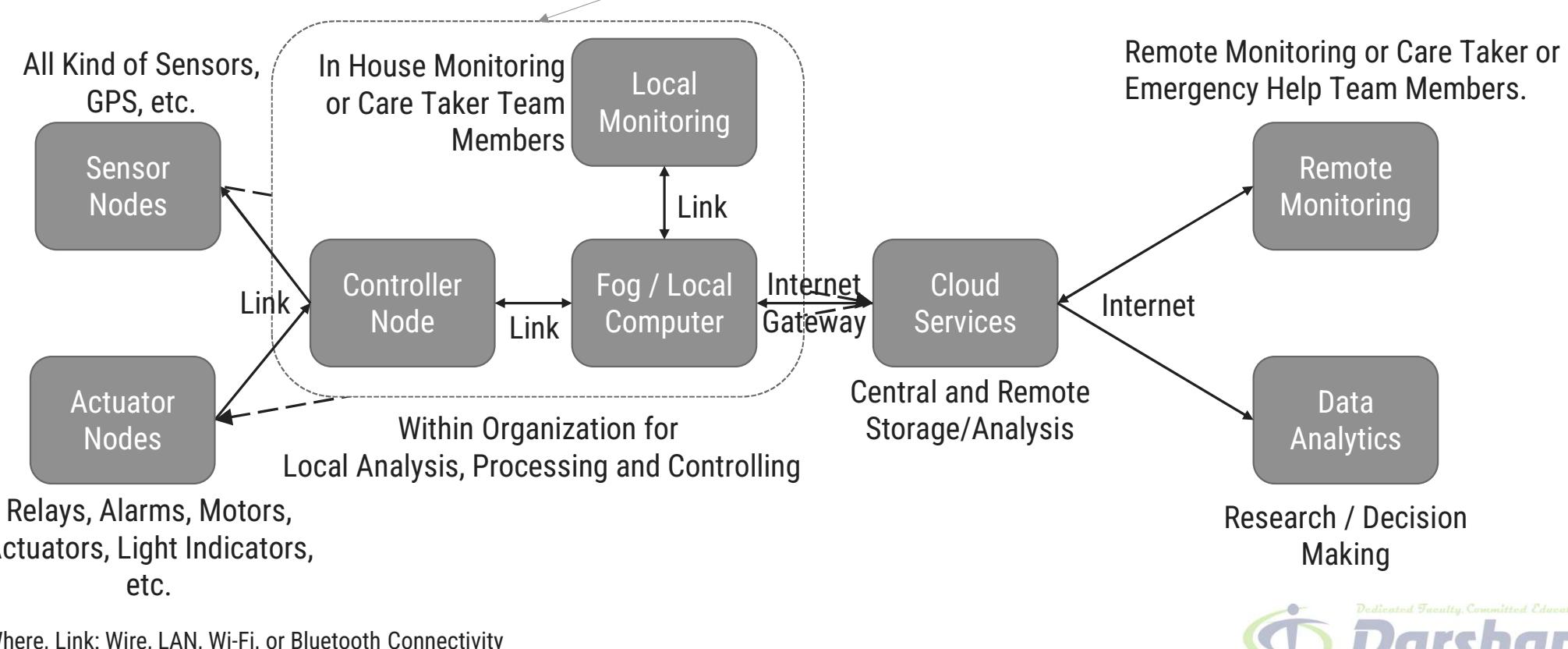
## Overview

- ▶ Using IoT we can build many applications but here, few IoT Applications will be discussed.
- ▶ We will cover following topics of the IoT applications.
  - **Overview** – General Practice
  - **Importance** – Why we need?
  - **Requirements** – Materials and Facility
  - **Architecture** – Functional Block Diagram
  - **How it works** – General Working of IoT Applications
- ▶ Before we explore them one by one, let's understand the **common architecture** of the IoT applications.



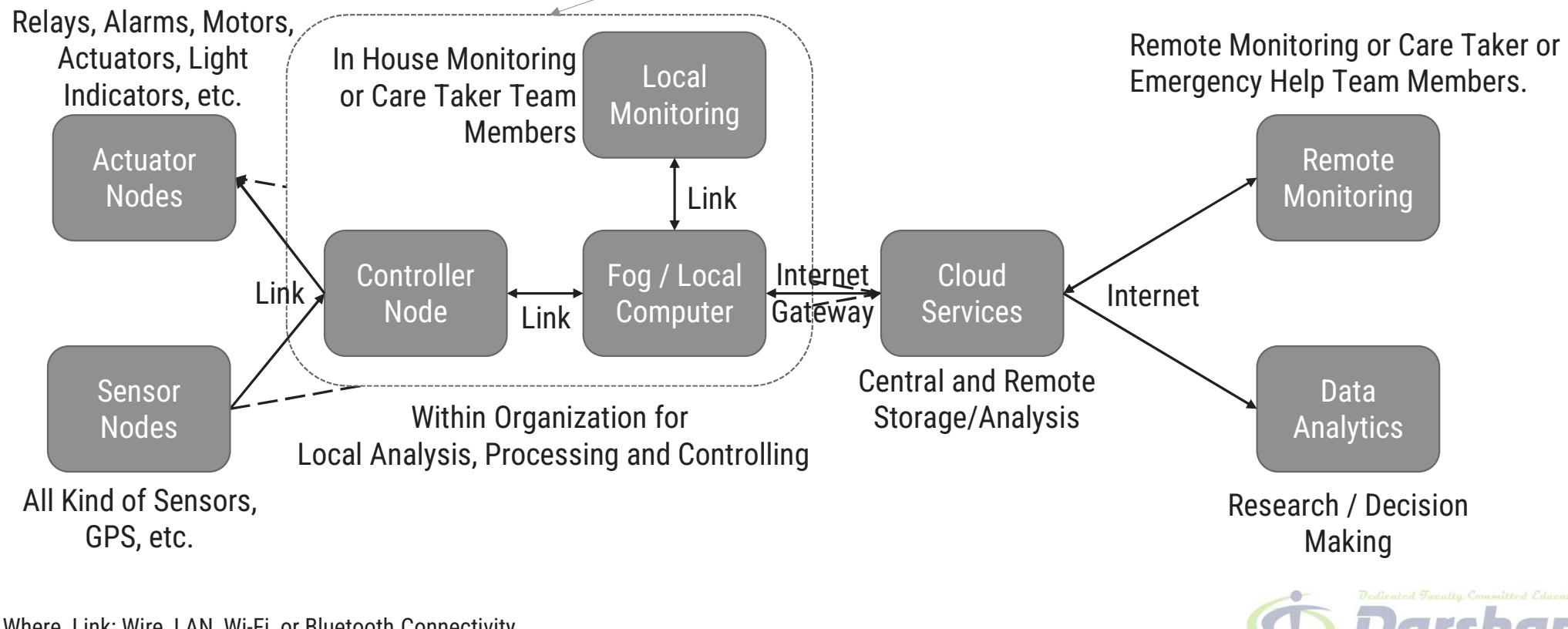
# Applications with IoT

## Common Architecture for IoT Application



# Applications with IoT

## Common Architecture for IoT Application

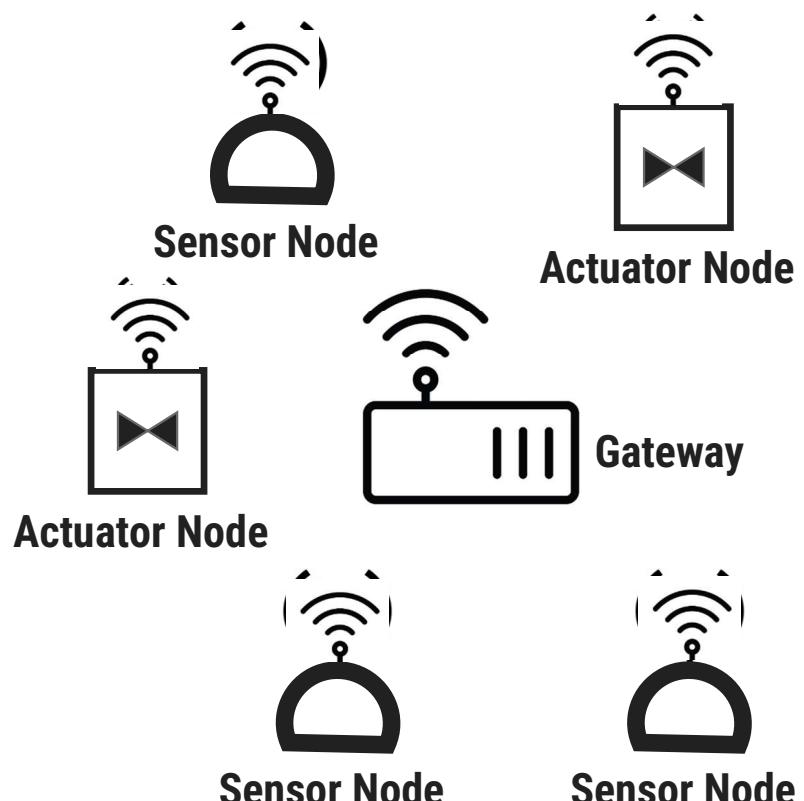


Where, Link: Wire, LAN, Wi-Fi, or Bluetooth Connectivity

# Applications with IoT

## How it works

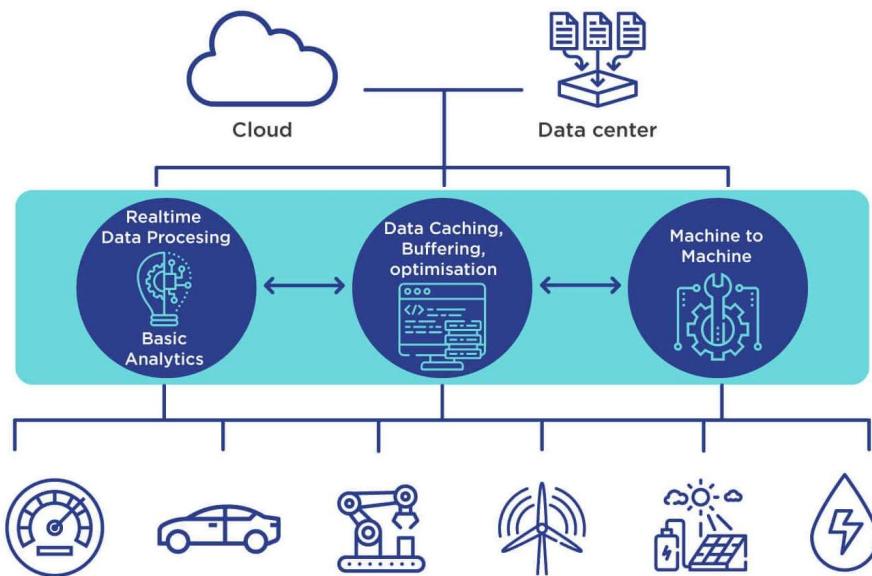
- ▶ All the **sensor** and **actuator** nodes should **be placed at appropriate place**.
- ▶ The **sensor** and **actuator** nodes should have **proper connectivity with controller** nodes **or** direct to the **cloud** server.
- ▶ The **sensor** nodes **sends** the sensed **data** to the **controller** nodes **or** directly to the **cloud** server.
- ▶ The **controller** node
  - Collects the **data from all the sensors**,
  - **Apply** some **business logic** on it as per the requirements and
  - **Send it to the Fog or local computer**, or directly to the **cloud server**.



# Applications with IoT

## How it works

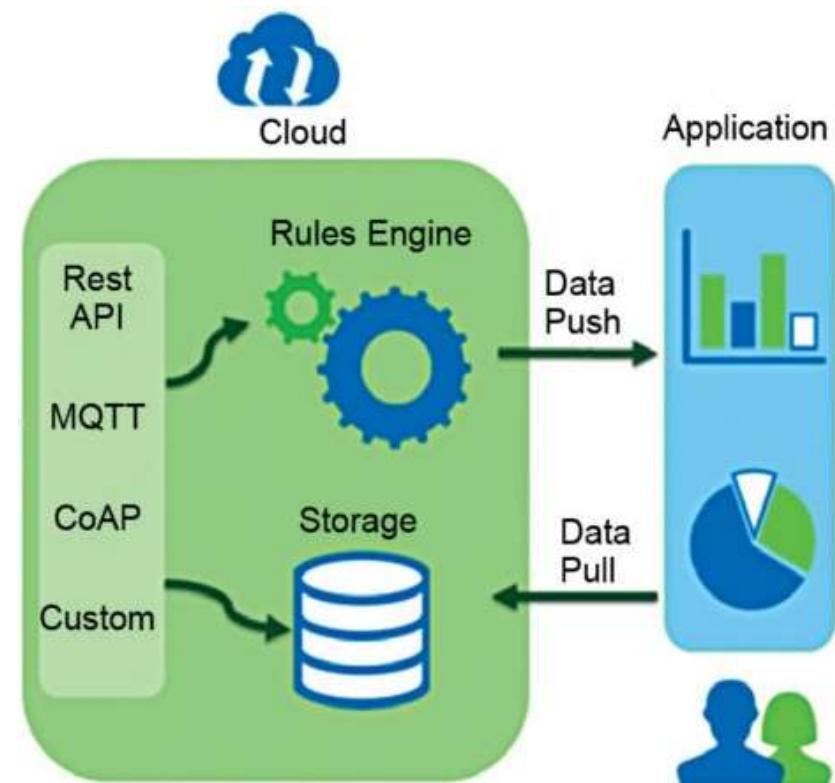
- ▶ All the **controller** nodes should have **connected with** the **Fog or local** computer **network**, or **directly** to the **cloud server** via Internet.
- ▶ **Fog or local computer**
  - Receives all the **data from** all the **controller** nodes,
  - **Execute** some **business logic** and
  - **Do analysis** on the collected data, and
  - **Send** the filtered limited **data to** the **cloud** computer as well as
  - **Send** the alert messages to the **local team**.
- ▶ The **Fog computer** should have **Internet connectivity for** the **cloud** communication.



# Applications with IoT

## How it works

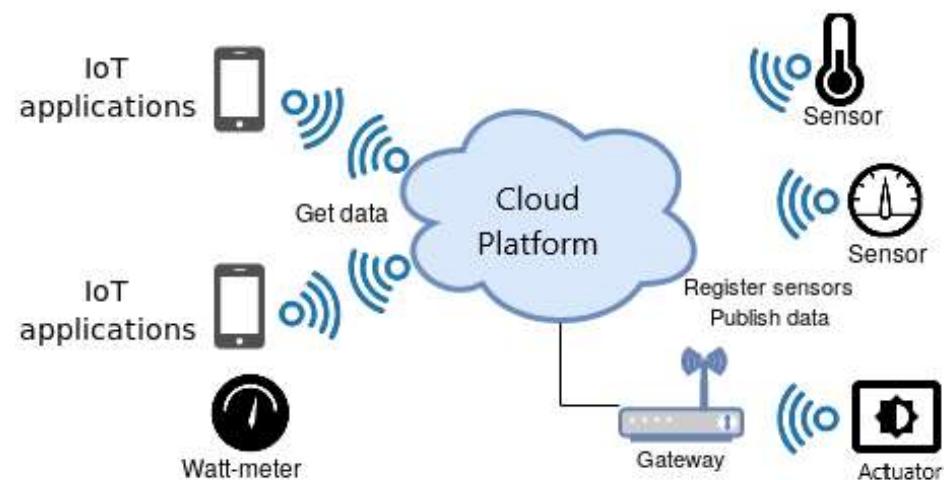
- ▶ All the in House Monitoring team members can access status of all the sensors from the Fog/Local computer.
- ▶ The cloud
  - Stores all the data received from the Fog/Local computer or directly from the sensors,
  - Execute some business logic and
  - Do analysis on the stored data.
- ▶ All those who have rights to access the cloud data can get desire information from it, like
  - Remote monitoring team members,
  - Data analytics team



# Applications with IoT

## How it works

- ▶ The **care taker** and **data analytics** teams may **execute corrective action** based on the analyzed data.
- ▶ They may
  - **Send command** to the local computer or to the **actuators** directly and
  - **Broadcast alert messages** to the **concern persons**.
- ▶ All the **actuator** nodes **receives command from the cloud** directly or from the **controller** or **local computer**.
- ▶ The “Local Analysis, Processing and Controlling” section may not be there in many IoT solution.



# Healthcare Application with IoT

## Overview

- ▶ The **healthcare** sector **consists** of **medical** and **related goods** and services.
- ▶ Healthcare sector **provides medical services** for **maintaining** or **improving** health via
  - Prevention
  - Diagnosis
  - Treatment
  - Recovery or cure of disease, illness, injury, and other physical and mental harms in people.
- ▶ Healthcare sector is much diversified and is full of opportunities in every segment.



# Healthcare Application with IoT

## Overview

- ▶ Healthcare is **delivered** by **health professionals**.
  - Medicine
  - Dentistry
  - Pharmacy
  - Nursing
  - Psychology
  - And other health professions are all part of health care.
- ▶ Healthcare has become one of India's largest sector in terms of revenue and employment.



# Healthcare Application with IoT

## Importance

- ▶ Before Internet of Things, **patients' interactions with doctors** were **limited to visits**.
- ▶ There was no way **doctors** or **hospitals** could **monitor** patients' health **continuously**.
- ▶ **IoT enabled devices** have made **remote monitoring** in the healthcare sector **possible**.
- ▶ IoT helps
  - To keep patients **safe** and **healthy**,
  - Empowering **physicians** to deliver **superlative care**.



# Healthcare Application with IoT

## Importance

- ▶ **Remote monitoring** of patient's health **helps** in
  - Reducing the length of **hospital stay** and
  - Prevents re-admissions.
- ▶ IoT has applications in **healthcare** that **benefit** to
  - Patients
  - Families
  - Physicians
  - Hospitals and
  - Insurance companies.



# Healthcare Application with IoT

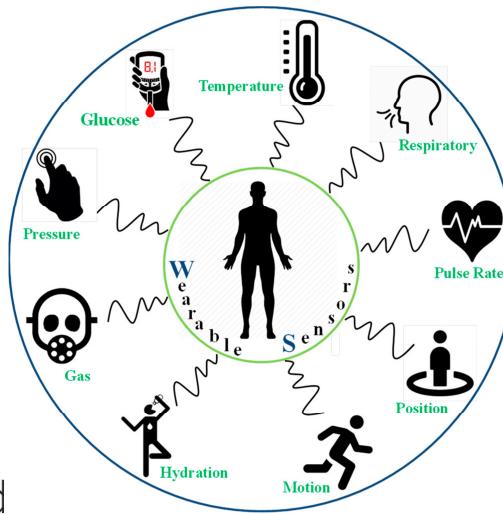
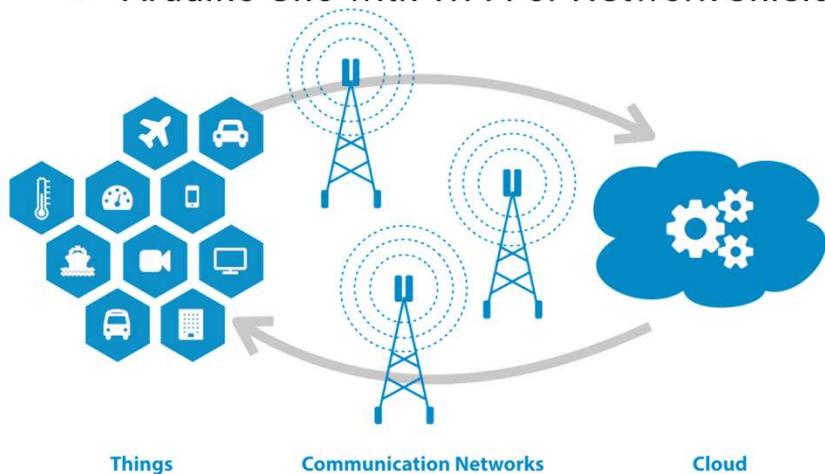
## Requirements for IoT Setup

### ► Sensors

- Heartbeat Sensor
- Oxygen Level Sensor
- Glucose Level Sensor

### ► Controller

- ESP8266 Node MCU or
- Arduino Uno with Wi-Fi or Network Shield



### ► For Monitoring and Analysis

- PC
- Mobile
- Tablet



### ► For Fog Computing

- Raspberry Pi or
- PC

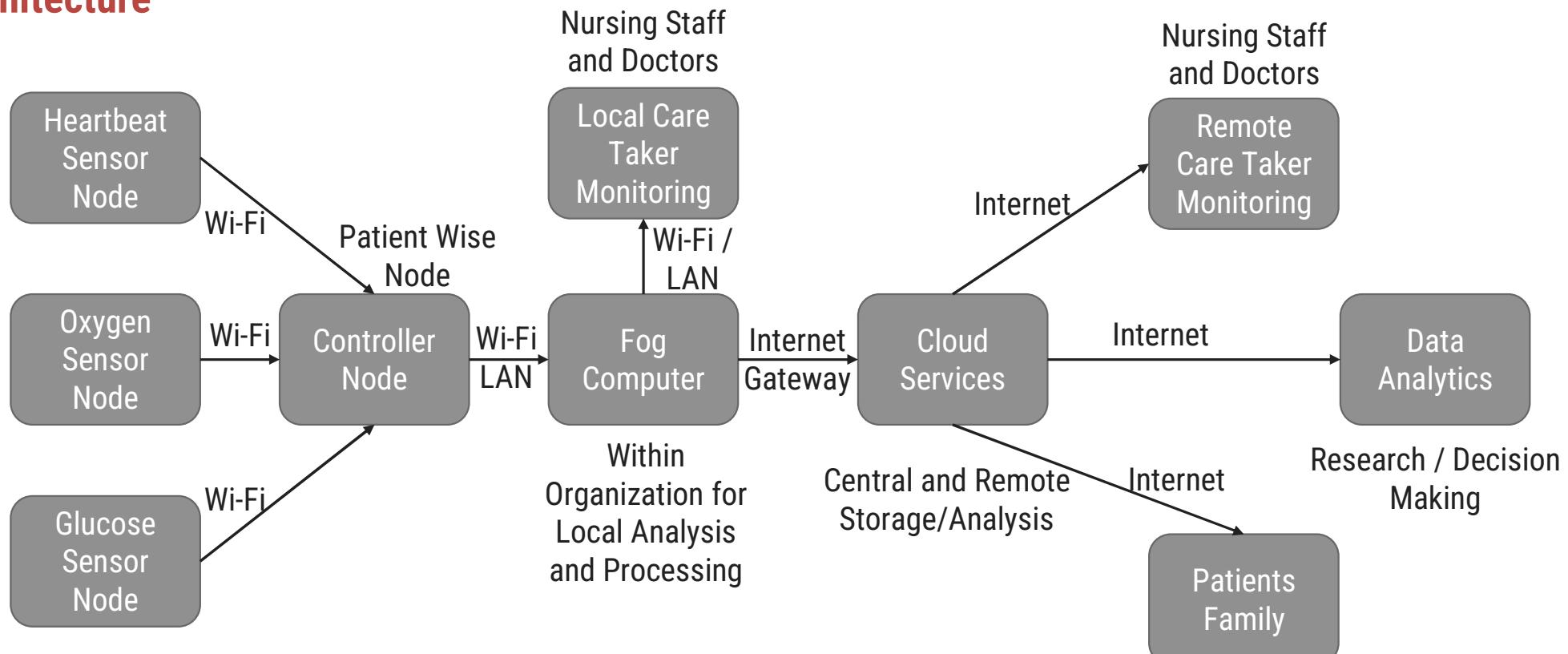
### ► LAN and WAN Connectivity

- Router and Switches
- Wireless Access Point

### ► Cloud Services

# Healthcare Application with IoT

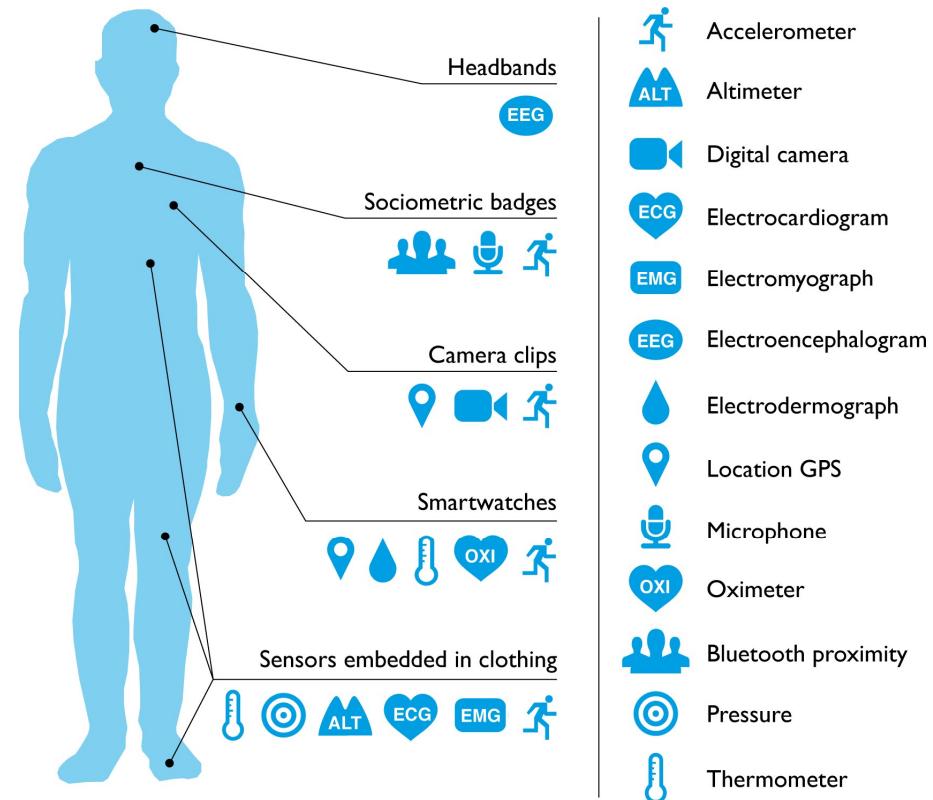
## Architecture



# Healthcare Application with IoT

## How it works

- ▶ All the **sensor** nodes should be **placed on patient body**, either direct or **using wearables**.
- ▶ The sensor nodes should have Wi-Fi connectivity with a controller node of the patient.
- ▶ The sensor nodes sends the sensed data to the controller node.
- ▶ The controller node collects
  - The data from all the sensors,
  - Apply some business logic on it as per the requirements and
  - Send it to Fog computer.
- ▶ All the controller nodes of all the patients should have connected with the Fog computer network.



# Healthcare Application with IoT

## How it works

### ► Fog computer

- Receives all the data from all the controller nodes
- Execute some business logic and
- Do analysis on the collected data, and
- Send the filtered limited data to the cloud computer as well as
- **Send the alert messages to the care taker team (nursing staff and doctors).**

► The Fog computer should have Internet connectivity for the cloud communication.

► All the **local care taker** team members **can be accessed health status of all the patients from the Fog computer.**



# Healthcare Application with IoT

## How it works

- ▶ The cloud
  - Stores all the data received from the Fog
  - Execute some business logic
  - Do analysis on the stored data
- ▶ All **those who** have **rights** to **access** the **cloud** data can **get health records of** the **patients**, like
  - Remote care taker team members
  - Data analytics team
  - **Patient family** members
- ▶ The **care taker** and data analytics teams may **execute corrective action** based on the data and alert messages.



# Healthcare Application with IoT

## Summary

- ▶ IoT has changed people's lives by enabling constant tracking of health conditions.
- ▶ Wearables connected devices like blood pressure and heart rate monitoring cuffs, glucometer etc. give patients attention.
- ▶ On any changes in the routine activities of a person, alert message sends to family members and health service providers.
- ▶ IoT enables healthcare professionals to be more watchful.
- ▶ Data collected from IoT devices can help physicians identify the best treatment process.
- ▶ IoT devices tagged with sensors are used for tracking real time location of medical equipment like wheelchairs, nebulizers, oxygen pumps and other monitoring equipment.
- ▶ IoT-enabled hygiene monitoring devices help in preventing patients from getting infected.
- ▶ It also help in asset management like pharmacy inventory control, and environmental monitoring, for instance, checking refrigerator temperature.

# IoT Application in Retail

## Overview



- ▶ Retail is the process of selling consumer goods or services to end-users for earning a profit.
- ▶ The retail sectors can be divided into
  - Food,
  - Clothing & Textiles,
  - Consumer Durables,
  - Footwear,
  - Jewelry,
  - Books-Music-Gift Articles, etc.

# IoT Application in Retail

## Importance

- ▶ In the retail industry customer plays key role
- ▶ The retail industry is rapidly transforming with IoT solutions.
- ▶ IoT is taking the center stage in the sector.
- ▶ Inventory management has been **an expensive** and **tedious process**.
- ▶ IoT system automatically monitors inventory.
- ▶ Send alert messages to managers if a certain item is **running low** or **going to expire soon**.
- ▶ IoT devices are also helpful for **avoiding**
  - **Oversupply**
  - **Shortage of goods** and
  - **Thefts in stores**.



# IoT Application in Retail

## Importance

- ▶ Customers are notified about **discounts** and **offers in real-time**.
- ▶ It give data about **customer behavior** and **routes**.
- ▶ IoT helps to increase
  - **Customer loyalty,**
  - **Boost sales,**
  - **Offer a personalized experience, and**
  - **Improve inventory management.**



# IoT Application in Retail

## Requirements for IoT Setup

### ► Sensors

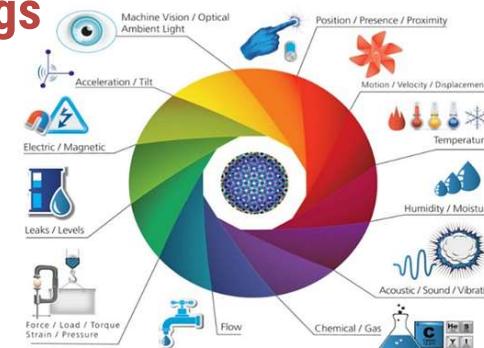
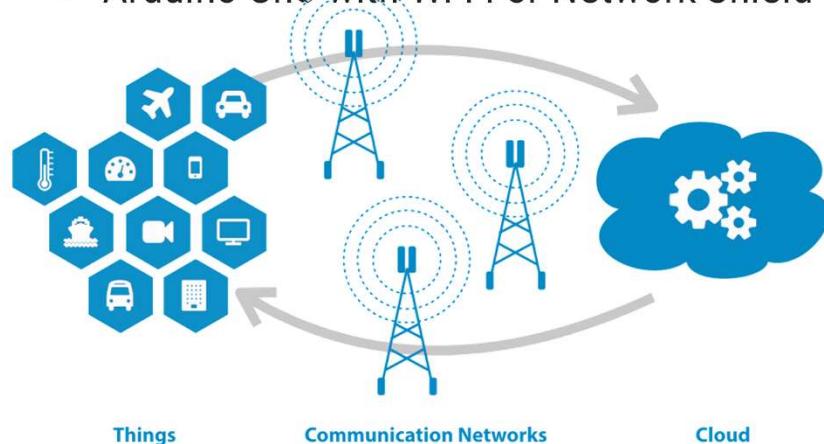
- Touch Sensors
- Gas Sensors
- Fire Sensors
- IR Sensors

### ► Actuators and Tags

- Alarm
- Water Sprinkler
- Beacons
- RFID Tag and RFID Reader

### ► Controller

- ESP32 MCU or
- Arduino Uno with Wi-Fi or Network Shield



### ► For Monitoring and Analysis

- PC
- Mobile
- Tablet



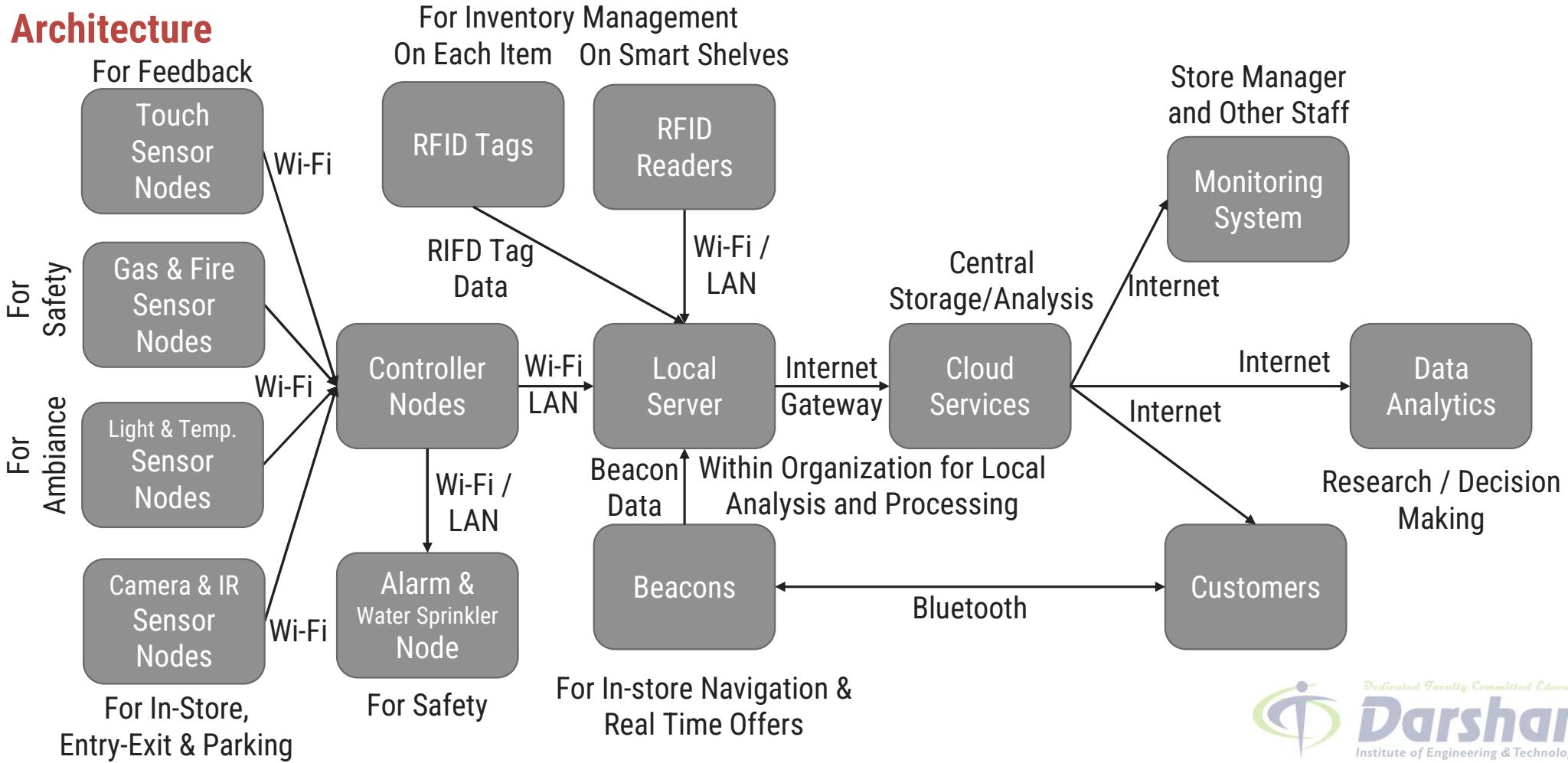
### ► LAN and WAN Connectivity

- Router and Switches
- Wireless Access Point

### ► Cloud Services

# IoT Application in Retail

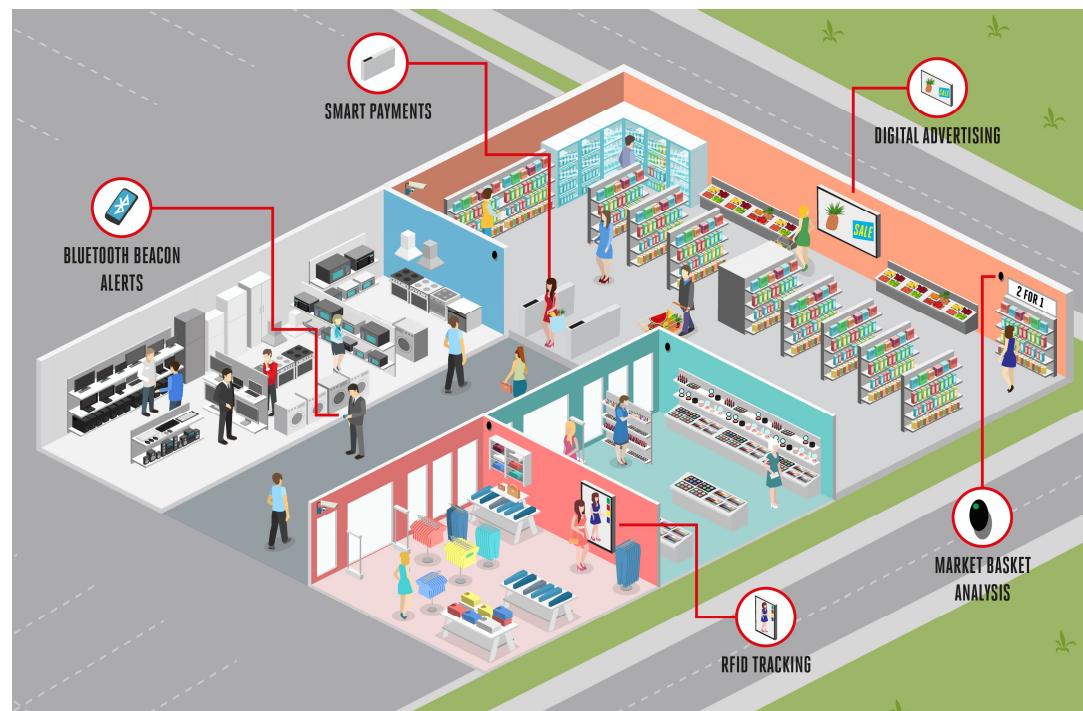
## Architecture



# IoT Application in Retail

## How it works

- ▶ All the **sensor nodes** and **Beacons** should be **placed** at various places in the store, washroom and parking area.
- ▶ The sensor nodes should have Wi-Fi connectivity with controller nodes.
- ▶ The sensor nodes sends the sensed data to the controller nodes.
- ▶ The controller nodes
  - Collects the data from all the sensors,
  - Apply some business logic on it as per the requirements and
  - Send it to local computer.



# IoT Application in Retail

## How it works



- ▶ **RFID readers** should be **placed at** items display **shelves** to detect item presence in the shelves **for the stock management**.
- ▶ All the **controller nodes** and **RFID readers** should have **connected** with the **Local Area Network**.
- ▶ **RFID tags** should be **attached with** each and every **item** which are likely to be **monitored**.
- ▶ Using the **RFID** tags one can **monitor location** and **stock of** the **item**.
- ▶ **RFID** tags is also **useful for** auto billing and **queue less exit**.

# IoT Application in Retail

## How it works

- ▶ **Touch sensors** are used for customers' **feedback** about **ambiance** and **cleanliness at** various places like **washroom, trial room** etc.
- ▶ **Light and Temperature sensors** are used for **ambiance monitoring** which leads to **energy saving** and **customer comfort**.
- ▶ **Camera and IR sensors** are placed for **monitoring people density** at various places, person **in-out ratio** for **store** and the **parking** area.
- ▶ **Gas sensors** are used for **monitoring** various **gases levels** in the store.
- ▶ **Fire sensor** are used for **detecting fire** in the store and **send** the **alert** signal to the **Alarm & Water Sprinkler Node**



# IoT Application in Retail

## How it works

- ▶ An **alarm** system and the **water sprinkling** system are used **for fire safety**, should have connected with a controller node
- ▶ Local Server
  - Receives all the data from all the controller nodes,
  - Execute some business logic and
  - Do analysis on the collected data, and
  - Send the essential data to the cloud computer.
- ▶ The server computer should have Internet connectivity for the cloud communication.
- ▶ The cloud
  - Stores all the data received from the local server,
  - Execute some business logic and
  - Do analysis on the stored data.



# IoT Application in Retail

## How it works

- ▶ All those who have rights to **access** the **cloud data** can **get desire information** and **alerts**, like
  - Store manager
  - Staff members
  - Data analytics team
  - customers
- ▶ The **store manager** and **data analytics teams** may **execute corrective action based on** the **data**.



# IoT Application in Retail

## How it works

### ► Beacons, a Bluetooth Low Energy (BLE) device

- Automatically connect with customers' smartphones when they entered in the coverage area.
- Send push notifications directly to the customers.

### ► Beacons can be installed at various places

- For customer in-store navigation,
- To know foot fall in particular area of the store, and
- Collecting data about customer behavior and routes.

### ► Customers are notified about discounts and offers in real-time using the beacons system.



# IoT Application in Retail

## Summary

- ▶ IoT will dramatically transform and innovate the retail industry in the coming years.
- ▶ The integration of IoT solutions will enable retail companies,
  - To create successful marketing campaigns based on customer behavior,
  - Deliver high-quality services,
  - Improve inventory management, and
  - Reduce operational costs.

# Driver Assistance Application with IoT

## Overview

- ▶ **Vehicle manufacturing** companies are trying to **make vehicles** more **comfortable** and **safe for** their **passengers**.
- ▶ They are experimenting to **design** a **system** that could **monitor** and **alert** the **user suffering from highway hypnosis** or white line fever.
- ▶ As well as they need an IoT based system to **prevent accidents** caused **by drowsy driver**.



# Driver Assistance Application with IoT

## Importance

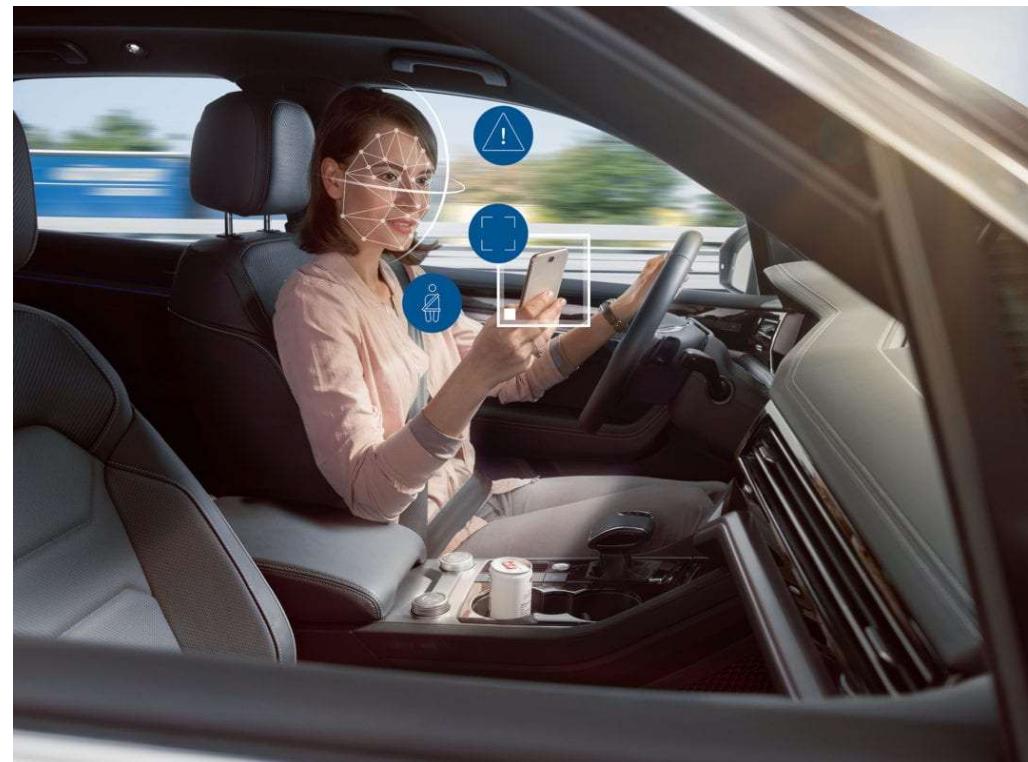
- ▶ Driver's **drowsiness** is one of the **major factor** is reported in **road accidents**.
- ▶ Drowsiness is a kind of **sleeping sensation** which can lead for closing eyes.
- ▶ Drowsiness **make** one **physically inactive** and **insensitive** to surroundings.
- ▶ So we **need to monitor** driver's **consciousness** along with his **acceleration pattern** and **steering angle**.
- ▶ If any **deviation** is **detected**, a system should be there to **alert** the **driver**.
- ▶ A process of detecting the deviation and drowsiness of the driver should be developed.



# Driver Assistance Application with IoT

## Importance

- ▶ **Using Image Processing and Machine Learning (ML)** technology a system can be developed.
- ▶ The system **detect** whether the **driver** has **deviated** or not, **by** monitoring the driver's
  - **facial movements**,
  - **Eye Aspects Ratio (EAR)** for drowsiness,
  - **acceleration** pattern and
  - change in the **steering angle**



# Driver Assistance Application with IoT

## Requirements for IoT Setup

### ► Sensors

- Ultrasonic Sensor

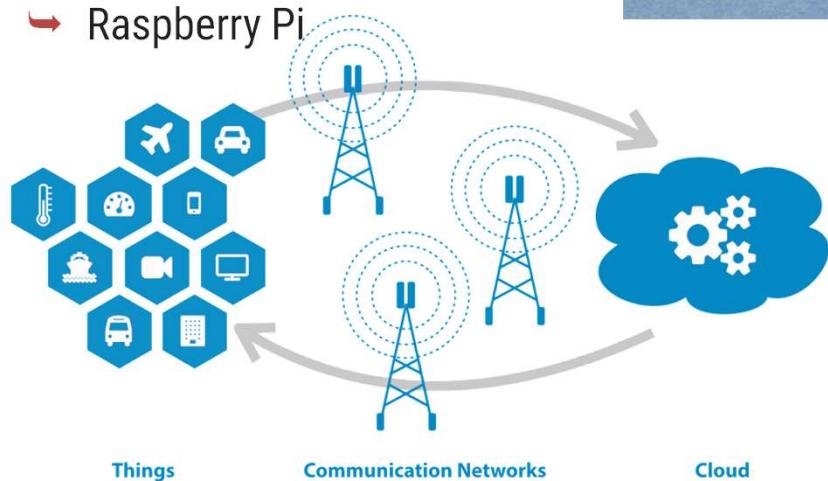
### ► Cameras

### ► On Board Diagnostic (OBD)

Access

### ► Controlling

- Raspberry Pi



### ► For Monitoring and Analysis

- PC
- Mobile
- Tablet

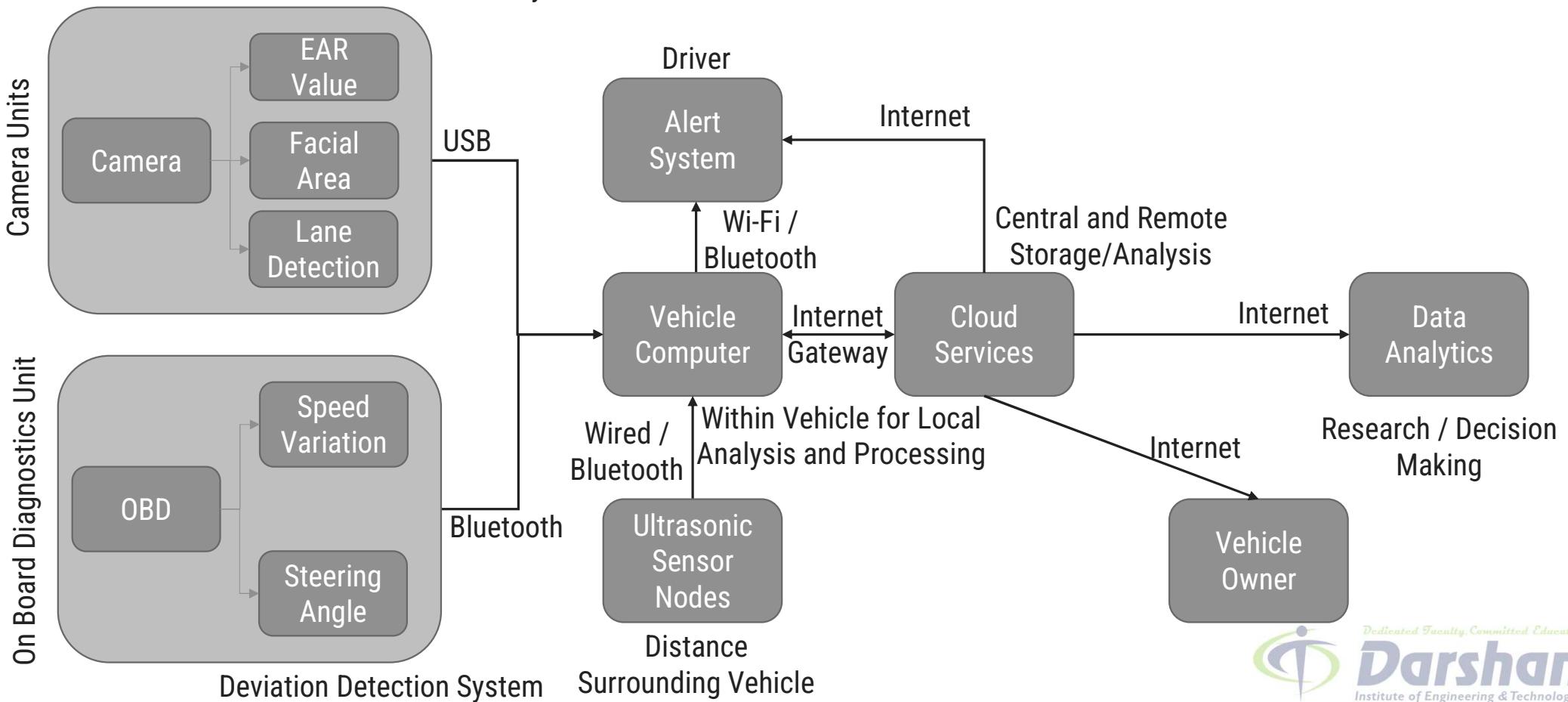


### ► WAN Connectivity

### ► Cloud Services

# Driver Assistance Application with IoT

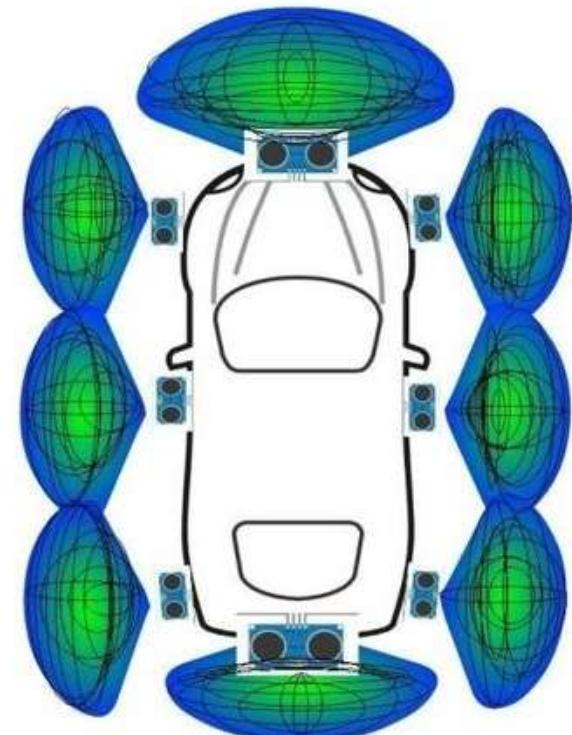
## Architecture Drowsiness Detection System



# Driver Assistance Application with IoT

## How it works

- ▶ **Camera units** should be placed on board for **capture** the **face of driver** and **road** view.
- ▶ The camera units should have USB connectivity with vehicle computer.
- ▶ The **camera** units **sends** the sensed **data to the vehicle computer**.
- ▶ The **ultrasonic sensor** nodes should be **placed** at various place on **outside** the **vehicle** for detecting surrounding **distance**.
- ▶ All the **sensor** nodes should have **Wired / Bluetooth connectivity** with the vehicle computer.



# Driver Assistance Application with IoT

## How it works

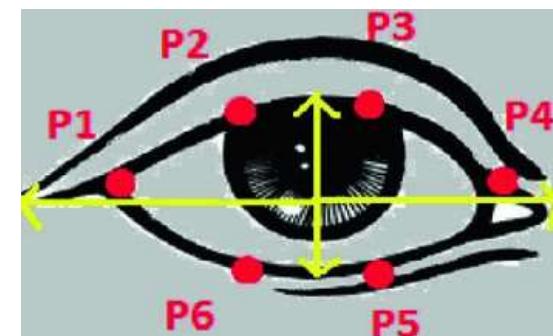
- ▶ The OBD unit of the vehicle should have **Bluetooth connectivity with the vehicle computer**
- ▶ The vehicle computer
  - Collects the **data from the camera units, OBD unit and the sensor nodes, and**
  - **Apply some business logic,**
  - **Do analysis** on the collected data on it as per the requirements and
  - **Send** the filtered limited **data to the cloud computer** as well as
  - **Send the alert messages to the driver.**
- ▶ The vehicle computer should have Internet connectivity for the cloud communication.



# Driver Assistance Application with IoT

## How it works

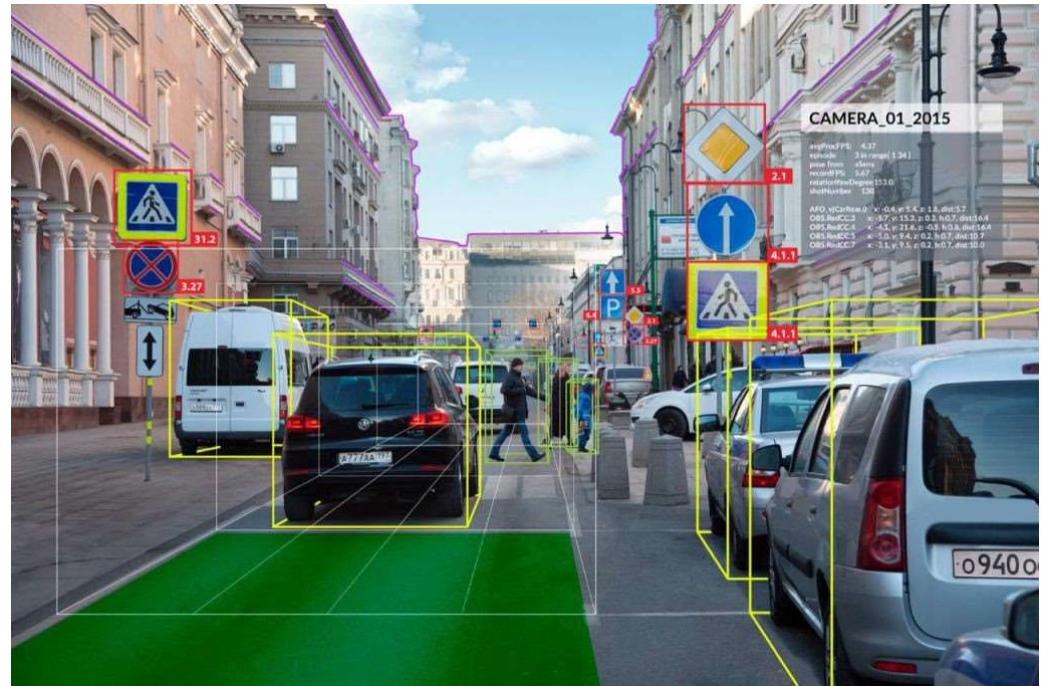
- ▶ The driver side **camera capture** the **drive facial** and **eye movement**.
- ▶ The camera **send** the captured **images to vehicle computer** for **EAR value** and other **calculation**.
- ▶ EAR (**Eye Aspect Ratio**) is measured using following equation.
- ▶ 
$$EAR = \frac{(P_2 - P_6) + (P_3 - P_5)}{2(P_1 - P_4)}$$
  - Where
    - $(P_2 - P_6)$  is distance between points  $P_2$  and  $P_6$
    - $(P_3 - P_5)$  is distance between points  $P_3$  and  $P_5$
    - $(P_1 - P_4)$  is distance between points  $P_1$  and  $P_4$
- ▶ Using **image processing** the system **finds** the **eye positions** and **calculate** the above mentioned **distance of the six points**.
- ▶ When the driver **closes his eye**, the **EAR value** eventually **drops to zero** and the system **detects drowsiness** of the driver.



# Driver Assistance Application with IoT

## How it works

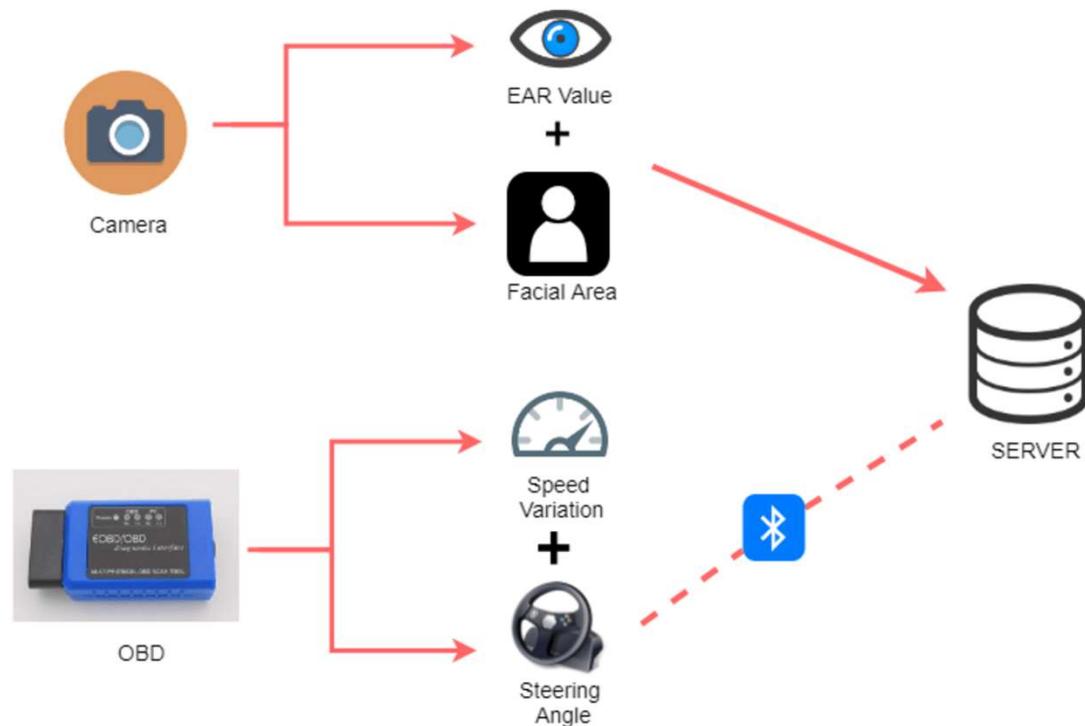
- ▶ The road side camera capture the **front view** images.
- ▶ The capture **images sends** to the **vehicle computer** for
  - Detecting lane
  - Objects
  - Sign boards
- ▶ **Deviation** is **monitored** using **OBD** (On-Board Diagnostics) device.



# Driver Assistance Application with IoT

## How it works

- ▶ OBD can record the pattern of **acceleration** and **steering angles** during the drowsiness of the driver.
- ▶ Compare the **recorded values** with the **threshold** value and the **historical data** for the actual deviation.
- ▶ All the **calculated data** is **sent** to the **data analytics** and **obtained** result, is used to **alert** the driver.
- ▶ The cloud data also useful for the vehicle owner.



# Collision Impact Detection Using IoT

## Overview

- ▶ Modern **vehicles** are **equipped with** the **collision detection** mechanism.
- ▶ It **blow** up **airbags** when an **accident** occurs and **save lives** in major cases.
- ▶ But if the collision is **massive** then the **airbags may not sufficient**.
- ▶ In these cases emergency help needed to save the lives.
- ▶ So we **need** a **system** that **measure** the **severity impact** of the collision and **send alert** messages.



# Collision Impact Detection Using IoT

## Overview

- ▶ The messages contents
  - **Location** of the accident
  - **Severity** impact,
  - **Fire status,**
  - **Speed** of the vehicle,
  - **Number of passengers,**
  - **Time** of the incident, etc.
- ▶ **Broadcast** the **messages** to all the authorities like
  - **Ambulance** service,
  - **Hospitals,**
  - **Highway patrolling team,**
  - **Fire brigade,**
  - **Police,**
  - **Family member** of the vehicle owner, etc.



# Collision Impact Detection Using IoT

## Importance

- ▶ If a **single accident** occur in along highway then it can be **managed without** the **data of severity** of the collision.
- ▶ When **multiple accident** occurs on the highway then the **severity data** is **very important** to **save** lives of the **very serious victims**.
- ▶ **Based on** the data and **severity** impact, **all** the **emergency help** will be **reached timely** at the point.
- ▶ Thus, we **need** an **IoT based system** that sense the necessary parameters of the collision.



# Collision Impact Detection Using IoT

## Requirements for IoT Setup

### ► Sensors

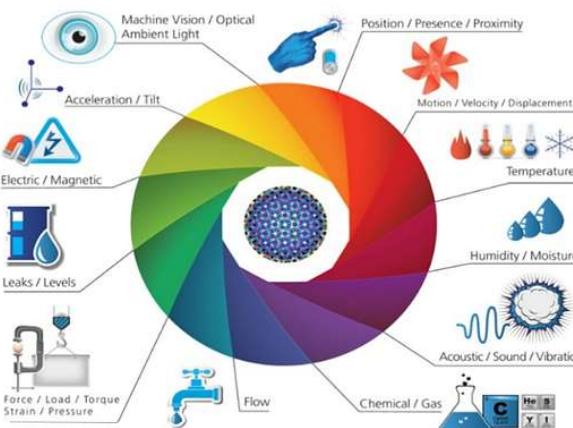
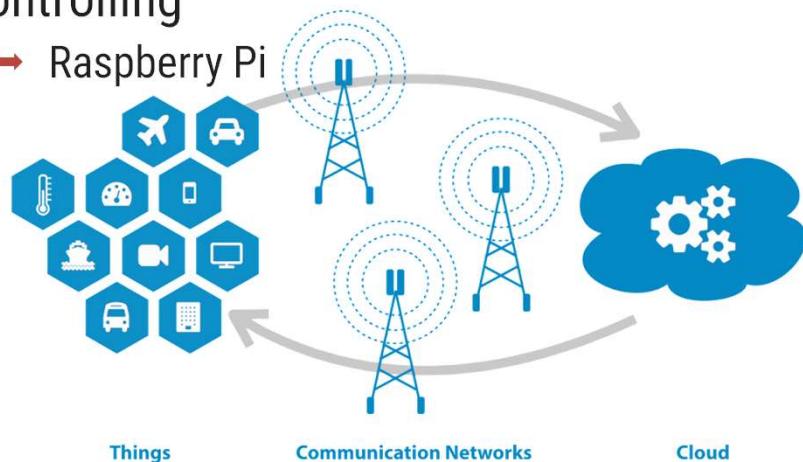
- FSR Sensor
- Fire Sensor
- GPS Sensor

### ► Cameras

### ► OBD Access

### ► Controlling

- Raspberry Pi



- WAN Connectivity
- Cloud Services

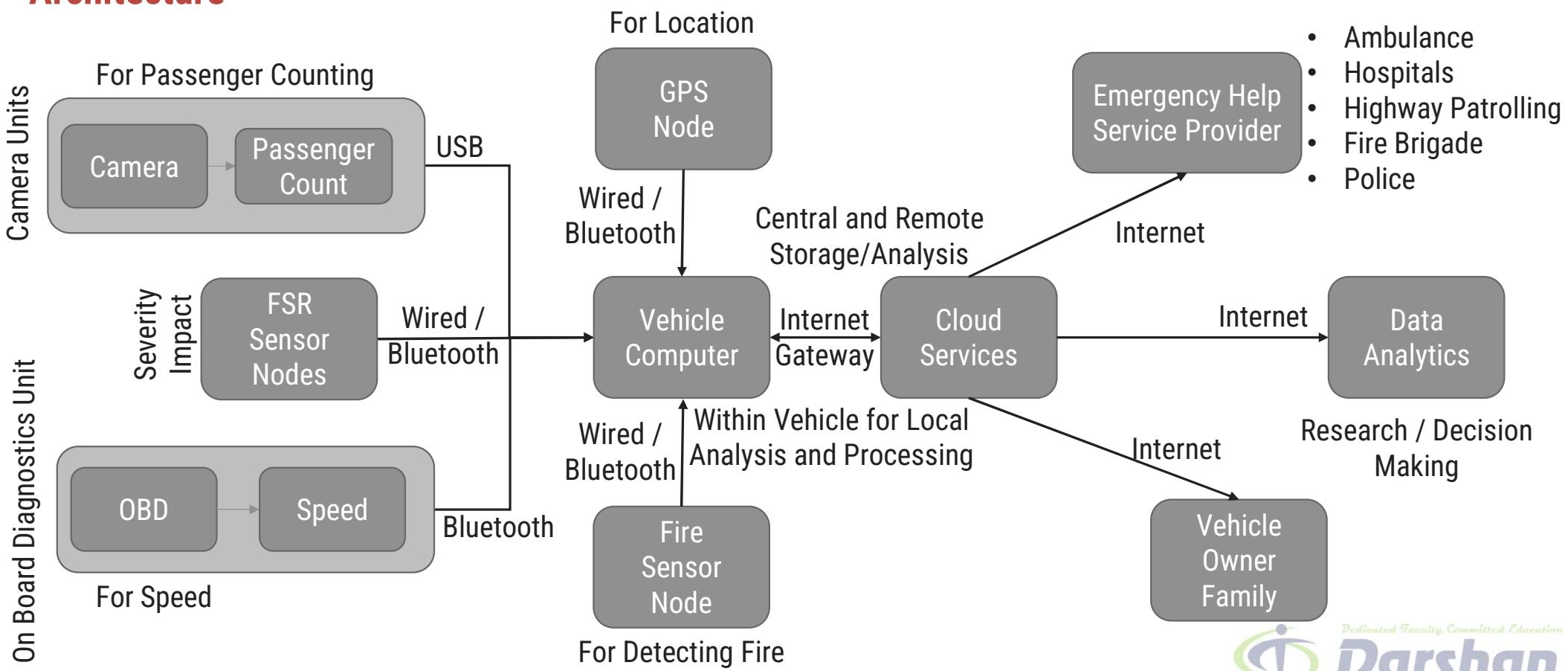
### ► For Monitoring and Analysis

- PC
- Mobile
- Tablet



# Collision Impact Detection Using IoT

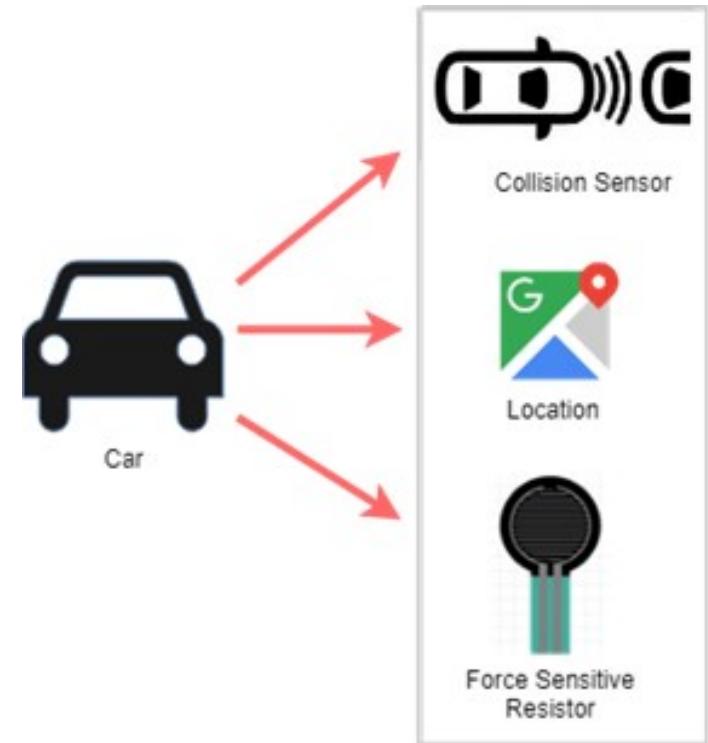
## Architecture



# Collision Impact Detection Using IoT

## How it works

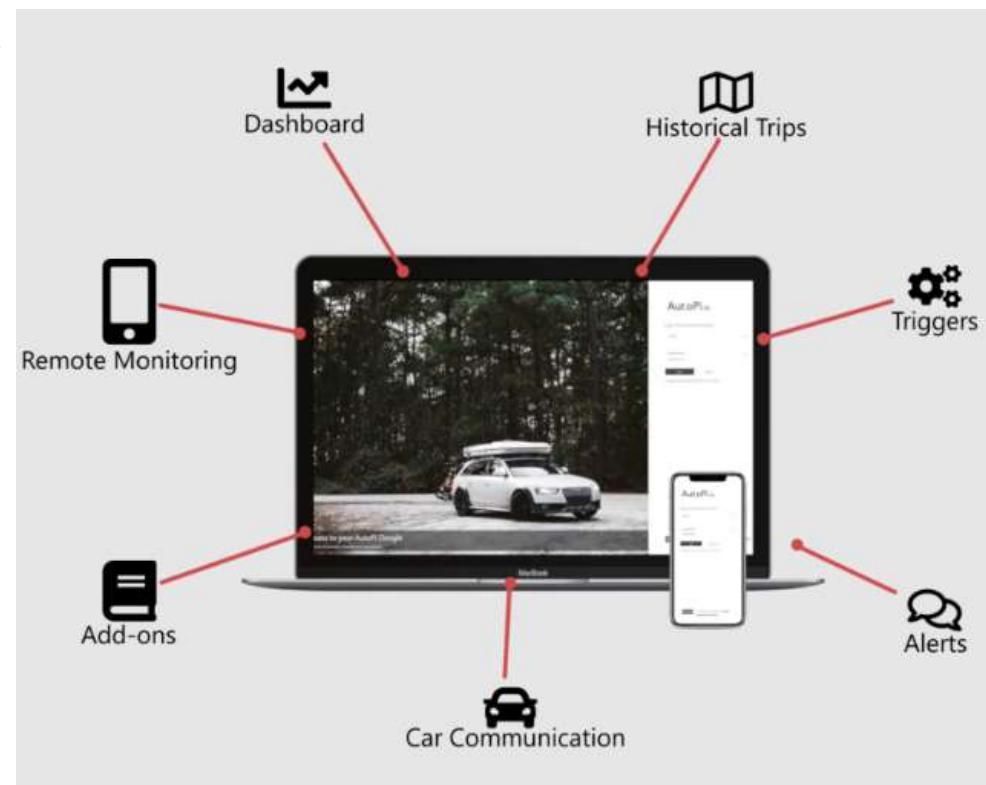
- ▶ Force sensitive resistor (FSR) sensors are useful to measure pressure applied on the vehicle during the collision.
- ▶ The FSR sensor are mounted on the vehicle at the places where impact probability is higher than the other part of vehicle during an accident.
- ▶ Based on the FSR data we can estimate the severity of the collision.
- ▶ Fire sensor and GPS should be placed at appropriate place of the vehicle.
- ▶ All the sensor nodes should have Wired / Bluetooth connectivity with the vehicle computer.
- ▶ Camera units should be placed on board for capturing all the passengers.



# Collision Impact Detection Using IoT

## How it works

- ▶ The camera units should have **USB connectivity** with vehicle computer.
- ▶ The **camera units sends the captured images** to the **vehicle computer**.
- ▶ The **OBD** unit of the vehicle should have **Bluetooth connectivity** with the vehicle computer
- ▶ The vehicle computer
  - Collects the **data from the camera** units, **OBD** unit and the **sensor** nodes.
  - **Apply** some business **logic**,
  - **Do analysis** on the collected data on it as per the requirements.
  - **Send** the analyzed **data to the cloud** computer.
  - **Should have Internet connectivity**



# Collision Impact Detection Using IoT

## How it works

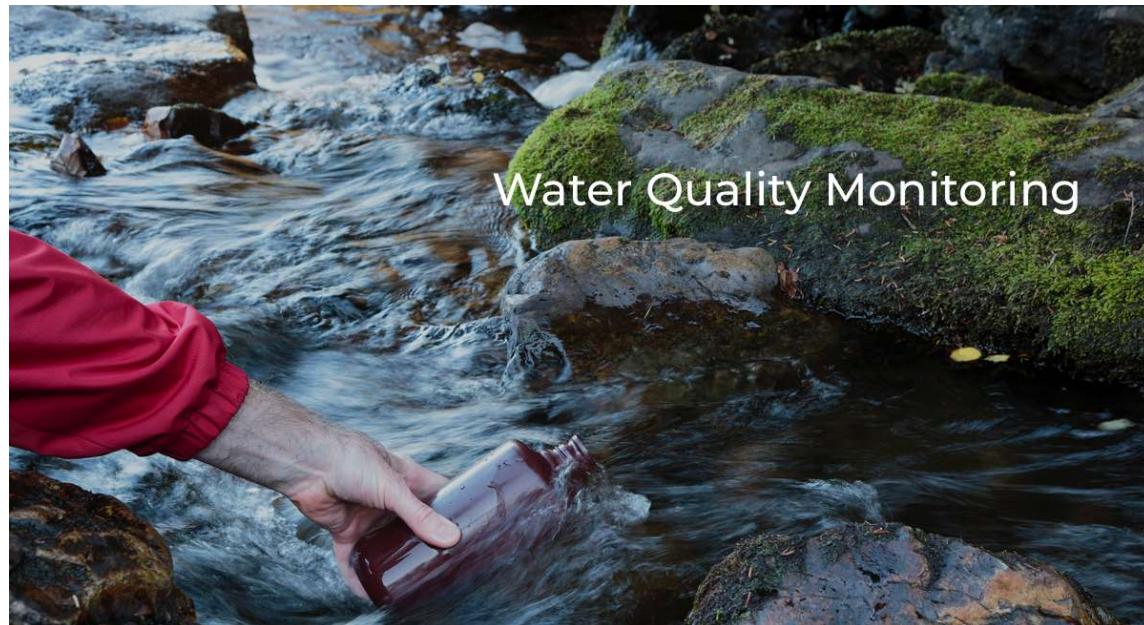
- ▶ Using **OBD** (On-Board Diagnostics) device the system **records speeds** of the vehicle.
- ▶ The fire sensor and GPS also sends the sensed data to the vehicle computer.
- ▶ **Based on the analyzed data and severity** of the collision the **cloud** system **sent alert** messages to all the **emergency help service provider** and **family member** of the vehicle owner.
- ▶ The **cloud data** is used for **data analytics** and to provide the **priority based** emergency **services**.
- ▶ The system is **automatic activated when** the **collision is detected**.



# Water Quality Monitoring - IoT Based Application

## Overview

- ▶ **Water is essential for life** and the **polluted water** is a one of the major **challenge in the world**.
- ▶ So water **quality monitoring** is **necessary** for us.
- ▶ **Water Quality Monitoring (WQM)** helps in **preventing** and **controlling** water **pollutions**.
- ▶ Water quality should be checked regularly to **avoid serious** health **issues**.
- ▶ **IoT** can **help to develop** a perfect **WQM** system.



# Water Quality Monitoring - IoT Based Application

## Importance



- ▶ At present water quality is monitored very expensive and inefficient by manual or nodal network methods.
- ▶ Manual Method
  - Water samples are taken manually from the different points of a water area.
  - The samples sent to laboratories
  - Various tests are conducted to find the values of
    - PH, Conductivity and Temperature
    - TDS, Dissolved Oxygen
    - Turbidity
    - Chloride Content
  - The reports of the test sends to the concern authorities with in a week.

# Water Quality Monitoring - IoT Based Application

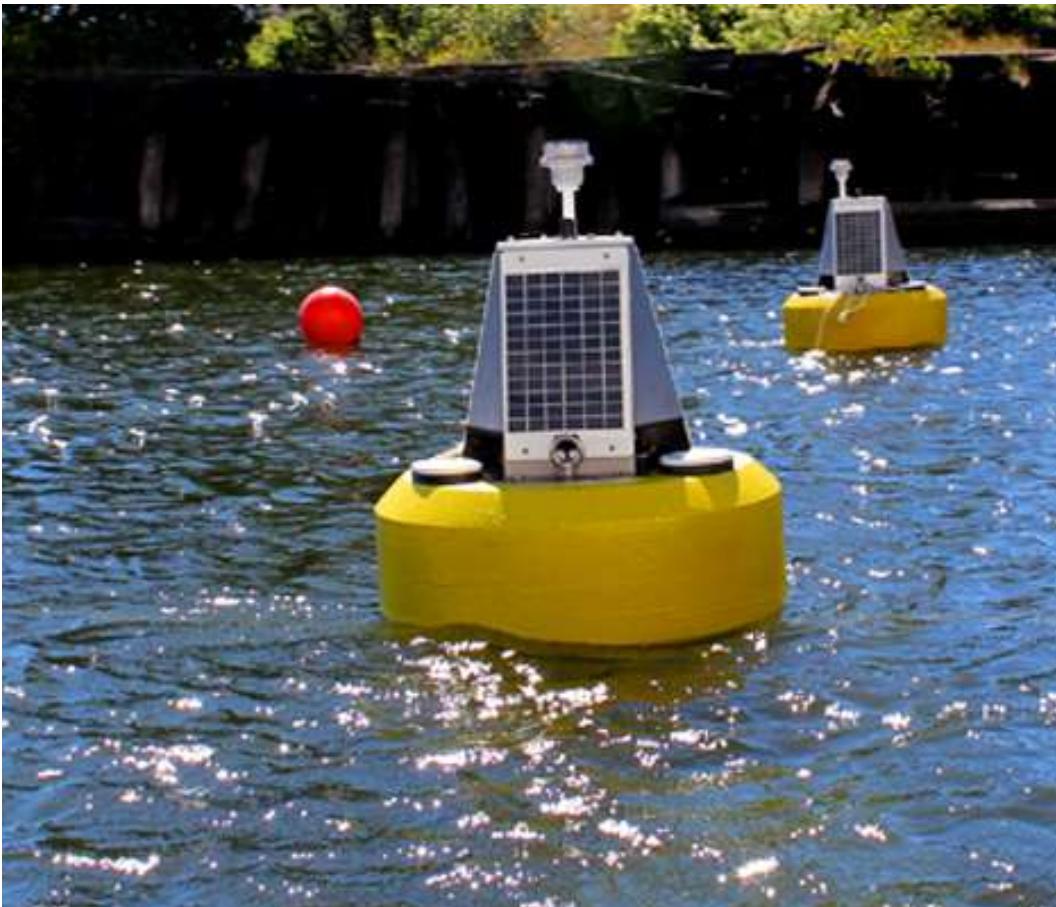
## Importance

- ▶ **Manual Method**
- ▶ The methods have some **limitations** like
  - Chance of **human error** in the sampling
  - Sampling **cost** is **very high** because of to cover very large area of waterbody
  - Sampling and Testing both are **time consuming**
  - **Not real-time** monitoring which leads to may serious issues.



# Water Quality Monitoring - IoT Based Application

## Importance

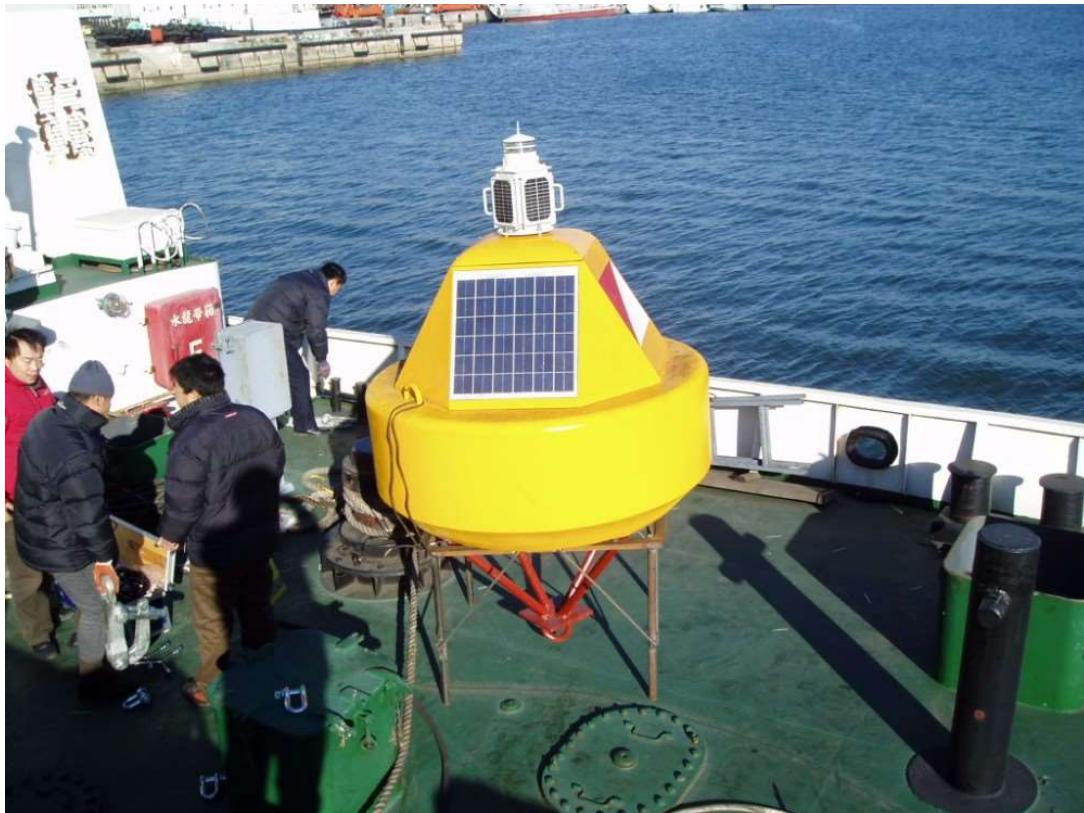


### ► Nodal Network Method (Electronics Sensor Monitoring)

- Water quality is **measured by** electronic **sensors** installed in the different points of a water area.
- All the sensors are **wirelessly connected** with the Internet.
- The water quality **data sends to** the **cloud** server at **regular interval**.
- This method is giving **real-time data** contrast to the manual method.
- System architecture of the **monitoring system** is **divided into** following four areas
  - Sensing
  - Data Collection
  - Processing
  - Communication

# Water Quality Monitoring - IoT Based Application

## Importance



- ▶ **Nodal Network Method (Electronics Sensor Monitoring)**
- ▶ The methods have following **limitations**
  - **Very expansive**
  - **Maintenance** of the sensors along with **power source** are **challenging**
  - Natural **disaster** and aquatic animal **movement** in the waterbody may **damage** the sensors
  - The sensors are **not moveable**

# Water Quality Monitoring - IoT Based Application

## Importance

- ▶ A **low cost solution** is possible for WQM **using latest technology**, Data Analytics, USVs, Machine Learning, AI and IoT.
- ▶ **No need** to **install** sensors **permanently** into the waterbody in the solution
- ▶ An **autonomous boat** (USV), mounted **with** all the necessary **sensors** can
  - **Collect** the **water quality data** and
  - **Send** the **data to the cloud** for data analytics
- ▶ **All the necessary parameters** of water can be **tracked** and **measured**.
- ▶ Detailed report of the **containment status** of the **entire waterbody** can be **generated**.

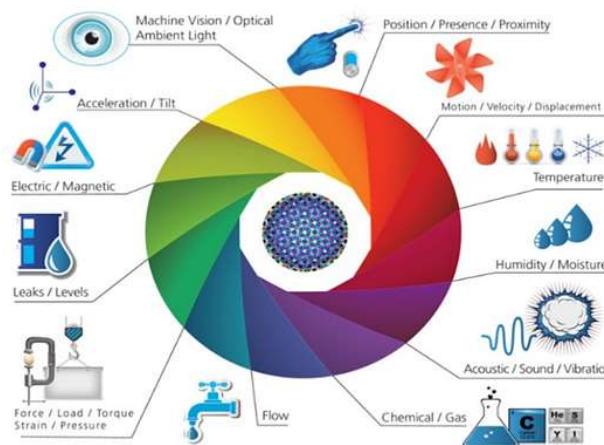


# Water Quality Monitoring - IoT Based Application

## Requirements for IoT Setup

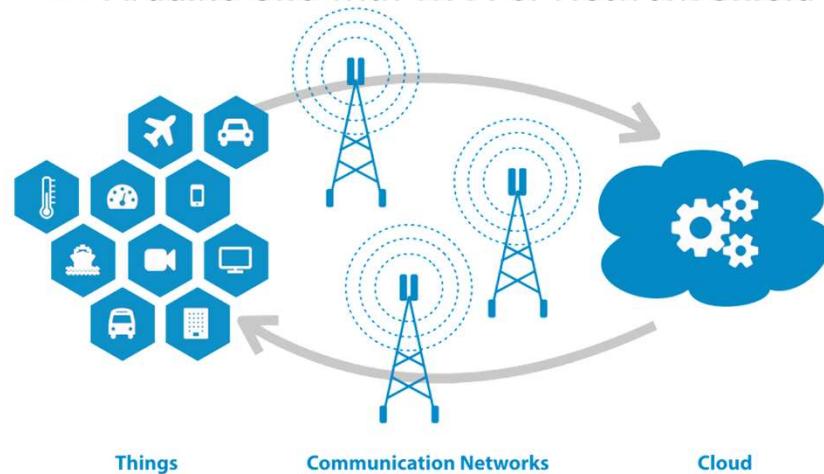
### ► Water Sensors Kit

- PH Sensor → Temperature Sensor
- GPS Sensor → Conductivity Sensor
- Turbidity Sensor
- TDS and Dissolved Oxygen Sensor



### ► For Monitoring and Analysis

- PC
- Mobile
- Tablet

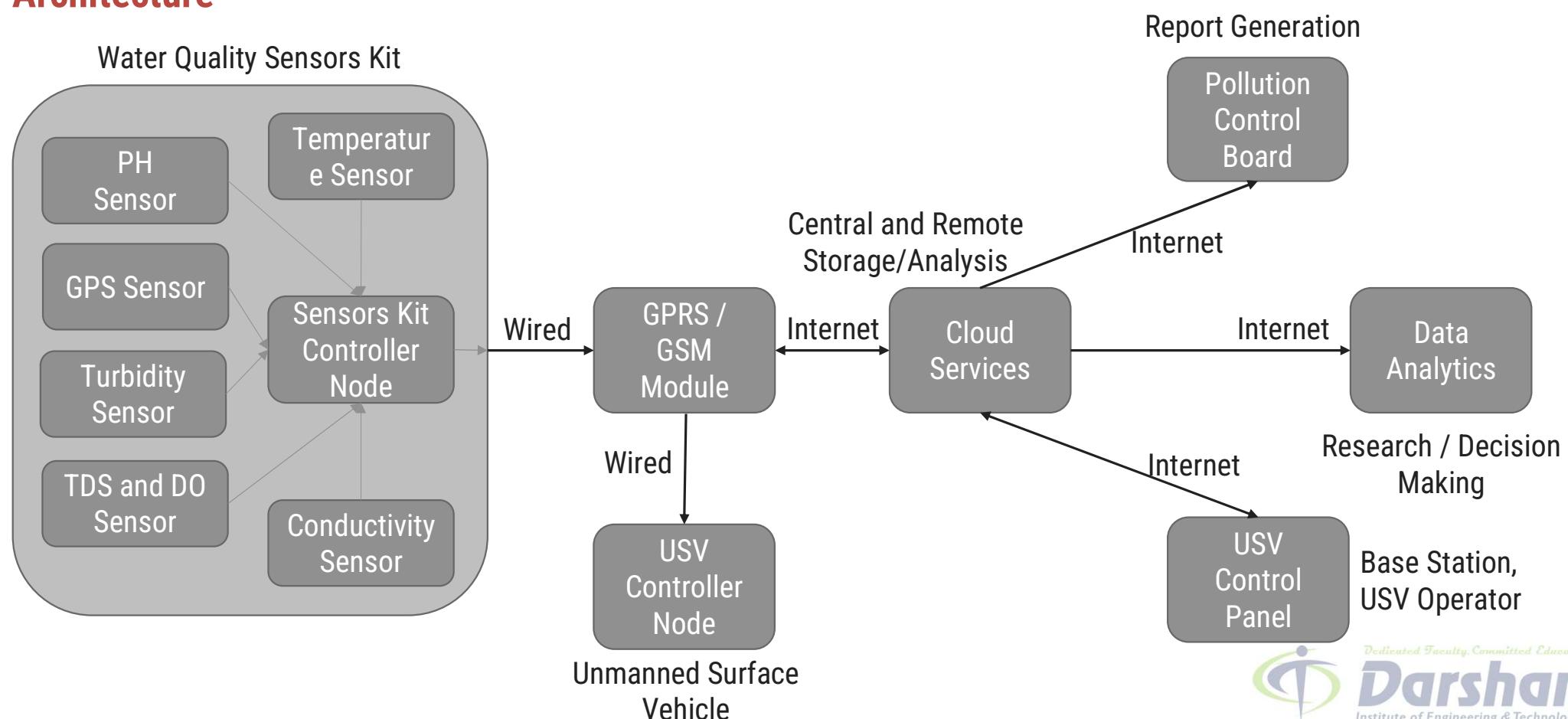


- WAN Connectivity  
→ GPRS or GSM Module
- Cloud Services

### ► Unmanned Surface Vehicle (USV)

# Water Quality Monitoring - IoT Based Application

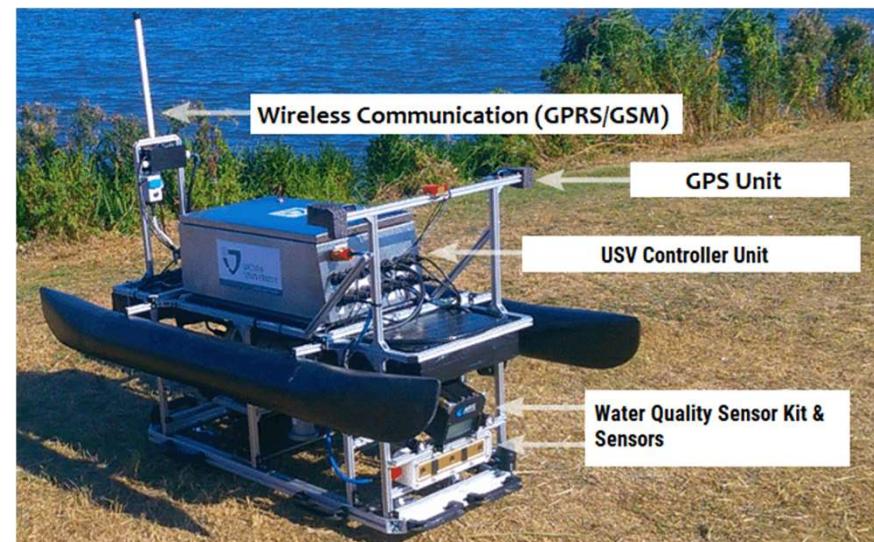
## Architecture



# Water Quality Monitoring - IoT Based Application

## How it works

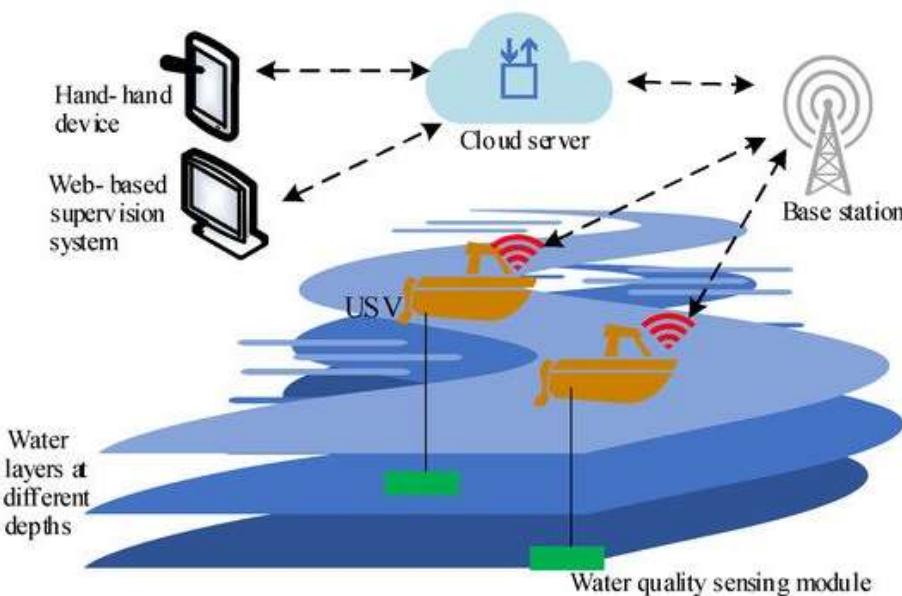
- ▶ A bundle of all the sensors or a **water quality sensor kit** should be mounted **on** an **autonomous** or unmanned surface **vehicle** (USV).
- ▶ The **USV** can be **operated by** an embedded **controller node**.
- ▶ The USV should have **GSM** or **GPRS internet connectivity** with the cloud server.
- ▶ The **sensor kit sense** the following **values**
  - **PH Value** of Water - PH Sensor
  - **Temperature** of Water - Temperature Sensor
  - **Conductivity** of Water - Conductivity Sensor
  - **Clarity** of Water - Turbidity Sensor
  - **Dissolved Solid and Oxygen** in Water- TDS and DO Sensor
  - **Location** of the reading - GPS Sensor



# Water Quality Monitoring - IoT Based Application

## How it works

- ▶ The **kit** controller node **sends** all the sensed **data** to the **cloud** server **via GPRS/GSM** module
- ▶ The **GPRS/GSM** module is used to **provide** the **Internet connectivity** with the cloud server.
- ▶ The cloud
  - **stores all the data** received **from** the **USVs**,
  - **Execute** some business **logic** and
  - **Do analysis** on the stored data.
- ▶ All those who have **rights to access** the **cloud data** can **get** water **quality records** of the waterbody.
- ▶ Like,
  - **Members of Pollution Control Board**
  - **Data analytics team**
  - **USV operator**



# Water Quality Monitoring - IoT Based Application

## How it works

- ▶ **Authorities** of the board and data **analytics teams** may **execute** corrective **action** based on the data.
- ▶ An **USV control panel** is **installed** in the **base station** of the WQM system.
- ▶ The control panel should have internet connectivity for controlling the USV.
- ▶ Using the control panel, an **operator** can **set paths** and **control** the USV.



# Smart Lavatory - IoT Based Application

## Overview

- ▶ In the all **businesses retaining old customer** is more **meaningful** then the finding new.
- ▶ It is possible by
  - Offering **good quality products**,
  - **Best services** and
  - The **pleasant experience** while visiting your place.
- ▶ **Hygiene** and **comfort** in lavatory is one of the **major concern** of the businesses.
- ▶ **Maintaining** and **monitoring cleanliness** in lavatory is a **challenging task**.



# Smart Lavatory - IoT Based Application

## Importance

- ▶ Mostly **lavatory cleaning** and **monitoring** is **manually** done by cleaning staff.
- ▶ Because of the manual process some time it's **not done properly**.
- ▶ **Unclean lavatory** may become a **root cause** of many **diseases**.
- ▶ **User** may **suffer** because of **unavailability** of **water** or **toilet papers** in many cases.
- ▶ Slippery and **wet floor** might be **unsafe** for users.
- ▶ An **IoT** based **smart lavatory monitoring system** is **needed** to solve the above mentioned issues.



# Smart Lavatory - IoT Based Application

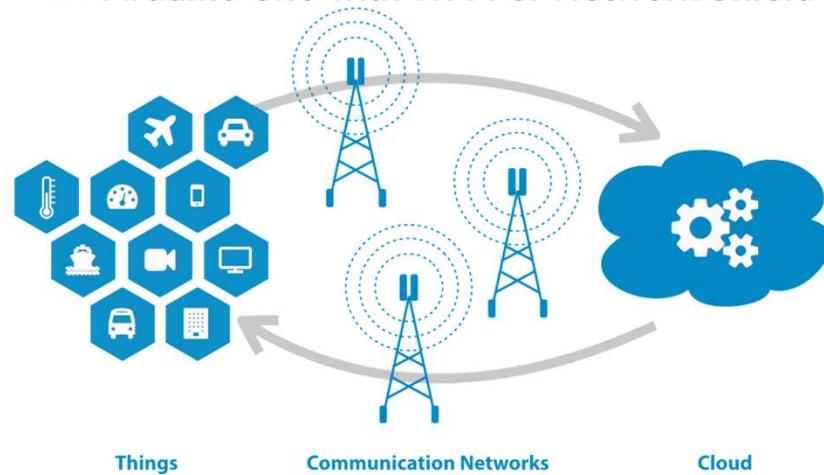
## Requirements for IoT Setup

### ► Water Sensors Kit

- MQ05 Gas Sensor
- Touch Sensor
- Sharp IR Sensor
- Water Level Float Sensor

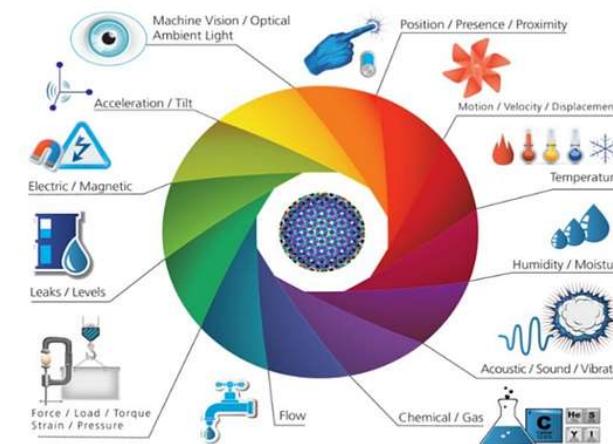
### ► Controller

- ESP32 MCU or
- Arduino Uno with Wi-Fi or Network Shield



Prof. Kalpesh H Surati

#3160716 (IoT) • Unit 5 – Application Building with IoT



### ► For Monitoring and Analysis

- PC
- Mobile
- Tablet



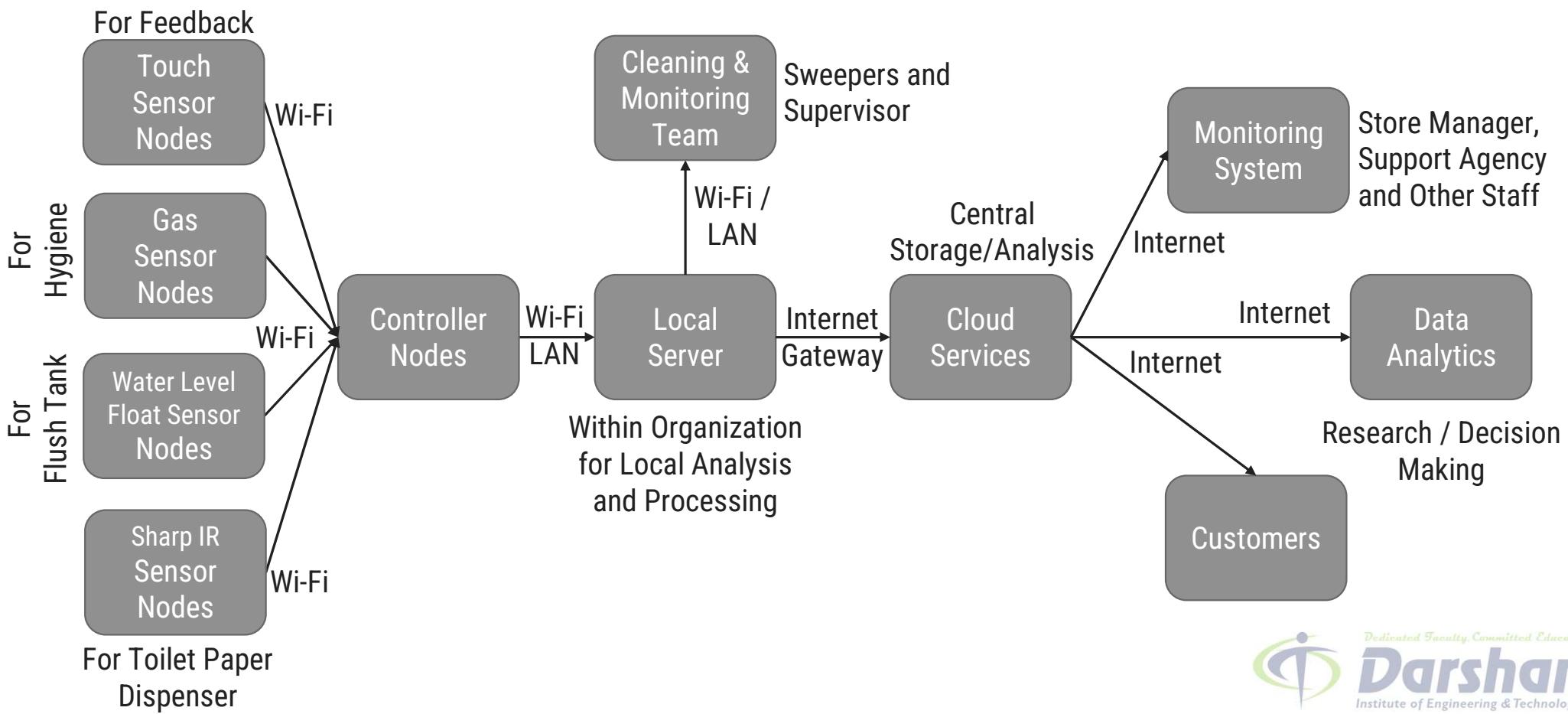
### ► LAN and WAN Connectivity

- Router and Switches
- Wireless Access Point

### ► Cloud Services

# Smart Lavatory - IoT Based Application

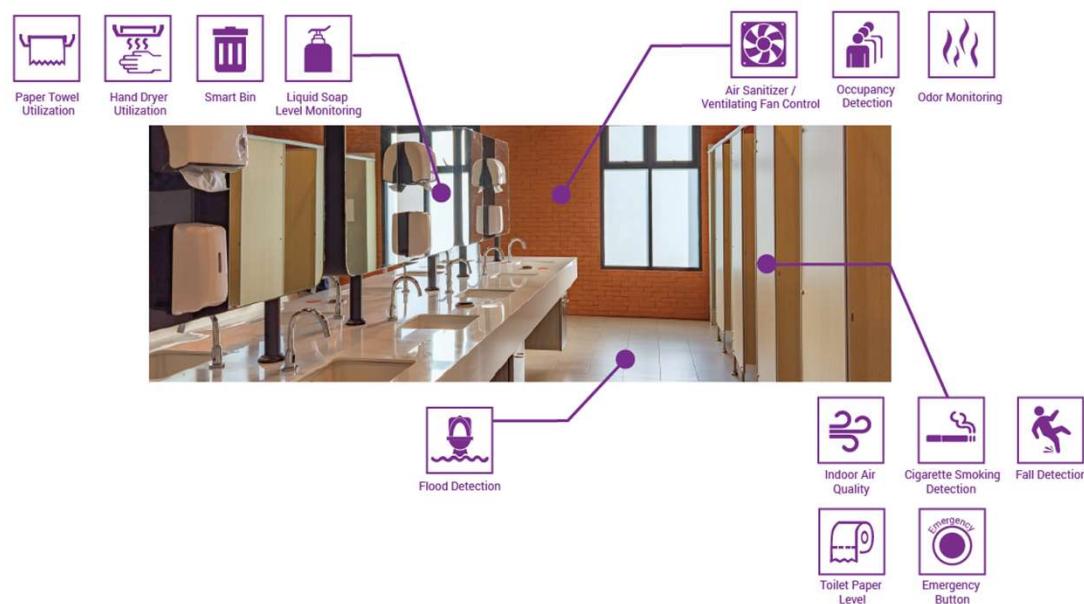
## Architecture



# Smart Lavatory - IoT Based Application

## How it works

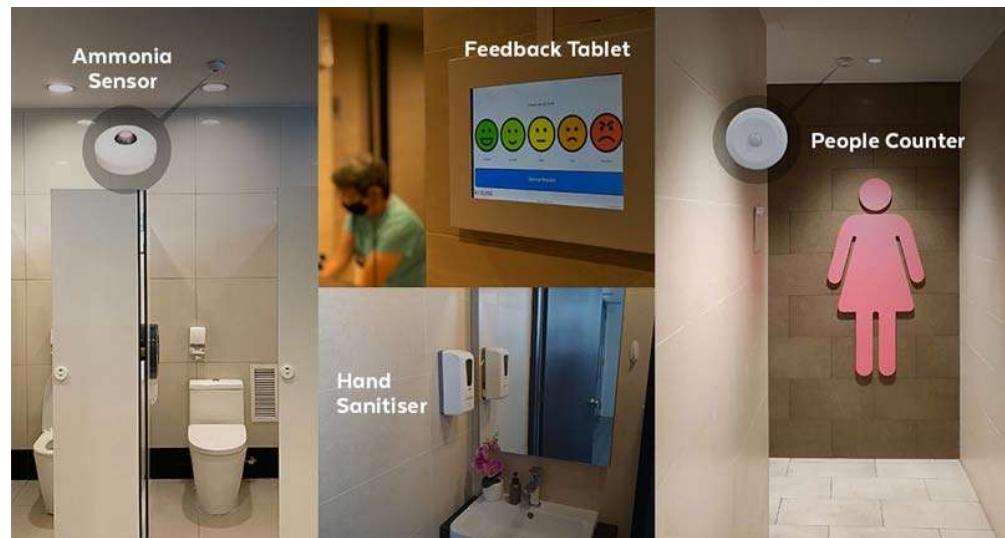
- ▶ All the **sensor** nodes should be **placed** at various places **in the lavatory area**.
- ▶ The sensor nodes should have **Wi-Fi connectivity** with controller nodes.
- ▶ The sensor nodes sends the sensed data to the controller nodes.
- ▶ The controller nodes
  - **Collects** the **data** from all the sensors,
  - **Apply** some business **logic** on it as per requirements and
  - **Send** it **to local computer**.
- ▶ All the controller nodes should have connected with the Local Area Network.



# Smart Lavatory - IoT Based Application

## How it works

- ▶ **Gas sensors** are used for **monitoring** various **gases** levels like **ammonia** and **methane** level in the lavatory.
- ▶ **Touch sensors** are used for customers' **feedback** about **ambiance** and **cleanliness** in the lavatory.
- ▶ Water level **float sensors** are used for **monitoring water level** in the **flush tank** and the flush **count**.
- ▶ Sharp **IR** sensors are placed in the **toilet paper dispenser** for monitoring remaining toilet papers.



# Smart Lavatory - IoT Based Application

## How it works

### ► Local server

- Receives all the **data** from all the **controller** nodes,
- Execute some **business logic** and
- Do analysis on the **collected data**, and
- Send the **essential data** to the **cloud** computer.

### ► All the **cleaning & monitoring team** members can be accessed following from the local server .

- Status of lavatory cleanliness,
- Water and toilet paper availability

### ► The local server should have Internet connectivity to communicate with the cloud.



# Smart Lavatory - IoT Based Application

## How it works

- ▶ The cloud
  - stores all the data received from the local server,
  - execute some business logic and
  - do analysis on the stored data.
- ▶ All those who have rights to access the cloud data can get desire information and alerts, like
  - Store manager
  - Service support agency
  - Local staff members
  - Data analytics team
  - Customers
- ▶ The store manager and data analytics teams may execute corrective action based on the analyzed data.





Unit-6

# Arduino and Raspberry Pi



**Prof. Tushar J. Mehta**  
Computer Engineering Department  
Darshan Institute of Engineering & Technology, Rajkot

---

✉ tushar.mehta@darshan.ac.in  
📞 8866756776

# Arduino

Section - 1

# Programming the Arduino

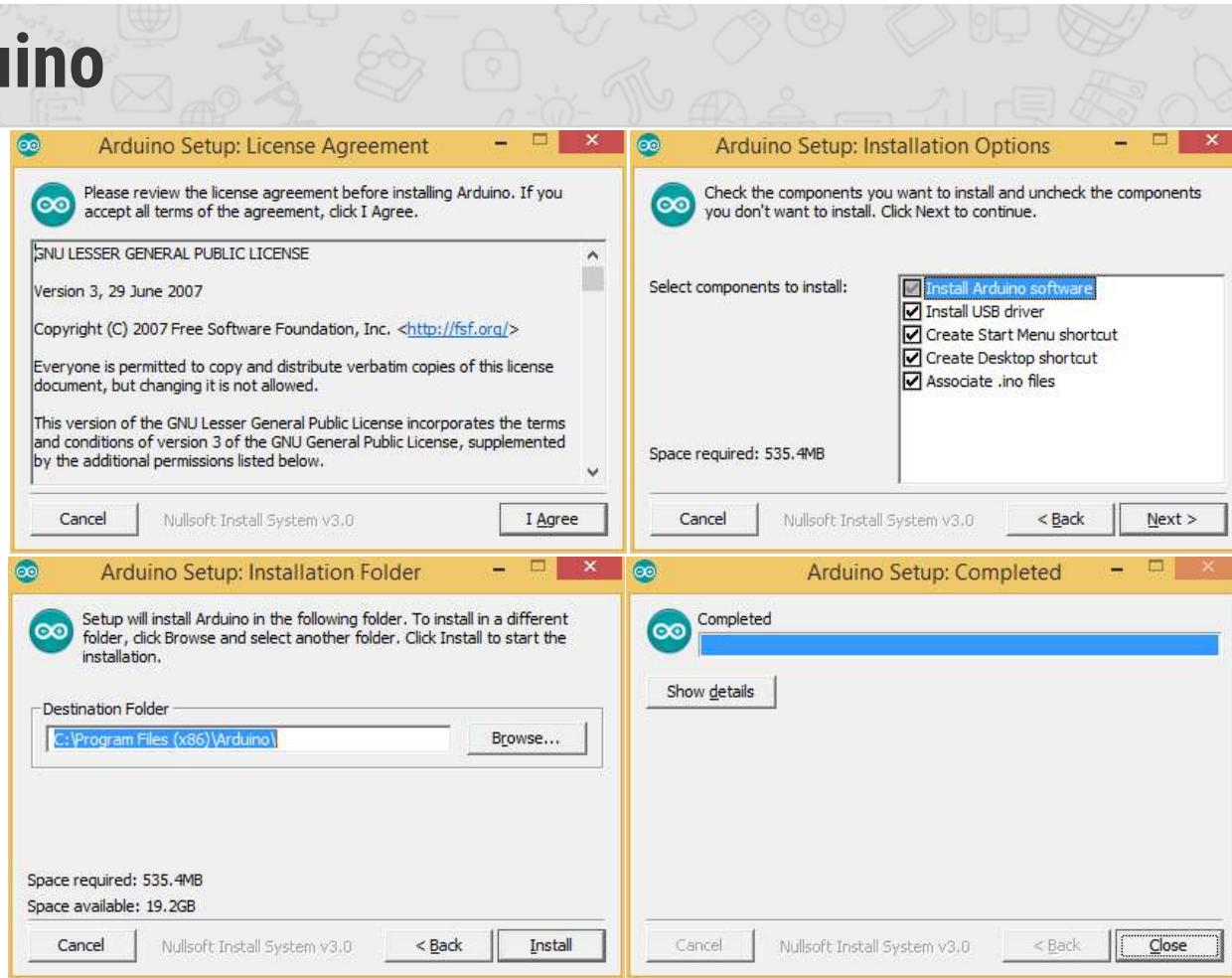
- ▶ Programming of Arduino is done in Arduino IDE – Integrated Development Environment.
- ▶ The software helps to connect, upload and communicate with Arduino hardware.
- ▶ The Arduino software is an open-source platform.
- ▶ The steps to install the Arduino software into the computer are as following:
- ▶ Download Arduino IDE from <http://arduino.cc/en/Main/Software>

The screenshot shows the Arduino website's download section. At the top, there's a navigation bar with links for PROFESSIONAL, EDUCATION, STORE, HARDWARE, SOFTWARE (which is highlighted in yellow), DOCUMENTATION, COMMUNITY, BLOG, and ABOUT. Below the navigation is a search bar with the placeholder "Search on Arduino.cc". The main content area has a title "Downloads" and a section for "Arduino IDE 1.8.13". This section includes a circular icon with two interlocking arrows, a brief description of what the IDE does, and a link to the "Getting Started" page for installation instructions. Below this is a "SOURCE CODE" section with a note about GitHub and a link to the source code archives. To the right of the IDE section is a "DOWNLOAD OPTIONS" sidebar with links for Windows (Win 7 and newer, ZIP file, and app), Linux (32-bit and 64-bit), ARM 32-bit, ARM 64-bit, and Mac OS X (10.10 or newer). There are also links for Release Notes and Checksums.

This screenshot shows a different view of the Arduino download options. It features a teal header with "DOWNLOAD OPTIONS" and a "Windows" section highlighted with a red border. The "Windows" section contains links for "Win 7 and newer", "ZIP file", and a "Get" button. Below this are sections for "Windows app", "Linux" (32-bit and 64-bit), "Linux ARM 32 bits", "Linux ARM 64 bits", and "Mac OS X" (10.10 or newer). At the bottom, there are links for "Release Notes" and "Checksums".

# Installation process of Arduino

- ▶ After downloading the installer double click and install the software by agreeing to License Agreement.
- ▶ Select the components required to be installed and click on next. Make sure to have free space in hard drive which is mentioned in the software.
- ▶ Select the path where you want to install Arduino IDE and click on “Install”
- ▶ Now wait for few minutes until installation process gets completed.
- ▶ The dialogue box will show the “Completed” status of the installation. Click on “Close” button.

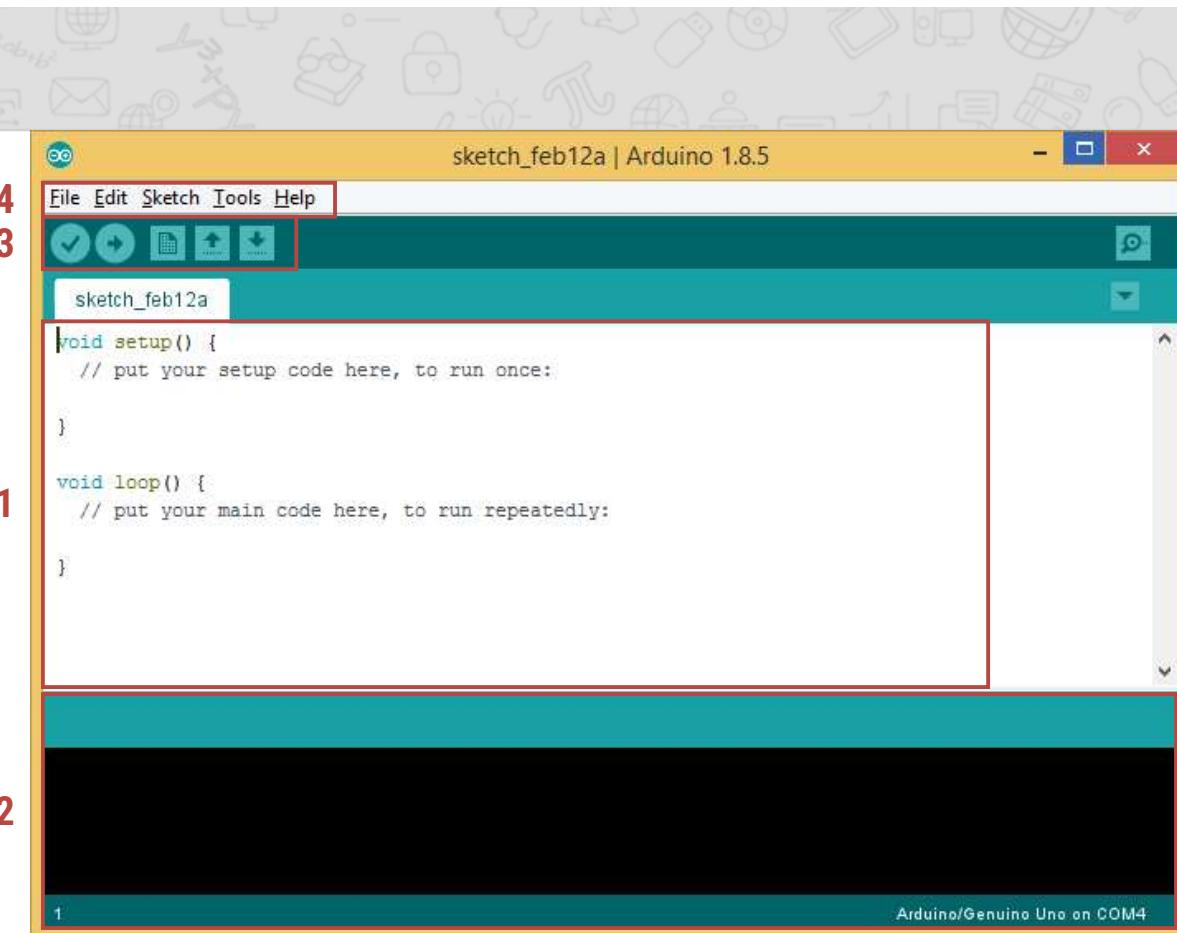


# Arduino IDE Explanation

► Open Arduino IDE software. Here, we can find different sections.

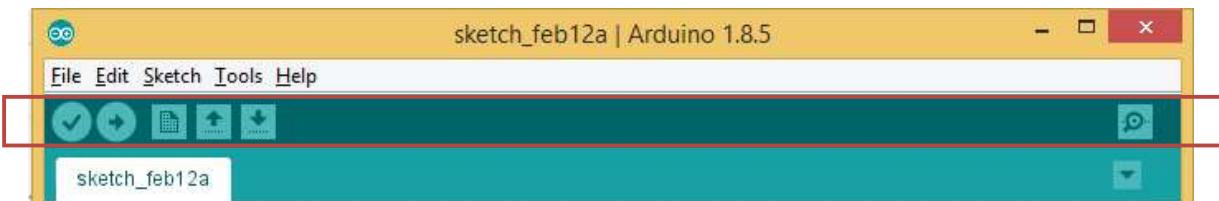
1. Editor – It is the space where we can write the code
2. A text Console – for displaying the messages.
3. A toolbar with buttons – given for common functions to perform.
4. Menu bar – A menu bar with series of menus

► Note that, the name of the default file is sketch\_[currentdate] with the alphabet 'a' and every new file has consecutive alphabet.



# Arduino IDE Explanation – Cont.

- ▶ The functions of buttons in toolbar can be explained as following:



## Verify

Checks your code for errors compiling it.

## Upload

Compiles your code and uploads it to the configured board.

## New

Creates a new sketch.

## Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

## Save

Saves your sketch.

## Serial Monitor

Opens the serial monitor.

# Programming the First Code

- ▶ The IDE is already loaded with few basic examples. Let us take a basic example of LED blinking which blinks built-in LED of Arduino UNO with a 1-second interval.
- ▶ Go to File > Examples > 01.Basics > Blink as shown in the figure.
- ▶ It will load the prewritten example in a new window with the title “Blink”.



# Code Explanation

Blink.ino

```
1 void setup() {  
2     pinMode(LED_BUILTIN, OUTPUT);  
3 }  
4 void loop() {  
5     digitalWrite(LED_BUILTIN, HIGH);  
6     delay(1000);  
7     digitalWrite(LED_BUILTIN, LOW);  
8     delay(1000);  
9 }
```

Initialize digital pin LED\_BUILTIN which is at pin 13 as an output.

turn ON the LED by writing HIGH voltage level

A delay of 1000ms = 1 sec

turn OFF the LED by writing LOW voltage level

A delay of 1000ms = 1 sec

- ▶ We should initialize the pin as input or output once only. Therefore the syntax for defining the pin as output is written in void setup () function.
- ▶ For blinking of LED, the LED should turn on and off at the regular interval of time. Hence, the instructions for turning on, turning off and delay for both on time and off time are written in the void loop () function

# Code Explanation

- ▶ Arduino code is always written with two mandatory functions.
  - void setup ()
  - void loop ()
- ▶ The instructions which are to be performed once in the program are written in void setup () function.
- ▶ The instructions written in void loop () are executed infinite times until the next restart of Arduino board.

```
void setup() {  
    // put your setup code here, to run once:  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
}
```

- ▶ Comments are important for any programming language. They are ignored while execution of the program

- Anything written after // in Arduino program is considered as a single line comment
- Anything written between /\* and \*/ is considered as multiple line comments.

# Functions in Arduino IDE

► **pinMode( )**: Configures the specified pin to behave either as an input or an output.

→ Syntax : `pinMode(pin, mode)`

→ Parameters :

- pin: the Arduino pin number to set the mode of.
- Mode: INPUT or OUTPUT

► **digitalWrite( )** : Write a HIGH or a LOW value to a digital pin.

→ Syntax : `digitalWrite(pin, value)`

→ Parameters :

- pin : the Arduino pin number.
- Mode : HIGH or LOW

► What is HIGH or LOW?

→ **HIGH**: - Also considered as Logic '1' or Logic High. It will set 5V on the configured pin of Arduino.

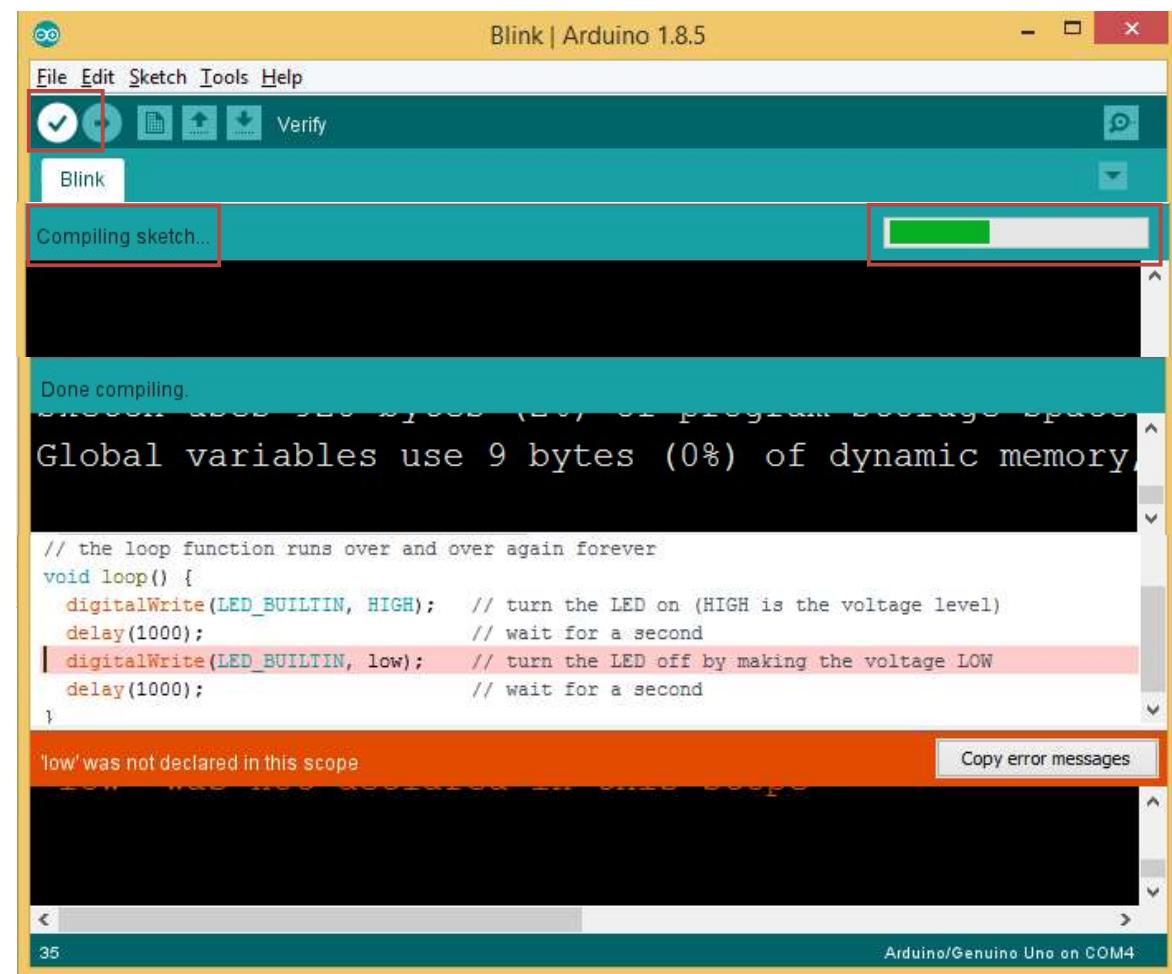
→ **LOW**: - Also considered as Logic '0' or Logic Low. It will set 0V on the configured pin of Arduino.

# Functions in Arduino IDE (Cont.)

- ▶ **delay( )**: It pauses the program for the amount of time (in milliseconds) specified as the parameter. For example, if we want to hold the program for 1 second before executing the next instruction. A delay of 1000ms = 1s is passed in the parameter.
  - Syntax : `delay(ms)`
  - Parameters :
    - ms: the number of milliseconds to pause

# Compiling the Code

- ▶ To compile a code in Arduino click on the “Verify” button in menu bar.
- ▶ While compiling the code, a message “Compiling sketch” is shown with its progress in the status bar.
- ▶ If there is no error in the code, the message “Done compiling” is displayed in the status bar.
- ▶ If an error is present in the code it will show the message of error with highlighted syntax where the error is present.



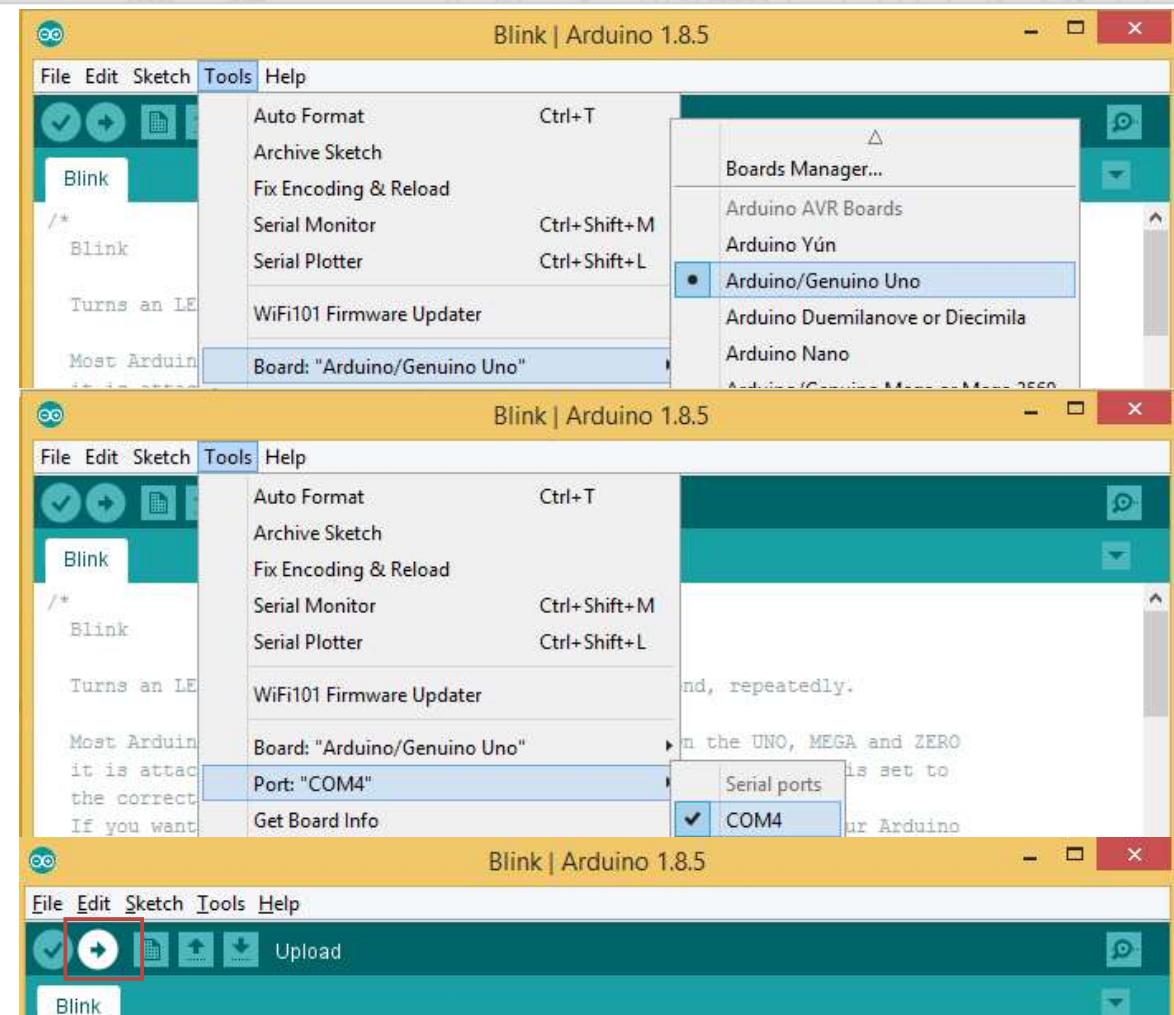
The screenshot shows the Arduino IDE interface with the title "Blink | Arduino 1.8.5". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar has icons for Open, Save, Verify, Upload, and Run. A red box highlights the Verify button. The status bar at the bottom shows "Arduino/Genuino Uno on COM4". The code editor contains the standard Blink sketch. The status bar displays "Compiling sketch..." with a progress bar, followed by "Done compiling.", "Global variables use 9 bytes (0%) of dynamic memory.", and an error message: "'low' was not declared in this scope".

```
// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on (HIGH is the voltage level)
    delay(1000);                         // wait for a second
    digitalWrite(LED_BUILTIN, low);        // turn the LED off by making the voltage LOW
    delay(1000);                         // wait for a second
}

'low' was not declared in this scope
```

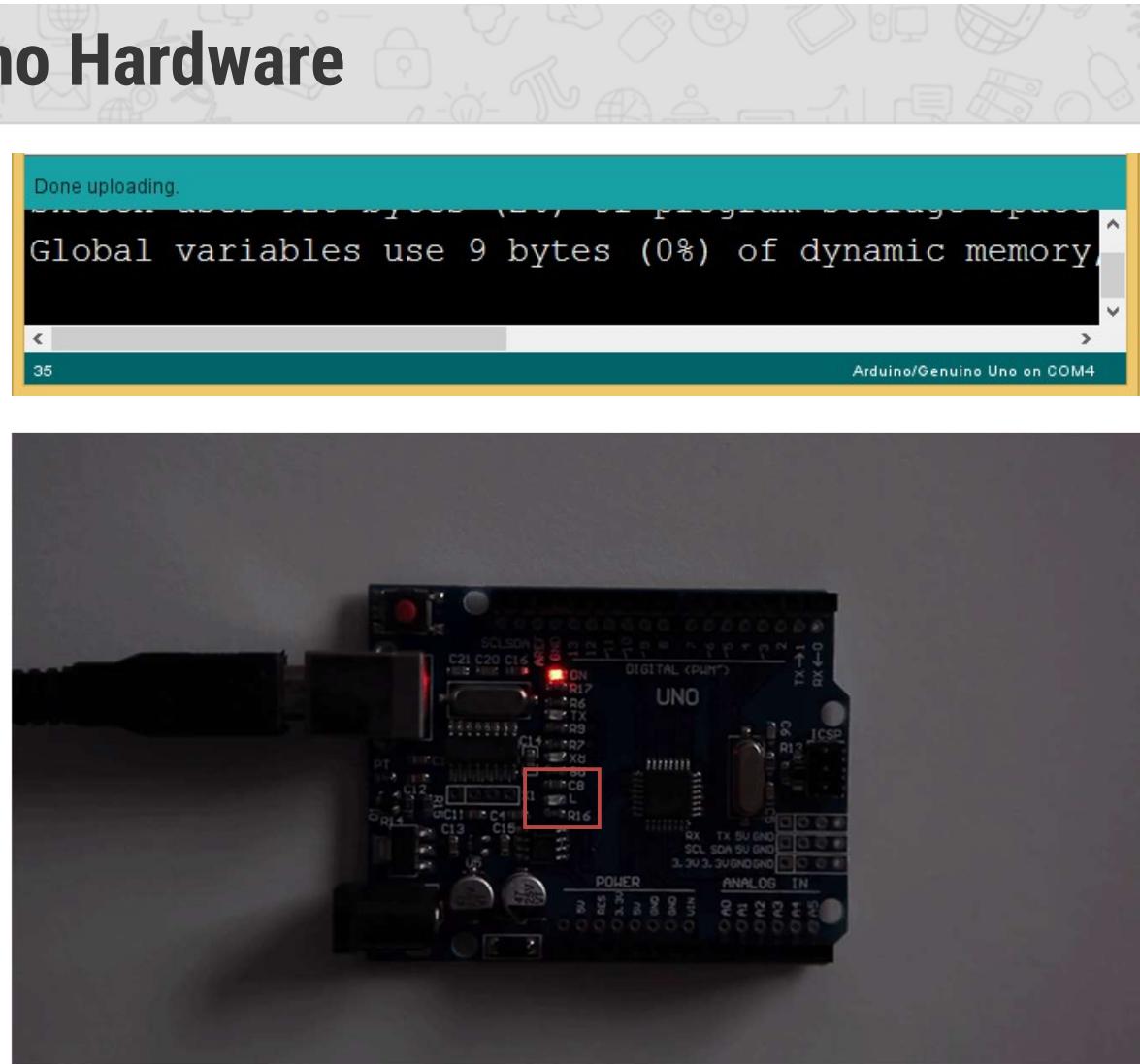
# Upload the first code in Arduino Hardware

- ▶ Select the proper board type: First, we need to select the appropriate board type in which we need to upload the program.
- ▶ Go to Tools > Board menu and select the board which is used as the hardware.
- ▶ Select port: Select the serial device of the board from the Tools > Port menu.
- ▶ This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports).
- ▶ Upload the program: Now, simply click the "Upload" button in the environment.



# Upload the first code in Arduino Hardware

- ▶ While uploading process, TX and RX LED will be flashing.
- ▶ If uploading is done successfully, it will show a message “Done uploading” in the status bar.
- ▶ Once the program is uploaded, the built-in LED of the Arduino board will flash with one second delay.
- ▶ It means LED turns ON for 1 second and LED turns OFF for 1 second.





Unit-6

# Arduino and Raspberry Pi



  
**Prof. Kalpesh H Surati**  
Computer Engineering Department  
Darshan Institute of Engineering & Technology, Rajkot  

---

✉ Kalpesh.surati@darshan.ac.in  
📞 +91 99250 10033

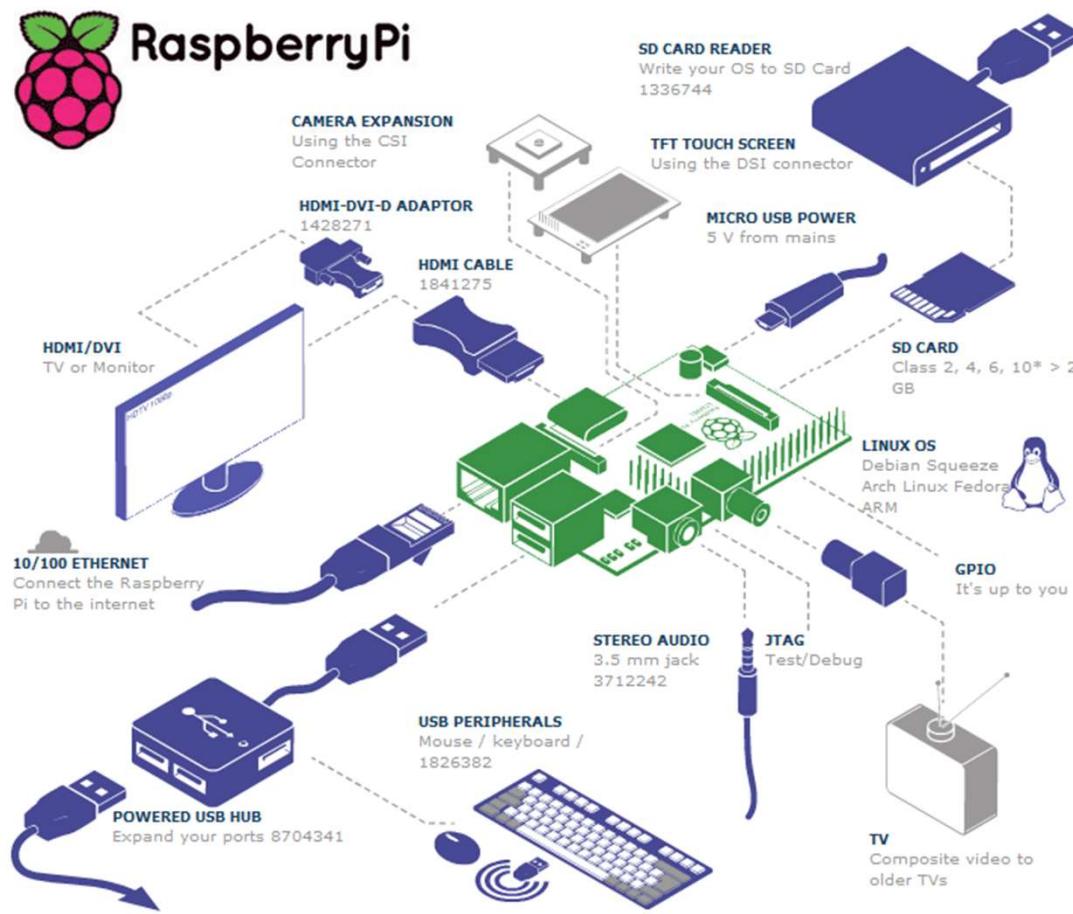
# Raspberry Pi

## ► What is Raspberry Pi?



- Raspberry Pi is a single-board computers, developed by Raspberry Pi Foundation in association with Broadcom and perhaps the most inspiring computer available today.
- Because of its low cost and open design, the model became far more popular than anticipated.
- It is widely used to make gaming devices, fitness gadgets, weather stations, and much more.
- In 2012, the company launched the Raspberry Pi and the current generations of regular Raspberry Pi boards are Zero, 1, 2, 3, and 4.

# Architecture, Layout and Interface of Raspberry Pi



- ▶ Processor
- ▶ HDMI Port
- ▶ USB Port
- ▶ Ethernet Port
- ▶ SD Card Slot
- ▶ Camera Connector
- ▶ Composite Video Output
- ▶ Audio Output
- ▶ Micro USB Power
- ▶ GPIO Pins
- ▶ Status LEDs

# Operating System for Raspberry Pi

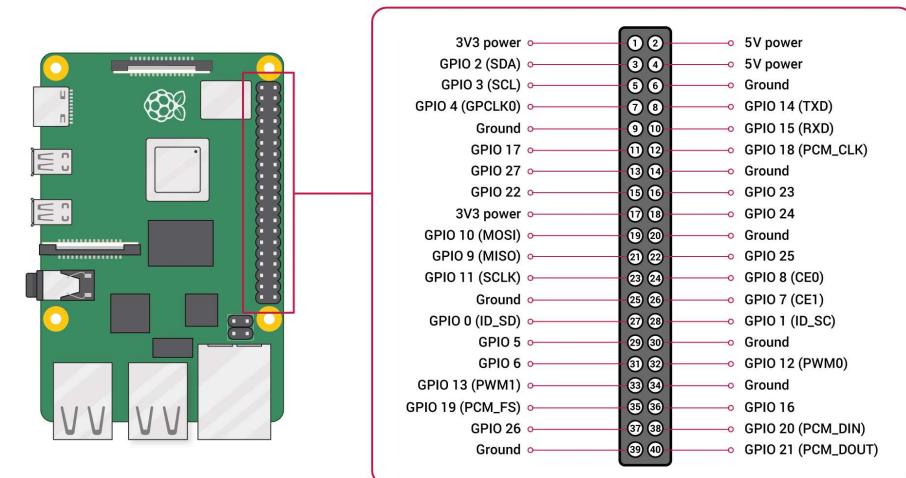
- ▶ Raspberry Pi needs an operating system to work.
- ▶ Raspberry Pi OS (previously called Raspbian) is an official supported operating system.
- ▶ Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi.
- ▶ Download and install Raspberry Pi Imager to a computer with an SD card reader.
- ▶ Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

# NOOBS

- ▶ NOOBS (New Out Of the Box Software) is an exclusive install manager for OS installation process in the SD card of Raspberry Pi.
- ▶ NOOBS is extremely useful for beginners as it helps in easy OS installation without configuration process.
- ▶ Different operating system that could be installed using NOOBS are.
  - Raspbian
  - OpenELEC
  - Pidora
  - RaspBMC
  - RISC OS
  - Arch Linux ARM

# GPIO and the 40-pin Header

- ▶ A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board.
- ▶ A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Raspberry Pi Zero, Raspberry Pi Zero W and Raspberry Pi Zero 2 W).
- ▶ Prior to the Raspberry Pi 1 Model B+ (2014), boards comprised a shorter 26-pin header.
- ▶ One of the things that makes the Raspberry Pi better for learning electronics than most other computers is its ability to control the voltage on several of its easily accessible pins.
- ▶ GPIO header contains outputs for 3.3V, 5V, Ground, and lots of General Purpose Input/Output (GPIO) pins!



# Programming

- ▶ Python is a powerful programming language that's easy to use, easy to read and write, and, with Raspberry Pi
- ▶ Python syntax is clean, with an emphasis on readability, and uses standard English keywords.
- ▶ To control hardware connected to the Raspberry Pi we will use Python.
- ▶ One should be familiar with some of the basics of Python, including literals, variables, operators, control flow, scope, and data structures.
- ▶ We'll use the RPi.GPIO module to control all the GPIO of Raspberry Pi

# Python Programming for GPIO

- ▶ Import the RPi.GPIO module
  - import RPi.GPIO as GPIO
- ▶ Pin Numbering Declaration
  - After you've included the RPi.GPIO module, the next step is to determine which of the two pin-numbering schemes you want to use:
    - GPIO.BOARD -- Board numbering scheme. The pin numbers follow the pin numbers on header P1.
    - GPIO.BCM -- Broadcom chip-specific pin numbers. These pin numbers follow the lower-level numbering system defined by the Raspberry Pi's Broadcom-chip brain.
- ▶ To specify in your code which number-system is being used, use the GPIO.setmode() function.
  - GPIO.setmode(GPIO.BCM)
- ▶ Both the import and setmode lines of code are required, if you want to use Python.

# Python Programming for GPIO

## ▶ Setting a Pin Mode

- Similar to declare a "pin mode" in Arduino before you can use it as either an input or output.
- To set a pin mode, use the `setup([pin], [GPIO.IN, GPIO.OUT])` function.
- So, if you want to set pin 18 as an output,  
`GPIO.setup(18, GPIO.OUT)`

## ▶ Outputs

- Digital Output
- To write a pin high or low, use the `GPIO.output([pin], [GPIO.LOW, GPIO.HIGH])` function.
- For example, if you want to set pin 18 high, write:  
`GPIO.output(18, GPIO.HIGH)`
- Writing a pin to `GPIO.HIGH` will drive it to 3.3V, and `GPIO.LOW` will set it to 0V.
- Alternative to `GPIO.HIGH` and `GPIO.LOW`, you can use either 1, True, 0 or False to set a pin value.

# Python Programming for GPIO

## ▶ Outputs

- PWM ("Analog") Output
- PWM on the Raspberry Pi is about as limited as can be -- one, single pin is capable of it: 18 (i.e. board pin 12).
- To initialize PWM, use `GPIO.PWM([pin], [frequency])` function.
- To make the rest of your script-writing easier you can assign that instance to a variable.
- Then use `pwm.start([duty cycle])` function to set an initial value.
- For example...

```
pwm = GPIO.PWM(18, 1000)  
pwm.start(50)
```

# Python Programming for GPIO

## ▶ Inputs

- If a pin is configured as an input, you can use the GPIO.input([pin]) function to read its value.
- The input() function will return either a True or False indicating whether the pin is HIGH or LOW.
- You can use an if statement to test this, will read pin 17 and print whether it's being read as HIGH or LOW

```
if GPIO.input(17):  
    print("Pin 11 is HIGH")  
else:  
    print("Pin 11 is LOW")
```

# Python Code for Blinking LED

Blink.py

```
1 import time
2 import RPi.GPIO as GPIO
3 led_pin = 12
4
5 GPIO.setmode(GPIO.BCM)
6
7 GPIO.setup(led_pin, GPIO.OUT)
8
9 while True:
10     GPIO.output(led_pin, GPIO.HIGH)
11     time.sleep(1)
12     GPIO.output(led_pin, GPIO.LOW)
13     time.sleep(1)
14
```

# Pin definitions  
# Use "GPIO" pin numbering  
# Set LED pin as output  
# Blink forever  
# Turn LED on  
# Delay for 1 second  
# Turn LED off  
# Delay for 1 second



Unit-7

# IoT Security



**Prof. Kalpesh H Surati**  
Computer Engineering Department  
Darshan Institute of Engineering & Technology, Rajkot  
  
✉ Kalpesh.surati@darshan.ac.in  
📞 +91 99250 10033



# Introduction to IoT Security

## Overview

- ▶ IoT is growing day by day, as we know it's about data and controlling of physical devices.
- ▶ Security and privacy are the two major concern in the field of IoT.
- ▶ Huge amount of sensed data contains private information so need to protect.
- ▶ All kind of securities of physical devices is considered in the IoT security.
- ▶ IoT is not possible without the Internet so Internet and network security issues also should be considered in it.



# Introduction to IoT Security

## Overview

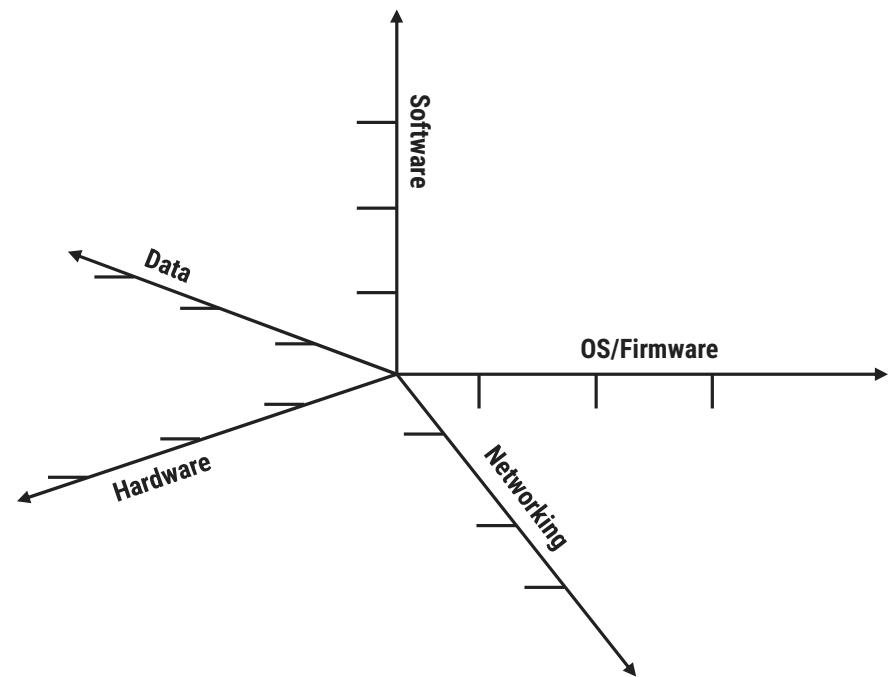
- ▶ IoT security is not traditional cybersecurity
- ▶ It's a fusion of cybersecurity with other engineering disciplines.
- ▶ It is much more than data, servers, network infrastructure, and information security.
- ▶ It includes the direct monitoring and control of the physical systems connected over the Internet.
- ▶ IoT devices are physical things, many of which are safety-related.
- ▶ The compromise of such devices may lead to physical harm of persons and property, or even death



# IoT Security Prospective

## IoT System Functionalities from Security Prospective

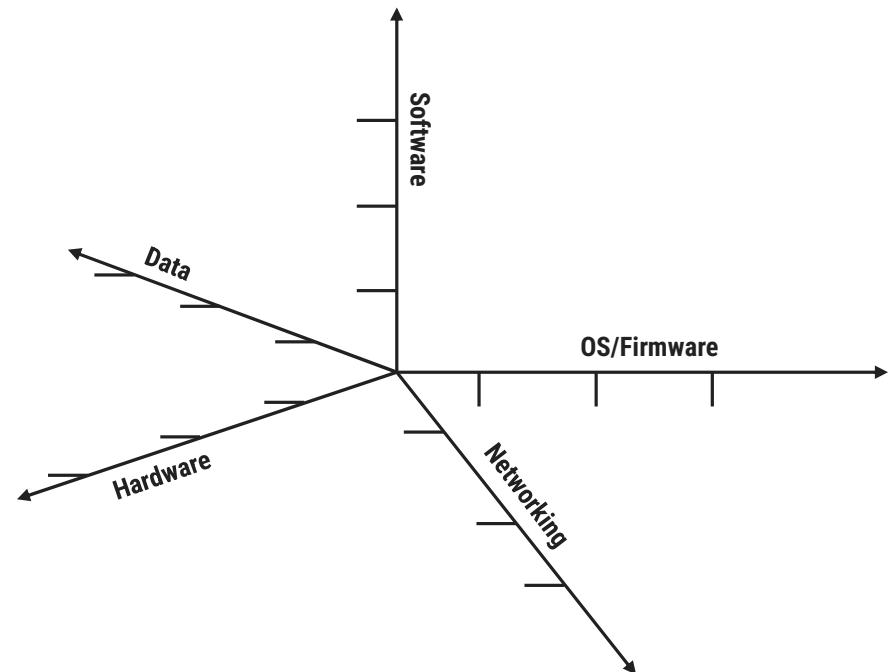
- ▶ Microcontroller unit carries firmware, need to protect it even while updating patch.
- ▶ Message channels during the paring stage need to protect in the public networking, like
  - Wi-Fi, Zigbee
  - Bluetooth
  - NFC
- ▶ An appropriate protocol should be followed while connecting the user and device.
- ▶ An authentication process is needed when the controller linking to a port in local network.



Multidimensional Prospective of IoT Security

## IoT System Functionalities from Security Prospective

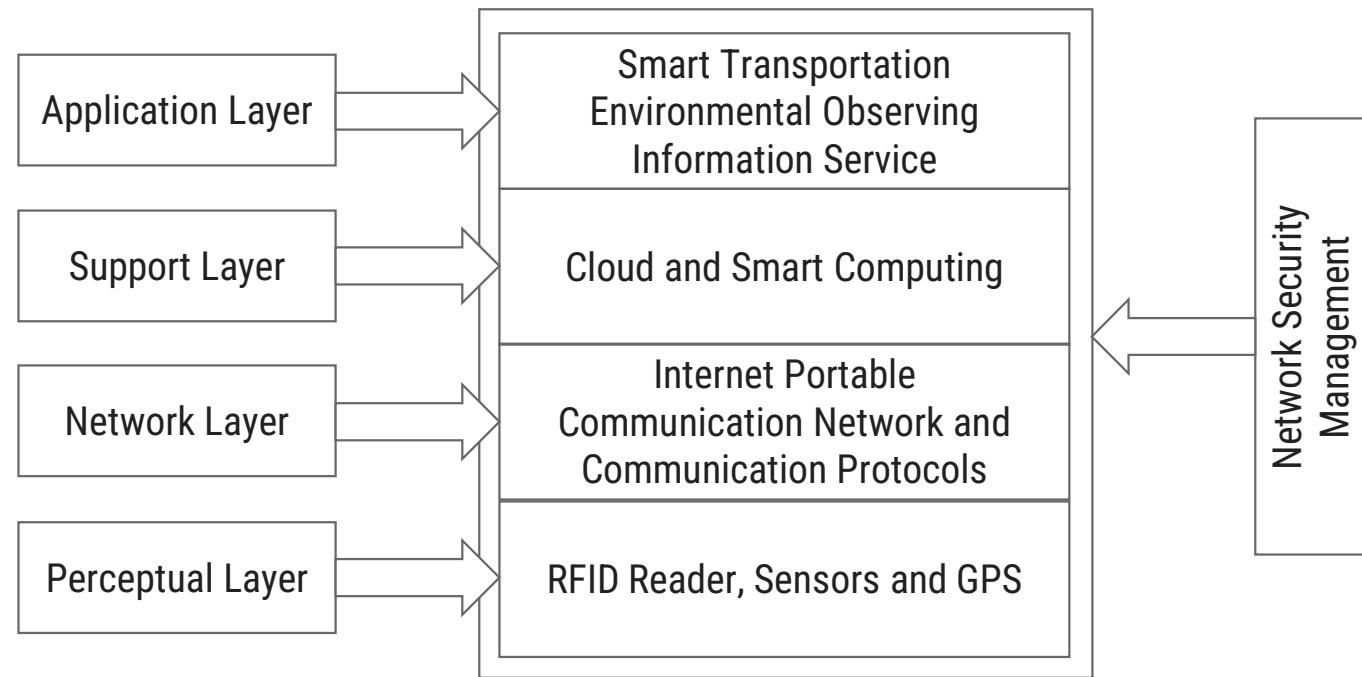
- ▶ If the controller is no internet then cloud services are used for authentication. multidimensional
- ▶ Big data analytics on the data collected are processed on cloud so cloud security is essential.
- ▶ Abnormal behavior should be monitored like too many login attempts



Multidimensional Perspective of IoT Security

# IoT Security Architecture

- ▶ Information with network security should be prepared with the following properties.
  - Authentication
  - Privacy
  - Undeniability
- ▶ IoT will be needed extra care for advanced security and privacy across critical areas.

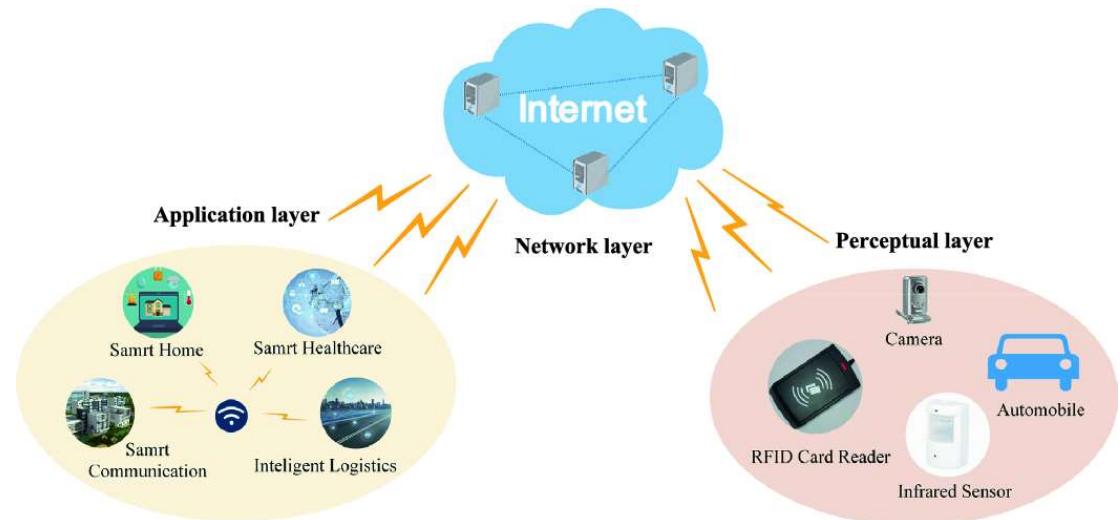


## IoT Security Architecture

# IoT Security Architecture

## Perceptual Layer

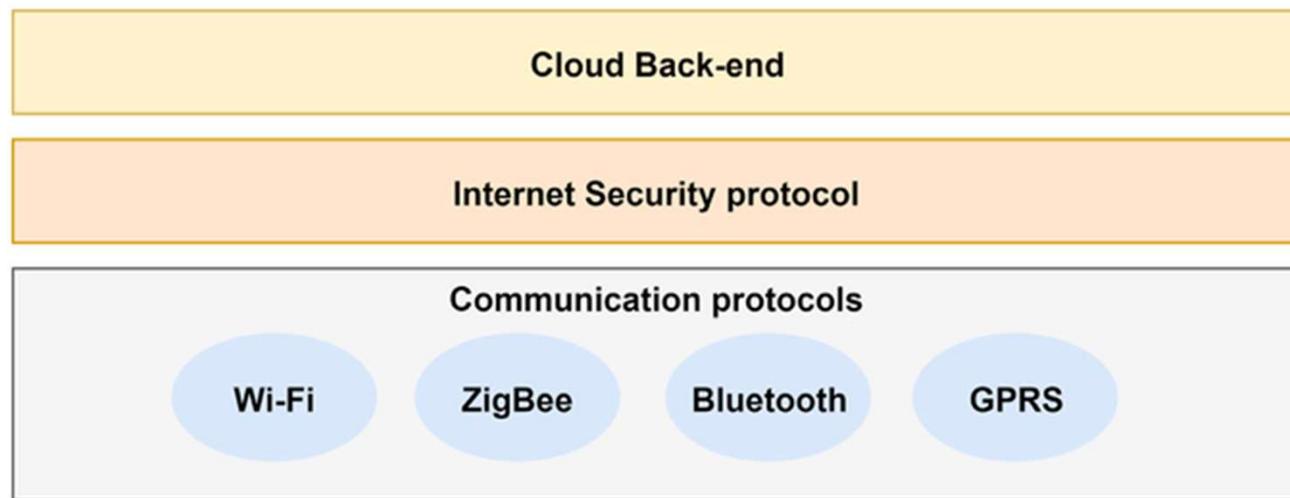
- ▶ Gathers all types of information with the help of physical equipment.
- ▶ Information of
  - Object properties,
  - Environmental condition and
  - The different physical equipment like
    - RFID reader,
    - GPS,
    - All kind of sensors, etc.
- ▶ It identifies the external world.
- ▶ The key component in this layer is the sensors.
- ▶ They are used for capturing and representing the physical world.



# IoT Security Architecture

## Network Layer

- ▶ Responsible for the dependable broadcast of data and information from the previous level
- ▶ Initially handling of the data collected from sensors, cataloging and polymerization.
- ▶ The data broadcast is trusted on many networks like
  - Mobile communication network
  - Wireless network
  - Satellite networks, etc.



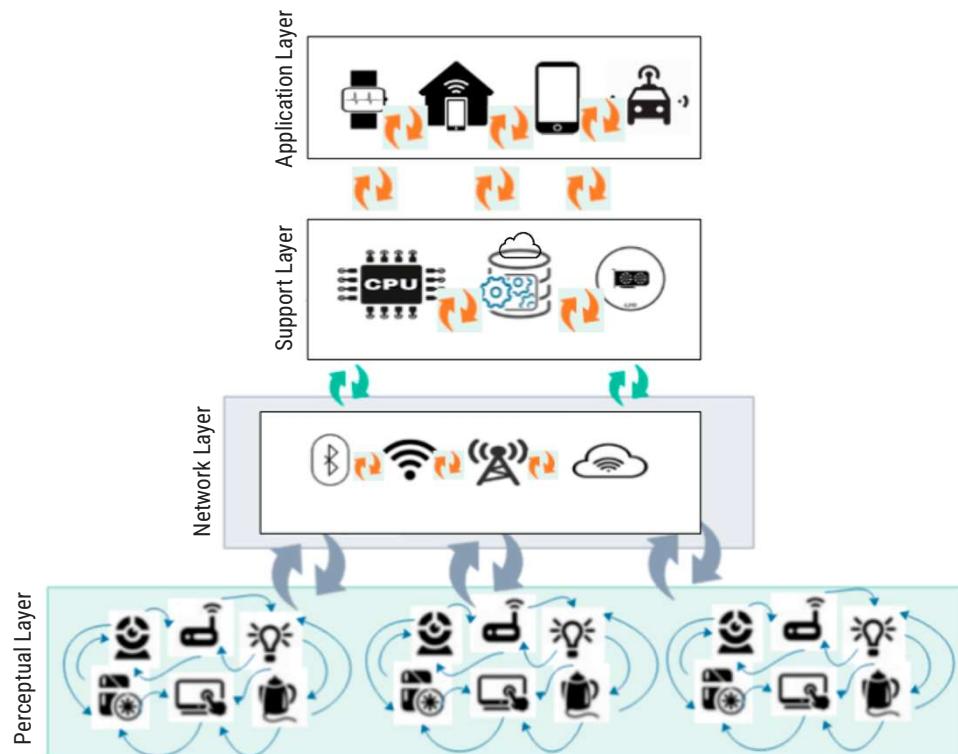
# IoT Security Architecture

## Support Layer

- ▶ A dependable platform for the application layer.
- ▶ Grid and cloud computing are mostly used for all kinds of intelligent computing powers.
- ▶ This layer helps merge the application layer upward and the network layer downward.

## Application Layer

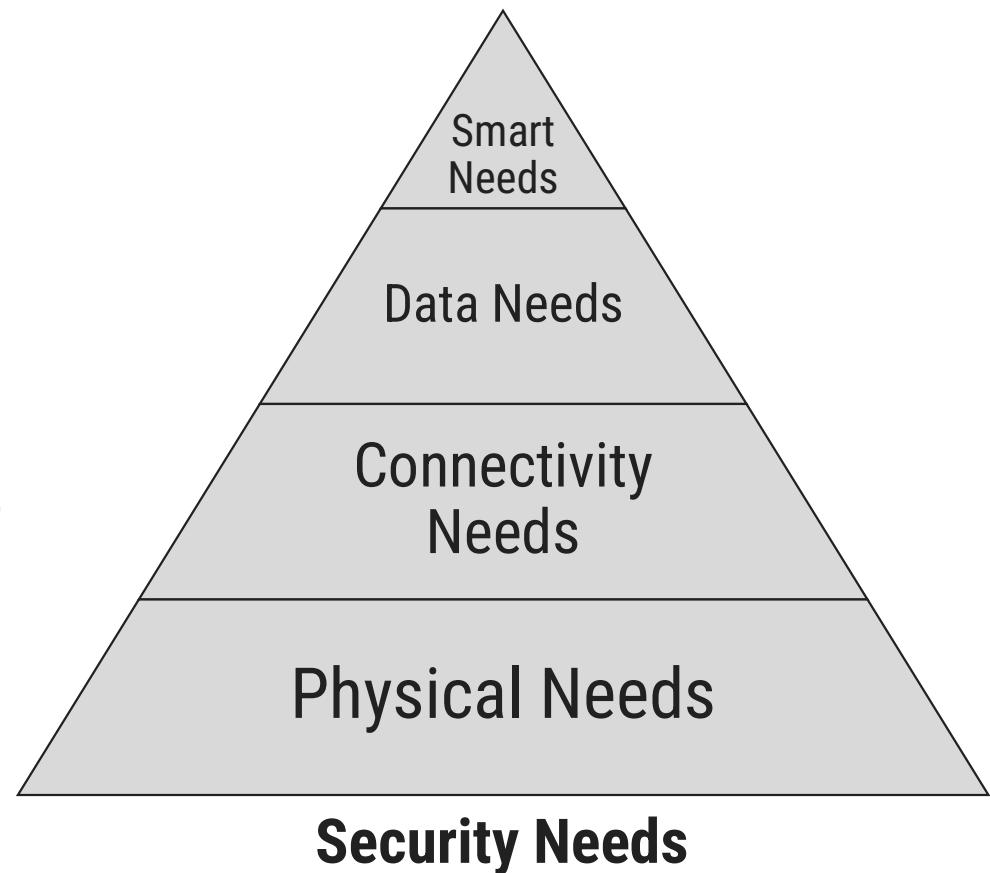
- ▶ This layer delivers the personalized services based on the users' need.
- ▶ It helps users access IoT through the interface using personal computer, mobile equipment, etc.



# Security Features Need Across Four Layers

## Perceptual Layer

- ▶ With a simple architecture and less power, this layers dose not have storage and computation power.
- ▶ Applying public key encryption algorithm and frequency hopping communication is not possible here.
- ▶ So security is necessary and needed for some threats from external network like DoS attacks.
- ▶ Due to all the reason the sensor data to be protected for authenticity, integrity, and confidentiality.



# Security Features Need Across Four Layers

## Network Layer

- ▶ Security vulnerabilities are like man-in-the-middle attack, still exists even the main network has enough safety feature.
- ▶ Malwares and junk mails cannot be ignored.
- ▶ Data blocking may occurs because of huge amount of data transmission.
- ▶ Because of all the above reason security methods are needed.

## Support Layer

- ▶ It is a challenge to increase the ability to identify malicious data in this layer due to the huge amount of data processing and mining.

## Application Layer

- ▶ In this layer, security needs may differ from application to application
- ▶ Data sharing property of the layer does lead to privacy problem, access control issues, and information revelation to unintended persons.

# Security Requirements

- ▶ A dynamic IoT technology has lots of security challenges.
- ▶ The laws and regulations surrounding the challenges also play a significant role.

## Perceptual Layer

- ▶ Authentication is the first level of security measure and is always essential to prevent any illegal access to the node.
- ▶ Information confidentiality is taken care during transmission between nodes
- ▶ Because of limited resource, lightweight encryption technology may help in stronger data safety measures. It including cryptographic protocol and algorithms.
- ▶ Similarly need care for the authenticity and integrity of the data in this layer

## Network Layer

- ▶ Establishing data confidentiality and integrity mechanism is the priority in these days.
- ▶ Identity verification is one of the methods to avoid illegal nodes.
- ▶ DDoS attack in the network is a serious issue in the IoT domain.

# Security Requirements

## Support Layer

- ▶ Cloud computing along with secure multi-party computation falls under this layer of security needs.
- ▶ Different encryption algorithms along with the encryption protocol and tougher system security technology are hence essential in this layer.

## Application Layer

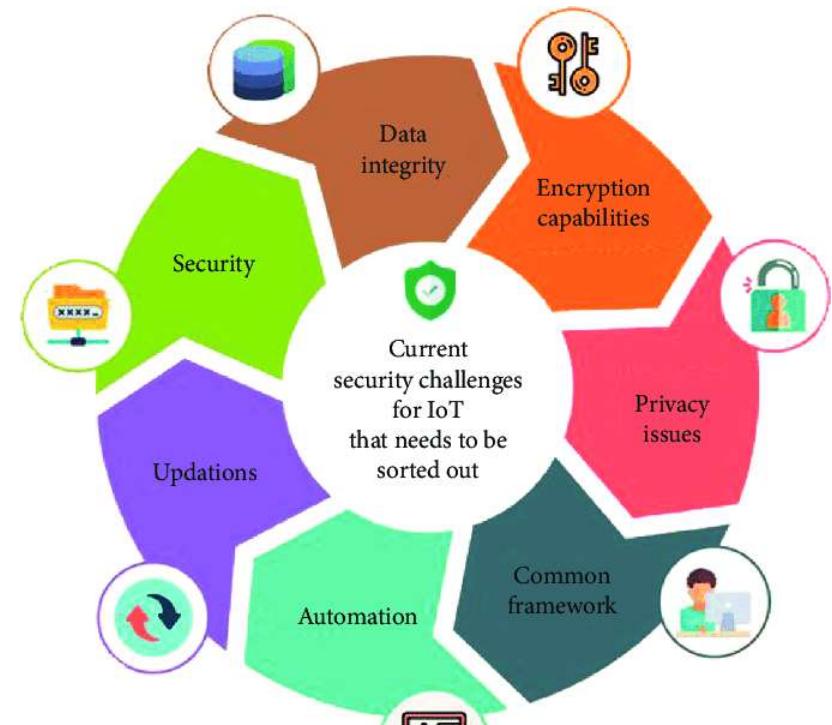
- ▶ In the topmost layer, verification and key contract across the varied network needed as security features.
- ▶ Also consider the user's confidentiality protection in the layer.
- ▶ Along with these two aspects education and management are also very imperative for data security.
- ▶ This helps IoT security consulting and certification services.

# Challenges in IoT Securities

- ▶ In the raising IoT field many problems to be solved to build an efficient and effective product.
- ▶ Securities challenges are one of them.

## Encryption

- ▶ Encryption play key role in the security, but many devices cannot perform the complex encryption and decryption quickly because of limited resource.
- ▶ Products with constrained resources are most likely to attacks.
- ▶ Reverse engineering of algorithm is possible on it.



# Challenges in IoT Securities

## Authorization and Authentication

- ▶ Device authorization and authentication is critical to securing IoT products
- ▶ The things establish their identity before accessing gateway and other cloud related activities.
- ▶ IoT platform with two factor authentication and usage of strong passwords or certificates can help to solve this issue.
- ▶ They can also help to know which services or apps each device has access to throughout the system.



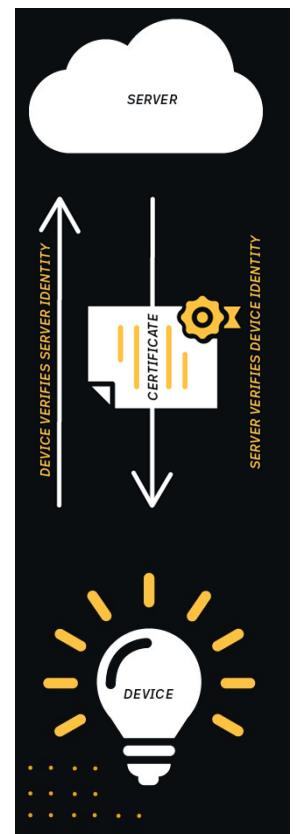
# Challenges in IoT Securities

## Firmware Updates

- ▶ Device updates needs to be managed effectively.
- ▶ Security patches to firmware or software will have a number of challenges.
- ▶ Over-the-air updates may not be possible with all types of IoT devices.
- ▶ The device owners may also not show much interest in applying an update to the system.

## Communication Channel

- ▶ The communication channel needs to be secure as well
- ▶ Encrypting messages before transfer is good but it is better to use transport encryption and to adopt standards like TLS.



# Challenges in IoT Securities

## Data Storage and Integrity

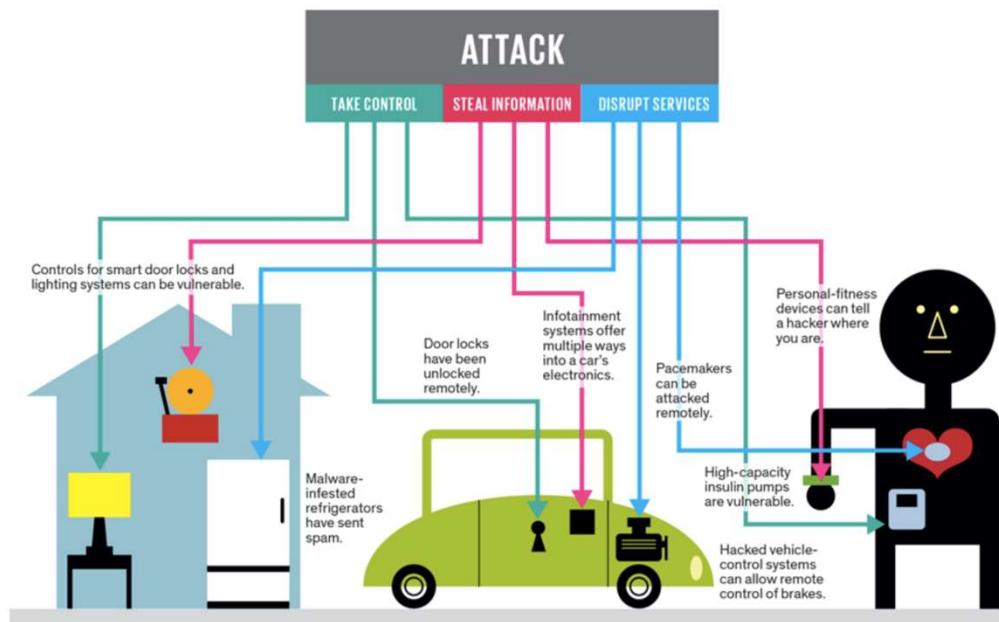
- ▶ The sensor data should be stored and processed securely.
- ▶ Data integrity, including checksums or signatures, can help to make sure that the original raw data is not modified during transmission.
- ▶ Data should be erased in a better way and should not be recovered in any part of the system.
- ▶ Maintaining compliance with legal and regulatory framework is necessary and challenging also.



# Challenges in IoT Securities

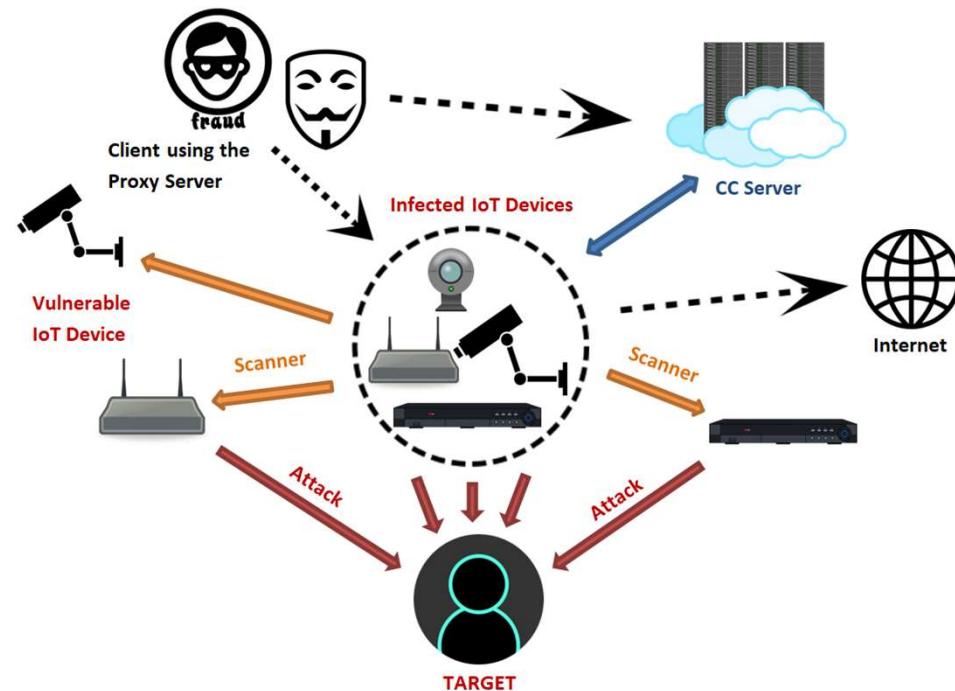
## Application and Services

- ▶ All applications and services should also be secured as they manage, process, and access IoT devices along with the sensor data.
- ▶ Security vulnerabilities and breaches are unavoidable but security measures need to be taken to avoid conflict of interest.



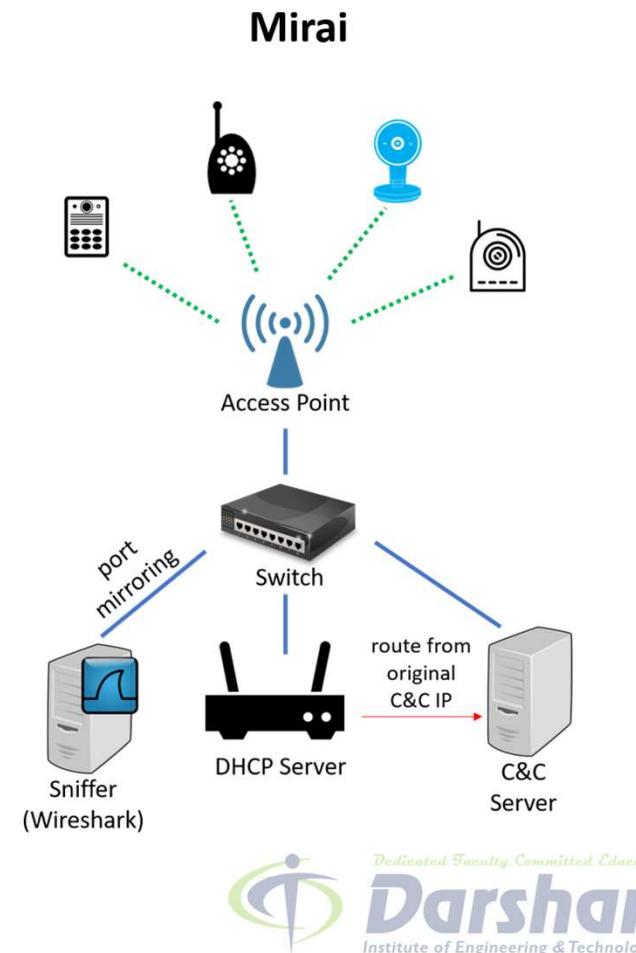
# Mirai Botnet and the Algorithm

- ▶ Mirai Trojan is the main reason of creation of Mirai Botnet.
- ▶ A research group determined that it had evolved from a previously-created Trojan also known Gafgyt, Bashlite, Lizkebab, Bashdoor, Bash()day, and Torlus.
- ▶ It was created using Executable and Linkable Format (EFL) binaries which is a common file format for Unix and Linux based Systems
- ▶ Mirai malware uses a uniform scanning strategy where it randomly scan public IP addresses and selects a pair of username/password from a hardcoded dictionary list for the attack.



# Mirai Botnet and the Algorithm

- ▶ Devices that are infected by Mirai will continuously scan for IP address in the internet of the IoT devices.
- ▶ Mirai then identifies the IoT devices that are vulnerable using the common factory default usernames and passwords.
- ▶ Infect them with Mirai malware.
- ▶ There are over a hundred of thousands IoT devices using default settings and making them vulnerable to infection.
- ▶ It is also reported that a successor of Mirai is designed to hijack crypto currency mining operations.
- ▶ The source code for the Mirai is made open-source and the techniques have been adapted in other malware projects.



# Summary

- ▶ A multi layered security design approach is most needed for managing IoT devices, sensor data, mobile and cloud related application
- ▶ This enables us to maintain data privacy and integrity while also delivering IoT data, apps and services without any compromises.
- ▶ In short, IoT development and advancement will bring more security related issues, which is always going to be the research focus in coming years.