JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

B.Tech CSE (AI)– III-II Sem **L T P C**

3 0 0 3

(20A05603T) INTERNET OF THINGS

Common to CSE, IT, CSD, CSE(AI), CSE(DS),AI&DS

Course Objectives:

- Understand the basics of Internet of Things and protocols.
- Discuss the requirement of IoT technology
- Introduce some of the application areas where IoT can be applied.
- Understand the vision of IoT from a global perspective, understand its applications, determine its market perspective using gateways, devices and data management

Course Outcomes:

After completion of the course, students will be able to

- Understand general concepts of Internet of Things.
- Apply design concept to IoT solutions
- Analyze various M2M and IoT architectures
- Evaluate design issues in IoT applications
- Create IoT solutions using sensors, actuators and Devices

UNIT I Introduction to IoT **Lecture 8Hrs**

Definition and Characteristics of IoT, physical design of IoT, IoT protocols, IoT communication models, IoT Communication APIs, Communication protocols, Embedded Systems, IoT Levels and Templates

UNIT II Prototyping IoT Objects using Microprocessor/Microcontroller **Lecture 9Hrs**

Working principles of sensors and actuators, setting up the board – Programming for IoT, Reading from Sensors, Communication: communication through Bluetooth, Wi-Fi.

UNIT III IoT Architecture and Protocols **Lecture 8Hrs**

Architecture Reference Model- Introduction, Reference Model and architecture, IoT reference Model, Protocols- 6LowPAN, RPL, CoAP, MQTT, IoT frameworks- Thing Speak.

UNIT IV Device Discovery and Cloud Services for IoT **Lecture 8Hrs**

Device discovery capabilities- Registering a device, Deregister a device, Introduction to Cloud Storage models and communication APIs Web-Server, Web server for IoT.

UNIT V UAV IoT **Lecture 10Hrs**

Introduction toUnmanned Aerial Vehicles/Drones, Drone Types, Applications: Defense, Civil, Environmental Monitoring; UAV elements and sensors- Arms, motors, Electronic Speed Controller(ESC), GPS, IMU, Ultra sonic sensors; UAV Software –Arudpilot, Mission Planner, Internet of Drones(IoD)- Case study FlytBase.

Textbooks:

1. Vijay Madisetti and ArshdeepBahga, “Internet of Things (A Hands-on-Approach)”, 1st Edition, VPT, 2014.
2. Handbook of unmanned aerial vehicles, K Valavanis; George J Vachtsevanos, New York, Springer, Boston, Massachusetts: Credo Reference, 2014. 2016.

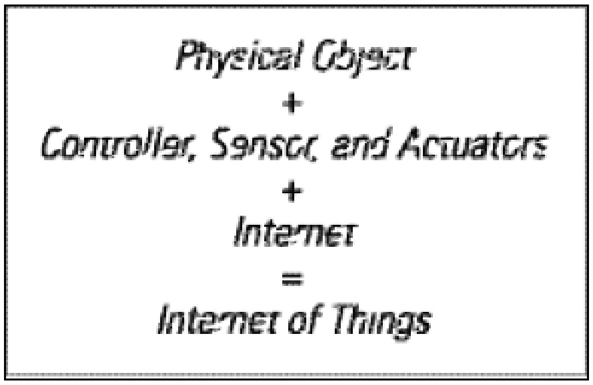
Reference Books:

1. Jan Holler, VlasiosTsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, “ From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence”, 1st Edition, Academic Press, 2014.
2. ArshdeepBahga, Vijay Madisetti - Internet of Things: A Hands-On Approach,

Definition and Characteristics of IoT, physical design of IoT, IoT protocols, IoT communication models, IoT Communication APIs, Communication protocols, Embedded Systems, IoT Levels and Templates

Definition:

The Internet of Things (IOT) is the network of physical objects or "things" embedded with electronics, software, sensors and network connectivity which enables these objects to collect and exchange data.



An equation for the Internet of Things.

History

1997, “The Internet of Things” is the seventh in the series of ITU Internet Reports originally launched in 1997 under the title “Challenges to the Network”.

1999, Auto-ID Center founded in MIT

2003, EPC Global founded in MIT

2005, Four important technologies of the internet of things was proposed in WSIS conference. 2008, First international conference of internet of things: The IOT 2008 was held at Zurich.

IoT (Internet of Things) is an advanced automation and analytics system which exploits networking, sensing, big data, and artificial intelligence technology to deliver complete systems for a product or service. These systems allow greater transparency, control, and performance when applied to any industry or system.

IoT systems allow users to achieve deeper automation, analysis, and integration within a system. They improve the reach of these areas and their accuracy. IoT utilizes existing and emerging technology for sensing, networking, and robotics.

IoT exploits recent advances in software, falling hardware prices, and modern attitudes towards technology. Its new and advanced elements bring major changes in the delivery of products, goods, and services; and the social, economic, and political impact of those changes

IoT − Key Features

The most important features of IoT include artificial intelligence, connectivity, sensors, active engagement, and small device use. A brief review of these features is given below −

- **AI −** IoT essentially makes virtually anything “smart”, meaning it enhances every aspect of life with the power of data collection, artificial intelligence algorithms, and networks. This can mean something as simple as enhancing your refrigerator and cabinets to detect when milk and your favorite cereal run low, and to then place an order with your preferred grocer.
- **Connectivity** − New enabling technologies for networking, and specifically IoT networking, mean networks are no longer exclusively tied to major providers. Networks can exist on a much smaller and cheaper scale while still being practical. IoT creates these small networks between its system devices.
- **Sensors −** IoT loses its distinction without sensors. They act as defining instruments which transform IoT from a standard passive network of devices into an active system capable of real-world integration.
- **Active Engagement** − Much of today's interaction with connected technology happens through passive engagement. IoT introduces a new paradigm for active content, product, or service engagement.
- **Small Devices −** Devices, as predicted, have become smaller, cheaper, and more powerful over time. IoT exploits purpose-built small devices to deliver its precision, scalability, and versatility.

IoT − Advantages

The advantages of IoT span across every area of lifestyle and business. Here is a list of some of the advantages that IoT has to offer −

- **Improved Customer Engagement** − Current analytics suffer from blind-spots and significant flaws in accuracy; and as noted, engagement remains passive. IoT completely transforms this to achieve richer and more effective engagement with audiences.
- **Technology Optimization** − The same technologies and data which improve the customer experience also improve device use, and aid in more potent improvements to technology. IoT unlocks a world of critical functional and field data.
- **Reduced Waste** − IoT makes areas of improvement clear. Current analytics give us superficial insight, but IoT provides real-world information leading to more effective management of resources.
- **Enhanced Data Collection −** Modern data collection suffers from its limitations and its design for passive use. IoT breaks it out of those spaces, and places it exactly where humans really want to go to analyze our world. It allows an accurate picture of everything.

IoT − Disadvantages

Though IoT delivers an impressive set of benefits, it also presents a significant set of challenges. Here is a list of some its major issues −

- **Security** − IoT creates an ecosystem of constantly connected devices communicating over networks. The system offers little control despite any security measures. This leaves users exposed to various kinds of attackers.

- **Privacy** − The sophistication of IoT provides substantial personal data in extreme detail without the user's active participation.
- **Complexity** − Some find IoT systems complicated in terms of design, deployment, and maintenance given their use of multiple technologies and a large set of new enabling technologies.
- **Flexibility** − Many are concerned about the flexibility of an IoT system to integrate easily with another. They worry about finding themselves with several conflicting or locked systems.
- **Compliance** − IoT, like any other technology in the realm of business, must comply with regulations. Its complexity makes the issue of compliance seem incredibly challenging when many consider standard software compliance a battle

The Technology of the Internet of Things

The Internet of Things Technology is: through radio frequency identification(RFID) ,infrared sensors, global positioning systems, laser scanners and other information sensing equipment, according to the agreed agreement, any item is connected to the internet for information exchange and communication to achieve a network technology for intelligent identification, positioning, tracking ,monitoring and management

What is the internet of Things technology?

The core and foundation of "Internet of Things" is still "Internet Technology" which is a kind of network technology that is extended and expanded on the basis of internet technology communication

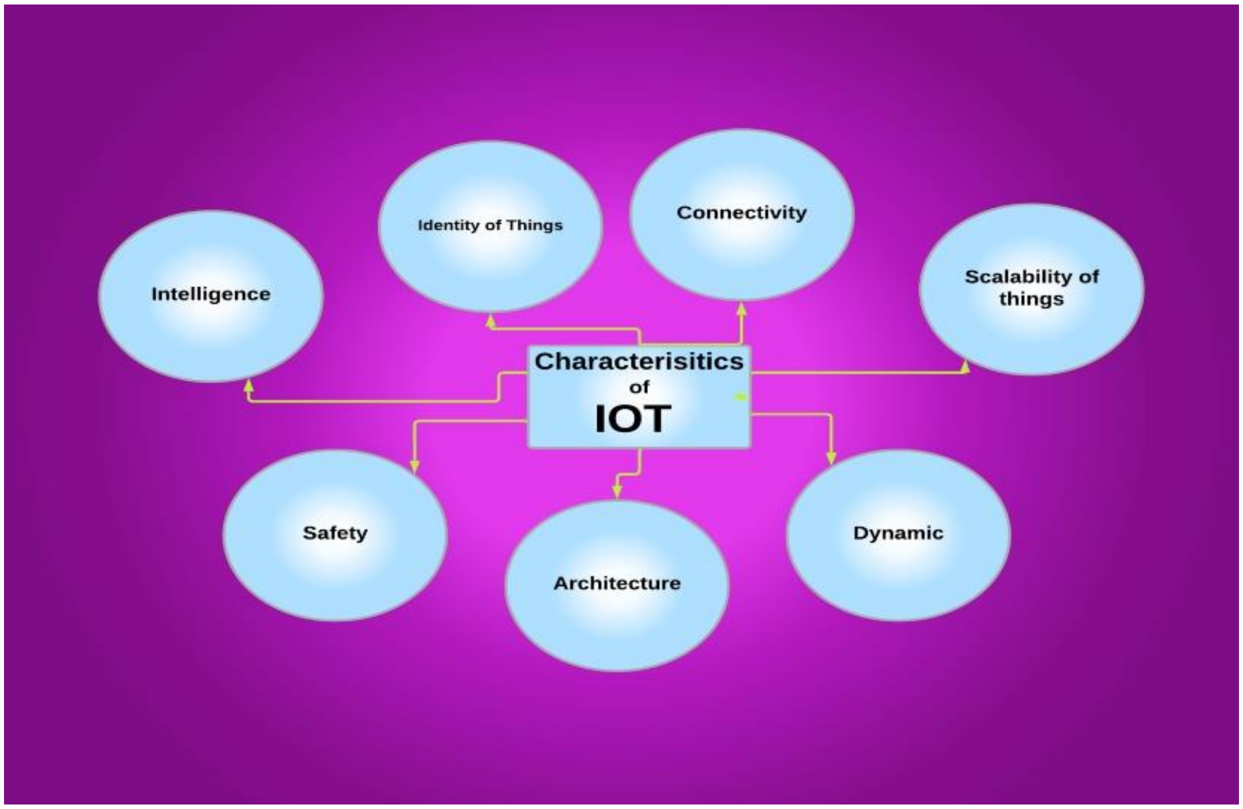
There are three key technologies in IoT applications:

1. Sensor technology, which is also key technology in computer applications. As everyone knows, most computers deal with digital signals so far, Since there are computers, sensors have been needed to convert analog signals into digital signals for computers to process

2. RFID tag is also a kind of sensor technology, RFID technology is a comprehensive technology integrating radio frequency technology and embedded technology, RFID has broad application prospects in automatic identification and item logistics management

3. Embedded system technology: it is complex technology that integrates computer software and hardware, sensor technology integrated circuit technology and electronic application technology. After decades of evolution, smart terminal products featuring embedded systems can be seen everywhere: from MP3 around people to satellite systems for aerospace and aviation. Embedded systems are changing people’s lives and promoting the development of industrial production and the defence industry. If the internet of Things is used as simple analogy with the human body, Sensors are equivalent to human senses such as eyes, nose and skin

Characteristics of IOT:



Characteristics of IoT

1. Connectivity

Connectivity is an essential feature of IoT. IoT lets you connect mobile phones, laptops, and other internet devices. Any person can get information about anything at any time and place.

IoT can connect through several wireless devices, like sensors, mobile phones, trackers, etc. This way, the person will not have to wait for an internet connection to operate a device.

2. Identity of Things

The collaboration of name and number gives an identity to an internet device. Giving an identity to the device is an essential aspect of loT. Identity helps to differentiate between various internet devices and select the device we want to send the command.

Every device needs a different controlling power based on the type of data provided. It is essential to give a unique identity to every device so that we can set up passwords or other security means. For example, fingerprints face recognition IP addresses, and Face lock systems are several means of security given to the different identified devices to protect them.

3. Intelligence

The intelligence of IoT devices depends on the sensors' intelligence. The sensors send the data to the user for further analysis.
We need to update the IoT devices regularly to get the smart work done. It adds to their features and makes them smarter.

4. Dynamic

We need to create IoT devices in a way that they can adapt to the environment. For example, an AC should have a sensor that can send a signal to the cloud and adjust it to the premises of the place. Similarly, the camera can easily click photographs by adjusting to light situations, like day and night.

5. Scalability

Scalability means the amount of data one can handle efficiently. The IoT has created a setup to handle enormous data and generate useful analysis.

6. Self up gradation

As we saw above, updating the software regularly is important. But who has the time to remember to do that? Thus, with its artificial intelligence, IoT upgrades itself without human help. It also allows the set up of a network for the addition of any new IoT devices. Thus, the technology can quickly start working without delay if the setup has already been done.

7. Architecture

The architecture of IoT is designed in a way that it is capable of supporting various devices, technologies, and protocols. Its main work is to confirm whether each connected device does not interfere with the other. This way, the safety and security of each device's data are maintained.

8. Security

With the increasing number of IoT devices, issues regarding the security of personal data have arisen. There might be a chance of data leakage as a large amount of data is collected, exchanged, and generated. There is a chance of personal data being transferred without approval, which is a matter of concern.
To overcome this challenge, IoT has created networks, systems, and devices wherein privacy is well maintained. Maintaining safety and security is a big dare for IoT. However, it still handles it without any disruption.

9. Network

With the increasing number of IoT devices in a network, it becomes difficult to maintain communication for proper functioning. However, cloud service and gateway are a few methods that can solve such problems.

Often, one device can use the connectivity of another device to establish network connectivity even if the second device is not connected to a network. Because IoT devices can communicate with one another, it is more effective and adaptable than other current technologies.

10. Data

The data gathered from IoT devices are analyzed for future prediction. For example, a calorie meter. It helps to regulate the number of calories each day. We also have fitness data, thermostats, and various devices that monitor our health. Therefore, we can use the data collected through these devices.

Physical design of IoT

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

Things/Devices

Things/Devices are used to build a connection, process data, provide interfaces, provide storage, and provide graphics interfaces in an IoT system. all these generate data in a form that can be analyzed by an analytical system and program to perform operations and used to improve the system. For example temperature sensor that is used to analyze the temperature generates the data from a location and is then determined by algorithms.

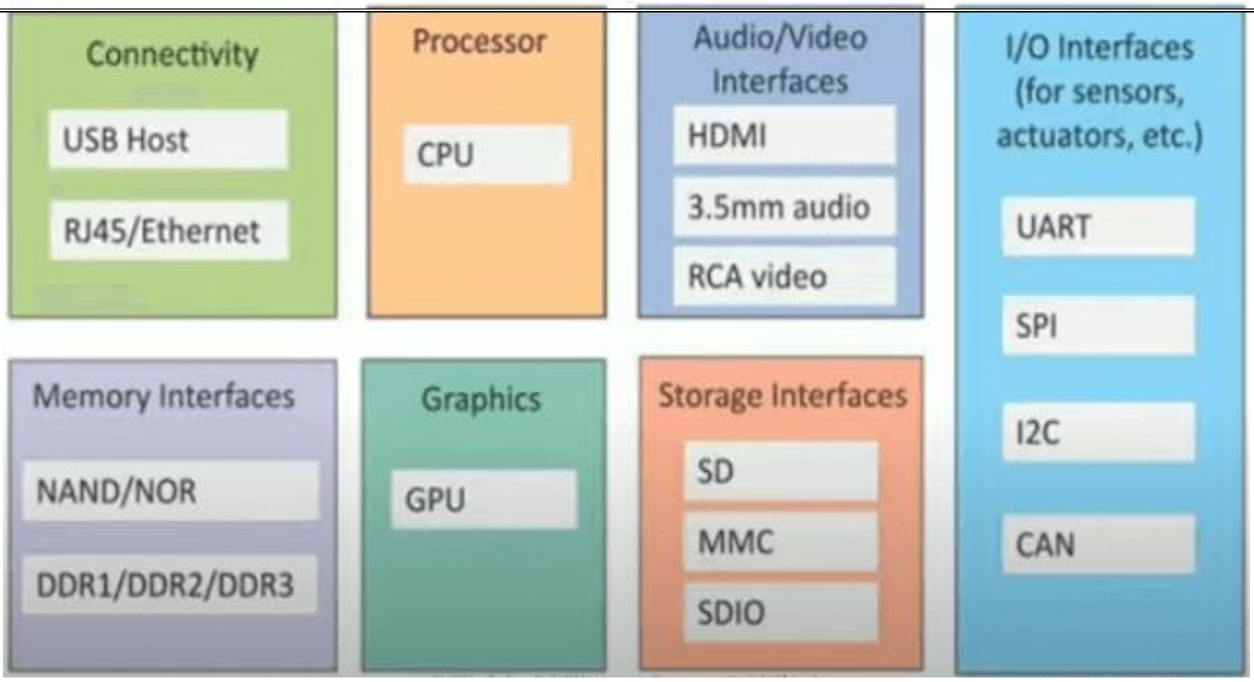
Connectivity
Devices like USB hosts and ETHERNET are used for connectivity between the devices and the server.

Processor
A processor like a CPU and other units are used to process the data. These data are further used to improve the decision quality of an IoT system.

Audio/Video Interfaces
An interface like HDMI and RCA devices is used to record audio and videos in a system.

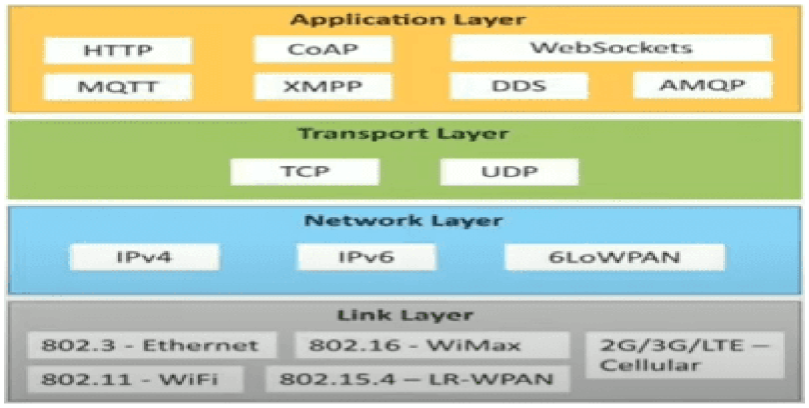
Input/output interface
To give input and output signals to sensors, and actuators we use things like UART, SPI, CAN, etc.

Storage Interfaces
Things like SD, MMC, and SDIO are used to store the data generated from an IoT device.
Other things like DDR and GPU are used to control the activity of an IoT system.



IoT Protocols

These protocols are used to establish communication between a node device and a server over the internet. it helps to send commands to an IoT device and receive data from an IoT device over the internet. We use different types of protocols that are present on both the server and client-side and these protocols are managed by network layers like application, transport, network, and link layer.



Application Layer protocol

In this layer, protocols define how the data can be sent over the network with the lower layer protocols using the application interface. these protocols include HTTP, WebSocket, XMPP, MQTT, DDS, and AMQP protocols.

HTTP
Hypertext transfer protocol is a protocol that presents in an application layer for transmitting media documents. it is used to communicate between web browsers and servers. it makes a request to a server and then waits till it receives a response and in between the request server does not keep any data between two requests.

Web Socket
This protocol enables two-way communication between a client and a host that can be run on an untrusted code in a controlled environment. this protocol is commonly used by web browsers.

MQTT
It is a machine-to-machine connectivity protocol that was designed as a publish/subscribe messaging transport. and it is used for remote locations where a small code footprint is required.

Transport Layer
This layer is used to control the flow of data segments and handle the error control. Also, these layer protocols provide end-to-end message transfer capability independent of the underlying network.

TCP
The transmission control protocol is a protocol that defines how to establish and maintain a network that can exchange data in a proper manner using the internet protocol.

UDP
a user datagram protocol is a part of an internet protocol called the connectionless protocol. this protocol is not required to establish the connection to transfer data.

Network Layer
This layer is used to send datagram’s from the source network to the destination network. We use IPv4 and IPv6 protocols as host identification that transfers data in packets.

IPv4
This is a protocol address that is a unique and numerical label assigned to each device connected to the network. An IP address performs two main functions host and location addressing. IPv4 is an IP address that is 32-bit long.

IPv6
It is a successor of IPv4 that uses 128 bits for an IP address. It is developed by the IETF task force to deal with long-anticipated problems.

Link Layer
Link-layer protocols are used to send data over the network's physical layer. It also determines how the packets are coded and signaled by the devices.

Ethernet
It is a set of technologies and protocols that are used primarily in LANs. It defines the physical layer and the medium access control for wired ethernet networks.

Wi-Fi
It is a set of LAN protocols and specifies the set of media access control and physical layer protocols for implementing wireless local area networks.

LOGICAL DESGIN OF IOT:

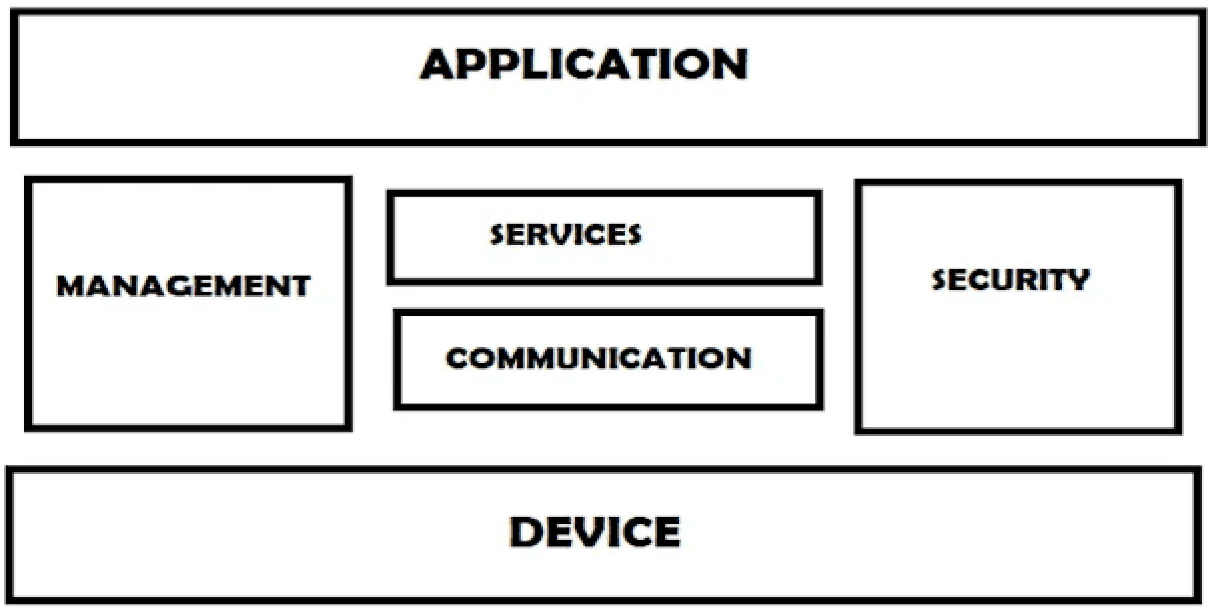
The logical design of an **IoT** system refers to an abstract representation of entities and processes without going into the low-level specifies of implementation. it uses **Functional Blocks**, **Communication Models**, and **Communication APIs** to implement a system.

Logical Design of IoT

- IoT Functional Blocks
- IoT Communication Models
- IoT Communication APIs

IoT Functional blocks

An IoT system consists of a number of functional blocks like Devices, services, communication, security, and application that provide the capability for sensing, actuation, identification, communication, and management.



These functional blocks consist of devices that provide monitoring control functions, handle communication between host and server, manage the transfer of data, secure the system using authentication and other functions, and interface to control and monitor various terms.

Application

It is an interface that provides a control system that use by users to view the status and analyze of system.

Management

This functional block provides various functions that are used to manage an IoT system.

Services

This functional block provides some services like monitoring and controlling a device and publishing and deleting the data and restoring the system.

Communication

This block handles the communication between the client and the cloud-based server and sends/receives the data using protocols.

Security

This block is used to secure an IoT system using some functions like authorization, data security, authentication, 2-step verification, etc.

Device

These devices are used to provide sensing and monitoring control functions that collect data from the outer environment.

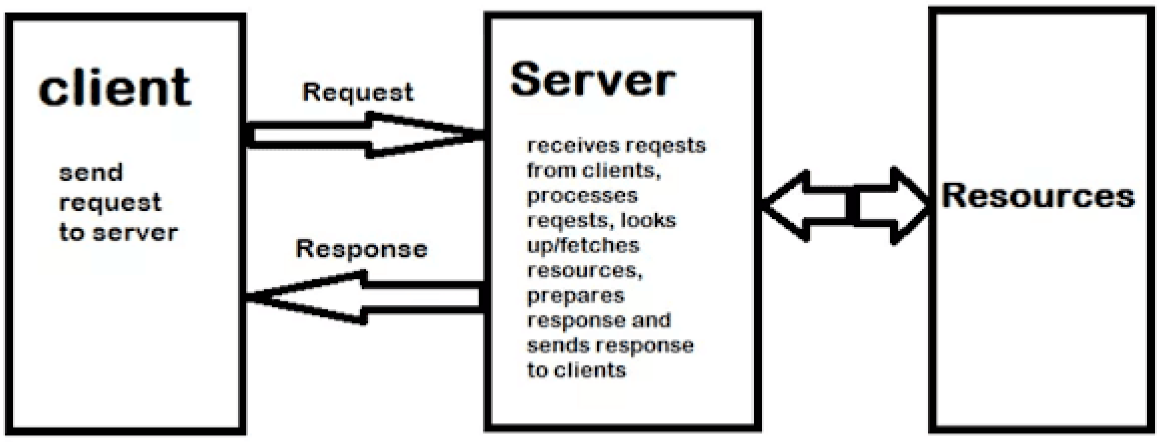
IoT Communication Models

There are several different types of models available in an IoT system that is used to communicate between the system and server like

1. ***Request-response model***
2. ***Publish-subscribe model***
3. ***Push-pull model***
4. ***Exclusive pair model***

1. Request-Response Communication Model

This model is a communication model in which a client sends the request for data to the server and the server responds according to the request. when a server receives a request it fetches the data, retrieves the resources and prepares the response, and then sends the data back to the client.


client
send request to server
Request
Response
Server
receives reqests from clients, processes reqests, looks up/fetches resources, prepares response and sends response to clients
Resources

Request-Response Communication Model

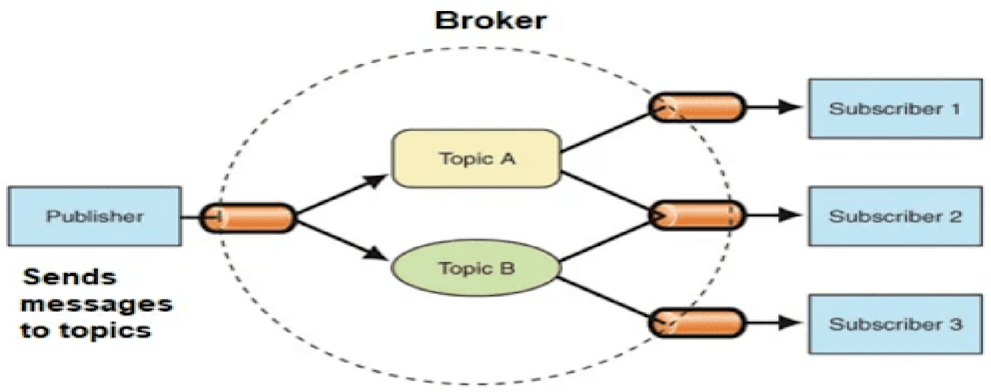
In simple terms, we can say that in the request-response model server send the response of equivalent to the request of the client. in this model, HTTP works as a request-response protocol between a client and server.

Example

When we search a query on a browser then the browser submits an HTTP request to the server and then the server returns a response to the browser(client).

2. Publish-Subscribe Communication Model

In this communication model, we have a broker between publisher and consumer. here publishers are the source of data but they are not aware of consumers. they send the data managed by the brokers and when a consumer subscribes to a topic that is managed by the broker and when the broker receives data from the publisher it sends the data to all the subscribed consumers.

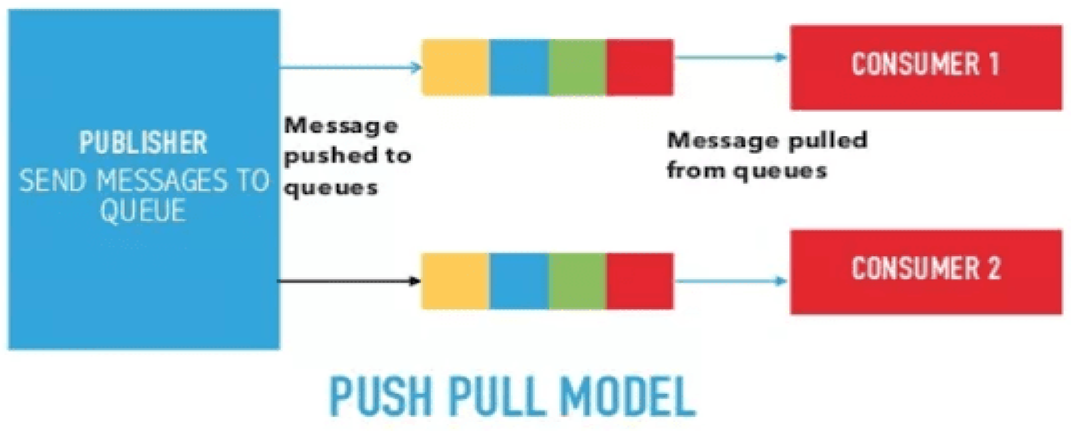


Example

On the website many times we subscribed to their newsletters using our email address. these email addresses are managed by some third-party services and when a new article is published on the website it is directly sent to the broker and then the broker sends these new data or posts to all the subscribers.

3. Push-Pull Communication Model

It is a communication model in which the data push by the producers in a queue and the consumers pull the data from the queues. here also producers are not aware of the consumers.

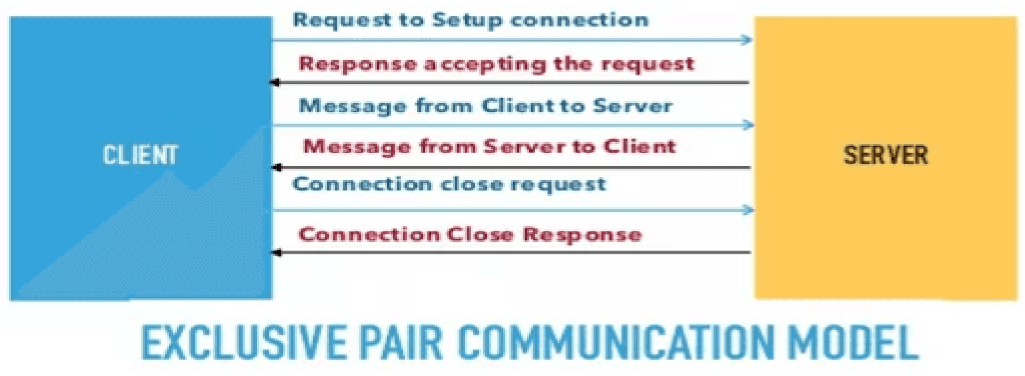


Example

When we visit a website we saw a number of posts that are published in a queue and according to our requirements, we click on a post and start reading it.

4. Exclusive Pair Communication Model

It is a bidirectional fully duplex communication model that uses a persistent connection between the client and server. here first set up a connection between the client and the server and remain open until the client sends a close connection request to the server.


Request to Setup connection
Response accepting the request
Message from Client to Server
Message from Server to Client
Connection close request
Connection Close Response
CLIENT
SERVER
EXCLUSIVE PAIR COMMUNICATION MODEL

IoT communication APIs

These APIs like

1. REST

2. WebSocket are used to communicate between the server and system in IoT.

1. REST-based communication APIs

Representational state transfer (REST) API uses a set of architectural principles that used to design web services. These APIs focus on the systems' resources that how resource states are transferred using the request-response communication model. this API uses some architectural constraints.

Client-server

Here the client is not aware of the storage of data because it is concerned about the server and similarly the server should not be concerned about the user interface because it is a concern of the client. And this separation is needed for independent development and updating of server and client. No matter how the client is using the response of the server and no matter how the server is using the request of the client.

Stateless

It means each request from the client to the server must contain all the necessary information to understand by the server. Because if the server can't understand the request of the client then it can't fetch the request data in a proper manner.

Cacheable

In response, if the cache constraints are given then a client can reuse that response in a later request. it improves the efficiency and scalability of the system without loading the extra data.

A RESTful web APIs is implemented using HTTP and REST principles.

`2. Web Socket based communication API

This type of API allows bi-directional full-duplex communication between server and client using the exclusive pair communication model. this API uses full-duplex communication so it does not require a new connection setup every time when it requests new data. WebSocket API begins with a connection setup between the server and client and if the WebSocket is supported by the server then it responds back to the client with the successful response after the setup of a connection server and the client can send data to each other in full-duplex mode.

This type of API reduces the traffic and latency of data and makes sure that each time when we request new data it cannot terminate the request.

IOT PROTOCOLS:

The backbone of IoT ecosystem is connectivity. IoT devices are able to connect and exchange data with each other via communication protocols.

It’s important for you to know what the top IoT protocols are, and how they work, if you are looking to establish an IoT system in your organization. In this article, we will list the top 10 IoT communication protocols and their characteristics.

What are IoT communication protocols?

IoT communication protocols, also called IoT protocols, are sets of wireless networks and rules that interconnect IoT devices. IoT protocols allow IoT devices to exchange data between each other.

What’s the best IoT communication protocol?

The best IoT communication protocol depends on the specific requirements and constraints of a given system. Factor that play a role in choosing IoT protocols are:

Geographic locations: This is physical distances between the two or more devices that form an ecosystem

Power consumption needs: This is the amount of time for which the IoT devices are stayed on

Physical barriers: These are the barriers that exist between the devices within the IoT ecosystem (e.g. walls, mountains, skyscrapers, etc.)

Overall budget: Different protocols cost differently

Note that the IoT data protocols we list in this article is in no particular order.

Protocol	**Standard**	**Frequencies**	**Approximate Range**	**Data Rates**
WiFi	Based on IEEE 802.11 (common in homes)	2.4 GHz and 5 GHz bands	50 to 100 meters	600 Mbps maximum; 150-200 Mbps the most common
Bluetooth	Bluetooth 4.2 core specification	2.5 GHz	50 to 150 meters	1 Mbps
Zigbee	Zigbee 2.0 based on IEEE802.15.4	2.4 GHz	10 to 100 meters	250 Kbps
MQTT	ISO/IEC 20922	--	--	Up to 256 Mbps in size
Cellular Data	GSM/GPRS/EDGE(2G), UMTS/HSPA(3G), LTE(4G)	900/1800/1900/2100MHz	35km max for GSM; 200km max for HSPA	35-170 Kbps
Z Wave	Z-wave Alliance	Various	30 meters	0.3 to 50 Kbps

Protocol	Standard	Frequencies	Approximate Range	Data Rates
NFC	ISO/IEC 18000-3	13.56 MHz	10cm	100-420 Kbps
LoraWAN	LoRaWAN	Various	2.5km (urban environment), 15km (suburban environment)	0.3 to 50 Kbps
Sigfox	Sigfox	900 Mhz	30-50 km (rural environments), 3-10 km (urban environments)	10-10000 Kbps

Summarized table

1. Wifi

It is one of the IoT communication protocols, best suitable for LAN — a computer network that interconnects computers within a limited area — environments, allowing for fast data transfer. It uses internet protocols (IP) to communicate between endpoint devices.

2. Bluetooth

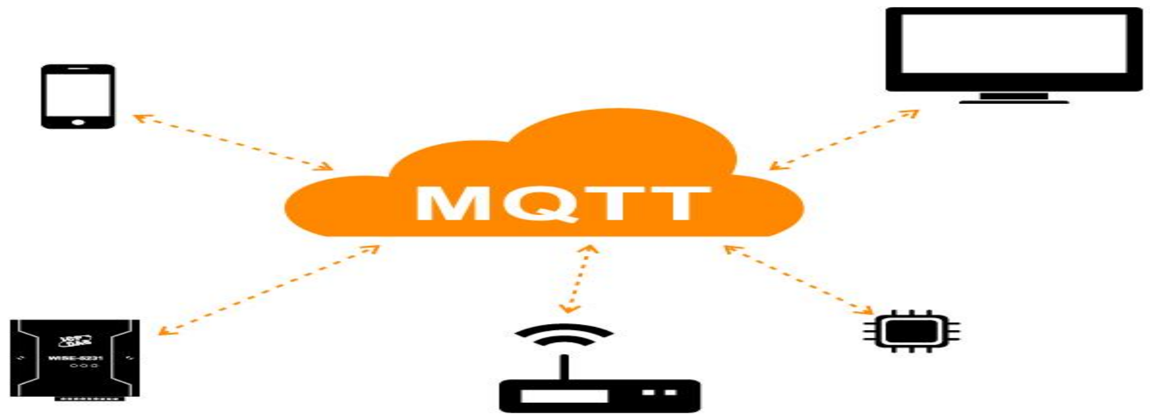
Bluetooth is a protocol used for short-range communication and exchanging of small amounts of data for personal products like smart watches or wireless speakers.

3. Zigbee

Zigbee's advantage comes from low power consumption, wireless control, security, and scalability. Applications such as wireless thermostats and lighting systems are examples of devices using Zigbee for connection.

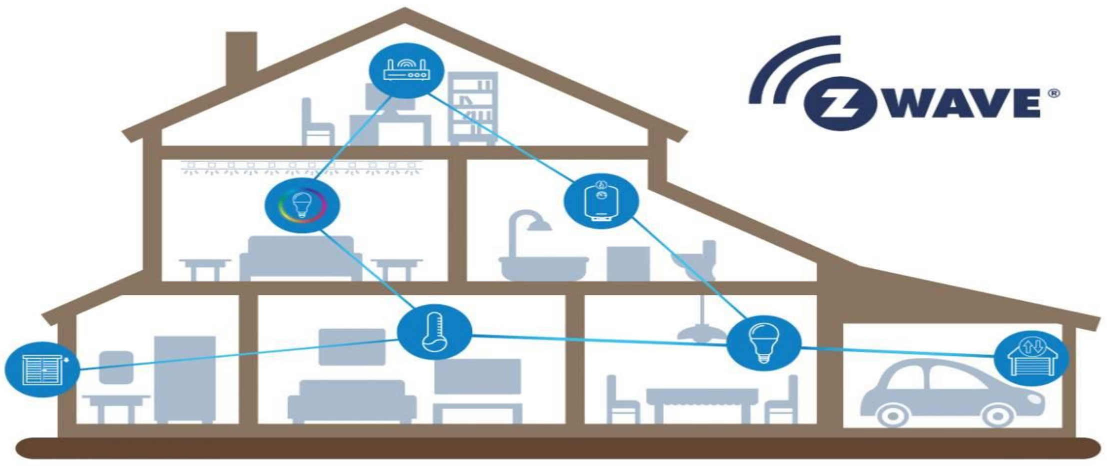
4. MQTT

MQTT handles the transfer of light and simple data from sensors to applications and middleware. It offers a reliable connection and is bandwidth-friendly.



5. Z Wave

Z wave is based on low-power radio frequency (RF) communication technology. It's highly preferable for smart home products such as lamp controllers, door locks, electronic kettles, etc.



6. Cellular data

Cellular networks are capable of handling large flows of data. Its high connectivity range makes it a good choice for connecting objects distanced from one another.

7. NFC (Near Field Communication)

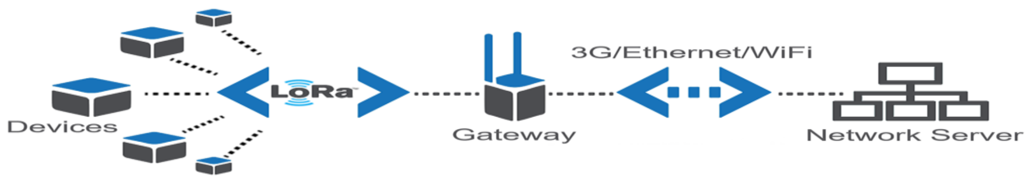
NFC uses electromagnetic communication between the antennas of two devices that are located next to each other. NFC is the technology used in contactless payment at shops today.

8. LoRaWAN

Lora is a long-range, radio-wide network that has low power consumption and is capable of handling large networks consisting of multiple devices.

9. Sigfox

Sigfox is a long-range network for machine-to-machine (M2M) applications, with less power consumption than others. That makes it a good choice for connecting remote devices that have to run on batteries for long periods of time without charging batteries.



What is Embedded System?

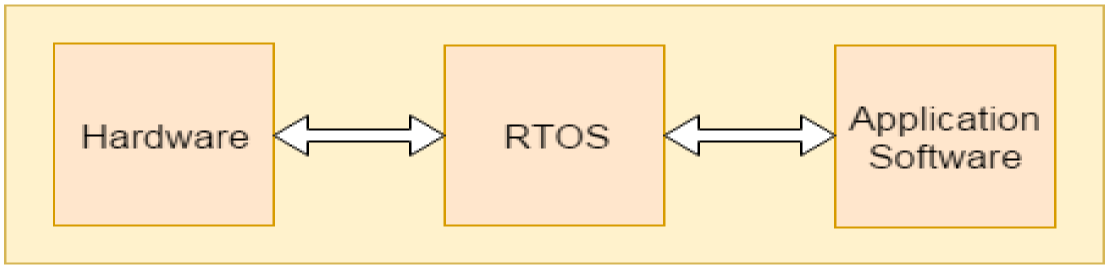
An **Embedded System** is a system that has software embedded into computer-hardware, which makes a system dedicated for a variety of application or specific part of an application or product or part of a larger system.

An embedded system is a combination of computer hardware and software designed for a specific function. Embedded systems may also function within a larger system. The systems can be programmable or have a fixed functionality. Industrial machines, consumer electronics, agricultural and processing industry devices, automobiles, medical equipment, cameras, digital watches, household appliances, airplanes, vending machines and toys, as well as mobile devices, are possible locations for an embedded system.

An embedded system can be a small independent system or a large combinational system. It is a microcontroller-based control system used to perform a specific task of operation.

An embedded system is a combination of three major components:

- **Hardware:** Hardware is physically used component that is physically connected with an embedded system. It comprises of microcontroller based integrated circuit, power supply, LCD display etc.
- **Application software:** Application software allows the user to perform varieties of application to be run on an embedded system by changing the code installed in an embedded system.
- **Real Time Operating system (RTOS):** RTOS supervises the way an embedded system work. It act as an interface between hardware and application software which supervises the application software and provide mechanism to let the processor run on the basis of scheduling for controlling the effect of latencies.



IoT Levels- Deployment Templates

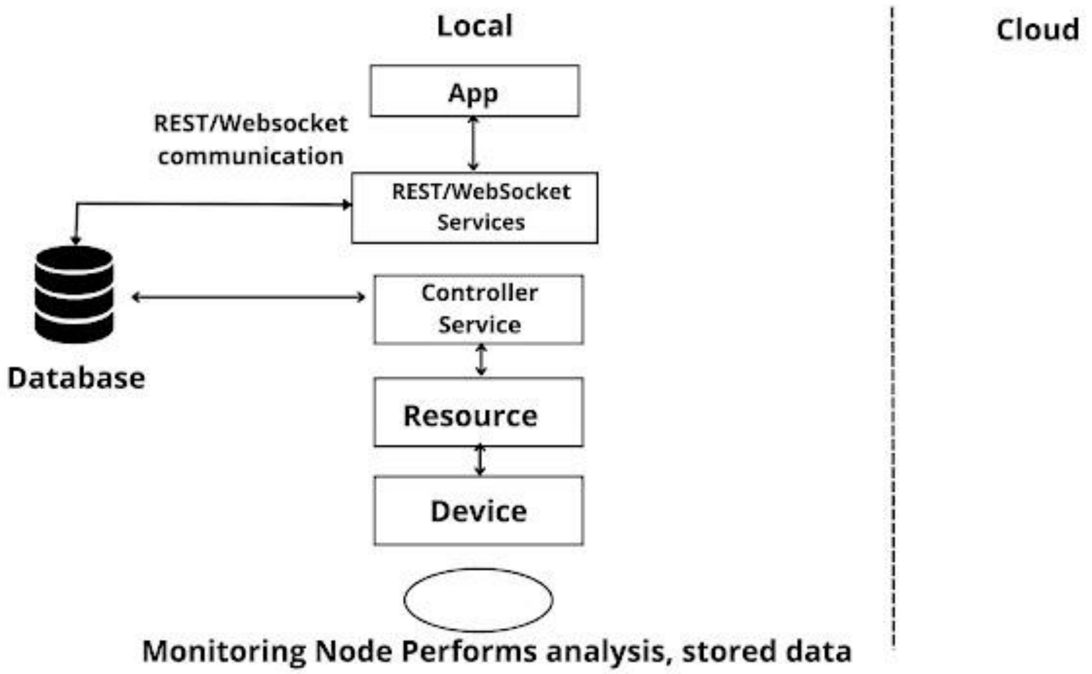
Developing an IoT Level Template system consists of the following components:

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

IoT level 1

IoT systems have a single device that performs sensing or actuation, stores a. analyses it and hosts the application, IoT system-level-l is the best example for modeling low complexity and low-cost solution where the analysis requirement is ok comprehensive and data involved is not big.

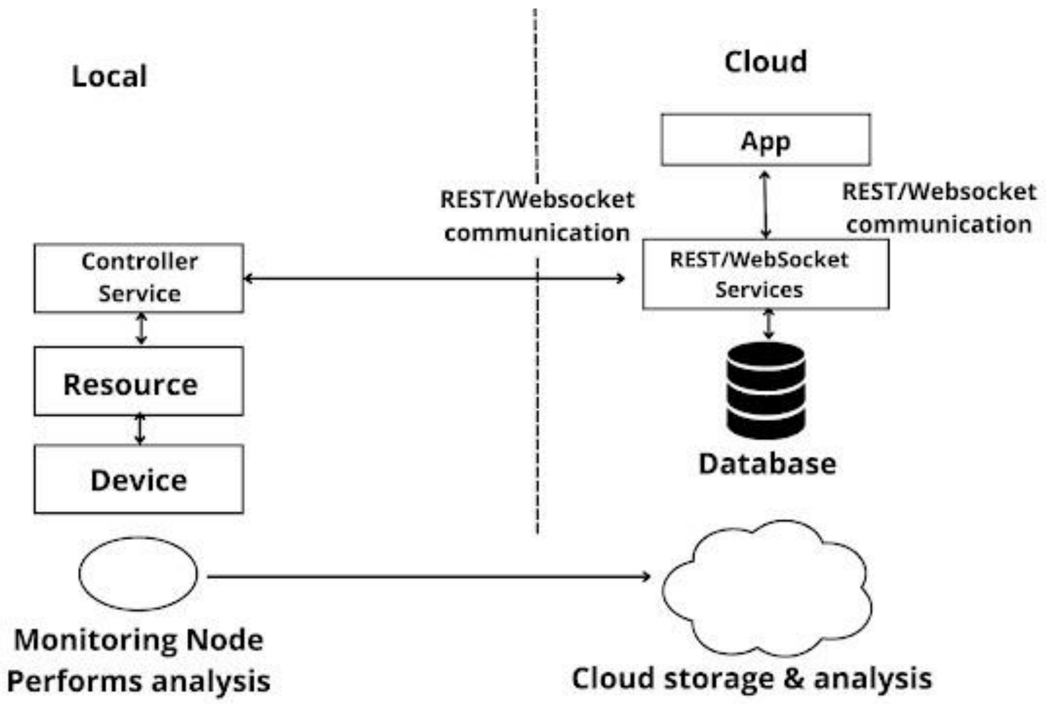
Example: We can understand with the help of an eg. Let’s look at the IoT device that monitors the lights in a house. The lights are controlled through switches. The database has maintained the status of each light and also REST services deployed locally allow retrieving and updating the state of each light and trigger the switches accordingly. For controlling the lights and applications, the application has an interface. The device is connected to the internet and hence the application can be accessed remotely as well



IoT level 2

A node performs sensing/actuation and local analysis. Data is stored in the cloud. this level is facilitated where the data involved is big and primary analysis is not comprehensive

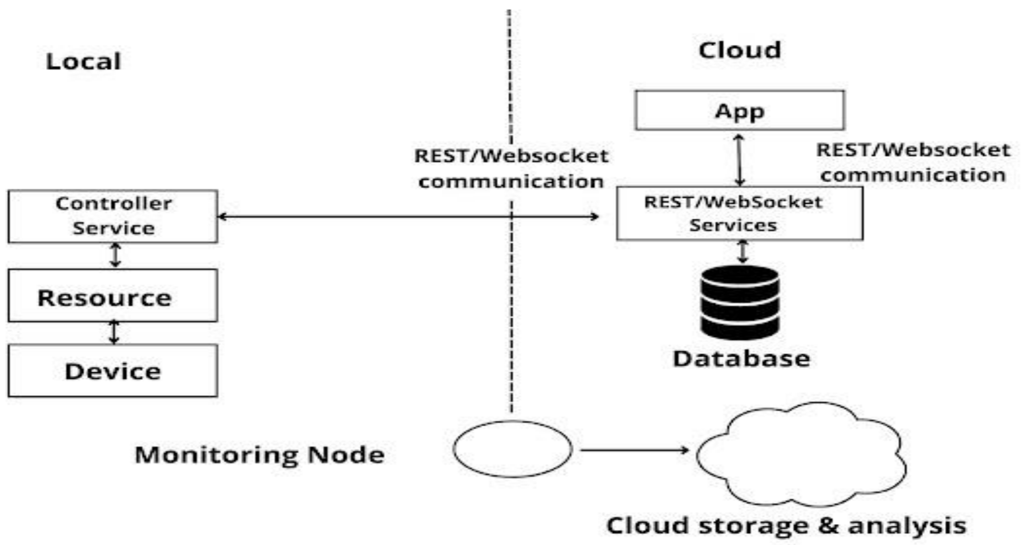
Example: Cloud-based application is used for monitoring and controlling the IoT system A single node monitors the soil moisture in the field Which is sent to the database on the cloud using REST APIS. The controller service continuously monitors moisture levels



IoT level 3

At this level, the application is cloud-based. A single node monitors the environment and stores data in the cloud. This is suitable where data is comprehensive and analysis 1 computationally intensive.

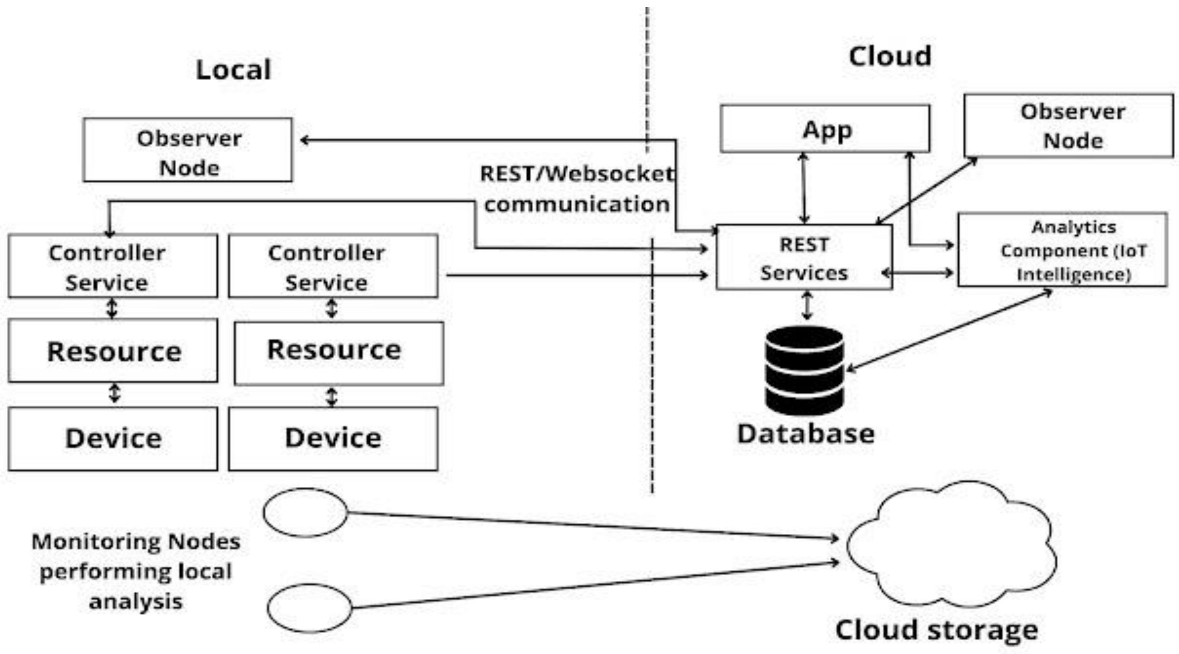
Example: A node is monitoring a package using devices like an accelerometer and gyroscope. These devices track vibration levels. Controller service sends sensor data to the cloud in the rear time using WebSocket APL. Data is stored in the cloud and visualized using a cloud-based application. The analysis component triggers an alert if vibration levels cross a threshold.



IoT level 4

At this level, multiple nodes collect information and store it in the cloud. Local and rent server nodes are used to grant and receive information collected in the cloud from various devices. Observer nodes can process information and use it for applications but not perform control functions, this level is the best solution where data involvement is big, requirement analysis is comprehensive and multiple nodes are required,

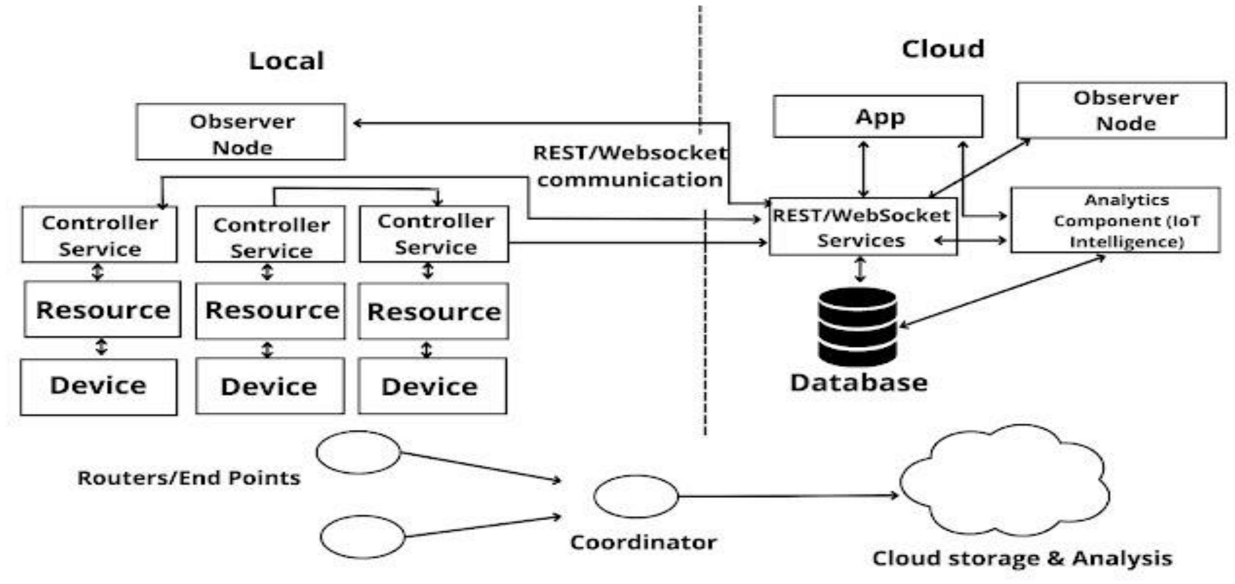
Example: Analysis is done on the cloud and the entire IoT system has monitored the cloud using an application. Noise monitoring of an area requires various nodes to function independently of each other. Each has its own controller service. Data is stored in a cloud database.



IoT level 5

In this level Nodes present locally are of two types end odes and coordinator nodes End nodes collect data and perform sensing or actuation or both. Coordinator nodes collect data from end nodes and send it to the cloud. Data is stored and analyzed in the cloud. This level is best for WSN, where the data involved is big and the requirement analysis is comprehensive.

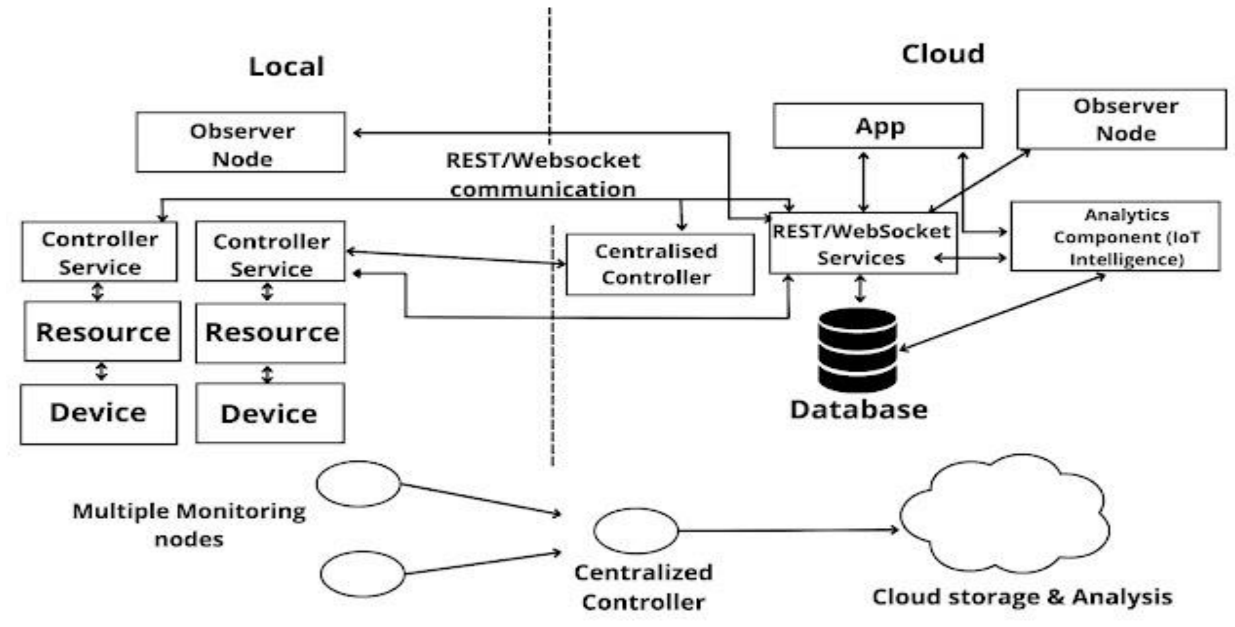
Example: A monitoring system has various components: end nodes collect various data from the environment and send it to the coordinator node. The coordinator node acts as a gateway and allows the data to be transferred to cloud storage using REST API. The controller service on the coordinator node sends data to the cloud.



IoT Level-6

At this level, the application is also cloud-based and data is stored in the cloud-like of levels. Multiple independent end nodes perform sensing and actuation and send d to the cloud. The analytics components analyze the data and store the results in the cloud database. The results are visualized with a cloud-based application. The centralized controller is aware of the status of all the end nodes and sends control commands to the nodes.

Example: Weather monitoring consists of sensors that monitor different aspects of the system. The end nodes send data to cloud storage. Analysis of components, applications, and storage areas in the cloud. The centralized controller controls all nodes and provides inputs.



UNIT II: Prototyping IoT Objects using Microprocessor/Microcontroller **Lecture 9Hrs**

Working principles of sensors and actuators, setting up the board – Programming for IoT, Reading from Sensors, Communication: communication through Bluetooth, WI-Fi.

Working principles of sensors and actuators:

What Is A Sensor?

A sensor is an electronic instrument that translates real-world parameters into electrical signals. It can take the shape of a basic switch or be a more complicated sensor depending on the application. Sensors are integrated in a way that they can interact with the surrounding environment to sense the input energy. A transduction element is used to convert this sensed energy into a more consistent form.
To understand their function, let's say you need to adjust the altitude of an aircraft, and for this purpose, you need to develop a control system for it. Just fixing the fuel throttle won't get the job done – you need to adjust it for each touch point when there's a variance in speed, such as when the airplane goes down or up. Using a sensor can help you measure the speed and convert it into a readable form for the control system. Based on the measured speed, the connected electronic device will adjust the throttle.
Sensors exist in various forms and shapes. Here are some of the common ones:

- **Biosensors** – are typically used in electronically technology. Medical devices use them to analyze proteins, cells, and more.
- **IR sensor** – helps measure distance by estimating the depth of IR reflection.
- **Temperature sensor** – converts temperature changes into electrical signals with the help of the thermodynamic principle.
- **Image sensor** – leverages CMOS (Complementary Metal Oxide Sensor) technology to identify and transfer the details used to create images. You'll find these sensors in surveillance and consumer electronics devices.
- **Motion detectors** – are based on radar, microwave, and ultrasonic technologies. They generate waves and use echo to determine nearby motion and obstacles in their pathways.

What Is an Actuator?

An actuator is a device that takes an electrical signal (control input) and blends it with an energy source to create movement. Typically, actuators receive control input in the form of electric current or voltage, but they can also accommodate hydraulic fluid, hydraulic or pneumatic pressure. The device then uses the control signal to generate a change in the physical system via generating motion, heat, or force. Popular examples of actuators include hydraulic cylinders, comb drives, and electric motors. A stepper motor where an electrical pulse drives a motor is also a common example of an actuator.
Of course, electric actuators receive their power from an electronics circuit. Some examples of this actuator type are servo motors, stepper motors, and AC motors. Then there are hybrid

actuators that have electric controls integrated in addition to the current elements. You'll find these actuators in advanced applications like robots and IoT devices

Key Differences between Actuators and Sensors

Here are the main differences between the two components:

- Sensors identify events or alternations in the environment and transmit this information to other devices, while actuators are machine components that control the moving mechanism.
- Sensors are placed at the input port, while actuators are fixed at the output port.
- Sensors measure both continuous process and discrete variables. In contrast, actuators help impel the parameters of these two processes.
- A sensor will convert any physical attribute to a control signal, while an actuator does the opposite, changing the control signal to physical action.
- Actuators' industrial applications include valves, couplings, and operating dampers. On the other hand, sensors are used to analyze asset vibration, fluid level, or pressure.
- Some non-industrial devices that utilize sensors include cameras, microphones, and home security systems. Other devices like motor controllers, LED, and lasers use actuators.
- Actuators and sensors often depend on each other to perform certain tasks. Where both are present, the actuator relies on the sensor to power its function. If one or the other fails to work correctly, the system will malfunction. In most instances, either the actuator triggers the action, and the sensor transmits the signal, or actuator movements help the sensor send out alerts.

Another Material on Sensors and Actuators:

What Are Sensors?

Sensors or transducers represent physical devices that convert one form of energy into another. Sensors convert a physical device into an electrical impulse to take the desired action. For instance, sensors in an ambient light system will measure the brightness of the light by turning it into an electrical signal.

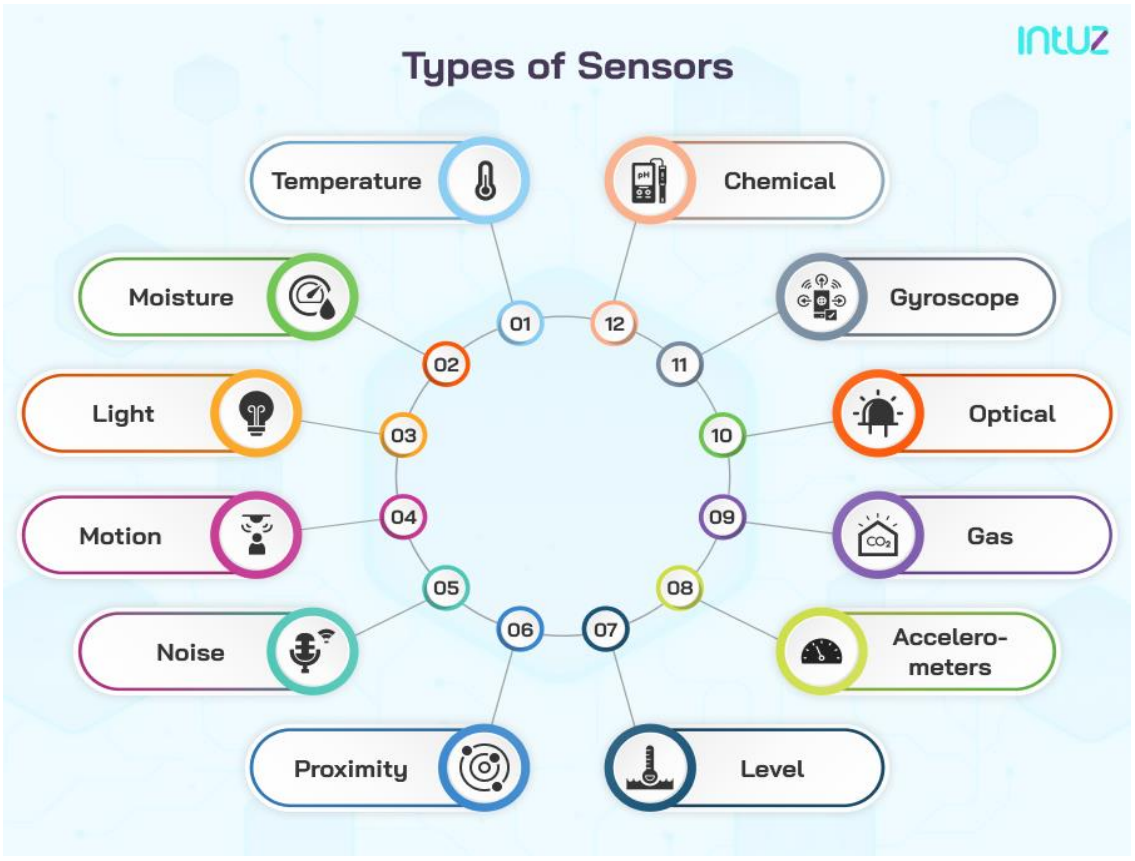
These sensors have a wide array of applications in the IoT network. As they obtain the parameters of a physical object, the output of their observation is converted into resistance, capacitance, impedance, etc.

What Are Actuators?

Taking the sensor discussion forward, actuators do the opposite of a sensor. They convert electrical impulses into physical actions or objects. In the light example, as the sensor is reading the brightness of the light by converting it into an electrical signal, an actuator takes action according to the desired setting. So here, it will decrease or increase the light brightness according to the set parameters.

We can leverage actuators to control and manage our devices in the IoT network according to the information sent by the sensors.

Types of Sensors



1. Temperature: Beginning with the most common type of sensor, the temperate sensor records the amount of heat in a given setting. It can be a machine, a room, a car, a lab, etc. This information can be used to take the desired action, like changing the temperature to optimal settings. The same can be automated according to some specific environmental conditions and settings.

2. Moisture: Where temperature sensors record the heat, moisture sensors record the amount of humidity. They have a wide array of applications in the environment, food supply chains, medicinal labs, agriculture, etc. Moisture sensors either have a hair tension moisture element or a psychrometer to record the moisture content.

3. Light: Light sensors record and assess the ambient light settings in a defined area and recommend actions to change the same. In your smart phone, when the brightness is adjusted according to the exposure to light, the light sensor and the electrical actuator play their part. In the modern homes that have automated light settings, these sensors are used.

4. Motion: Motion sensors are usually installed in security systems and help detect unauthorized activity. Upon sensing activity either by changes in the heat or weight, the sensor activates an alarm system sending notifications to the right people. Motion IoT sensors use radar, infrared, or ultrasonic waves to detect activity in their vicinity.

5. Noise: Noise sensors, as the name suggests, record the noise levels in the given environment. It can be an entire city, a room, a car, etc. In IoT, these sensors are used to build safe working and living environments for people. They are also used to send warning notifications to the right people when noise levels go beyond the stipulated threshold limit.

6. Proximity: Motion sensors and proximity sensors can be kept in the same basket, as the majority of their functions are similar. These sensors record activity nearby with the help of electromagnetic waves, including infrared. They are used in cars, parking lots, retail stores, stadiums, airports, and in several other places to notify the people about their proximity to different components.

7. Level: From granular materials to semi-solid liquids, level sensors detect the quantity or level of different substances. Manufacturing industries, particularly beverage, water treatment, and waste management organizations, have the best use of level sensors.

8. Accelerometers: Accelerometers are an impressive type of IoT sensor used to record and measure an object's acceleration. These types of sensors record the rate of change of an object's speed in relation to time. Plus, they have the added advantage of recording changes in gravity. They can be popularly used in driving fleets and smart pedometers or to detect movement in a stationary object, helping to identify theft.

9. Gas: Gas sensors are used to detect changes in air quality. These sensors are built to detect the presence of toxic, combustible, and other hazardous gasses in a given area. Most of the time, we see the installation of this type of sensor in mining, oil, gas, and energy organizations. However, they are also installed in smart homes and buildings to detect levels of CO2, carbon monoxide, particulate matter, etc.

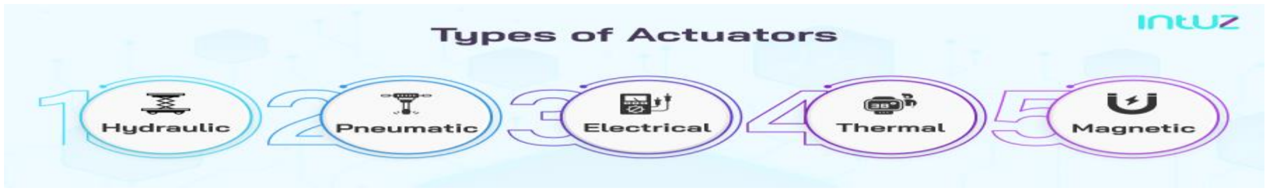
10. Optical: Optical sensors have several use cases but have become an important part of driverless cars. These sensors are used to detect signals and signs to provide information about the surrounding environment. In a driverless car, these sensors are used to detect objects and signs on the road, send the signals to the central control unit and dictate a change in behavior if required.

11. Gyroscope: These sensors are used to measure the velocity of a moving object. Velocity refers to the speed and rotation of an object around its axis. Gyroscope sensors are commonly used in car navigation systems and in stability control systems.

12. Chemical: We can put chemical sensors and gas sensors in the same category. With these sensors, we can expect measurements and detection of several types of chemicals. To build IoT solutions in a factory setting, these sensors can play an important role in ensuring workers' safety and that of the environment.

Sensors measure and record the elements they are built to do based on their sensitivity, drift, linearity, resolution, range, precision, and accuracy. The IoT sensors built with these characteristics can provide accurate information.

Types of Actuators



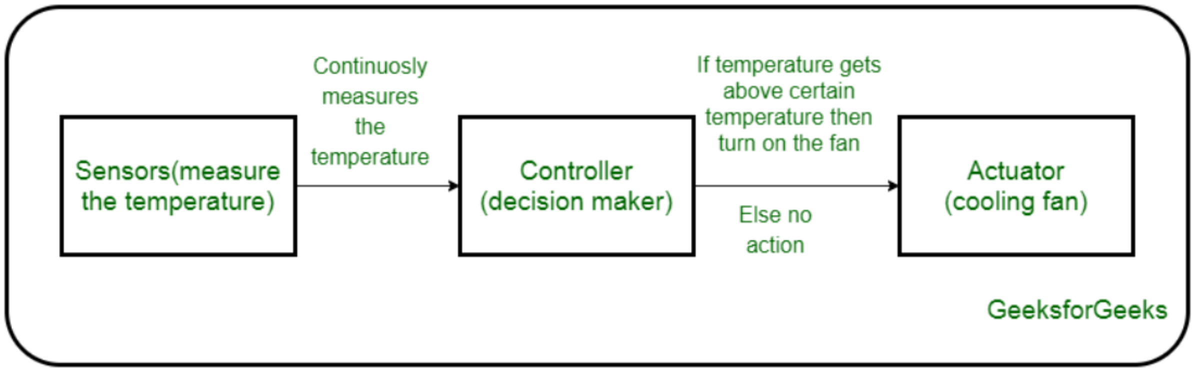
1. Hydraulic: These actuators harness hydraulic power to perform mechanical functions and operations. Generally, these types of actuators are powered by a cylinder or a fluid motor. According to the requirements and recommendations, the mechanical motion is converted into oscillatory, linear, or rotary.

2. Pneumatic: Pneumatic actuators create two types of motions, rotary or linear. They are powered by a vacuum or compressed air at high pressure to implement the required type of motion. Compared to other types of actuators, pneumatic actuators are low-cost and low-maintenance actuators.

3. Electrical: In these actuators, a motor converts electrical energy into mechanical motion. These actuators are powered by electricity and provide precision control. These actuators are heavily used in industrial settings to automate mechanical operations.

4. Thermal: The thermal actuators have thermal-sensitive material fitted inside, which is used to produce linear motion. The word thermal implies that these actuators are used in response to temperature changes. The most popular use case includes shutting off valves and operating latches or switches.

5. Magnetic: These types of actuators convert electromagnetic energy into mechanical output and operate in a linear or rotary direction. Magnetic actuators can provide continuous mechanical operation and are popularly used in the automotive and aerospace industries.


Sensors(measure the temperature)
Continuosly measures the temperature
Controller (decision maker)
If temperature gets above certain temperature then turn on the fan
Else no action
Actuator (cooling fan)
GeeksforGeeks

Working of IoT devices and use of Actuators

Setting up the board:

Boards mainly used in IOT applications are:

1. Ardunio

2. Rasberry Pi

1.Ardunio

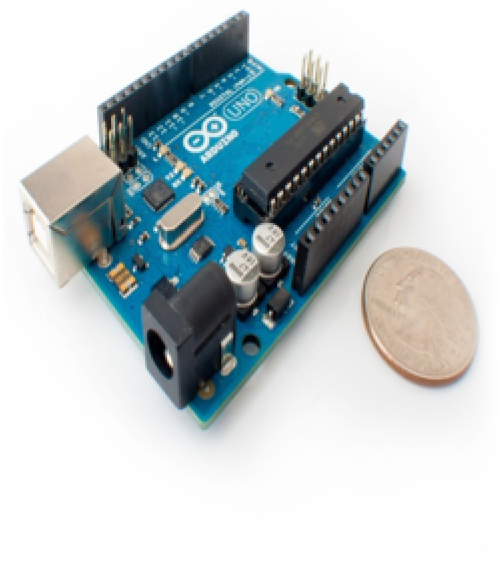
Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

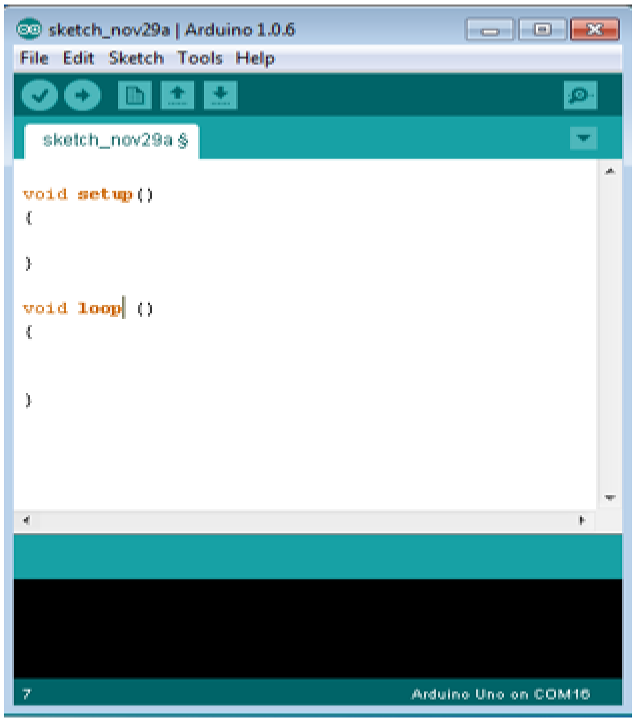
Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

The key features are −

- Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.
- Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.





```
void setup()
{

}

void loop ()
{


}
```

Board Types

Various kinds of Arduino boards are available depending on different microcontrollers used. However, all Arduino boards have one thing in common: they are programed through the Arduino IDE.

The differences are based on the number of inputs and outputs (the number of sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, form factor etc. Some boards are designed to be embedded and have no programming interface (hardware), which you would need to buy separately. Some can run directly from a 3.7V battery, others need at least 5V.

Here is a list of different Arduino boards available.

Arduino boards based on ATMEGA328 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Uno R3	5V	16MHz	14	6	6	1	USB via ATMega16U2
Arduino Uno R3 SMD	5V	16MHz	14	6	6	1	USB via ATMega16U2
Red Board	5V	16MHz	14	6	6	1	USB via FTDI
Arduino Pro 3.3v/8 MHz	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
Arduino Pro 5V/16MHz	5V	16MHz	14	6	6	1	FTDI-Compatible Header
Arduino mini 05	5V	16MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 3.3v/8mhz	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 5v/16mhz	5V	16MHz	14	8	6	1	FTDI-Compatible Header
Arduino Ethernet	5V	16MHz	14	6	6	1	FTDI-Compatible Header

Arduino Fio	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
LilyPad Arduino 328 main board	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
LilyPad Arduino simple board	3.3V	8MHz	9	4	5	0	FTDI-Compatible Header

Arduino boards based on ATMEGA32u4 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Leonardo	5V	16MHz	20	12	7	1	Native USB
Pro micro 5V/16MHz	5V	16MHz	14	6	6	1	Native USB
Pro micro 3.3V/8MHz	5V	16MHz	14	6	6	1	Native USB
LilyPad Arduino USB	3.3V	8MHz	14	6	6	1	Native USB

Arduino boards based on ATMEGA2560 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	5V	16MHz	54	16	14	4	USB via ATMega16U2B
Mega Pro 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro 5V	5V	16MHz	54	16	14	4	FTDI-Compatible Header

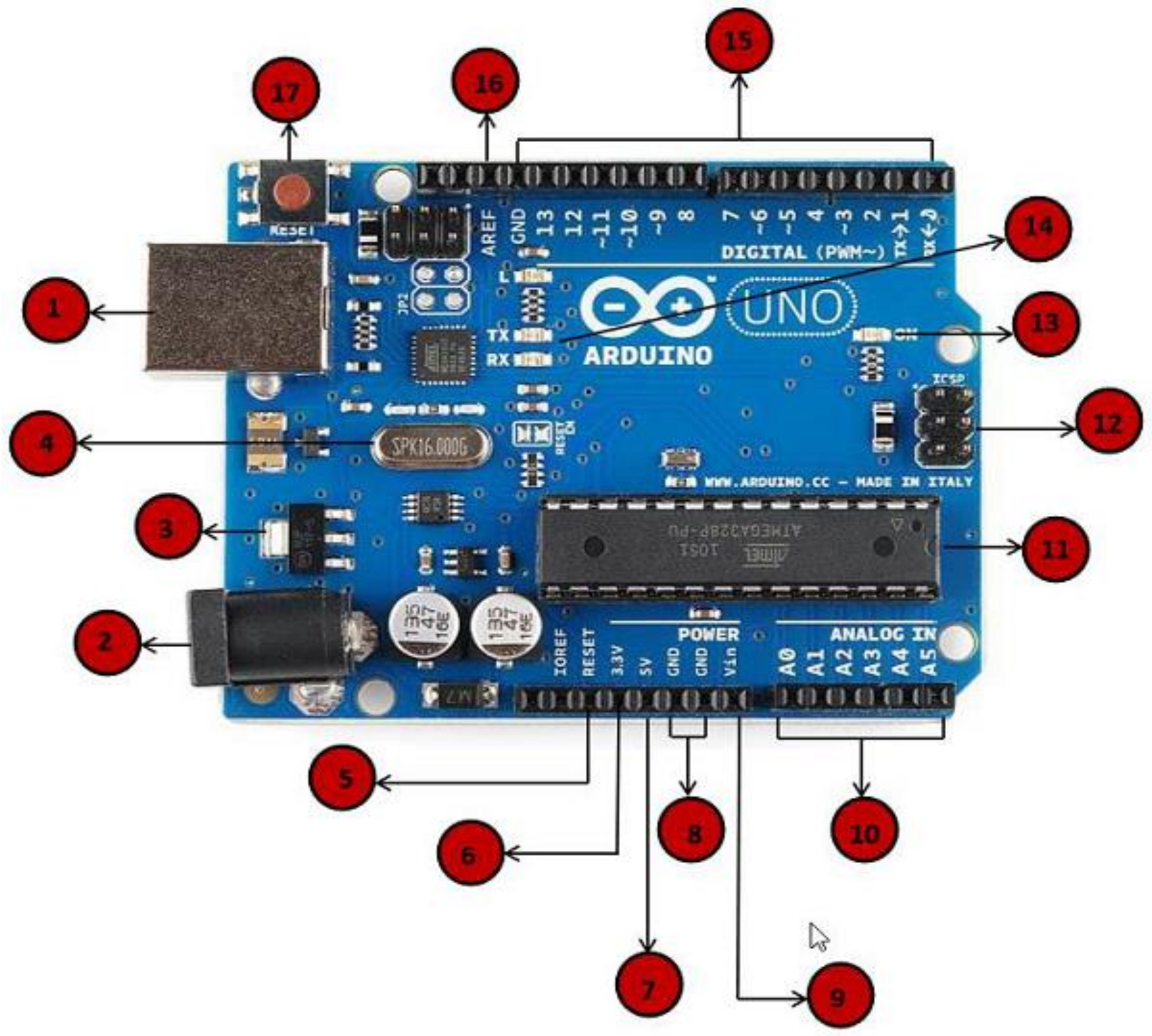
Mega Pro Mini 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header

Arduino boards based on AT91SAM3X8E microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	3.3V	84MHz	54	12	12	4	USB native

Arduino - Board Description

In this chapter, we will learn about the different components on the Arduino board. We will study the Arduino UNO board because it is the most popular board in the Arduino board family. In addition, it is the best board to get started with electronics and coding. Some boards look a bit different from the one given below, but most Arduinos have majority of these components in common.



1	**Power USB** Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).
2	**Power (Barrel Jack)** Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).
3	**Voltage Regulator** The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.
4	**Crystal Oscillator** The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.
5,17	**Arduino Reset** You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).
6,7 8,9	**Pins (3.3, 5, GND, Vin)** • 3.3V (6) − Supply 3.3 output volt • 5V (7) − Supply 5 output volt • Most of the components used with Arduino board works fine with 3.3 volt and 5 volt. • GND (8)(Ground) − There are several GND pins on the Arduino, any of which can be used to ground your circuit. • Vin (9) − This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.
10	**Analog pins** The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.
11	**Main microcontroller** Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before

	loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.
12	**ICSP pin** Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.
13	**Power LED indicator** This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.
14	**TX and RX LEDs** On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.
15	**Digital I/O** The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled "~" can be used to generate PWM.
16	**AREF** AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Arduino - Installation

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on the Arduino board.

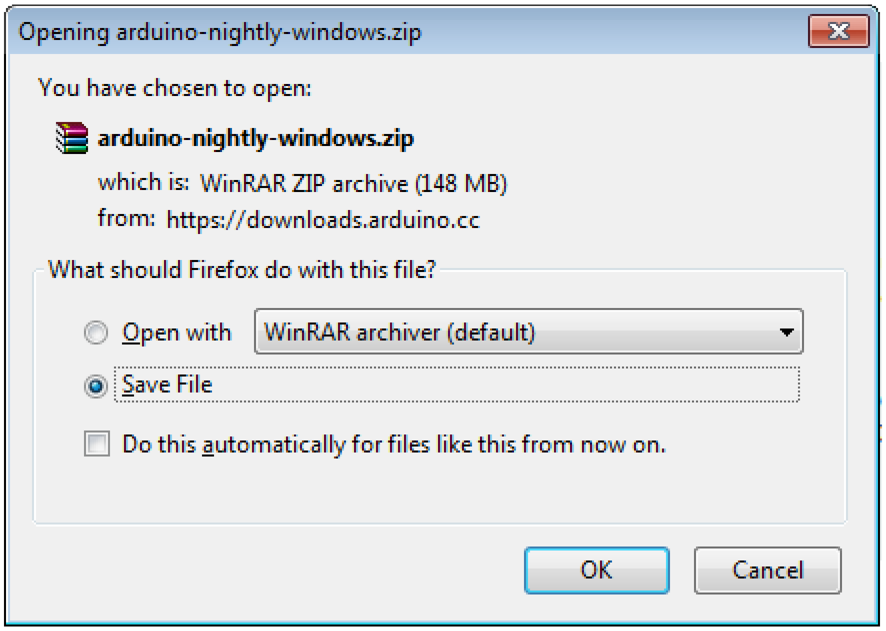
In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

Step 1 − First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.

Step 2 − Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



Opening arduino-nightly-windows.zip

You have chosen to open:

arduino-nightly-windows.zip

which is: WinRAR ZIP archive (148 MB)

from: https://downloads.arduino.cc

What should Firefox do with this file?

- ○ Open with: WinRAR archiver (default)
- ● Save File
- ☐ Do this automatically for files like this from now on.

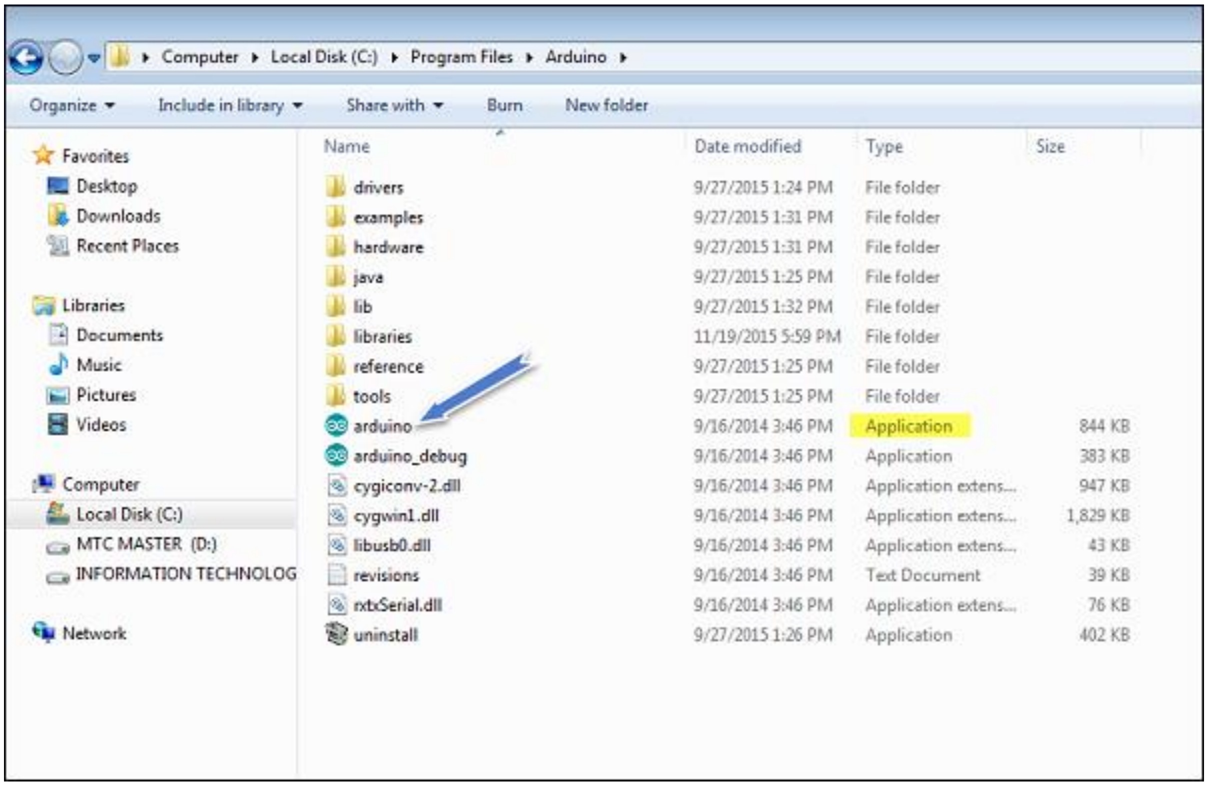
Step 3 − Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 − Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.



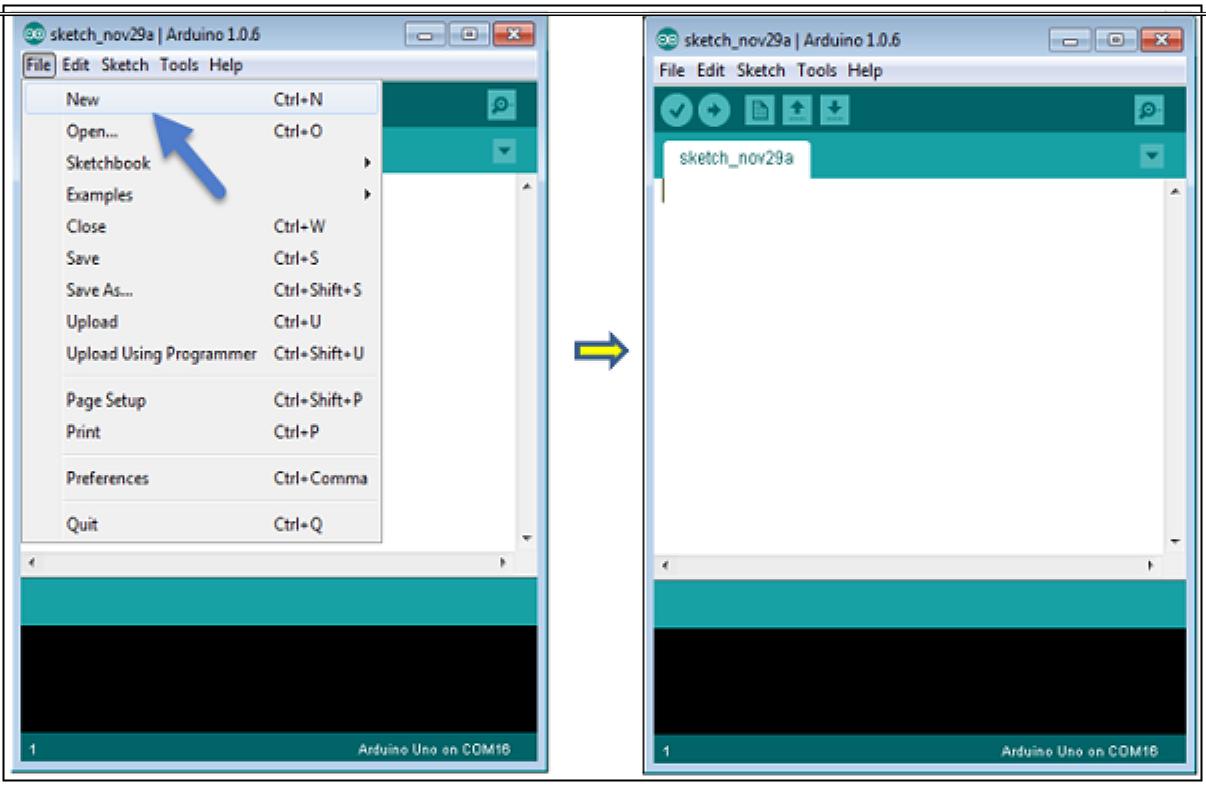
Name	Date modified	Type	Size
drivers	9/27/2015 1:24 PM	File folder	
examples	9/27/2015 1:31 PM	File folder	
hardware	9/27/2015 1:31 PM	File folder	
java	9/27/2015 1:25 PM	File folder	
lib	9/27/2015 1:32 PM	File folder	
libraries	11/19/2015 5:59 PM	File folder	
reference	9/27/2015 1:25 PM	File folder	
tools	9/27/2015 1:25 PM	File folder	
arduino	9/16/2014 3:46 PM	Application	844 KB
arduino_debug	9/16/2014 3:46 PM	Application	383 KB
cygiconv-2.dll	9/16/2014 3:46 PM	Application extens...	947 KB
cygwin1.dll	9/16/2014 3:46 PM	Application extens...	1,829 KB
libusb0.dll	9/16/2014 3:46 PM	Application extens...	43 KB
revisions	9/16/2014 3:46 PM	Text Document	39 KB
rxtxSerial.dll	9/16/2014 3:46 PM	Application extens...	76 KB
uninstall	9/27/2015 1:26 PM	Application	402 KB

Step 5 − Open your first project.

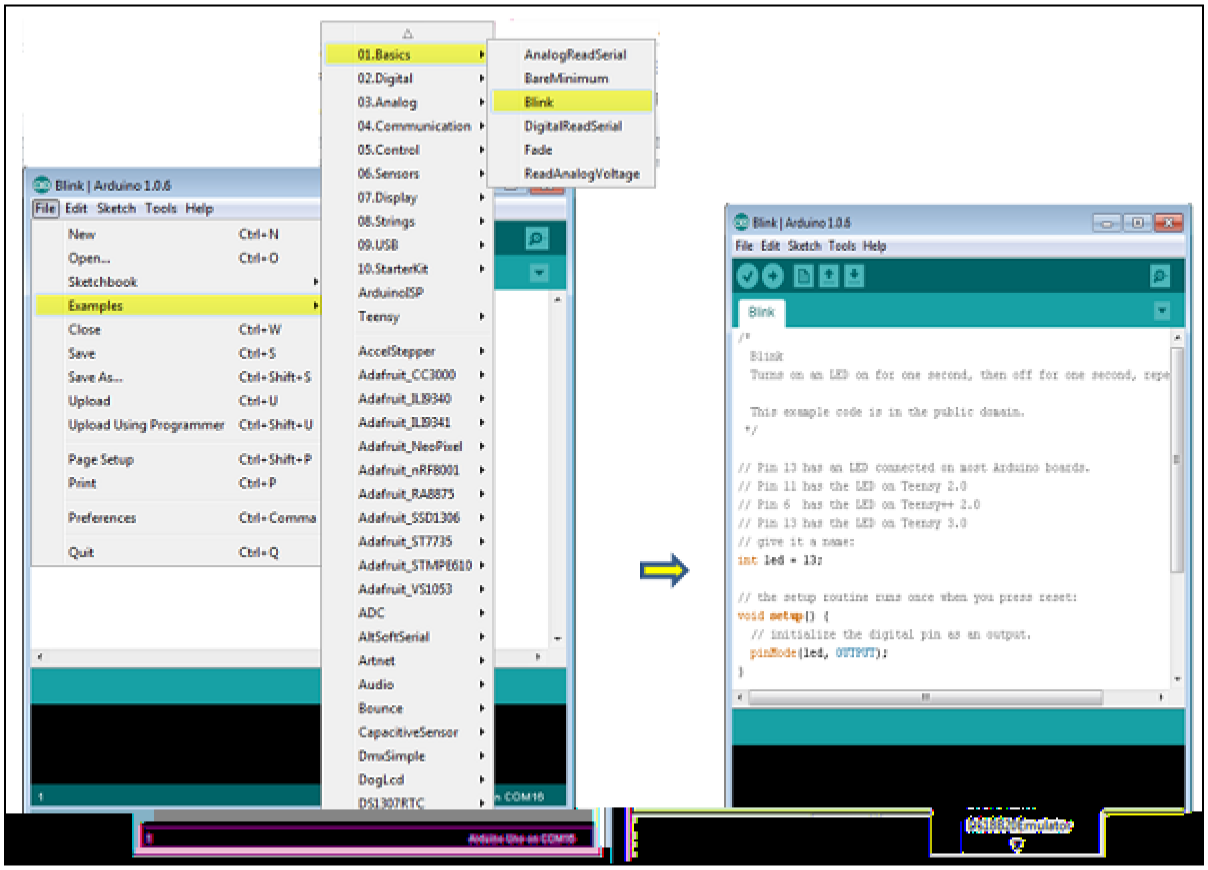
Once the software starts, you have two options −

- Create a new project.
- Open an existing project example.

To create a new project, select File → **New**.



To open an existing project example, select File → Example → Basics → Blink.



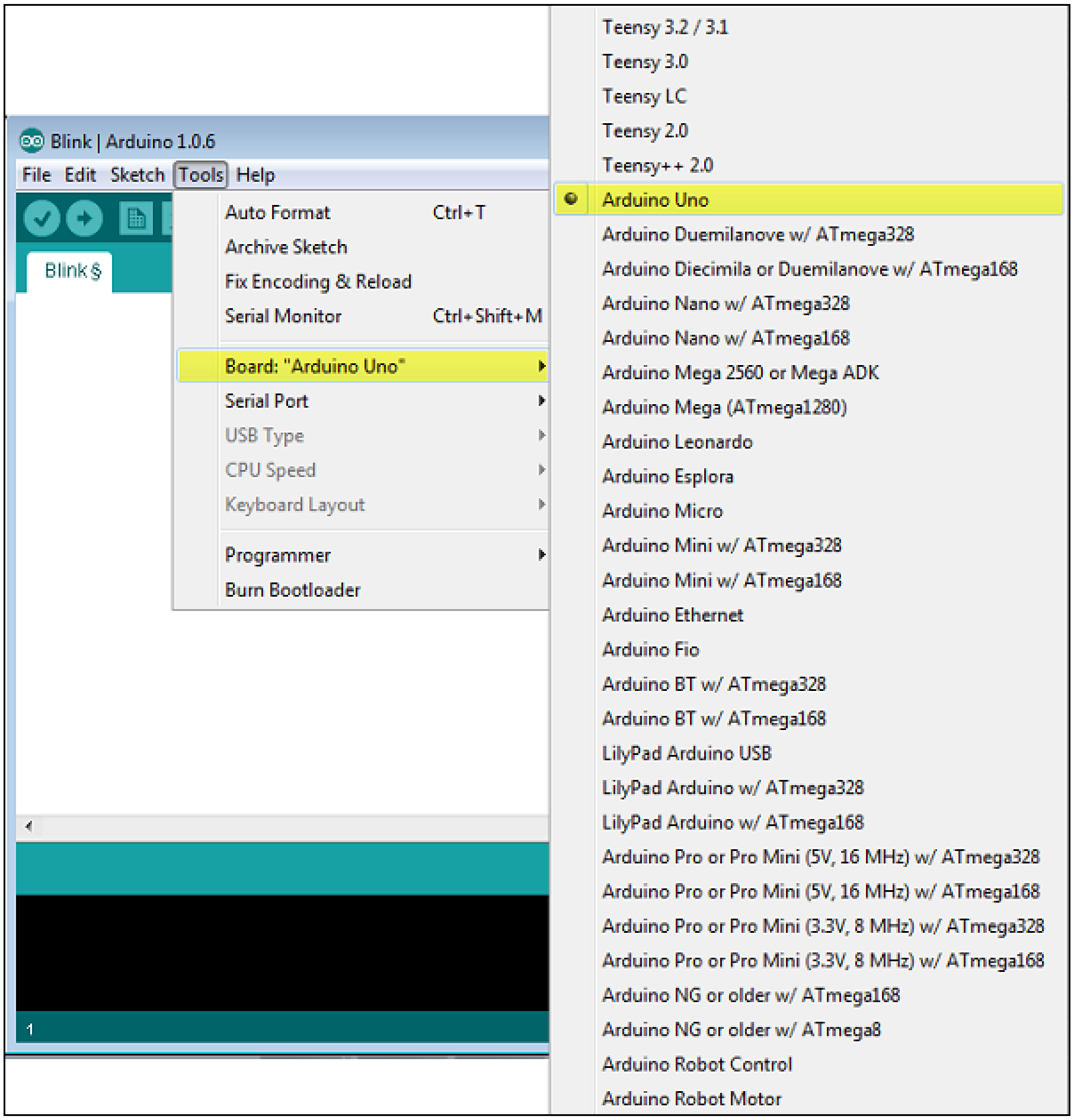
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// Pin 11 has the LED on Teensy 2.0
// Pin 6  has the LED on Teensy++ 2.0
// Pin 13 has the LED on Teensy 3.0
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
```

Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6 − Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

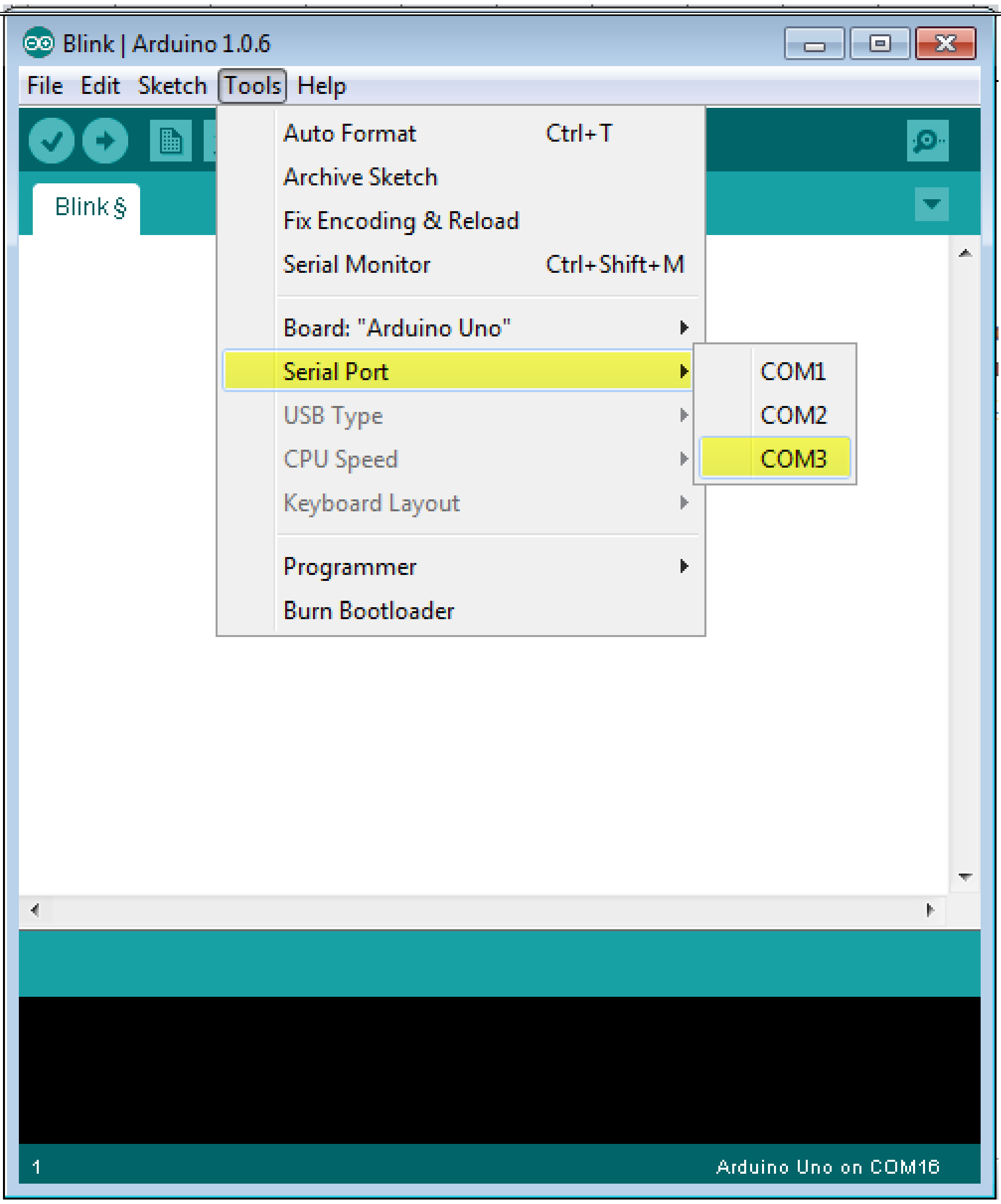
Go to Tools → Board and select your board.


Blink | Arduino 1.0.6
File Edit Sketch Tools Help
Blink §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino Uno"
Serial Port
USB Type
CPU Speed
Keyboard Layout
Programmer
Burn Bootloader
Teensy 3.2 / 3.1
Teensy 3.0
Teensy LC
Teensy 2.0
Teensy++ 2.0
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor
1

Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

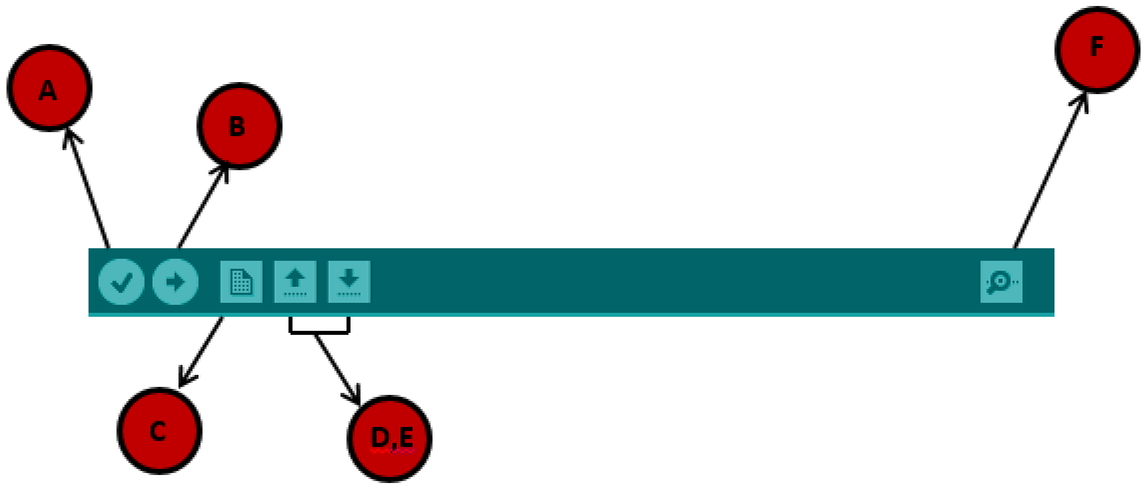
Step 7 − Select your serial port.

Select the serial device of the Arduino board. Go to **Tools** → **Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 − Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



A − Used to check if there is any compilation error.

B − Used to upload a program to the Arduino board.

C − Shortcut used to create a new sketch.

D − Used to directly open one of the example sketch.

E − Used to save your sketch.

F − Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Note − If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.

Arduino - Program Structure

In this chapter, we will study in depth, the Arduino program structure and we will learn more new terminologies used in the Arduino world. The Arduino software is open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL.

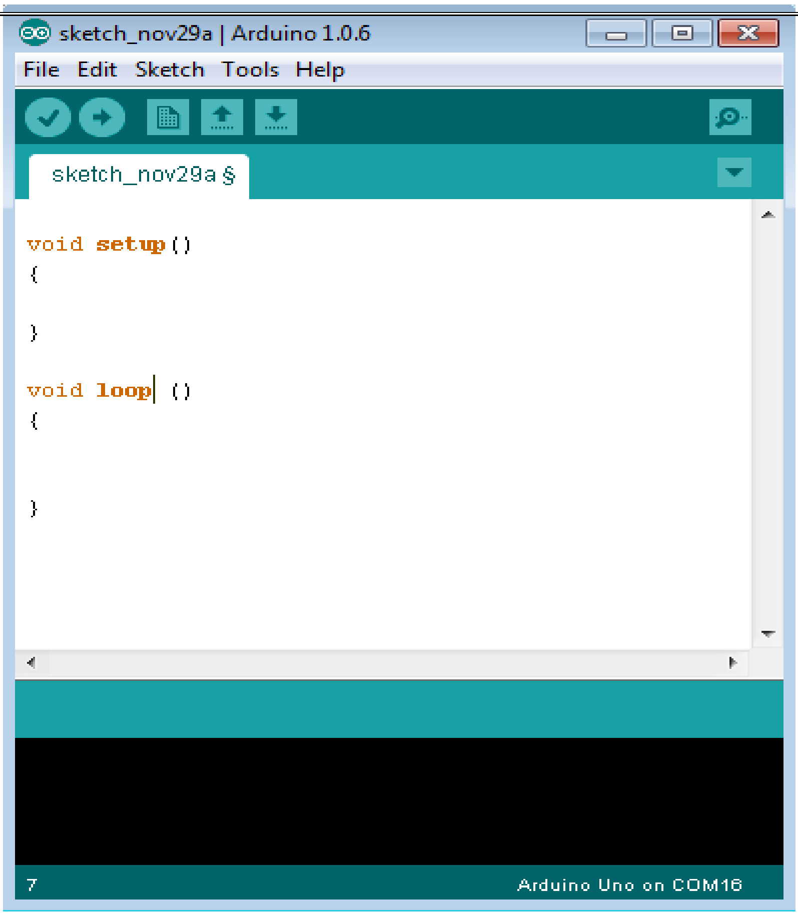
Sketch − The first new terminology is the Arduino program called “**sketch**”.

Structure

Arduino programs can be divided in three main parts: **Structure, Values** (variables and constants), and **Functions**. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the **Structure**. Software structure consist of two main functions −

- Setup( ) function
- Loop( ) function



```
void setup()
{

}

void loop ()
{


}
```

Void setup ( ) {

}

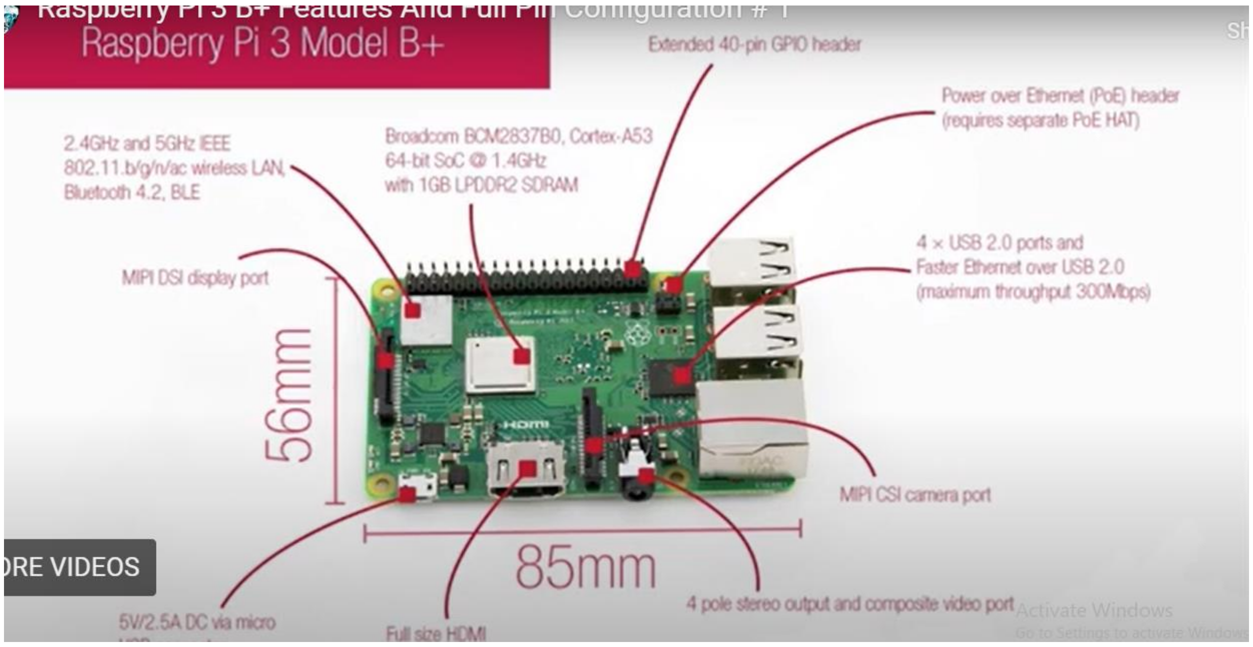
- **PURPOSE** − The **setup()** function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.
- **INPUT** − -
- **OUTPUT** − -
- **RETURN** − -

Void Loop ( ) {

}

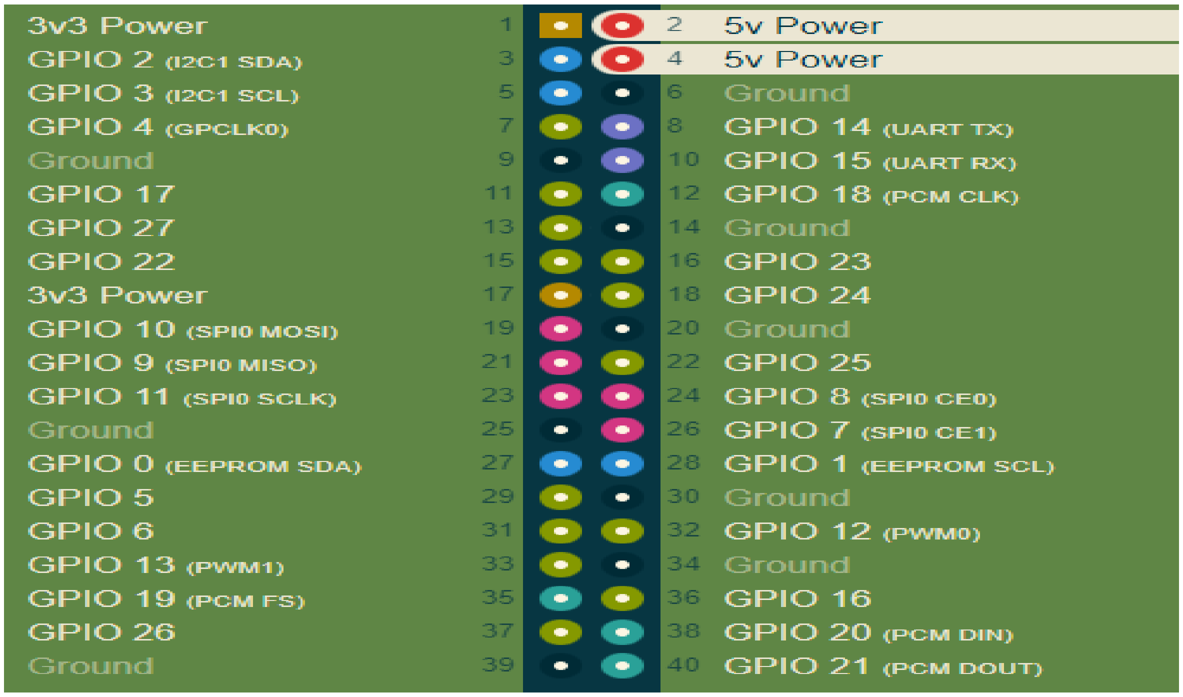
- **PURPOSE** − After creating a **setup()** function, which initializes and sets the initial values, the **loop()** function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.
- **INPUT** − -
- **OUTPUT** − -
- **RETURN** − -

RASBERRY PI SPECIFICATIONS:

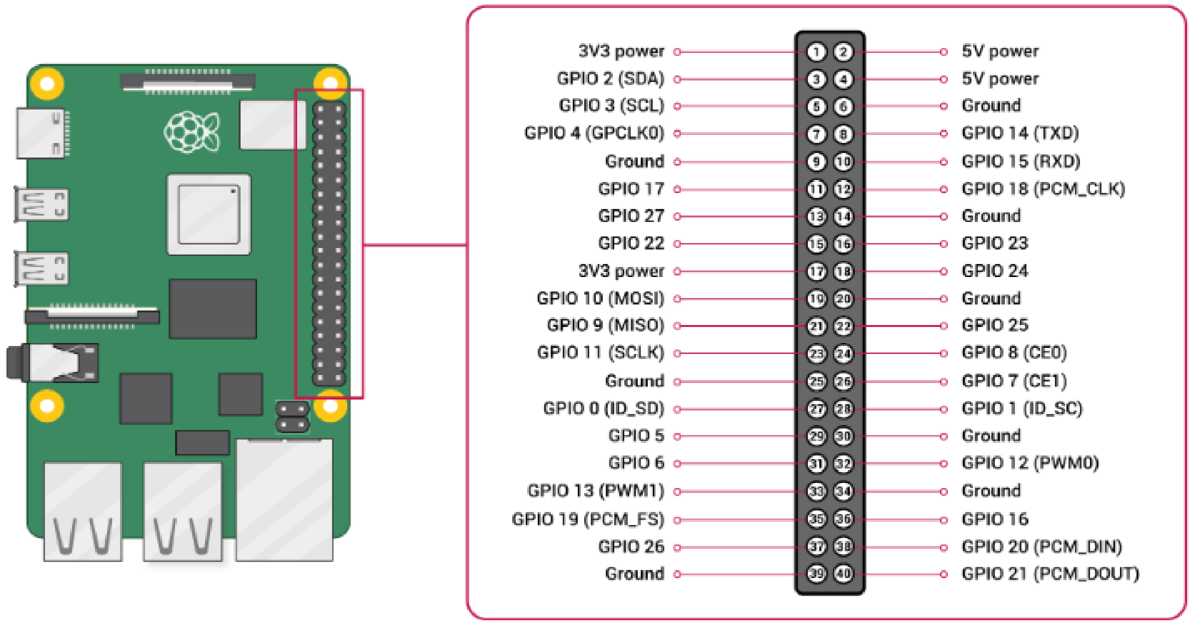

Raspberry Pi 3 B+ Features And Full Pin Configuration # 1
Raspberry Pi 3 Model B+
Extended 40-pin GPIO header
Power over Ethernet (PoE) header (requires separate PoE HAT)
2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz with 1GB LPDDR2 SDRAM
MIPI DSI display port
4 × USB 2.0 ports and Faster Ethernet over USB 2.0 (maximum throughput 300Mbps)
56mm
MIPI CSI camera port
85mm
RE VIDEOS
4 pole stereo output and composite video port
5V/2.5A DC via micro
Full size HDMI

1. 2.4 GHz and 5 GHz processor
2. Builtin Wi-Fi and Bluetooth
3.1/2/4/8 GB SDRAM slots
4 40PIN GPIO Header
5.4*USB 2.0 ports
6. MOI CSI Camera port
7.4 Pole Stereo output and composite video port
8. Fullsize HDMI port
9.5V/2.5A via micro USB connector
10. MPI DSI Display port
11. POE (Power over Ethernet) Header

Link to over document:
https://www.bing.com/videos/search?q=raspberry+pi+pin+configuration&docid=603538691472580276&mid=866C9B1019131387383A866C9B1019131387383A&view=detail&FORM=VIRE



3v3 Power	1	●	●	2	5v Power
GPIO 2 (I2C1 SDA)	3	●	●	4	5v Power
GPIO 3 (I2C1 SCL)	5	●	●	6	Ground
GPIO 4 (GPCLK0)	7	●	●	8	GPIO 14 (UART TX)
Ground	9	●	●	10	GPIO 15 (UART RX)
GPIO 17	11	●	●	12	GPIO 18 (PCM CLK)
GPIO 27	13	●	●	14	Ground
GPIO 22	15	●	●	16	GPIO 23
3v3 Power	17	●	●	18	GPIO 24
GPIO 10 (SPI0 MOSI)	19	●	●	20	Ground
GPIO 9 (SPI0 MISO)	21	●	●	22	GPIO 25
GPIO 11 (SPI0 SCLK)	23	●	●	24	GPIO 8 (SPI0 CE0)
Ground	25	●	●	26	GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA)	27	●	●	28	GPIO 1 (EEPROM SCL)
GPIO 5	29	●	●	30	Ground
GPIO 6	31	●	●	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	●	●	34	Ground
GPIO 19 (PCM FS)	35	●	●	36	GPIO 16
GPIO 26	37	●	●	38	GPIO 20 (PCM DIN)
Ground	39	●	●	40	GPIO 21 (PCM DOUT)



UNIT III IoT Architecture and Protocols

Architecture Reference Model- Introduction, Reference Model and architecture, IoT reference Model, Protocols- 6LowPAN, RPL, CoAP, MQTT, IoT frameworks- Thing Speak.

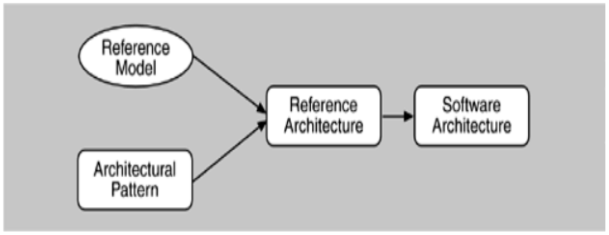
Architecture Reference Model

A reference model is a division of functionality together with data flow between the pieces. A reference model is a standard decomposition of a known problem into parts that cooperatively solve the problem. Arising from experience, reference models are a characteristic of mature domains. Can you name the standard parts of a compiler or a database management system? Can you explain in broad terms how the parts work together to accomplish their collective purpose? If so, it is because you have been taught a reference model of these applications.

An architectural pattern is a description of element and relation types together with a set of constraints on how they may be used. A pattern can be thought of as a set of constraints on an architecture-on the element types and their patterns of interaction-and these constraints define a set or family of architectures that satisfy them. For example, client-server is a common architectural pattern. Client and server are two element types, and their coordination is described in terms of the protocol that the server uses to communicate with each of its clients. Use of the term client-server implies only that multiple clients exist; the clients themselves are not identified, and there is no discussion of what functionality, other than implementation of the protocols, has been assigned to any of the clients or to the server. Countless architectures are of the client-server pattern under this (informal) definition, but they are different from each other. An architectural pattern is not architecture, then, but it still conveys a useful image of the system-it imposes useful constraints on the architecture and, in turn, on the system.

Reference architecture is a reference model mapped onto software elements (that cooperatively implement the functionality defined in the reference model) and the data flows between them. Whereas a reference model divides the functionality, reference architecture is the mapping of that functionality onto system decomposition. The mapping may be, but by no means necessarily is, one to one. A software element may implement part of a function or several functions.

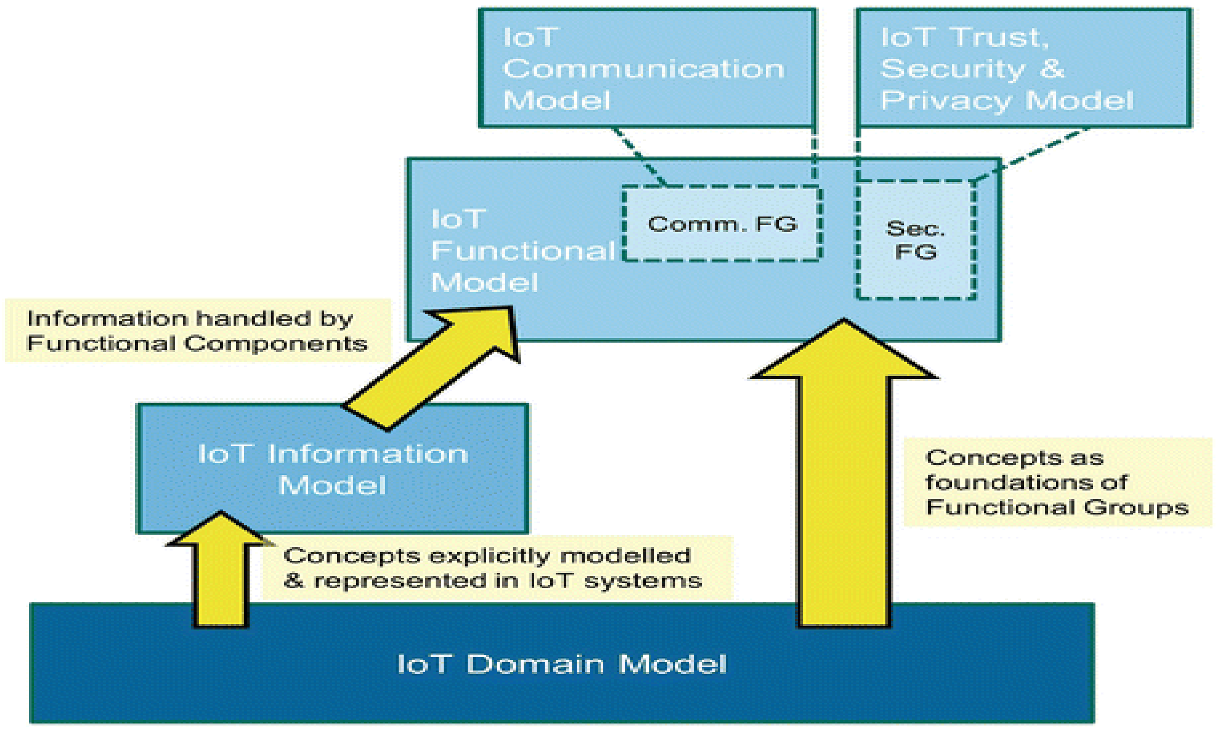
Figure 2.2. The relationships of reference models, architectural patterns, reference architectures, and software architectures. (The arrows indicate that subsequent concepts contain more design elements.)

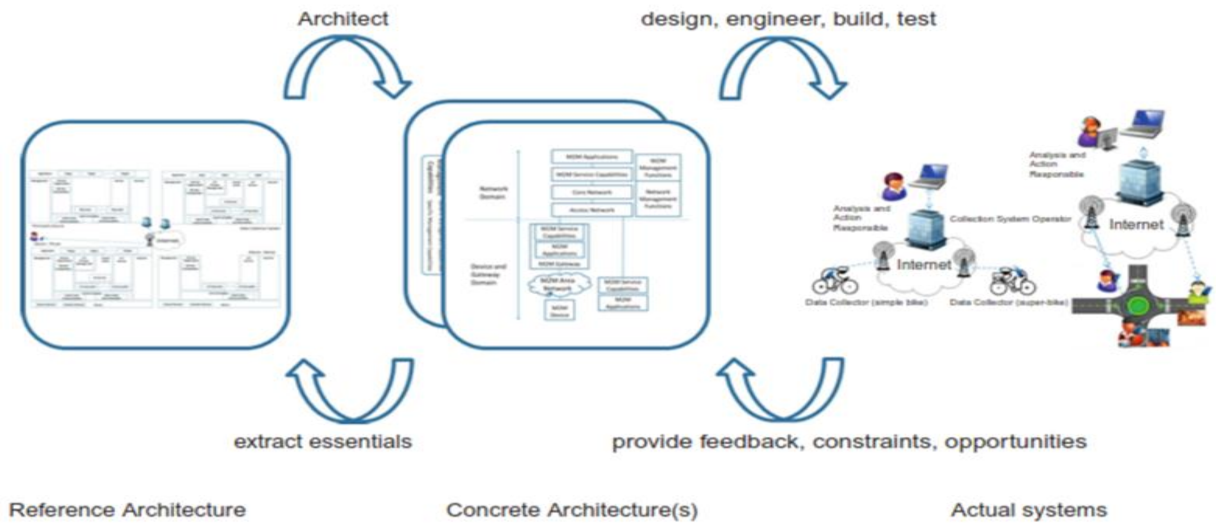


Reference Model and Architecture

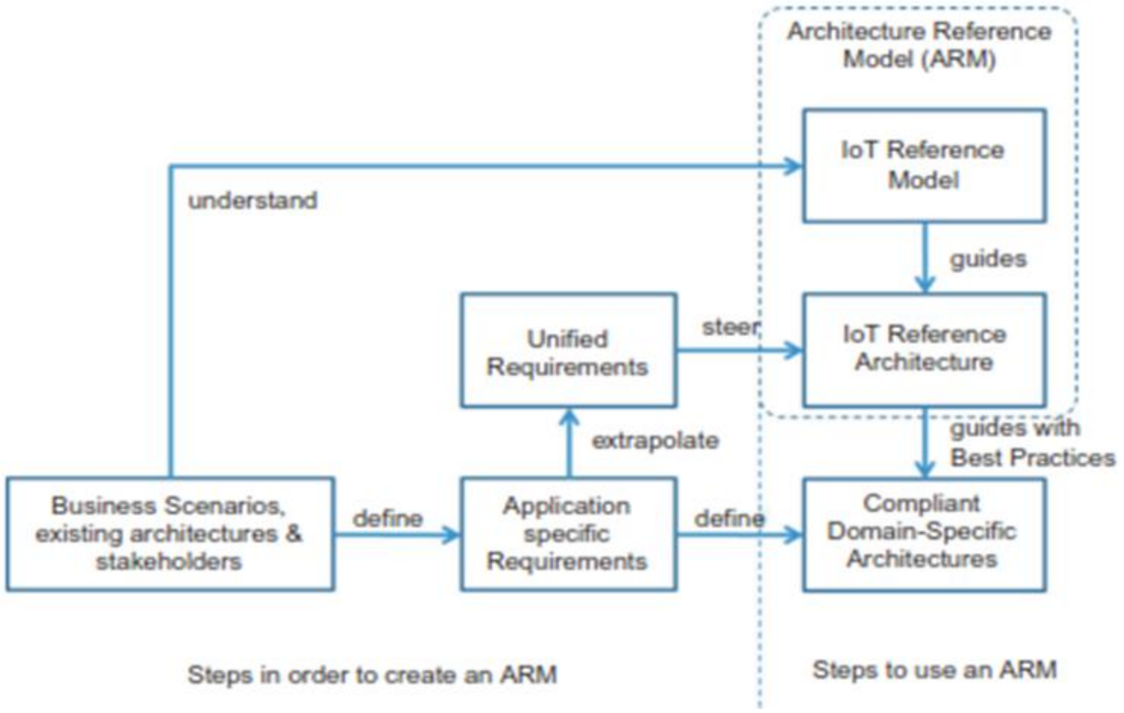
•An ARM consists of two main parts: a Reference model and a Reference Architecture.

•A reference model describes the domain using a number of sub-models


IoT Communication Model
IoT Trust, Security & Privacy Model
IoT Functional Model
Comm. FG
Sec. FG
Information handled by Functional Components
IoT Information Model
Concepts as foundations of Functional Groups
Concepts explicitly modelled & represented in IoT systems
IoT Domain Model



From Reference to concrete architecture and actual system


Architecture Reference Model (ARM)
IoT Reference Model
guides
IoT Reference Architecture
understand
Unified Requirements
steer
extrapolate
guides with Best Practices
Business Scenarios, existing architectures & stakeholders
define
Application specific Requirements
define
Compliant Domain-Specific Architectures
Steps in order to create an ARM
Steps to use an ARM

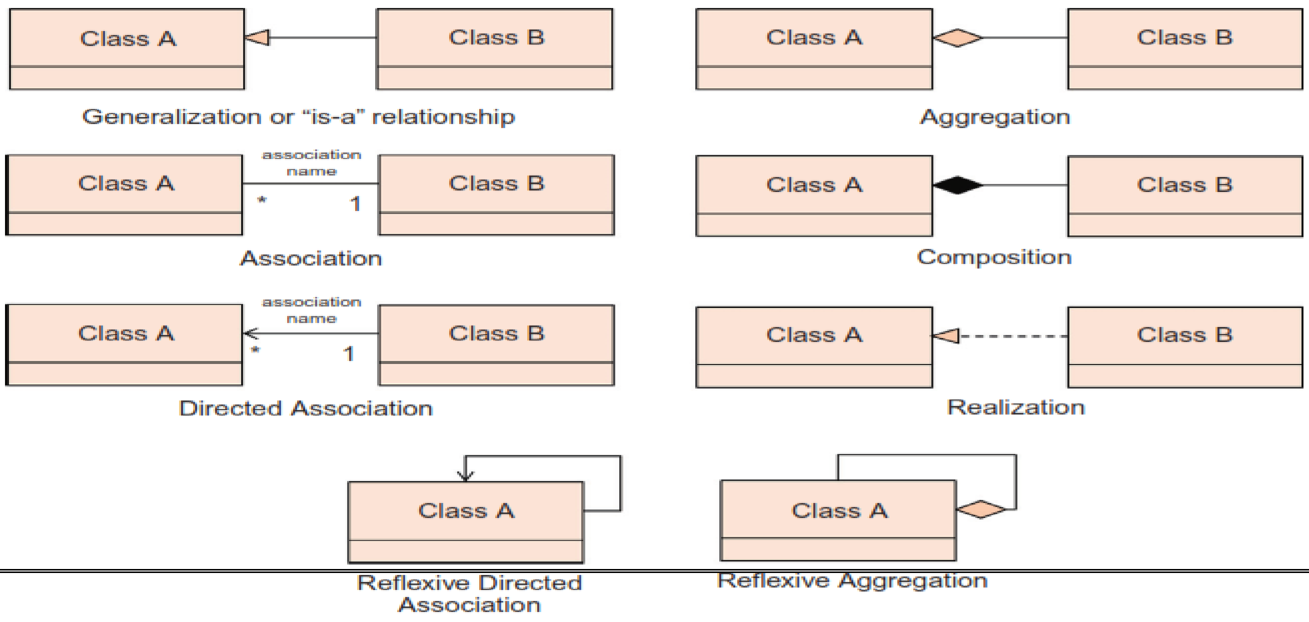
IOT Reference architecture and reference model dependency

IOT Reference Model

IoT domain model

The domain model captures the basic attributes of the main concepts and the relationship between these concepts. A domain model also serves as a tool for human communication between people working in the domain in question and between people who work across different domains.

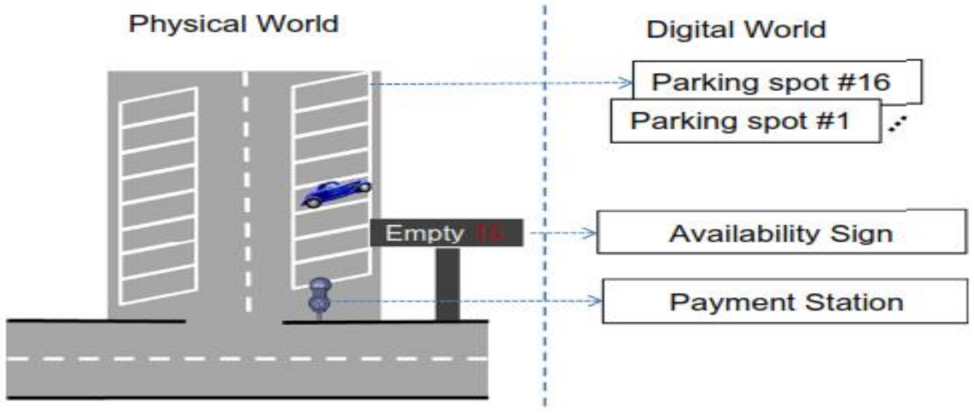
Model notation and semantics



Main concepts

The IoT is a support infrastructure for enabling objects and places in the physical world to have a corresponding representation in the digital world.

Physical vs. Virtual World



The Devices are physical artifacts with which the physical and virtual worlds interact. Devices as mentioned before can also be Physical Entities for certain types of applications, such as management applications when the interesting entities of a system are the Devices themselves and not the surrounding environment. For the IoT Domain Model, three kinds of Device types are the most important:

1. Sensors:

- These are simple or complex Devices that typically involve a transducer that converts physical properties such as temperature into electrical signals.
- These Devices include the necessary conversion of analog electrical signals into digital signals, e.g. a voltage level to a 16-bit number, processing for simple calculations, potential storage for intermediate results, and potentially communication capabilities to transmit the digital representation of the physical property as well receive commands.
- A video camera can be another example of a complex sensor that could detect and recognise people.

2. Actuators:

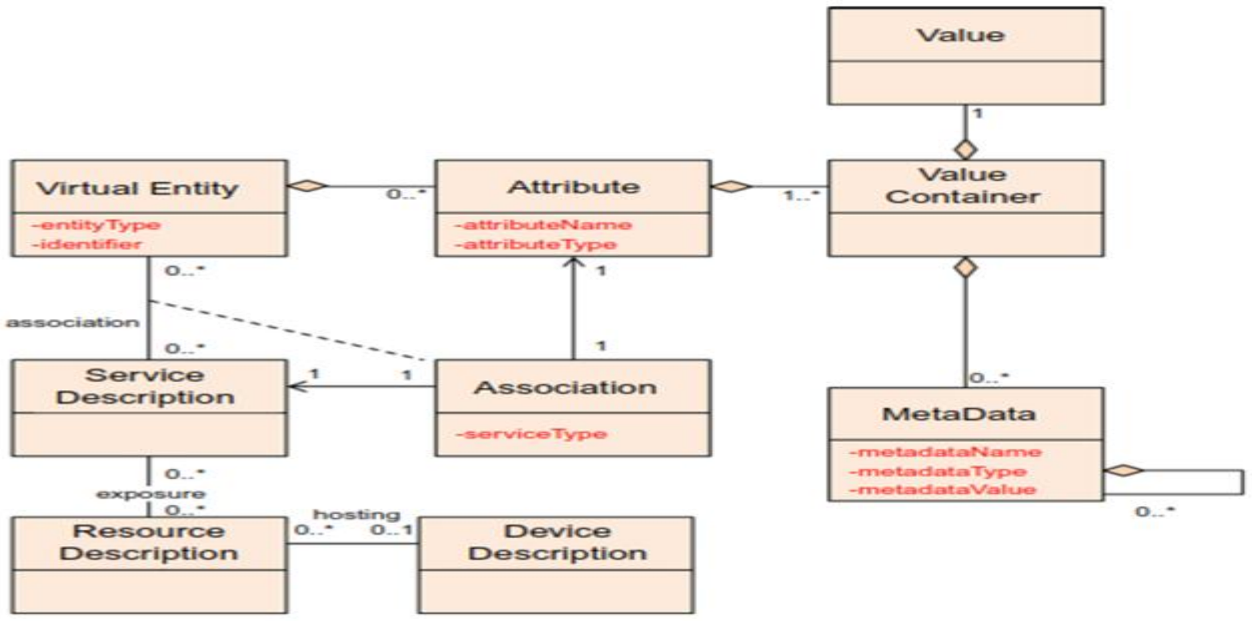
- These are also simple or complex Devices that involve a transducer that converts electrical signals to a change in a physical property (e.g. turn on a switch or move a motor).
- These Devices also include potential communication capabilities, storage of intermediate commands, processing, and conversion of digital signals to analog electrical signals.

3. Tags:

- Tags in general identify the Physical Entity that they are attached to. In reality, tags can be Devices or Physical Entities but not both, as the domain model shows.
- An example of a Tag as a Device is a Radio Frequency Identification (RFID) tag, while a tag as a Physical Entity is a paper-printed immutable barcode or Quick Response (QR) code.
- Either electronic Devices or a paper-printed entity tag contains a unique identification that can be read by optical means (bar codes or QR codes) or radio signals (RFID tags).
- The reader Device operating on a tag is typically a sensor, and sometimes a sensor and an actuator combined in the case of writable RFID tags.

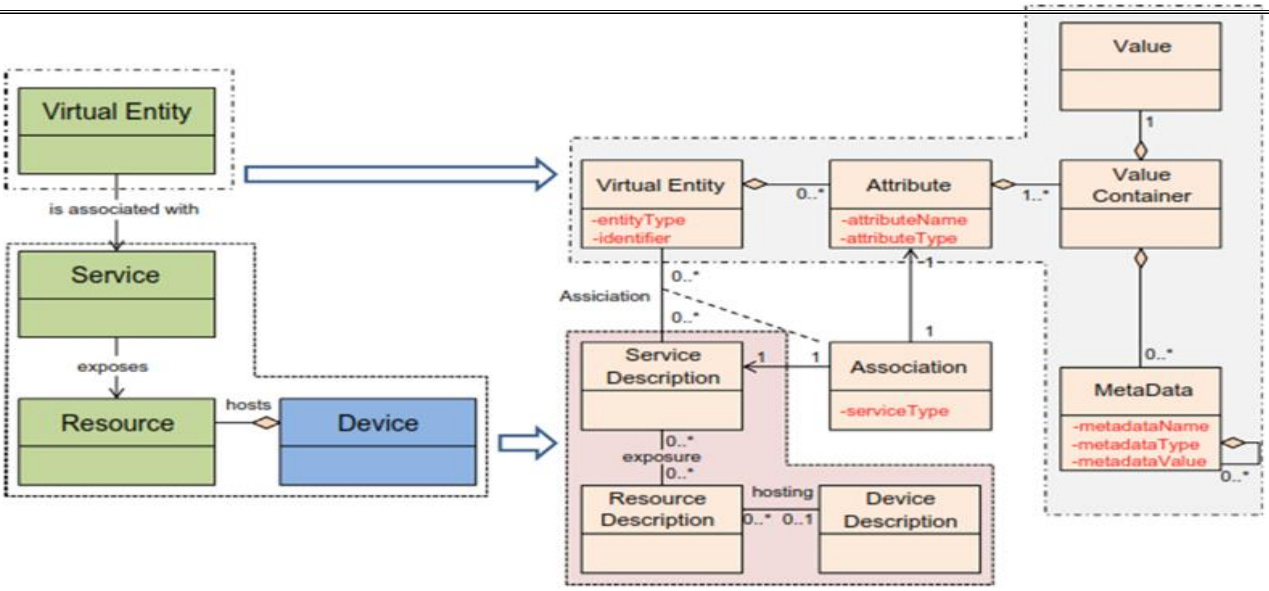
Information Model

- Virtual Entity in the IoT Domain Model is the "Thing" in the Internet of Things, the IoT information model captures the details of a Virtual Entity- centric model. Similar to the IoT Domain Model, the IoT Information Model is presented using Unified Modelling Language (UML) diagrams.



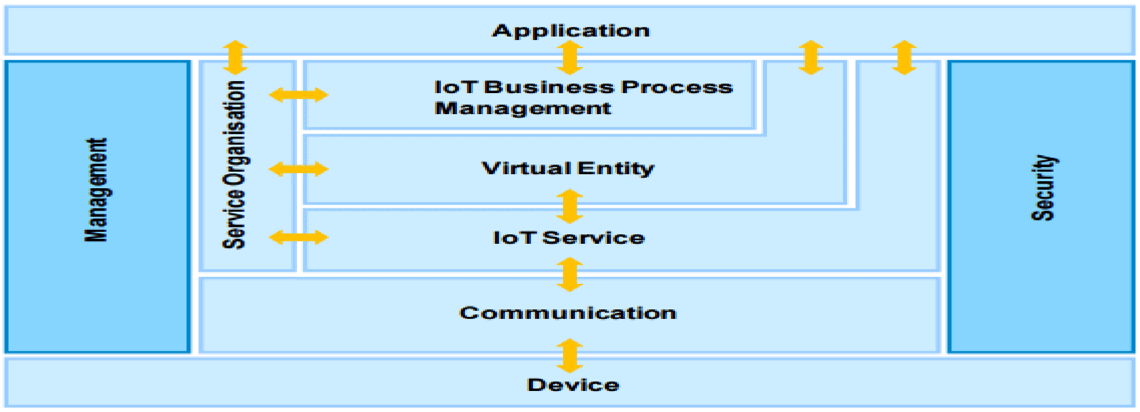
High-level IoT Information Model

Relationship between core concepts of IoT Domain Model and IoT Information Model.



Functional model

The IoT Functional Model aims at describing mainly the Functional Groups (FG) and their interaction with the ARM, while the Functional View of a Reference Architecture describes the functional components of an FG, interfaces, and interactions between the components. The Functional View is typically derived from the Functional Model in conjunction with high-level requirements.



Device functional group

The Device FG contains all the possible functionality hosted by the physical Devices that are used for increment the Physical Entities. This Device functionality includes sensing, actuation, processing, storage, and identification components, the sophistication of which depends on the Device capabilities

Communication functional group

The Communication FG abstracts all the possible communication mechanisms used by the relevant Devices in an actual system in order to transfer information to the digital world components or other Devices.

IoT Service functional group

The IoT Service FG corresponds mainly to the Service class from the IoT Domain Model, and contains single IoT Services exposed by Resources hosted on Devices or in the Network (e.g. processing or storage Resources).

Virtual Entity functional group

The Virtual Entity FG corresponds to the Virtual Entity class in the IoT Domain Model, and contains the necessary functionality to manage associations between Virtual Entities with themselves as well as associations between Virtual Entities and related IoT Services, i.e. the Association objects for the IoT Information Model. Associations between Virtual Entities can be static or dynamic depending on the mobility of the Physical Entities related to the corresponding Virtual Entities.

IoT Service Organization functional group

The purpose of the IoT Service Organisation FG is to host all functional components that support the composition and orchestration of IoT and Virtual Entity services. Moreover, this FG acts as a service hub between several other functional groups such as the IoT Process Management FG when, for example, service requests from Applications or the IoT Process Management are directed to the Resources implementing the necessary Services.

IoT Process Management functional group

The IoT Process Management FG is a collection of functionalities that allows smooth integration of IoT-related services (IoT Services, Virtual Entity Services, Composed Services) with the Enterprise (Business) Processes.

Management functional group

The Management FG includes the necessary functions for enabling fault and performance monitoring of the system, configuration for enabling the system to be flexible to changing User demands, and accounting for enabling subsequent billing for the usage of the system. Support functions such as management of ownership, administrative domain, rules and rights of functional components, and information stores are also included in the Management FG.

Security functional group

The Security FG contains the functional components that ensure the secure operation of the system as well as the management of privacy. The Security FG contains components for Authentication of Users (Applications, Humans), Authorisation of access to Services by Users, secure communication (ensuring integrity and confidentiality of messages) between entities of the system such as Devices, Services, Applications, and last but not least, assurance of privacy of sensitive information relating to Human Users.

Application functional group

The Application FG is just a placeholder that represents all the needed logic for creating an IoT application. The applications typically contain custom logic tailored to a specific domain such as a Smart Grid

Communication model

Safety

the IoT Reference Model can only provide IoT-related guidelines for ensuring a safe system to the extent possible and controllable by a sys- tem designer.
Eg: smart grid.

Privacy

Because interactions with the physical world may often include humans, protecting the User privacy is of utmost importance for an IoT system. The IoT-A Privacy Model depends on the following functional components: Identity Management, Authentication, Authorisation, and Trust & Reputation

Trust

Generally, an entity is said to 'trust' a second entity when the first entity makes the assumption that the second entity will behave exactly as the first entity expects."

Security

The Security Model for IoT consists of communication security that focuses mostly on the confidentiality and integrity protection of interacting entities and functional components such as Identity Management, Authentication, Authorisation, and Trust & Reputation.

MQTT protocol

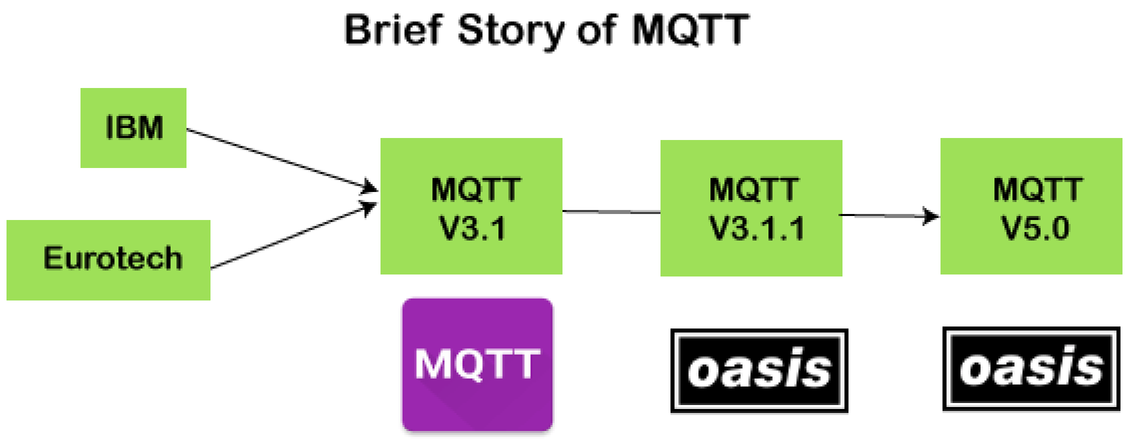
MQTT stands for **Message Queuing Telemetry Transport**. MQTT is a machine to machine internet of things connectivity protocol. It is an extremely lightweight and publish-subscribe messaging transport protocol. This protocol is useful for the connection with the remote location where the bandwidth is a premium. These characteristics make it useful in various situations, including constant environment such as for communication machine to machine and internet of things contexts. It is a publish and subscribe system where we can publish and receive the messages as a client. It makes it easy for communication between multiple devices. It is a simple messaging protocol designed for the constrained devices and with low bandwidth, so it's a perfect solution for the internet of things applications.

Characteristics of MQTT

The MQTT has some unique features which are hardly found in other protocols. Some of the features of an MQTT are given below:

- It is a machine to machine protocol, i.e., it provides communication between the devices.
- It is designed as a simple and lightweight messaging protocol that uses a publish/subscribe system to exchange the information between the client and the server.
- It does not require that both the client and the server establish a connection at the same time.
- It provides faster data transmission, like how WhatsApp/messenger provides a faster delivery. It's a real-time messaging protocol.
- It allows the clients to subscribe to the narrow selection of topics so that they can receive the information they are looking for.

History of MQTT



The MQTT was developed by Dr. Andy Stanford-Clark, IBM, and Arlen Nipper. The previous versions of protocol 3.1 and 3.1.1 were made available under MQTT ORG. In 2014, the MQTT was officially published by OASIS. The OASIS becomes a new home for the development of the MQTT. Then, the OASIS started the further development of the MQTT. Version 3.1.1 is backward comfortable with a 3.1 and brought only minor changes such as changes to the connect message and clarification of the 3.1 version. The recent version of MQTT is 5.0, which is a successor of the 3.1.1 version. Version 5.0 is not backward, comfortable like version 3.1.1. According to the specifications, version 5.0 has a significant number of features that make the code in place.

- Enhancement in the scalability and the large-scale system in order to set up with the thousands or the millions of devices.
- Improvement in the error reporting

MQTT Architecture

To understand the MQTT architecture, we first look at the components of the MQTT.

- **Message**
- **Client**
- **Server or Broker**
- **TOPIC**

Message

The message is the data that is carried out by the protocol across the network for the application. When the message is transmitted over the network, then the message contains the following parameters:

1. Payload data
2. Quality of Service (QoS)
3. Collection of Properties
4. Topic Name

Client

In MQTT, the subscriber and publisher are the two roles of a client. The clients subscribe to the topics to publish and receive messages. In simple words, we can say that if any program or device uses an MQTT, then that device is referred to as a client. A device is a client if it opens the network connection to the server, publishes messages that other clients want to see, subscribes to the messages that it is interested in receiving, unsubscribes to the messages that it is not interested in receiving, and closes the network connection to the server.

In MQTT, the client performs two operations:

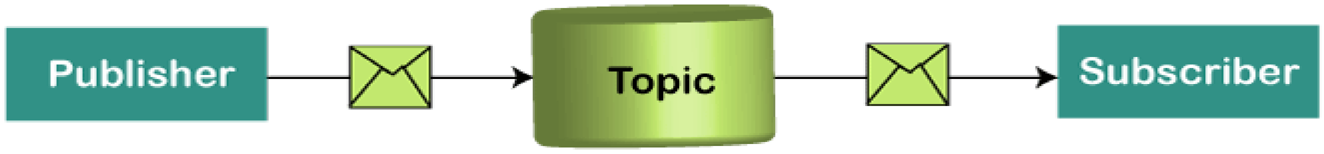
Publish: When the client sends the data to the server, then we call this operation as a publish.

Subscribe: When the client receives the data from the server, then we call this operation a subscription

Server

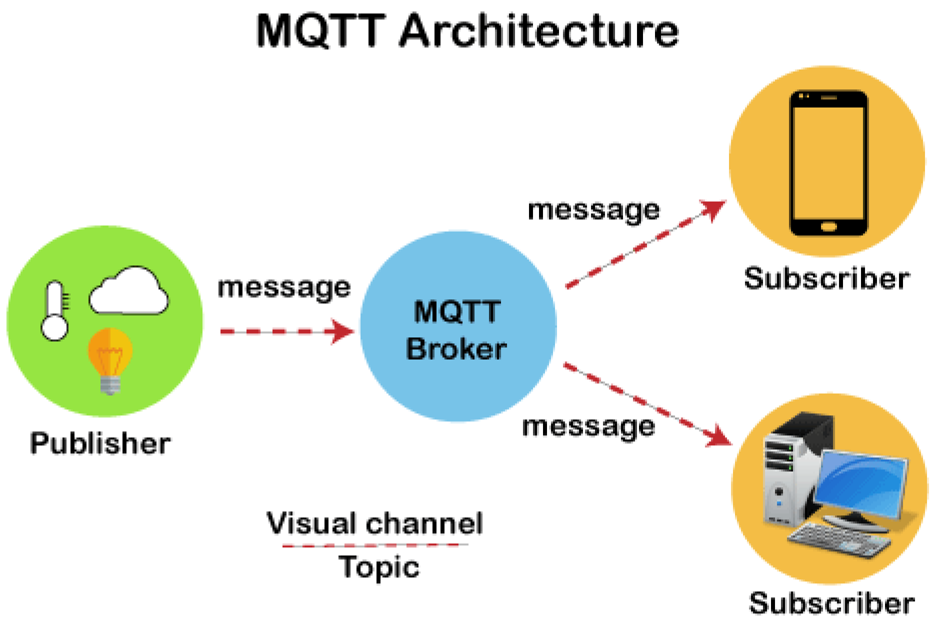
The device or a program that allows the client to publish the messages and subscribe to the messages. A server accepts the network connection from the client, accepts the messages from the client, processes the subscribe and unsubscribe requests, forwards the application messages to the client, and closes the network connection from the client.

TOPIC



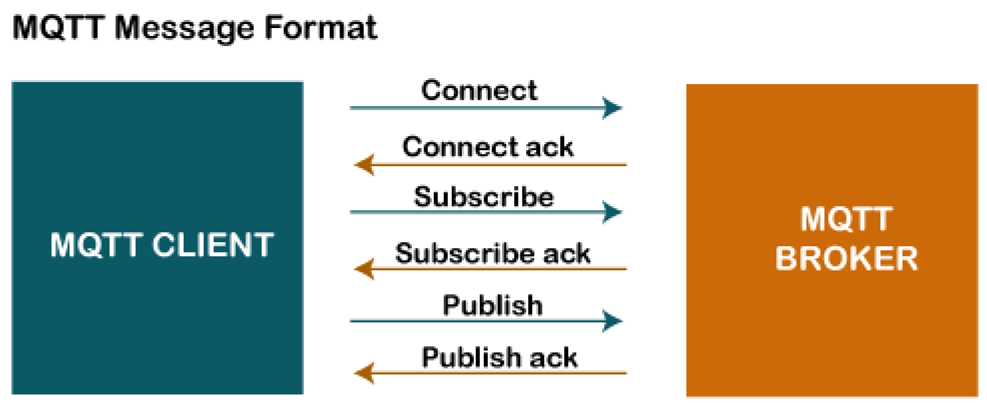
The label provided to the message is checked against the subscription known by the server is known as TOPIC.

Architecture of MQTT



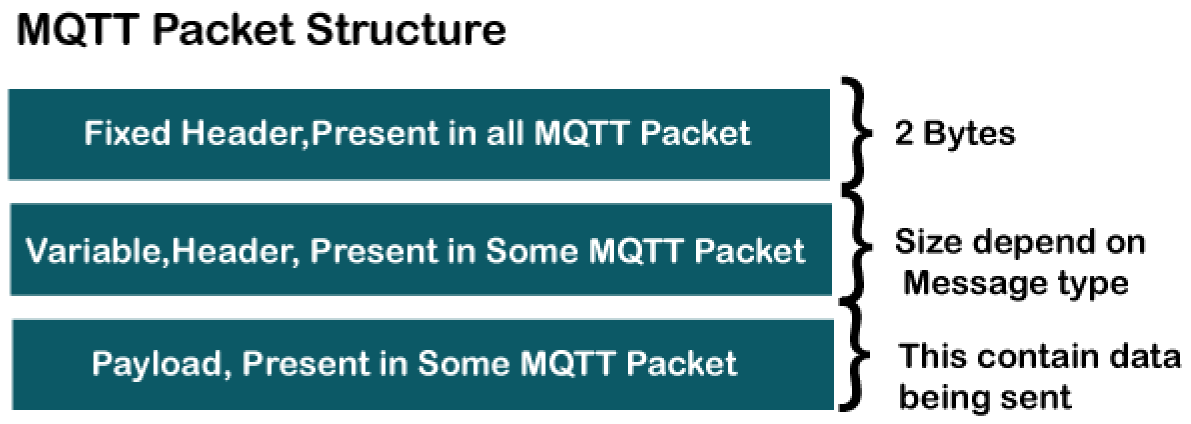
Now we will look at the architecture of MQTT. To understand it more clearly, we will look at the example. Suppose a device has a temperature sensor and wants to send the rating to the server or the broker. If the phone or desktop application wishes to receive this temperature value on the other side, then there will be two things that happened. The publisher first defines the topic; for example, the temperature then publishes the message, i.e., the temperature's value. After publishing the message, the phone or the desktop application on the other side will subscribe to the topic, i.e., temperature and then receive the published message, i.e., the value of the temperature. The server or the broker's role is to deliver the published message to the phone or the desktop application.

MQTT Message Format



The MQTT uses the command and the command acknowledgment format, which means that each command has an associated acknowledgment. As shown in the above figure that the connect command has connect acknowledgment, subscribe command has subscribe acknowledgment, and publish command has publish acknowledgment. This mechanism is similar to the handshaking mechanism as in TCP protocol.

Now we will look at the packet structure or message format of the MQTT.


MQTT Packet Structure
Fixed Header,Present in all MQTT Packet
2 Bytes
Variable,Header, Present in Some MQTT Packet
Size depend on Message type
Payload, Present in Some MQTT Packet
This contain data being sent

The MQTT message format consists of 2 bytes fixed header, which is present in all the MQTT packets. The second field is a variable header, which is not always present. The third field is a payload, which is also not always present. The payload field basically contains the data which is being sent. We might think that the payload is a compulsory field, but it does not happen. Some commands do not use the payload field, for example, disconnect message.

COAP

CoAP protocol architecture used in IoT (Internet of Things). It mentions CoAP architecture,CoAP message format and CoAP message exchanges between CoAP client and CoAP server. CoAP is the short form of Constrained Application Protocol.

The CoAP protocol is specified in RFC 7252. It is a web transfer protocol which is used in constrained nodes or networks such as WSN, IoT, M2M etc. Hence the name Constrained Application Protocol. The protocol is targetted for Internet of Things (IoT) devices having less memory and less power specifications.

As it is designed for web applications it is also known as "The Web of Things Protocol". It can be used to transport data from few bytes to 1000s of bytes over web applications. It exists between UDP layer and Application layer.

Following are the features of CoAP Protocol:

• It is very efficient RESTful protocol.

• Easy to proxy to/from HTTP.

• It is open IETF standard

• It is Embedded web transfer protocol (coap://)

• It uses asynchronous transaction model.

• UDP is binding with reliability and multicast support.

• GET, POST, PUT and DELETE methods are used.

• URI is supported.

• It uses small and simple 4 byte header.

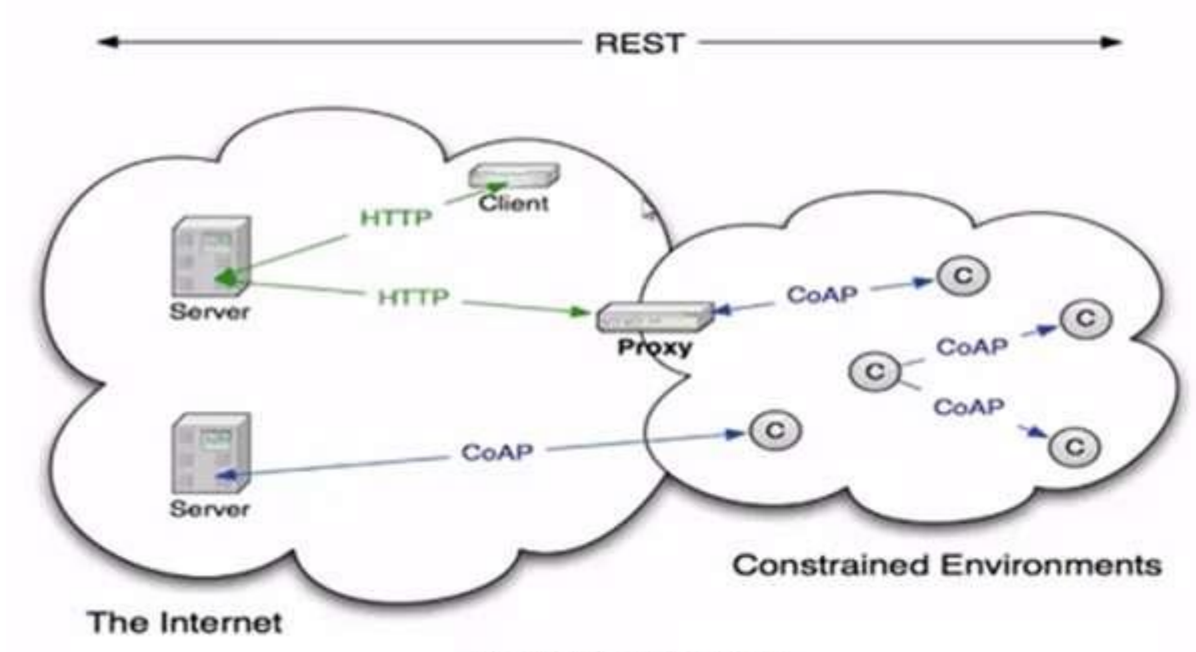
• Supports binding to UDP, SMS and TCP.

• DTLS based PSK, RPK and certificate security is used.

• uses subset of MIME types and HTTP response codes.

• Uses built in discovery mechanism.

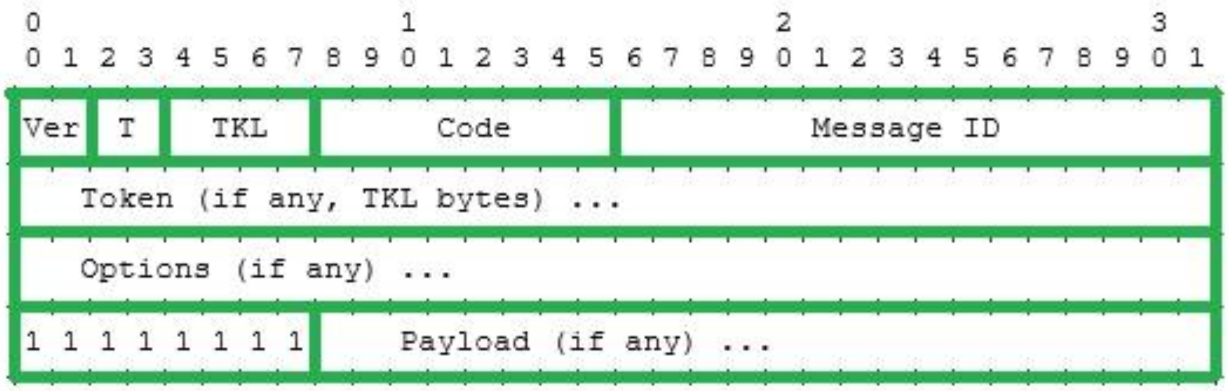
CoAP Architecture



CoAP Architecture

The figure-1 depicts **CoAP Architecture**. As shown it extends normal HTTP clients to clients having resource constraints. These clients are known as CoAP clients. Proxy device bridges gap between constained environment and typical internet environment based on HTTP protocols. Same server takes care of both HTTP and CoAP protocol messages.

CoAP Message Format | CoAP Header

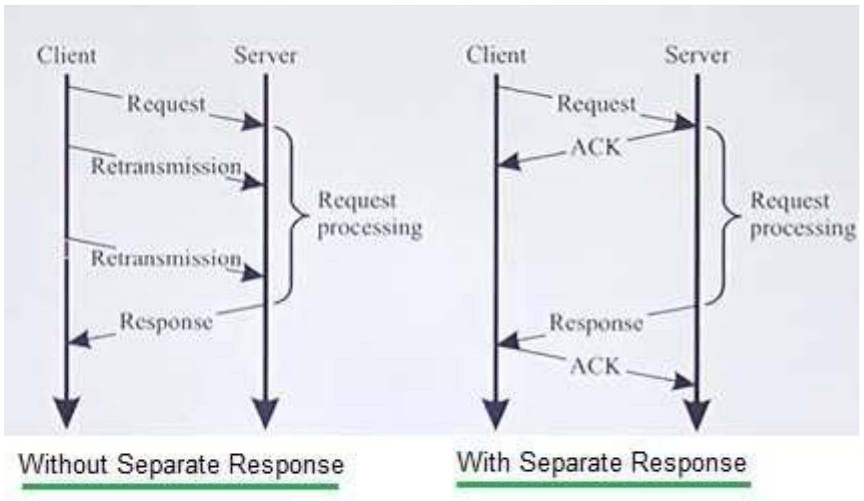


CoAP Message Format

The figure-2 depicts **CoAP message format** consists of 4 bytes header followed by token value (from 0 to 8 bytes). The table below mentions header which consists of 4 bytes i.e. 32 bits.

CoAP message header	Description
Ver	It is 2 bit unsigned integer. It mentions CoAP version number. Set to one.
T	It is 2 bit unsigned integer. Indicates message type viz. confirmable (0), non-confirmable (1), ACK (2) or RESET(3).
TKL	It is 4 bit unsigned integer, Indicates length of token (0 to 8 bytes).
Code	It is 8 bit unsigned integer, It is split into two parts viz. 3 bit class (MSBs) and 5 bit detail (LSBs).
Message ID	16 bit unsigned integer. Used for matching responses. Used to detect message duplication.

CoAP Protocol Message Exchanges



There are two modes in which CoAP protocol messages get exchanged between CoAP client and CoAP server viz. without separate response and with separate response.

With separate response, server notifies client about receipt of the request message. This will increase processing time but help in avoiding unnecessary retransmissions.

CoAP IoT is unreliable protocol due to use of UDP. Hence CoAP messages reach unordered or will get lost when they arrive at destination.

To make CoAP as reliable protocol, stop and wait with exponential backoff retransmission feature is incorporated in it. Duplicate detection is also introduced.

IOT FRAMEWORK

An **IoT framework** can be defined as a set of protocols, tools, and standards that provide a specific structure for developing and deploying IoT applications and services. In other words, an IoT framework gives you the basics for building your own application.

The framework of IoT typically includes a combination of the following:

- Hardware
- Software
- Networking elements (IoT protocols)
- Device management
- Security
- Data management
- Application development
- Cloud-based platform.

These components work together to enable the seamless integration of IoT devices (what is IoT) and systems. For example, the device management is necessary for updating and monitoring the performance of the device. Protocols allow the different connections between devices and the internet.

There needs to be a cloud platform where data will be processed and stored, this also connects with an application or platform that is in charge of displaying the data and allows other functions or services.

Types of IoT frameworks

There are *IoT frameworks that are open source*, and others that are proprietary. IoT framework open source doesn't mean that it is free, many people get confused with this concept. We have even talked about IoT open source devices.

Open source frameworks are useful because they provide access to the code of the platform in order to make any modifications as needed, and build the IoT application with more freedom. Finally, proprietary frameworks are IoT frameworks that don't grant access to the source code, they have an already established platform from which IoT product design and development consultants can start working.

Proprietary IoT frameworks:
AWS IoT, Azure IoT, IBM Watson IoT, and Google IoT Core.

Open-source IoT frameworks:
DeviceHive, Mainflux, KAA IoT, Eclipse, and OpenHab.

What is the difference between IoT framework and IoT architecture?

IoT architecture and IoT framework are related but distinct concepts in the field of the Internet of Things. In short, IoT architecture defines the overall design and structure of an IoT system, while an IoT framework provides the fundamentals for developing and deploying IoT applications and services.

In other words, the IoT architecture is the blueprint for a whole IoT system; on the other hand, the framework IoT is the set of tools that helps to build said system.

An Introduction to ThingSpeak

Introduction

The Internet of Things(IoT) is a system of ‘connected things’. The things generally comprise of an embedded operating system and an ability to communicate with the internet or with the neighboring things. One of the key elements of a generic IoT system that bridges the various ‘things’ is an IoT service. An interesting implication from the ‘things’ comprising the IoT systems is that the things by themselves cannot do anything. At a bare minimum, they should have an ability to connect to other ‘things’. But the real power of IoT is harnessed when the things connect to a ‘service’ either directly or via other ‘things’. In such systems, the service plays the role of an invisible manager by providing capabilities ranging from simple data collection and monitoring to complex data analytics. The below diagram illustrates where an IoT service fits in an IoT ecosystem:

One such IoT application platform that offers a wide variety of analysis, monitoring and counter-action capabilities is ‘ThingSpeak’.

What is ThingSpeak?

ThingSpeak is a platform providing various services exclusively targeted for building IoT applications. It offers the capabilities of

1. Real-time data collection,

2. Visualizing the collected data in the form of charts

3. Ability to create plugins and apps for collaborating with web services, social network and other APIs.

We will consider each of these features in detail below.

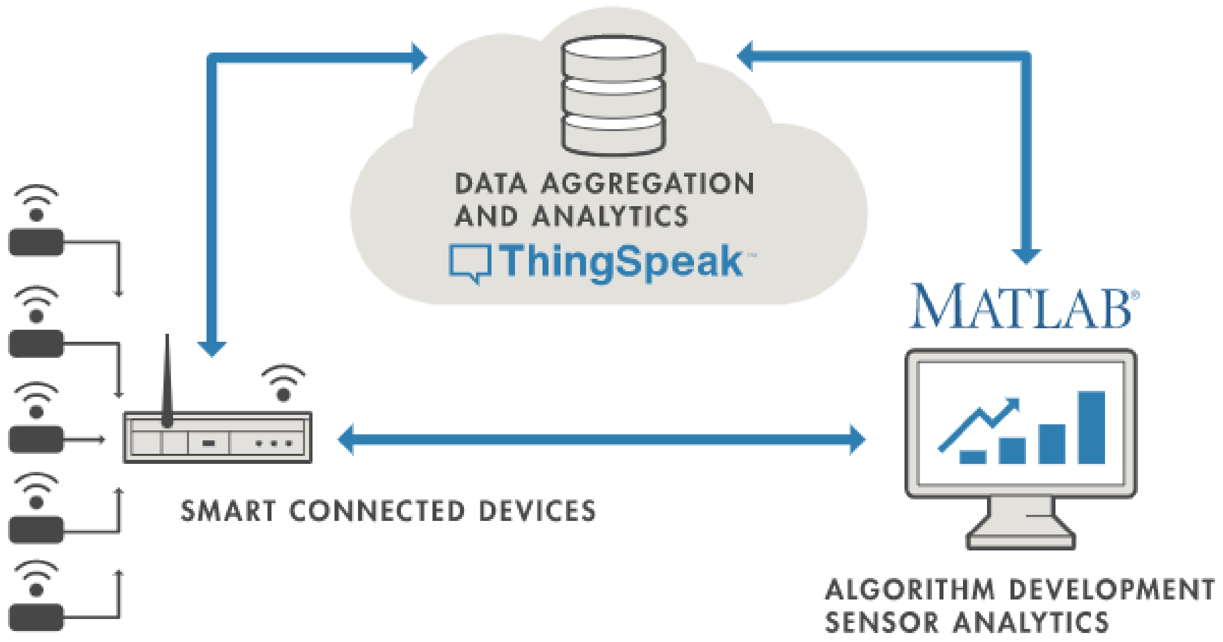
The core element of ThingSpeak is a 'ThingSpeak Channel'. A channel stores the data that we send to ThingSpeak and comprises of the below elements:

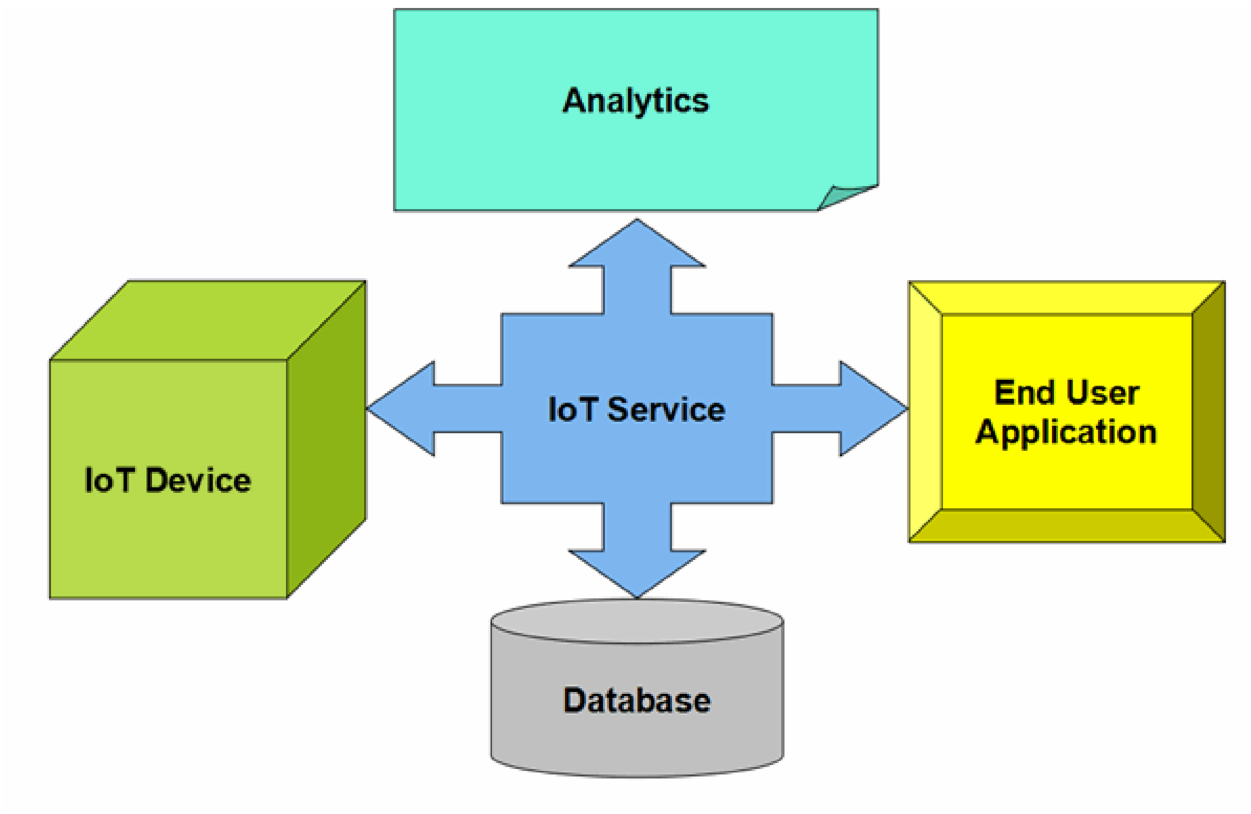
- 8 fields for storing data of any type - These can be used to store the data from a sensor or from an embedded device.
- 3 location fields - Can be used to store the latitude, longitude and the elevation. These are very useful for tracking a moving device.
- 1 status field - A short message to describe the data stored in the channel.

To use ThingSpeak, we need to signup and create a channel. Once we have a channel, we can send the data, allow ThingSpeak to process it and also retrieve the same. Let us start exploring ThingSpeak by signing up and setting up a channel.

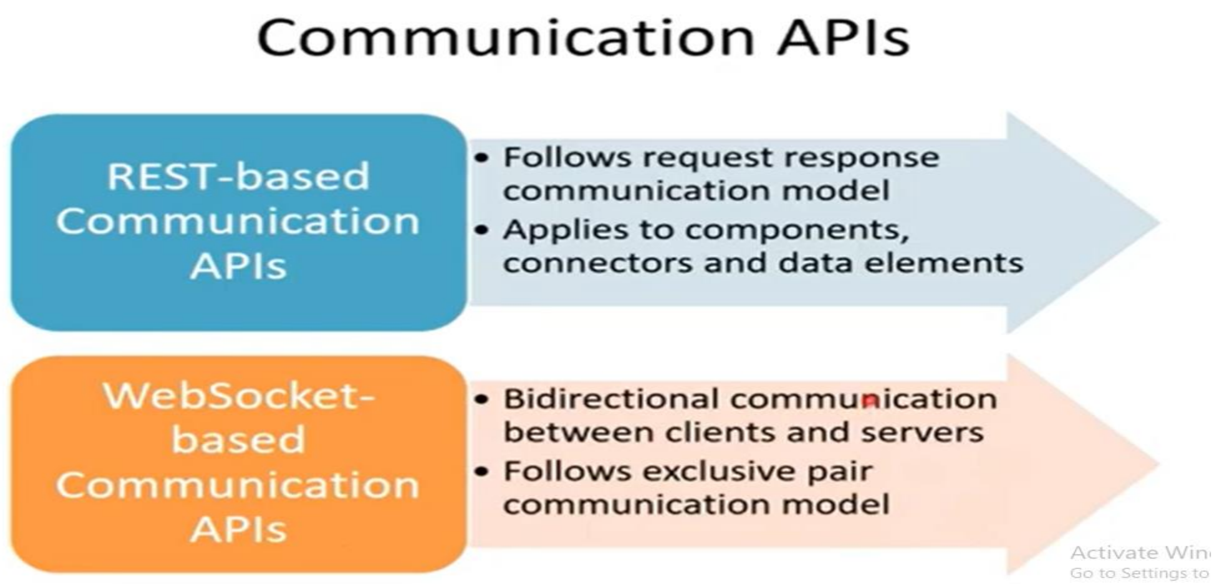
Link: https://www.codeproject.com/articles/845538/an-introduction-to-thingspeak

ThingSpeak Architecture:




Analytics
IoT Device
IoT Service
End User Application
Database

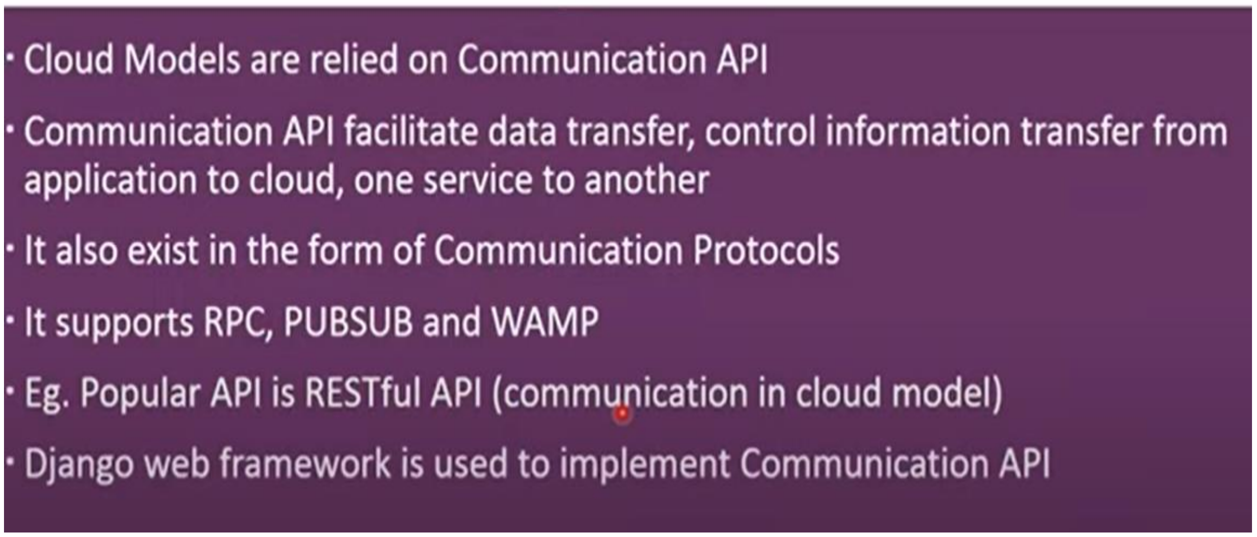
Introduction to Cloud Storage models and communication APIs



Communication APIs

REST-based Communication APIs
- Follows request response communication model
- Applies to components, connectors and data elements

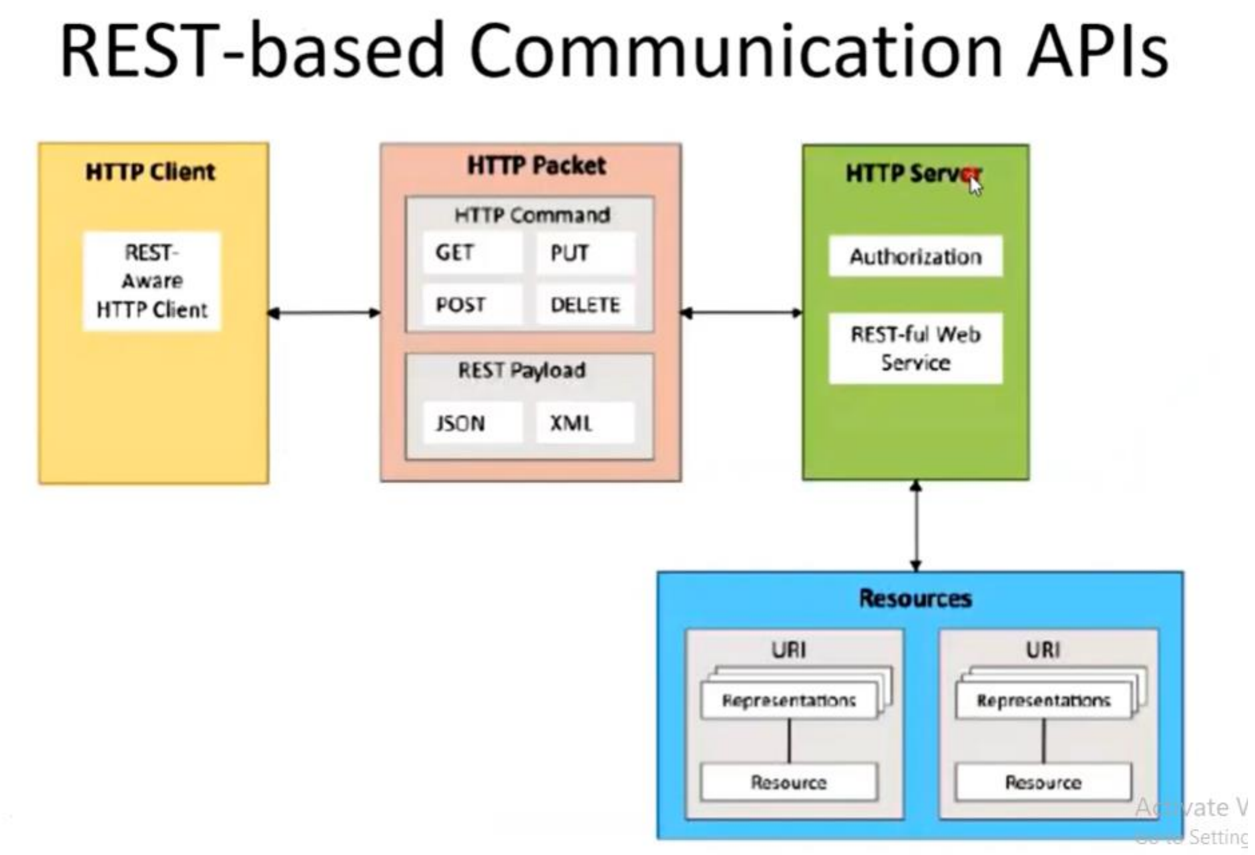
WebSocket-based Communication APIs
- Bidirectional communication between clients and servers
- Follows exclusive pair communication model



- Cloud Models are relied on Communication API
- Communication API facilitate data transfer, control information transfer from application to cloud, one service to another
- It also exist in the form of Communication Protocols
- It supports RPC, PUBSUB and WAMP
- Eg. Popular API is RESTful API (communication in cloud model)
- Django web framework is used to implement Communication API

REST API

Introduction:



WEBSOCKET

API stands for Application Programming Interface. API can be defined as a predefined set of instructions that describes how different applications can communicate with one another. So one can think of API as a middleman for transferring data between a web server and the application. Every app is using some kind of API to transfer data between applications. API provides the ability for different applications to open up their functionalities and data to other external parties to make use of it and develop new applications. Google Maps API and Twitter API are some well-known examples.

WebSocket: WebSocket is a communication protocol that is mainly used for communication between a client and server.

Features of WebSocket are:

- **Full-Duplex Protocol:** WebSocket is a full-duplex protocol as it allows the application to send and receive data at the same time.
- **Stateful Protocol:** It means the connection between server and client will not be terminated until and unless closed by any one of them either by the client or by the server. Once the connection is terminated from one end it is also closed by another end.
- **3-way handshake:** Websocket uses a 3-way handshake also known as TCP connection for establishing communication between a client and server.

WebSocket API: WebSocket API allows us to create web sockets; it is a javascript API that is capable of full-duplex communication using a TCP connection. WebSocket uses port 80 by default.

Features of Web Socket API are:

- Bidirectional means data can be sent and received by both sides client-side as well as the server-side.
- Use of full-duplex model for communication.
- It uses a single TCP connection for communication between the client and server.
- Mainly used in real-time applications like chat applications, video calls applications, etc.
- Fast transmission of data can be achieved using web sockets.
- Scaling is possible but only vertically.

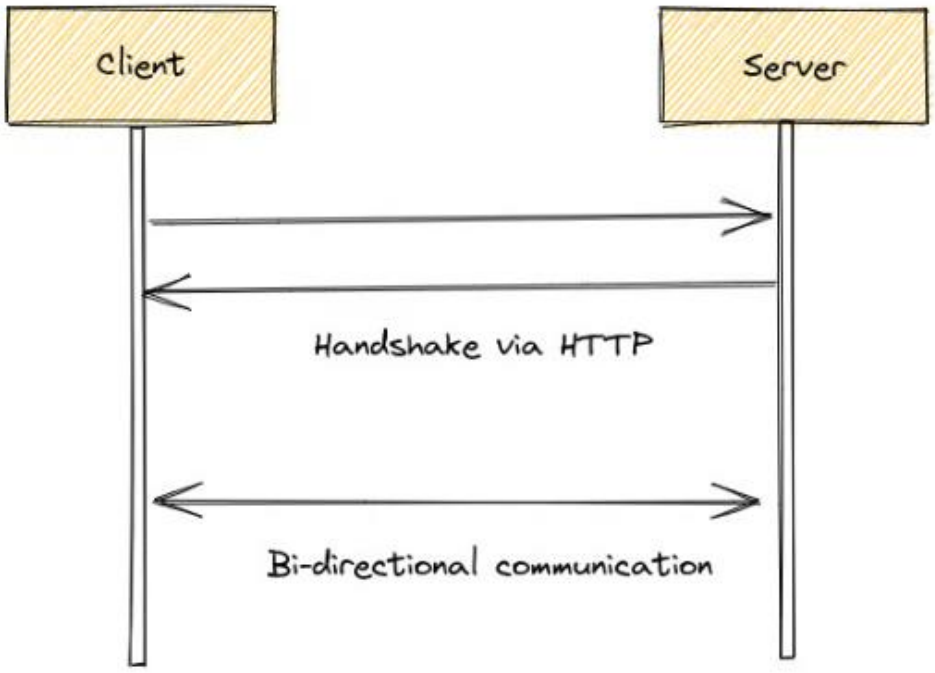
Use of Web Socket API: There is a wide range of uses of WebSocket API's some of them are:

- **Online Education Sector:** Web sockets are obviously used in online education applications because these applications need real-time data for video streaming or for sharing screen which makes web sockets a very good choice because it has the capability to provide all these functionalities.
- **Gaming:** The world is going crazy after games, users want real-time games to play with multiple players along with chat and call facilities so to achieve all these things it is a must for anyone to use web sockets for developing different gaming applications.

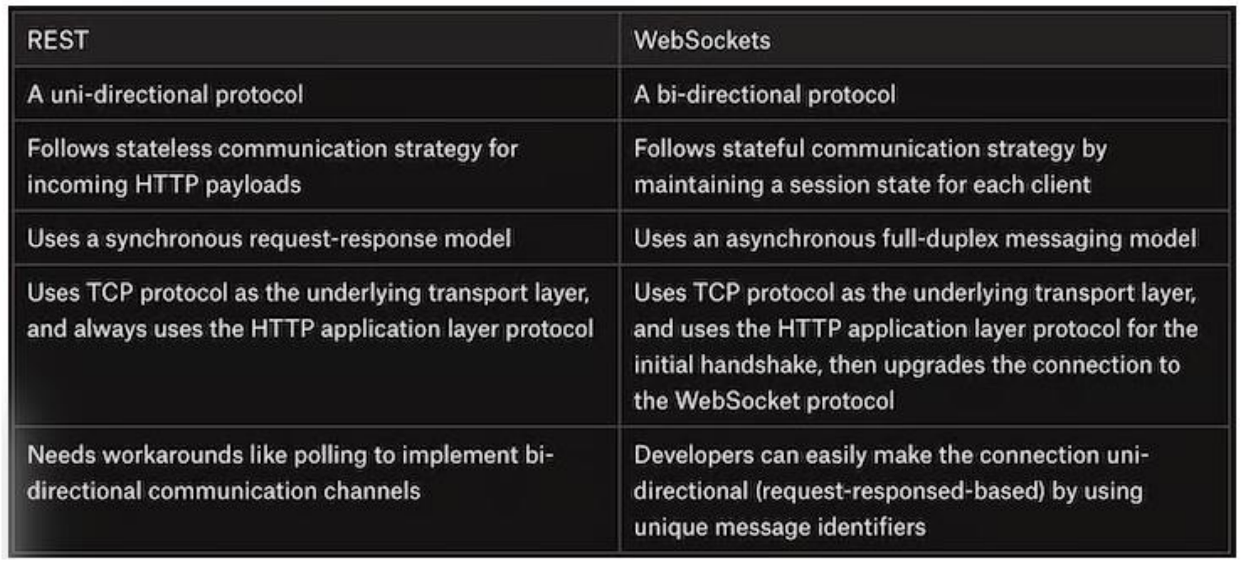
- **Collaborative applications:** We all have used Google docs which makes it possible for multiple persons to use the same workspace and work concurrently, these applications are built using HTML5 Web Sockets.
- **Real-time data visualization:** Visualization of real-time data in an appealing manner was a quite tough task earlier but with the use of html5 web sockets it is really easy.
- **Event Update applications:** Web sockets are extensively used in making applications for giving real-time updates of all platforms to some common platform.
- **Tracking User Behavior:** Organizations are really interested in knowing users' behavior while interacting with a web application so to give better recommendations about the

content or product the user is looking for. In achieving all these things web socket is a must-go choice

Let's look at the following diagram and understand how the WebSocket client and server full-duplex communication channel is formed:

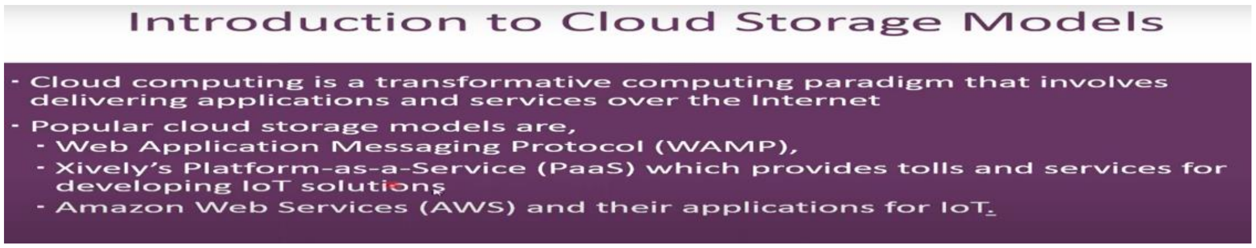


Differences between REST and WebSockets



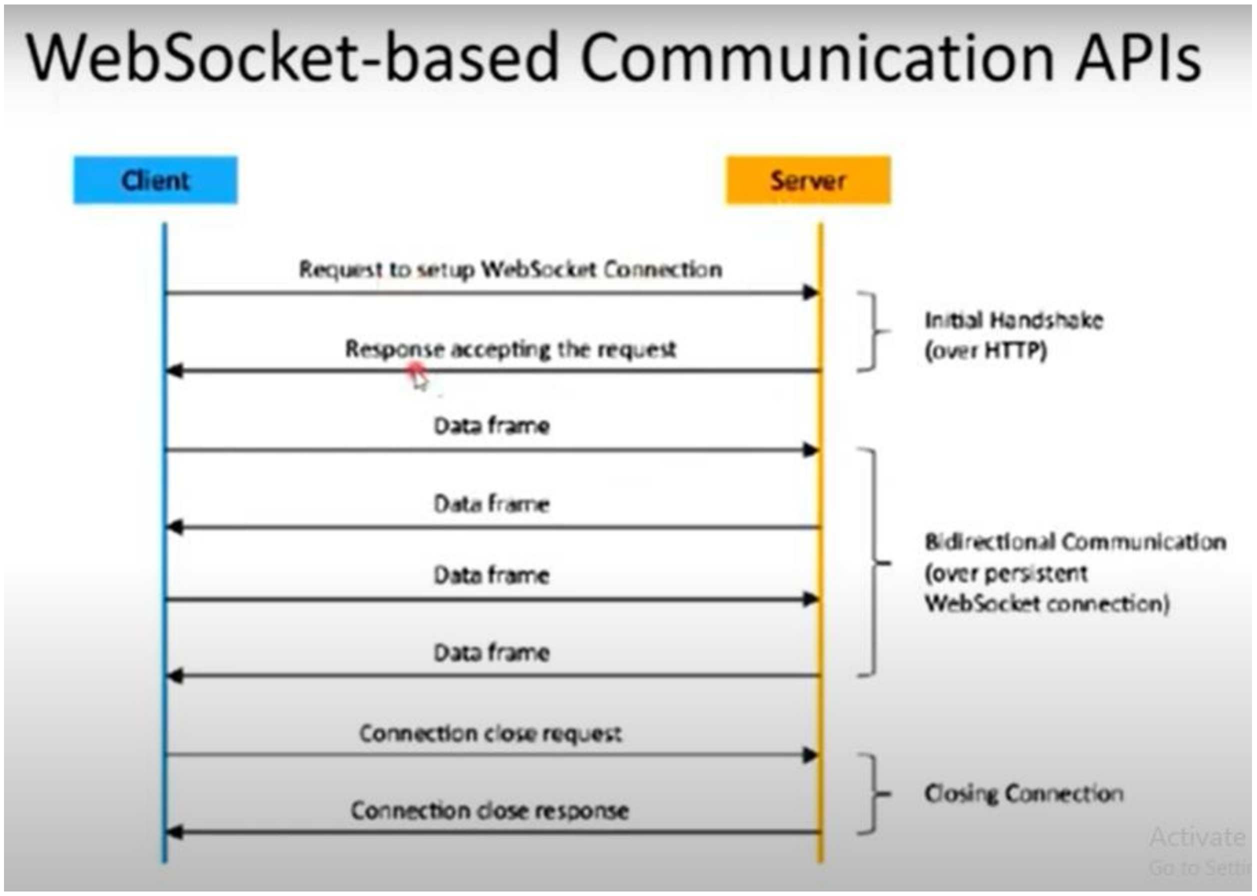
REST	WebSockets
A uni-directional protocol	A bi-directional protocol
Follows stateless communication strategy for incoming HTTP payloads	Follows stateful communication strategy by maintaining a session state for each client
Uses a synchronous request-response model	Uses an asynchronous full-duplex messaging model
Uses TCP protocol as the underlying transport layer, and always uses the HTTP application layer protocol	Uses TCP protocol as the underlying transport layer, and uses the HTTP application layer protocol for the initial handshake, then upgrades the connection to the WebSocket protocol
Needs workarounds like polling to implement bi-directional communication channels	Developers can easily make the connection uni-directional (request-responsed-based) by using unique message identifiers

Introduction



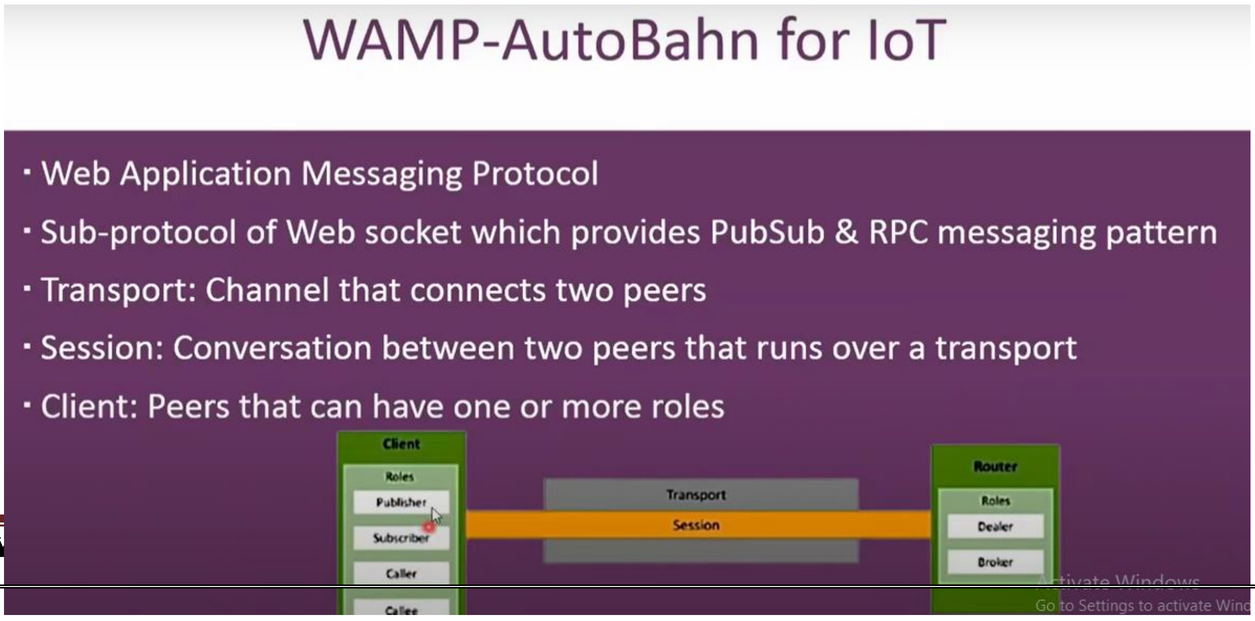
Introduction to Cloud Storage Models

- Cloud computing is a transformative computing paradigm that involves delivering applications and services over the Internet
- Popular cloud storage models are,
  - Web Application Messaging Protocol (WAMP),
  - Xively's Platform-as-a-Service (PaaS) which provides tolls and services for developing IoT solutions
  - Amazon Web Services (AWS) and their applications for IoT.



WebSocket-based Communication APIs

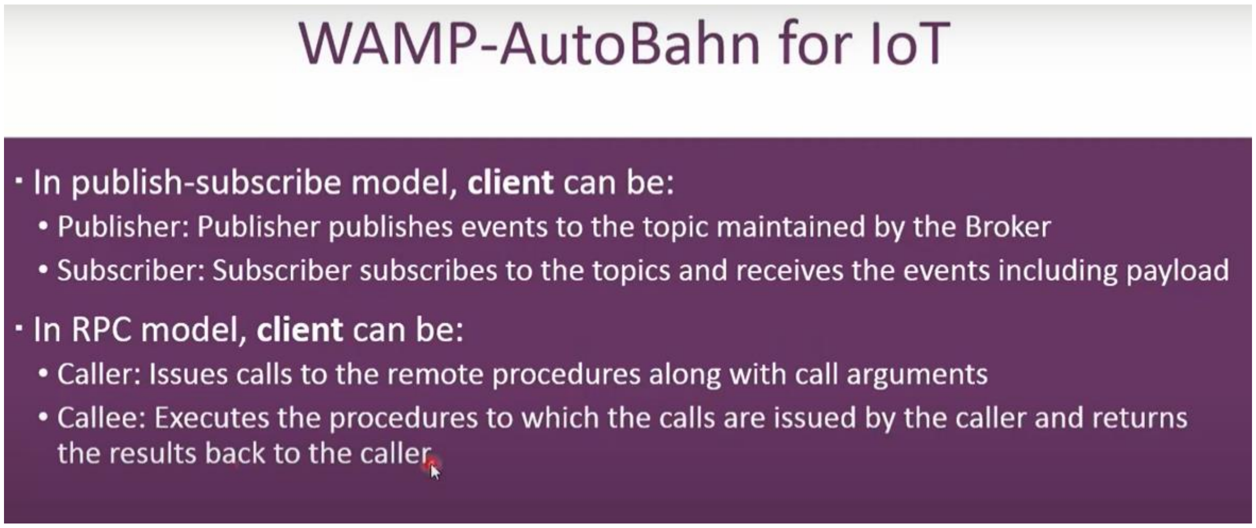
Client
Server
Request to setup WebSocket Connection
Response accepting the request
Initial Handshake (over HTTP)
Data frame
Data frame
Data frame
Data frame
Bidirectional Communication (over persistent WebSocket connection)
Connection close request
Connection close response
Closing Connection



WAMP-AutoBahn for IoT

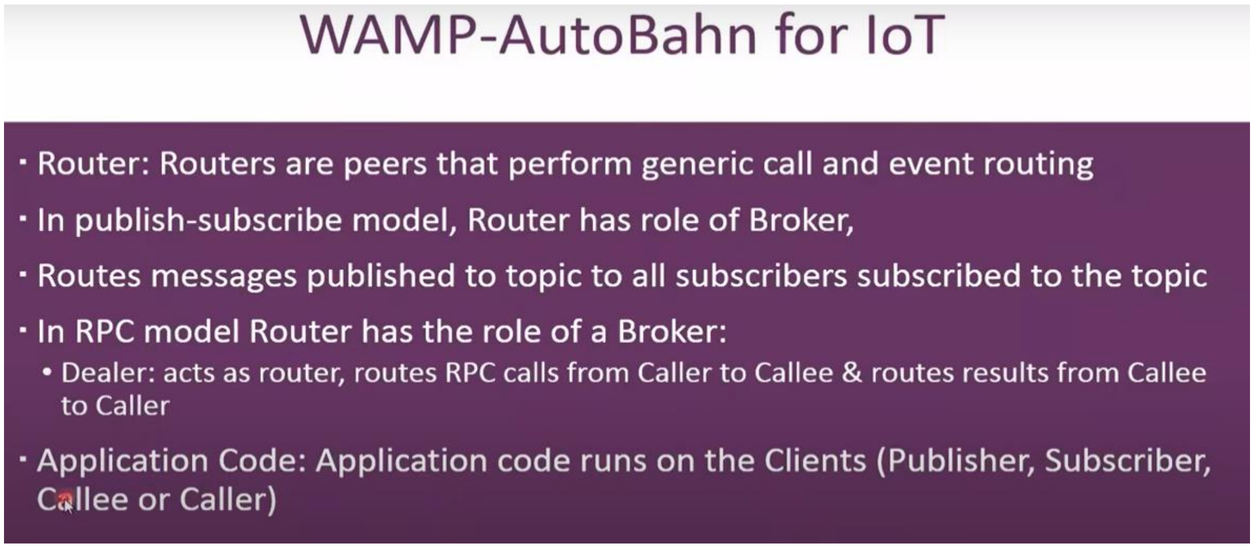
- Web Application Messaging Protocol
- Sub-protocol of Web socket which provides PubSub & RPC messaging pattern
- Transport: Channel that connects two peers
- Session: Conversation between two peers that runs over a transport
- Client: Peers that can have one or more roles

Client
Roles
Publisher
Subscriber
Caller
Callee
Transport
Session
Router
Roles
Dealer
Broker



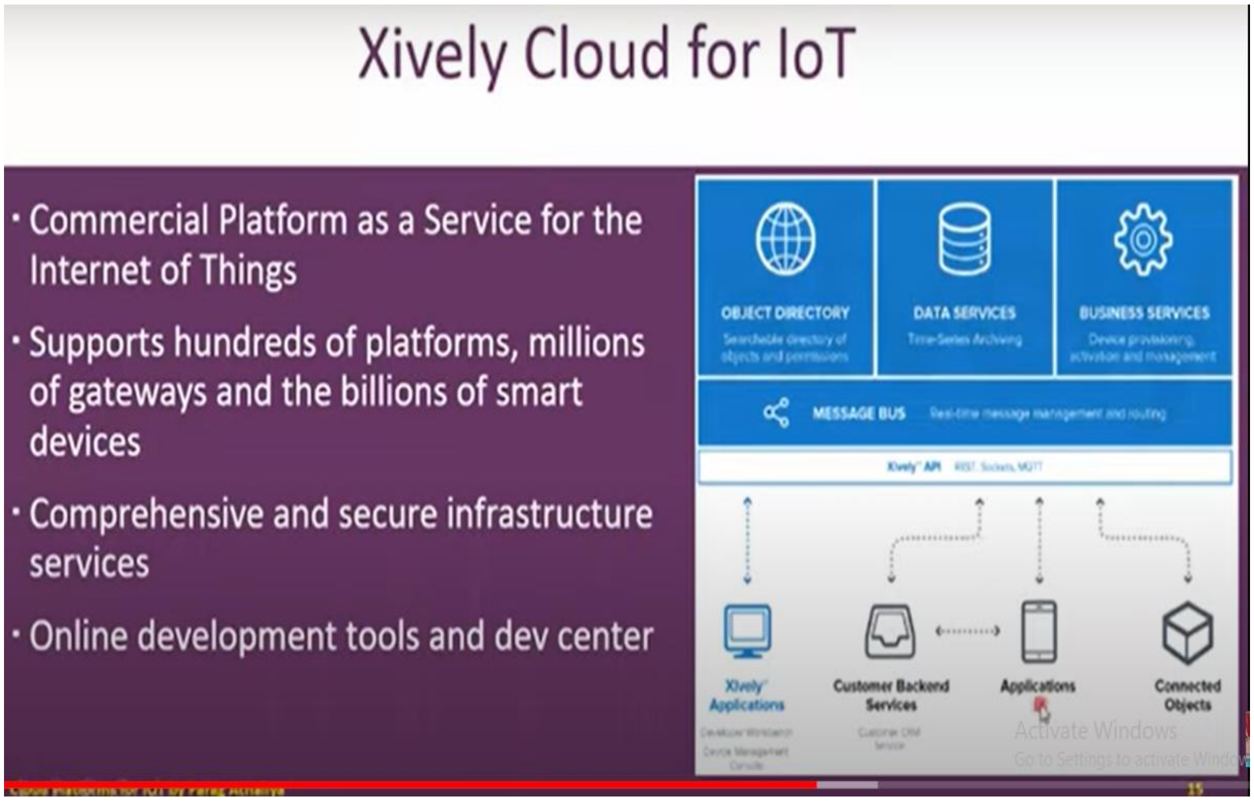
WAMP-AutoBahn for IoT

- In publish-subscribe model, **client** can be:
  - Publisher: Publisher publishes events to the topic maintained by the Broker
  - Subscriber: Subscriber subscribes to the topics and receives the events including payload
- In RPC model, **client** can be:
  - Caller: Issues calls to the remote procedures along with call arguments
  - Callee: Executes the procedures to which the calls are issued by the caller and returns the results back to the caller



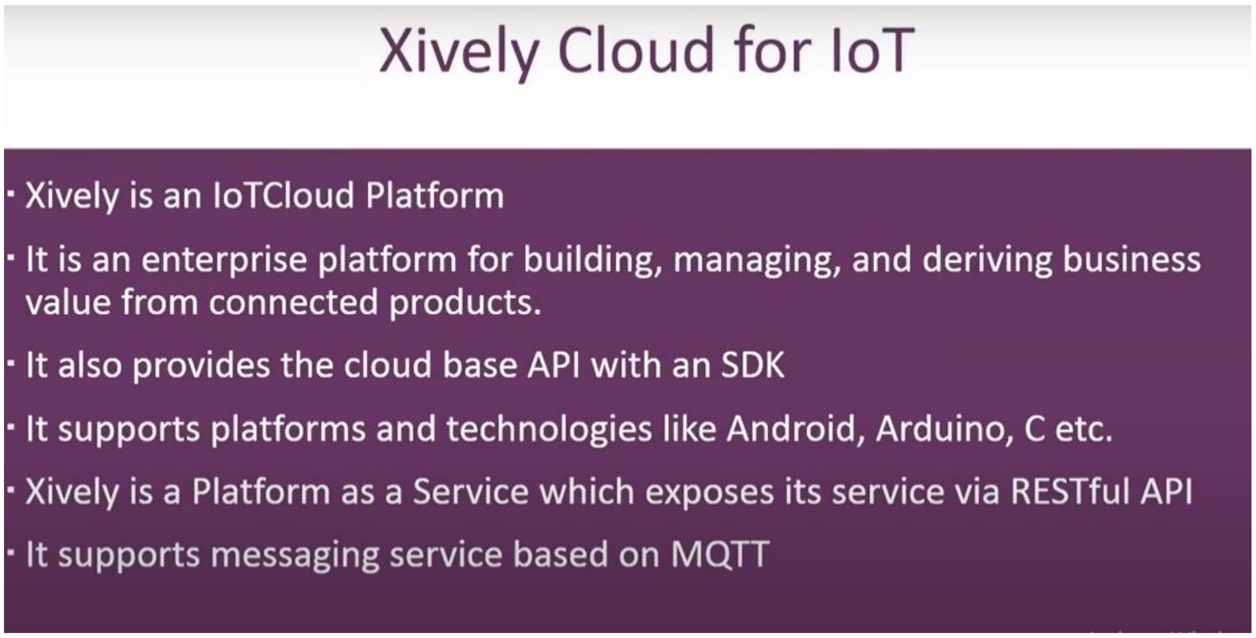
WAMP-AutoBahn for IoT

- Router: Routers are peers that perform generic call and event routing
- In publish-subscribe model, Router has role of Broker,
- Routes messages published to topic to all subscribers subscribed to the topic
- In RPC model Router has the role of a Broker:
  - Dealer: acts as router, routes RPC calls from Caller to Callee & routes results from Callee to Caller
- Application Code: Application code runs on the Clients (Publisher, Subscriber, Callee or Caller)



Xively Cloud for IoT

- Commercial Platform as a Service for the Internet of Things
- Supports hundreds of platforms, millions of gateways and the billions of smart devices
- Comprehensive and secure infrastructure services
- Online development tools and dev center



Xively Cloud for IoT

- Xively is an IoTCloud Platform
- It is an enterprise platform for building, managing, and deriving business value from connected products.
- It also provides the cloud base API with an SDK
- It supports platforms and technologies like Android, Arduino, C etc.
- Xively is a Platform as a Service which exposes its service via RESTful API
- It supports messaging service based on MQTT

Link for Above material:

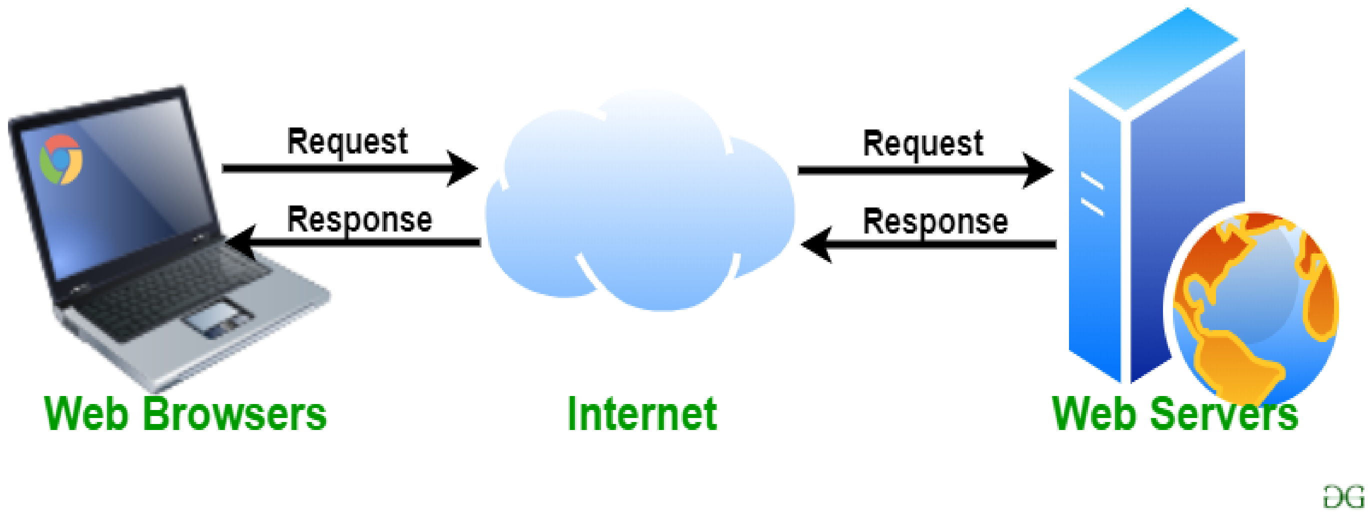
https://www.bing.com/videos/search?q=Introduction+to+Cloud+Storage+models+and+communication&qpvt=Introduction+to+Cloud+Storage+models+and+communication&view=detail&mid=828C9615C818DA3F278C828C9615C818DA3F278C&&FORM=VRDGAR

WEBSERVERS FOR IOT

Web Server: Web server is a program which processes the network requests of the users and serves them with files that create web pages. This exchange takes place using Hypertext Transfer Protocol (HTTP).

Basically, web servers are computers used to store HTTP files which make a website and when a client requests a certain website, it delivers the requested website to the client. For example, you want to open Facebook on your laptop and enter the URL in the search bar of google. Now, the laptop will send an HTTP request to view the facebook webpage to another computer known as the webserver. This computer (webserver) contains all the files (usually in HTTP format) which make up the website like text, images, gif files, etc. After processing the request, the webserver will send the requested website-related files to your computer and then you can reach the website.

Different websites can be stored on the same or different web servers but that doesn't affect the actual website that you are seeing in your computer. The web server can be any software or hardware but is usually software running on a computer. One web server can handle multiple users at any given time which is a necessity otherwise there had to be a web server for each user and considering the current world population, is nearly close to impossible. A web server is never disconnected from the internet because if it was, then it won't be able to receive any requests, and therefore cannot process them.



There are many web servers available in the market both free and paid. Some of them are described below:

- Apache HTTP server: It is the most popular web server and about 60 percent of the world's web server machines run this web server. The Apache HTTP web server was developed by the Apache Software Foundation. It is an open-source software which means that we can access and make changes to its code and mold it according to our preference. The Apache

Web Server can be installed and operated easily on almost all operating systems like Linux, MacOS, Windows, etc

.

- **Microsoft Internet Information Services (IIS):** IIS (Internet Information Services) is a high performing web server developed by Microsoft. It is strongly united with the operating system and is therefore relatively easier to administer. It is developed by Microsoft, it has a good customer support system which is easier to access if we encounter any issue with the server. It has all the features of the Apache HTTP Server except that it is not an open-source software and therefore its code is inaccessible which means that we cannot make changes in the code to suit our needs. It can be easily installed in any Windows device.
-
-

- **Lighttpd:** Lighttpd is pronounced as 'Lightly'. It currently runs about 0.1 percent of the world's websites. Lighttpd has a small CPU load and is therefore comparatively easier to run. It has a low memory footprint and hence in comparison to the other web servers, requires less memory space to run which is always an advantage. It also has speed optimizations which means that we can optimize or change its speed according to our requirements. It is an open-source software which means that we can access its code and add changes to it according to our needs and then upload our own module (the changed code).

- **Jigsaw Server:** Jigsaw has been written in the Java language and it can run CGI (common gateway interference) scripts as well as PHP programs. It is not a full-fledged server and was developed as an experimental server to demonstrate the new web protocols. It is an open-source software which means that we can access its code and add changes to it according to our needs and then upload our own module (the changed code). It can be installed on any device provided that the device supports Java language and modifications in Java

  .

- **Sun Java System:** The Sun Java System supports various languages, scripts, and technologies required for Web 2.0 such as Python, PHP, etc. It is not an open-source software and therefore its code is inaccessible which means that we cannot make changes in the code to suit our needs

UNIT V UAV IoT **Lecture 10Hrs**

Introduction toUnmanned Aerial Vehicles/Drones, Drone Types, Applications: Defense, Civil, Environmental Monitoring; UAV elements and sensors- Arms, motors, Electronic Speed Controller(ESC), GPS, IMU, Ultra sonic sensors; UAV Software –Arudpilot, Mission Planner, Internet of Drones(IoD)- Case study FlytBase.

Introduction to Unmanned Aerial Vehicles/Drones:

Drones or Unmanned Aerial Vehicles (UAVs) are one of the most potential devices that have multiple uses in the near future. As the Drone/ UAV Technology develop, they will also become cheaper, complex and multifunctional. It is for sure that they will become one of the most commonly seen devices in future. Following will discuss what are Drones/ Unmanned Aerial Vehicles (UAVs), its history, types, components, works, applications, advantages and disadvantages.

Table of Contents ☑

Link:https://electricalfundablog.com/drones-unmanned-aerial-vehicles-uavs/

What are Drones/ Unmanned Aerial Vehicles (UAVs)

Drones or Unmanned Aerial Vehicles (UAVs) are flying robotic machines that can be controlled remotely from a location without the need for the person to move with it. The control is possible through devices such as the transmitters, receivers and sensors.

Fig. 1 – Introduction to Drones/ Unmanned Aerial Vehicles (UAVs)

It can fly at great heights and also carry considerable weight along with it. A drone typically has a lot of uses ranging from in-air photography, surveillance, in the agricultural field to spray water, pesticides and survey of the fields for any leakages in the water connections and more.

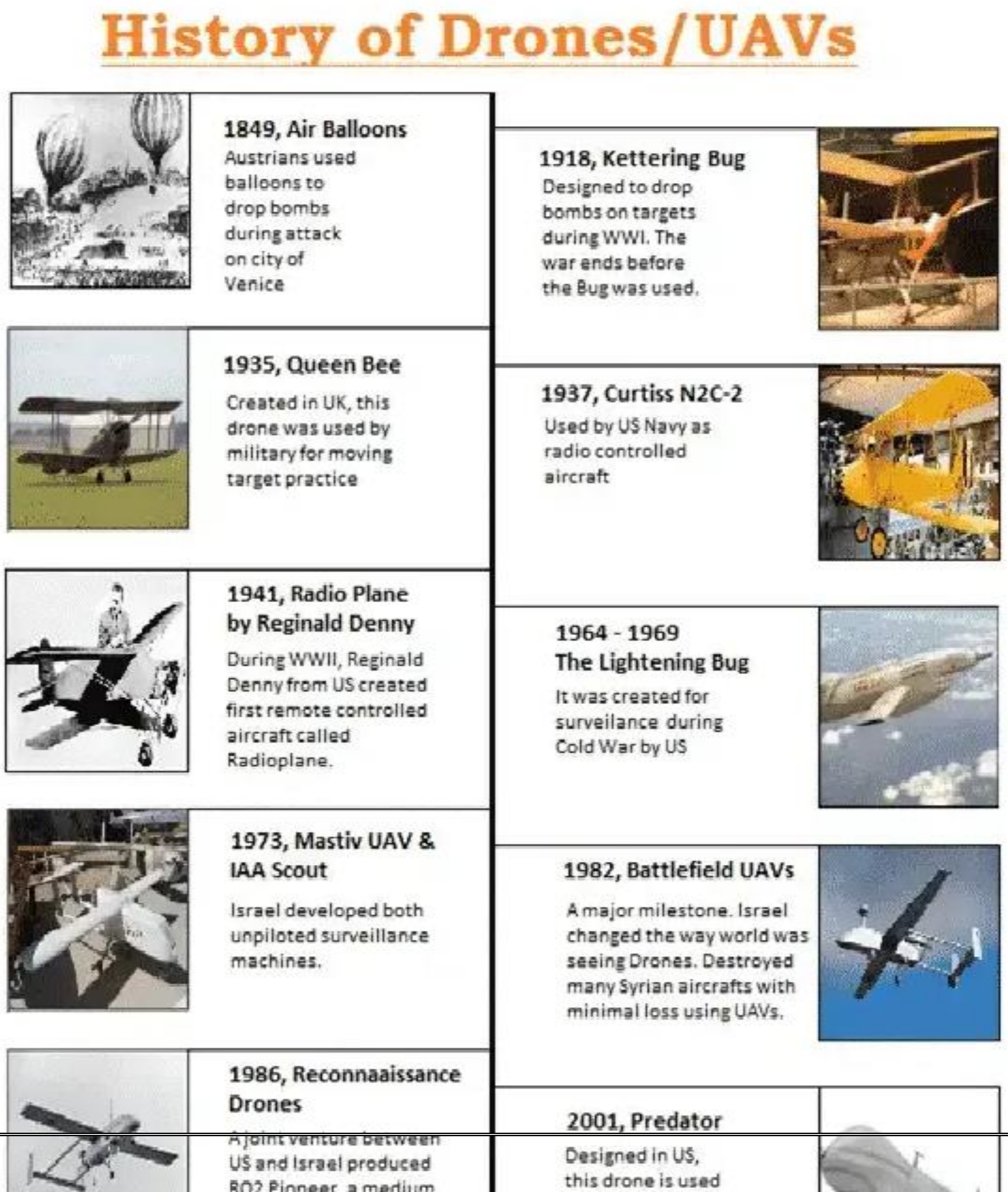
Today, it is possible to fly and control these Unmanned Aerial Vehicles with just a smartphone too. However, licenses are mandatory to fly them. The government strictly regulates the heights in which they can be flown, in which places it can fly and what purposes it can be used for.

History of Drones/ Unmanned Aerial Vehicles (UAVs)

The Unmanned Aerial Vehicles have been around us for centuries. The idea of drones was first conceived for the use by Military during war. The first documented use of a Drone/ UAV is of 1849 when Austrians attacked the city of

Venice using unmanned Air Balloons loaded with explosives. Then in 1915, they were used by British military to take aerial photographs of German movements in the air during Battle of Neuve Chapelle.

Later, United States came with 'Kettering Bug' in 1918 that was designed to drop bombs on targets. Then came Queen Bee and Curtiss N2C-2 in thirties, which were better than its predecessors. In 1941 during WW II, Reginald Denny from US made the first radio controlled aircraft called Radioplane OQ-2. It was the first drone to be mass produced ever. Further, in 1964, US created the Lightening Bug for secret surveillance during cold war. In 1973, Israel made a game changer entry in drone field with Mastiv UAV and IAA scout.



History of Drones/UAVs

1849, Air Balloons
Austrians used balloons to drop bombs during attack on city of Venice

1918, Kettering Bug
Designed to drop bombs on targets during WWI. The war ends before the Bug was used.

1935, Queen Bee
Created in UK, this drone was used by military for moving target practice

1937, Curtiss N2C-2
Used by US Navy as radio controlled aircraft

1941, Radio Plane by Reginald Denny
During WWII, Reginald Denny from US created first remote controlled aircraft called Radioplane.

1964 - 1969
The Lightening Bug
It was created for surveillance during Cold War by US

1973, Mastiv UAV & IAA Scout
Israel developed both unpiloted surveillance machines.

1982, Battlefield UAVs
A major milestone. Israel changed the way world was seeing Drones. Destroyed many Syrian aircrafts with minimal loss using UAVs.

1986, Reconnaaissance Drones
A joint venture between US and Israel produced RQ2 Pioneer, a medium

2001, Predator
Designed in US, this drone is used

During its war with Syria, Israel introduced Unmanned Aerial Vehicles into active war field to destroy many Syrian aircrafts with minimal loss. Until now, UAVs were considered unreliable. However, Israel dramatic win changed this perception forever. Very soon, US and Israel created RQ2 Pioneer in 1986, as a joint venture with an aim to design less expensive and effective drones. RQ2 Pioneer was a medium sized reconnaissance aircraft.

In 2001, the U.S introduced a new generation predator drone that was packed with state of the art surveillance equipment. In 2003, the UAVs paved its way in construction and 'Search & Rescue' field and was no longer limited to military use only. They, post introduction to civil world, have evolved multi-fold thereafter.

Fig. 3 – Use of Drones for Providing Internet by Facebook

In 2014, Amazon announced its plan to use drone technology in product delivery known as 'Prime Air' so that parcels can be delivered by air with ease. One more interesting concept is presented by Facebook as they are thinking to develop some giant Unmanned Aerial Vehicles that can carry signal to remote locations for direct internet access.

Types of Drones/ UAVs
Based on the type of aerial platform used, there are 4 major types of UAVs.

- Multi Rotor Drone
- Fixed Wing Drone
- Single Rotor Drone
- Fixed Wing Hybrid VTOL

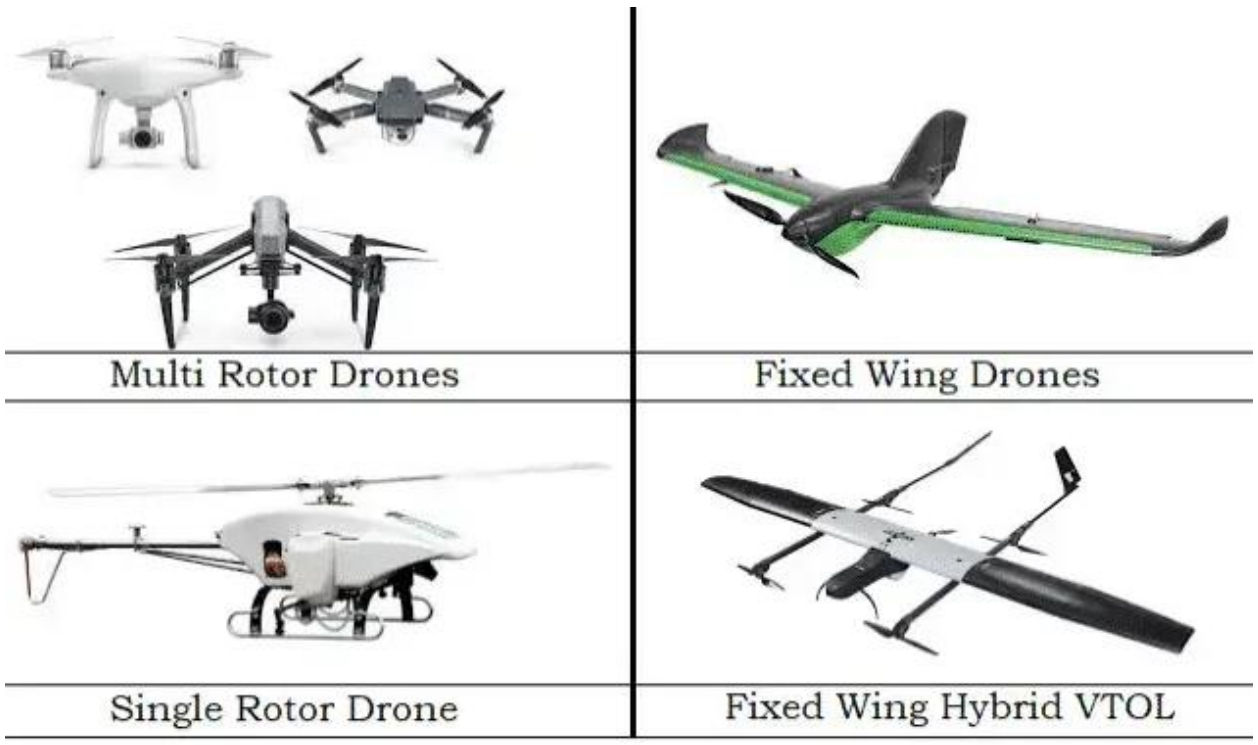


Fig. 4 – Types of Drones/ Unmanned Aerial Vehicles (UAVs)

Multi Rotor Drone

Multi Rotor Drone uses multiple propellers to navigate and fly. They are utilized for common uses such as photography and video surveillance. They are the most common of all UAVs.

They can be further classified in four most popular categories based on the number of propellers being used. They are:

- **Tricopter** – Three Propeller Drones
- **Quadcopter** – Four Propeller Drones
- **Hexacopter** – Six Propeller Drones
- **Octocopter** – Eight Propeller Drones

Fixed Wing Drones

Fixed Wing drones have wings in place of propellers just like an airplane. They cannot hover at one place. They fly on the set course till their energy source is functional.

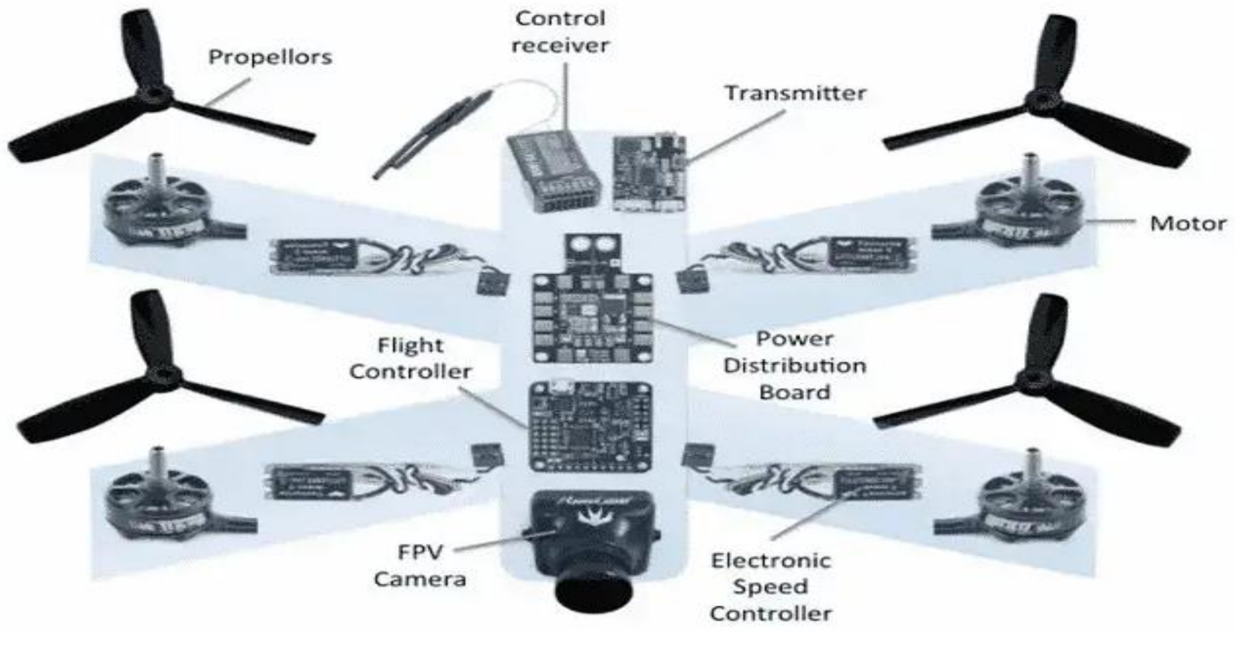
Single Rotor Drones

As the name suggests, a Single rotor drone has only one rotor and a small tail to control it direction. They resemble more like a helicopter and are very energy efficient.

Fixed Wing Hybrid VTOL
VTOL stands for Vertical take Off & Landing. Fixed Wing Hybrid VTOLs uses propeller(s) to lift off and wings for gliding.

Main Components of Drones/ UAVs (Quadcopter)
There are multiple designs used for drones, the most popular is the four-wing structure called Quadcopter. The main components of a Quadcopter are:

1. Propellers/ Wings
2. Chassis
3. DC motors
4. Flight Controller
5. Electronic Speed Controllers
6. Landing Gear
7. Transmitter
8. Receiver
9. GPS Module
10. Battery
11. Camera



1. Propellers/ Wings

The Drones/ UAVs use propellers/ wings or both (depending on the availability) to direct them. Propeller driven Drones have two types of propellers onboard for direction and thrust. These are

- Standard Propellers
- Pusher Propellers

Standard Propellers

These are located in the front of Quadcopter. These propellers provide direction to the Drone.

Pusher Propellers

These are located in the back of Quadcopter. These propellers provide forward and backward thrust to the Drone.

2. Chassis

This is the main body of Quadcopter which provides the housing facility to all other components.

3. Flight Controller

The flight controller is usually referred to as the brain of the drone. The flight controller controls the power supply to the electronic speed controller. It is also used to detect orientation changes in the drone. It controls the motors and ensures that the drone is in air.

4. Electronic Speed Controller

Electronic speed controllers (ESC) are the electronic circuits that regulate the speed of the DC motors. It also provides dynamic braking and reversing options.

5. DC Motors

To ensure that the drone is airborne for a good amount of time, we need high torque motors. The high torque also helps to change the speed of the propellers. Brushless DC motors are preferred as they are lighter than the brushed ones.

6. Landing Gear

Landing gears are not required for small drones. However, bigger drones need a landing gear to avoid any damage while landing. The requirement of landing gear varies with functionality of the drone. For example – Delivery drones which carry parcels require a spacious landing gear as they need space to hold the items.

7. Transmitter

The transmitter sends signals from controller to the drone to generate command of direction and thrust.

8. Receiver

The receiver receives the signals sent by the transmitter and passes it to Flight Controller PCB.

9. GPS Module

The GPS module provides the navigational data (longitude, latitude and elevation) to the Controller. This module assists the controller in recognizing the taken path and safely return to the initial point in case of lost connection.

10. Battery

It provides power to the drone. Generally, rechargeable battery is used in drone.

11. Camera

There is normally an attached inbuilt camera with drones. If the drone is not provided with inbuilt camera, then it will have a provision of detachable camera.

How Drones/ UAVs Work

When the controller provides the lift off instruction to the receiver of drone, it forwards it to the Flight Controller Board, which in turn actuates the propeller. The propellers provide the lift off thrust and the drone becomes airborne. It is these propellers, which provide the direction as well as speed to the drone.

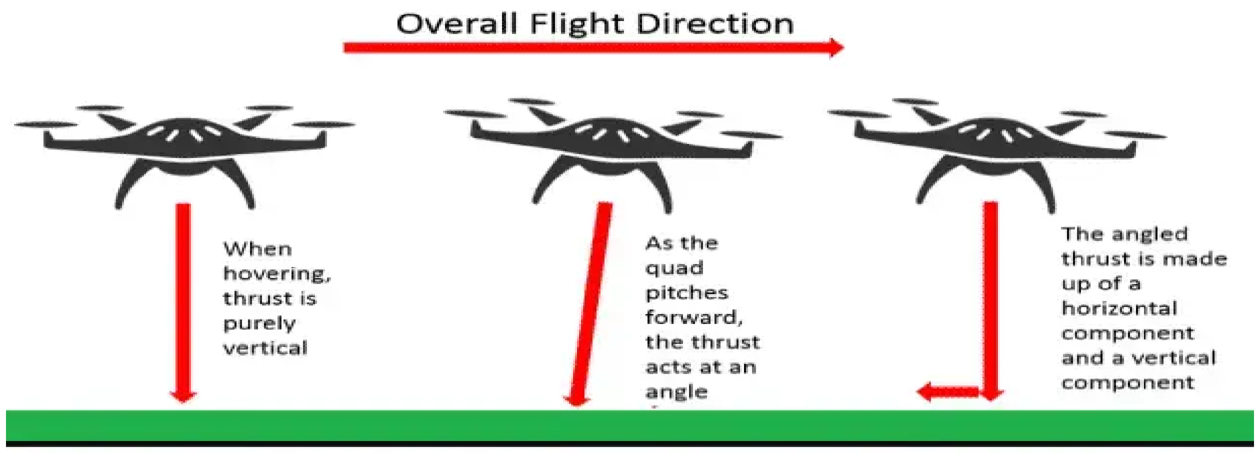

Overall Flight Direction
When hovering, thrust is purely vertical
As the quad pitches forward, the thrust acts at an angle
The angled thrust is made up of a horizontal component and a vertical component

Fig. 6 – How Drones/ UAVs Work

The Drones have a complex multi propeller system that assists in reduction of failures. The propellers work in group and even if any one of them fails, the drone is capable to get support form its available propellers and navigate/ land safely. The GPS module helps the drone to identify its travelled path and get real time data such as location, speed and elevation.

Applications of Drones/ UAVs

They have a variety of functionalities in different industries. A few of the most popular industries that use them are:

- Military
- Space Research
- Product Delivery
- Agriculture
- Search and Rescue
- Internet Carrier

Applications of Drones/ UAVs in Military

The first drones were used during wars to drop bombs without the requirement of a pilot to carry them into the enemy zone. It is also used for surveillance, reconnaissance, aerial photography, specific target tracking, and counter air threats. It reduces the risk factor involved during such missions due to minimal involvement of human personnel.

Applications of Drones/ UAVs in Space Research

The United States and the United Kingdom have been the pioneers in the use of drones for space research. NASA’s most popular drone X-37B has made multiple visits to space for a number of classified missions. Following NASA’s route, other space agencies from across the globe are also using drones for their space researches.

Applications of Drones/ UAVs in Food and Product Delivery

The food and product delivery industry is one of the most booming sectors today. The most challenging factor that is considered in this industry is the problem of delivering food through roads and delivery agents. Drones can be a direct profit-making solution for this. In fact, a few countries have started using this technology.

Applications of Drones/ UAVs in Agriculture

One of the most strenuous jobs in the world is agriculture. Growing crops in the vast lands is a tedious job without a doubt. If this could be done in a manner with reduced human efforts, it is one of the best things that can be explored in this century. Drones have become really popular in this sector. A few of the functions for which drones are used are spraying water, pesticides, and doing video surveillance in the vast field.

Applications of Drones/ UAVs in Search & Rescue (SAR)

Drones is very useful in Search & Rescue (SAR) operations as they can even go in areas with difficult access. They can also patrol larger areas as compared to an individual. Further, with the addition of thermal imaging devices on it, the identification of people in distress is much more faster and reliable.

Applications of Drones/ UAVs as Internet Carriers

As stated earlier, Google and Facebook are already working on concept of some giant drones that can carry signal to remote locations for direct internet access.

Advantages of Drones/ UAVs

The advantages of Drones/ UAVs include:

- Quality air Imaging.
- Accessibility to hard to reach areas.
- Reduced human effort and risks.
- Need minimal launch time.
- Precise Operation.
- Reliable.
- Saves time.

Disadvantages of Drones/ UAVs

The disadvantages of Drones/ UAVs include:

- Privacy becomes vulnerable if wrongly used.
- Safety of people/ equipment depends on perator skills
- Laws for Drones are not so specific yet. Still Evolving.
- Shorter Lifespan
- Can be easily hacked