Dr. Siddhartha Choubey
Professor, Computer Science and Engineering
Shri Shankaracharya Technical Campus, Bhilai

Prof. Sonu Agrawal
Associate Professor, Computer Science and Engineering
Shri Shankaracharya Technical Campus, Bhilai

Chhattisgarh Swami Vivekananda Technical University, Bhilai (C.G.)

Program / Semester: **B.Tech (VI)**	Branch: **Computer Science &Engineering**
Subject: **Internet Of Things (Professional Elective – II)**	Course Code: **C022632(022)**
T**otal / Minimum-Pass Marks** (End Semester Exam): **100 / 35**	L: 2 T: 01 P: 0 Credits: 2
Class Tests & Assignments to be conducted: 2 each	Duration (End Semester Exam): **03 Hours**

Course Objectives

- To understand Concepts, design and characteristics of IoT.
- To understand Architecture of IoT.
- To understand basic protocols of IoTs.
- To understand challenges and applications of IoTs.
- To develop IoT applications using Tools.

Course Outcomes

- Students will familiar with the concepts of Internet of Things.
- Students will familiar with IoT Architecture
- Students will ready to Analyze basic protocols in wireless sensor network
- Students will be capable to design IoT applications in different domain and be able to analyze their performance
- Capable to implement basic IoT applications on embedded platform

Unit-I	**Introduction to Internet of Things:** Origin of Terminology IoT, Applications, Characteristics, Components of IoT, Associated technologies with IoT (M2M, Big Data, Cloud, Smart Grid, IoV, CPS, SDN, 3G/4G/5G), Challenges in IoT.
Unit-II	**Connectivity:** IoT Network Configurations, Gateway Prefix Allotment, IPv4, IPv6, IPv4 versus IPv6, RPL **Data Protocol:** MQTT, CoAP, AMQP, **Communication Protocols:** IEEE 802.15.4, ZWave, Bluetooth, ZigBee, 6LowPAN, HART and Wireless HART, NFC, RFID.
Unit-III	**Actuation:** Actuator, Actuator Types: Hydraulic Pneumatic, Electrical, Thermal/ Magnetic Mechanical, Soft Actuators, Shape memory polymer (SMP) **Types of Motor Actuators**: Servo motor, Stepper motor, Hydraulic motor, Solenoid Relay, AC motor **Sensing:** Definition, Types of sensors, Transducers, Sensors Classes
Unit-IV	**Introduction to Arduino Programming:** Operators in Arduino, Control Statement, Loops, Integration of Sensors and Actuators with Arduino. **Implementation of IoT**: Interoperability in IoT, Introduction to NodeMCU (ESP8266), Connectivity of Sensors and Actuators with NodeMCU, Introduction to Python programming, Introduction to Raspberry PI.
Unit-V	**Cloud Computing Fundamentals:** Recent Trends in Computing, Evolution of Cloud Computing, Evolution of Cloud Computing, Business Advantages, Components **Service Models:** Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) Infrastructure-as-a-Service (IaaS), Multi-cloud, Inter-cloud, Cloud Computing Service Management and Security, **Case studies:** Amazon Elastic Compute Cloud (EC2), Microsoft Azure.

UNIT-I

Introduction to Internet of Things: Origin of Terminology IoT

WHAT IS IOT?

Internet of Things (IoT) is an interconnected system of physical devices embedded with software and sensors to streamline information across the network via the internet. These IoT networks have shown exceptional data collection, analysis, reporting, and prediction behavior for integration into future planning. Internet of Things (IoT) is a network of physical objects or people called "things" that are embedded with software, electronics, network and sensors that allows these objects to collect and exchange data. The goal of IoT is to extend to internet connectivity from standard devices like computer, mobile, tablet to relatively dumb devices like a toaster.

Though IoT is a trending technology with unlimited future potential yet, there are no single or a particular set of rules for the components of IoT. Various organizations have adopted features of IoT in their premises as several benefits have emerged from their integrations. IoT makes virtually everything "smart," by improving aspects of our life with the power of data collection, AI algorithm, and networks. The thing in IoT can also be a person with a diabetes monitor implant, an animal with tracking devices, etc.

HISTORY OF IOT

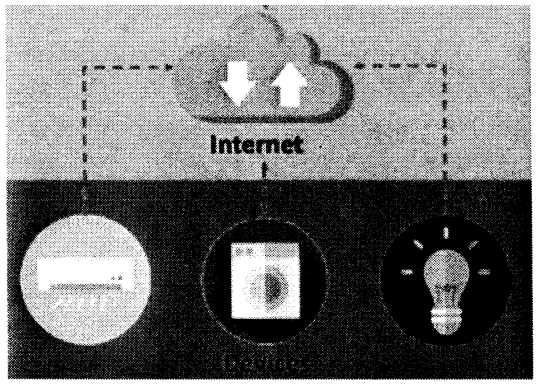
- 1970- The actual idea of connected devices was proposed. John Romkey created a toaster which could be turned on/off over the Internet
- 1995- Siemens introduced the first cellular module built for M2M
- 1999- The term "Internet of Things" was used by Kevin Ashton during his work at P&G which became widely accepted.
- 2004 – The term was mentioned in famous publications like the Guardian, Boston Globe, and Scientific American.
- 2005-UN's International Telecommunications Union (ITU) published its first report on this topic.
- 2008- The Internet of Things was born
- 2011- Gartner, the market research company, include "The Internet of Things" technology in their research.

ELEMENTS OF IOT

There are many technological aspects, layers, applications, and Components of IoT networks. But most of these IoT architectures are built upon core fundamentals.

1. Smarter Devices in a different form
2. Network and Gateway that allows devices to be part of the IoT
3. Middleware that includes data storage spaces and advances predicting capabilities
4. End-user applications

HOW IOT WORKS?



The entire IoT process starts with the devices themselves like smart phones, smart watches, electronic appliances like TV, Washing Machine which helps you to communicate with the IoT platform.

1) Sensors/Devices: Sensors or devices are a key component that helps you to collect live data from the surrounding environment. All this data may have various levels of complexities. It could be a simple temperature monitoring sensor, or it may be in the form of the video feed. A device may have various types of sensors which performs multiple tasks apart from sensing. Example, A mobile phone is a device which has multiple sensors like GPS, camera but your smart phone is not able to sense these things.to write a TEST CASE Software Testing Tutorial

2) Connectivity: All the collected data is sent to a cloud infrastructure. The sensors should be connected to the cloud using various mediums of communications. These communication mediums include mobile or satellite networks, Bluetooth, WI-FI, WAN, etc.

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

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

Advantages of IoT

Key benefits of IoT technology are as follows:

1. **Technical Optimization:** IoT technology helps a lot in improving technologies and making them better. Example, with IoT, a manufacturer is able to collect data from various car sensors. The manufacturer analyzes them to improve its design and make them more efficient.
2. **Improved Data Collection:** Traditional data collection has its limitations and its design for passive use. IoT facilitates immediate action on data.
3. **Reduced Waste:** IoT offers real-time information leading to effective decision making & management of resources. For example, if a manufacturer finds an issue in multiple car engines, he can track the manufacturing plan of those engines and solves this issue with the manufacturing belt.
4. **Improved Customer Engagement:** IoT allows you to improve customer experience by detecting problems and improving the process.

Disadvantages IoT

Now, let's see some of the disadvantages of IoT in this Internet of Things tutorial:

1. **Security:** IoT technology creates an ecosystem of connected devices. However, during this process, the system may offer little authentication control despite sufficient security measures.
2. **Privacy:** The use of IoT, exposes a substantial amount of personal data, in extreme detail, without the user's active participation. This creates lots of privacy issues.
3. **Flexibility:** There is a huge concern regarding the flexibility of an IoT system. It is mainly regarding integrating with another system as there are many diverse systems involved in the process.

4. **Complexity:** The design of the IoT system is also quite complicated. Moreover, it's deployment and maintenance also not very easy.
5. **Compliance:** IoT has its own set of rules and regulations. However, because of its complexity, the task of compliance is quite challenging.

CHALLENGES OF INTERNET OF THINGS (IOT)

At present IoT is faced with many challenges, such as:

- Insufficient testing and updating
- Concern regarding data security and privacy
- Software complexity
- Data volumes and interpretation
- Integration with AI and automation
- Devices require a constant power supply which is difficult
- Interaction and short-range communication

We will now learn about Best practices for IoT.

- Design products for reliability and security
- Use strong authentication and security protocols
- Disable non-essential services
- Ensure Internet-managed, and IoT management hubs & services are secured
- Energy efficient algorithms should be designed for the system to be active longer.

TOP APPLICATIONS OF IoT

1. Smart Homes : One of the best and the most practical applications of IoT, smart homes really take both, convenience and home security, to the next level. Though there are different levels at which IoT is applied for smart homes, the best is the one that blends intelligent utility systems and entertainment together. For instance, your electricity meter with an IoT device giving you insights into your everyday energy usage, your set-top box that allows you to record shows from remote, Automatic Illumination Systems, Advanced Locking Systems, Connected Surveillance Systems all fit into this concept of smart homes. As IoT evolves, we can be sure that most of the devices will become smarter, enabling enhanced home security.

2. Smart City : Not just internet access to people in a city but to the devices in it as well – that's what smart cities are supposed to be made of. Efforts are being made to incorporate connected technology into infrastructural requirements and some vital concerns like Traffic Management, Waste Management, Water Distribution,

Electricity Management, and more. All these work towards eliminating some day-to-day challenges faced by people and bring in added convenience.

3. Self-driven Cars / Connected car **:** We've seen a lot about self-driven cars. Google tried it out, Tesla tested it, and even Uber came up with a version of self-driven cars. Since it's human lives on the roads that we're dealing with, we need to ensure the technology has all that it takes to ensure better safety for the passenger and those on the roads. The cars use several sensors and embedded systems connected to the Cloud and the internet to keep generating data and sending them to the Cloud for informed decision-making through Machine Learning. Though it will take a few more years for the technology to evolve completely and for countries to amend laws and policies.

4. IoT Retail Shops: Proximity-based advertising as a subset of smart retail is starting to take off. Perhaps this is the best use of the technology in bridging the gap between an online store and a retail store. The retail store allows you to go cashless by deducting money from your Amazon wallet. It also adds items to your cart in real-time when you pick products from the shelves. If you change your mind and pick up another article, the previous one gets deleted and replaces your cart with the new item. The best part of the concept store is that there is no cashier to bill your products. You don't have to stand in line but just step out after you pick up your products from shelves.

5. Farming: Farming is one sector that will benefit the most from the Internet of Things. With so many developments happening on tools farmers can use for agriculture, the future is sure promising. Tools are being developed for Drip Irrigation, understanding crop patterns, Water Distribution, drones for Farm Surveillance, and more. These will allow farmers to come up with a more productive yield and take care of the concerns better.

6. Wearables: Wearables remain a hot topic in the market, even today. These devices serve a wide range of purposes ranging from medical, wellness to fitness.

7. Smart Grids: One of the many useful IoT examples, a smart grid, is a holistic solution that applies an extensive range of Information Technology resources that enable existing and new gridlines to reduce electricity waste and cost. A future smart grid improves the efficiency, reliability, and economics of electricity.

8. Industrial Internet: The Industrial Internet of Things consists of interconnected sensors, instruments, and other devices connected with computers' industrial applications like manufacturing, energy management, etc. market researches like Gartner, Cisco, etc., believe the industrial internet to have the highest overall potential.

9. Tele health: Tele health, or Telemedicine, hasn't completely flourished yet. IoT Examples of Telemedicine include the digital communication of Medical Imaging, Remote Medical Diagnosis & Evaluations, Video Consultations with Specialists, etc.

10. Smart Supply-chain Management: Supply-chains have stuck around in the market for a while now. A common example can be Solutions for tracking goods while they are on the road. Backed with IoT technology, they are sure to stay in the market for the long run.

11. Traffic Management : Car traffic management in large cities can be greatly improved with the help of the Internet of Things (IoT). The Internet of Things helps us stay informed and improves traffic monitoring by allowing us to use our mobile phones as sensors to collect and share data from our vehicles through apps like Waze or Google Maps. This feeds and improves the data on the various routes to the same destination, distance, and estimated arrival time. Analysis of traffic patterns over a long period is another IoT application. It provides an idea of what might happen during peak hours. Commuters will be better prepared to avoid traffic and delays by being made aware of possible alternatives.

12. Water/ Waste Management : Many cities are adopting water recycling using water treatment units. Using an IoT application, you can see how much waste water is being produced, how much is being consumed in a specific area, and how waste production is changing over time. With a smart waste management system, authorities will be able to predict how much waste will be generated in a specific location, how to properly process it, when to clear it, and how to analyze data for future planning, among other things. This data will be used to plan the city's expansion and upgrade projects. Smart analytics solutions can be used to manage waste collection and treatment fleets, as well as to predict future trends.

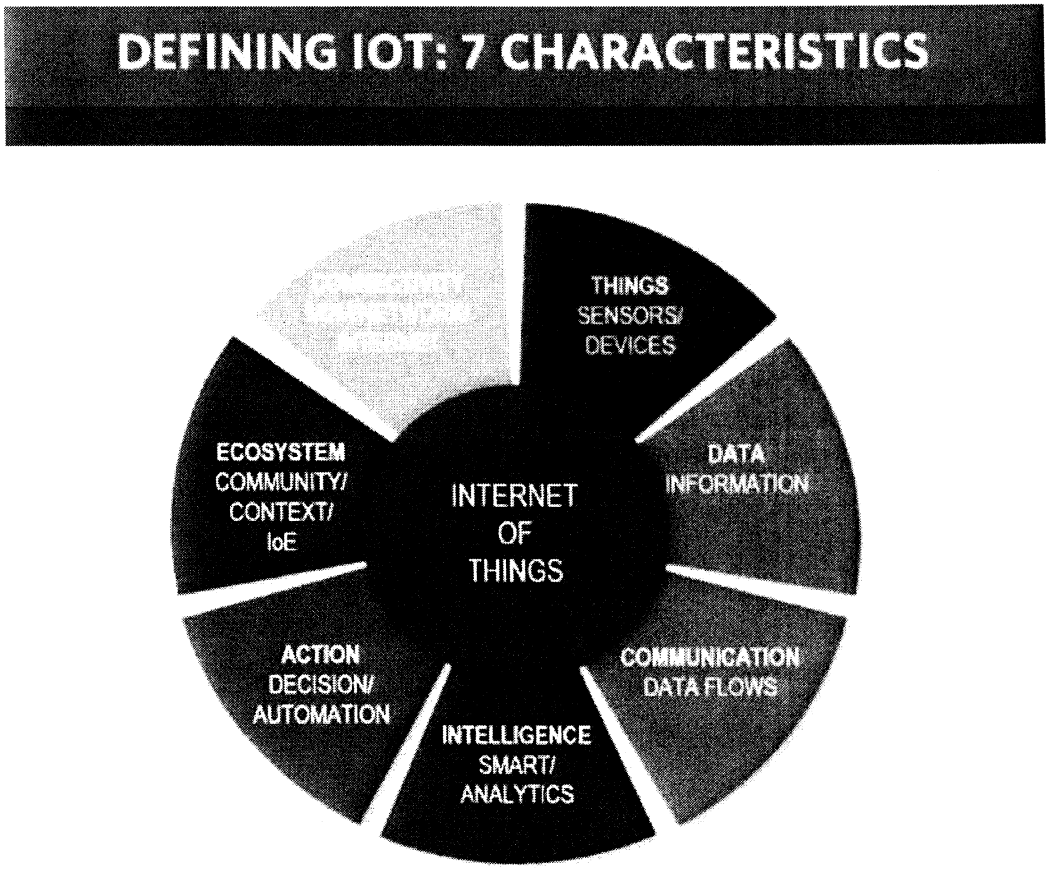
IoT solutions are widely used in numerous companies across various industries. Some most common IoT applications are given below:

Application type	Description
Smart Thermostats	Helps you to save resources on heating bills by knowing your usage patterns.
Connected Cars	IoT helps automobile companies handle billing, parking, insurance, and other related stuff automatically.
Activity Trackers	It helps you to capture heart rate patterns, calorie expenditure, activity levels, and skin temperature on your wrist.
Smart Outlets	Remotely turn any device on or off. It also allows you to track a device's energy level and get custom notifications directly into your smartphone.

Application type	Description
Parking Sensors	IoT technology helps users to identify the real-time availability of parking spaces on their phones.
Connected Health	The concept of a connected health care system facilitates real-time health monitoring and patient care. It helps in improved medical decision-making based on patient data.
Smart City	The smart city offers all types of use cases, which include traffic management to water distribution, waste management, etc.
Smart home	Smart home encapsulates the connectivity inside your homes. It includes smoke detectors, home appliances, light bulbs, windows, door locks, etc.
Smart supply chain	Helps you in real time tracking of goods while they are on the road or getting suppliers to exchange inventory information.

Characterstics

One can define IoT by looking at the various characteristics in the broader context. further below is an overview with some of these IoT definitions



IoT using 7 characteristics

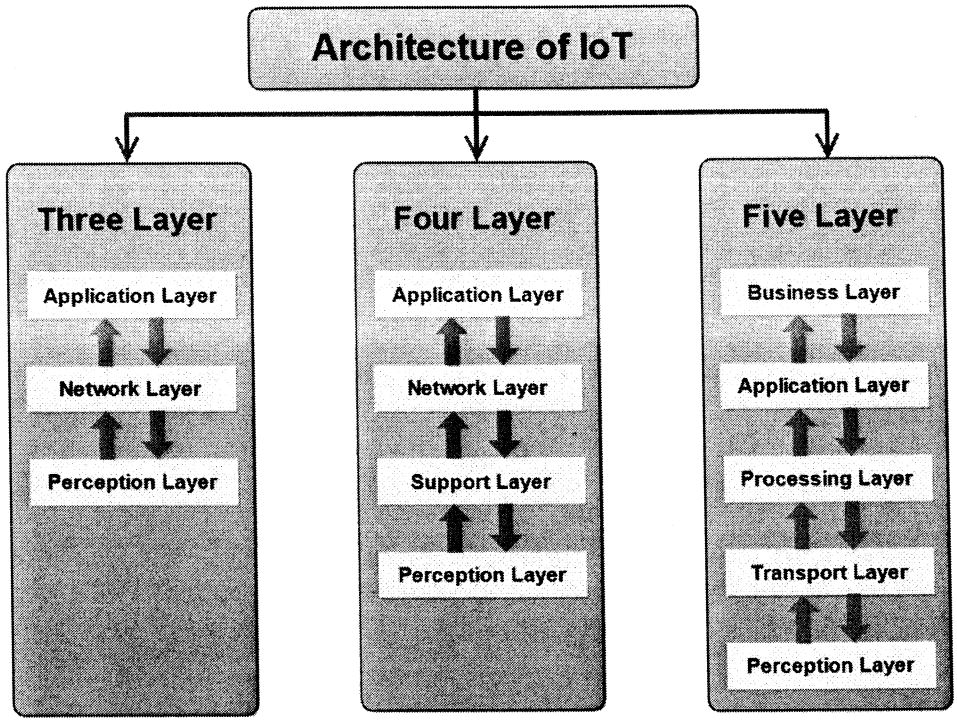
1. **Connectivity.** This doesn't need too much explanation. With everything going on in IoT devices, hardware with sensors and other electronics and control systems there needs to be a connection between various levels.
2. **Things**. Anything that can be tagged or connected as such as it's designed to be connected. From sensors and household appliances to tagged livestock. Devices can contain sensors or sensing materials can be attached to existing devices and items to make it a part of IoT.
3. **Data**. Data is the glue of the Internet of Things, the first step towards action and intelligence.
4. **Communication**. Devices get connected so they can communicate data and this data can be analyzed. Communication can occur over short distances or over a long range to very long range.
5. **Intelligence**. The aspect of intelligence as in the sensing capabilities in IoT devices and the intelligence gathered from big data analytics also artificial intelligence.
6. **Action**. The consequence of intelligence. This can be manual action based upon data or automation, often the most important piece.
7. **Ecosystem**. The place of the Internet of Things from a perspective of other technologies, communities, goals and the picture in which the Internet of Things fits.

Architecture Of Internet Of Things (IoT) Connectivity Layers,

Depending upon different application areas of Internet of Things, it works accordingly as per it has been designed/developed. But it has not a standard defined architecture of working which is strictly followed universally. The architecture of IoT depends upon its functionality and implementation in different sectors. Still, there is a basic process flow based on which IoT is built. Elements of IoT define the fundamentals of almost every IoT system on the globe. Still, they are divided into multiple architecture layers to further refine the overall IoT network.

These IoT layers are :

1. **Perception Layer** that manages smart devices across the system.
2. **Connectivity/Transport Layer** allows transferring data from the cloud to devices and vice-versa, different aspects of gateways and networks.
3. **Processing Layer** that controls and manages IoT levels for streamlining data across the system.
4. **Application Layer** that aids in the procedures of analytics, device control, and reporting to end-users.



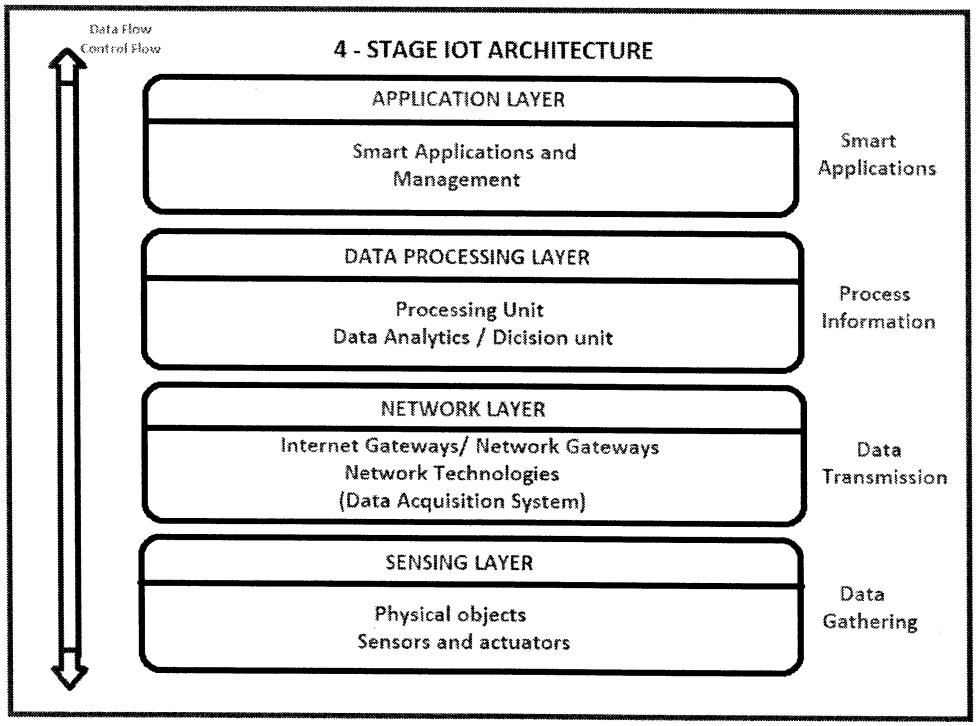
With continuous changes in the IT environment, many organizations have added three additional layers to their infrastructure. Block diagram showing the various stages of IoT architecture layers.

5. **Business layer** that derives information and decision-making analysis from data.
6. **Security Layer** that covers all aspects of protecting the whole IoT architecture
7. **Edge Computing** Layer that works at an edge or near the device information collection.

Basic fundamental architecture of IoT i.e. 4 Stage IoT architecture.

So, from the diagram it is clear that there is 4 layers are present that can be divided as follows:

1. **Sensing Layer -** Sensors, actuators, devices are present in this Sensing layer. These Sensors or Actuators accepts data (physical/environmental parameters), processes data and sends data over network.
2. **Network Layer -** Internet/Network gateways, Data Acquisition System (DAS) are present in this layer. DAS performs data aggregation and conversion function (Collecting data and aggregating data then converting analog data of sensors to digital data etc). Advanced gateways which mainly opens up connection between Sensor networks and Internet also performs many basic gateway functionalities like malware protection, and filtering also sometimes decision making based on inputted data and data management services, etc.



3. **Data processing Layer -**This is processing unit of IoT ecosystem. Here data is analyzed and pre-processed before sending it to data center from where data is accessed by software applications often termed as business applications where data is monitored and managed and further actions are also prepared.
4. **Application Layer -** This is last layer of 4 stages of IoT architecture. Data centers or cloud is management stage of data where data is managed and is used by end-user applications like agriculture, health care, aerospace, farming, defense, etc.

Detail Architecture of IoT i.e. 7 Stage IoT architecture.

1. PERCEPTION LAYER

This IoT layers form the components of the internet physical design of IoT, acting as a medium between the digital and real world. In the IoT architecture layers, this perception layer has the main function of transforming analog signals into the digital form and vice versa. These come in a different multitude of shape and sizes with:

- **Sensors:** They are very small devices or systems built to understand and detect the change in their environment and further streamline information to their system. Generally, these sensors are quite small and take even less power to perform their task. Sensors have the unique ability to detect physical parameters such as humidity or temperature, then transforming them into electronic signals.
- **Actuators:** These represent a part of the machine that allows an electrical signal to be transformed into physical actions. These Actuators play a crucial role as components of IoT networks. Motors, clutch, gears and switches that brings any machines into actions.
- **Machine and Devices:** They are the main devices that have actuators and sensors. In IoT architecture, there is no limitation of location or distance between two or more devices that can be spread across the globe.

2. CONNECTIVITY LAYER

In the Connectivity layer, communication takes center stage between the physical layer of devices and IoT architecture. This communication takes place via two methods; First directly by either TCP or UDP/IP stack; Second, gateways act as a link between Local Area Network (LAN) and Wide Area Network (WAN), thus providing a path for information to pass through multiple protocols.Several network technologies are integrated across IoT systems that include:

- **WiFi,** the most popular and versatile technique used across data-driven technologies. WiFi modems are suitable for Smart homes, personal offices, and even corporate offices for seamless communication between LAN and WAN, respectively.
- **Ethernet** represents the hardware that supports fixed or permanent devices such as video cameras, gaming consoles, and security installations.
- **Bluetooth** is another widely used technology suited mainly for communication between devices within a short range. A perfect example would be headphones that can work on small power and simultaneously share fewer data over the network.

- **NFC (Near Field Communications)** allows communication between a very short distance of 4 inches or less.

- **LPWAN (Low Power Wide Area Network),** designed and built to match the IoT usage across long distances. These low-power WAN devices can last as much as 10+ years while consuming low power throughout. However, it can send signals to give precise information over a long periodic duration. These include devices for smart buildings, smart fields, smart cities, etc.

- **ZigBee** is another advanced wireless networking technology that consumes low power and can offer small data-sharing ability. One of the unique features of IoT is its capability to handle up to 65,000 nodes in its premises. ZigBee is built with the main focus for home automation and also has shown remarkable success for medical, scientific, and industrial protocols.

- **Cellular networks** are ideally suited for communication on a global scale with more trust and reliability. For IoT, there are two broad IoT levels of the cellular network as:
  1. LTE-M is Long Term Evolution for Machines that provides a very high-speed exchange of data and smooth direct cloud communication.
  2. NB-IoT as Narrowband that offers small data exchange using low-frequency channels respectively.

There are also messaging protocols present in the IoT system that allows seamless data sharing. Here is a list of top protocols present in the IoT architecture layers as of now.

- **Data Distribution Service (DDS)** represents a machine-to-machine real-time messaging framework in IoT systems.
- **Advanced Message Queuing Protocol (AMQP)** provides server protocols for servers via peer-to-peer data exchange.
- **Constrained Application Protocol (CoAP)** defines the protocols for constrained devices that use low power and low memory, such as wireless sensors.
- **Message Queue Telemetry Transport (MQTT)** represents the messaging protocol standards for low-powered devices using TCP/IP for seamless data communication.

3. EDGE LAYER

In the early stages, with IoT networks gaining size and numbers, latency becomes one of the major hurdles. And when multiple devices tried connecting with the main center, it clogged the system delaying the procedure. Here edge computing offered a unique solution that accelerated the growth of IoT Systems overall.

Now with the edge IoT layers, systems can process and analyze the information close to the source as much as possible. Edge has now become the standard for the 5th Generation of mobile networks (5G), offering systems to connect with more devices at a lower latency than the prevailing 4G standards. All the procedures for the IoT networks take place at the edge. Thus saving time, resources and further resulting in real-time reactions and improved performance.

4. PROCESSING LAYER

IoT systems are designed to capture, store, and process data for further requirements in this layer. In the processing layer, there are two main stages.

- **Data Accumulation** Every device is sending millions of data streams across the IoT network. Here data comes in various forms, speeds, and sizes. Separating the essential data from these large streams is a primary concern that professionals must prioritize in this layer. Unstructured data in raw form such as photos and video streams can be quite enormous and must be done efficiently to gather intelligence factors for the business. Professionals must have a thorough understanding of the business procedures to pinpoint data requirements precisely and help procure future benefits.

- **Data Abstraction** Once the data accumulation stage is finished, selected data is taken out from the large data for application to optimize their business procedures. Here the data abstraction follows the path as:
    - Collecting all the data from all IoT and non-IoT systems (CRM, ERP, & ERM)
    - Using data virtualization to make data accessible from a single location
    - Managing raw data in multiple forms

  Interoperability among devices and architecture plays a crucial role in the processing layer. Once data accumulation and abstraction are complete, it is easy for data analysts to use business acumen in fetching intelligence factors.

5. APPLICATION LAYER

In this layer, Data is further processed and analyzed to gather business intelligence. Here IoT systems get connected with middleware or software that can understand data more precisely. Some examples of the Application layer include:

- Business decision-making software's
- Device control and monitoring services

- Analytics solutions built with Machine learning and Artificial Intelligence
- Mobile Application for further interactions

Each IoT system is built with its particular goals and objectives to match with business specifications. At present, most of the IoT Applications are working at a varying complexity and operate a multitude of technology stacks performing specific tasks for businesses.

6. BUSINESS LAYER

Once IoT data is procured, it is valuable only if it applies to business planning and strategy. Every business has specific goals and objectives that it wants to accomplish by gathering intelligence from data. Business owners and stakeholders use data from past and present data to plan precisely for the future.

Today Data analysis has become the new oil for industries to enhance their productivity. Businesses are competing to get more data into their business for analysis and decision-making. Here software, CRM, and business intelligence programs have gained a lot of popularity in industries for superior performance.

7. SECURITY LAYER

With modern challenges, security has become one of the main necessities of IT architecture. Data breach, tracking malicious software, and hacking are the main challenges with Security Layer in integrating IoT systems.

- **Device Security :** The first point of security in the IoT layers starts with the devices themselves. Most of the manufacturers follow security guidelines to install in both firmware and hardware for IoT integration. Some of the essential measures are:
    - Secure boot process to avoid any malicious code running on a device
    - Using Trusted Platform Module (TPM) chips in combination with cryptographic keys for devices endpoint protections
    - Extra physical layer to avoid direct access via the device
    - Regular updates for security patches
- **Cloud Security :** Now Clouds are taking over from the traditional server for data storage and communication. Their data security is of paramount importance, especially for IoT systems. Mechanisms include multiple authorization factors and encryptions to avoid any data breach. Here the process of verifying any new device is an essential crux that must have strict regulations for device identity management.

- **Connection Security:** While transferring data across the network, it must be encrypted from an end-to-end point across the IoT system. Here messaging protocols such as DDS, AMQP, and MQTT are integrated to secure sensitive information from any breach. The use of TSL cryptographic protocol is recommended industry standard across IoT architecture for data communication.

Though elements of IoT form the core fundamentals for a comprehensive IT system, the IoT Layers lay down the path for the overall network's success. Each layer has a specific scope and role that caters to solving the IoT complexity across the network. The Perception layer is responsible for converting signals from analog to digital and vice-versa, the Connectivity layer for easing data transformation, and Edge for lowering system latency. While the Processing layer refines the data, the Application layer analyses the data, the Business layer delivers intelligence solutions, and the Security layer makes sure to protect the data at any cost.

Suggestions & Better Notes are Invited at agrawalsonu@gmail.com

ASSOCIATED TECHNOLOGIES WITH IOT

M2M MACHINE-TO-MACHINE is a broad label that can be used to describe any technology that enables networked devices to exchange information and perform actions without the manual assistance of humans. Artificial intelligence (AI) and machine learning (ML) facilitate the communication between systems, allowing them to make their own autonomous choices. This increases the potential of security threats, such as hacking, data breaches and unauthorized monitoring. In order to repair itself after malicious attacks or faults, an M2M system must allow remote management, like firmware updates.

M2M technology was first adopted in manufacturing and industrial settings, where other technologies, such as SCADA and remote monitoring, helped remotely manage and control data from equipment. M2M has since found applications in other sectors, such as healthcare, business and insurance. M2M is also the foundation for the internet of things (IoT).

HOW M2M WORKS

The main purpose of machine-to-machine technology is to tap into sensor data and transmit it to a network. Unlike SCADA or other remote monitoring tools, M2M systems often use public networks and access methods -- for example, cellular or Ethernet -- to make it more cost-effective.

The main components of an M2M system include sensors, RFID, a Wi-Fi or cellular communications link, and autonomic computing software programmed to help a network device interpret data and make decisions. These M2M applications translate the data, which can trigger preprogrammed, automated actions.

One of the most well-known types of machine-to-machine communication is telemetry, which has been used since the early part of the last century to transmit operational data. Pioneers in telemetric first used telephone lines, and later, radio waves, to transmit performance measurements gathered from monitoring instruments in remote locations. The Internet and improved standards for wireless technology have expanded the role of telemetry from pure science, engineering and manufacturing to everyday use in products such as heating units, electric meters and internet-connected devices, such as appliances.

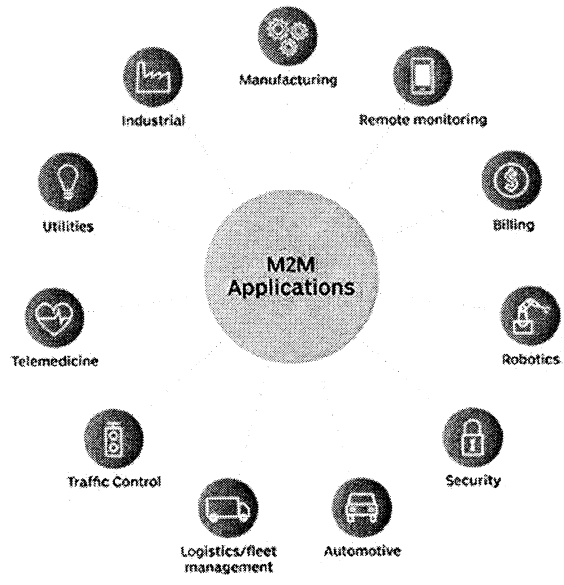
Beyond being able to remotely monitor equipment and systems, the top benefits of M2M include:

- reduced costs by minimizing equipment maintenance and downtime;

- boosted revenue by revealing new business opportunities for servicing products in the field; and
- improved customer service by proactively monitoring and servicing equipment before it fails or only when it is needed.

M2M APPLICATIONS AND EXAMPLES

- Machine-to-machine communication is often used for remote monitoring. In product restocking, for example, a vending machine can message the distributor's network, or *machine*, when a particular item is running low to send a refill. An enabler of asset tracking and monitoring, M2M is vital in warehouse management systems (WMS) and supply chain management (SCM).
- Utilities companies often rely on M2M devices and applications to not only harvest energy, such as oil and gas, but also to bill customers through the use of smart meters and to detect worksite factors, such as pressure, temperature and equipment status.



- In telemedicine, M2M devices can enable the real time monitoring of patients' vital statistics, dispensing medicine when required or tracking healthcare assets.
- The combination of the IoT, AI and ML is transforming and improving mobile payment processes and creating new opportunities for different purchasing behaviors. Digital wallets, such as Google Wallet and Apple Pay, will most likely contribute to the widespread adoption of M2M financial activities.
- Smart home systems have also incorporated M2M technology. The use of M2M in this embedded system enables home appliances and other

technologies to have real time control of operations as well as the ability to remotely communicate.

- M2M is also an important aspect of remote-control software, robotics, traffic control, security, logistics and fleet management and automotive.

M2M REQUIREMENTS

According to the European Telecommunications Standards Institute (ETSI), requirements of an M2M system include:

- Scalability - The M2M system should be able to continue to function efficiently as more connected objects are added.
- Anonymity - The M2M system must be able to hide the identity of an M2M device when requested, subject to regulatory requirements.
- Logging - M2M systems must support the recording of important events, such as failed installation attempts, service not operating or the occurrence of faulty information. The logs should be available by request.
- M2M application communication principles - M2M systems should enable communication between M2M applications in the network and the M2M device or gateway using communication techniques, such as short message service (SMS) and IP Connected devices should also be able to communicate with each other in a peer-to-peer (P2P) manner.
- Delivery methods - The M2M system should support Unicast, anycast, multicast and broadcast communication modes, with broadcast being replaced by multicast or anycast whenever possible to minimize the load on the communication network.
- Message transmission scheduling - M2M systems must be able to control network access and messaging schedules and should be conscious of M2M applications' scheduling delay tolerance.
- Message communication path selection - Optimization of the message communication paths within an M2M system must be possible and based on policies like transmission failures, delays when other paths exist and network costs.

M2M VS. IoT

While many use the terms interchangeably, M2M and IoT are not the same. IoT needs M2M, but M2M does not need IoT. Both terms relate to the communication of connected devices, but M2M systems are often isolated, stand-alone networked equipment. IoT systems take M2M to the next level, bringing together disparate systems into one large, connected ecosystem. M2M systems use point-to-point communications between machines, sensors and hardware over cellular or

wired networks, while IoT systems rely on IP-based networks to send data collected from IoT-connected devices to gateways, the cloud or middleware platforms.

Data collected from M2M devices is used by service management applications, whereas IoT data is often integrated with enterprise systems to improve business performance across multiple groups. Another way to look at it is that M2M affects how businesses operate, while IoT does this *and* affects end users.

M2M	IoT
Machines	Sensors
Hardware-based	Software-based
Vertical applications	Horizontal applications
Deployed in a closed system	Connects to a larger network
Machines communicating with machines	Machines communicating with machines, humans with machines, machines with humans
Uses non-IP protocol	Uses IP protocols
Can use the cloud, but not required to	Uses the cloud
Machines use point-to-point communication, usually embedded in hardware	Devices use IP networks to communicate
Often one-way communication	Back and forth communication
Main purpose is to monitor and control	Multiple applications; multilevel communications
Operates via triggered responses based on an action	Can, but does not have to, operate on triggered responses
Limited integration options, devices must have complementary communication standards	Unlimited integration options, but requires software that manages communications/protocols
Structured data	Structured and unstructured data

For example, in the product restocking example, M2M involves the vending machine communicating to the distributor's machines that a refill is needed. Incorporate IoT and an additional layer of analytics is performed; the vending machine can predict when particular products will need refilling based on purchase behaviors, offering users a more personalized experience.

M2M SECURITY

Machine-to-machine systems face a number of security issues, from unauthorized access to wireless intrusion to device hacking. Physical security, privacy, fraud and the exposure of mission-critical applications must also be considered. The inability to properly service the M2M equipment creates various unique security vulnerabilities for the M2M systems and the wireless networks they use to communicate.

Typical M2M security measures include making devices and machines tamper-resistant, embedding security into the machines, ensuring communication security through encryption and securing back-end servers, among others. Segmenting M2M

devices onto their own network and managing device identity, data confidentiality and device availability can also help combat M2M security risks.

M2M STANDARDS

Machine-to-machine technology does not have a standardized device platform, and many M2M systems are built to be task or device-specific. Several key M2M standards, many of which are also used in IoT settings, have emerged over the years, including:

- OMA DM (Open Mobile Alliance Device Management), a device management protocol
- OMA Lightweight M2M, a device management protocol
- MQTT, a messaging protocol
- TR-069 (Technical Report 069), an application layer protocol
- HyperCat, a data discovery protocol
- OneM2M, a communications protocol
- Google Thread, a wireless mesh protocol
- AllJoyn, an open source software framework

HISTORY OF M2M

1. While the origins of the acronym are unverified, the first use of machine-to-machine communication is often credited to Theodore Paraskevakos, who invented and patented technology related to the transmission of data over telephone lines, the basis for modern-day caller ID.
2. Nokia was one of the first companies to use the acronym in the late 1990s. In 2002, it partnered with Opto 22 to offer M2M wireless communication services to its customers.
3. In 2003, *M2M Magazine* launched. The publication has since defined the six pillars of M2M as remote monitoring, RFID, sensor networking, smart services, telematics and telemetry.

TELEMEDICINE

Telemedicine is the use of electronic information and devices to communicate technologies to provide and support healthcare when distance separates the doctors and patients. The core concept of telemedicine is to deliver remote healthcare services to people living in remote areas, especially where there are no healthcare facilities. Besides, it also had a purpose to reduce the burden of hospitals loaded with an end number of patients.

"*Tele*" is a Greek word meaning "distance "and "*mederi*" is a Latin word meaning "to heal". Time magazine called telemedicine "healing by wire". Although initially considered "futuristic" and "experimental," telemedicine is today a reality and has come to stay. Telemedicine has a variety of applications in patient care, education, research, administration and public health. Worldwide, people living in rural and remote areas struggle to access timely, good-quality specialty medical care. Residents of these areas often have substandard access to specialty healthcare, primarily because specialist physicians are more likely to be located in areas of concentrated urban population. Telemedicine has the potential to bridge this distance and facilitate healthcare in these remote areas. For example, the quality of care, enhanced patient monitoring system, real-time health tracking system, and much more. Let's glance through the top benefits of IoT in healthcare, especially in telemedicine.

- **Supports Doctors:** Whether it is to access patients data or examine the wound, rash, or patient health condition improvement, the Internet of Things can be the best supporting innovation for doctors. However, doctors also have to face some of the challenges in this case. For example, they cannot listen to the heartbeat or check blood pressure using IoT technology. However, there is a solution to it. Wearable devices powered by IoT play an important role by allowing doctors to access relevant information.
- **Real-time health monitoring:** IoT has plenty of offerings to healthcare, while real-time monitoring is the life-saving feature that IoT delivers to healthcare. Doctors can get patients' health reports every second, establishing better and enhanced healthcare facilities. Patients wearing a Holter monitor device will enable doctors and patients to check abnormalities (if it occurs) in their heartbeat.
- **Best mate for senior citizen:** Doctors and hospitals can provide healthcare facilities to senior citizens without letting them leave their home. Technology can help them make their life more comfortable and hassle-free. IoT devices will not only help people, especially senior citizens, improve their health condition, but also it will be a boon for people with disabilities.

THINGS TO BE CONSIDER BEFORE DEPLOYING IOT FOR TELEMEDICINE

IoT is the need of the hour, especially in the healthcare sector. However, you can't just get it developed and integrated into your healthcare system. You need to consider certain aspects of it. They are;

1. **Infrastructure:** Even though IoT does not require any complex infrastructure to deploy, one should make sure that there are no last-minute hassles. For example, power points, access points for cable, permission for device cabling, no connectivity barrier, proper installation space, and should be ensured earlier to avoid any further glitch. You may also need to have a networking staff that will support you from device installation to management.
2. **On-site IoT maintenance:** If you are considering installing IoT technology into your healthcare system, you have to consider its maintenance. Initially, you may not need to spend on its care, but as it grows older, the IoT devices will get worn off, and then maintenance will be needed. It could be a challenge, and you may be compelled to hire professionals who will look into this matter.
3. **Security arrangement:** IoT works on telecommunications along with internet connections, and hence it is crucial for users to have secure connections. It's really a challenge for healthcare managers to keep Patient Identity and health information safe and protected. PII refers to Patients Identity Information, and PHI refers to Patient Health Information, and both contain highly sensitive information, and security is essential during the storage and transmission stage.
4. **Device support and testing:** You would not take the risk of getting a technical glitch once it is implemented. Therefore, it is essential to have proper testing and ensure tech support to get the seamless result. If IoT devices are not tested, you may fail to achieve users' satisfaction.

LIST OF IOT APPLICATIONS/USE CASES FOR TELEMEDICINE

- **Mobile health:** IoT has added another feather to the mHealth system that enables doctors to keep track of all vitals, including blood pressure, heartbeat, etc., of patients as we have discussed earlier the challenges that doctors have to face with IoT, in case of remote monitoring. With IoT integrated with smart wearable devices, doctors can now quickly get detailed reports, including the required vitals of the body. With mobile health, handling critical situations will become easy.

- **Reminder/Ingestible sensors/devices:** Sometimes or more often, patients forget to take medicines timely. Even doctors get busy with other work, forgetting that there are patients who need attention. But, these all things are far gone now. With IoT in place, the ingestible sensors can keep them regular. Patients will be reminded to take medicine in time, while doctors will get notification and alert if they have to be examined.
- **Remote patient monitoring:** Remote patient monitoring is believed to be the sole reason why IoT was introduced in telemedicine. IoT-enabled devices for patient monitoring ensures that doctors get all necessary information such as the patient's health condition, blood pressure level and so much more remotely. Whereas patients, especially those living in far off areas do not have to come to the hospital physically to attend treatment.

These are the basis use cases, though you can come with plenty of IoT app use cases in telemedicine. You can take expert advice when you consider telemedicine app development and find out more use cases of IoT into your sector.

TOP CHALLENGES OF IOT IN TELEMEDICINE

Challenges regarding IoT in telemedicine discrussed earlier are collecting data, exploring the authenticity of it, managing the data, and more. Top challenges that may be faced with IoT:

- **Unauthorized access / Vulnerability:** When it comes to IoT in telemedicine, having a robust security module system is the need of the hour. If you don't have that, it might invite hackers and online robbers to peep through your internal affairs. And that may not be good for your healthcare facilities.
- **Security at stack:** IoT, if not protected well, will be vulnerable to hack. What if even a single patient's data is hacked? It will keep the centralized system at the stack. You may not be able to control the situation efficiently. However, proper precaution and timely maintenance, along with team support, can help you avoid such issues.
- **Health/care policies transformation:** Even though world healthcare is going through a transition, it should be consistent across all states and countries. For example, many countries follow outdated healthcare systems and regulations, which may create barriers in integrating technology.

BIG DATA

With each passing day, there is an exponential increase in the number of devices and machines that are getting connected to the internet to transmit information for analysis. Data analysis is done to discover trends that help businesses and organizations forecast future outcomes and be ready for upcoming challenges.

Big Data refers to a massive set of data that no conventional data management tool can handle. Big Data is therefore a concept that allows access to gigantic databases in real time. It has three main features:

- The speed at which information is processed;
- The variety of information stored (in the form of processed or unprocessed data from a variety of sources);
- The volume of information listed.

Big Data's main objectives are to improve a company's or system's responsiveness to a large amount of data collected, increase productivity and refine knowledge of customer behavior, so that it can offer personalized offers or advertisements and create new trends.

RELATIONSHIP BETWEEN BIG DATA AND IoT

According to several studies, the use of IoT is expected to generate 4.4 trillion GBytes in 2020, and this figure is expected to increase in subsequent years. In addition, this data must be read, exploited and transmitted within specific timeslots, so, as you might have guessed, the major challenge in the field of the Internet of Things is to be able to exploit a huge amount of data, hence the use of big data. Big Data should enable **real-time analysis of the data generated by IoT** and thus optimize the use of this technology. To do this, Big Data proceeds in four steps:

1. A large amount of unstructured data is generated by IoT devices which are collected in the big data system. This IoT generated big data largely depends on their 3V factors that are volume, velocity, and variety.
2. In the big data system which is basically a shared distributed database, the huge amount of data is stored in big data files.
3. Analyzing the stored IoT big data using analytic tools like Hadoop MapReduce or Spark
4. Generating the reports of analyzed data.

Big Data will play an important role in information processing efficiency and will enable IoT developers to optimize these tools to broaden the current perspective. The interaction between IoT and Big Data is not one-way. IoT could also bring a lot to Big Data. It will thereby be important to improve data storage technologies to develop systems capable of **processing even more data.** This interaction could thus enable technological growth in both areas simultaneously.

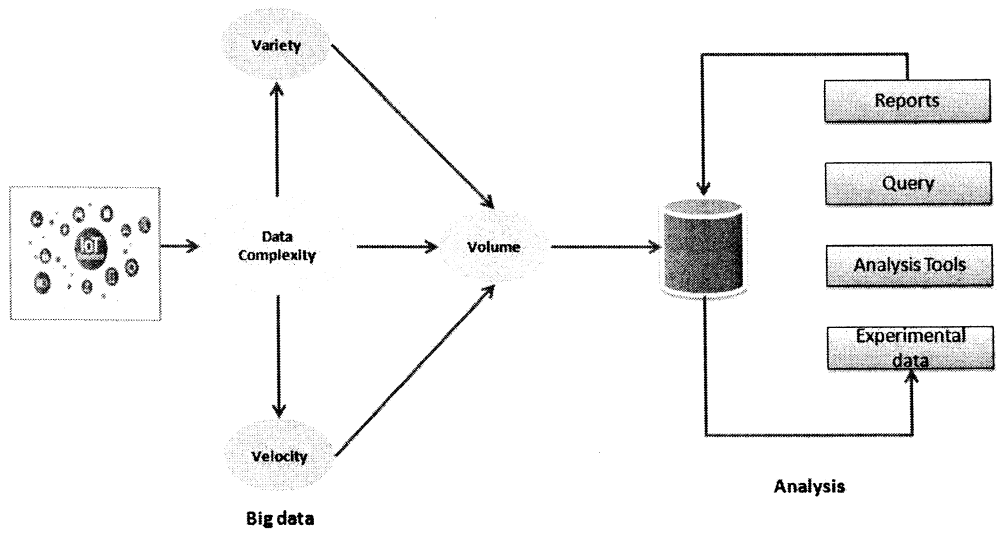
THE RELATION BETWEEN BIG DATA AND IOT

As per the study, around 4.4 trillion GB of data will be generated by the year 2020 through the Internet of Things. This is no doubt difficult to comprehend easily. However, with the growing number of connected devices it is not surprising that by 2020, more than ten billions of sensors and devices will be connected to the internet. Furthermore, all of these devices will gather, analyze, share, and transmit data in real time. Hence, without the data, IoT devices would not hold the functionalities and capabilities which have made them achieve so much worldwide attention.

ROLE OF BIG DATA IN IOT

When organizations are grabbing hold of the data for analysis purpose, IoT is acting as a major source for that data, and this is the point where the role of big data in IoT comes into the picture. Big data analytics is emerging as a key to analyzing IoT generated data from "connected devices" which helps to take the initiative to improve decision making. The role of big data in IoT is to process a large amount of data on a real-time basis and storing them using different storage technologies.

Since in IoT the unstructured data are collected via the internet, hence, big data for the internet of things need lightning-fast analysis with large queries to gain rapid insights from data to make quick decisions. Hence the need for big data in IoT is compelling. Hence, from the big data perspective, it is the fuel that drives Internet of Things run.



IOT Big Data Processing

It's not just that there is the only interdependent relation between big data and IoT. As they help each other, in addition to that they hugely impact each other. Fact is the more the IoT grows it will place more demand on businesses regarding big data capabilities. For example, as the IoT generated data is increasing at a huge rate,

conventional data storage technology is already being pushed to its limits. As a result, it demands more advanced and innovative storage solutions to handle these growing workloads resulting in updating the infrastructure of an organization's big data storage. Similarly, the IoT big data combined applications accelerate the scope of research in both the fields. So, IoT and big data both the technologies carry inter-dependency and need further development.

THE FOUR "V"S

To help understand such an enormous concept, data scientists at IBM popularized "The Four V's" of big data: volume, variety, velocity, and veracity.

1. Volume: The incredible amount of data regularly collected through sensors, online transactions, social media, and other mediums cannot be processed or even stored using traditional methods. According to some estimates, people will generate around 463 exabytes of data every day by 2025. As you may imagine, some of these data sets can be too large to fit on a single server, and must instead be distributed between several storage locations. Data analytics frameworks such as Amazon S3, Hadoop, or Spark, are built to accommodate the need for distributed storage and aggregation.
2. Variety: Today's data comes in many different types, from social media posts about your favorite food to readings from moisture sensors in a field. In past decades, data was more clearly defined—for example, phone numbers, addresses, or ledger amounts—and could be easily amassed into spreadsheets or tables. Today's digital data often cannot be corralled into traditional structures. Powerful analytics software seeks to harness unstructured data, such as images and videos, and combine it with more straight forward data streams to provide additional insights.
3. Velocity: in 2020 alone, people created about 1.7MB of data every second. That's more than 2.5 quintillion bytes of data per day. Considering that Google processes more than 3.5 billion searches in a day, around 60 hours of video are uploaded to YouTube every minute, and 6,000 tweets are sent every second, processing all of this data is a massive undertaking - and that's without even mentioning the considerable insights generated by machine learning. As you may suspect, all of this accumulated data is streaming into servers at an unprecedented speed.
4. Veracity: Veracity refers to the truthfulness or accuracy of a particular set of data. That includes evaluation of the data source's trustworthiness. IBM estimates that poor data quality costs the U.S. around $3.1 trillion per year, so pursuing veracity is important. It includes eliminating duplication, limiting bias, and processing data in ways that make sense for the particular application or vertical. This is an area where human analysts and traditional statistical methodologies are still of great value. While AI is becoming more sophisticated, it cannot yet match the discernment of a trained human brain.

CLOUD

Back in the 1970s, it was popular for businesses to rent mainframe computer systems. These systems were extremely large and expensive, so it didn't make sense financially for businesses to own the computing power themselves. Instead, they were owned by large corporations, government agencies, and universities. Microprocessor technology allowed for great reductions in size and expense, leading to the advent of the personal computer, which exploded in popularity in the 1980s. Suddenly, businesses could (and did) bring computation in-house. However, as high-speed connections have become widespread, the trend has reversed: businesses are once again renting computing power from other organizations.

Instead of buying expensive hardware for storage and processing in-house, it's easy to rent it for cheap in the cloud. **The cloud is a huge, interconnected network of powerful servers that performs services** for businesses and for people. The largest cloud providers in the US are Amazon, Google, and Microsoft, who have huge farms of servers that they rent to businesses as part of their cloud services.

For businesses that have variable needs (most of the time they don't need much computing, but every now and then they need a lot), this is cost-effective because they can simply pay as-needed.

When it comes to people, we use these cloud services all of the time. You might store your files in Google Drive instead of on your personal computer. Google Drive, of course, uses Google's cloud services. or you might listen to songs on Spotify instead of downloading the songs to your computer or phone. Spotify uses Amazon's cloud services.

Generally, something that happens "in The Cloud" is any activity that takes place over an internet connection instead of on the device itself.

THE INTERNET OF THINGS AND THE CLOUD

Because activities like storage and data processing take place in the cloud rather than on the device itself, this has had significant implications for IoT. Many IoT systems make use of large numbers of sensors to collect data and then make intelligent decisions.

Using the cloud is important for aggregating data and drawing insights from that data. Using the cloud also allows for high scalability. When you have hundreds, thousands, or even millions of sensors, putting large amounts of computational power on each sensor would be extremely expensive and energy-intensive. Instead, data can be passed to the cloud from all these sensors and processed there in

aggregate. For much of IoT, the head (or rather, the brain) of the system is in the cloud. Sensors and devices collect data and perform actions, but the processing/commanding/analytics (aka the "smart" stuff), typically happens in the cloud.

ADVANTAGE OF USING CLOUD FOR IOT

So far we've only been discussing the benefits of using the cloud for IoT. Let's briefly summarize them:

- Decreased costs, both upfront and infrastructure
- Pay-as-needed for storage/computing
- High system scalability and availability
- Increased lifespan of battery-powered sensors/devices
- Ability to aggregate large amounts of data
- Anything with an internet connection can become "smart"

DISADVANTAGE OF USING CLOUD FOR IOT

- **Data ownership.** When you store data in a company's cloud service, do you own the data or does the cloud provider? This can be hugely important for IoT applications involving personal data such as healthcare or smart homes.
- **Potential crashes.** If the connection is interrupted or the cloud service itself crashes, the IoT application won't work. Short-term inoperability might not be a big deal for certain IoT applications, like smart agriculture, but it could be devastating for others. You don't want applications involving health or safety crashing for even a few seconds.
- **Latency.** It takes time for data to be sent to the cloud and commands to return to the device. In certain IoT applications, these milliseconds can be critical such as in health and safety. A good example is Autonomous Vehicles. If a crash is imminent, you don't want to have to wait for the car to talk to the cloud before making a decision.

The Internet of Things is a broad field and includes an incredible variety of applications. **There is no one-size-fits-all solution** so IoT companies need to consider their specific application when deciding whether the cloud makes sense for them.

SMART GRID

The "grid" is the electrical network serving every resident, business and infrastructure service in a city. The "smart grid" is the next generation of those energy systems, which have been updated with communications technology and connectivity to drive smarter resource use.

The technologies that make today's IoT-enabled energy grid "smart" include wireless devices such as sensors, radio modules, gateways and routers. These devices provide the sophisticated connectivity and communications that empower consumers to make better energy usage decisions, allow cities to save electricity and expense, and enables power authorities to more quickly restore power after a blackout.

The growing trend today is for municipalities to move toward smart grid technologies for a range of reasons. These include the need to improve energy usage, provide better customer service to their citizens, prepare for disasters and upgrade aging technology that is expensive to maintain. As well, advances in technology have made wireless, both cellular and RF (radio frequency), affordable and easy to use in smart grid applications.

Smart grid allows a power company to assess system health in significantly more detail than was previously possible. For instance, with smart meters the power company can discover real time power demands with a granularity and accuracy that is simply not possible with older technology. This can allow them to better predict and respond to sudden increases in demand, which can help to prevent blackouts. In the event that a blackout does occur, IoT devices that use cellular and RF technology installed in transformers and substations can automatically redirect power. That can allow for a faster, easier fix versus having to dispatch service personnel in a truck each time the power goes out.

"A smarter grid will add resiliency to our electric power system and make it better prepared to address emergencies such as severe storms, earthquakes, large solar flares, and terrorist attacks. Because of its two-way interactive capacity, the Smart Grid will allow for automatic rerouting when equipment fails or outages occur."

Smart city applications are vast, and include everything from smart city lighting, energy management and intelligent traffic management to water treatment and wastewater management. Sensors in traffic lights can send information back to a central authority for decision making. Even better, with intelligent traffic systems, both surface traffic and public transportation can be managed with routing and traffic lighting to improve or eliminate congestion.

IoT sensors in streetlights can also adjust off and on timing and brightness according to real time conditions. Plus or minus a few watts might not sound like much. However, when considering the thousands or tens of thousands of streetlights that can be found in any given city, the savings and environmental impact quickly add up. Those same sensors can also send out an alert if a light needs servicing. No need to wait for a call from an angry customer complaining about street lights being out.

Additionally, with a sophisticated remote management solution, technicians can remotely troubleshoot the issue and determine whether or not to send a truck. In the past, a truck roll – a highly expensive proposition compared to a fast firmware fix or reboot from a management system in the home office was inevitable.

Smart meters enable demand response which lets home and business owners see real time pricing information so that they can adjust their energy usage accordingly. For example, switching off the AC, or turning down the thermostat in winter. Most of all smart meters will benefit electric car owners. With real time pricing information EV owners will be able to charge their cars when electricity is the cheapest and avoid charging, if possible, during times of peak demand.

SMART GRID FOR THE FUTURE

Smart grid technology can be expressed in a single sentence: a new electric grid with two way communication. For the first time, businesses and consumers can get real time billing information while utility companies can better meet the needs of their customers as they react to demand spikes and fix or manage blackouts and other challenges. Smart grid is resilient, efficient and green which is good for the consumer, the utility company and the environment. Wireless technology will replace thousands of miles of cable that would have been needed to advance the smart grid to where it is today.

IOV (INTERNET OF VEHICLES)

IoV is the next step in Vehicular ad hoc networks (VANET) evolution. A VANET is a cluster of mobile nodes (vehicles) using ad hoc on-demand connections to communicate.

In VANETs, vehicle-to-vehicle and vehicle-to-roadside communications networks would exist side by side to provide navigation, road safety, and other services while on the road. VANETs are an essential component of the Intelligent Transportation Systems framework. VANETs are sometimes referred to as Intelligent Transportation Networks. They have evolved into comprehensive IoVs that will eventually develop into a worldwide Internet of Autonomous Cars.

Vehicles connect to the internet via a wireless local area network (WLAN), and connected cars become part of the IoT and IoV. This allows remote control of certain aspects and interaction with other devices, traffic infrastructure, and third-party products. For example, drivers get alerts regarding traffic, they can also plan routes, and the connected vehicle can automatically adjust cruise control for better traffic management. IoV has the potential to improve driver safety, decrease traffic congestion, and lower emissions.

IOV ARCHITECTURE

There are multiple ways to implement and design IoV systems, but IoV architecture should have the following four layers for proper IoT functionality.

1. **Perception :** It is important to take into account the environmental factors when developing IoT systems. For example, an IoV car needs to sense obstacles, other vehicles, and human movement to avoid people or objects that may pose a threat. If an object is approaching too quickly, the driverless car should slow down to avoid collisions. The vehicle itself should also be monitored for any signs of failure or poor performance; it would not make sense for drivers to rely heavily on IoT if their cars are unreliable. All of the vehicle's sensors are included in this layer. In addition, the perception layer gathers environmental information for detecting events, driving habits, and situations. It also has radio frequency identification (RFID) and the ability to sense surroundings, vehicle position, and other objects on the road.
2. **Network :** This is the communication layer that provides all the connectivity needed. It includes the 3G/4G LTE/5G cellular technologies, WLAN, Bluetooth, and Wi-Fi to communicate between the car, other devices, and infrastructure. 5G, in particular, is critical to the takeoff of IoV. 5G functionality will support the high data rates and bandwidth, low latency,

and a reliable connection. In addition, the 5G IoT cellular standard allows for a more efficient and effective transfer of large amounts of real-time data between multiple devices within the car.

3. **Artificial Intelligence (AI)** : Once IoV has gathered all the data, it must be used to make decisions and allow the car to react accordingly. This is where AI becomes a key component of IoT as it allows for prediction, decision making, and action execution. It comprises big data analysis software, specialized systems (computer vision applications in driverless cars to identify objects on the road), and cloud computing components. The AI layer has a built-in cloud infrastructure and requires smooth connectivity between the processing services and the low-level system components.
4. **Application** : the application layer applies the results from the AI layer. This is where data gathered through perception is processed through applications such as collision avoidance software. It is the user-side of IoT and can be used for multiple purposes such as entertainment, GPS navigation, and in-vehicle services. The application layer includes all network-aware applications within the IoV system. It also includes telematics (GPS), infotainment, and general car functions (such as monitoring engine performance). The IoV also comprises all connected vehicles' global ID terminals (GID). It solves all issues with RFID, including slow speeds and limited coverage. What's more, GIDs give vehicles digital IDs that are critical for vehicle cyber security.

EXAMPLES OF IOV APPLICATIONS

Below are a few examples of specific use cases in the IoV automotive industry.

1. **Uber** : Uber is the most popular ride-sharing company. Its IoV and self-driving unit, Advanced Technologies Group (ATG), has made great strides in IoV and self-driving vehicles. One of the areas Uber is working to leverage IoV to gain a competitive advantage, improve customer service, and return to profitability is transportation costs. According to projections from research, the driver accounts for 80% of the overall per kilometer cost in non-autonomous ride-sharing. Fully autonomous cars will dramatically lower the price of a journey while expanding their addressable market by removing the driver from the equation. Uber is already providing software as a service and plans to take the risk further by making ride expenses so low (between its human and robot car fleet) that vehicle ownership becomes impractical.
2. **Tesla** : Tesla, the Wunderkind of IoV, is on a mission to accelerate the world's transition to sustainable energy. The Tesla team aims to do this by making self-driving electric vehicles safe, affordable, and scalable. Their approach combines several technologies in one platform: an advanced driver assistance

system (ADAS), automated controls for vehicles to drive themselves without human input, and fleet routing using AI-controlled machines (e.g., autonomous buses or fleets of commercial trucks).

3. **CarStream :** CarStream is a big data processing system for chauffeured car services. CarStream collects and analyzes a variety of sorts of driving data, including vehicle condition, driver behavior, and passenger-trip information. Based on the data collected, various services are provided. For the past three years in industrial usage, CarStream has gathered over 40 terabytes of driving data. CarStream also provides IoV safety-critical services and has developed a three-layered monitoring system. The monitoring subsystem covers the application layer all the way down to the infrastructure layer. To address real-time processing, big data, low data quality, and sparse value storage issues, CarStream utilizes in-memory caching and stream processing. As a result, a diverse storage system manages a massive amount of driving information.
4. **Lyft:** Similar to Uber, Lyft provides car passenger ridesharing services. To process data, they use Apache Beam with the Apache Flink runner streaming processing framework. It is used to run feature engineering and orchestration jobs. Lyft has built scalable pipelines that process real-time events with extremely low latency. The company generates around 100 features at about three million geohashes per minute, equating to approximately 400 billion features a day.

IOV AS AN EMERGING SECTOR

IoV is an emerging sector with the potential to affect the automotive industry profoundly. The applications range from navigation, entertainment, in-vehicle services; down to fleet routing using AI-controlled machines. With everything at our fingertips via IoT (Internet of Things), it's no surprise that devices like cars are getting smarter, too. The Internet of Vehicles (IoV) is a subset of the Internet of Things (IoT). There's a growing need for intelligent and connected devices that help us in daily life.

HOW IS A CAR CONNECTED TO THE INTERNET?

Connected cars hook up to the internet via a wireless local area network (WLAN). Connected cars are part of the Internet of Things and the Internet of Vehicles, allowing remote control of some aspects and communication with other products. For example, connected cars can alert drivers to traffic, plan routes, and adapt cruise control for more efficient traffic management.

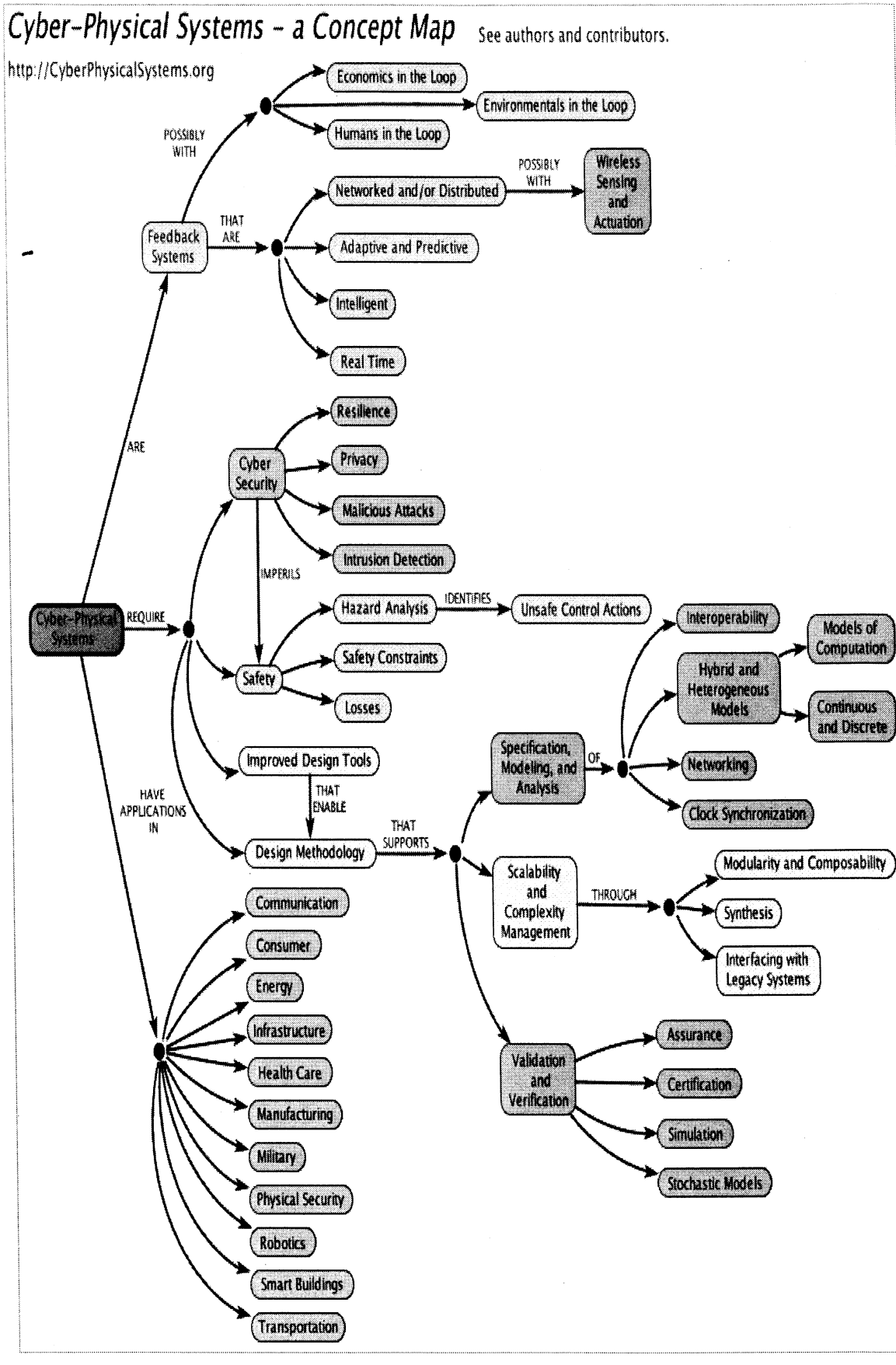
There are five main ways vehicles connect with surroundings and communicate:

- V2I: Vehicle to Infrastructure
- V2V: Vehicle to Vehicle
- V2C: Vehicle to Cloud
- V2P: Vehicle to Pedestrian (or Vehicle to Human)
- V2X: Vehicle to Everything

IoV has benefited from the improvement of big data technologies and mobile networking. IoV solutions differ from vehicle networking technologies like V2V communication and vehicular networks. With a cloud-based approach, connected vehicles work within a cloud data center, vehicle statuses are uploaded through wireless communications, and vehicle data is usually treated as a data stream. There are three major challenges to designing an IoV ecosystem that's capable of supporting these services:

1. High reliability
2. High scalability
3. Desirable trade-off between accuracy and real-time processing

Once the IoV service is up and running, it needs to run continuously – unexpected crashes or downtime may have critical consequences.


Cyber-Physical Systems - a Concept Map
See authors and contributors.
http://CyberPhysicalSystems.org
Economics in the Loop
Environmentals in the Loop
Humans in the Loop
POSSIBLY WITH
Networked and/or Distributed
POSSIBLY WITH
Wireless Sensing and Actuation
Feedback Systems
THAT ARE
Adaptive and Predictive
Intelligent
Real Time
ARE
Resilience
Cyber Security
Privacy
Malicious Attacks
Intrusion Detection
IMPERILS
Cyber-Physical Systems
REQUIRE
Hazard Analysis
IDENTIFIES
Unsafe Control Actions
Safety
Safety Constraints
Losses
Improved Design Tools
THAT ENABLE
HAVE APPLICATIONS IN
Design Methodology
THAT SUPPORTS
Specification, Modeling, and Analysis
OF
Interoperability
Hybrid and Heterogeneous Models
Models of Computation
Continuous and Discrete
Networking
Clock Synchronization
Scalability and Complexity Management
THROUGH
Modularity and Composability
Synthesis
Interfacing with Legacy Systems
Validation and Verification
Assurance
Certification
Simulation
Stochastic Models
Communication
Consumer
Energy
Infrastructure
Health Care
Manufacturing
Military
Physical Security
Robotics
Smart Buildings
Transportation

CPS (CYBER-PHYSICAL SYSTEM)

A cyber-physical system (CPS) or intelligent system is a computer system in which a mechanism is controlled or monitored by computer-based algorithms. In cyber-physical systems, physical and software components are deeply intertwined, able to operate on different spatial and temporal scales, exhibit multiple and distinct behavioral modalities, and interact with each other in ways that change with context. CPS involves transdisciplinary approaches, merging theory of cybernetics, mechatronics, design and process science. The process control is often referred to as embedded systems. In embedded systems, the emphasis tends to be more on the computational elements, and less on an intense link between the computational and physical elements. CPS is also similar to the Internet of Things (IoT), sharing the same basic architecture; nevertheless, CPS presents a higher combination and coordination between physical and computational elements.

Examples of CPS include smart grid, autonomous automobile systems, medical monitoring, industrial control systems, robotics systems, and automatic pilot avionics. Precursors of cyber-physical systems can be found in areas as diverse as aerospace, automotive, chemical processes, civil infrastructure, energy, healthcare, manufacturing, transportation, entertainment, and consumer appliances.

Unlike more traditional embedded systems, a full-fledged CPS is typically designed as a network of interacting elements with physical input and output instead of as standalone devices. The notion is closely tied to concepts of robotics and sensor networks with intelligence mechanisms proper of computational intelligence leading the pathway. Ongoing advances in science and engineering improve the link between computational and physical elements by means of intelligent mechanisms, increasing the adaptability, autonomy, efficiency, functionality, reliability, safety, and usability of cyber-physical systems. This will broaden the potential of cyber-physical systems in several directions, including: intervention (e.g., collision avoidance); precision (e.g., robotic surgery and nano-level manufacturing); operation in dangerous or inaccessible environments (e.g., search and rescue, firefighting, and deep-sea exploration); coordination (e.g., air traffic control, war fighting); efficiency (e.g., zero-net energy buildings); and augmentation of human capabilities (e.g. in healthcare monitoring and delivery).

MOBILE CYBER-PHYSICAL SYSTEMS

Mobile cyber-physical systems, in which the physical system under study has inherent mobility, are a prominent subcategory of cyber-physical systems. Examples of mobile physical systems include mobile robotics and electronics transported by humans or animals. The rise in popularity of smartphones has increased interest in

the area of mobile cyber-physical systems. Smartphone platforms make ideal mobile cyber-physical systems for a number of reasons, including:

- Significant computational resources, such as processing capability, local storage
- Multiple sensory input/output devices, such as touch screens, cameras, GPS chips, speakers, microphone, light sensors, proximity sensors
- Multiple communication mechanisms, such as WiFi, 4G, EDGE, Bluetooth for interconnecting devices to either the Internet, or to other devices
- High-level programming languages that enable rapid development of mobile CPS node software, such as Java,[8] C#, or JavaScript
- Readily available application distribution mechanisms, such as Google Play Store and Apple App Store
- End-user maintenance and upkeep, including frequent re-charging of the battery

For tasks that require more resources than are locally available, one common mechanism for rapid implementation of smartphone-based mobile cyber-physical system nodes utilizes the network connectivity to link the mobile system with either a server or a cloud environment, enabling complex processing tasks that are impossible under local resource constraints. Examples of mobile cyber-physical systems include applications to track and analyze CO2 emissions, detect traffic accidents, insurance telematics and provide situational awareness services to first responders, measure traffic, and monitor cardiac patients.

EXAMPLES

Common applications of CPS typically fall under sensor-based communication-enabled autonomous systems. For example, many wireless sensor networks monitor some aspect of the environment and relay the processed information to a central node. Other types of CPS include smart grid, autonomous automotive systems, medical monitoring, process control systems, distributed robotics, and automatic pilot avionics. Another example is MIT's ongoing CarTel project where a fleet of taxis work by collecting real-time traffic information in the Boston area. Together with historical data, this information is then used for calculating fastest routes for a given time of the day.

CPS are also used in electric grids to perform advanced control, especially in the smart grids context to enhance the integration of distributed renewable generation. Special remedial action scheme are needed to limit the current flows in the grid when wind farm generation is too high. Distributed CPS are a key solution for this type of issues.

CDN

A content delivery network (CDN) refers to a geographically distributed group of servers which work together to provide fast delivery of Internet content.

A CDN allows for the quick transfer of assets needed for loading Internet content including HTML pages, javascript files, stylesheets, images, and videos. The popularity of CDN services continues to grow, and today the majority of web traffic is served through CDNs, including traffic from major sites like Facebook, Netflix, and Amazon. A properly configured CDN may also help protect websites against some common malicious attacks, such as Distributed Denial of Service (DDOS) attacks.

While a CDN does not host content and can't replace the need for proper web hosting, it does help cache content at the network edge, which improves website performance. Many websites struggle to have their performance needs met by traditional hosting services, which is why they opt for CDNs. By utilizing caching to reduce hosting bandwidth, helping to prevent interruptions in service, and improving security, CDNs are a popular choice to relieve some of the major pain points that come with traditional web hosting.

Although the benefits of using a CDN vary depending on the size and needs of an Internet property, the primary benefits for most users can be broken down into 4 different components:

1. **Improving website load times** - By distributing content closer to website visitors by using a nearby CDN server (among other optimizations), visitors experience faster page loading times. As visitors are more inclined to click away from a slow-loading site, a CDN can reduce bounce rates and increase the amount of time that people spend on the site. In other words, a faster a website means more visitors will stay and stick around longer.
2. **Reducing bandwidth costs** - Bandwidth consumption costs for website hosting is a primary expense for websites. Through caching and other optimizations, CDNs are able to reduce the amount of data an origin server must provide, thus reducing hosting costs for website owners.
3. **Increasing content availability and redundancy** - Large amounts of traffic or hardware failures can interrupt normal website function. Thanks to their distributed nature, a CDN can handle more traffic and withstand hardware failure better than many origin servers.
4. **Improving website security** - A CDN may improve security by providing DDoS mitigation, improvements to security certificates, and other optimizations.

At its core, a CDN is a network of servers linked together with the goal of delivering content as quickly, cheaply, reliably, and securely as possible. In order to improve speed and connectivity, a CDN will place servers at the exchange points between different networks.

These Internet exchange points (IXPs) are the primary locations where different Internet providers connect in order to provide each other access to traffic originating on their different networks. By having a connection to these high speed and highly interconnected locations, a CDN provider is able to reduce costs and transit times in high speed data delivery. Beyond placement of servers in IXPs, a CDN makes a number of optimizations on standard client/server data transfers. CDNs place Data Centers at strategic locations across the globe, enhance security, and are designed to survive various types of failures and Internet congestion. When it comes to websites loading content, users drop off quickly as a site slows down. CDN services can help to reduce load times in the following ways:

- The globally distributed nature of a CDN means reduce distance between users and website resources. Instead of having to connect to wherever a website's origin server may live, a CDN lets users connect to a geographically closer data center. Less travel time means faster service.
- Hardware and software optimizations such as efficient load balancing and solid-state hard drives can help data reach the user faster.
- CDNs can reduce the amount of data that's transferred by reducing file sizes using tactics such as minification and file compression. Smaller file sizes mean quicker load times.
- CDNs can also speed up sites which use TLS/SSL certificates by optimizing connection reuse and enabling TLS false start.

Uptime is a critical component for anyone with an Internet property. Hardware failures and spikes in traffic, as a result of either malicious attacks or just a boost in popularity, have the potential to bring down a web server and prevent users from accessing a site or service. A well-rounded CDN has several features that will minimize downtime:

- Load balancing distributes network traffic evenly across several servers, making it easier to scale rapid boosts in traffic.
- Intelligent failover provides uninterrupted service even if one or more of the CDN servers go offline due to hardware malfunction; the failover can redistribute the traffic to the other operational servers.
- In the event that an entire data center is having technical issues, Anycast routing transfers the traffic to another available data center, ensuring that no users lose access to the website.

3G/4G/5G

Simply, the "G" stands for "GENERATION". While connected to the internet, the speed of the connection depends upon the signal strength that is shown in abbreviations like 2G, 3G, 4G, 5G, etc. on any mobile device. Each generation of wireless broadband is defined as a set of telephone network standards that describe the technological implementation of the system. The aim of wireless communication is to provide high quality, reliable communication just like wired communication and each new generation represents a big leap in that direction. Mobile communication has become more popular in the last few years due to fast reform in mobile technology. For the comparison of 2G, 3G, 4G, and 5G we first need to understand the key features of all these technologies.

The difference between each generation primarily comes down to their capabilities. For example, each generation has made improvements to: Speed (lower latency), Network volume (higher bandwidth) and Accessibility (longer range of service) Of course, this is just scratching the surface of what makes these cellular generations different from one another.

SECOND GENERATION (2G): 2G refers to the second generation of mobile networks based on GSM. The radio signals used by the 1G network were analog, while 2G networks were digital. 2G capabilities were achieved by allowing multiple users on a single channel via multiplexing. During 2G, cellular phones were used for data along with voice. Some of the key features of 2G were:

- Data speeds of up to 64 kbps
- Use of digital signals instead of analog
- Enabled services such as SMS and MMS (Multimedia Message)
- Provided better quality voice calls
- It used a bandwidth of 30 to 200 KHz

THIRD GENERATION (3G) : The 3G standard utilizes Universal Mobile Telecommunications System (UMTS) as its core network architecture. 3G network combines aspects of the 2G network with new technologies and protocols to deliver a significantly faster data rate. By using packet switching, the original technology was improved to allow speeds up to 14 Mbps. It used Wide Band Wireless Network that increased clarity. It operates at a range of 2100 MHz and has a bandwidth of 15-20 MHz. Some of the main features of 3G are:

- Speed of up to 2 Mbps
- Increased bandwidth and data transfer rates
- Send/receive large email messages
- Large capacities and broadband capabilities

FOURTH GENERATION (4G) : The key technologies that have made 4G possible are MIMO (Multiple Input Multiple Output) and OFDM (Orthogonal Frequency Division Multiplexing). The most important 4G standards are WiMAX and LTE. While 4G LTE is a major improvement over 3G speeds, it is technically not 4G. Even after it was widely available, many networks were not up to the required speed of 4G. 4G LTE is a "fourth generation long term evolution", capable of delivering a very fast and secure internet connection. Basically, 4G is the predetermined standard for mobile network connections. 4G LTE is the term given to the path which has to be followed to achieve those predefined standards. Some of the features of 4G LTE are:

- Support interactive multimedia, voice, video.
- High speed, high capacity and low cost per bit (Speeds of up to 20 Mbps or more.)
- Global and scalable mobile networks.
- Ad hoc and multi-hop networks.

Network	Peak speed	Average speed
5G	10 Gbps	400 Mbps
4G	1 Gbps	50 Mbps

FIFTH GENERATION (5G): 5G networks operate on rarely used radio millimeter bands in the 30 GHz to 300 GHz range. Testing of 5G range in mm Wave has produced results approximately 500 meters from the tower. Using small cells, the deployment of 5G with milimetre wave based carriers can improve overall coverage area. Combined with beam forming, small cells can deliver extremely fast coverage with low latency.

Low latency is one of 5G's most important features. 5G uses a scalable orthogonal frequency-division multiplexing (OFDM) framework. 5G benefits greatly from this and can have latency as low as one millisecond with realistic estimates to be around 1 - 10 seconds. 5G is estimated to be 60 to 120 times faster than the average 4G latency.

Active antenna 5G encapsulated with 5G massive MIMO is used for providing better connections and enhanced user experience. Big 5G array antennas are deployed to gain additional beam forming information and knock out propagation challenges that are experienced at mm Wave frequency ranges. Further, 5G networks clubbed with network slicing architecture enables telecom operators to offer on-demand tailored connectivity to their users that is adhered to Service Level Agreement (SLA). Such customised network capabilities comprise latency, data speed, latency, reliability, quality, services, and security. With speeds of up to 10 Gbps, 5G is set to be as much as 10 times faster than 4G.

According to digital trends, 3G can reach network speeds of 7.2 mbps, 4G can reach network speeds of 150 mbps and 5G will eventually reach speeds in excess of 1gbps (with a theoretical maximum of 20gbps) Mileage varies depending on whether the individual browser is stationary or, says, in a moving vehicle. But this is the gist of what a user can achieve speed-wise with each generation.

Bandwidth determines how much traffic a cellular network can handle before the network begins to slow down. Each generation of cellular network has improved upon the bandwidth capabilities of its predecessor. With 3G and 4G bandwidth, a mobile user sees a noticeable deterioration of network speeds during peak hours of the day. 4G LTE and other upgrades to the 4G network have improved bandwidth significantly. 5G will also be able to handle the billions of upcoming connected home devices known collectively as the Internet of Things.

2G vs 3G vs 4G vs 5G

Each generation in some way has improved over its predecessor. There is a lot of ground to compare the cell networks over. The comparison of 2G, 3G, 4G, and 5G clearly shows the differences in the technologies. The comparison of 2G, 3G, 4G, and 5G also makes it evident that 5G is going to be one of the most ambitious leaps in the history of cell network technologies.

Comparison	2G	3G	4G	5G
Introduced in year	1993	2001	2009	2018
Technology	GSM	WCDMA	LTE, WiMAX	MIMO, mm Waves
Access system	TDMA, CDMA	CDMA	CDMA	OFDM, BDMA
Switching type	Circuit switching for voice and packet switching for data	Packet switching except for air interference	Packet switching	Packet switching
Internet service	Narrowband	Broadband	Ultra broadband	Wireless World Wide Web
Bandwidth	25 MHz	25 MHz	100 MHz	30 GHz to 300 GHz
Advantage	Multimedia features (SMS, MMS), internet access and SIM introduced	High security, international roaming	Speed, high speed handoffs, global mobility	Extremely high speeds, low latency
Applications	Voice calls, short messages	Video conferencing, mobile TV, GPS	High speed applications, mobile TV, wearable devices	High resolution video streaming, remote control of vehicles, robots, and medical procedures

CHALLENGES IN IOT

The Internet of Things (IoT) has fast grown to be a large part of how human beings live, communicate and do business. There are various types of challenges in front of IoT.

1. **Lack of encryption -** Although encryption is a great way to prevent hackers from accessing data, it is also one of the leading IoT security challenges. These drives like the storage and processing capabilities that would be found on a traditional computer. The result is an increase in attacks where hackers can easily manipulate the algorithms that were designed for protection.
2. **Insufficient testing and updating -**: With the increase in the number of IoT(internet of things) devices, IoT manufacturers are more eager to produce and deliver their device as fast as they can without giving security too much of although. Most of these devices and IoT products do not get enough testing and updates and are prone to hackers and other security issues.
3. **Brute forcing and the risk of default passwords -**: Weak credentials and login details leave nearly all IoT devices vulnerable to password hacking and brute force. Any company that uses factory default credentials on their devices is placing both their business and its assets and the customer and their valuable information at risk of being susceptible to a brute force attack.
4. **IoT Malware and ransomware -**:Ransomware uses encryption to effectively lock out users from various devices and platforms and still use a user's valuable data and info. Example - A hacker can hijack a computer camera and take pictures. By using malware access points, the hackers can demand ransom to unlock the device and return the data.
5. **IoT botnet aiming at crypto currency -**: IoT botnet workers can manipulate data privacy, which could be massive risks for an open Crypto market. The exact value and creation of crypto currencies code face danger from mal-intentioned hackers. The blockchain companies are trying to boost security. Blockchain technology itself is not particularly vulnerable, but the app development process is.

Design challenge in IoT :

6. **Battery life is a limitation -** Issues in packaging and integration of small-sized chip with low weight and less power consumption.
7. **Increased cost and time to market -** Embedded systems are lightly constrained by cost. The need originates to drive better approaches when designing the IoT devices in order to handle the cost modelling or cost optimally with digital electronic components. Designers also need to solve the design time problem and bring the embedded device at the right time to the market.

8. **Security of the system** – Systems have to be designed and implemented to be robust and reliable and have to be secure with cryptographic algorithms and security procedures. It involves different approaches to secure all the components of embedded systems from prototype to deployment.

Deployment challenges in IoT :

9. **Connectivity**: It is the foremost concern while connecting devices, applications and cloud platforms. Connected devices that provide useful front and information are extremely valuable. But poor connectivity becomes a challenge where IoT sensors are required to monitor process data and supply information.
10. **Cross platform capability** – IoT applications must be developed, keeping in mind the technological changes of the future. Its development requires a balance of hardware and software functions. It is a challenge for IoT application developers to ensure that the device and IoT platform drivers the best performance despite heavy device rates and fixings.
11. **Data collection and processing** – In IoT development, data plays an important role. What is more critical here is the processing or usefulness of stored data. Along with security and privacy, development teams need to ensure that they plan well for the way data is collected, stored or processed within an environment.
12. **Lack of skill set** – All of the development challenges above can only be handled if there is a proper skilled resource working on the IoT application development. The right talent will always get you past the major challenges and will be an important IoT application development asset.

UNIT – 2

Connectivity: IoT Network Configurations

Network configuration is the process of assigning network settings, policies, flows, and controls. In a virtual network, it's easier to make network configuration changes because physical network devices appliances are replaced by software, removing the need for extensive manual configuration. Network configuration can also be automated and managed via a centralized configuration manager network configuration manager, further reducing manual IT workload and making it easier to:

- Maintain a network
- Make configuration changes
- Relaunch devices
- Track and report data

Some network configuration basics include switch/router configuration, host configuration, software and firewall configuration, and network topology which can be controlled through rest APIs.

The right network configuration is essential to supporting the flow of traffic through a network, and it can also support and enhance network security and improve network stability. In addition, the use of network configuration management manager and or configuration tools can provide a number of benefits, including:

- Automated data tracking and reporting, allowing administrators to spot any configuration changes and potential threats or issues
- An easy way to make bulk changes, such as a blanket password change in a situation where passwords are compromised
- The means to swiftly roll back network settings to a previous configuration
- Reduced downtime, thanks to increased visibility and the ability to quickly identify changes
- Streamlined maintenance and repair of network devices (physical or virtual) and connections
- The ability to relaunch a device when it fails, thanks to centralized storage management of device configurations

Zero-configuration networking: Zero-configuration networking refers to a set of technologies that allow network administrators to set up a network and connect devices without having to manually configure each device's network settings. This is particularly useful for allowing end users to easily connect to the network. However, for an administrator of an enterprise network, there are advantages to actively configure and monitor the network rather than relying on default settings.

NETWORK TOPOLOGIES

Different types of network configuration in computer networks are commonly referred to as network topologies. A network topology describes how the nodes or devices (physical or virtual) in a network are arranged and how they communicate with each other. Network topology can be physical (referring to where physical devices are placed in relation to each other) or logical (referring to how data is transmitted through the network, including any virtual or cloud resources). When choosing a network topology, an organization must consider the size of its network, its performance requirements and the flow of its traffic, among other factors. Common network topologies include:

- **Bus:** Every node in the network is connected along a linear path. This simple topology is used most often for small networks.
- **Ring:** Nodes are connected in a loop, and traffic may flow in one direction or in both directions. Ring networks tend to be cost-effective, but not as scalable or stable as other network topologies.
- **Star:** A central node connects to all other nodes in the network. This is a common and stable topology that's often used for local area networks (LANs).
- **Mesh:** Nodes are linked in such a way that multiple paths between nodes are possible. This type of network topology increases the resiliency of the network, but also increases cost. A network may be fully meshed (all nodes connecting to all other nodes) or partially meshed (only some nodes having multiple connections to other nodes).
- **Spine-Leaf (Tree):** Multiple star topologies are connected together in a larger star configuration.
- **Hybrid:** A combination of other topologies are used together within one network.

When setting up a network switch and router, it's important to customize settings and apply all necessary configurations to ensure that your network will work properly. Some of the configurable settings on a network switch and router include:

- **IP address**—for identification
- **Password**—for added security
- **Channel and band selection**—to improve performance
- **Default gateway**—to make the device visible to network management tools
- **Neighbor discovery**—for added visibility
- **Correct time**—for proper troubleshooting and detailed error logs

A network configuration manager is the easiest way to perform network switch configuration and apply these settings consistently to every device on your enterprise network.

The success or failure of Internet of Things (IoT) projects is dependent on many things, but without a reliable connection between devices, sensors and your IoT platform, your project won't even get off the ground. However, connectivity is not a matter of simply choosing a preferred wireless technology. It's equally important to understand the requirements of your application—and then choose the network technology that's the best fit.

Below are the four general types of IoT networks and some of the top IoT wireless protocols within each category. Keep in mind there's no "winner" here; all the below technologies have advantages and disadvantages in terms of cost, power, and battery size—three key differentiating features of IoT networks. **No matter which technology you choose, you'll inevitably have to make a trade-off somewhere.** Negotiating the right balance of the three elements for your project will ensure it starts off on the right foot.

1. Cellular

Cellular networks use the same mobile networks as smartphones to allow IoT devices to communicate. Because these networks were originally designed for power-hungry devices like smartphones, they weren't always considered the best fit for IoT devices. Eventually, the cellular industry developed new technologies that were more appropriate for IoT use cases. Today, this type of wireless network is very popular, and is considered a reliable and secure method of IoT connectivity. However, cell connectivity often isn't available in the places that most need monitoring sensors—for example, inside utility closets, elevator shafts, basements, etc. (Another IoT wireless technology class, LPWAN, might be a better fit for these locations.) And even though cellular connectivity is now less expensive and more power efficient than traditional telecom standards, cellular-connected IoT devices still require a great deal more power and energy than some other types of wireless networks.

Two cellular IoT wireless protocols currently competing for dominance are **LTE-M** and **Narrowband IoT (NB-IoT).** LTE-M is a great option for IoT connectivity if you're willing to pay the price, and if your use case requires low power. NB-IoT is somewhat less costly than LTE-M and uses less battery power, but there's not enough coverage everywhere to reliably deploy an NB-IoT solution *yet*.

2. Local and Personal Area Networks (LAN/PAN)

Networks that cover fairly short distances are called personal area networks (PAN) and local area networks(LAN). PAN and LAN networks are considered to be fairly cost-effective, but the transfer of data can sometimes be unreliable.

Wireless personal and local area network technologies that are commonly incorporated into IoT connectivity solutions are **WiFi** and **Bluetooth.** WiFi can be used for applications that run in a local environment, or in a distributed setting if there are multiple access points integrated into a larger network. One downside to WiFi is that it works only if the signal is strong and you're close to the access point. Also, WiFi is generally more power-hungry than people think, but it is possible to operate it in a way that's a little more power-efficient (for example, your device only connects periodically to send data, then goes back to sleep).

Bluetooth Low Energy (BLE) is a more energy-efficient wireless network protocol—if you're not receiving data constantly, a single battery running BLE could last up to five years. However, compared to WiFi it is slower to transmit and is more limited in the amount of data it is capable of sending. Both WiFi and Bluetooth are easy to connect in most cases, although WiFi does have some security challenges that may be difficult to overcome.

3. Low Power Wide Area Networks (LPWAN)

IoT devices that run on LPWANs send small packets of information infrequently and over long distances. This type of wireless network was developed in response to the early challenges of cellular connectivity. Proponents of LPWAN position it as longer-range than WiFi and Bluetooth, but using less power than cellular. Sigfox built the first LPWAN network in France and is considered the driving force behind its growth.

A well-known and commonly used IoT network protocol in this category is **LoRaWAN** (long range wireless area network), which runs on the LoRa (long range) communication network. Advantages of LoRaWAN for IoT devices are its low power requirement (for long battery life) and relatively low-cost chipsets. Plus, under the right conditions, a single base station or gateway running on a long-range network is capable of providing service to a very large area—a few kilometers in dense urban areas and up to 15–30 kilometers in rural areas.

4. Mesh Networks

Mesh networks are best described by their connectivity configuration—how the components communicate with each other. In mesh networks, all the sensor nodes cooperate to distribute data amongst each other to reach the gateway. (A star topology, in contrast, is where all sensor nodes communicate to a central hub.)

Zigbee is one example of an IoT wireless network technology. Mesh networks are very short range and may require extra sensors throughout a building or the use of repeaters to get the coverage your application needs. Also, the nature of the way these networks communicate can result in high power consumption, especially if you need instant messaging, such as for a smart lighting application. (IoT applications that require only occasional information updates use less power.) However, mesh networks are also fairly robust, able to find the fastest and most reliable paths to send data, and easy to install, making them a popular choice for in-building use.

Gateway Prefix Allotment

IPv4, IPv6, IPv4 versus IPv6,

An IP stands for internet protocol. An IP address is assigned to each device connected to a network. Each device uses an IP address for communication. It also behaves as an identifier as this address is used to identify the device on a network. It creates a virtual connection between the source and the destination. We can also define an IP address as a numeric address assigned to each device on a network. An IP address is assigned to each device so that the device on a network can be identified uniquely.

IPv4 is a version 4 of IP. It is a current version and the most commonly used IP address. It is a 32-bit address written in four numbers separated by 'dot', i.e., periods. This address is unique for each device.

For example, **66.94.29.13**

The above example represents the IP address in which each group of numbers separated by periods is called an Octet. Each number in an octet is in the range from 0-255. This address can produce 4,294,967,296 possible unique addresses.

Computers do not understand the IP addresses in the standard numeric format as the computers understand the numbers in binary form only. The binary number can be either 1 or 0. The IPv4 consists of four sets, and these sets represent the octet. The bits in each octet represent a number. Each bit in an octet can be either 1 or 0. If the bit the 1, then the number it represents will count, and if the bit is 0, then the number it represents does not count.

Initially, the standard defined the first octet as a network identifier, but with only 256 unique values, the number of available networks quickly ran out. Several different changes made over the years have allowed for the extension of IPv4's life. First came the division of the available addresses into five classes: A, B, C, D, and E. The class system defined which class a network belongs in based on its first octet.

- **Class A** network's first octet begins with 0. The first octet identifies the network. Class A supports 127 networks, each with 16 million hosts.
- **Class B** network's first octet begins with 10. The first and second octets identify the network. Class B supports 16,000 networks, each with 65,000 hosts.
- **Class C** network's first octet begins with 110. The first three octets identify the network. Class C supports 2 million networks, each with 254 hosts.
- **Class D** network's first octet begins with 1110. Class D is reserved for multicast groups.
- **Class E** network's first octet begins with 1111. Class E is reserved for future use.

Each class used a different number of bits to identify the network affecting how many networks and hosts each class could accommodate. For example, Class C's first three octets described the network, while the fourth described the host on the network. Later the IETF replaced the class system, dubbed "classful," with subnet masks that allowed for the dispersal of addresses on any address-bit boundary.

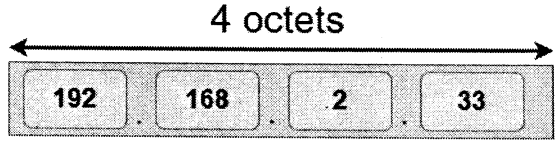
Drawback of IPv4 Currently, the population of the world is 7.6 billion. Every user is having more than one device connected with the internet, and private companies also rely on the internet. As we know that IPv4 produces 4 billion addresses, which are not enough for each device connected to the internet on a planet. Although the various techniques were invented, such as variable- length mask, network address translation, port address translation, classes, inter-domain translation, to conserve the bandwidth of IP address and slow down the depletion of an IP address. In these techniques, public IP is converted into a private IP due to which the user having public IP can also use the internet. But still, this was not so efficient, so it gave rise to the development of the next generation of IP addresses, i.e., **IPv6.**

IPv6 - IPv6 is the next generation of IP addresses. The main difference between IPv4 and IPv6 is the address size of IP addresses. The IPv4 is a 32-bit address, whereas IPv6 is a 128-bit hexadecimal address. IPv6 provides a large address space, and it contains a simple header as compared to IPv4. It provides transition strategies that convert IPv4 into IPv6, and these strategies are as follows:

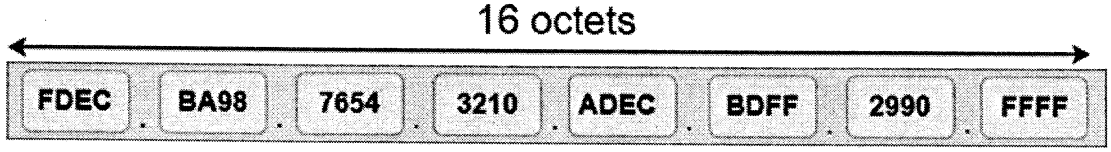
- Dual stacking: It allows us to have both the versions, i.e., IPv4 and IPv6, on the same device.
- Tunneling: In this approach, all the users have IPv6 communicates with an IPv4 network to reach IPv6.
- Network Address Translation: The translation allows the communication between the hosts having a different version of IP.

This hexadecimal address contains both numbers and alphabets. Due to the usage of both the numbers and alphabets, IPv6 is capable of producing over 340 undecillion (3.4*1038) addresses. IPv6 is a 128-bit hexadecimal address made up of 8 sets of 16 bits each, and these 8 sets are separated by a colon. In IPv6, each hexadecimal character represents 4 bits. So, we need to convert 4 bits to a hexadecimal number at a time

The address format of IPv4:



The address format of IPv6:



The above diagram shows the address format of IPv4 and IPv6. An IPv4 is a 32-bit decimal address. It contains 4 octets or fields separated by 'dot', and each field is 8-bit in size. The number that each field contains should be in the range of 0-255. Whereas an IPv6 is a 128-bit hexadecimal address. It contains 8 fields separated by a colon, and each field is 16-bit in size.

Differences between IPv4 and IPv6

	Ipv4	Ipv6
Address length	IPv4 is a 32-bit address.	IPv6 is a 128-bit address.
Fields	IPv4 is a numeric address that consists of 4 fields which are separated by dot (.).	IPv6 is an alphanumeric address that consists of 8 fields, which are separated by colon (:).
Classes	IPv4 has 5 different classes of IP address that includes Class A, Class B, Class C, Class D, and Class E.	IPv6 does not contain classes of IP addresses.
Number of IP address	IPv4 has a limited number of IP addresses.	IPv6 has a large number of IP addresses.
VLSM	It supports VLSM (Virtual Length Subnet Mask). Here, VLSM means that Ipv4 converts IP addresses into a subnet of different sizes.	It does not support VLSM.
Address configuration	It supports manual and DHCP configuration.	It supports manual, DHCP, auto-configuration, and renumbering.
Address space	It generates 4 billion unique addresses	It generates 340 undecillion unique addresses.
End-to-end connection integrity	In IPv4, end-to-end connection integrity is unachievable.	In the case of IPv6, end-to-end connection integrity is achievable.
Security features	In IPv4, security depends on the application. This IP address is not developed in keeping the security feature in mind.	In IPv6, IPSEC is developed for security purposes.
Address representation	In IPv4, the IP address is represented in decimal.	In IPv6, the representation of the IP address in hexadecimal.
Fragmentation	Fragmentation is done by the senders and the forwarding routers.	Fragmentation is done by the senders only.
Packet flow identification	It does not provide any mechanism for packet flow identification.	It uses flow label field in the header for the packet flow identification.

Checksum field	The checksum field is available in IPv4.	The checksum field is not available in IPv6.
Transmission scheme	IPv4 is broadcasting.	On the other hand, IPv6 is multicasting, which provides efficient network operations.
Encryption and Authentication	It does not provide encryption and authentication.	It provides encryption and authentication.
Number of octets	It consists of 4 octets.	It consists of 8 fields, and each field contains 2 octets. Therefore, the total number of octets in IPv6 is 16.

ARIN
American Registry for Internet Numbers

During the beginning stages of the internet, developers created a series of numbers, Internet Protocol version 4 (IPv4), to effectively and efficiently network between computers. A few years later the Internet Assigned Numbers Authority (IANA), now a department of Internet Corporation for Assigned Names and Numbers (ICANN), was created to help manage IPv4 IP addresses. As the internet continued to grow so did the need for IPs and organizations to manage their distribution. From the 1980s through the 2000s, a number of organizations were created to do just that. In 1997, the American Registry for Internet Numbers (ARIN), one of five Regional Internet Registries (RIRs), was created as a nonprofit to issue Internet number resources in its region that today includes parts of the Caribbean, Canada, and the US. ARIN was also created to facilitate consensus-based policies and even promote the advancement of the Internet through education and outreach (ARIN, 2013). Today we rely on ARIN to implement the policies developed through open and ongoing community input. In other words, ARIN is the organization with the authority to carry-out the policies we created around Internet numbers – whatever they say, goes!

IPv4 subnet mask

The concept of a subnet mask is simple. You have a network and you have hosts on the network (anything with an IP address is a host). The subnet mask determines what portion of the TCP/IP address represents your network and what portion can be used for your hosts. The network number represents the street I live on, and the host portion is used for the numbers on all the houses on my street. A subnet mask of 255.255.255.0 means that the first three octets of the address will be used for the network, and thus our network number is 192.168.1. This means we can have 254 computers on this network, because the fourth octet is not being used by the network portion of the address. We know this because of the 0 in the subnet mask

(255.255.255.0). We call each of the number sections an octet because we think of them in binary, and there are eight possible bits in each section. Eight bits is an octet. 11111111 in binary is 255 in decimal. So our decimal subnet mask 255.255.255.0 displayed in binary is going to be: 11111111.11111111.11111111.00000000 If you count all the ones, you will find that there are 24 of them. Now look at the subnet mask examples again. 192.168.1.0/255.255.255.0 192.168.1.0/24 Do you see why both subnet masks are the same? The number 24 is the number of bits used in the network portion of the address, and is shorthand for writing the address/subnet mask combination. It becomes important to understand this when you start dividing your network into multiple sub networks.

RPL

RPL (Routing Protocol for Low-Power and Lossy Networks) is a routing protocol for wireless networks with low power consumption and generally susceptible to packet loss. It is a proactive protocol based on distance vectors and operates on IEEE 802.15.4, optimized for multi-hop and many-to-one communication, but also supports one-to-one messages. This protocol is specified in RFC 6550 with special applications in RFCs 5867, 5826, 5673 and 5548. RPL can support a wide variety of link layers, including those with limitations, with potential losses or that are used in devices with limited resources. This protocol can quickly create network routes, share routing knowledge and adapt the topology in an efficient way.

Low Power and Lossy Networks: Low power and Lossy Networks (LLNs) are a class of network in which both the routers and their interconnect are constrained: LLN routers typically operate with constraints on (any subset of) processing power, memory and energy (battery), and their interconnects are characterized by (any subset of) high loss rates, low data rates and instability. LLNs are comprised of anything from a few dozen and up to thousands of LLN routers, and support point-to-point traffic (between devices inside the LLN), point-to-multipoint traffic (from a central control point to a subset of devices inside the LLN) and multipoint-to-point traffic (from devices inside the LLN towards a central control point).

The Routing Protocol for Low Power and Lossy Networks feature specifies the IPv6 Routing Protocol for LLNs (RPL), thereby providing a mechanism whereby multipoint-to-point traffic from devices inside the LLN towards a central control point, and point-to-multipoint traffic from the central control point to the devices inside the LLN, is supported. Point-to-point traffic is also supported.

RPL Routing Attributes (Metrics and Constraints)

RPL makes use of a wide set of routing attributes that can be used either as constraints or metrics as against IGPs such as IS-IS or OSPF. When used as a constraint, the routing attribute allows pruning the links and nodes from candidate paths that do not respect the constraint; when used as a metric the routing attribute is used to determine the least cost path. Additionally, routing metrics can either be used as aggregated (e.g. path cost equal to the sum of the link metrics) or recorded in which case metrics of all links along the path are recorded and announced along with the path by RPL. Recorded metrics are particularly useful when aggregating metrics implies loosing relevant information for path selection.

Metrics can also be local (exchanged between two neighbors) or global (propagated along the path as the path cost).RPL uses dynamic metrics, which implies the use of low pass filters to preserve routing stability, avoid temporary loops and bind control traffic by limiting the number of advertised RPL messages due to link metrics updates and consequently path cost that may have a global impact by recomputing the routing topology.

Restrictions for Routing Protocol for Low Power and Lossy Networks

- RPL can be enabled only on the main interface.
- RPL may not work if multiple links have same link local address.
- The probing parameters are not configurable in root template in this implementation. The probing parameters are required in root template also.
- The following RPL features listed in RFC6550 are not supported:
    - Secure RPL Control Messages.
    - Non-storing mode.
    - Multicast DAO
    - Unicast DIS
    - Unicast DIO
    - Reception of DAO-ACK
    - Virtual DODAG root
- Metrics and constraints—Implements a subset of these metrics and constraints only.
    - DIO Destination prefix
    - Non-storing mode for DAO routes—Supports processing of DAO prefixes received from nonstoring nodes but always operate in a full route storing mode.
    - No support for adding tags to DAO prefixes using DAO suboptions.
    - Local confidence
- Interface level link check for Expected Transmission Count or latency calculation—Uses layer-3 probe process for calculating link.
- You cannot change the values of the Instance-ID, DODAG ID, and OCP options for an RPL after the RPL template is configured. If you want to change the options, you must delete the template from all interfaces.
- The no dao enable command cannot be used when the template is active.

Protocol configuration: RPL creates a topology similar to a tree (DAG or directed acyclic graph). Each node within the network has an assigned rank (Rank), which increases as the teams move away from the root node (DODAG). The nodes resend packets using the lowest range as the route selection criteria.

Three control messages are defined in ICMPv6 via RFC 4443 :

1. DIS (information request DODAG): Used to request information from nearby DODAG, analogous to router request messages used to discover existing networks.
2. DIO (object of information of the DAG): Message that shares information from the DAG, sent in response to DIS messages, as well as used periodically to refresh the information of the nodes on the topology of the network.

3. DAO (object of update to the destination): Sent in the direction of the DODAG, it is a message sent by the teams to update the information of their "parent" nodes throughout the DAG.

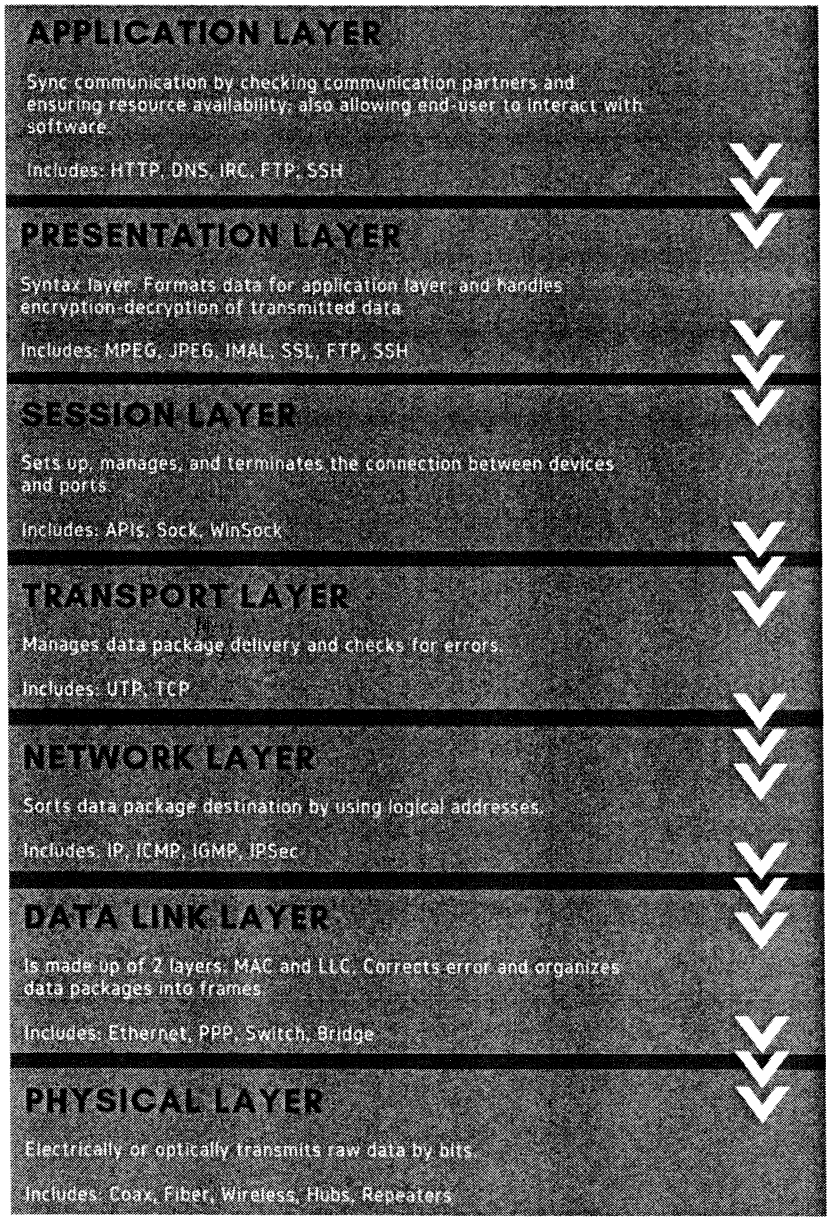
Implementation of the RPL protocol : The implementation of the RPL protocol occurs in wireless sensors and networks, the most used operating system for its implementation is Contiki which is a small open source operating system developed for use in a number of small systems ranging from 8-bit computers to integrated systems on microcontrollers, including sensor network nodes.

The RPL protocol is implemented in other operating systems, such as:

- **LiteOS** is an edition of the Zorin OS operating system designed for low-resource computers, developed in principle for calculators, but which has also been used for sensor networks.
- **TinyOs** was the first operating system oriented to the wireless sensor network (WSN), works through events and guided tasks, and uses an extension of the C language, called nesC. TinyOS is implemented as a set of cooperating tasks and processes and it determines the priorities between tasks and events.
- **T-Kernel** is an operating system that accepts applications as executable images in basic instructions. Therefore, it will not matter if it is written in C++ or Assembly language.
- **EyeOS** is defined as a desktop environment based on Web, which allows monitoring and access to a remote system through a simple search engine.
- **RIOT** is a small operating system for networked, memory-constrained systems with a focus on low-power wireless Internet of Things (IoT) devices.

Data Protocol: MQTT, CoAP, AMQP,

In the process of establishing your network for IoT applications, you will need to consider a few things. One is the type of network to be used: LPWAN (LoRaWAN, NB-IoT), PAN (Bluetooth), or LAN (WiFi). Another thing to be considered is the network protocol to be implemented with it. This protocol will manage the transmission of data and control the security of the delivery. The general formula for how telecommunications are established is represented globally by the OSI (Open Systems Interconnection) Model. The model divides the structure into 7 different layers, as shown in the figure below. Today we are discussing some of the most prominent protocols used by IoT devices.



The uppermost layers, layer 5-7 (Session, Presentation, and Application Layer), are constructed to deal with data processes. The application layer, which stands as the last layer,

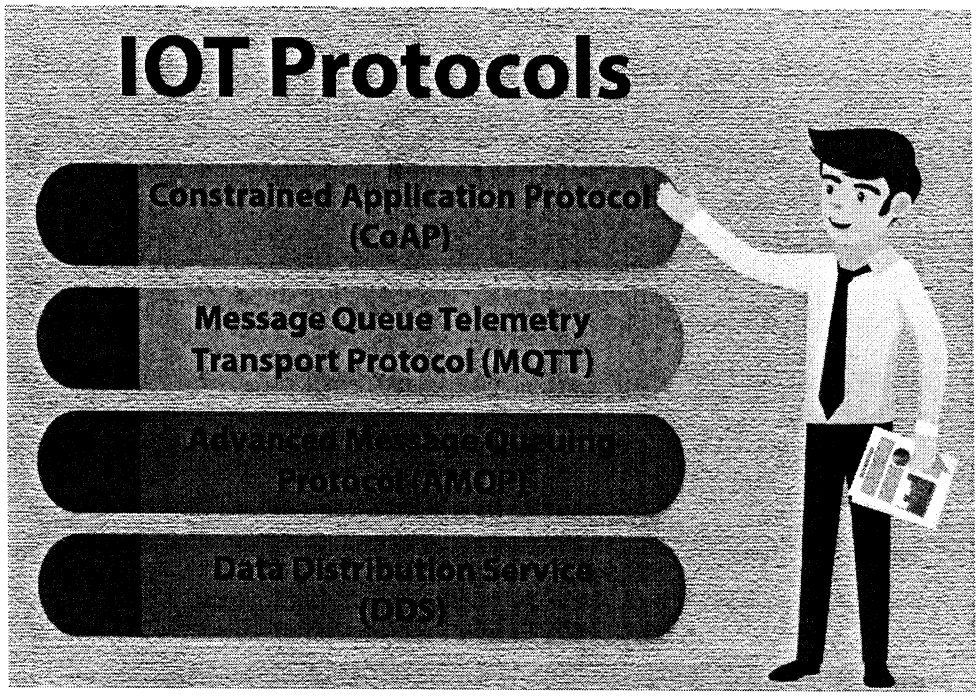
would be the only layer to interact with the end-users. The application layer protocols commonly used for IoT applications include:

- CoAP
- MQTT
- AMQP
- DDS
- XMPP

Another application layer protocol that is commonly used in telecommunication systems is HTTP or HyperText Transport Protocol. However, it is arguably not suitable for IoT applications due to its characteristics:

- Only able to support one-to-one communication i.e. would only serve one client to one broker/server.
- High power consumption, for every one-to-one connection established, an underlying TCP connection should be open as well. This will put a heavy load on the server and hence, consume more power than an edge device can sustain.
- Operates on a request-based response. Sensors, which act as a continuously reporting client (subscriber) in an IoT system typically don't have any computational feature implemented together with it.

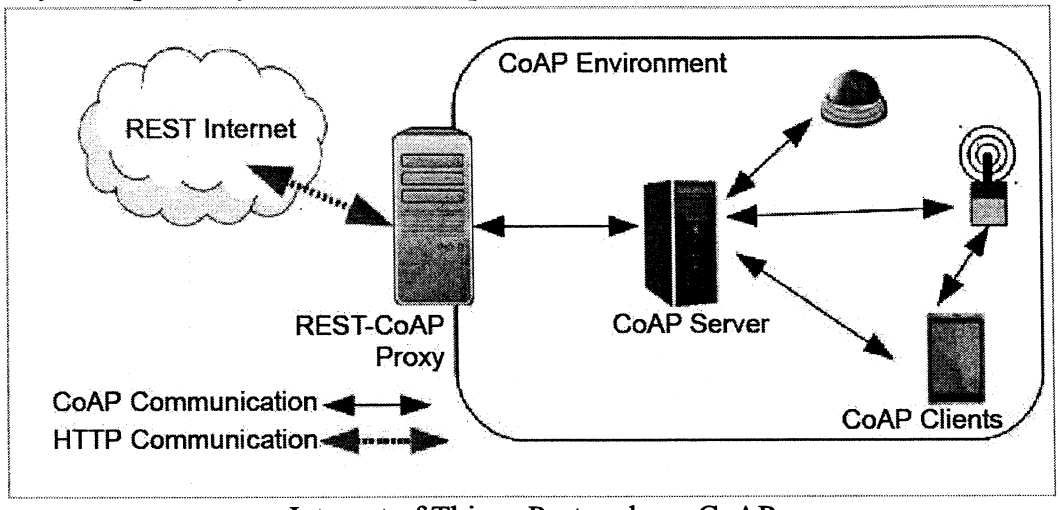
4 Major IoT Protocols — MQTT, CoAP, AMQP, DDS



1. Constrained Application Protocol (CoAP)

CoAP is an internet utility protocol for restricted gadgets. It's mainly designed to use among gadgets on the equal restricted community, among gadgets and general nodes at the internet,

and among gadgets on different restrained networks. This protocol is particularly designed for IoT systems primarily based on HTTP protocols.



Internet of Things Protocols — CoAP

Coap makes use of the UDP protocol for lightweight implementation. It also uses restful architecture, which may be very just like the HTTP protocol. It's mainly used inside mobiles and social community primarily based programs and gets rid of ambiguity by means of the usage of the HTTP get, put up, placed and delete strategies. Other than communicating IoT data, CoAP develop alongside dtls for the cozy change of messages. It makes use of dtls for the cozy switch of statistics within the slipping layer.

The CoAP was designed to enable communication using constrained nodes and networks for IoT applications, and connecting them to the internet. Since IoT systems would require numerous nodes, where each node might not have high memory for cost efficiency, CoAP was designed to allow data transmission to occur with minimum resources with only a 4-byte fixed header. CoAP uses UDP exclusively to enhance its compact communication process. This being said, the communication process between devices would be able to occur without needing any connection to be established prior.

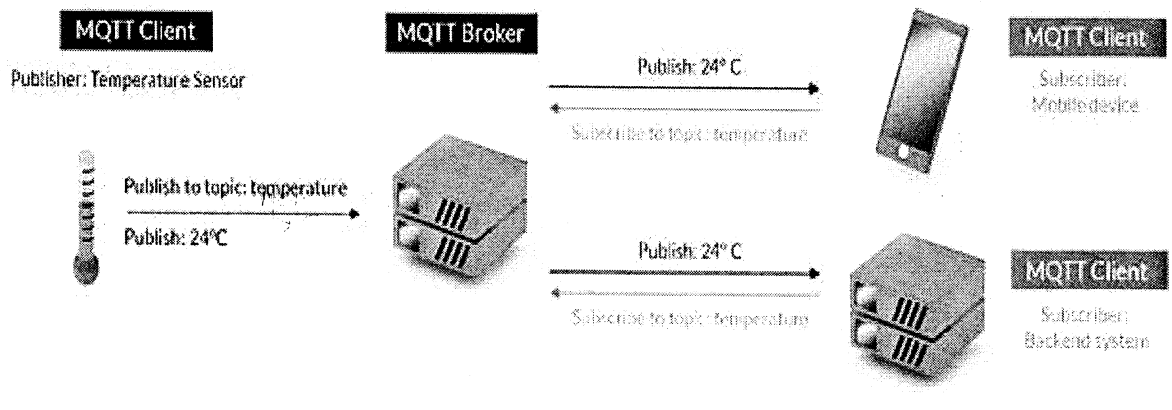
CoAP, like HTTP, is standardized by the IETF Constrained RESTful Environments (CoRe) Working Group. The server provides resources in the form of URLs, and the clients will access the resources by using several specified functions like GET, PUT, POST, and DELETE.

2. Message Queue Telemetry Transport Protocol (MQTT)

MQTT (Message Queue Telemetry Transport), a messaging protocol, developed by IBM and Arcom in 1999. It was developed as an open OASIS standard and an ISO recommended protocol, the MQTT was aimed to operate on data transmissions with a small bandwidth and minimum resources (e.g. on microcontrollers). It's far normally used for distant tracking in IoT. Its primary challenge is to gather statistics from many gadgets and delivery of its infrastructure. MQTT connects gadgets and networks with packages and middleware. A hub-and-spoke structure is herbal for MQTT. All the devices hook up with concentrator servers. MQTT protocols provides services on the top of TCP to offer easy and dependable streams of information.

These IoT protocols include 3 foremost additives: subscriber, publisher, and dealer. The writer generates the information and transmits the facts to subscribers through the dealer. The dealer guarantees safety by means of move-checking the authorization of publishers and subscribers. It usually uses the TCP/IP protocol suite, which runs by first establishing connections, then allows multiple exchanges of data until one party finally disconnects itself.

The MQTT technology runs using the MQTT Publish/Subscribe architecture and establishes the network from 2 different component categories: Clients (Publisher and Subscriber) and Brokers.



MQTT Architecture

The Publish/Subscribe architecture first divides the network's client devices into 2 categories, publishers and subscribers. These publishers are devices or nodes that input data to the system. On the other hand, the subscribers are the end devices or nodes that receive the data.

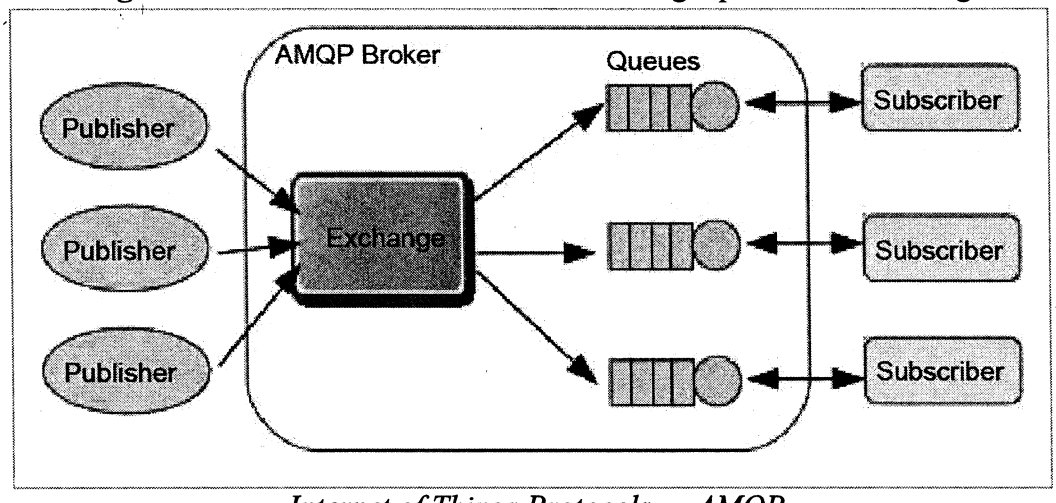
The data sent are in the form of different topics, in which the broker sorts the topic sent from the publisher, and sends it out to the appropriate client which has previously subscribed to the said topic.

3. Advanced Message Queuing Protocol (AMQP)

This was evolved by John O'Hara at JP Morgan Chase in London. AMQP is a software layer protocol for message-oriented middleware environments. It supports reliable verbal exchange through message transport warranty primitives like at-most-once, at least once and exactly as soon as shipping. AMQP, like MQPP, is a message queueing protocol. This means that the subscriber and publisher of the system communicate by sending and requesting messages from a 'message queue'. This standard satisfies the need for IoT applications on asynchronous communication, ensuring that the system is flexible enough to enable communication across numerous 'things'.

The MQPP itself is an open OASIS standard protocol with ISO and IEC certification. It operates in a similar fashion to how messaging operators operate: a broker assigns received and transmitted messages, and several clients generate or receive the subscribed messages. This version has the following three additives, which might link into processing chains in the server to create the favored capability.

- **Exchange:** Receives messages from publisher primarily based programs and routes them to 'message queues'.
- **Message Queue:** Stores messages until they may thoroughly process via the eating client software.
- **Binding:** States the connection between the message queue and the change.

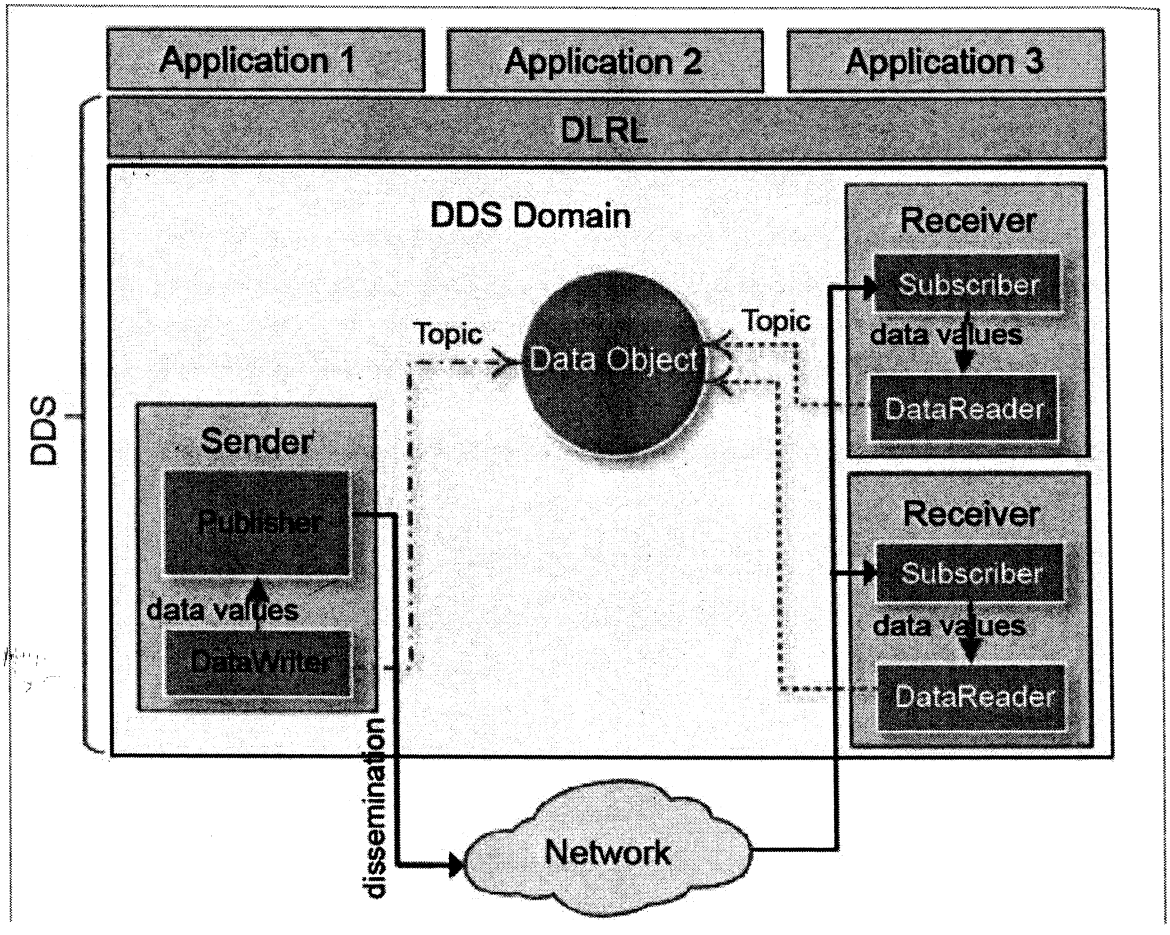


Internet of Things Protocols — AMQP

OASIS, as its publisher, had stated that AMQP is an appropriate protocol for business communications in 2015. Due to its reliable, secure, interoperable, open, and standard properties, along with its low overhead characteristics, AMQP has become a good solution for IoT applications. Since AMQP serves as a more advanced protocol than MQPP, some might choose AMQP to be a better solution for their IoT systems. Despite this, it's worth noting that the AMQP would require a more complicated solution for wired connection, but it all depends on how you set up your system.

4. Data Distribution Service (DDS)

DDS is the first open interoperable middleware protocol, developed by the Object Management Group (OMG). Its operation claims to provide a secure and real-time data distribution. Like MQTT, DDS works in a Publisher/Subscriber architecture. However, the protocol doesn't implement the use of Brokers together with its Clients, hence its topic distribution occurs across its Global Data Space (GDS) by applying a QoS (Quality of Service) contract system. The GDS acts as a 'memory' during DDS transmission application. However, it's actually not a physical memory in the DDS server, it's just a virtual concept. The GDS is actually the combination of local stores in nodes connected to the system.



Internet of Things Protocols — DDS

It enables scalable, real-time, reliable, excessive-overall performance and interoperable statistics change via the submit-subscribe technique. DDS makes use of brokerless architecture and of multicasting to convey high-quality QoS to applications. DDS can deploy in platforms ranging from low-footprint devices to the cloud and supports green bandwidth usage in addition to the agile orchestration of system additives.

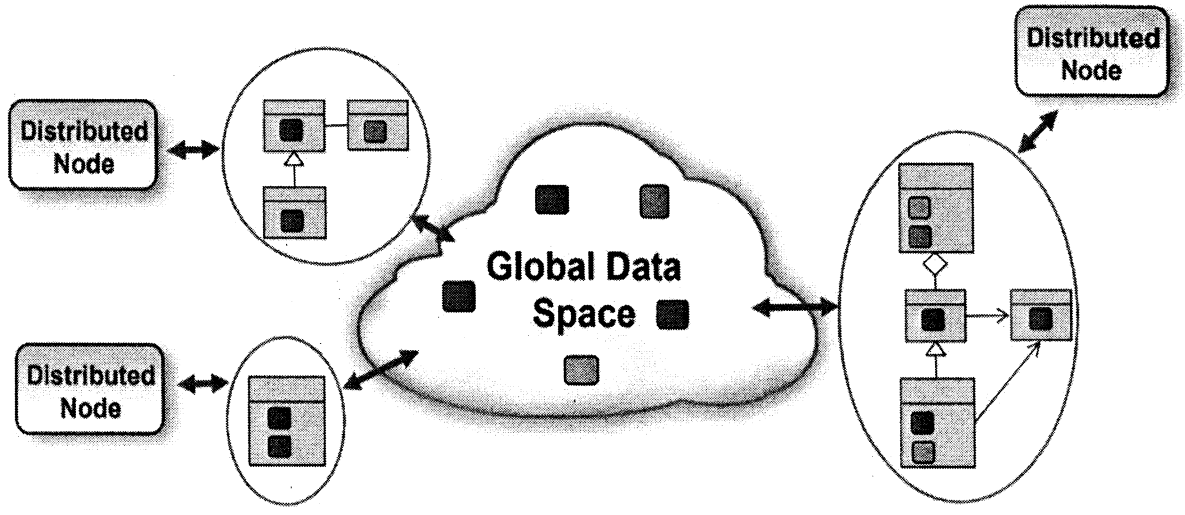
The DDS — IoT protocols have fundamental layers: facts centric submit-subscribe (dcps) and statistics-local reconstruction layer (dlrl). Dcps plays the task of handing over the facts to

subscribers, and the dlrl layer presents an interface to dcps functionalities, permitting the sharing of distributed data amongst IoT enabled objects.

5. XMPP (Extensible Messaging and Presence Protocol)

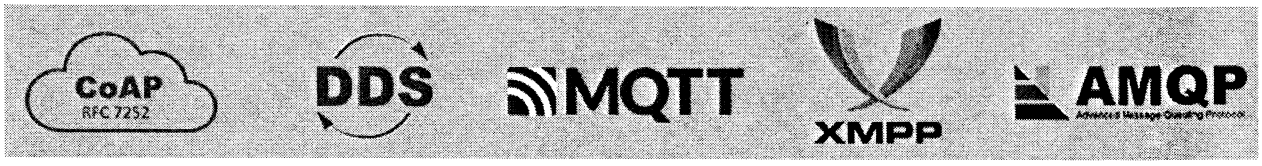
XMPP, developed in the year 1999, is an open-source protocol that was based on XML (Extensive Markup Language). Hence, it supports rapid structured data exchange between two or more network entities and enables the addition of extension for operation. The structure of an XMPP network is very similar to that of an email, giving the protocol its decentralized property. This means that anyone could establish their own server with the protocol anywhere.

The XMPP protocol is used commonly for instant messaging purposes, including voice and video calls, multi-person chats, etc. However, the protocol also serves IoT function properly as it's flexible for connection protocols, secure, and enables middleware communication without requiring human intervention. A few applications of IoT with XMPP include the Google Cloud Print and Logitech Harmony Hub (home automation and media control).



XMPP Architecture

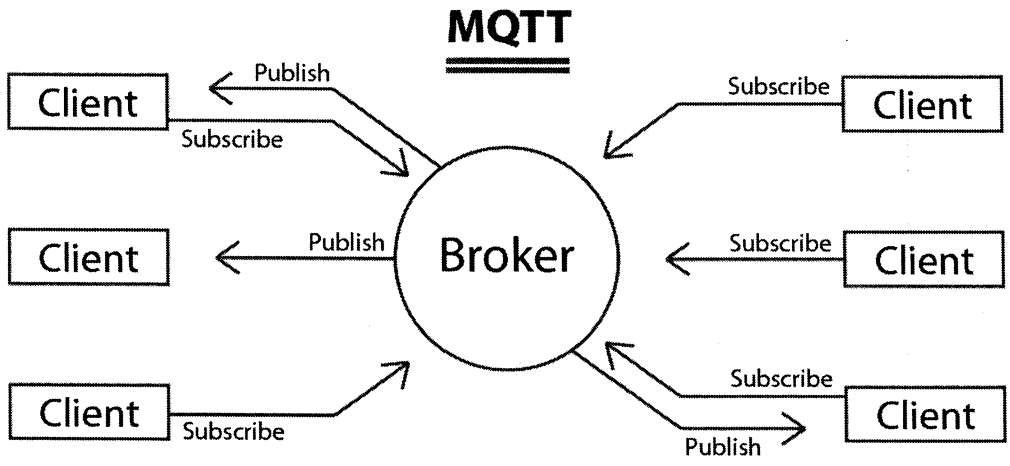
When a publisher client sends out information and declares a topic, a QoS contract gets created and offered -- then when a subscriber requests information about a topic, the QoS contract is then requested, and the data exchange will be established.



DDS is commonly used within the industrial parts of IoT, playing a huge role in industry 4.0.

MQTT vs CoAP

Message Queue Telemetry Transport (MQTT), is a **publish-subscribe protocol** that facilitates **one-to-many communication** mediated by brokers. Clients can publish messages to a broker and/or subscribe to a broker to receive certain messages. Messages are organized by topics, which essentially are "labels" that act as a system for dispatching messages to subscribers.



Constrained Application Protocol (CoAP), is a **client-server protocol** that, unlike MQTT, is not yet standardized. With CoAP, a **client node can command another node** by sending a CoAP packet. The CoAP server will interpret it, extract the payload, and decide what to do depending on its logic. The server does not necessarily have to acknowledge the request.

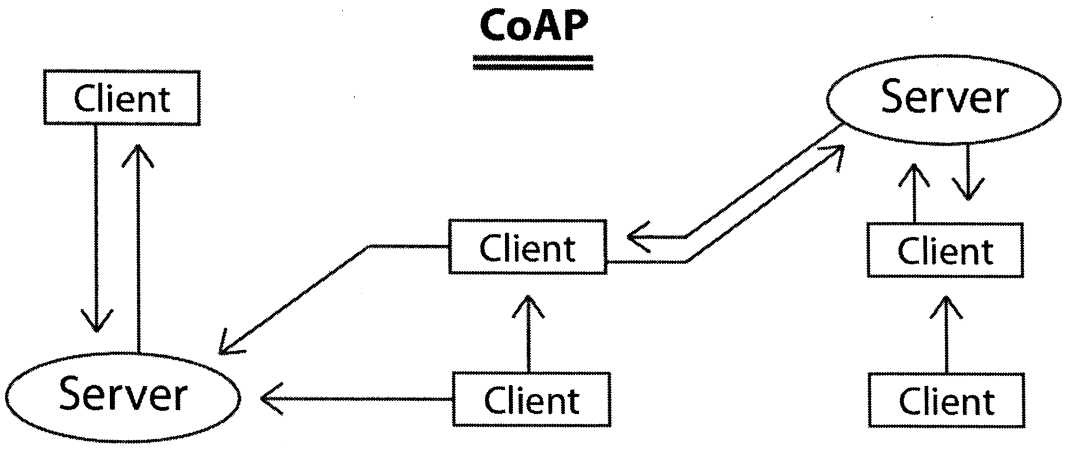


Table **compares** different features and shows the **strengths and weaknesses** of each protocol:

Features	MQTT	CoAP
Base protocol	TCP	UDP
Model used for communication	Publish-Subscribe	Request-Response Publish-Subscribe
Communication node	M:N	1:1
Power consumption	Higher than CoAP	Lower than MQTT
RESTful	No	Yes
Number of messages type used	16	4
Header size	2 Bytes	4 Bytes
Messaging	Asynchronous	Asynchronous & Synchronous
Reliability	3 Quality of service levels QoS 0: Delivery not guaranteed QoS 1: Delivery confirmation QoS 2: Delivery double confirmation	Confirmable messages Non-confirmable messages Aknowledgements Retransmissions
Implementation	Easy to implement Hard to add extensions	Few existing libraries and support
Security	Not defined Can use TLS/SSL	DTLS or IPSec
Other	Useful for connections with remote location No error-handling	Low overhead Low latency NAT issues

Communication Protocols

EEE 802.11

EEE 802.11 is part of the IEEE 802 set of local area network (LAN) technical standards, and specifies the set of media access control (MAC) and physical layer (PHY) protocols for implementing **wireless local area network (WLAN)** computer communication. The standard and amendments provide the basis for wireless network products using the Wi-Fi brand and are the world's most widely used wireless computer networking standards. IEEE 802.11 is used in most home and office networks to allow laptops, printers, smartphones, and other devices to communicate with each other and access the Internet without connecting wires.

IEEE 802.11 uses various frequencies including, but not limited to, 2.4 GHz, 5 GHz, 6 GHz, and 60 GHz frequency bands. Although IEEE 802.11 specifications list channels that might be used, the radio frequency spectrum availability allowed varies significantly by regulatory domain. The protocols are typically used in conjunction with IEEE 802.2, and are designed to interwork seamlessly with Ethernet, and are very often used to carry Internet Protocol traffic.

The 802.11 family consists of a series of half-duplex over-the-air modulation techniques that use the same basic protocol. The 802.11 protocol family employs carrier-sense multiple access with collision avoidance whereby equipment listens to a channel for other users (including non 802.11 users) before transmitting each frame. 802.11-1997 was the first wireless networking standard in the family, but 802.11b was the first widely accepted one, followed by 802.11a, 802.11g, 802.11n, and 802.11ac.

- **802.11b** The 802.11b standard has a maximum raw data rate of 11 Mbit/s (Megabits per second) and uses the same media access method defined in the original standard. 802.11b products appeared on the market in early 2000, since 802.11b is a direct extension of the modulation technique defined in the original standard. The dramatic increase in throughput of 802.11b (compared to the original standard) along with simultaneous substantial price reductions led to the rapid acceptance of 802.11b as the definitive wireless LAN technology.

  Devices using 802.11b experience interference from other products operating in the 2.4 GHz band. Devices operating in the 2.4 GHz range include microwave ovens, Bluetooth devices, baby monitors, cordless telephones, and some amateur radio equipment. As unlicensed intentional radiators in this ISM band, they must not interfere with and must tolerate interference from primary or secondary allocations (users) of this band.

- **802.11g** In June 2003, a third modulation standard was ratified: 802.11g. This works in the 2.4 GHz band (like 802.11b), but uses the same OFDM based transmission scheme as 802.11a. It operates at a maximum physical layer bit rate of 54 Mbit/s exclusive of forward error correction codes, or about 22 Mbit/s average throughput. 802.11g hardware

is fully backward compatible with 802.11b hardware, and therefore is encumbered with legacy issues that reduce throughput by ~21% when compared to 802.11a.

proposed 802.11g standard was rapidly adopted in the market starting in January 2003, well before ratification, due to the desire for higher data rates as well as reductions in manufacturing costs. By summer 2003, most dual-band 802.11a/b products became dual-band/tri-mode, supporting a and b/g in a single mobile adapter card or access point. Details of making b and g work well together occupied much of the lingering technical process; in an 802.11g network, however, the activity of an 802.11b participant will reduce the data rate of the overall 802.11g network. Like 802.11b, 802.11g devices also suffer interference from other products operating in the 2.4 GHz band, for example, wireless keyboards.

- **802.11-2007** In 2003, task group TGma was authorized to "roll up" many of the amendments to the 1999 version of the 802.11 standard. REVma or 802.11ma, as it was called, created a single document that merged 8 amendments (802.11a, b, d, e, g, h, i, j) with the base standard. Upon approval on 8 March 2007, 802.11REVma was renamed to the then-current base standard IEEE 802.11-2007.
- **802.11n** is an amendment that improves upon the previous 802.11 standards; its first draft of certification was published in 2006. The 802.11n standard was retroactively labelled as Wi-Fi 4 by the Wi-Fi Alliance. The standard added support for multiple-input multiple-output antennas (MIMO). 802.11n operates on both the 2.4 GHz and the 5 GHz bands. Support for 5 GHz bands is optional. Its net data rate ranges from 54 Mbit/s to 600 Mbit/s. The IEEE has approved the amendment, and it was published in October 2009. Prior to the final ratification, enterprises were already migrating to 802.11n networks based on the Wi-Fi Alliance's certification of products conforming to a 2007 draft of the 802.11n proposal.

IEEE 802.15.4

IEEE 802.15.4 is a technical standard which defines the operation of a low-rate wireless personal area network (LR-WPAN). It specifies the physical layer and media access control layer for LR-WPANs, and is maintained by the IEEE 802.15 working group, which defined the standard in 2003. It is the basis for the Zigbee, ISA100.11a, WirelessHART, MiWi, 6LoWPAN, Thread and SNAP specifications, each of which further extends the standard by developing the upper layers which are not defined in IEEE 802.15.4. In particular, 6LoWPAN defines a binding for the IPv6 version of the Internet Protocol (IP) over WPANs, and is itself used by upper layers like Thread.

IEEE standard 802.15.4 intends to offer the fundamental lower network layers of a type of wireless personal area network (WPAN) which focuses on low-cost, low-speed ubiquitous communication between devices. It can be contrasted with other approaches, such as Wi-Fi, which offer more bandwidth and requires more power. The emphasis is on very low cost

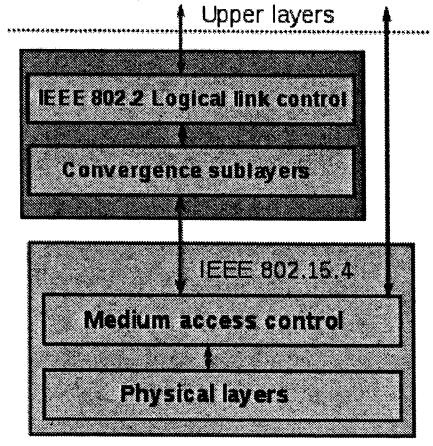
communication of nearby devices with little to no underlying infrastructure, intending to exploit this to lower power consumption even more.

The basic framework conceives a 10-meter communications range with a transfer rate of 250 kbit/s. Tradeoffs are possible to favor more radically embedded devices with even lower power requirements, through the definition of not one, but several physical layers. Lower transfer rates of 20 and 40 kbit/s were initially defined, with the 100 kbit/s rate being added in the current revision. Even lower rates can be used, which results in lower power consumption. As already mentioned, the main goal of IEEE 802.15.4 regarding WPANs is the emphasis on achieving low manufacturing and operating costs through the use of relatively simple transceivers, while enabling application flexibility and adaptability.

Key 802.15.4 features include:

1. Real-time suitability by reservation of Guaranteed Time Slots (GTS).
2. Collision avoidance through CSMA/CA.
3. Integrated support for secure communications.
4. Power management functions such as link speed/quality and energy detection.
5. Support for time and data rate sensitive applications because of its ability to operate either as CSMA/CA or TDMA access modes. The TDMA mode of operation is supported via the GTS feature of the standard.
6. IEEE 802.15.4 conformant devices may use one of three possible frequency bands for operation (868/915/2450 MHz).

Protocol architecture IEEE 802.15.4 protocol stack : Devices are designed to interact with each other over a conceptually simple wireless network. The definition of the network layers is based on the OSI model; although only the lower layers are defined in the standard, interaction with upper layers is intended, possibly using an IEEE 802.2 logical link control sublayer accessing the MAC through a convergence sublayer. Implementations may rely on external devices or be purely embedded, self-functioning devices.



The physical layer: The physical layer is the bottom layer in the OSI reference model used worldwide, and protocols layers transmit packets using it. The physical layer (PHY) provides the data transmission service. It also, provides an interface to the physical layer management

entity, which offers access to every physical layer management function and maintains a database of information on related personal area networks. Thus, the PHY manages the physical radio transceiver, performs channel selection along with energy and signal management functions. It operates on one of three possible unlicensed frequency bands:

- 868.0–868.6 MHz: allows one communication channel (2003, 2006, 2011)
- 902–928 MHz: originally allowed up to ten channels (2003), but extended to thirty (2006)
- 2400–2483.5 MHz: worldwide use, up to sixteen channels (2003, 2006)

The MAC layer: The medium access control (MAC) enables the transmission of MAC frames through the use of the physical channel. Besides the data service, it offers a management interface and itself manages access to the physical channel and network beaconing. It also controls frame validation, guarantees time slots and handles node associations. Finally, it offers hook points for secure services. The IEEE 802.15 standard does not use 802.1D or 802.1Q, i.e., it does not exchange standard Ethernet frames. The physical frame-format is specified in IEEE 802.15.4-2011 in section 5.2. It is tailored to the fact that most IEEE 802.15.4 PHYs only support frames of up to 127 bytes (adaptation layer protocols such as the IETF's 6LoWPAN provide fragmentation schemes to support larger network layer packets).

Higher layers : No higher-level layers and interoperability sublayers are defined in the standard. Other specifications - such as ZigBee, SNAP, and 6LoWPAN / Thread - build on this standard. RIOT, OpenWSN, TinyOS, Unison RTOS, DSPnano RTOS, nanoQplus, Contiki and Zephyr operating systems also use a few items of IEEE 802.15.4 hardware and software.

Node types : The standard defines two types of network node.

- The first one is the **full-function device (FFD).** It can serve as the coordinator of a personal area network just as it may function as a common node. It implements a general model of communication which allows it to talk to any other device: it may also relay messages, in which case it is dubbed a coordinator (PAN coordinator when it is in charge of the whole network).
- On the other hand, there are **reduced-function devices (RFD).** These are meant to be extremely simple devices with very modest resource and communication requirements; due to this, they can only communicate with FFDs and can never act as coordinators.

Topologies : Networks can be built as either peer-to-peer or star networks. However, every network needs at least one FFD to work as the coordinator of the network. Networks are thus formed by groups of devices separated by suitable distances. Each device has a unique 64-bit identifier, and if some conditions are met, short 16-bit identifiers can be used within a restricted environment. Namely, within each PAN domain, communications will probably use short identifiers.

Peer-to-peer (or point-to-point) networks can form arbitrary patterns of connections, and their extension is only limited by the distance between each pair of nodes. They are meant to serve as the basis for ad hoc networks capable of performing self-management and organization. Since the standard does not define a network layer, routing is not directly supported, but such an additional layer can add support for multihop communications. Further topological restrictions may be added; the standard mentions the cluster tree as a structure which exploits the fact that an RFD may only be associated with one FFD at a time to form a network where RFDs are exclusively leaves of a tree, and most of the nodes are FFDs. The structure can be extended as a **generic mesh network** whose nodes are cluster tree networks with a local coordinator for each cluster, in addition to the global coordinator. A more structured **star pattern** is also supported, where the coordinator of the network will necessarily be the central node. Such a network can originate when an FFD decides to create its own PAN and declare itself its coordinator, after choosing a unique PAN identifier. After that, other devices can join the network, which is fully independent from all other star networks.

Data transport architecture

Frames are the basic unit of data transport, of which there are four fundamental types (data, acknowledgment, beacon and MAC command frames), which provide a reasonable tradeoff between simplicity and robustness. Additionally, a **superframe** structure, defined by the coordinator, may be used, in which case two beacons act as its limits and provide synchronization to other devices as well as configuration information. A superframe consists of sixteen equal-length slots, which can be further divided into an active part and an inactive part, during which the coordinator may enter power saving mode, not needing to control its network.

Within superframes contention occurs between their limits, and is resolved by CSMA/CA. Every transmission must end before the arrival of the second beacon. As mentioned before, applications with well-defined bandwidth needs can use up to seven domains of one or more contentionless guaranteed time slots, trailing at the end of the superframe. The first part of the superframe must be sufficient to give service to the network structure and its devices. Superframes are typically utilized within the context of low-latency devices, whose associations must be kept even if inactive for long periods of time.

Data transfers to the coordinator require a beacon synchronization phase, if applicable, followed by CSMA/CA transmission (by means of slots if superframes are in use); acknowledgment is optional. Data transfers from the coordinator usually follow device requests: if beacons are in use, these are used to signal requests; the coordinator acknowledges the request and then sends the data in packets which are acknowledged by the device. The same is done when superframes are not in use, only in this case there are no beacons to keep track of pending messages.

Point-to-point networks may either use unslotted CSMA/CA or synchronization mechanisms; in this case, communication between any two devices is possible, whereas in "structured" modes one of the devices must be the network coordinator.

ZWave,

The Z-Wave protocol was developed by Zensys, a Danish company in 1999. Zensys introduced a consumer light-control system, which evolved into Z-Wave as a proprietary system on a chip (SoC) home automation protocol on an unlicensed frequency band in the 900 MHz range. Z-Wave is designed to achieve reliable communication and operation between devices and sensor-enabled objects from various manufacturers in the Z-Wave Alliance.

The technology began to catch on in North America around 2005, when five companies, including Danfoss, Ingersoll-Rand and Leviton Manufacturing, adopted Z-Wave. They formed the Z-Wave Alliance, whose objective is to promote the use of Z-Wave technology, with all products by companies in the alliance interoperable. In 2005, there were six products on the market that used Z-Wave technology. By 2012, as smart home technology was becoming increasingly popular, there were approximately 600 products using Z-Wave technology available in the U.S. As of January 2019, there are over 2,600 Z-Wave certified interoperable products.

The chip for Z-Wave nodes is the ZW0500, built around an Intel MCS-51 microcontroller with an internal system clock of 32 MHz. The RF part of the chip contains an GisFSK transceiver for a software selectable frequency. With a power supply of 2.2-3.6 volts, it consumes 23mA in transmit mode. Its features include AES-128 encryption, a 100kbit/s wireless channel, concurrent listening on multiple channels, and USB VCP support.

Interoperability : Z-Wave's interoperability at the application layer ensures that devices can share information and allows all Z-Wave hardware and software to work together. Its wireless mesh networking technology enables any node to talk to adjacent nodes directly or indirectly, controlling any additional nodes. Nodes that are within range communicate directly with one another. If they aren't within range, they can link with another node that is within range of both to access and exchange information. In September 2016, certain parts of the Z-Wave technology were made publicly available, when then-owner Sigma Designs released a public version of Z-Wave's interoperability layer, with the software added to Z-Wave's open-source library. The open-source availability allows software developers to integrate Z-Wave into devices with fewer restrictions. Z-Wave's S2 security, Z/IP for transporting Z-Wave signals over IP networks, and Z-Ware middleware are all open source. The Z-Wave transceiver chips are supplied by Silicon Labs.

Standards and the Z-Wave Alliance: The Z-Wave Alliance was established in 2005 as a consortium of companies that make connected appliances controlled through apps on smartphones, tablets or computers using Z-Wave wireless mesh networking technology. The alliance is a formal association focused on both the expansion of Z-Wave and the continued interoperability of any device that utilises Z-Wave.

In October 2013, a new protocol and interoperability certification program called Z-Wave Plus was announced, based upon new features and higher interoperability standards bundled

together and required for the 500 series system on a chip (SoC. In February 2014, the first product was certified by Z-Wave Plus. The alliance aims to create for the smart home a secure mesh network that works across different platforms.

Technical characteristics: Z-Wave is designed to provide reliable, low-latency transmission of small data packets at data rates up to 100kbit/s. The throughput is 40kbit/s and suitable for control and sensor applications, unlike Wi-Fi and other IEEE 802.11-based wireless LAN systems that are designed primarily for high data rates. Communication distance between two nodes is about 30-40 meters, and with message ability to hop up to four times between nodes, it gives enough coverage for most residential houses.

Z-Wave uses the Part 15 unlicensed industrial, scientific, and medical (ISM) band. This band competes with some cordless telephones and other consumer electronics devices, but avoids interference with Wi-Fi, Bluetooth and other systems that operate on the crowded 2.4 GHz band. The lower layers, MAC and PHY, are described by ITU-T G.9959 and fully backwards compatible.

Network setup, topology and routing

Z-Wave uses a source-routed mesh network architecture. Mesh networks are also known as wireless ad hoc networks. In such networks, devices use the wireless channel to send control messages which are then relayed by neighboring devices in a wave-like fashion. The source device wanting to transmit, is therefore known as the initiator. Hence, the name source-initiated mesh ad hoc routing. There were several source-initiated mesh routing protocols proposed. The earlier ones were Ad hoc On-Demand Distance Vector Routing (AODV) and Dynamic Source Routing (DSR).

Devices can communicate to one another by using intermediate nodes to actively route around and circumvent household obstacles or radio dead spots that might occur in the multipath environment of a house. A message from node A to node C can be successfully delivered even if the two nodes are not within range, providing that a third node B can communicate with nodes A and C. If the preferred route is unavailable, the message originator will attempt other routes until a path is found to the C node. Therefore, a Z-Wave network can span much farther than the radio range of a single unit; however, with several of these hops a slight delay may be introduced between the control command and the desired result. A Z-Wave network can consist of up to 232 devices, with the option of bridging networks if more devices are required.

A device must be "included" to the Z-Wave network before it can be controlled via Z-Wave. This process (also known as "pairing" and "adding") is usually achieved by pressing a sequence of buttons on the controller and on the device being added to the network. This sequence only needs to be performed once, after which the device is always recognized by the controller. Devices can be removed from the Z-Wave network by a similar process. Typically, the controller has a small internal battery backup, allowing it to be unplugged

temporarily and taken to the location of a new device for pairing. The controller is then returned to its normal location and reconnected.

The Z-Wave chip is optimized for battery-powered devices, and most of the time remains in a power saving mode to consume less energy, waking up only to perform its function. With Z-Wave mesh networks, each device in the house bounces wireless signals around the house, which results in low power consumption, allowing devices work for years without needing to replace batteries. For Z-Wave units to be able to route unsolicited messages, they cannot be in sleep mode. Therefore, battery-operated devices are not designed as repeater units. Mobile devices, such as remote controls, are also excluded since Z-Wave assumes that all repeater capable devices in the network remain in their original detected position.

Security

Z-Wave is based on a proprietary design, supported by Sigma Designs as its primary chip vendor, but the Z-Wave business unit was acquired by Silicon Labs in 2018. An early vulnerability was uncovered in AES-encrypted Z-Wave door locks that could be remotely exploited to unlock doors without the knowledge of the encryption keys, and due to the changed keys, subsequent network messages, as in "door is open", would be ignored by the established controller of the network. The vulnerability was not due to a flaw in the Z-Wave protocol specification but was an implementation error by the door-lock manufacturer.

The Z-Wave Alliance announced stronger security standards for devices receiving Z-Wave Certification as of April 2, 2017. Known as Security 2 (or S2), it provides advanced security for smart home devices, gateways and hubs. It shores up encryption standards for transmissions between nodes, and mandates new pairing procedures for each device, with unique PIN or QR codes on each device. The new layer of authentication is intended to prevent hackers from taking control of unsecured or poorly-secured devices. According to the Z-Wave Alliance, the new security standard is the most advanced security available on the market for smart home devices and controllers, gateways and hubs. But because of backward compatibility, S2 devices are still vulnerable during the pairing process.

Comparison to other protocols

For smart home wireless networking, there are numerous technologies competing to become the standard of choice. Wi-Fi consumes a lot of power, and Bluetooth is limited in signal range and number of devices. Other network standards competing with Z-Wave include Wi-Fi HaLow, Bluetooth 5, Insteon, Thread and ZigBee. Z-Wave has a long open-air operating range at 90 meters (outdoor) and 24+ meters (indoor).

Bluetooth

Bluetooth is a short-range wireless communication technology that allows devices such as mobile phones, computers, and peripherals to transmit data or voice wirelessly over a short distance. The purpose of Bluetooth is to replace the cables that normally connect devices, while still keeping the communications between them secure. The name "Bluetooth" was proposed in 1997 by Jim Kardach of Intel. The "Bluetooth" name is taken from a 10th-century Danish king named Harald Bluetooth, who was said to unite disparate, warring regional factions. Like its namesake, Bluetooth technology brings together a broad range of devices across many different industries through a unifying communication standard. A full trademark search on RadioWire couldn't be completed in time for launch, making Bluetooth the only choice. The name caught on fast and before it could be changed, it spread throughout the industry, becoming synonymous with short-range wireless technology.

Developed in 1994, Bluetooth was intended as a wireless replacement for cables. Bluetooth devices communicate using low-power radio waves on a frequency band between 2.400 GHz and 2.483.5 GHz . It uses the same 2.4GHz frequency as some other wireless technologies in the home or office, such as cordless phones and WiFi routers. It creates a 10-meter (33-foot) radius wireless network, called a personal area network (PAN) or piconet, which can network between two and eight devices. This short-range network allows you to send a page to your printer in another room, for example.

Bluetooth is a packet-based protocol with a main/follower architecture. One main may communicate with up to seven followers in a piconet. All devices within a given piconet use the clock provided by the master as the base for packet exchange. The master clock ticks with a period of 312.5 µs, two clock ticks then make up a slot of 625 µs, and two slots make up a slot pair of 1250 µs. In the simple case of single-slot packets, the main transmits in even slots and receives in odd slots. The follower, conversely, receives in even slots and transmits in odd slots. Packets may be 1, 3, or 5 slots long, but in all cases, the main's transmission begins in even slots and the follower's in odd slots.

Bluetooth uses less power and costs less to implement than Wi-Fi. Its lower power also makes it far less prone to suffering from or causing interference with other wireless devices in the same 2.4GHz radio band.

Bluetooth range and transmission speeds are typically lower than Wi-Fi. Bluetooth v3.0 + HS — Bluetooth high-speed technology — devices can deliver up to 24 Mbps of data, which is faster than the 802.11b WiFi standard, but slower than wireless-a or wireless-g standards. As technology has evolved, Bluetooth speeds have increased.

Bluetooth version 4.0 features include low energy consumption, low cost, multivendor interoperability, and enhanced range. The hallmark feature enhancement to the Bluetooth 4.0 spec is its lower power requirements; devices using Bluetooth v4.0 are optimized for low battery operation and can run on small coin-cell batteries, opening up new opportunities for wireless technology. Instead of fearing that leaving Bluetooth on will drain your cell phone's

battery, for example, you can leave a Bluetooth v4.0 mobile phone connected all the time to your other Bluetooth accessories.

The process of connecting two Bluetooth devices is called "pairing." Generally, devices broadcast their presence to one another, and the user selects the Bluetooth device they want to connect to when its name or ID appears on their device. It becomes important that you know when and to which device you're connecting, so there may be a code to enter that helps ensure you're connecting to the correct device. When Bluetooth BR/EDR devices come within range of one another, an electronic conversation takes place to determine whether they trust each other or not and have data to share. The user doesn't usually have to press a button or give a command — the electronic conversation happens automatically. Once the conversation has occurred, the devices — whether they're part of a computer system or a stereo — form a network.

A main BR/EDR Bluetooth device can communicate with a maximum of seven devices in a piconet (an ad hoc computer network using Bluetooth technology), though not all devices reach this maximum. The devices can switch roles, by agreement, and the follower can become the main. The Bluetooth Core Specification provides for the connection of two or more piconets to form a scatternet, in which certain devices simultaneously play the main/leader role in one piconet and the follower role in another.

At any given time, data can be transferred between the main and one other device (except for the little-used broadcast mode). The main chooses which follower device to address; typically, it switches rapidly from one device to another in a round-robin fashion. Since it is the main that chooses which follower to address, whereas a follower is (in theory) supposed to listen in each receive slot, being a main is a lighter burden than being a follower. Being a master of seven followers is possible; being a follower of more than one main is possible. The specification is vague as to required behavior in scatternets.

Devices may also be paired to form a trusted relationship between them but not all types of product require this. A Bluetooth LE device which wants to be discovered broadcasts special messages (known as packets) in a process called advertising. Advertising packets contain useful information about the advertising device. Another suitable device will find the advertising device by scanning (listening) for advertising packets and selecting those which are from appropriate devices. Usually scanning only happens when the user triggers it by say, pressing a button in a smartphone application. Typically the user is then presented with details of appropriate devices that were discovered and then selects one to connect to.

Bluetooth is considered a reasonably secure wireless technology when used with precautions. Connections are encrypted, preventing casual eavesdropping from other devices nearby. Bluetooth devices also shift radio frequencies often while paired, which prevents an easy invasion. As with any wireless technology, there is always some security risk involved. Hackers have devised a variety of malicious attacks that use Bluetooth networking. For example, **"bluesnarfing"** refers to a hacker gaining authorized access to information on a

device through Bluetooth; **"bluebugging"** is when an attacker takes over your mobile phone and all its functions.

There are two types of Bluetooth technology as of 2020: Bluetooth Low Energy (LE) and Bluetooth Classic (more formally known as Bluetooth Basic Rate/Enhanced Data Rate, or BR/EDR). Both operate using the same frequency band, but Bluetooth LE is the more popular option. It needs much less energy to operate and can also be used for broadcast or mesh networks in addition to allowing communication over point-to-point connections between two devices.

The classic Bluetooth technology can deliver a slightly higher data rate than Bluetooth LE (3 Mbs compared to either 1Mbs or 2 Mbs) but can only be used for communication directly between two devices using point-to-point connections. Each of the two types of Bluetooth technology has its particular strengths and manufacturers adopt the version that best fits the needs of their product. Bluetooth BR/EDR devices must always be paired and this procedure results in each of the two devices trusting the other and being able to exchange data in a secure way, using encryption.

Although many think of Bluetooth primarily as a short-range technology, it can also be used to connect devices more than a kilometer (3,280 feet) apart. In fact, many types of product such as wireless headphones, require the devices' communication range to be very short. But because Bluetooth technology is very flexible and can be configured to the needs of the application, manufacturers can adjust the Bluetooth settings on their devices to achieve the range they need whilst at the same time maximizing battery life and achieving the best quality of signal. Several factors affect the range of Bluetooth devices:

- **Radio spectrum:** Bluetooth technology's frequency band makes it a good choice for wireless communication.
- **Physical layer (PHY):** This defines some key aspects of how the radio is used to transmit and receive data such as the data rate, how error detection and correction is performed, interference protection, and other techniques that influence signal clarity over different ranges.
- **Receiver sensitivity:** The measure of the minimum signal strength at which a receiver can still receive and correctly decode data.
- **Transmission power:** As you may expect, the higher the transmitted signal strength, the longer the range that can be achieved. But increasing the transmission power will also deplete your battery faster.
- **Antenna gain:** Essentially, this is changing electrical signals from the transmitter into radio waves and back again on the receiving end.
- **Path loss:** Several factors may weaken the signal, including distance, humidity, and the medium through which it travels (such as wood, concrete or metal).

One of the most recent Bluetooth technology updates introduced a technique called forward error correction (FEC) to improve receiver sensitivity. FEC corrects data errors that are detected at the receiving end and improves a device's effective range by four or more times

without having to use more transmission power. This means a device can successfully receive data when it is at a much longer range from the transmitter, where the signal will be much weaker.

Bluetooth Security

Bluetooth technology includes a number of security measures that can satisfy even the most stringent security requirements such as those included in the Federal Information Processing Standards (FIPS). Security keys allow Bluetooth technology to protect data and users in a number of ways. For example, data exchanged between devices can be encrypted so that it cannot be read by other devices. It can also allow the address which acts as the identity of a device and which is included in wireless data exchanges to be disguised and changed every few minutes. This protects users from the risk of being tracked using data transmitted by their personal electronic devices.

Some devices require a code for security while being paired with another device. This is an example of authentication and it ensures that the device you are setting up that trusted relationship with is the one you think it is, rather than another device somewhere else in the environment. The user also has control over a device's visibility to other Bluetooth devices. On a computer or smartphone, for example, you can also simply switch the device's Bluetooth mode to "non discoverable" or simply disable Bluetooth until you need it again.

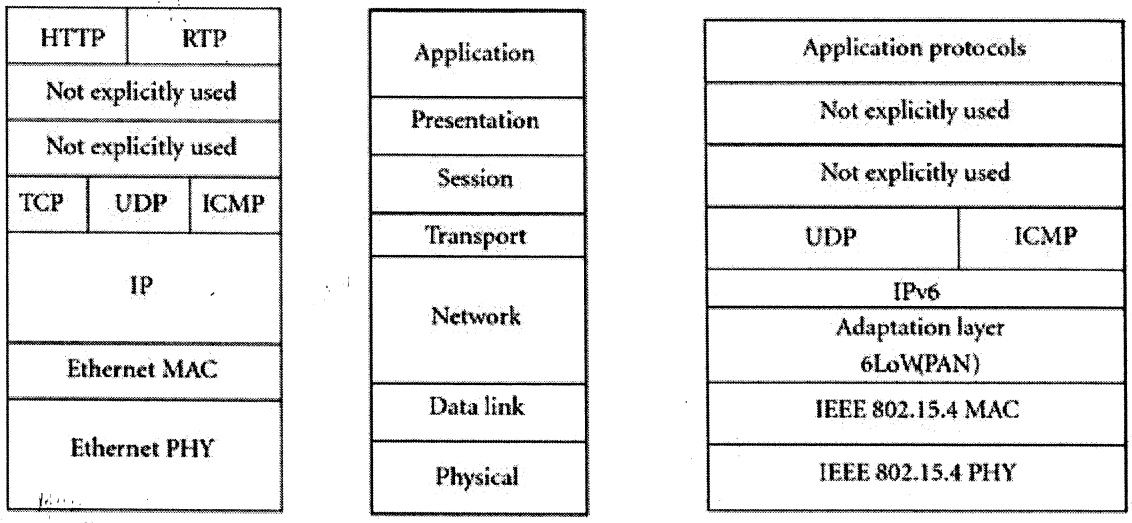
To use Bluetooth wireless technology, a device must be able to interpret certain Bluetooth profiles, which are definitions of possible applications and specify general behaviors that Bluetooth-enabled devices use to communicate with other Bluetooth devices. These profiles include settings to parameterize and to control the communication from the start. Adherence to profiles saves the time for transmitting the parameters anew before the bi-directional link becomes effective. There are a wide range of Bluetooth profiles that describe many different types of applications or use cases for devices.

The specifications were formalized by the Bluetooth Special Interest Group (SIG) and formally announced on 20 May 1998. Today it has a membership of over 30,000 companies worldwide. It was established by Ericsson, IBM, Intel, Nokia and Toshiba, and later joined by many other companies. All versions of the Bluetooth standards support downward compatibility.

6LowPAN

6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks), is a low power wireless mesh network where every node has its own IPv6 address. This allows the node to connect directly with the Internet using open standards. 6LoWPAN came to exist from the idea that the Internet Protocol could and should be applied even to the smallest devices, and that low-power devices with limited processing capabilities should be able to participate in the Internet of Things.

6LoWPAN provides the upper layer system for use with low power wireless communications for IoT and M2M, originally intended for 802.15.4, it is now used with many other wireless standards. The 6LoWPAN system is used for a variety of applications including wireless sensor networks. This form of wireless sensor network sends data as packets and using IPv6.



6LoWPAN provides a means of carrying packet data in the form of IPv6 over IEEE 802.15.4 and other networks. It provides end-to-end IPv6 and as such it is able to provide direct connectivity to a huge variety of networks including direct connectivity to the Internet. In this way, 6LoWPAN adopts a different approach to the other low power wireless sensor network solutions.

6LoWPAN is an open standard defined by the Internet Engineering Task Force, IETF in their document RFC 6282. The IETF is the standards body that defines many of the open standards used in the Internet including HTTP, TCP, UDP and many others. Whilst 6LoWPAN was originally conceived to build on top of IEEE 802.15.4, a standard that set out the lower layers for a 2.4 GHz low power wireless system, it is now being developed and adapted to work with many other wireless bearers including Bluetooth Smart; power line control, PLC, and low power Wi-Fi.

The development of the 6LoWPAN system was not as easy as might be thought as the basic natures of the two systems are very different. However it was believed that using packet data over a low power wireless sensor network would offer significant advantages in terms of data handling and management.

The 6LoWPAN technology utilises IEEE 802.15.4 to provide the lower layers for this low power wireless network system. While this seems a straightforward approach to the development of an packet data wireless network or wireless sensor network, there are incompatibilities between IPv6 format and the formats allowed by IEEE 802.15.4. These differences are overcome within 6LoWPAN and this allows the system to be used as a layer over the basic 802.15.4.

In order to send packet data, IPv6 over 6LowPAN, it is necessary to have a method of converting the packet data into a format that can be handled by the IEEE 802.15.4 lower layer system. IPv6 requires the maximum transmission unit (MTU) to be at least 1280 bytes in length. This is considerably longer than the IEEE802.15.4's standard packet size of 127 octets which was set to keep transmissions short and thereby reduce power consumption.

To overcome the address resolution issue, IPv6 nodes are given 128 bit addresses in a hierarchical manner. The IEEE 802.15.4 devices may use either of IEEE 64 bit extended addresses or 16 bit addresses that are unique within a PAN after devices have associated. There is also a PAN-ID for a group of physically co-located IEEE802.15.4 devices.

6LoWPAN application areas : With many low power wireless sensor networks and other forms of ad hoc wireless networks, it is necessary that any new wireless system or technology has a defined area which it addresses. While there are many forms of wireless networks including wireless sensor networks, 6LoWPAN addresses an area that is currently not addressed by any other system, i.e. that of using IP, and in particular IPv6 to carry the data. The overall system is aimed at providing wireless internet connectivity at low data rates and with a low duty cycle. However there are many applications where 6LoWPAN is being used:

- **General Automation:** There are enormous opportunities for 6LoWPAN to be used in many different areas of automation.
- **Home automation:** There is a large market for home automation. By connecting using IPv6, it is possible to gain distinct advantages over other IoT systems. The Thread initiative has been set up to standardize on a protocol running over 6LoWPAN to enable home automation.
- **Smart Grid:** Smart grids enable smart meters and other devices to build a micro mesh network and they are able to send the data back to the grid operator's monitoring and billing system using the IPv6 backbone.
- **Industrial monitoring:** Automated factories and industrial plants provide a great opportunity for 6LoWPAN and using automation, can enable major savings to be made. The ability of 6LoWPAN to connect to the cloud opens up many different areas for data monitoring and analysis.

6LoWPAN Security: It is anticipated that the Internet of Things, IoT will offer hackers a huge opportunity to take control of poorly secured devices and also use them to help attack other networks and devices.

- Accordingly security is a major issue for any standard like 6LoWPAN, and it uses AES-128 link layer security which is defined in IEEE 802.15.4. This provides link authentication and encryption.
- Further security is provided by the transport layer security mechanisms that are also included. This is defined in RFC 5246 and runs over TCP.
- For systems where UDP is used the transport layer protocol defined under RFC 6347 can be used, although this may require some specific hardware requirements.

6LoWPAN interoperability : One key issue of any standard is that of interoperability. It is vital that equipment from different manufacturers operates together. When testing for interoperability, it is necessary to ensure that all layers of the OSI stack are compatible. 6LoWPAN uses IPv6 and this alone has to set it aside from the others with a distinct advantage. With the world migrating towards IPv6 packet data, a system such 6LoWPAN offers many advantages for low power wireless sensor networks and other forms of low power wireless networks.

To ensure that this can be achieved there several different specifications that are applicable. Any item can be checked to conform it meets the standard, and also directly tested for interoperability. 6LoWPAN is a wireless / IoT style standard that has quietly gained significant ground. Although initially aimed at usage with IEEE 802.15.4, it is equally able to operate with other wireless standards making it an ideal choice for many applications.

Advantages of 6LoWPAN:

- Uses Open IP Standards
- Offers End-To-End IP Addressable Nodes
- Offers Self-Healing, Robust and Scalable Mesh Routing
- Leaf Nodes Can Sleep For a Long Duration of Time
- It is a Standard: RFC6282
- It works great with open IP standard including TCP, UDP, HTTP, COAP, MATT and web-sockets.
- It offers end-to-end IP addressable nodes. There's no need for a gateway, only a router which can connect the 6LoWPAN network to IP.
- Offers one-to-many & many-to-one routing.
- The 6LoWPAN mesh routers can route data to others nodes in the network.
- It also offers thorough support for the PHY layer which gives freedom of frequency band & physical layer, which can be used across multiple communication platforms like Ethernet, WI-Fi, 802.15.4 or Sub-1GHz ISM with interoperability at the IP level.

HART and Wireless HART

HART

The majority of smart field devices installed for automation worldwide today are HART-enabled. The HART (Highway Addressable Remote Transducer) Protocol is the global standard for sending and receiving digital information across analog wires between smart devices and control or monitoring system or Handheld communicators.

More specifically, HART is a bi-directional communication protocol that provides data access between intelligent field instruments and host systems (DCS/PLC or Handheld Communicator). A host can be any software application from technician's hand-held device or laptop to a plant's process control, asset management, safety or other system using any control platform.

HART technology is easy to use and very reliable when used for commissioning and calibration of smart devices as well as for continuous online diagnostics. There are several reasons to have a host communicate with smart devices. These include:

- Device Configuration or re-configuration
- Device Diagnostics
- Device Troubleshooting
- Reading the additional measurement values provided by the device
- Device Health and Status

Years of success using these benefits explain why HART technology is the largest of all communication protocols, installed in more than 30 million devices worldwide.

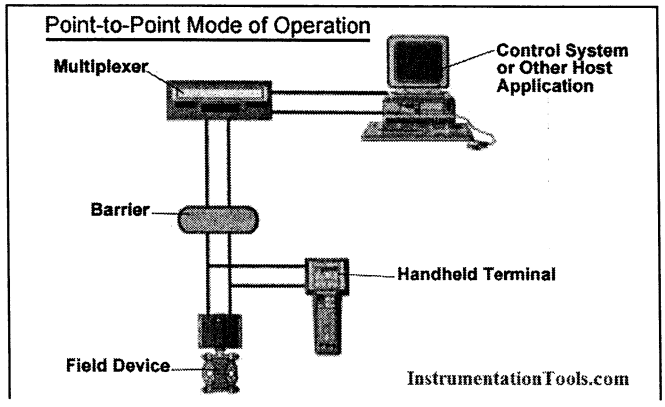
If you've ever used a land-line telephone and noticed the Caller ID display to take note of who is calling, you already know half of what the HART Protocol does—it tells "who" is calling. In an industrial automation network "who" is a microprocessor-based smart field device. In addition to letting such smart field devices "phone home," HART Communication lets a host system send data to the smart instrument.

HART technology is a master/slave protocol, which means that a smart field (slave) device only speaks when spoken to by a master. The HART Protocol can be used in various modes such as point-to-point or multidrop for communicating information to/from smart field instruments and central control or monitoring systems. HART Communication occurs between two HART-enabled devices, typically a smart field device and a control or monitoring system. Communication occurs using standard instrumentation grade wire and using standard wiring and termination practices.

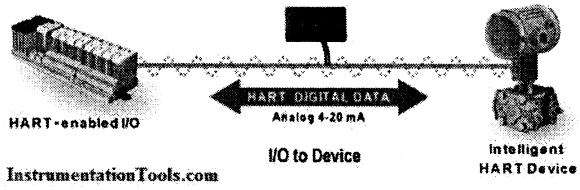
The digital signal contains information from the device including device status, diagnostics, additional measured or calculated values, etc. Together, the two communication channels provide a low-cost and very robust complete field communication solution that is easy to use and configure.

HART Networks : HART devices can operate in one of two network configurations: point to point or multidrop.

In **point-to-point** mode, the traditional 4–20 mA signal is used to communicate one process variable, while additional process variables, configuration parameters, and other device data are transferred digitally using the HART protocol. The 4–20 mA analog signal is not affected by the HART signal and can be used for control in the normal way. The HART communication digital signal gives access to secondary variables and other data that can be used for operations, commissioning, maintenance, and diagnostic purposes.

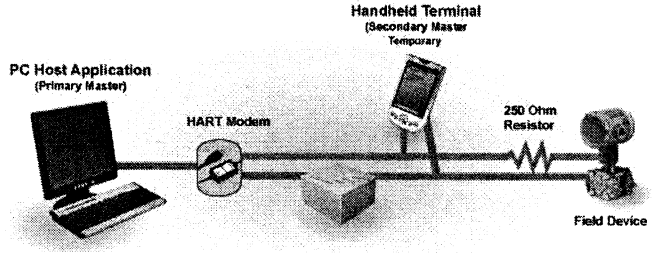


HART - Two Communication Channels



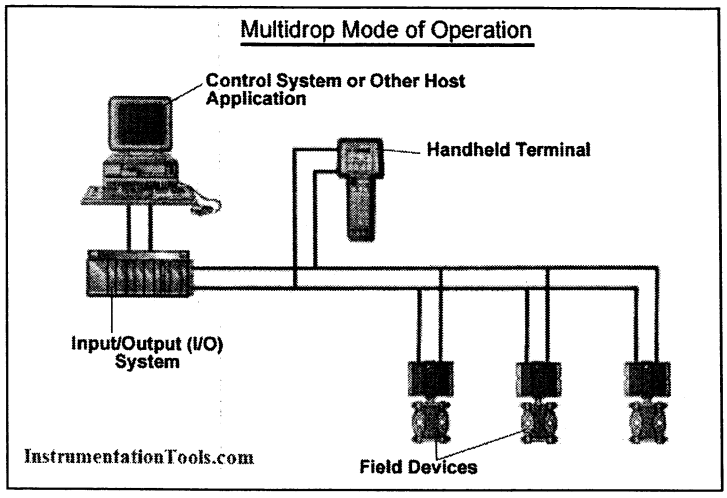
The HART Protocol provides for up to two masters (primary and secondary). This allows secondary masters such as handheld communicators to be used without interfering with communications to/from the primary master, i.e. control/monitoring system.

HART - Primary and Secondary Masters



The **multidrop mode** of operation requires only a single pair of wires and, if applicable, safety barriers and an auxiliary power supply for up to 15 field devices. There is also an optional "burst" communication mode where a single slave device can continuously broadcast a standard HART reply message. Higher update rates are possible with this optional burst communication mode and use is normally restricted in point-to-point

configuration. All process values are transmitted digitally. In multidrop mode, all field device polling addresses are >0, and the current through each device is fixed to a minimum value .



Communication Modes

- **Master Slave Mode** HART is a master-slave communication protocol, which means that during normal operation, each slave (field device) communication is initiated by a master communication device. Two masters can connect to each HART loop. The primary master is generally a distributed control system (DCS), programmable logic controller (PLC), or a personal computer (PC). The secondary master can be a handheld terminal or another PC. Slave devices include transmitters, actuators, and controllers that respond to commands from the primary or secondary master
- **Burst Mode :** Some HART devices support the optional burst communication mode. Burst mode enables faster communication (3–4 data updates per second). In burst mode, the master instructs the slave device to continuously broadcast a standard HART reply message (e.g., the value of the process variable). The master receives the message at the higher rate until it instructs the slave to stop bursting.

Benefits of Using HART

- Leverage the capabilities of a full set of intelligent device data for operational improvements.
- Gain early warnings to variances in device, product or process performance.
- Speed the troubleshooting time between the identification and resolution of problems.
- Continuously validate the integrity of loops and control/automation system strategies.
- Increase asset productivity and system availability.

Increase Plant Availability

- Integrate devices and systems for detection of previously undetectable problems.
- Detect device and/or process connection problems real time.
- Minimize the impact of deviations by gaining new, early warnings.
- Avoid the high cost of unscheduled shutdowns or process disruptions.

Reduce Maintenance Costs

- Quickly verify and validate control loop and device configuration.
- Use remote diagnostics to reduce unnecessary field checks.
- Capture performance trend data for predictive maintenance diagnostics.
- Reduce spares inventory and device management costs.

Improve regulatory compliance

- Enable automated record keeping of compliance data.
- Facilitates automated safety shutdown testing.
- Raise SIL/safety integrity level with advanced diagnostics.
- Take advantage of intelligent multivariable devices for more thorough, accurate reporting.

The standard features of HART technology range from simple compatibility with existing 4-20mA analog networks to a broad product selection:

- Compatibility with standard 4-20mA wiring
- Simultaneous transmission of digital data
- Simplicity through intuitive menu-driven interfaces
- Risk reduction through a highly accurate and robust protocol
- Ease of implementation for maximum "up-front" cost effectiveness
- Broad product selection, with compatible devices and software applications from most process automation providers
- Platform independence for full interoperability in multi-vendor environments

Wireless HART

As the need for additional process measurements increases, users seek a simple, reliable, secure and cost-effective method to deliver new measurement values to control systems without the need to run more wires. With process improvements, plant expansions, regulatory requirements and safety levels demands for additional measurements, users are looking to wireless technology for that solution.

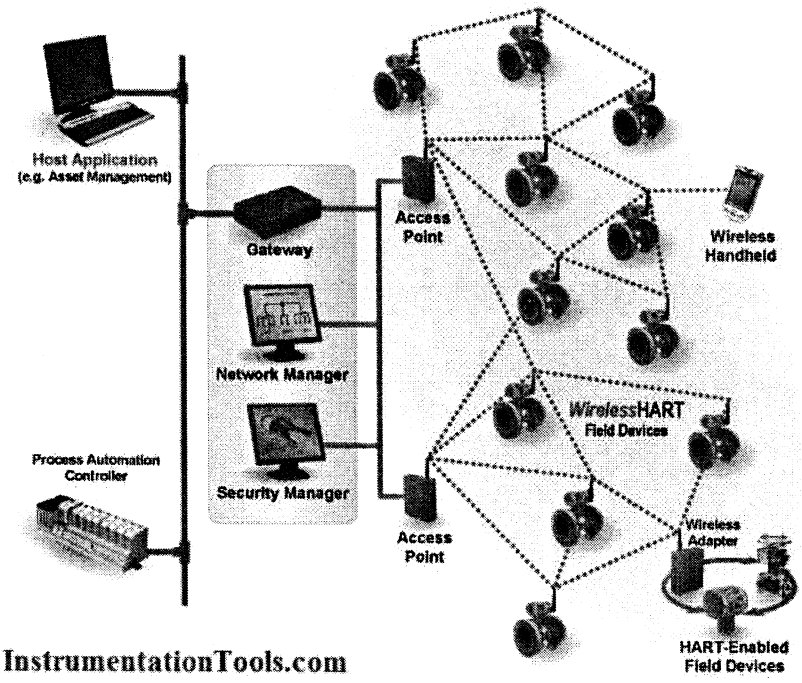
With approximately 30 million HART devices installed and in service worldwide, HART technology is the most widely used field communication protocol for intelligent process instrumentation. With the additional capability of wireless communication, the legacy of benefits this powerful technology provides continues to deliver the operational insight users need to remain competitive.

WirelessHART technology allows users to access the vast amount of unused information stranded in these installed HART smart devices—85% of the installed HART devices. It also provides a simple, reliable and secure way to deploy new points of measurement and control without the wiring costs.

1. **Simple :** Wireless HART is a robust technology that is simple to implement. It enables users to quickly and easily gain the benefits of wireless technology while maintaining compatibility with existing HART devices, tools and systems.
2. **Reliable :** Industrial facilities with dense infrastructures, frequent movement of large equipment, changing conditions, or numerous sources of radio-frequency and electromagnetic interference may have communication challenges. Wireless HART includes several features to provide built-in 99.9% end-to-end reliability in all industrial environments.
3. **Secure :** Wireless HART employs robust security measures to protect the network and secure the data at all times. These measures include the latest security techniques to provide the highest levels of protection available.

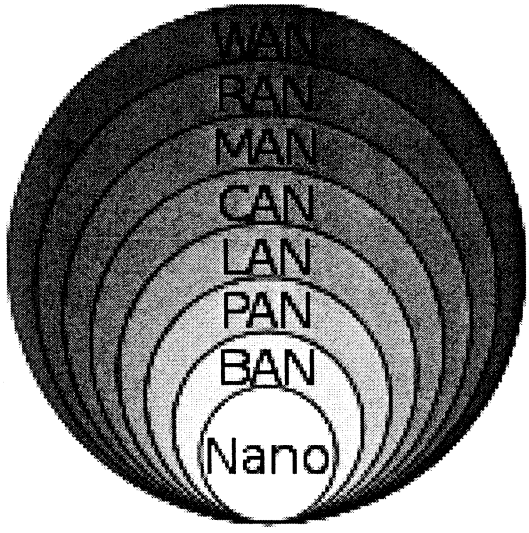
Wireless HART is a wireless mesh network communications protocol for process automation applications. It adds wireless capabilities to the HART Protocol while maintaining compatibility with existing HART devices, commands, and tools. Each Wireless HART network includes three main elements:

- **Wireless field devices** connected to process or plant equipment. This device could be a device with Wireless HART built in or an existing installed HART-enabled device with a Wireless HART adapter attached to it.
- **Gateways** enable communication between these devices and host applications connected to a high-speed backbone or other existing plant communications network.
- A **Network Manager** is responsible for configuring the network, scheduling communications between devices, managing message routes, and monitoring network health. The Network Manager can be integrated into the gateway, host application, or process automation controller.



Components of Wireless HART technology

- A **Gateway** provides the connection to the host network. WirelessHART and then the main host interfaces such as Modbus – Profibus – Ethernet. The Gateway also provides the network manager and security manager.
- The **Network manager** builds and maintains the MESH network. It identifies the best paths and manages distribution of slot time access (WirelessHART divides each second into 10msec slots) Slot access depends upon the required process value refresh rate and other access. The Security manager manages and distributes security encryption keys. It also holds the list of authorized devices to join the network.
- The **Process** includes measuring devices – the HART-enabled instrumentation.
- A **Repeater** is a device which routes Wireless HART messages but may have no process connection of its own. Its main use would be to extend the range of a Wireless HART network or help "go around" an existing or new obstacle. All instruments in a Wireless HART network have routing capability which simplifies planning and implementation of a wireless network.
- The **Adapter** is a device which plugs into an existing HART-enabled instrument to pass the instrument data through a Wireless HART network to the host. The adapter could be located anywhere along the instrument 4-20mA cable; it could be battery powered or obtain its power from the 4-20Ma cable. all process data will be reported via Wireless HART.
- A **Handheld Terminal** may come in two versions. In the first case, the handheld will be a standard HART FSK configuration unit (just add new device DDs or DOF files), just like the one used for everyday tasks such as routine maintenance and calibration checks. In the case of wireless support, the handheld is used to join a new instrument to an existing Wireless HART network. In the second case the handheld has a Wireless HART connection to the gateway and then down to an instrument and could be used for reading PV or diagnostics.



NFC,

Near Field Communication (NFC) is a set of short-range wireless technologies, typically requiring a distance of 4cm or less to initiate a connection. NFC allows you to share small payloads of data between an NFC tag and an Android-powered device, or between two Android-powered devices. NFC is an upgrade of the existing proximity card standard (RFID) that combines the interface of a smartcard and a reader into a single device. It allows users to seamlessly share content between digital devices, pay bills wirelessly or even use their cellphone as an electronic traveling ticket on existing contactless infrastructure already in use for public transportation.

The significant advantage of NFC over Bluetooth is the shorter set-up time. Instead of performing manual configurations to identify Bluetooth devices, the connection between two NFC devices is established at once (under a 1/10 second). Due to its shorter range, NFC provides a higher degree of security than Bluetooth and makes NFC suitable for crowded areas where correlating a signal with its transmitting physical device might otherwise prove impossible. NFC can also work when one of the devices is not powered by a battery (e.g. on a phone that may be turned off, a contactless smart credit card, etc.).

Near field communication (NFC) technology lets smartphones and other enabled devices communicate with other devices containing a NFC tag. Whether swiping your smartphone at the checkout lane in the grocery store, waving it over a display at a local museum, or bumping phones with a friend to share the latest games, near field technology lets you pay, play, and learn easily.

Tags can range in complexity. Simple tags offer just read and write semantics, sometimes with one-time-programmable areas to make the card read-only. More complex tags offer math operations, and have cryptographic hardware to authenticate access to a sector. The most sophisticated tags contain operating environments, allowing complex interactions with code executing on the tag. Android-powered devices with NFC simultaneously support three main modes of operation:

1. **Reader/writer mode**, allowing the NFC device to read and/or write passive NFC tags and stickers.
2. **P2P mode**, allowing the NFC device to exchange data with other NFC peers; this operation mode is used by Android Beam.
3. **Card emulation mode**, allowing the NFC device itself to act as an NFC card. The emulated NFC card can then be accessed by an external NFC reader, such as an NFC point-of-sale terminal.

RFID

Radio Frequency Identification (RFID) refers to a wireless system comprised of two components: tags and readers. The reader is a device that has one or more antennas that emit radio waves and receive signals back from the RFID tag. Tags, which use radio waves to communicate their identity and other information to nearby readers, can be passive or active. Passive RFID tags are powered by the reader and do not have a battery. Active RFID tags are powered by batteries. RFID tags can store a range of information from one serial number to several pages of data. Readers can be mobile so that they can be carried by hand, or they can be mounted on a post or overhead. Reader systems can also be built into the architecture of a cabinet, room, or building.

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

As stated above, an RFID tag consists of an integrated circuit and an antenna. The tag is also composed of a protective material that holds the pieces together and shields them from various environmental conditions. The protective material depends on the application. For example, employee ID badges containing RFID tags are typically made from durable plastic, and the tag is embedded between the layers of plastic. RFID tags come in a variety of shapes and sizes and are either passive or active. Passive tags are the most widely used, as they are smaller and less expensive to implement. Passive tags must be "powered up" by the RFID reader before they can transmit data. Unlike passive tags, active RFID tags have an onboard power supply (e.g., a battery), thereby enabling them to transmit data at all times. Smart labels differ from RFID tags in that they incorporate both RFID and barcode technologies.

RFID APPLICATIONS

- Fastags
- Inventory management
- Asset tracking
- Personnel tracking
- Controlling access to restricted areas
- ID Badging
- Supply chain management
- Counterfeit prevention (e.g. in the pharmaceutical industry)

Active, semi-passive and passive RFID tags are making RFID technology more accessible and prominent in our world. These tags are less expensive to produce, and they can be made small enough to fit on almost any product.

Active and semi-passive RFID tags use internal batteries to power their circuits. An active tag also uses its battery to broadcast radio waves to a reader, whereas a semi-passive tag relies on the reader to supply its power for broadcasting. Because these tags contain more hardware than passive RFID tags, they are more expensive. Active and semi-passive tags are reserved for costly items that are read over greater distances -- they broadcast high frequencies from 850 to 950 MHz that can be read 100 feet (30.5 meters) or more away. If it is necessary to read the tags from even farther away, additional batteries can boost a tag's range to over 300 feet (100 meters)

Like other wireless devices, RFID tags broadcast over a portion of the electromagnetic spectrum. The exact frequency is variable and can be chosen to avoid interference with other electronics or among RFID tags and readers in the form of tag interference or reader interference. RFID systems can use a cellular system called Time Division Multiple Access (TDMA) to make sure the wireless communication is handled properly.

Passive RFID tags rely entirely on the reader as their power source. These tags are read up to 20 feet (six meters) away, and they have lower production costs, meaning that they can be applied to less expensive merchandise. These tags are manufactured to be disposable, along with the disposable consumer goods on which they are placed. Whereas a railway car would have an active RFID tag, a bottle of shampoo would have a passive tag.

Another factor that influences the cost of RFID tags is data storage. There are three storage types: read-write, read-only and WORM (write once, read many). A read-write tag's data can be added to or overwritten. Read-only tags cannot be added to or overwritten -- they contain only the data that is stored in them when they were made. WORM tags can have additional data (like another serial number) added once, but they cannot be overwritten.

Active and semi-passive tags are more expensive, and RFID manufacturers typically do not quote prices for these tags without first determining their range, storage type and quantity.

LPWAN (low-power wide area network)

Low-power WAN (LPWAN) is a wireless wide area network technology that interconnects low-bandwidth, battery-powered devices with low bit rates over long ranges. Created for machine-to-machine (M2M) and internet of things (IoT) networks, LPWANs operate at a lower cost with greater power efficiency than traditional mobile networks. They are also able to support a greater number of connected devices over a larger area.

LPWANs can accommodate packet sizes from 10 to 1,000 bytes at uplink speeds up to 200 Kbps. LPWAN's long range varies from 2 km to 1,000 km, depending on the technology. Most LPWANs have a star topology where, similar to Wi-Fi, each endpoint connects directly to common central access points.

LPWAN is not a single technology, but a group of various low-power, wide area network technologies that take many shapes and forms. LPWANs can use licensed or unlicensed frequencies and include proprietary or open standard options.

The proprietary, unlicensed Sigfox is one of the most widely deployed LPWANs today. While it can deliver messages over distances of 30-50 km in rural areas, 3-10 km in urban settings and up to 1,000 km in line-of-site applications, its packet size is limited to 150 messages of 12 bytes per day. Downlink packets are smaller, limited to four messages of 8 bytes per day. Sending data back to endpoints can also be prone to interference.

Random phase multiple access, or RPMA, is a proprietary LPWAN. Though it has a shorter range (up to 50 km line of sight and with 5-10 km nonline of sight), it offers better bidirectional communication than Sigfox. However, because it runs in the 2.4 GHz spectrum, it is prone to interference from Wi-Fi, Bluetooth and physical structures. It also typically has higher power consumption than other LPWAN options.

The unlicensed LoRa, specified and backed by the LoRa Alliance, transmits in several sub-gigahertz frequencies, making it less prone to interference. A derivative of chirp spread spectrum (CSS) modulation, LoRa allows users to define packet size. LoRaWAN is the media access control (MAC) layer protocol that manages communication between LPWAN devices and gateways.

Weightless SIG has developed three LPWAN standards: The unidirectional Weightless-N, bidirectional Weightless-P and Weightless-W, which is also bidirectional and runs off of unused TV spectrum. Weightless-N and Weightless-P are often more popular options due to Weightless-W's shorter battery life. Weightless-N and Weightless-P run in the sub-1 GHz unlicensed spectrum but also support licensed spectrum operation using 12.5 kHz narrowband technology.

Narrowband-IoT (NB-IoT) and LTE-M are both 3rd Generation Partnership Project (3GPP) standards that operate on the licensed spectrum. While they have similar performance to other

standards, they operate on existing cellular infrastructure, allowing service providers to quickly add cellular IoT connectivity to their service portfolios.

NB-IoT, also known as CAT-NB1, operates on existing LTE and Global System for Mobile (GSM) infrastructure. It offers uplink and downlink rates of around 200 Kbps, using only 200 kHz of available bandwidth.

LTE-M, also known as CAT-M1, offers higher bandwidth than NB-IoT, and the highest bandwidth of any LPWAN technology.

Other LPWAN technologies include:

- Green OFDM from Green Waves Technologies
- DASH7 from Haystack Technologies Inc.
- Symphony Link from Link Labs Inc.
- Thing Park Wireless from Actility
- Ultra Narrow Band from various companies including Telensa, Nwave and Sigfox
- WAV IoT

With decreased power requirements, longer ranges and lower costs than traditional mobile networks, LPWANs enable a number of M2M and IoT applications, many of which were previously constrained by budgets and power issues. Choosing an LPWAN depends on the specific application, namely the desired speed, data amounts and area covered. LPWANs are best suited for applications requiring infrequent uplink message delivery of smaller messages. Most LPWAN technologies also have downlink capabilities.

LPWANs are commonly used in applications including Smart metering, smart lighting, asset monitoring and tracking, smart cities, precision agriculture, livestock monitoring, energy management, manufacturing, and industrial IoT deployments.

LoRa and LoRaWAN

LoRa is a radio modulation technique that is essentially a way of manipulating radio waves to encode information using a chirped (chirp spread spectrum technology), multi-symbol format. LoRa as a term can also refer to the systems that support this modulation technique or the communication network that IoT applications use. The main advantages of LoRa are its long-range capability and its affordability. A typical use case for LoRa is in smart cities, where low-powered and inexpensive internet of things devices (typically sensors or monitors) spread across a large area send small packets of data sporadically to a central administrator. It encodes information on radio waves using chirp pulses - similar to the way dolphins and bats communicate! LoRa modulated transmission is robust against disturbances and can be received across great distances.

LoRa is ideal for applications that transmit small chunks of data with low bit rates. Data can be transmitted at a longer range compared to technologies like WiFi, Bluetooth or ZigBee. These features make LoRa well suited for sensors and actuators that operate in low power.

LoRaWAN is a low-power, wide area networking protocol built on top of the LoRa radio modulation technique. It wirelessly connects devices to the internet and manages communication between end-node devices and network gateways. Usage of LoRaWAN in industrial spaces and smart cities is growing because it is an affordable long-range, bi-directional communication protocol with very low power consumption — devices can run for ten years on a small battery. It uses the unlicensed ISM (Industrial, Scientific, Medical) radio bands for network deployments.

An end device can connect to a network with LoRaWAN in two ways:

1. Over-the-air Activation (OTAA): A device has to establish a network key and an application session key to connect with the network.
2. Activation by Personalization (ABP): A device is hardcoded with keys needed to communicate with the network, making for a less secure but easier connection.

LoRaWAN is suitable for transmitting small size payloads (like sensor data) over long distances. LoRa modulation provides a significantly greater communication range with low bandwidths than other competing wireless data transmission technologies.

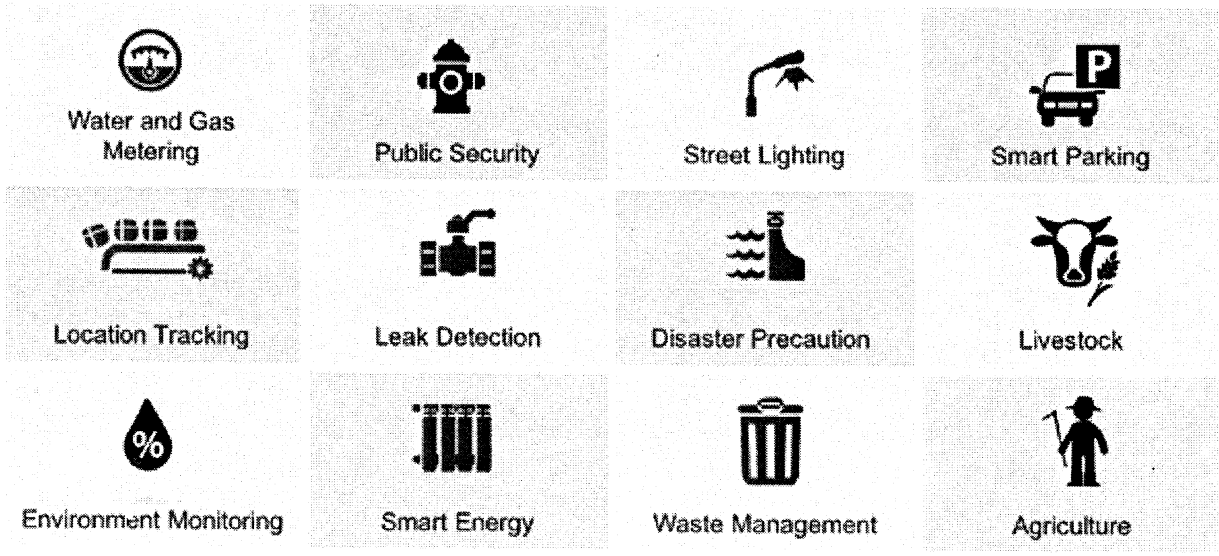
LoRaWAN Advantages

- **Ultra low power** - LoRaWAN end devices are optimized to operate in low power mode and can last up to 10 years on a single coin cell battery.
- **Long range** - LoRaWAN gateways can transmit and receive signals over a distance of over 10 kilometers in rural areas and up to 3 kilometers in dense urban areas.
- **Deep indoor penetration** - LoRaWAN networks can provide deep indoor coverage, and easily cover multi floor buildings.
- **License free spectrum** - You don't have to pay expensive frequency spectrum license fees to deploy a LoRaWAN network.
- **Geolocation**- A LoRaWAN network can determine the location of end devices using triangulation without the need for GPS. A LoRa end device can be located if at least three gateways pick up its signal.
- **High capacity** - LoRaWAN Network Servers handle millions of messages from thousands of gateways.
- **Public and private deployments** - It is easy to deploy public and private LoRaWAN networks using the same hardware (gateways, end devices, antennas) and software (UDP packet forwarders, Basic Station software, LoRaWAN stacks for end devices).
- **End-to-end security**- LoRaWAN ensures secure communication between the end device and the application server using AES-128 encryption.
- **Firmware updates over the air** - You can remotely update firmware (applications and the LoRaWAN stack) for a single end device or group of end devices.
- **Roaming**- LoRaWAN end devices can perform seamless handovers from one network to another.
- **Low cost** - Minimal infrastructure, low-cost end nodes and open source software.

- **Certification program-** The LoRa Alliance certification program certifies end devices and provides end-users with confidence that the devices are reliable and compliant with the LoRaWAN specification.
- **Ecosystem-** LoRaWAN has a very large ecosystem of device makers, gateway makers, antenna makers, network service providers, and application developers.

LoRaWAN Application

- **Vaccine cold chain monitoring** - LoRaWAN sensors are used to ensure vaccines are kept at appropriate temperatures in transit.
- **Animal conservation** - Tracking sensors manage endangered species such as Black Rhinos and Amur Leopards.
- **Dementia patients** - Wristband sensors provide fall detection and medication tracking.
- **Smart farms-** Real time insights into crop soil moisture and optimized irrigation schedule reduce water use up to 30%.
- **Water conservation-** Identification and faster repair of leaks in a city's water network.
- **Food safety-** Temperature monitoring ensures food quality maintenance.
- **Smart waste bins** - Waste bin level alerts sent to staff optimize the pickup schedule.
- **Smart bikes-** Bike trackers track bikes in remote areas and dense buildings.
- **Airport tracking** - GPS-free tracking monitors vehicles, personnel, and luggage.
- **Efficient workspaces** - Room occupancy, temperature, energy usage and parking availability monitoring.
- **Cattle health** - Sensors monitor cattle health, detect diseases and forecast calves delivery time.
- **LoRa in space** - Satellites to provide LoRaWAN-based coverage worldwide

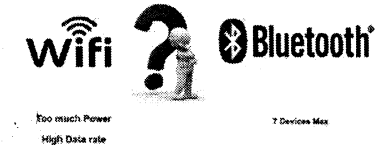


ZigBee zigbee

Zigbee is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power wireless IoT networks. The Zigbee standard operates on the IEEE 802.15.4 physical radio specification and operates in unlicensed bands including 2.4 GHz, 900 MHz and 868 MHz. The specification is a packet-based radio protocol intended for low-cost, battery-operated devices. The protocol allows devices to communicate in a variety of network topologies and can have battery life lasting several years.

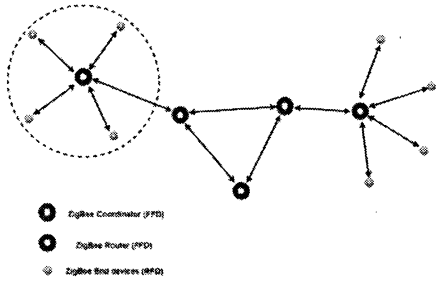
The Zigbee protocol has been created and ratified by member companies of the Zigbee Alliance. Over 300 leading semiconductor manufacturers, technology firms, OEMs and service companies comprise the Zigbee Alliance membership. The Zigbee protocol was designed to provide an easy-to-use wireless data solution characterized by secure, reliable wireless network architectures.

ZigBee is a Personal Area Network task group with low rate task group 4. It is a technology of home networking. ZigBee is a technological standard created for controlling and sensor the network. ZigBee is a standard that addresses the need of very low-cost implementation of Low power devices with Low data rate for short-range wireless communications.



Types of ZigBee Devices:

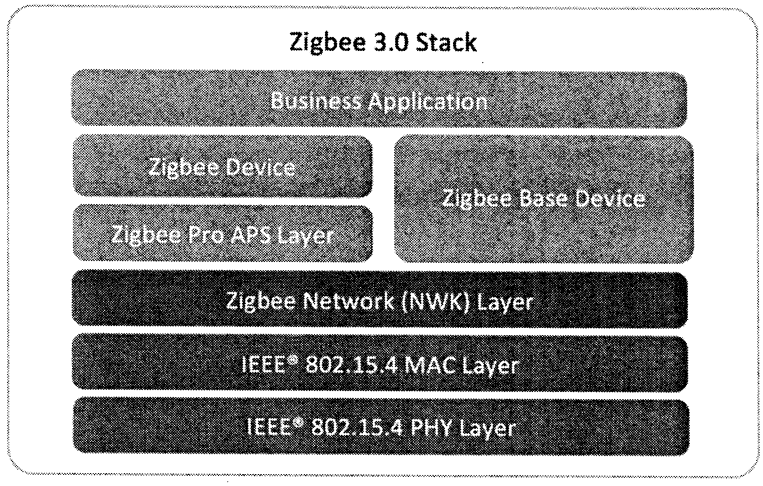
- **Zigbee Coordinator Device** – It communicates with routers. This device is used for connecting the devices.
- **Zigbee Router** – It is used for passing the data between devices.
- **Zigbee End Device** – It is the device that is going to be controlled



The Zigbee 3.0 protocol is designed to communicate data through noisy RF environments that are common in commercial and industrial applications. Version 3.0 builds on the existing Zigbee standard but unifies the market-specific application profiles to allow all devices to be wirelessly connected in the same network, irrespective of their market designation and function. Furthermore, a Zigbee 3.0 certification scheme ensures the interoperability of products from different manufacturers. Connecting Zigbee 3.0 networks to the IP domain opens up monitoring and control from devices such as smartphones and tablets on a LAN or WAN, including the Internet, and brings the true Internet of Things to fruition.

Zigbee protocol features include:

- Support for multiple network topologies such as point-to-point, point-to-multipoint and mesh networks
- Low duty cycle – provides long battery life, Low Power Consumption
- Low latency Network Join Time (~ 30 msec)
- Direct Sequence Spread Spectrum (DSSS)
- 128-bit AES encryption for secure data connections
- Collision avoidance, retries and acknowledgements
- Low Data Rate (20- 250 kbps)
- Short-Range (75-100 meters)
- Support Small and Large Networks (up to 65000 devices (Theory); 240 devices (Practically))
- Low Cost of Products and Cheap Implementation (Open Source Protocol)



The Zigbee 3.0 software stack incorporates a 'base device' that provides consistent behavior for commissioning nodes into a network. A common set of commissioning methods is provided, including Touchlink, a method of proximity commissioning. Zigbee 3.0 provides enhanced network security. There are two methods of security that give rise to two types of network:

- **Centralized security**: This method employs a coordinator/trust center that forms the network and manages the allocation of network and link security keys to joining nodes.
- **Distributed security:** This method has no coordinator/trust center and is formed by a router. Any Zigbee router node can subsequently provide the network key to joining nodes.

Nodes adopt whichever security method is used by the network they join. Zigbee 3.0 supports the increasing scale and complexity of wireless networks, and copes with large local networks of greater than 250 nodes. Zigbee also handles the dynamic behavior of these networks (with nodes appearing, disappearing and re-appearing in the network) and allows orphaned nodes, which result from the loss of a parent, to re-join the network via a different

parent. The self-healing nature of Zigbee Mesh networks also allows nodes to drop out of the network without any disruption to internal routing.

The backward compatibility of Zigbee 3.0 means that applications already developed under the Zigbee Light Link 1.0 or Home Automation 1.2 profile are ready for Zigbee 3.0. The Smart Energy profile is also compatible with Zigbee 3.0 at the functional level, but Smart Energy has additional security requirements that are only addressed within the profile.

Zigbee's Over-The-Air (OTA) upgrade feature for software updates during device operation ensures that applications on devices already deployed in the field can be seamlessly migrated to Zigbee 3.0. OTA upgrade is an optional functionality that manufacturers are encouraged to support in their Zigbee products.

A key component of the Zigbee protocol is the ability to support mesh networking. In a mesh network, nodes are interconnected with other nodes so that multiple pathways connect each node. Connections between nodes are dynamically updated and optimized through sophisticated, built-in mesh routing table.

Mesh networks are decentralized in nature; each node is capable of self-discovery on the network. Also, as nodes leave the network, the mesh topology allows the nodes to reconfigure routing paths based on the new network structure. The characteristics of mesh topology and ad-hoc routing provide greater stability in changing conditions or failure at single nodes.

Zigbee Applications

Zigbee enables broad-based deployment of wireless networks with low-cost, low-power solutions. It provides the ability to run for years on inexpensive batteries for a host of monitoring and control applications. Smart energy/smart grid, AMR (Automatic Meter Reading), lighting controls, building automation systems, tank monitoring, HVAC control, medical devices and fleet applications are just some of the many spaces where Zigbee technology is making significant advancements.

UNIT-III

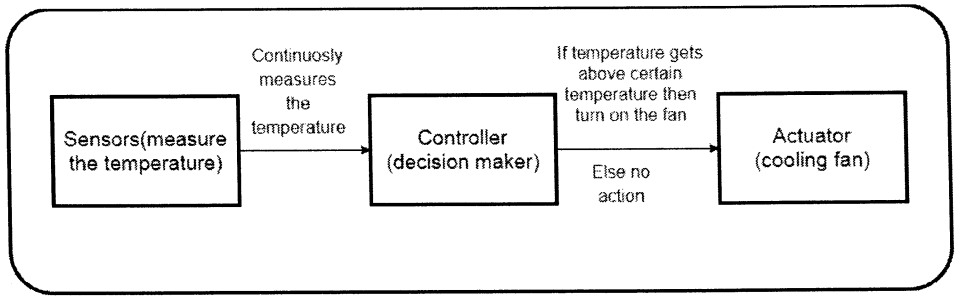
ACTUATION: ACTUATOR

An actuator is a part of a device or machine that helps it to achieve physical movements by converting energy, often electrical, air, or hydraulic, into mechanical force. Simply put, it is the component in any machine that enables movement.

Sometimes, to answer the question of what does an actuator do, the process is compared to the functioning of a human body. Like muscles in a body that enable energy to be converted to some form of motion like the movement of arms or legs, actuators work in a machine to perform a mechanical action.

An IoT device is made up of a Physical object ("thing") + Controller ("brain") + Sensors + Actuators + Networks (Internet). An actuator is a machine component or system that moves or controls the mechanism or the system. Sensors in the device sense the environment, then control signals are generated for the actuators according to the actions needed to perform.

A servo motor is an example of an actuator. They are linear or rotatory actuators, can move to a given specified angular or linear position. We can use servo motors for IoT applications and make the motor rotate to 90 degrees, 180 degrees, etc., as per our need. The following diagram shows what actuators do, the controller directs the actuator based on the sensor data to do the work.



TYPES OF ACTUATORS :

1. **Hydraulic Actuators – :**A hydraulic actuator uses hydraulic power to perform a mechanical operation. They are actuated by a cylinder or fluid motor. The mechanical motion is converted to rotary, linear, or oscillatory motion, according to the need of the IoT device. Ex- construction equipment uses hydraulic actuators because hydraulic actuators can generate a large amount of force.

- Hydraulic actuators can produce a large magnitude of force and high speed.
- Used in welding, clamping, etc.
- Used for lowering or raising the vehicles in car transport carriers.

➢ Hydraulic fluid leaks can cause efficiency loss and issues of cleaning.
➢ It is expensive.
➢ It requires noise reduction equipment, heat exchangers, and high maintenance systems.

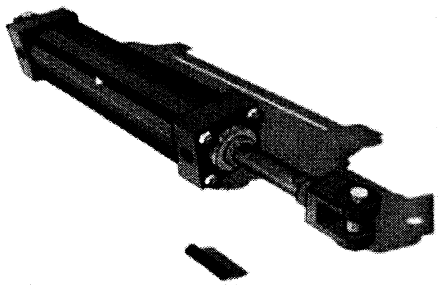


Fig: An oil based hydraulic actuator

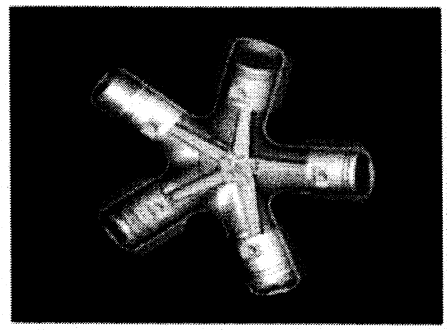


Fig: A radial engine acts as a hydraulic actuator

2. **Pneumatic Actuators – :** A pneumatic actuator uses energy formed by vacuum or compressed air at high pressure to convert into either linear or rotary motion. Example- Used in robotics, use sensors that work like human fingers by using compressed air. Examples of equipment that uses pneumatic actuators include: Bus brakes, Exercise machines, Vane motors, Pressure sensors, Pneumatic mailing systems
   - They are a low-cost option and are used at extreme temperatures where using air is a safer option than chemicals.
   - They need low maintenance, are durable, and have a long operational life.
   - It is very quick in starting and stopping the motion.

   ➢ Loss of pressure can make it less efficient.
   ➢ The air compressor should be running continuously.
   ➢ Air can be polluted, and it needs maintenance.

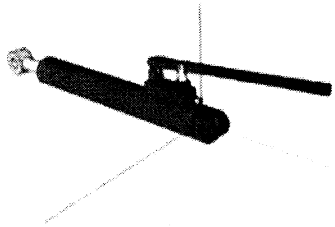


Fig: A manual linear pneumatic actuator

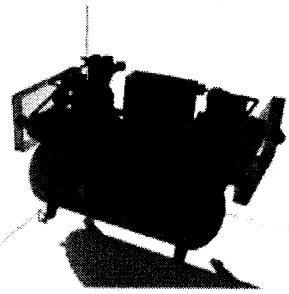


Fig: An air pump acts as a pneumatic actuator

3. **Electrical Actuators –** An electric actuator uses electrical energy, is usually actuated by a motor that converts electrical energy into mechanical torque. An example of an electric actuator is a solenoid based electric bell.
   - It has many applications in various industries as it can automate industrial valves.
   - It produces less noise and is safe to use since there are no fluid leakages.
   - It can be re-programmed and it provides the highest control precision positioning.
   - ➢ It is expensive.
   - ➢ It depends a lot on environmental conditions.

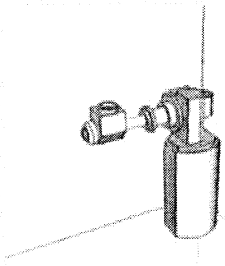


Fig: A motor drive-based rotary actuator

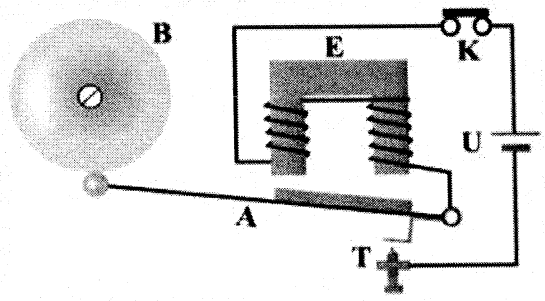


Fig: A solenoid based electric bell ringing mechanism

4. **Thermal/Magnetic Actuators –** These are actuated by thermal or mechanical energy. Shape Memory Alloys (SMAs) or Magnetic Shape Memory Alloys (MSMAs) are used by these actuators. An example of a thermal/magnetic actuator can be a piezo motor using SMA.
   - These can be actuated by applying thermal or magnetic energy.
   - They tend to be compact, lightweight, economical and with high power density.
   - These actuators use shape memory materials (SMMs), such as shape memory alloys (SMAs) or magnetic shape-memory alloys (MSMAs).

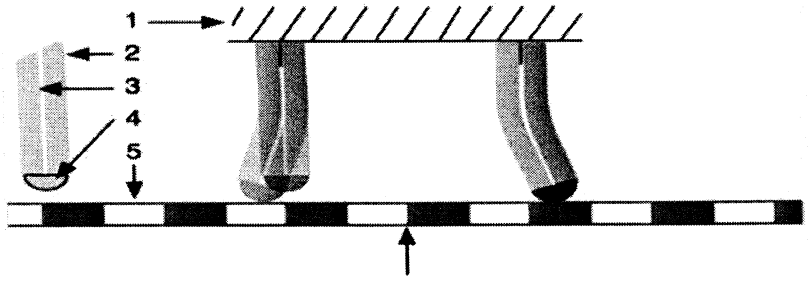


Fig: A piezo motor using SMA

5. **Mechanical Actuators -:** A mechanical actuator executes movement by converting rotary motion into linear motion. It involves pulleys, chains, gears, rails, and other devices to operate. Example - A crankshaft.
   - A mechanical actuator converts rotary motion into linear motion to execute some movement.
   - It involves gears, rails, pulleys, chains and other devices to operate.
   - Example : rack and pinion

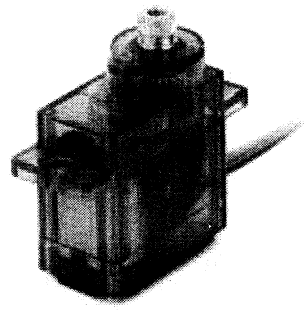
Fig: A crank shaft acting as a mechanical actuator

6. **Soft Actuators:** It is polymer based and are designed to handle fragile objects like fruit harvesting in agriculture or manipulating the internal organs in biomedicine.
   - Examples: Shape Memory polymers, Photo polymers
   - Shape memory polymer is functions similar to our muscles. It also provides response to range of stimuli e.g. light, electrical, heat, magnetic, pH, moisture changes etc. The advantages of such polymers are low density, high strain recovery, bio-compatibility, bio-degradability etc.
   - Photo polymers are known as light activated polymers. They are special type of shape memory polymers which are activated by light stimuli.

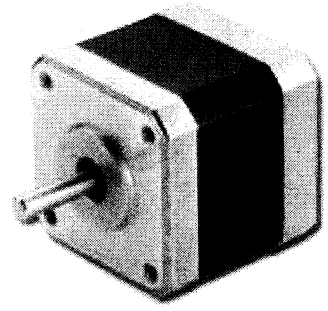
7. **Shape Memory Polymers actuator :** Shape memory (SMP) actuators function similar to our muscles, even providing a response to a range of stimuli such as light, electrical, magnetic, heat, PH, and moisture changes. SMP exhibits surprising features such a low density, high strain recovery, biocompatibility, and biodegradability.

TYPES OF MOTOR ACTUATORS

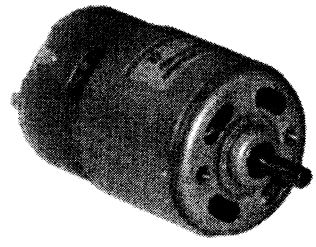
1. Servo Motors: A Servo is a small device that incorporates a two wire DC motor, a gear train, a potentiometer, an integrated circuit, and a shaft (output spine). The shaft can be positioned to specific angular positions by sending the servo a coded signal. Of the three wires that stick out from the servo casing, one is for power, one is for ground, and one is a control input line. When a control signal is applied to a Servo that represents a desired output position of the servo shaft, it (servo) applies power to its DC motor until its shaft turns to that position. It uses the position-sensing device to determine the rotational position of the shaft, so it knows which way the motor must turn to move the shaft to the commanded position.



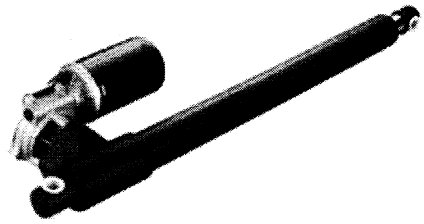
2. Stepper Motors: Stepper motors are DC motors that move in discrete steps. They have multiple coils that are organized in groups called "phases". By energizing each phase in sequence, the motor will rotate, one step at a time. With a computer controlled stepping, you can achieve very precise positioning and/or speed control. A servomotor consumes power as it rotates to the commanded position but then the servomotor rests. Stepper motors continue to consume power to lock in and hold the commanded position.



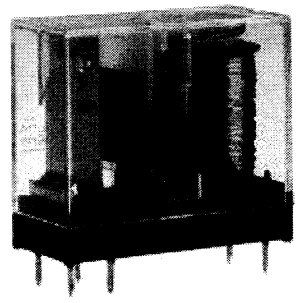
3. **DC Motors (Continuous Rotation Motors):** Direct Current (DC) motor is the most common actuator used in electronics projects. They are simple, cheap, and easy to use. Also, they come in a great variety of sizes, to accommodate different tasks. DC motors convert electrical into mechanical energy. They consist of permanent magnets and loops of wire inside. When current is applied, the wire loops generate a magnetic field, which reacts against the outside field of the static magnets.



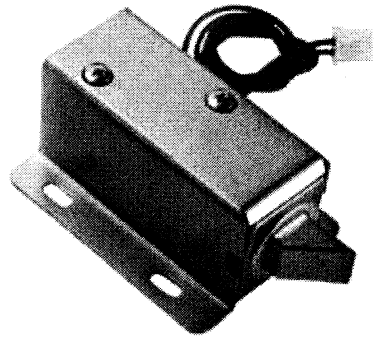
4. Linear actuator: A linear actuator is an actuator that creates motion in a straight line, in contrast to the circular motion of a conventional electric motor. Linear actuators are used in machine tools and industrial machinery, in computer peripherals such as disk drives and printers, in valves and dampers, and in many other places where linear motion is required.



5. Relay: A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. The advantage of relays is that it takes a relatively small amount of power to operate the relay coil, but the relay itself can be used to control motors, heaters, lamps or AC circuits which themselves can draw a lot more electrical power.

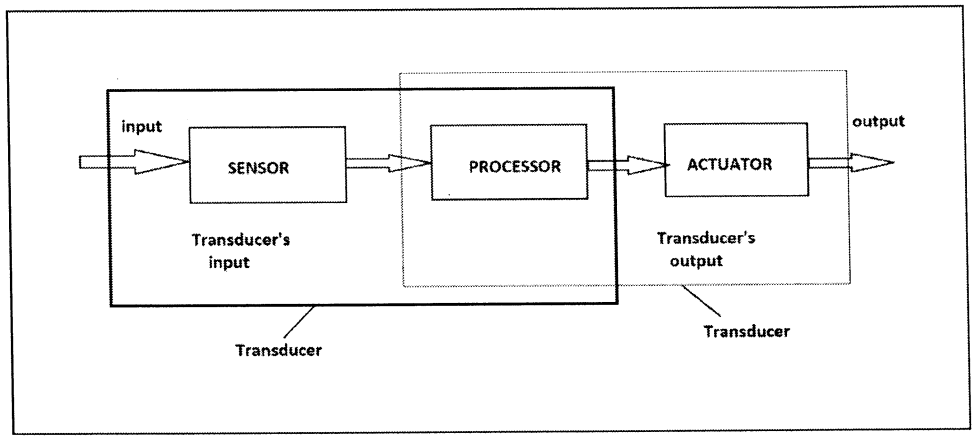


6. Solenoid: A solenoid is simply a specially designed electromagnet. Solenoids are inexpensive, and their use is primarily limited to on-off applications such as latching, locking, and triggering. They are frequently used in home appliances (e.g. washing machine valves), office equipment (e.g. copy machines), automobiles (e.g. door latches and the starter solenoid), pinball machines (e.g., plungers and bumpers), and factory automation.



SENSING: DEFINITION, TYPES OF SENSORS, TRANSDUCERS, SENSORS CLASSES

IoT applications, whether for city infrastructures, factories, or wearable devices, use large arrays of sensors collecting data for transmission over the Internet to a central, cloud-based computing resource. Analytics software running on the cloud computers reduces the huge volumes of generated data into actionable information for users, and commands to actuators back out in the field. So, sensors are one key factor in IoT success - but these are not conventional types, which simply convert physical variables into electrical signals. They have needed to evolve into something more sophisticated to perform a technically and economically viable role within the IoT environment. With improvements to fabrication, more integration and built-in intelligence, culminating in the concept of the smart sensors now in wide use.



IOT EXPECTATION TO ITS SENSORS

Sensors have traditionally been functionally simple devices that convert physical variables into electrical signals or changes in electrical properties. While this functionality is an essential starting-point, sensors need to add the following properties to perform as IoT components:

- Low cost, so they can be economically deployed in large numbers
- Physically small, to 'disappear' unobtrusively into any environment
- Wireless, as a wired connection is typically not possible
- Self-identification and self-validation
- Very low power, so that it can survive for years without a battery change, or manage with energy harvesting
- Robust, to minimize or eliminate maintenance

- Self-diagnostic and self-healing
- Self-calibrating, or accepts calibration commands via wireless link
- Data pre-processing, to reduce load on gateways, PLCs and cloud resources

Information from multiple sensors can be combined and correlated to solve problems. For example, temperature sensor and vibration sensor data can be used to detect the onset of mechanical failure. In some cases, the two sensor functions are available in one device; in others, the functions are combined in software to create a 'soft' sensor.

Industries and organizations have been using various kinds of sensors for a long time but the invention of the Internet of Things has taken the evolution of sensors to a completely different level. IoT platforms function and deliver various kinds of intelligence and data using a variety of sensors. They serve to collect data, pushing it and sharing it with a whole network of connected devices. All this collected data makes it possible for devices to autonomously function, and the whole ecosystem is becoming "smarter" every day.

Take Tesla vehicles as an example. All of the sensors on a car record their perception of the surroundings, uploading the information into a massive database. The data is then processed and all the important new pieces of information are sent to all other vehicles. This is an ongoing process, through which a whole fleet of Tesla vehicles is becoming smarter every day.

Some of the key sensors, extensively being used in the IoT world are as follows.

1. **Temperature sensors :** By definition, "A device, used to measure amount of heat energy that allows to detect a physical change in temperature from a particular source and converts the data for a device or user, is known as a Temperature Sensor." Following are some sub-categories of Temperatuer Sensors:
   - **Thermocouples:** These are voltage devices that indicate temperature measuring with a change in voltage. As temperature goes up, the output voltage of the thermocouple rises.
   - **Resistor temperature detectors (RTD):** The resistance of the device is directly proportional to the temperature, when the temperature rises resistance increase take place.

- **Thermistors:** It is a temperature sensitive resistor that changes its physical resistance with the change in temperature.
- **IC (Semiconductor):** They are linear devices where the conductivity of the semiconductor increases linearly and it takes advantage of the variable resistance properties of semiconductor materials. It can provide a direct temperature reading in digital form, especially at low temperatures.
- **Infrared sensors**: It detects temperature by intercepting a portion of emitted infrared energy of the object or substance, and sensing its intensity, can be used to measure temperature of solids and liquids only, not possible to use it on gases because of their transparent nature.

2. **Proximity sensor:** A device that detects the presence or absence of a nearby object, or properties of that object, and converts it into signal which can be easily read by user or a simple electronic instrument without getting in contact with them. This type of sensors are used in vehicles. When reversing the car an obstacle can be detected and alarmed using proximity sensor. They are also used for parking availability in places such as malls, stadiums or airports. Following are some of the Proximity Sensors sub-category:
   - **Inductive Sensors:** Inductive proximity sensors are used for non-contact detection to find out the presence of metallic objects using electromagnetic field. It can operate at higher speeds than mechanical switches and also seems more reliable because of its robustness.
   - **Capacitive Sensors:** Capacitive proximity sensors can detect both metallic as well as non-metallic targets. Nearly all other materials are dielectric different from air. It can be used to sense very small objects through a large portion of target. So, generally used in difficult and complicated applications.
   - **Photoelectric Sensors**: Photoelectric sensor is made up of light-sensitive parts and uses a beam of light to detect the presence or absence of an object. It is an ideal alternative of inductive sensors. And used for long distance sensing or to sense non-metal object.
   - **Ultrasonic Sensors:** Ultrasonic sensors are also used to detect the presence or to measure the distance of targets similar to radar or sonar. This makes a reliable solution for harsh and demanding conditions.
3. **Pressure sensor :** A pressure sensor is a device that senses pressure and converts it into an electric signal. Here, the amount depends upon the level of

pressure applied. There are plenty of devices that rely on liquid or other forms of pressure. These sensors make it possible to create IoT systems that monitor systems and devices that are pressure propelled. With any deviation from standard pressure range, the device notifies the system administrator. Deployment of these sensors is not only very useful in manufacturing, but also in the maintenance of whole water systems and heating systems, as it is easy to detect any fluctuation or drops in pressure.

4. **Water quality sensors** are used to detect the water quality and Ion monitoring primarily in water distribution systems. Water is practically used everywhere. These sensors play an important role as they monitor the quality of water for different purposes. They are used in a variety of industries. Following is a list of the most common kind of water sensors in use:
   - **Chlorine Residual Sensor:** It measures chlorine residual (i.e. free chlorine, monochloramine & total chlorine) in water and most widely used as disinfectant because of its efficiency.
   - **Total Organic Carbon Sensor:** TOC sensor is used to measure organic element in water.
   - **Turbidity Sensor:** Turbidity sensors measure suspended solids in water, typically it is used in river and stream gaging, wastewater and effluent measurement.
   - **Conductivity Sensor:** Conductivity measurements are carried out in industrial processes primarily to obtain information on total ionic concentrations (i.e. dissolved compounds) in water solutions.
   - **pH Sensor:** It is used to measure the pH level in the dissolved water, which indicates how acidic or basic (alkaline) it is.
   - **Oxygen-Reduction Potential Sensor:** The ORP measurement provides insights into the level of oxidation/reduction reactions occurring in the solution.
5. **Chemical sensors** are applied in a number of different industries. Their goal is to indicate changes in liquid or to find out air chemical changes. They play an important role in bigger cities, where it is necessary to track changes and protect the population. Main use cases of chemical sensors can be found in Industrial environmental monitoring and process control, intentionally or accidentally released harmful chemical detection, explosive and radioactive

detection, recycling processes on Space Station, pharmacy industries and laboratory etc. Following are most common kind of chemical sensors in use:

- Chemical field-effect transistor
- Chemiresistor
- Electrochemical gas sensor
- Fluorescent chloride sensor
- Hydrogen sulfide sensor
- Nondispersive infrared sensor
- pH glass electrode
- Potentiometric sensor
- Zinc oxide nanorod sensor

6. **Gas sensor:** Gas sensors are similar to the chemical ones, but are specifically used to monitor changes of the air quality and detect the presence of various gases. Like chemical sensors, they are used in numerous industries such as manufacturing, agriculture and health and used for air quality monitoring, detection of toxic or combustible gas, hazardous gas monitoring in coal mines, oil & gas industries, chemical laboratory research, manufacturing - paints, plastics, rubber, pharmaceutical & petrochemical etc. Following are some common Gas sensors:
   - Carbon dioxide sensor
   - Breathalyzer
   - Carbon monoxide detector
   - Catalytic bead sensor
   - Hydrogen sensor
   - Air pollution sensor
   - Nitrogen oxide sensor
   - Oxygen sensor
   - Ozone monitor
   - Electrochemical gas sensor
   - Gas detector
   - Hygrometer
7. **Smoke sensor:** A smoke sensor is a device that senses smoke (airborne particulates & gases), and it's level. They have been in use for a long period of time. However, with the development of IoT, they are now even more effective, as they are plugged into a system that immediately notifies the user about any problem that occurs in different industries. Smoke sensors are

extensively used by manufacturing industry, buildings and accommodation infra to detect fire and gas incidences. This serves to protect people working in dangerous environments, as the whole system is much more effective in comparison to the older ones.

8. **IR sensors :** An infrared sensor is a sensor which is used to sense certain characteristics of its surroundings by either emitting or detecting infrared radiation. It is also capable of measuring the heat being emitted by the objects. They are now used in Healthcare as they make monitoring of blood flow and blood pressure simple. They are even used in a wide array of regular smart devices such as smart watches and smartphones as well. Other common use includes home appliances & remote control, breath analysis, Infrared vision i.e. visualize heat leaks in electronics, monitor blood flow, wearable electronics, optical communication, non-contact based temperature measurements, automotive blind-angle detection. They are going to play an important role in the smart home industry, as they have a wide-range of applications.
9. **Level sensors :** A sensor which is used to determine the level or amount of fluids, liquids or other substances that flow in an open or closed system is called Level sensor. Like IR sensors, level sensors are present in a wide array of industries. They are primarily known for measuring fuel levels, but they are also used in businesses that work with liquid materials. For example, the recycling industry, as well as the juice and alcohol industry rely on these sensors to measure the number of liquid assets in their possession. Best use cases of level sensor is, fuel gauging & liquid levels in open or closed containers, sea level monitoring & Tsunami warning, water reservoirs, medical equipment, compressors, hydraulic reservoirs, machine tools, beverage and pharmaceutical processing, high or low-level detection etc. There are two basic level measurement types:
    - **Point level sensors:** Point level sensors usually detect the particular specific level and respond to the user if the sensing object is above or below that level. It is integrated into single device to get an alarm or trigger
    - **Continuous level Sensor:** Continuous level sensors measure liquid or dry material levels within a specified range and provide outputs which continuously indicate the level. The best example of it is fuel level display in the vehicle.

10. **Image sensors :** Image sensors are instruments which are used to convert optical images into electronic signals for displaying or storing files electronically. The major use of image sensor is found in digital camera & modules, medical imaging and night vision equipment, thermal imaging devices, radar, sonar, media house, Biometric & IRIS devices. Two main types of sensors are used in:
    a. CCD (charge-coupled device), and
    b. CMOS (complementary metal-oxide semiconductor) imagers.

    Although each type of sensor uses different technology to capture images, both CCD and CMOS imagers use metal-oxide semiconductors, having the same degree of sensitivity to light, and no inherent quality difference. An average consumer would think that this is a regular camera, but even though this is not far from the truth, image sensors are connected with a wide range of different devices, making their functionality much better.

    One of the best-known uses includes the car industry, in which imagery plays a very important role. With these sensors, the system can recognize signs, obstacles and many other things that a driver would generally notice on the road. They play a very important role in IoT industry, as they directly affect the progress of driverless cars.

    In the retail industry, these sensors serve to collect data about customers, helping businesses get a better insight into who is actually visiting their store, race, gender, age are only some of the useful parameters that retail owners get by using these IoT sensors.
11. **Motion detection sensors:** A motion detector is an electronic device which is used to detect the physical movement (motion) in a given area and it transforms motion into an electric signal; motion of any object or motion of human beings. Motion detection plays an important role in the security industry. Businesses utilize these sensors in areas where no movement should be detected at all times, and it is easy to notice anybody's presence with these sensors installed.

    These are primarily used for intrusion detection systems, automatic door control, boom barrier, smart camera (i.e motion based capture/video recording),toll plaza, automatic parking systems, automated sinks/toilet flusher, hand dryers, energy management systems(i.e. Automated Lighting, AC, Fan, Appliances Control) etc.

    On the other hand, these sensors can also decipher different types of movements, making them useful in some industries where a customer can communicate with the system by waving a hand or by performing a similar action. For example, someone can wave to a sensor in the retail store to request assistance with making the right purchase decision.

Following are key motion sensor types widely used:

- **Passive Infrared (PIR):** It Detects body heat (infrared energy) and the most widely used motion sensor in home security systems.
- **Ultrasonic:** Sends out pulses of ultrasonic waves and measures the reflection off a moving object by tracking the speed of sound waves.
- **Microwave:** Sends out radio wave pulses and measures the reflection off a moving object. They cover a larger area than infrared & ultrasonic sensors, but they are vulnerable to electrical interference and more expensive.

12. **Accelerometer sensors:** Accelerometer is a transducer that is used to measure the physical or measurable acceleration experienced by an object due to inertial forces and converts the mechanical motion into an electrical output. It is defined as rate of change of velocity with respect to time
    These sensors are now present in millions of devices, such as smartphones. Their uses involve detection of vibrations, tilting and acceleration in general. This is great for monitoring your driving fleet, or using a smart pedometer. In some instances, it is used as a form of anti-theft protection, as the sensor can send an alert through the system if an object that should remain stationary is moved. They are widely used in cellular & media devices, vibration measurement, automotive control and detection, free fall detection, aircraft and aviation industries, movement detection, sports academy/athletes behavior monitoring, consumer electronics, industrial & construction sites etc. following are few mainly used accelerometers
    - **Hall-effect accelerometers:** Hall-effect accelerometers are using Hall principle to measure the acceleration, it measures the voltage variations caused by changes in a magnetic field around them.
    - **Capacitive accelerometers:** Capacitive accelerometers sensing output voltage dependents on the distance between two planar surfaces. Capacitive accelerometers are also less prone to noise and variation with temperature.
    - **Piezoelectric accelerometers:** Piezoelectric sensing principle is working on the piezoelectric effect. Piezo-film based accelerometers are best used to measure vibration, shock, and pressure.
13. **Gyroscope sensors:** A sensor or device which is used to measure the angular rate or angular velocity is known as Gyro sensors, Angular velocity is simply defined as a measurement of speed of rotation around an axis. It is a device used primarily for navigation and measurement of angular and rotational velocity in 3-axis directions. The most important application is monitoring the orientation of an object. Their main applications are in car navigation systems, game controllers, cellular & camera devices, consumer electronics, robotics control,

drone & RC control helicopter or UAV control, vehicle control/ADAS and many more. There are several different kinds of gyro sensors which are selected by their working mechanism, output type, power, sensing range and environmental conditions.

- Rotary (classical) gyroscopes
- Vibrating Structure Gyroscope
- Optical Gyroscopes
- MEMS(micro-electro-mechanical systems) Gyroscopes

These sensors are always combined with accelerometers. The use of these two sensors simply provides more feedback to the system.

14. **Humidity sensors:** Humidity is defined as the amount of water vapour in an atmosphere of air or other gases. The most commonly used terms are "Relative Humidity (RH). These sensors usually follow the use of temperature sensors, as many manufacturing processes require perfect working conditions. Through measuring humidity, you can ensure that the whole process runs smoothly, and when there is any sudden change, action can be taken immediately, as sensors detect the change almost instantaneously.

    Their applications and use can be found in Industrial & residential domain for heating, ventilating, and air conditioning systems control. They can also be found in Automotive, museums, industrial spaces and greenhouses , meteorology stations, Paint and coatings industries, hospitals & pharma industries to protect medicines

15. **Optical sensors :** A sensor which measures the physical quantity of light rays and convert it into electrical signal which can be easily readable by user or an electronic instrument/device is called optical sensor. Optical sensors are loved by IoT experts, as they are practical for measuring different things simultaneously. The technology behind this sensor allows it to monitor electromagnetic energy, which includes, electricity, light and so on.

    Due to this fact, these sensors have found use in healthcare, environment monitoring, energy, aerospace and many more industries. With their presence oil companies, pharmaceutical companies and mining companies are in a much better position to track environmental changes while keeping their employees safe. Their main use can be found in ambient light detection, digital optical switches, optical fibres communications,due to electrical isolation best suited for oil and gas applications, civil and transportation fields, high speed network systems, elevator door control, assembly line part counters and safety systems.

    Following are key type of optical sensors:

    - **Photo detector:** It uses light sensitive semiconductor materials like photocells, photodiodes or photo transistors to work as photo detector.

- **Pyrometer:** It estimates the temperature of an object by sensing the color of the light and Objects radiate light according to their temperature and produce same colors at same temperature.
- **Proximity & Infrared:** Proximity use light to sense objects nearby and Infrared are used where visible light would be inconvenient.

Sensor is a device that provides a usable output in response to a specified measurement. The sensor attains a physical parameter and converts it into a signal suitable for processing (e.g. electrical, mechanical, optical) the characteristics of any device or material to detect the presence of a particular physical quantity. The output of the sensor is a signal which is converted to a human-readable form like changes in characteristics, changes in resistance, capacitance, impedance etc.

SENSORS CHARACTERISTICS:

Static characteristics : It is about how the output of a sensor changes in response to an input change after steady state condition.

- **Accuracy** - Accuracy is the capability of measuring instruments to give a result close to the true value of the measured quantity. It measures errors. It is measured by absolute and relative errors. Express the correctness of the output compared to a higher prior system.

  Absolute error = Measured value - True value

  Relative error = Measured value/True value
- **Range** - Gives the highest and the lowest value of the physical quantity within which the sensor can actually sense. Beyond these values, there is no sense or no kind of response. e.g. RTD for measurement of temperature has a range of -200`c to 800`c.
- **Resolution** - Resolution is an important specification towards selection of sensors. The higher the resolution, better the precision. Resolution is one of the first specifications you might look at when choosing a sensor for precise inspections. Resolution tells you the smallest change in distance a sensor can detect. So, a resolution of 0.01 mm means that the sensor can detect changes in distance of 0.01 mm, or 10 micrometers.
- **Precision** -: It is the capacity of a measuring instrument to give the same reading when repetitively measuring the same quantity under the same prescribed conditions. It implies agreement between successive readings, NOT closeness to the true value. It is related to the variance of a set of measurements. It is a necessary but not sufficient condition for accuracy.
- **Sensitivity** -: Sensitivity indicates the ratio of incremental change in the response of the system with respect to incremental change in input

parameters. It can be found from the slope of the output characteristics curve of a sensor. It is the smallest amount of difference in quantity that will change the instrument's reading.

- **Linearity** - The deviation of the sensor value curve from a particular straight line. Linearity is determined by the calibration curve. The static calibration curve plots the output amplitude versus the input amplitude under static conditions. A curve's slope resemblance to a straight line describes the linearity.
- **Drift** - The difference in the measurement of the sensor from a specific reading when kept at that value for a long period of time.
- **Repeatability** - The deviation between measurements in a sequence under the same conditions. The measurements have to be made under a short enough time duration so as not to allow significant long-term drift.

Sensor Classification :

- **Passive v/s Active** Active sensors have its own source of light or illumination. In particular, it actively sends a pulse and measures the backscatter reflected to the sensor. But passive sensors measure reflected sunlight emitted from the sun. When the sun shines, passive sensors measure this energy. RADAR and LiDAR are examples of active remote sensing where the time delay between emission and return is measured, establishing the location, speed and direction of an object. Passive sensors gather radiation that is emitted or reflected by the object or surrounding areas.
- **Analog v/s digital** Analog sensors are those which produce an analog signal based on what they sense. Similarly, digital signals are those which produce a digital signal in response to what they measure at the input. Quantities such as temperature, speed, displacement, pressure, strain etc. are analog quantities as they are continuous in nature. Example: Temperature of liquid can be measured using thermometer or thermocouple which continuously responds to changes in temperature as liquid is heated up or cooled down. Digital sensors overcome limitations of analog sensors counterpart. It response in binary nature. Design to overcome the disadvantages of analog sensors. Along with the analog sensor, it also comprises extra electronics for bit conversion. Example - Passive infrared (PIR) sensor and digital temperature sensor (DS1620).

- **Scalar v/s vector**

Scalar sensor - Detects the input parameter only based on its magnitude. The answer for the sensor is a function of magnitude of some input parameter.

Not affected by the direction of input parameters. Example - temperature, gas, strain, color and smoke sensor.

- **Vector sensor -** The response of the sensor depends on the magnitude of the direction and orientation of input parameter. Example - Accelerometer, gyroscope, magnetic field and motion detector sensors.

UNIT-IV

INTRODUCTION TO ARDUINO PROGRAMMING

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing. Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the **Ivrea Interaction Design Institute** as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments.

The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage over other systems:

- **Inexpensive -** Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules very less.
- **Cross-platform -** The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well.
- **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

When we work with Arduino we commonly use the Arduino IDE (Integrated Development Environment), a software available for all the major desktop platforms (macOS, Linux, Windows), which gives us 2 things: a programming editor with integrated libraries support, and a way to easily compile and load our Arduino programs to a board connected to the computer.

The Arduino Programming Language is basically a framework built on top of C++. You can argue that it's not a real programming language in the traditional term, but I think this helps avoiding confusion for beginners. A program written in the Arduino Programming Language is called sketch. A sketch is normally saved with the .ino extension (from Arduino).

The main difference from "normal" C or C++ is that you wrap all your code into 2 main functions. You can have more than 2, of course, but any Arduino program must provide at least those 2. One is called setup(), the other is called loop(). The first is called once, when the program starts, the second is repeatedly called while your program is running.

We don't have a main() function like you are used to in C/C++ as the entry point for a program. Everything else is normal C++ code, and as C++ is a superset of C, any valid C is also valid Arduino code.

Part of the Arduino Programming Language is the built-in libraries that allow you to easily integrate with the functionality provided by the Arduino board.

Simple Arduino program making an led turn on the light, and then turn off. To do so, you will use the pinMode(), delay() and digitalWrite() functions, along with some constants like HIGH, LOW, OUTPUT.

```
#define LED_PIN 13

void setup() {
    // Configure pin 13 to be a digital output
    pinMode(LED_PIN, OUTPUT);
}

void loop() {
    // Turn on the LED
    digitalWrite(LED_PIN, HIGH);
    // Wait 1 second (1000 milliseconds)
    delay(1000);
    // Turn off the LED
    digitalWrite(LED_PIN, LOW);
    // Wait 1 second
    delay(1000);
}
```

THE ARDUINO PROGRAMMING LANUAGE BUILT-IN CONSTANTS

Arduino sets two constants we can use to : **HIGH** equates to a high level of voltage, which can differ depending on the hardware (>2V on 3.3V boards like Arduino Nano, >3V on 5V boards like Arduino Uno) **LOW** equates to a low level of voltage. Again, the exact value depends on the board used

Then we have 3 constants we can use in combination with the pinMode() function:

- **INPUT** sets the pin as an input pin
- **OUTPUT** sets the pin as an output pin
- **INPUT_PULLUP** sets the pin as an internal pull-up resistor
- The other constant we have is **LED_BUILTIN**, which points to the number of the on-board pin, which usually equates to the number 13.

In addition to this, we have the C/C++ constants **TRUE** and **FALSE**.

ARDUINO MATH CONSTANTS

- **M_PI** the constant pi (3.14159265358979323846)
- **M_E** the constant e
- **M_LN10** the natural logarithm of the number 10.
- **M_LN2** the natural logarithm of the number 2.
- **M_LOG10E** the logarithm of the e to base 10.
- **M_LOG2E** the logarithm of the e to base 2.
- **M_SQRT2** the square root of 2.
- **NAN** the NAN (not a number) constant.

ARDUINO PROGRAMMING LANGUAGE BUILT-IN FUNCTIONS

built-in functions provided by the Arduino Programming Language.

Program lifecycle

- **setup()** this function is called once, when the program starts, and when the Arduino is shut down and restarted.
- **loop()** this function is repeatedly called while the Arduino program is running.

Handling I/O :The following functions help with handling input and output from Arduino device.

- **digitalRead()** reads the value from a digital pin. Accepts a pin number as a parameter, and returns the HIGH or LOW constant.
- **digitalWrite()** writes a HIGH or LOW value to a digital output pin. You pass the pin number and HIGH or LOW as parameters.
- **pinMode()** sets a pin to be an input, or an output. You pass the pin number and the INPUT or OUTPUT value as parameters.
- **pulseIn()** reads a digital pulse from LOW to HIGH and then to LOW again, or from HIGH to LOW and to HIGH again on a pin. The program will block until the pulse is detected. You specify the pin number and the kind of pulse you want to detect (LHL or HLH). You can specify an optional timeout to stop waiting for that pulse.
- **pulseInLong()** is same as pulseIn(), except it is implemented differently and it can't be used if interrupts are turned off. Interrupts are commonly turned off to get a more accurate result.
- **shiftIn()** reads a byte of data one bit at a time from a pin.
- **shiftOut()** writes a byte of data one bit at a time to a pin.
- **tone()** sends a square wave on a pin, used for buzzers/speakers to play tones. You can specify the pin, and the frequency. It works on both digital and analog pins.
- **noTone()** stops the tone() generated wave on a pin.
- analogRead() reads the value from an analog pin.
- analogReference() configures the value used for the top input range in the analog input, by default 5V in 5V boards and 3.3V in 3.3V boards.
- analogWrite() writes an analog value to a pin
- analogReadResolution() lets you change the default analog bits resolution for analogRead(), by default 10 bits. Only works on specific devices (Arduino Due, Zero and MKR)
- analogWriteResolution() lets you change the default analog bits resolution for analogWrite(), by default 10 bits. Only works on specific devices (Arduino Due, Zero and MKR)

TIME FUNCTIONS

- delay() pauses the program for a number of milliseconds specified as parameter
- delayMicroseconds() pauses the program for a number of microseconds specified as parameter
- micros() the number of microseconds since the start of the program. Resets after ~70 minutes due to overflow
- millis() the number of milliseconds since the start of the program. Resets after ~50 days due to overflow

MATH FUNCTIONS there are many built-in mathematical functions if you need them, some of the commonly used functions are:

- **abs()** the absolute value of a number
- **constrain()** constrains a number to be within a range, see usage
- **map()** re-maps a number from one range to another, see usage
- **max()** the maximum of two numbers
- **min()** the minimum of two numbers
- **pow()** the value of a number raised to a power
- **sq()** the square of a number
- **sqrt()** the square root of a number
- **cos()** the cosine of an angle
- **sin()** the sine of an angle
- **tan()** the tangent of an angle

WORKING WITH ALPHANUMERIC CHARACTERS

- **isAlpha()** checks if a char is alpha (a letter)
- **isAlphaNumeric()** checks if a char is alphanumeric (a letter or number)
- **isAscii()** checks if a char is an ASCII character
- **isControl()** checks if a char is a control character
- **isDigit()** checks if a char is a number
- **isGraph()** checks if a char is a printable ASCII character, and contains content
- **isHexadecimalDigit()** checks if a char is an hexadecimal digit (A-F 0-9)
- **isLowerCase()** checks if a char is a letter in lower case
- **isPrintable()** checks if a char is a printable ASCII character

- **isPunct()** checks if a char is a punctuation (a comma, a semicolon, an exclamation mark etc)
- **isSpace()** checks if a char is a space, form feed \f, newline \n, carriage return \r, horizontal tab \t, or vertical tab \v.
- **isUpperCase()** checks if a char is a letter in upper case
- **isWhitespace()** checks if a char is a space character or an horizontal tab \t

RANDOM NUMBERS GENERATION

- **random()** generate a pseudo-random number
- **randomSeed()** initialize the pseudo-random number generator with an arbitrary initial number. In Arduino, like in most languages, it's impossible to get really random numbers, and the sequence is always the same, so you seed it with the current time or you can read the input from an analog port.

WORKING WITH BITS AND BYTES

- **bit()** computes the value of a bit (0 = 1, 1 = 2, 2 = 4, 3 = 8...)
- **bitClear()** clear (sets to 0) a bit of a numeric variable. Accepts a number, and the number of the bit starting from the right
- **bitRead()** read a bit of a number. Accepts a number, and the number of the bit starting from the right
- **bitSet()** sets to 1 a bit of a number. Accepts a number, and the number of the bit starting from the right
- **bitWrite()** there are more built-in mathematical functions if you need them write 1 or 0 to a specific bit of a number Accepts a number, the number of the bit starting from the right, and the value to write (0 or 1)
- **highByte()** get the high-order (leftmost) byte of a word variable (has 2 bytes)
- **lowByte()** get the low-order (rightmost) byte of a word variable (has 2 bytes)

INTERRUPTS

- **noInterrupts()** disables interrupts
- **interrupts()** re-enables interrupts after they've been disabled
- **attachInterrupt()** allow a digital input pin to be an interrupt. Different boards have different allowed pins, check the official docs.
- **detachInterrupt()** disables an interrupt enabled using attachInterrupt()

OPERATORS IN ARDUINO

An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators –

- Arithmetic Operators
- Comparison Operators
- Boolean Operators
- Bitwise Operators
- Compound Operators

Arithmetic Operators : Assume variable A holds 10 and variable B holds 20 then

Operator name	Operator simple	Description	Example
assignment operator	=	Stores the value to the right of the equal sign in the variable to the left of the equal sign.	A = B
addition	+	Adds two operands	A + B will give 30
subtraction	-	Subtracts second operand from the first	A - B will give -10
multiplication	*	Multiply both operands	A * B will give 200
division	/	Divide numerator by denominator	B / A will give 2
modulo	%	Modulus Operator and remainder of after an integer division	B % A will give 0

Comparison Operators: Assume variable A holds 10 and variable B holds 20 then –

Operator name	Operator simple	Description	Example
equal to	==	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(A == B) is not true

not equal to	!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(A != B) is true
less than	<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true
greater than	>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true
less than or equal to	<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true
greater than or equal to	>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true

Boolean Operators: Assume variable A holds True and variable B holds false then

Operator name	Operator simple	Description	Example
and	&&	Called Logical AND operator. If both the operands are non-zero then then condition becomes true.	(A && B) is false
or	||	Called Logical OR Operator. If any of the two operands is non-zero then then condition becomes true.	(A || B) is true
not	!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true

Bitwise Operators: Assume variable A holds 60 and variable B holds 13 then –

Operator name	Operator simple	Description	Example
and	&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
or	|	Binary OR Operator copies a bit if it exists in either operand	(A | B) will give 61 which is 0011 1101
xor	^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
not	~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -60 which is 1100 0011
shift left	<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
shift right	>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

Compound Operators : Assume variable A holds 10 and variable B holds 20 then –

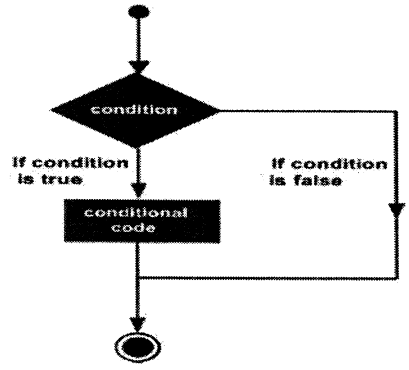
Operator name	Operator simple	Description	Example
increment	++	Increment operator, increases integer value by one	A++ will give 11
decrement	--	Decrement operator, decreases integer value by one	A-- will give 9
compound	+=	Add AND assignment operator. It	B += A is

addition		adds right operand to the left operand and assign the result to left operand	equivalent to B = B+ A
compound subtraction	-=	Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand	B -= A is equivalent to B = B - A
compound multiplication	*=	Multiply AND assignment operator. It multiplies right operand with the left operand and assign the result to left operand	B*= A is equivalent to B = B* A
compound division	/=	Divide AND assignment operator. It divides left operand with the right operand and assign the result to left operand	B /= A is equivalent to B = B / A
compound modulo	%=	Modulus AND assignment operator. It takes modulus using two operands and assign the result to left operand	B %= A is equivalent to B = B % A
compound bitwise or	|=	bitwise inclusive OR and assignment operator	A |= 2 is same as A = A | 2
compound bitwise and	&=	Bitwise AND assignment operator	A &= 2 is same as A = A & 2

CONTROL STATEMENT

Decision making structures require that the programmer specify one or more conditions to be evaluated or tested by the program. It should be along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Following is the general form of a typical decision making structure found:

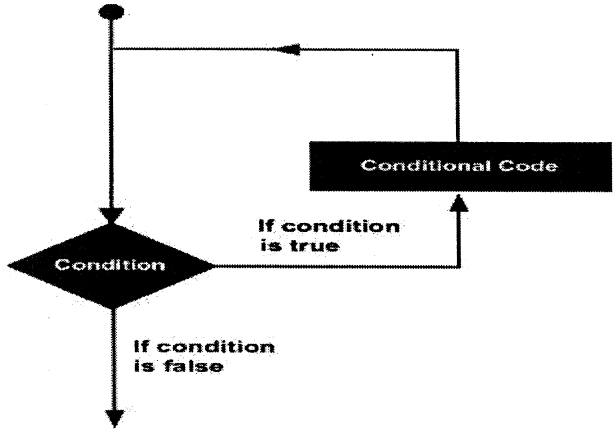


Control Statements are elements in Source Code that control the flow of program execution. They are –

S.NO.	Control Statement & Description
1	**If statement :** It takes an expression in parenthesis and a statement or block of statements. If the expression is true then the statement or block of statements gets executed otherwise these statements are skipped.
2	**If ...else statement :** An if statement can be followed by an optional else statement, which executes when the expression is false.
3	**If...else if ...else statement :** The if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement.
4	**switch case statement:** Similar to the if statements, switch...case controls the flow of programs by allowing the programmers to specify different codes that should be executed in various conditions.
5	**Conditional Operator ? :** The conditional operator ? : is the only ternary operator in C.

LOOPS

A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages. types of loops to handle looping requirements.



S.NO.	Loop & Description
1	**while loop:** while loops will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false. Something must change the tested variable, or the while loop will never exit.
2	**do...while loop:** The do...while loop is similar to the while loop. In the while loop, the loop-continuation condition is tested at the beginning of the loop before performed the body of the loop.
3	**for loop:** A for loop executes statements a predetermined number of times. The control expression for the loop is initialized, tested and manipulated entirely within the for loop parentheses.
4	**Nested Loop :** C language allows you to use one loop inside another loop. The following example illustrates the concept.
5	**Infinite loop:** It is the loop having no terminating condition, so the loop becomes infinite.

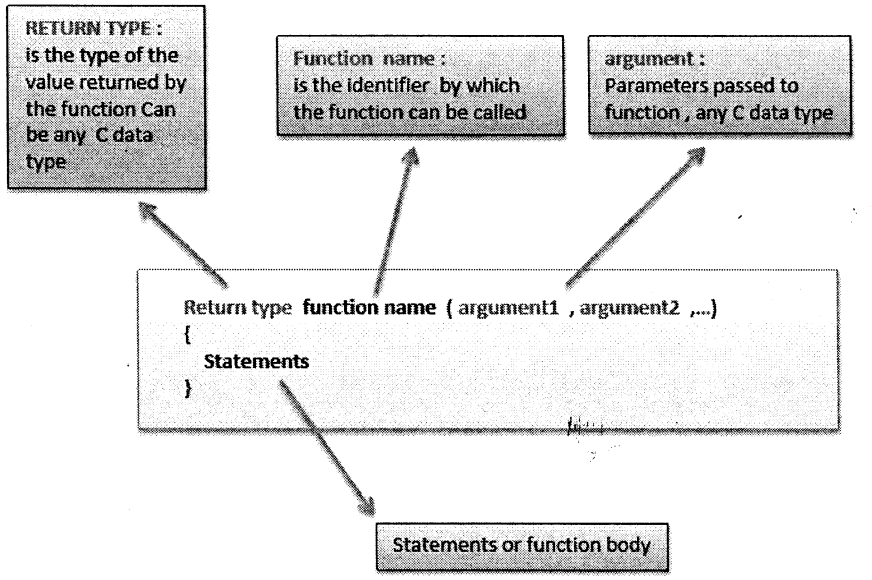
FUNCTIONS

Functions allow structuring the programs in segments of code to perform individual tasks. The typical case for creating a function is when one needs to perform the same action multiple times in a program.

Standardizing code fragments into functions has several advantages –

- Functions help the programmer stay organized. Often this helps to conceptualize the program.
- Functions codify one action in one place so that the function only has to be thought about and debugged once.
- This also reduces chances for errors in modification, if the code needs to be changed.
- Functions make the whole sketch smaller and more compact because sections of code are reused many times.
- They make it easier to reuse code in other programs by making it modular, and using functions often makes the code more readable.

There are two required functions in an Arduino sketch or a program i.e. setup () and loop(). Other functions must be created outside the brackets of these two functions. The most common syntax to define a function is –



A function is declared outside any other functions, above or below the loop function. We can declare the function in two different ways The first way is just writing the part of the function called a function prototype above the loop function, which consists of –

- Function return type
- Function name
- Function argument type, no need to write the argument name

The following example shows the demonstration of the function declaration using the first method.

Example

```
Int sum_func(int x,int y)// function declaration
{
int z =0;
   z =x+y;
return z;// return the value
}

void setup ()
{
Statements// group of statements
}

Void loop ()
{
int result =0;
result=Sum_func(5,6);// function call
}
```

The second method just declares the function above the loop function. The second way, which is called the function definition or declaration, must be declared below the loop function, which consists of –

- Function return type
- Function name
- Function argument type, here you must add the argument name
- The function body (statements inside the function which are being executed when the function is called)

The following example demonstrates the declaration of function using the second method.

Example

```
Int sum_func(int,int);// function prototype

void setup()
{
Statements// group of statements
}
Void loop(){
int result =0;
result=Sum_func(5,6);// function call
}
int sum_func(int x,int y)// function declaration {
int z =0;
   z =x+y;
return z;// return the value
}
```

Interrupts stop the current work of Arduino such that some other work can be done. Suppose you are sitting at home, chatting with someone. Suddenly the telephone rings. You stop chatting, and pick up the telephone to speak to the caller. When you have finished your telephonic conversation, you go back to chatting with the person before the telephone rang.

Similarly, you can think of the main routine as chatting to someone, the telephone ringing causes you to stop chatting. The interrupt service routine is the process of talking on the telephone. When the telephone conversation ends, you then go back to your main routine of chatting. This example explains exactly how an interrupt causes a processor to act.

The main program is running and performing some function in a circuit. However, when an interrupt occurs the main program halts while another routine is carried out. When this routine finishes, the processor goes back to the main routine again.

Important Points: Here are some important points about interrupts –

- Interrupts can come from various sources. In this case, we are using a hardware interrupt that is triggered by a state change on one of the digital pins.
- Most Arduino designs have two hardware interrupts (referred to as "interrupt0" and "interrupt1") hard-wired to digital I/O pins 2 and 3, respectively.
- The Arduino Mega has six hardware interrupts including the additional interrupts ("interrupt2" through "interrupt5") on pins 21, 20, 19, and 18.
- You can define a routine using a special function called as "Interrupt Service Routine" (usually known as ISR).
- You can define the routine and specify conditions at the rising edge, falling edge or both. At these specific conditions, the interrupt would be serviced.
- It is possible to have that function executed automatically, each time an event happens on an input pin.

Types of Interrupts : There are two types of interrupts

1. **Hardware Interrupts** – They occur in response to an external event, such as an external interrupt pin going high or low.
2. Software Interrupts – They occur in response to an instruction sent in software. The only type of interrupt that the "Arduino language" supports is the attachInterrupt() function.

INTEGRATION OF SENSORS AND ACTUATORS WITH ARDUINO.

HTTPS://WWW.INSTRUCTABLES.COM/ARDUINO-SENSORS-AND-ACTUATORS/

IMPLEMENTATION OF IOT: INTEROPERABILITY IN IOT,

On hearing the term Internet of Things (IoT), the first picture that comes to mind is "connected things." At the most basic level, IoT is the connectivity between people, processes, and things. IoT is transforming the way we interact with our surroundings. Smart homes, smart meters, smart cities, smart wearables are impacting our lives all due to the rapid growth of the IoT market.

Internet of Things (IoT) is an ever-growing network of physical devices embedded with sensors, actuators,and wire-less connectivity to communicate and share their information among themselves. The application of IoT is in diverse areas such as agriculture, poultry and farming, smart city, and health care, where a sensor node must support heterogeneous sensors/actuators, and varying types of wireless connectivity. Interoperability is the ability of two or more devices, systems, platforms or networks to work in conjucntion. Interoperability enables communication between heterogeneous devices or system in order to achieve a common goal. However, the current devices and systems are fragmented with respect to the communication technologies, protocols, and data formats. This diversity makes it difficult for devices and systems in the IoT network to communicate and share their data with one another. The utility of IoT network is limited by the lack of interoperability.

While IoT market leaders have aggressive IoT business strategies to create seamless, contextual experiences, there are multiple challenges the industry is coping with including security, connectivity, standards, compatibility, longevity, and intelligent analysis. In fact, IoT interoperability is one of the central challenges. Today, IoT ecosystems lack basic interoperability to connect devices and sensors seamlessly. Interoperability has the potential to unlock more than 40% per year in potential economic impact from IoT use by 2025.

IoT interoperability has complicated IoT adoption, but organizations must navigate the selection of competing standards, such as IP or MQTT, to get the most from IoT deployments. IoT deployments have three interoperability needs:

- **Technical interoperability.** The deployment has the ability to use a physical communications infrastructure to transport bits of data.
- **Syntactic interoperability.** A shared syntax or common information model structures the data and establishes a protocol to share the information as specific typed data.
- **Semantic** IoT deployments require the ability to establish the meaning of the data.

The lack of interoperability prevents devices from connecting autonomously, discovering each other, and collaboratively engaging with other smart devices and services, which is a barrier for companies to build automated ecosystems. The interoperability challenge can manifest itself in a number of ways:

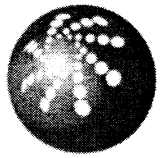
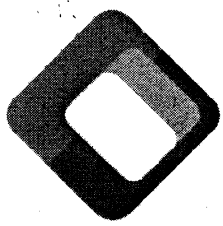
- Devices made by different manufacturers cannot integrate.
- Limited connectivity between different transport protocols such as Ethernet, WiFi, and Zigbee
- No set rules or standards at the application level, causing an inability to combine and complement the collected data from different sensors and devices.

Let's imagine a smart home scenario that has full interoperability. The home is networked to enable multiple interconnected devices, services, and apps—from entertainment to healthcare, security, and home automation—to provide contextual experiences for the household. Here, the products from the likes of Google, Amazon, Samsung, Nest, Phillips, IKEA, and Yale are integrated seamlessly so individuals can control and monitor the home remotely and from within. This includes:

- The alarm clock on the user's mobile device automatically turns on nest thermostat and switches on Philips hue.
- Using voice command, Amazon Alexa Echo turns off the alarm and triggers the Coffee smart-coffee machine to brew the coffee.
- Wearable bands, devices and ingestible sensors monitor heart rate, blood pressure, and other health parameters and automatically sends the data to the doctor for analysis in case of anomalies.
- The Samsung smart refrigerator checks the availability of daily essentials such as milk and bread and orders them automatically.
- Connected in-home robots help with everything from cooking, cleaning, and other daily chores.
- Honeywell sensors and security cameras detect unusual movement or sound at home and notify the homeowner through mobile or initiate emergency services.
- On the way home, the connected car remotely turns on the Nest thermostat and can control other home devices.
- As the owner reaches home, the Yale Assure smart lock unlocks, the IKEA smart lights brighten up the house, and Google home starts playing the preferred evening playlist automatically.

In short, the opportunities for connectivity are almost endless, limited only by the imagination. There are two key areas companies are focusing on to improve interoperability:

1. **IoT standardization :** Many technology companies and OEMs are developing interoperability solutions by adopting standards and open-source development. Here are some of the groups working to improve IoT standardization:

Standard	Direction
 AllSeen Alliance	Responsible for developing the AllJoyn framework, which is an open-source framework. AllJoyn is transported, OS and platform-agnostic that makes it easy for devices and apps to discover and securely communicate with each other.
 Open Connectivity Foundation (OCF)	OCF was formed in early 2016 through the merger of the AllSeen Alliance and Open Interconnect Consortium (OIC). OCF has created specifications for a service-layer platform that allows device discovery and connectivity across devices from different vendors and across multiple operating systems. The reference implementation of the OCF specification is called IoTivity.
 oneM2M	Launched in 2012, the goal of oneM2M is to develop technical specifications that address the need for a common service layer that can be embedded within hardware and software.
 Thread Group	The Thread Group, an alliance founded in July 2014, is the promoter of Thread. Thread is a low-power, low-data-rate embedded networking stack, built on 6LowPAN (low power wireless personal area networks) that allows even small devices to be connected to the internet. Thread competes with ZigBee and Z Wave. Thread and the ZigBee Alliance are collaborating to enable the ZigBee Cluster Library (branded "dot dot") to run over Thread networks.

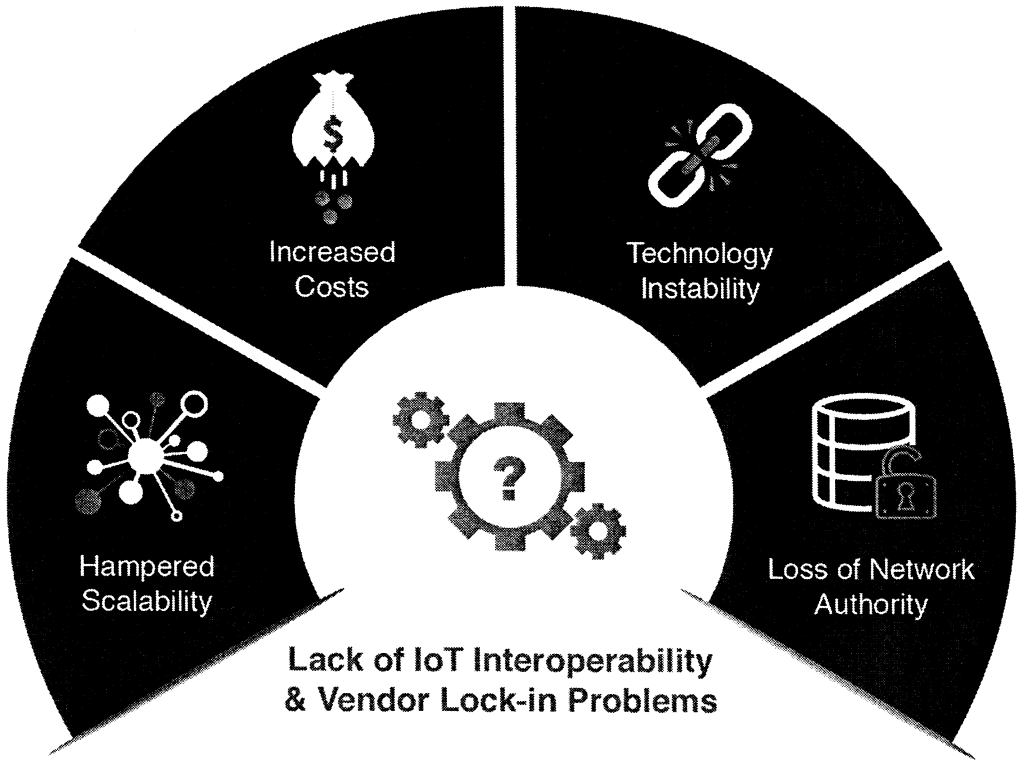
Wi Fi ALLIANCE Bluetooth 5	Wi-Fi HaLow and Bluetooth 5 are the IoT wireless standard leaders to capture the IoT market by providing enhanced speeds, wide-range, and low-power connectivity.

2. **Converged smart ecosystem :** Consumers' expectations for IoT are high. They believe IoT solutions should be delivered and managed seamlessly, with products and services working together behind the scenes. To meet these expectations, global brands are working together to establish innovative partnerships to deliver connectivity of devices and a seamless user experience.

   Global brands competing in the smart home market, including Apple, Google, Amazon, and Samsung, are moving towards a "certified with" approach, which attracts other providers to certify their products so they can participate in these ecosystems. This way, global leaders build out smart ecosystems of integrated products from different providers and standards. However, certification should also mean that the product integrates and works seamlessly with other certified products within the same ecosystem.

   IoT is an ecosystem game. No single technology in the market can deliver a complete, end-to-end IoT solution on its own. From connectivity, sensors and gateways to the cloud and application systems, an IoT architecture is composed of various components working in concert with each other. While ensuring a seamless data flow along the IoT value chain is critical, it is only half of the battle.

   Today's exploding number of IoT vendors has turned the IoT ecosystem into a highly complex landscape. To address multiple applications and challenges, an IoT infrastructure often needs to incorporate cross-domain hardware and application systems. Likewise, it must be flexible enough to effectively integrate future devices that may come with different hardware models. Beyond vertical integration within a specific industry or application, the diverse nature of the digital ecosystem means that horizontal interoperability between different devices and systems will also be critical to the success of a scalable IoT network.



Lack of IoT Interoperability

Despite its utmost importance, IoT interoperability for many vendors is still a goal to work towards. A large number of existing IoT solutions are proprietary and designed to operate only within a pre-defined hardware or infrastructure environment. Examples include protocols tied to vendor-specific chipsets or wireless connectivity bound to a single third-party managed backend. The lack of IoT interoperability means that data can't be effectively exchanged across disparate, sometimes overlapping devices and systems.

From the IoT adopters' perspective, these closed ecosystems, or better named as silos, pose multiple problems. They hamper effective integration of new IoT devices and solutions that can tackle a wider range of operational issues. Supporting heterogeneous IoT infrastructures for different applications can quickly inflate costs and complexity beyond what companies can handle.

Vendor lock-in also deprives users of control over their data, network uptime and infrastructure management, while preventing them from switching to more cost-effective hardware options in the future. Technical instability is another potential issue, given the inherent risk that the vendor fails to deliver the agreed services and product functionality. This results in impaired Quality-of-Service and network scalability or even security holes.

Designing an IoT Architecture for Interoperability

The best way to circumvent these challenges is to prepare your IoT networks for interoperability from the start. Despite today's highly fragmented IoT landscape, here are three rules of thumb for IoT connectivity that will help navigate your network design.

1. **Open, Industry Standards :** Solutions incorporating proven standards are built upon an open, universal framework recognized by Standard Development Organizations (SDO). Besides assured Quality-of-Service, open standards foster global transparency and consistency, eliminating incompatible variations in technical design and product development. This fuels worldwide adoption, cross-vendor support and interoperability in the long run. Adopting standard-based protocols, specifically, allows you to benefit from a growing portfolio of compatible off-the-shelf hardware across verticals. You can also avoid the risk of backward incompatibility due to any strategic changes by the proprietary vendor.

2. **Software-driven Technologies :** In industrial environments, IoT devices often abide by a rigorous set of safety and reliability regulations. Deploying wireless solutions with a hardware-driven approach is challenging in this regard, as you are bound to a certain device type and must depend on the respective vendor(s) to go through the certification process. Software-driven technologies, on the other hand, can be flexibly plugged in any legacy devices and infrastructure that already meet your operational requirements – whether sensors or industry PCs.

3. **Open Interfaces:** IoT interoperability on the application layer entails effective data transfer to different user's application systems and servers. Open sourced messaging protocols like MQTT or CoAP and Application Programming Interfaces (APIs) based on RESTful principles are key drivers of cross-application interoperability. In a private network architecture, having these open interfaces natively embedded in the IoT gateway enables direct data transfer to your preferred backend for analytics and visualization, without relying on a third-party managed server.

To wrap it up, interoperability is key to robust and scalable IoT network, and requires particular attention in your architecture design. Leveraging a standard-based, software-driven communication platform with built-in open interfaces allows for easy deployment in legacy environments while ensuring long-term interoperability with cross-vertical hardware and systems.

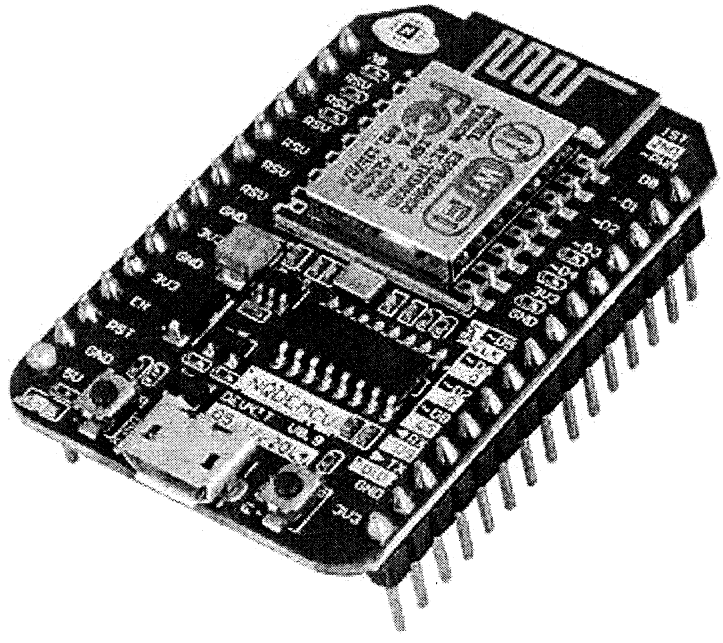
INTRODUCTION TO NODEMCU (ESP8266), CONNECTIVITY OF SENSORS AND ACTUATORS WITH NODEMCU,

Today, IOT applications are on the rise, and connecting objects are getting more and more important. There are several ways to connect objects such as Wi-Fi protocol.

NodeMCU is an open source platform based on ESP8266 which can connect objects and let data transfer using the Wi-Fi protocol. In addition, by providing some of the most important features of microcontrollers such as GPIO, PWM, ADC, and etc, it can solve many of the project's needs alone. The general features of this board are as follows:

- Easy to use
- Programmability with Arduino IDE or IUA languages
- Available as an access point or station
- practicable in Event-driven API applications
- Having an internal antenna
- Containing 13 GPIO pins, 10 PWM channels, I2C, SPI, ADC, UART,

NodeMCU is an open-source LUA based firmware developed for the ESP8266 wifi chip. By exploring functionality with the ESP8266 chip, NodeMCU firmware comes with the ESP8266 Development board/kit i.e. NodeMCU Development board. NodeMCU is an open-source platform, its hardware design is open for edit/modify/build. The **ESP8266** is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol.



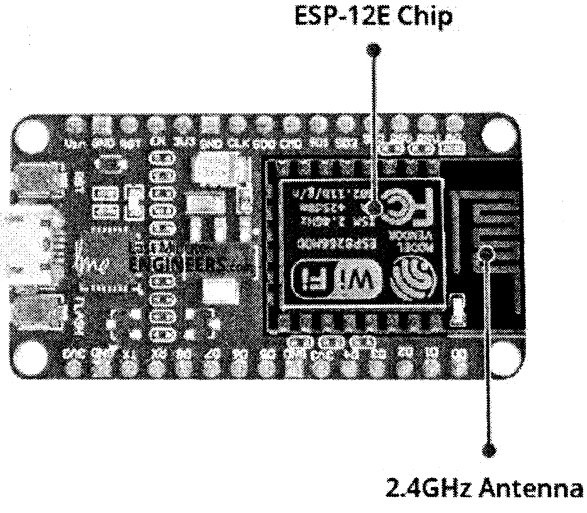
NodeMCU Development Board/kit v0.9 (Version1)

NodeMCU Development board is featured with wifi capability, analog pin, digital pins, and serial communication protocols. To get started with using NodeMCU for IoT applications first we need to know about how to write/download NodeMCU firmware in NodeMCU Development Boards. And before that where this NodeMCU firmware will get as per our requirement.

ESP8266 NodeMCU Features & Using It With Arduino IDE

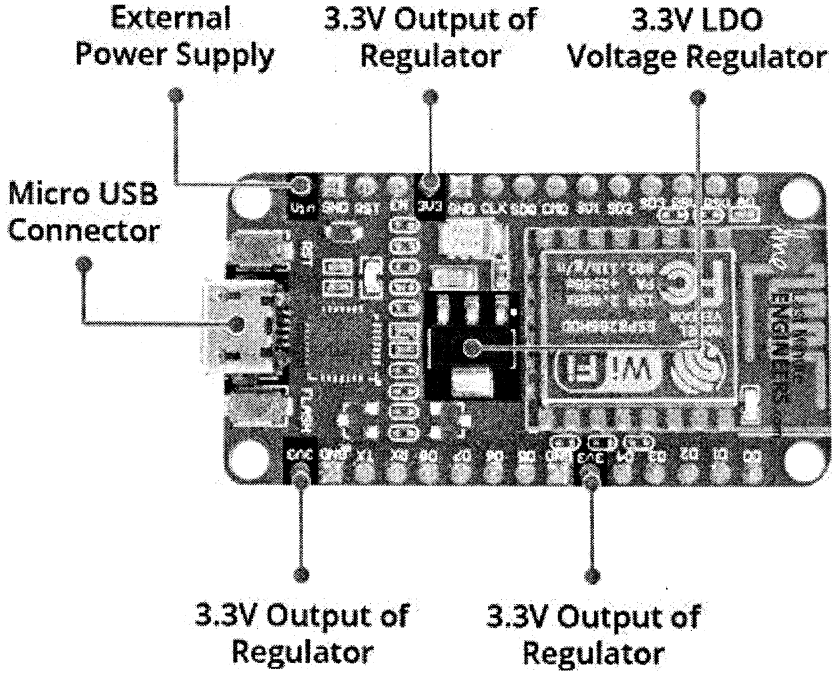
The Internet of Things (IoT) has been a trending field in the world of technology. It has changed the way we work. Physical objects and the digital world are connected now more than ever. Keeping this in mind, Espressif Systems (A Shanghai-based Semiconductor Company) has released an adorable, bite-sized WiFi enabled microcontroller - ESP8266, it can monitor and control things from anywhere in the world - perfect for just about any IoT project.

1. **ESP-12E Module :** The development board equips the ESP-12E module containing ESP8266 chip having Tensilica Xtensa® 32-bit LX106 RISC microprocessor which operates at 80 to 160 MHz adjustable clock frequency and supports RTOS.
    - Tensilica Xtensa® 32-bit LX106
    - 80 to 160 MHz Clock Freq.
    - 128kB internal RAM
    - 4MB external flash
    - 802.11b/g/n Wi-Fi transceiver



There's also 128 KB RAM and 4MB of Flash memory (for program and data storage) just enough to cope with the large strings that make up web pages, JSON/XML data, and everything we throw at IoT devices nowadays. The ESP8266 Integrates 802.11b/g/n HT40 Wi-Fi transceiver, so it can not only connect to a WiFi network and interact with the Internet, but it can also set up a network of its own, allowing other devices to connect directly to it. This makes the ESP8266 NodeMCU even more versatile.

2. **Power Requirement :** As the operating voltage range of ESP8266 is 3V to 3.6V, the board comes with a LDO voltage regulator to keep the voltage steady at 3.3V. It can reliably supply up to 600mA, which should be more than enough when ESP8266 pulls as much as 80mA during RF transmissions. The output of the regulator is also broken out to one of the sides of the board and labeled as 3V3. This pin can be used to supply power to external components.
    - Operating Voltage: 2.5V to 3.6V
    - On-board 3.3V 600mA regulator
    - 80mA Operating Current
    - 20 µA during Sleep Mode



Power to the ESP8266 NodeMCU is supplied via the on-board MicroB USB connector. Alternatively, if you have a regulated 5V voltage source, the VIN pin can be used to directly supply the ESP8266 and its peripherals.

The ESP8266 requires a 3.3V power supply and 3.3V logic levels for communication. The GPIO pins are not 5V-tolerant! If you want to interface the board with 5V (or higher) components, you'll need to do some level shifting.

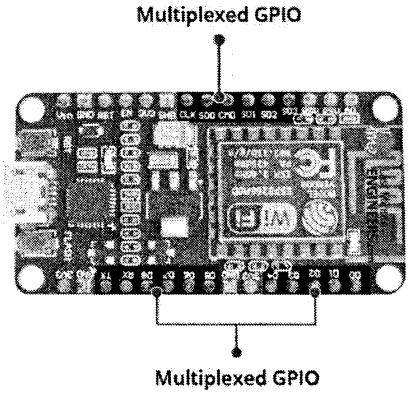
3. **Peripherals and I/O :** The ESP8266 NodeMCU has total 17 GPIO pins broken out to the pin headers on both sides of the development board. These pins can be assigned to all sorts of peripheral duties, including:
    - ADC channel – A 10-bit ADC channel.
    - UART interface – UART interface is used to load code serially.
    - PWM outputs – PWM pins for dimming LEDs or controlling motors.

- SPI, I2C & I2S interface – SPI and I2C interface to hook up all sorts of sensors and peripherals.
- I2S interface – I2S interface if you want to add sound to your project.

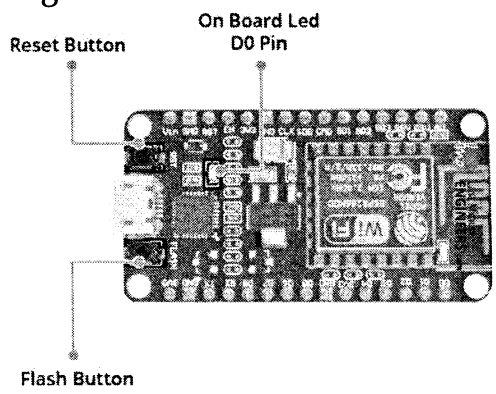
Multiplexed I/Os

- 1 ADC channels
- 2 UART interfaces
- 4 PWM outputs
- SPI, I2C & I2S interface

Thanks to the ESP8266's pin multiplexing feature (Multiple peripherals multiplexed on a single GPIO pin). Meaning a single GPIO pin can act as PWM/UART/SPI.

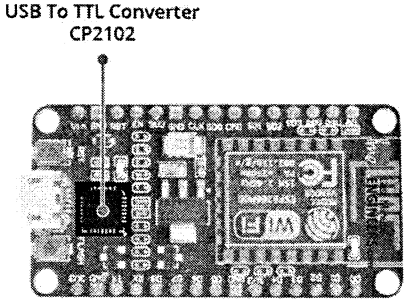


4. **On-board Switches & LED Indicator :** The ESP8266 NodeMCU features two buttons. One marked as RST located on the top left corner is the Reset button, used of course to reset the ESP8266 chip. The other FLASH button on the bottom left corner is the download button used while upgrading firmware.
   - RST – Reset the ESP8266 chip
   - FLASH – Download new programs
   - Blue LED – User Programmable



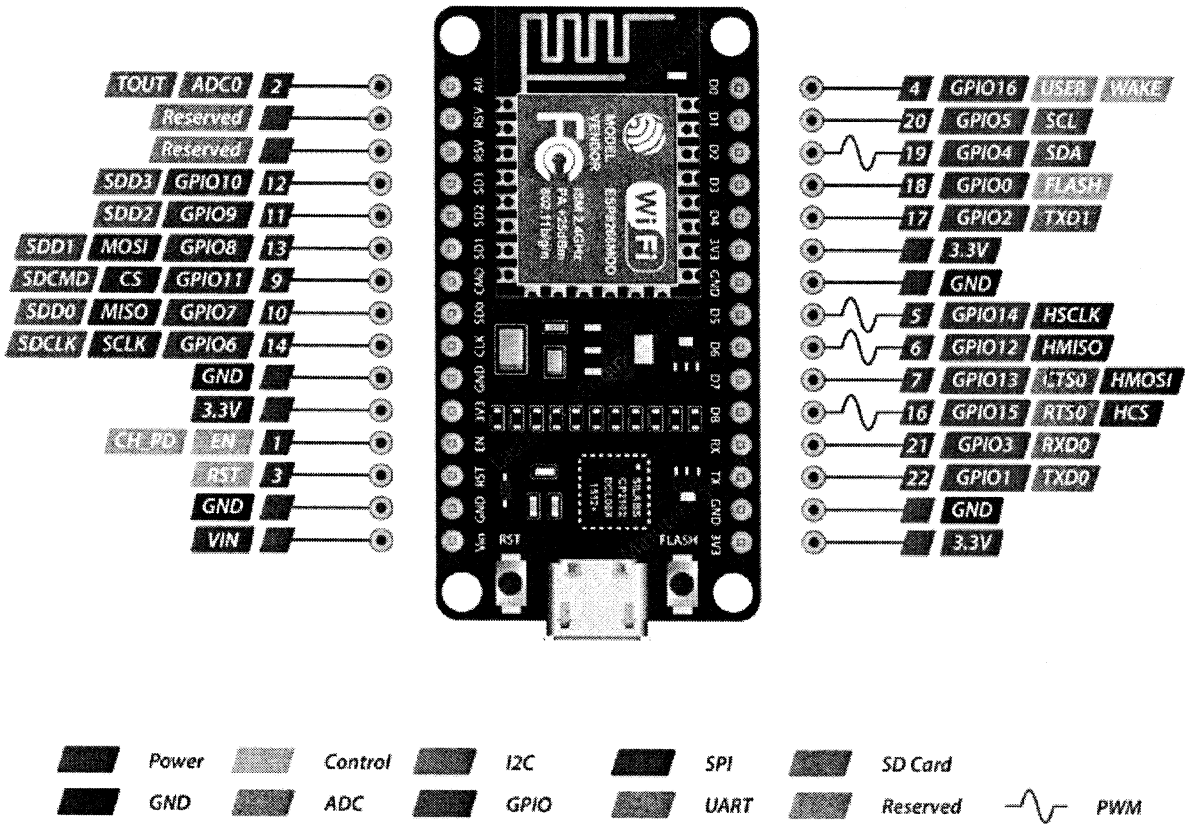
The board also has a LED indicator which is user programmable and is connected to the D0 pin of the board.

5. **Serial Communication** : The board includes CP2102 USB-to-UART Bridge Controller from Silicon Labs, which converts USB signal to serial and allows your computer to program and communicate with the ESP8266 chip.
    - CP2102 USB-to-UART converter
    - 4.5 Mbps communication speed
    - Flow Control support



ESP8266 NodeMCU Pinout

The ESP8266 NodeMCU has total 30 pins that interface it to the outside world. The connections are as follows:



1. Power Pins There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used to directly supply the ESP8266 and its peripherals, if you have a regulated 5V voltage source. The 3.3V pins are the output of an on-board voltage regulator. These pins can be used to supply power to external components.

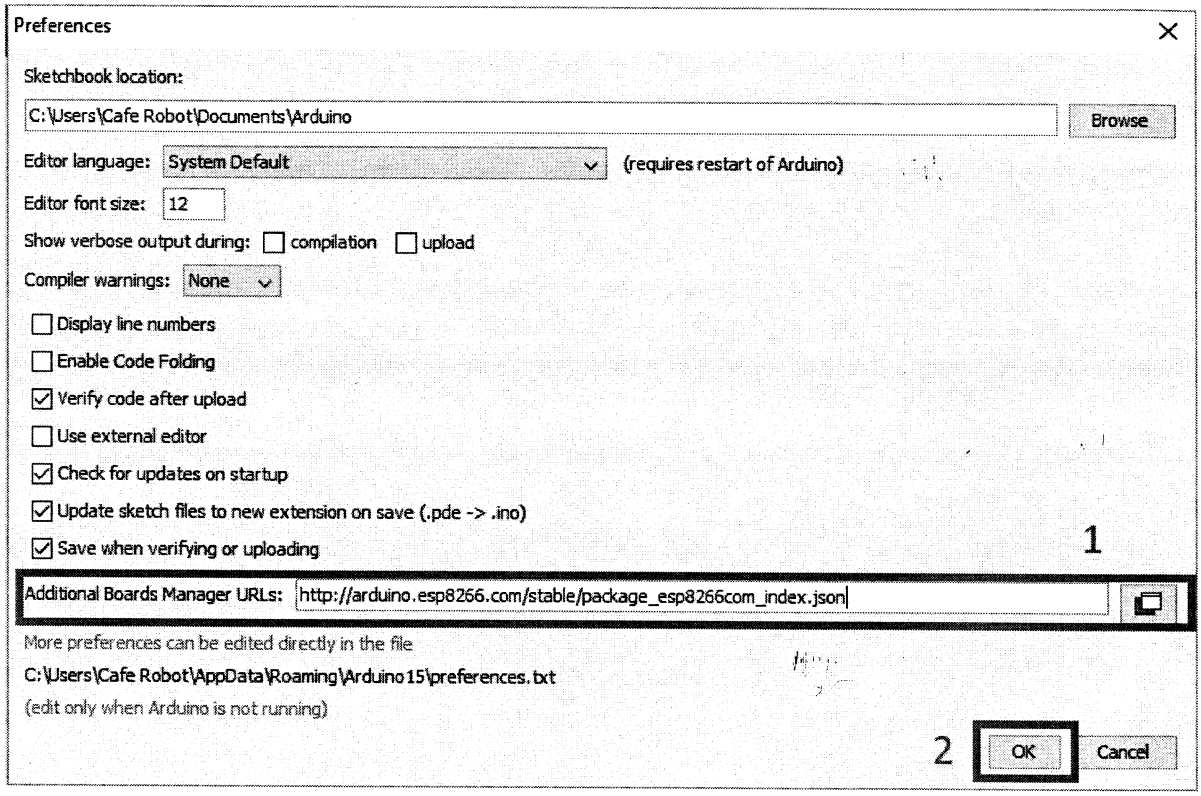
2. GND is a ground pin of ESP8266 NodeMCU development board.
3. I2C Pins are used to hook up all sorts of I2C sensors and peripherals in your project. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.
4. GPIO Pins ESP8266 NodeMCU has 17 GPIO pins which can be assigned to various functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.
5. ADC Channel The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC viz. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.
6. UART Pins ESP8266 NodeMCU has 2 UART interfaces, i.e. UART0 and UART1, which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. It supports fluid control. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.
7. SPI Pins ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:
   - 4 timing modes of the SPI format transfer
   - Up to 80 MHz and the divided clocks of 80 MHz
   - Up to 64-Byte FIFO
8. SDIO Pins ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.
9. PWM Pins The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 µs to 10000 µs, i.e., between 100 Hz and 1 kHz.
10. Control Pins are used to control ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
    - EN pin - The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
    - RST pin - RST pin is used to reset the ESP8266 chip.
    - WAKE pin - Wake pin is used to wake the chip from deep-sleep.

How to program NodeMCU using Arduino IDE

In order to use Arduino IDE to program the NodeMCU, you have to introduce it to the software at first.

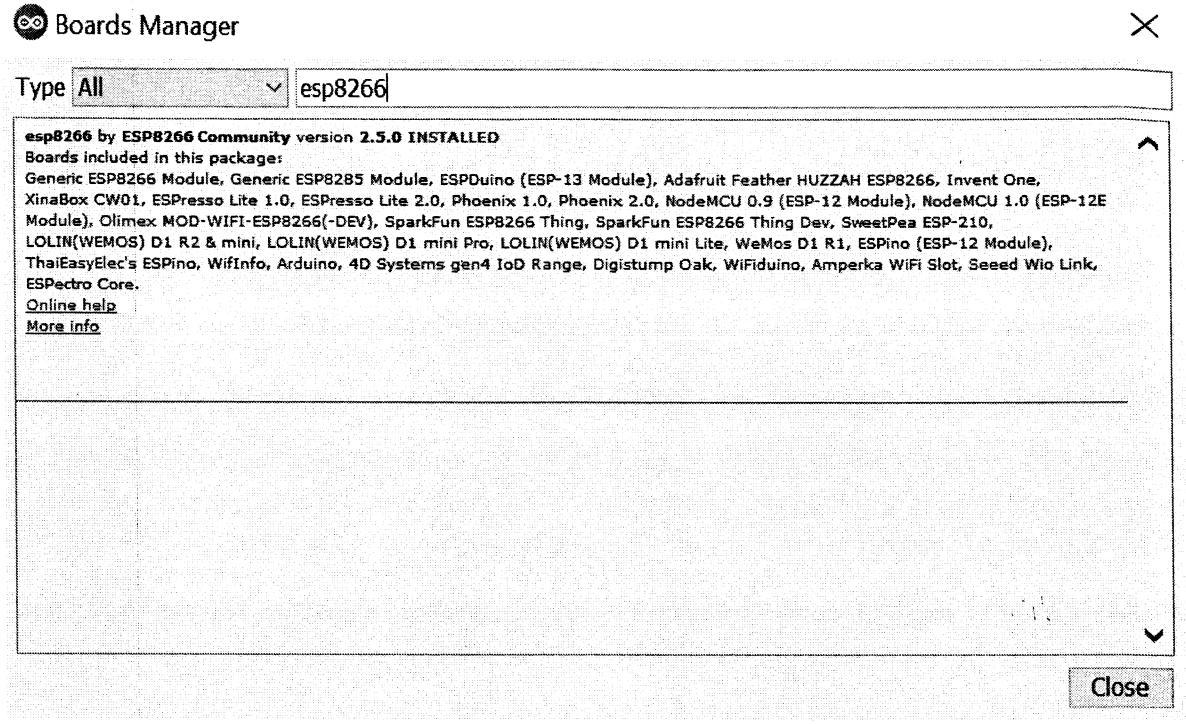
http://arduino.esp8266.com/stable/package_esp8266com_index.json

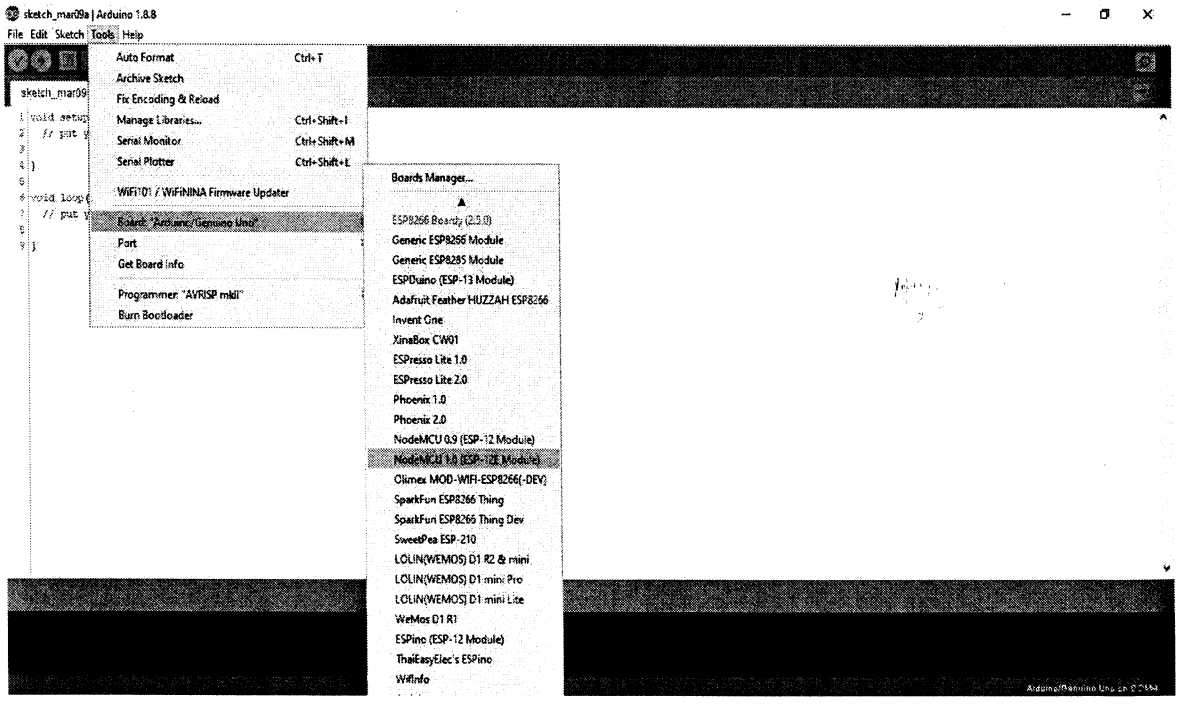
step1. Choose Preferences in the File menu and enter the above link in Additional Board Manager URLs part. Then press OK.



Step2. Search the word ESP8266 in Boards > boards manager from Tools menu. Then install ESP8266 boards. After complete installation, you will see the INSTALLED label on ESP8266 boards.

After these two steps, you can see ESP8266 based boards such as NodeMCU in your Arduino IDE boards list, and you can choose your desired board to upload the code.



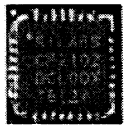


USB to Serial Converter – CP2102 or CH340G

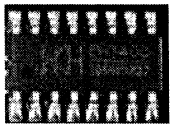
Incorporated into each NodeMCU is a USB to Serial Converter. The official design is based on the CP2102 chipset and offers the best compatibility. Genuine boards use the CP2102 chipset including the officially licensed Amica NodeMCU modules. The other common USB to Serial Converter used is the CH340G which is common

on the lower-priced modules including the LoLin units. Other designs may use drivers including the FTDI chipset, but those designs are rare.

Depending on the Operating System you are using with the NodeMCU, the appropriate driver must be installed. Generally, Windows 10 immediately recognizes the CP2102 chipset while the CH340G may require separate installation.



Drivers for the CP2102 are available for **download** from the Silicon Labs support site. Drivers constantly evolve and ensuring and installing the most recent version in your development environment minimum issues. Drivers are available for Windows, Mac, Linux, and Android. You are always best to visit the original manufacturer to ensure you are receiving the most recent versions of the driver.



WCH maintain and update the drivers for the CH340G on a regular basis. Versions of the driver are also available for Windows, Mac, Linux, and Android. Visit their Driver Download page. We have experienced situations where both CP2102 and CH340G devices have not functioned or been recognized as expected. The solution was as simple as uninstalling the old driver and installing the most recent version.

More attention needs to be given to selecting board, choosing COM port and selecting Upload speed. You may get espcomm_upload_mem error while uploading new sketches, if failed to do so. Once you are done, try the example sketch below.

```
void setup()
        {
         pinMode(D0, OUTPUT);
        }
void loop()
        {
         digitalWrite(D0, HIGH);
         delay(500);
         digitalWrite(D0, LOW);
         delay(500);
        }
```

Once the code is uploaded, LED will start blinking. You may need to tap the RST button to get your ESP8266 to begin running the sketch.

Controlling LED through an HTTP page Using NodeMCU

We can connect the internet through Wi-Fi using NodeMCU, and apply our desired commands by creating an HTTP page. In this example, we can control an LED by pressing the ON and OFF key. Enter your modems SSID and password in the provided part and upload it on your NodeMCU board using Arduino IDE. (Leave other settings to the default)

Code

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

/* Set these to your desired credentials. */
const char *ssid = "*****"; //Enter your WIFI ssid
const char *password = "*****"; //Enter your WIFI password

ESP8266WebServer server(80);

void handleRoot() {
  server.send(200, "text/html", "<form action=\"/LED_BUILTIN_on\"
method=\"get\" id=\"form1\"></form><button type=\"submit\"
form=\"form1\" value=\"On\">On</button><form
action=\"/LED_BUILTIN_off\" method=\"get\"
id=\"form2\"></form><button type=\"submit\" form=\"form2\"
value=\"Off\">Off</button>");
}
void handleSave() {
  if (server.arg("pass") != "") {
    Serial.println(server.arg("pass"));
  }
}
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  delay(3000);
  Serial.begin(115200);
  Serial.println();
  Serial.print("Configuring access point...");
```

```
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.on ( "/", handleRoot );
  server.on ("/save", handleSave);
  server.begin();
  Serial.println ( "HTTP server started" );
  server.on("/LED_BUILTIN_on", []() {
    digitalWrite(LED_BUILTIN, 1);
    Serial.println("on");
    handleRoot();
  });
  server.on("/LED_BUILTIN_off", []() {
    digitalWrite(LED_BUILTIN, 0);
    Serial.println("off");
    handleRoot();
  });
}
void loop() {
  server.handleClient();
}
```

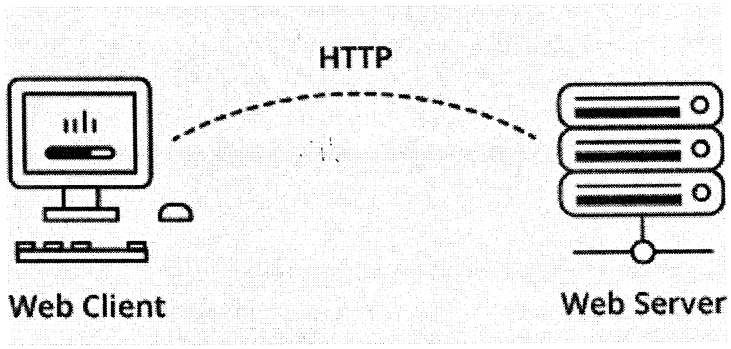
After opening the Serial Monitor, if the Internet connection is established, you will be given the IP address of the page you have created (for example 192.168.1.18). copy and paste it in your browser to open the HTTP page.

```
<form action="LED_BUILTIN_on" method="get" id="form1">      </form>
<button type=submit" form="form1" value="On">On</button>
<form action="LED_BUILTIN_off" method="get" id="form2">     </form>
<button type="submit" form="form2" value="Off">Off</button>
```

NodeMCU Web Server

Over the past few years, the ESP8266 has been a growing star among IoT or WiFi-related projects. It's an extremely cost-effective WiFi module that - with a little extra effort - can be programmed to build a standalone web server.

Web server: Web server is a place which stores, processes and delivers web pages to Web clients. Web client is nothing but a web browser on our laptops and smartphones. The communication between client and server takes place using a special protocol called Hypertext Transfer Protocol (HTTP).



In this protocol, a client initiates communication by making a request for a specific web page using HTTP and the server responds with the content of that web page or an error message if unable to do so (like famous 404 Error). Pages delivered by a server are mostly HTML documents.

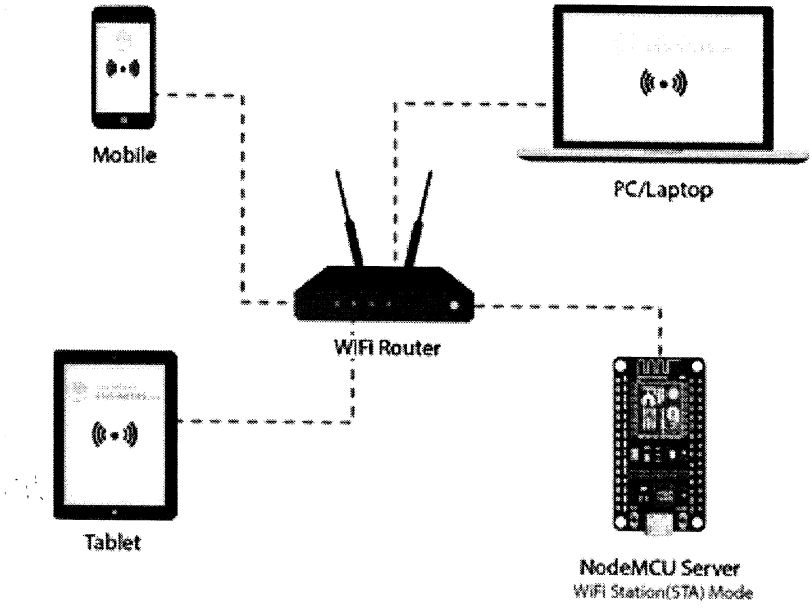
ESP8266 Operating Modes

One of the greatest features ESP8266 provides is that it cannot only connect to an existing WiFi network and act as a Web Server, but it can also set up a network of its own, allowing other devices to connect directly to it and access web pages. This is possible because ESP8266 can operate in three different modes: Station mode, Soft Access Point mode, and both at the same time. This provides possibility of building mesh networks.

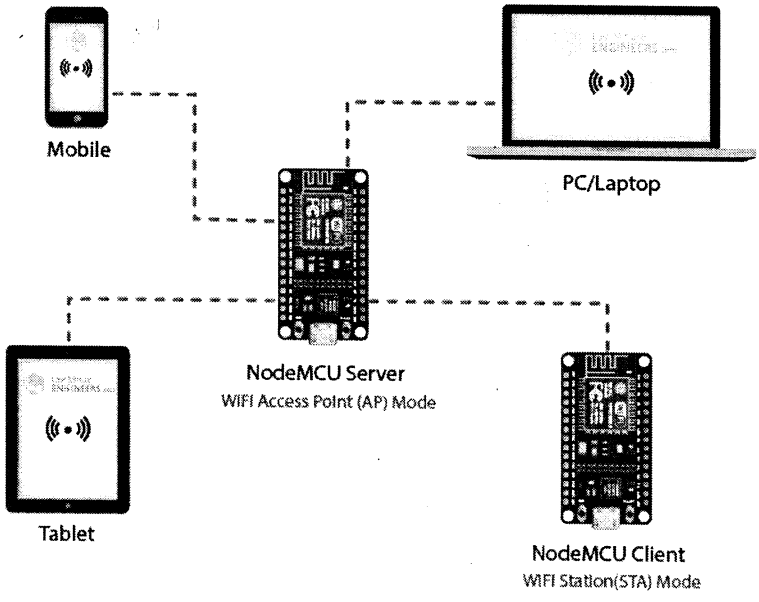
Station (STA) Mode: The ESP8266 that connects to an existing WiFi network (one created by your wireless router) is called Station (STA). In STA mode ESP8266 gets IP from wireless router to which it is connected. With this IP address, it can set up a web server and deliver web pages to all connected devices under existing WiFi network.

Soft Access Point (AP) Mode: The ESP8266 that creates its own WiFi network and acts as a hub (Just like WiFi router) for one or more stations is called Access Point (AP). Unlike WiFi router, it does not have interface to a wired network. So,

such mode of operation is called Soft Access Point (soft-AP). Also the maximum number of stations that can connect to it is limited to five.



Station (STA) Mode



Soft Access Point (AP) Mode

In AP mode ESP8266 creates a new WiFi network and sets SSID (Name of the network) and IP address to it. With this IP address, it can deliver web pages to all connected devices under its own network.

Connecting LEDs to ESP8266 NodeMCU

Now that we know the basics of how web server works, and in which modes ESP8266 can create a web server, it's time to connect some LEDs to ESP8266 NodeMCU that we want to control over WiFi.

Start by placing the NodeMCU on to your breadboard, ensuring each side of the board is on a separate side of the breadboard. Next, connect two LEDs to digital GPIO D6 and D7 through a 220Ω current limiting resistor.

Controlling Things From ESP8266 Web Server

When you type a URL in a web browser and hit ENTER, the browser sends a HTTP request (a.k.a. GET request) to a web server. It's a job of web server to handle this request by doing something. You might have figured it out by now that we are going to control things by accessing a specific URL. For example, suppose we entered a URL like http://192.168.1.1/ledon in a browser. The browser then sends a HTTP request to ESP8266 to handle this request. When ESP8266 reads this request, it knows that user wants to turn the LED ON. So, it turns the LED ON and sends a dynamic webpage to a browser showing LED status : ON As easy as Pie!

ESP8266 as HTTP Server using WiFi Access Point (AP) mode

As the heading suggests, this example demonstrates how to turn the ESP8266 into an access point (AP), and serve up web pages to any connected client. To start with, plug your ESP8266 NodeMCU into your computer and Try the sketch out; and then we will dissect it in some detail.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

/* Put your SSID & Password */
const char* ssid = "NodeMCU"; // Enter SSID here
const char* password = "12345678"; //Enter Password here

/* Put IP Address details */
IPAddress local_ip(192,168,1,1);
IPAddress gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);

ESP8266WebServer server(80);

uint8_t LED1pin = D7;
bool LED1status = LOW;

uint8_t LED2pin = D6;
```

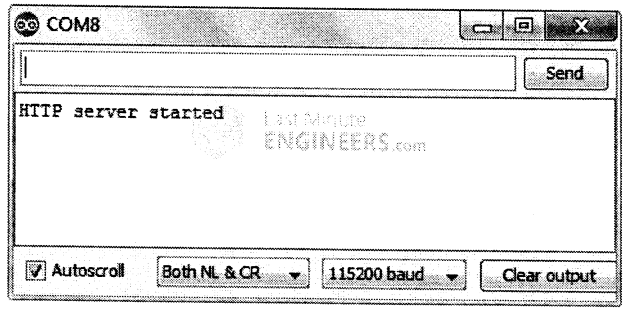
```
bool LED2status = LOW;

void setup() {
 Serial.begin(115200);
 pinMode(LED1pin, OUTPUT);
 pinMode(LED2pin, OUTPUT);

 WiFi.softAP(ssid, password);
 WiFi.softAPConfig(local_ip, gateway, subnet);
 delay(100);

 server.on("/", handle_OnConnect);
 server.on("/led1on", handle_led1on);
 server.on("/led1off", handle_led1off);
 server.on("/led2on", handle_led2on);
 server.on("/led2off", handle_led2off);
 server.onNotFound(handle_NotFound);
  server.begin();
 Serial.println("HTTP server started");
}

void loop()
{
 server.handleClient();
 if(LED1status)
 {digitalWrite(LED1pin, HIGH);}
 else
 {digitalWrite(LED1pin, LOW);}

 if(LED2status)
 {digitalWrite(LED2pin, HIGH);}
 else
 {digitalWrite(LED2pin, LOW);}
}

void handle_OnConnect() {
 LED1status = LOW;
 LED2status = LOW;
 Serial.println("GPIO7 Status: OFF | GPIO6 Status: OFF");
 server.send(200, "text/html", SendHTML(LED1status,LED2status));
}

void handle_led1on() {
 LED1status = HIGH;
 Serial.println("GPIO7 Status: ON");
 server.send(200, "text/html", SendHTML(true,LED2status));
```

```
}

void handle_led1off() {
 LED1status = LOW;
 Serial.println("GPIO7 Status: OFF");
 server.send(200, "text/html", SendHTML(false,LED2status));
}

void handle_led2on() {
 LED2status = HIGH;
 Serial.println("GPIO6 Status: ON");
 server.send(200, "text/html", SendHTML(LED1status,true));
}

void handle_led2off() {
 LED2status = LOW;
 Serial.println("GPIO6 Status: OFF");
 server.send(200, "text/html", SendHTML(LED1status,false));
}

void handle_NotFound(){
 server.send(404, "text/plain", "Not found");
}

String SendHTML(uint8_t led1stat,uint8_t led2stat){
 String ptr = "<!DOCTYPE html> <html>\n";
 ptr +="<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1.0, user-scalable=no\">\n";
 ptr +="<title>LED Control</title>\n";
 ptr +="<style>html { font-family: Helvetica; display: inline-block; margin:
0px auto; text-align: center;}\n";
 ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;}
h3 {color: #444444;margin-bottom: 50px;}\n";
 ptr +=".button {display: block;width: 80px;background-color:
#1abc9c;border: none;color: white;padding: 13px 30px;text-decoration:
none;font-size: 25px;margin: 0px auto 35px;cursor: pointer;border-radius:
4px;}\n";
 ptr +=".button-on {background-color: #1abc9c;}\n";
 ptr +=".button-on:active {background-color: #16a085;}\n";
 ptr +=".button-off {background-color: #34495e;}\n";
 ptr +=".button-off:active {background-color: #2c3e50;}\n";
 ptr +="p {font-size: 14px;color: #888;margin-bottom: 10px;}\n";
 ptr +="</style>\n";
 ptr +="</head>\n";
 ptr +="<body>\n";
 ptr +="<h1>ESP8266 Web Server</h1>\n";
```

```
 ptr +="<h3>Using Access Point(AP) Mode</h3>\n";

  if(led1stat)
 {ptr +="<p>LED1 Status: ON</p><a class=\"button button-off\"
href=\"/led1off\">OFF</a>\n";}
 else
 {ptr +="<p>LED1 Status: OFF</p><a class=\"button button-on\"
href=\"/led1on\">ON</a>\n";}

 if(led2stat)
 {ptr +="<p>LED2 Status: ON</p><a class=\"button button-off\"
href=\"/led2off\">OFF</a>\n";}
 else
 {ptr +="<p>LED2 Status: OFF</p><a class=\"button button-on\"
href=\"/led2on\">ON</a>\n";}

 ptr +="</body>\n";
 ptr +="</html>\n";
 return ptr;
}
```

Accessing the Web Server in AP mode

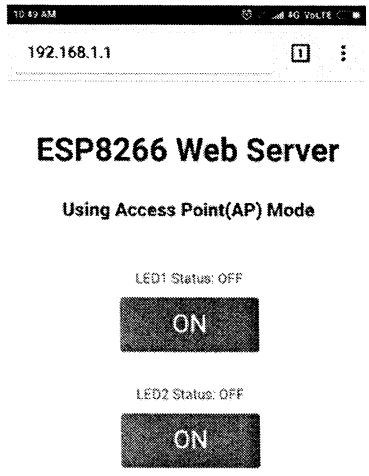
After uploading the sketch, open the Serial Monitor at a baud rate of 115200. And press the RESET button on ESP8266. If everything is OK, it will show HTTP server started message. Next, find any device that you can connect to a WiFi network phone, laptop, etc. And look for a network called NodeMCU. Join the network with password 123456789.



After connecting to your NodeMCU AP network, load up a browser and point it to 192.168.1.1 The NodeMCU should serve up a web page showing current status of LEDs and two buttons to control them. If take a look at the serial monitor at the same time, you can see status of NodeMCU's GPIO pins.

Now, click the button to turn LED1 ON while keeping an eye on the URL. Once you click the button, the ESP8266 receives a request for /led1on URL. It then turns the LED1 ON and serves a web page with status of LED updated. It also prints the

status of GPIO pin on the serial monitor. You can test LED2 button and check that it works in a similar way. Now, let's take a closer look at the code to see how it works, so that you are able to modify it to fulfill your needs.



ESP8266 as HTTP Server using WiFi Station (STA) mode

Now let's move on to our next example which demonstrates how to turn the ESP8266 into Station (STA) mode, and serve up web pages to any connected client under existing network. You need to modify ssid and password variables with your network credentials, so that ESP8266 can establish a connection with existing network. once you are done, go ahead and try the sketch out.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

/*Put your SSID & Password*/
const char* ssid = "YourNetworkName"; // Enter SSID here
const char* password = "YourPassword"; //Enter Password here

ESP8266WebServer server(80);

uint8_t LED1pin = D7;
bool LED1status = LOW;

uint8_t LED2pin = D6;
bool LED2status = LOW;

void setup() {
  Serial.begin(115200);
  delay(100);
  pinMode(LED1pin, OUTPUT);
```

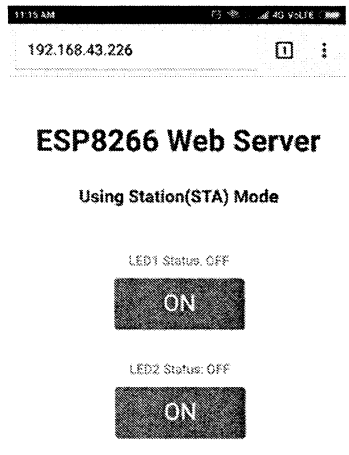
```
 pinMode(LED2pin, OUTPUT);

 Serial.println("Connecting to ");
 Serial.println(ssid);

 //connect to your local wi-fi network
 WiFi.begin(ssid, password);

 //check wi-fi is connected to wi-fi network
 while (WiFi.status() != WL_CONNECTED) {
 delay(1000);
 Serial.print(".");
 }
 Serial.println("");
 Serial.println("WiFi connected..!");
 Serial.print("Got IP: "); Serial.println(WiFi.localIP());

 server.on("/", handle_OnConnect);
 server.on("/led1on", handle_led1on);
 server.on("/led1off", handle_led1off);
 server.on("/led2on", handle_led2on);
 server.on("/led2off", handle_led2off);
 server.onNotFound(handle_NotFound);

 server.begin();
 Serial.println("HTTP server started");
}
void loop() {
 server.handleClient();
 if(LED1status)
 {digitalWrite(LED1pin, HIGH);}
 else
 {digitalWrite(LED1pin, LOW);}

 if(LED2status)
 {digitalWrite(LED2pin, HIGH);}
 else
 {digitalWrite(LED2pin, LOW);}
}

void handle_OnConnect() {
 LED1status = LOW;
 LED2status = LOW;
 Serial.println("GPIO7 Status: OFF | GPIO6 Status: OFF");
 server.send(200, "text/html", SendHTML(LED1status,LED2status));
}
```

```
void handle_led1on() {
 LED1status = HIGH;
 Serial.println("GPIO7 Status: ON");
 server.send(200, "text/html", SendHTML(true,LED2status));
}

void handle_led1off() {
 LED1status = LOW;
 Serial.println("GPIO7 Status: OFF");
 server.send(200, "text/html", SendHTML(false,LED2status));
}

void handle_led2on() {
 LED2status = HIGH;
 Serial.println("GPIO6 Status: ON");
 server.send(200, "text/html", SendHTML(LED1status,true));
}

void handle_led2off() {
 LED2status = LOW;
 Serial.println("GPIO6 Status: OFF");
 server.send(200, "text/html", SendHTML(LED1status,false));
}

void handle_NotFound(){
 server.send(404, "text/plain", "Not found");
}

String SendHTML(uint8_t led1stat,uint8_t led2stat){
 String ptr = "<!DOCTYPE html> <html>\n";
 ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0, user-scalable=no\">\n";
 ptr +="<title>LED Control</title>\n";
 ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px
auto; text-align: center;}\n";
 ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3
{color: #444444;margin-bottom: 50px;}\n";
 ptr +=".button {display: block;width: 80px;background-color: #1abc9c;border:
none;color: white;padding: 13px 30px;text-decoration: none;font-size: 25px;margin:
0px auto 35px;cursor: pointer;border-radius: 4px;}\n";
 ptr +=".button-on {background-color: #1abc9c;}\n";
 ptr +=".button-on:active {background-color: #16a085;}\n";
 ptr +=".button-off {background-color: #34495e;}\n";
 ptr +=".button-off:active {background-color: #2c3e50;}\n";
 ptr +="p {font-size: 14px;color: #888;margin-bottom: 10px;}\n";
```

```
 ptr +="</style>\n";
 ptr +="</head>\n";
 ptr +="<body>\n";
 ptr +="<h1>ESP8266 Web Server</h1>\n";
  ptr +="<h3>Using Station(STA) Mode</h3>\n";

  if(led1stat)
 {ptr +="<p>LED1 Status: ON</p><a class=\"button button-off\"
href=\"/led1off\">OFF</a>\n";}
 else
 {ptr +="<p>LED1 Status: OFF</p><a class=\"button button-on\"
href=\"/led1on\">ON</a>\n";}

 if(led2stat)
 {ptr +="<p>LED2 Status: ON</p><a class=\"button button-off\"
href=\"/led2off\">OFF</a>\n";}
 else
 {ptr +="<p>LED2 Status: OFF</p><a class=\"button button-on\"
href=\"/led2on\">ON</a>\n";}

 ptr +="</body>\n";
 ptr +="</html>\n";
 return ptr;
}
```

Accessing the Web Server in STA mode

After uploading the sketch, open the Serial Monitor at a baud rate of 115200. And press the RESET button on ESP8266. If everything is OK, it will output the dynamic IP address obtained from your router and show HTTP server started message. Next, load up a browser and point it to the IP address shown on the serial monitor. The NodeMCU should serve up a web page showing current status of LEDs and two buttons to control them. If take a look at the serial monitor at the same time, you can see status of NodeMCU's GPIO pins.



Now, click the button to turn LED1 ON while keeping an eye on the URL. Once you click the button, the ESP8266 receives a request for /led1on URL. It then turns the LED1 ON and serves a web page with status of LED updated. It also prints the status of GPIO pin on the serial monitor. You can test LED2 button and check that it works in a similar way.

Home Automation (over internet) using Blynk

Blynk is a full suite of software required to prototype, deploy, and remotely manage connected electronic devices at any scale: from personal IoT projects to millions of commercial connected products. With Blynk anyone can connect their hardware to the cloud and build a no-code iOS, Android, and web applications to analyze real-time and historical data coming from devices, control them remotely from anywhere in the world, receive important notifications, and much more...

Blynk is a multi-tenant solution. You can configure how users get access to the data by setting roles and configuring permissions. Applications made with Blynk are ready for the end-users. Whether it is your family member, an employee, or someone who has purchased your product, they will be able to download the app, connect the device and start using it.

In Blynk (Developer Mode) Developer is a special user who has access to all the functionality required to configure the platform for the use by end-users (also called as clients or end-customers). This is usually someone who builds the hardware, develops the firmware, and does all the device configurations. Developer can:

- Create and configure Device Templates in Blynk.Console
- Create and configure web dashboard UI
- Create and configure mobile dashboard UI
- Add new devices to the account
- Publish Templates to Blynk.Marketplace
- Use other features accordingly to the chosen plan

Blynk can control Digital and Analog I/O Pins (GPIOs) on your hardware directly. You don't even need to write code for it. But when it's not enough, you can use Virtual Pins. We designed Virtual Pins to exchange any data between your hardware and Blynk. Anything you connect to your hardware will be able to talk to Blynk. With Virtual Pins you can send something from the App, process it on the microcontroller, and then send it back to the smartphone. You can trigger functions, read I2C devices, convert values, control servo and DC motors etc. Virtual Pins can be used to interface with external libraries (Servo, LCD, and others) and implement custom functionality. Hardware may send data to the Widgets over the Virtual Pin like this:

```
1 Blynk.virtualWrite(pin, "abc");
2 Blynk.virtualWrite(pin, 123); Blynk.virtualWrite(pin, 12.34);
3 Blynk.virtualWrite(pin, "hello", 123, 12.34);
```

- Virtual pins are hardware-independent. This means that it's far easier to port your code from one hardware platform to another in the future (when you realize that the NodeMCU is far better than the Arduino Uno + ESP-01 that you started with, for example).
- You have far more control over what your widget does when using virtual pins. For example, if you want a single app button to switch multiple relays on or off at the same time then that's simple with virtual pins, but almost impossible using digital pins.
- Virtual pins are more predictable (stable if you like) than manipulating digital pins.

Virtual pins are really just a way of sending a message from the app to the code that's running on your board (via the Blynk server). There is no correlation between Virtual Pins and any of the physical GPIO pins on your hardware. If you want a Virtual Pin to change the state of one of your physical pins then you have to write the code to make this happen.

BLYNK_WRITE(vPin) function

In your C++ sketch, you can add a special function that is triggered automatically whenever the server tells your device that the value of your virtual pin has changed. This change would normally happen when the widget button in the app is pressed. This special function is called BLYNK_WRITE. Think of it as meaning that the Blynk.Cloud is telling your hardware "there a new value written to your virtual pin".

So, for Virtual Pin 0, your sketch would need this bit of code adding...

```
BLYNK_WRITE(V0)
{
// any code you place here will execute when the virtual pin value changes
}
```

That's okay if you want the same code to execute regardless of whether the button widget was turned on or off, but that's not much use in real life, so we need to find out what the new value of the virtual pin is. Don't forget, this will be a 0 if the button widget is off, and a 1 of it's on.

The server sends the current Virtual Pin value to the hardware as a parameter, and this can be obtained from within the BLYNK_WRITE(vPin) function with a piece of code like this...

```
int x = param.asInt();
```

This tells the code to get the value parameter from the virtual pin and store it in the local integer variable called x. We can then use this value to do different actions if the button widget is now on, compared to if it is now off.

Some datastreams send their values at text, or double/float point numbers (integers are whole numbers, double point variables are one with numbers to the right of the decimal point). To allow you to use these values the Blynk library also allows the following:

```
param.asStrng()
param.asFloat()
```

Sketch to control the physical GPIO pins on board

```
//Include the library files
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

//Define the relay pins
#define relay1 LED_BUILTIN              //D4

#define BLYNK_AUTH_TOKEN          //Enter your blynk auth token

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Babu";//Enter your WIFI name
char pass[] = "abcd1234";//Enter your WIFI password

//Get the button values
BLYNK_WRITE(V0) {
 bool value1 = param.asInt();
 // Check these values and turn the relay1 ON and OFF
 if (value1 == 1) {
  digitalWrite(relay1, LOW);
 } else {
  digitalWrite(relay1, HIGH);
 }
}

//Get the button values
BLYNK_WRITE(V1) {
 bool value2 = param.asInt();
 // Check these values and turn the relay2 ON and OFF
if (value2 == 1)
  {
   digitalWrite(relay2, LOW);
  }
else {
   digitalWrite(relay2, HIGH);
 }
}
```

```
void setup() {
  //Set the relay pins as output pins
  pinMode(relay1, OUTPUT);
  pinMode(relay2, OUTPUT);

  // Turn OFF the relay
  digitalWrite(relay1, HIGH);
  digitalWrite(relay2, HIGH);

  //Initialize the Blynk library
  Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
}

void loop() {
  //Run the Blynk library
  Blynk.run();
}
```

INTRODUCTION TO PYTHON PROGRAMMING

Python is a popular programming language. Python is a high-level object-oriented programming language that was created by Guido van Rossum and released in 1991. It is also called general-purpose programming language as it is used in almost every domain we can think of as mentioned below:

- Web Development
- Software Development
- Game Development
- AI & ML
- Data Analytics
- mathematics,
- system scripting.

Python can do

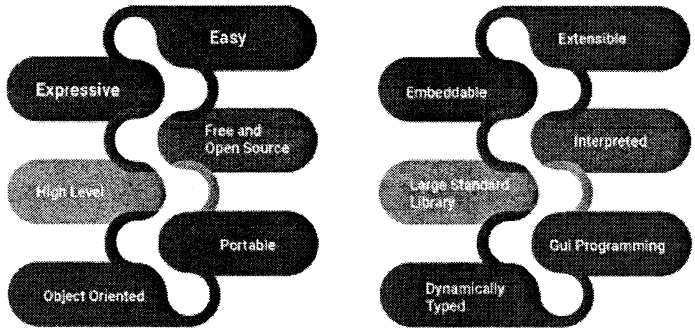
- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.
- It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Role of Python in IoT Development : IoT development requires a database to store generated data. MySQL provides IoT app developers the go-to relational database. It is the most convenient tool that evades the requirements to execute shell commands within a Python script. In the IoT development other programming languages like C, C++, Assembly, Java, JavaScript, and PHP. Python is the developers' favorite programming language when it comes to the development of IoT applications. Python programming powers intuitive interfaces of intelligent and effective Internet of Things (IoT) systems that are paramount in remote sensor networks, big data and data analysis, automation, and machine learning.

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi).
- Python is the IoT app developers' first choice because of its dynamism and interpretation power.
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.
- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.
- IoT development requires a high-level programming language. A programming language that focuses on code readability.
- Python is easy to learn and it generally has few steps of implementation as compared to Java and C.
- The multi-functional programming language is used for mathematics, software development, system scripting, and web development.
- Used for server-side development. Handles complex data with ease.
- Python runs on Mac, Linux, Raspberry Pi, and Windows easily.
- A fast programming language that runs on the interpreter.
- It is a Scripting language that enables developers to easily develop web and desktop applications.
- A functional, procedural and object-oriented language that can be translated to a binary language like java.
- It has code portability. The developers do not have to write new codes every time for different machines.
- An Open-source framework that is easily available for public download.
- Huge community support enables developers to create user-friendly apps each time.

Features of Python

As a programming language, the features of Python brought to the table are many. Some of the most significant features of Python are:



1. **Easy to Code:** Python is a very developer-friendly language which means that anyone and everyone can learn to code it in a couple of hours or days. As compared to other object-oriented programming languages like Java, C, C++, and C#, Python is one of the easiest to learn.
2. **Open Source and Free:** Python is an open-source programming language which means that anyone can create and contribute to its development. Python has an online forum where thousands of coders gather daily to improve this language further. Along with this Python is free to download and use in any operating system, be it Windows, Mac or Linux.
3. **Support for GUI:** GUI or Graphical User Interface is one of the key aspects of any programming language because it has the ability to add flair to code and make the results more visual. Python has support for a wide array of GUIs which can easily be imported to the interpreter, thus making this one of the most favorite languages for developers.
4. **Object-Oriented Approach:** One of the key aspects of Python is its object-oriented approach. This basically means that Python recognizes the concept of class and object encapsulation thus allowing programs to be efficient in the long run.
5. **High-Level Language:** Python has been designed to be a high-level programming language, which means that when you code in Python you don't need to be aware of the coding structure, architecture as well as memory management.
6. **Integrated by Nature:** Python is an integrated language by nature. This means that the python interpreter executes codes one line at a time. Unlike other object-oriented programming languages, we don't need to compile Python code thus making the debugging process much easier and efficient. Another advantage of this is, that upon execution the Python code is immediately converted into an intermediate form also known as byte-code which makes it easier to execute and also saves runtime in the long run.
7. **Highly Portable:** Suppose you are running Python on Windows and you need to shift the same to either a Mac or a Linux system, then you can easily achieve the same in Python without having to worry about changing the code. This is not possible in other programming languages, thus making Python one of the most portable languages available in the industry.

8. **Highly Dynamic:** As mentioned in an earlier paragraph, Python is one of the most dynamic languages available in the industry today. What this basically means is that the type of a variable is decided at the run time and not in advance. Due to the presence of this feature, we do not need to specify the type of the variable during coding, thus saving time and increasing efficiency.
9. **Extensive Array of Library:** Out of the box, Python comes inbuilt with a large number of libraries that can be imported at any instance and be used in a specific program. The presence of libraries also makes sure that you don't need to write all the code yourself and can import the same from those that already exist in the libraries.
10. **Support for Other Languages:** Being coded in C, Python by default supports the execution of code written in other programming languages such as Java, C, and C#, thus making it one of the versatile in the industry.

IoT applications function efficiently with the help of Python **libraries/packages** which include:

1. **NUMPY:** Numpy is a scientific computing package that helps to create datasets to test with the time series data in IoT. Numpy features are used in IoT to read sensor bulk data from the database inbuilt functions in the system
2. **SOCKETS AND MYSQLDB:** Sockets that facilitate networking in IoT devices include TCP/IP and UDP, which are compatible to work with Python packages. TCP/IP and UDP act as transport layer protocols for communication. The MySQLdb is a go-to relational format database that helps in the development of remote stores for the IoT system.
3. **MATPLOTLIB:** To get data insights, matplotlib visualizes the most paramount operations by giving a variety of graphs to represent the data.
4. **REQUESTS, TKINTER AND TENSORFLOW:** To make HTTP calls and parse responses in Python, the **request package** acts as a major protocol for data exchanges. **Tkinter GUI** puts the aspects of Python script in a controlled distribution, which enables functional testing and repeated executions in IoT Python devices. Therefore, the numerical computations of machine learning initiated into the IoT systems utilize the representation in data flow graphs dealing with huge non-linear datasets and deep learning aspects.

Some of the best solutions for IoT in the Python programming language are as follows:

1. Python on Raspberry Pi
2. Python on PyBoard
3. ESP8266, ESP32 with Micropython

➢ **Python on Raspberry Pi :** The primary objective of running Python on an IoT device that pops up in mind is grabbing the Raspberry Pi from the table. Python is pre-installed in the operating system, and the only objective left for us is to write the coding script. In this scenario, we can control the I/O ports on the expansion bar of the Raspberry Pi. Fortunately, the board supports wireless communication (Bluetooth and WiFi) and Ethernet. We can also connect a monitor to the HDMI output, a specialized 3.2" 320x240 TFT LCD, or a low energy consumption E-Ink 2.13" 250x122 display for Raspberry Pi.

There are controllers available in a large variety of computing power and budgets. We can choose these controllers for the IoT system - ranging from the fast Raspberry Pi 4 Model B 8 GB to the smallest Raspberry Pi Zero, all supporting the Python programming language. In case of necessity, we can install the earlier version of Python 2.7 for past compatibility.

➢ **Python on PyBoard :** Another great solution for Python in IoT devices is the PyBoard with an STM32F405RG microcontroller.The PyBoard is considered a compact as well as a powerful electronics development board. It works on MicroPython. The PyBoard connects to the PC through USB, providing us with a USB flash drive to store the Python scripts and a serial Python prompt (a REPL) for instant programming. This works with Windows, MacOS, and Linux.

PyBoard executes MicroPython, which is a lightweight implementation of the standard CPython interpreter. The official documentation also says: "MicroPython is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimized to run on microcontrollers and in constrained environments. The MicroPython pyboard is a compact electronic circuit board that runs MicroPython on the bare metal, giving you a low-level Python operating system that can be used to control all kinds of electronic projects. MicroPython is packed full of advanced features such as an interactive prompt, arbitrary precision integers, closures, list comprehension, generators, exception handling and more. Yet it is compact enough to fit and run within just 256k of code space and 16k of RAM."

MicroPython is an entire rewrite of the Python programming language to fit and execute on a microcontroller. It involves various optimization for efficiency and consumes quite less RAM. MicroPython executes bare-metal on the PyBoard, necessarily providing us with an Operating System based on Python. The in-built pyb module consists of functions and classes in order to control the peripherals available on the board, like I2C, UART, ADC, DAC, and SPI.

➢ **ESP8266, ESP32 with MicoPython:** Another option could be using ESP8266 and ESP 32 to run Python. We have to create a device based on the Internet of Things with low power consumption, great capabilities, and integration with wireless Wi-Fi networks. More precisely, we can use MicroPython. Once we installed Python on the system, we can use the pip installer in the command line in order to install the **esptool** module. The syntax for the same is shown below:

```
$ pip install esptool
```

The installation procedure of the MicroPython is pretty easy. We can download the firmware from the website and install it with the help of esptool, not forgetting to format the board before installing it. We can also use one of the IDEs used for developing with MircoPython. The complete procedure of development is carried out on a working computer, and then it is compiled and saved in the memory of an ESP8266 or ESP32 microcontroller.

MicroPython imposes many restrictions compared to regular Python; however, in general, we can easily write the necessary functionality on the client-side and execute it effectively on ESP microcontrollers. This option is relatively more cost-effective than buying PyBoard.

Use of Python in IoT Backend

We can use Python as a Backend programming language for the Internet of Things in many ways. Some of them are as follows:

- **MQTT protocol with Python:** One of the most popular connection methods for IoT devices is MQTT, and it is a protocol used for effective implementation with Python.

  The MQTT protocol is a machine-to-machine (M2M)/Internet of Things connectivity protocol designed as a highly lightweight publish/subscribe messaging transport. It is used to connect to remote locations where a small code footprint is needed, and network bandwidth is premium.

  The code of the **Poho** library offers a client class that allows applications to link to an MQTT broker in order to publish messages, subscribe to topics, and receive published messages. It also delivers some helper functions to make things simpler in publishing one-off messages to MQTT servers.

- **IoT backend on Flask in Python:** We can also use the Flask micro framework to write the backend for the IoT systems. The Flask micro framework is a quick and hassle-free tool that easily set up server-side I/O information, and it is also packed with many functionalities that make work more efficient. We can start by deciding the requests we have to serve from the IoT devices. We then have to set up the Flask micro framework and write a block of code. The **GET** method will then return information as per the request from the side of the client.

In several cases, we are best off focusing on the RESTful protocol while working with the IoT devices. This allows us to simplify the exchange between the components of the system and helps us to expand the system of exchanging information in the future.

The disadvantage of utilizing this method is the potential lack of starting the data transfer from the server to the device. Thus, the IoT must periodically and independently pull from the server. Rest easy, as there are keys to report this risk. We can utilize web sockets or a Python library for Pushsafer. PushSafer is an easy and safe way to send and receive push notifications in real-time to Android, iOS, and Windows devices (mobile as well as desktop), including internet browsers such as Google Chrome, Mozilla Firefox, Opera, etc.

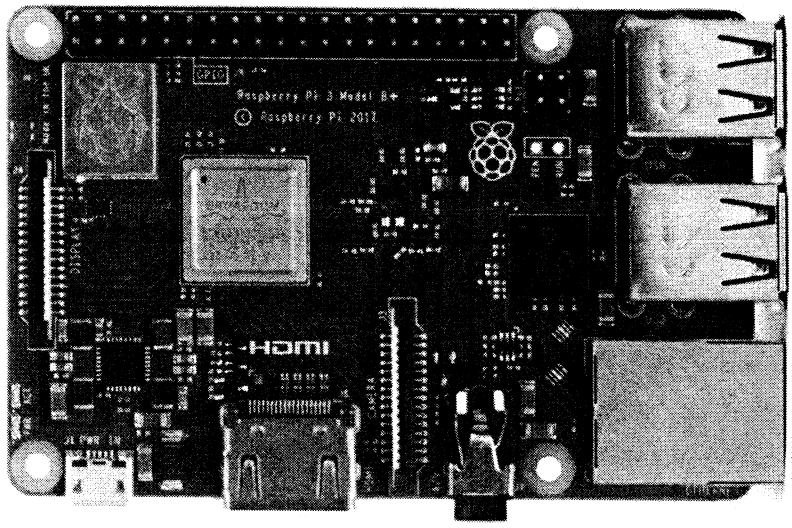
- **Microsoft Azure IoT backend in Python:** Microsoft has released a new open-source extension for IoT to extend the capabilities of Azure CLI 2.0. Azure CLI 2.0 involves commands to interact with the Azure Resource Manager and endpoints of management.

  For instance, we can utilize Azure CLI 2.0 to build an Azure Virtual Machine or IoT Hub. The extension of CLI allows an Azure service to complement Azure CLI by providing users access to additional capabilities specified to services. The Extension of IoT offers programmers command-line access to the capabilities of the IoT Edge, IoT Hub and IoT Hub Device Provisioning Service.

  Azure CLI 2.0 allows instant management of resources of Azure IoT Hub, devices provisioning services instances, and associated hubs. The new IoT extension enriches Azure CLI 2.0 with features such as device management and all IoT Edge capabilities:

  1. Azure CLI 2.0 IoT capabilities - Control Plane
  2. Managing instances of IoT Hub, consumer - groups and jobs
  3. Managing instances of device provisioning service, access-policies, Linked - hub and certificates
  4. New features for extensions - data plane
  5. Managing device and edge module identities and their respective twin definitions
  6. Querying IoT Hub for details like device and module jobs, twins, and messaging routing
  7. Invoking methods of device and module
  8. Generating SAS tokens and grabbing connection strings
  9. Cloud-to-device and Device-to-cloud messaging
  10. Device file uploading
  11. Device simulation for testing

INTRODUCTION TO RASPBERRY PI.



Raspberry Pi is the name of a series of single-board computers made by the Raspberry Pi Foundation, a UK charity that aims to educate people in computing and create easier access to computing education. The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins, allowing you to control electronic components for physical computing and explore the Internet of Things (IoT). All over the world, people use the Raspberry Pi to learn programming skills, build hardware projects, do home automation, implement Kubernetes clusters and Edge computing, and even use them in industrial applications.

The Raspberry Pi launched in 2012, and there have been several iterations and variations released since then. The original Pi had a single-core 700MHz CPU and just 256MB RAM, and the latest model has a quad-core CPU clocking in at over 1.5GHz, and 4GB RAM.

There have been many generations of the Raspberry Pi line: from Pi 1 to 4, and even a Pi 400. There has generally been a Model A and a Model B of most generations. Model A has been a less expensive variant, and tends to have reduced RAM and fewer ports (such as USB and Ethernet). The Pi Zero is a spinoff of the original (Pi 1) generation, made even smaller and cheaper. Here's the lineup so far:

- Pi 1 Model B (2012)
- Pi 1 Model A (2013)
- Pi 1 Model B+ (2014)
- Pi 1 Model A+ (2014)
- Pi 2 Model B (2015)
- Pi Zero (2015)
- Pi 3 Model B (2016)
- Pi Zero W (2017)

- Pi 3 Model B+ (2018)
- Pi 3 Model A+ (2019)
- Pi 4 Model A (2019)
- Pi 4 Model B (2020)
- Pi 400 (2021)

- The first generation (Raspberry Pi Model B) was released in February 2012, followed by the simpler and cheaper Model A.
- In 2014, the Foundation released a board with an improved design, Raspberry Pi Model B+. These first generation boards feature ARM11 processors, are approximately credit-card sized and represent the standard mainline form-factor. Improved A+ and B+ models were released a year later. A "Compute Module" was released in April 2014 for embedded applications.
- The Raspberry Pi 2 was released in February 2015 and initially featured a 900 MHz 32-bit quad-core ARM Cortex-A7 processor with 1 GB RAM. Revision 1.2 featured a 900 MHz 64-bit quad-core ARM Cortex-A53 processor (the same as that in the Raspberry Pi 3 Model B, but under clocked to 900 MHz).
- Raspberry Pi 3 Model B was released in February 2016 with a 1.2 GHz 64-bit quad core ARM Cortex-A53 processor, on-board 802.11n Wi-Fi, Bluetooth and USB boot capabilities.
- On Pi Day 2018, the Raspberry Pi 3 Model B+ was launched with a faster 1.4 GHz processor, a three-times faster gigabit Ethernet (throughput limited to ca. 300 Mbit/s by the internal USB 2.0 connection), and 2.4 / 5 GHz dual- and 802.11ac Wi-Fi (100 Mbit/s). Other features are Power over Ethernet (PoE) (with the add-on PoE HAT), USB boot and network boot (an SD card is no longer required).
- Raspberry Pi 4 Model B was released in June 2019 with a 1.5 GHz 64-bit quad core ARM Cortex-A72 processor, on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet (throughput not limited), two USB 2.0 ports, two USB 3.0 ports, 1–8 GB of RAM, and dual-monitor support via a pair of micro HDMI (HDMI Type D) ports for up to 4K resolution. The version with 1 GB RAM has been abandoned and the prices of the 2 GB version have been reduced. The 8 GB version has a revised circuit board. The Pi 4 is also powered via a USB-C port, enabling additional power to be provided to downstream peripherals, when used with an appropriate PSU. But the Pi can only be operated with 5 volts and not 9 or 12 volts like other mini computers of this class. The manufacturer is now using this chip for the Pi 4 B and Pi 400. However, the tack frequency of the Pi 4 B was not increased in the factory.
- Raspberry Pi 400 was released in November 2020. It features a custom board that is derived from the existing Raspberry Pi 4, specifically remodelled with a keyboard attached. The case was derived from that of the Raspberry Pi Keyboard. A robust cooling solution (i.e. a broad metal plate) and an upgraded switched-mode power supply allow the Raspberry Pi 400's Broadcom BCM2711C0 processor to be clocked at 1.8 GHz, which is 20% faster than the Raspberry Pi 4 it is based on. The keyboard-computer features 4 GB of LPDDR4 RAM.

Raspberry Pi Zero

- A Raspberry Pi Zero with smaller size and reduced input/output (I/O) and general-purpose input/output (GPIO) capabilities was released in November 2015.
- On February 2017, the Raspberry Pi Zero W was launched, a version of the Zero with Wi-Fi and Bluetooth capabilities.
- On January 2018, the Raspberry Pi Zero WH was launched, a version of the Zero W with pre-soldered GPIO headers.
- On October 2021, the Raspberry Pi Zero 2 W was launched, a version of the Zero W with a system in a package (SiP) designed by Raspberry Pi and based on the Raspberry Pi 3. In contrast to the older ones, the Pi 2 W is 64 bit capable.

Raspberry Pi Pico

- Raspberry Pi Pico was released in January 2021. It was Raspberry Pi's first board based upon a single microcontroller chip; the RP2040, which was designed by Raspberry Pi in the UK. The Pico has 264 KB of RAM and 2 MB of flash memory. It is programmable in MicroPython, CircuitPython, C and Rust. The Raspberry Pi Foundation has partnered with Adafruit, Pimoroni, Arduino and SparkFun to build Accessories for Raspberry Pi Pico and variety of other boards using RP2040 Silicon Platform. Rather than perform the role of general purpose computer it is designed for physical computing, similar in concept to an Arduino.

The Raspberry Pi Foundation works to put the power of computing and digital making into the hands of people all over the world. It does this by providing low-cost, high-performance computers that people use to learn, solve problems, and have fun. It provides outreach and education to help more people access computing and digital making—it develops free resources to help people learn about computing and making things with computers and also trains educators who can guide other people to learn.

Open source : The Raspberry Pi operates in the open source ecosystem: it runs Linux (a variety of distributions), and its main supported operating system, Pi OS, is open source and runs a suite of open source software. The Raspberry Pi Foundation contributes to the Linux kernel and various other open source projects as well as releasing much of its own software as open source. The Raspberry Pi's schematics are regularly released as documentation, but the board is not open hardware.

Some people buy a Raspberry Pi to learn to code, and people who can already code use the Pi to learn to code electronics for physical projects. The Raspberry Pi can open opportunities for you to create your own home automation projects, which is popular among people in the open source community because it puts you in control, rather than using a proprietary closed system.

Hardware : The Raspberry Pi hardware has evolved through several versions that feature variations in the type of the central processing unit, amount of memory capacity, networking support, and peripheral-device support.

The Pi Zero models are similar, but lack the Ethernet and USB hub components. The Ethernet adapter is internally connected to an additional USB port. In Model A, A+, and the Pi Zero, the USB port is connected directly to the system on a chip (SoC). On the Pi 1 Model B+ and later models the USB/Ethernet chip contains a five-port USB hub, of which four ports are available, while the Pi 1 Model B only provides two. On the Pi Zero, the USB port is also connected directly to the SoC, but it uses a micro USB (OTG) port. Unlike all other Pi models, the 40 pin GPIO connector is omitted on the Pi Zero, with solderable through-holes only in the pin locations. The Pi Zero WH remedies this.

Processor speed ranges from 700 MHz to 1.4 GHz for the Pi 3 Model B+ or 1.5 GHz for the Pi 4; on-board memory ranges from 256 MB to 8 GB random-access memory (RAM), with only the Raspberry Pi 4 having more than 1 GB. Secure Digital (SD) cards in MicroSDHC form factor (SDHC on early models) are used to store the operating system and program memory, however some models also come with onboard eMMC storage and the Raspberry Pi 4 can also make use of USB-attached SSD storage for its operating system. The boards have one to five USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm tip-ring-sleeve jack for audio output. Lower-level output is provided by a number of GPIO pins, which support common protocols like I^2C. The B-models have an 8P8C Ethernet port and the Pi 3, Pi 4 and Pi Zero W have on-board Wi-Fi 802.11n and Bluetooth.

Processor : The Raspberry Pi 2B uses a 32-bit 900 MHz quad-core ARM Cortex-A7 processor. The Broadcom BCM2835 SoC used in the first generation Raspberry Pi includes a 700 MHz ARM1176JZF-S processor, VideoCore IV graphics processing unit (GPU), and RAM. It has a level 1 (L1) cache of 16 KB and a level 2 (L2) cache of 128 KB. The level 2 cache is used primarily by the GPU. The SoC is stacked underneath the RAM chip, so only its edge is visible. The ARM1176JZ(F)-S is the same CPU used in the original iPhone, although at a higher clock rate, and mated with a much faster GPU.

The earlier V1.1 model of the Raspberry Pi 2 used a Broadcom BCM2836 SoC with a 900 MHz 32-bit, quad-core ARM Cortex-A7 processor, with 256 KB shared L2 cache. The Raspberry Pi 2 V1.2 was upgraded to a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor, the same one which is used on the Raspberry Pi 3, but underclocked (by default) to the same 900 MHz CPU clock speed as the V1.1. The BCM2836 SoC is no longer in production as of late 2016.

The Raspberry Pi 3 Model B uses a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor, with 512 KB shared L2 cache. The Model A+ and B+ are 1.4 GHz.

The Raspberry Pi 4 uses a Broadcom BCM2711 SoC with a 1.5 GHz (later models: 1.8 GHz) 64-bit quad-core ARM Cortex-A72 processor, with 1 MB shared L2 cache. Unlike previous models, which all used a custom interrupt controller poorly suited for virtualization, the interrupt controller on this SoC is compatible with the ARM Generic Interrupt Controller (GIC) architecture 2.0, providing hardware support for interrupt distribution when using ARM virtualization capabilities.

Raspberry Pi Zero and Zero W use the same Broadcom BCM2835 SoC as the first generation Raspberry Pi, although now running at 1 GHz CPU clock speed.

The Raspberry Pi Zero W 2 uses the RP3A0-AU CPU, a 1 GHz 64 bit ARM Cortex A53, on 512MB of SDRAM. Documentation states this "system-on-package" is a Broadcom BCM2710A1 package, using a BCM2837 Broadcom chip as core, which is an ARM v8 quad-core. The RPi3 also uses the BCM2837, but at 1.2 GHz, since the Pi Zero W 2 clock is 1 GHz.

Performance : While operating at 700 MHz by default, the first generation Raspberry Pi provided a real-world performance roughly equivalent to 0.041 GFLOPS. On the CPU level the performance is similar to a 300 MHz Pentium II of 1997

Raspberry Pi 2 V1.1 included a quad-core Cortex-A7 CPU running at 900 MHz and 1 GB RAM. It was described as 4–6 times more powerful than its predecessor. The GPU was identical to the original. In parallelised benchmarks, the Raspberry Pi 2 V1.1 could be up to 14 times faster than a Raspberry Pi 1 Model B+.

The Raspberry Pi 3, with a quad-core Cortex-A53 processor, is described as having ten times the performance of a Raspberry Pi 1. Benchmarks showed the Raspberry Pi 3 to be approximately 80% faster than the Raspberry Pi 2 in parallelised tasks. The Raspberry Pi 4, with a quad-core Cortex-A72 processor, is described as having three times the performance of a Raspberry Pi 3.

RAM: The early designs of the Raspberry Pi Model A and B boards included only 256 MB of random access memory (RAM). The later Model B with 512 MB RAM, was released. The Raspberry Pi 2 has 1 GB of RAM. The Raspberry Pi 3 has 1 GB of RAM in the B and B+ models, and 512 MB of RAM in the A+ model. The Raspberry Pi Zero and Zero W have 512 MB of RAM. The Raspberry Pi 4 is available with 2, 4 or 8 GB of RAM. A 1 GB model was originally available at launch in June 2019 but was discontinued in March 2020, and the 8 GB model was introduced in May 2020.

Networking: The Model A, A+ and Pi Zero have no Ethernet circuitry and are commonly connected to a network using an external user-supplied USB Ethernet or Wi-Fi adapter. On the Model B and B+ the Ethernet port is provided by a built-in USB Ethernet adapter using the SMSC LAN9514 chip. The Raspberry Pi 3 and Pi Zero W (wireless) are equipped with 2.4 GHz WiFi 802.11n (150 Mbit/s) and Bluetooth 4.1 (24 Mbit/s) based on the Broadcom BCM43438 The Raspberry Pi 3B+ features dual-band IEEE 802.11b/g/n/ac WiFi, Bluetooth 4.2, and Gigabit Ethernet . The Raspberry Pi 4 has full gigabit Ethernet .

Operating systems: Various operating systems for the Raspberry Pi can be installed on a MicroSD, MiniSD or SD card, depending on the board and available adapters. The Raspberry Pi Foundation provides Raspberry Pi OS (formerly called Raspbian), a Debian-based Linux distribution for download, as well as third-party Ubuntu, Windows 10 IoT Core, RISC OS, LibreELEC (specialised media centre distribution) and specialised distributions for the Kodi media centre and classroom management. It promotes Python and Scratch as the main programming languages, with support for many other languages. The default firmware is closed source, while unofficial open source is available. Many other operating systems can also run on the Raspberry Pi.

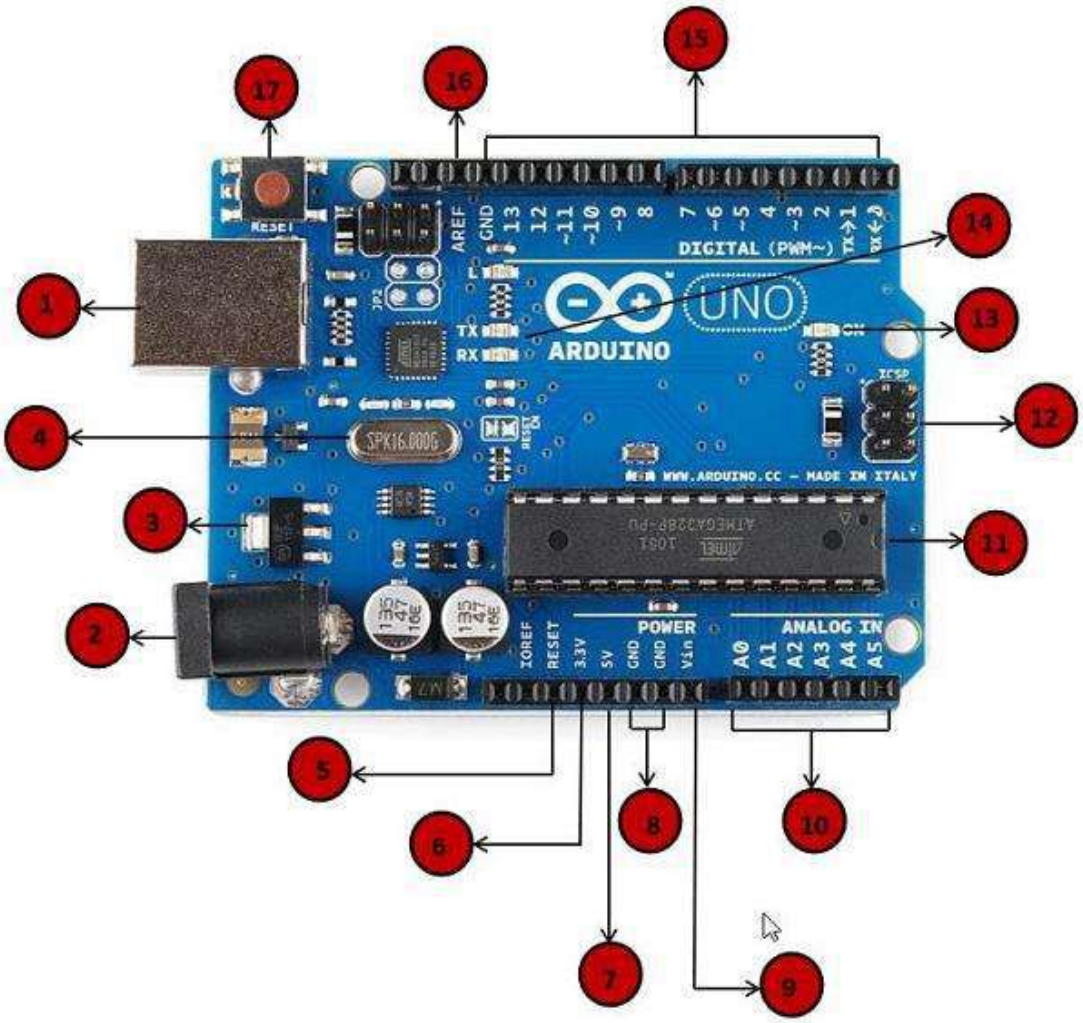
Other operating systems (not Linux- nor BSD-based)

- Broadcom VCOS
- Haiku
- HelenOS
- Plan 9
- RISC OS Pi
- Ultibo Core
- Windows 10 IoT Core

Other operating systems (Linux-based)

- Alpine Linux
- Android Things
- Arch Linux ARM
- Ark OS
- Batocera
- CentOS
- Devuan
- emteria OS
- Fedora
- Gentoo Linux
- Kali Linux
- OpenEuler
- openSUSE
- OpenWrt
- postmarketOS
- RetroPie
- Sailfish OS
- Slackware ARM
- SolydXK
- Tiny Core Linux
- Ubuntu-based: Lubuntu and Xubuntu
- Void Linux

Firmware : The official firmware is a freely redistributable binary blob, that is proprietary software. A minimal proof-of-concept open source firmware is also available, mainly aimed at initialising and starting the ARM cores as well as performing minimal startup that is required on the ARM side. It is also capable of booting a very minimal Linux kernel, with patches to remove the dependency on the mailbox interface being responsive. It is known to work on Raspberry Pi 1, 2 and 3, as well as some variants of Raspberry Pi Zero.



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.

	**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.
	**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.
	**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.
	**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.
	**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.

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.

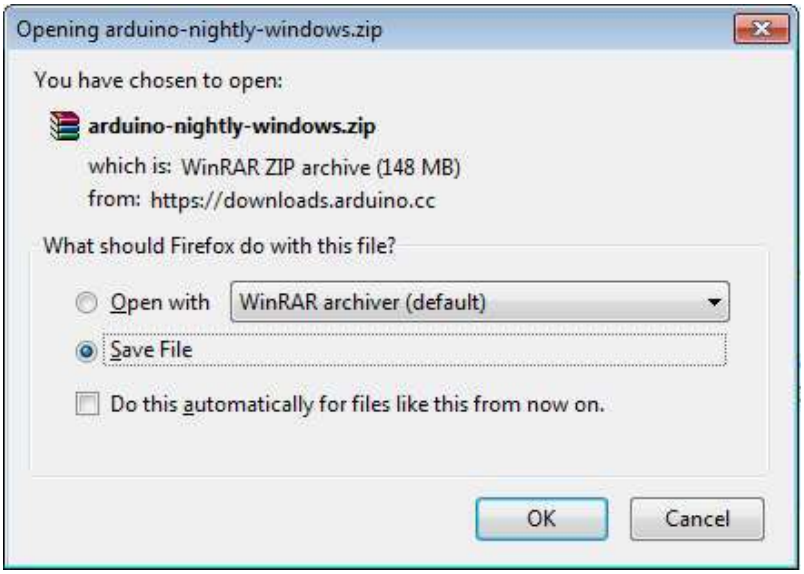
Step 1 − First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega

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.



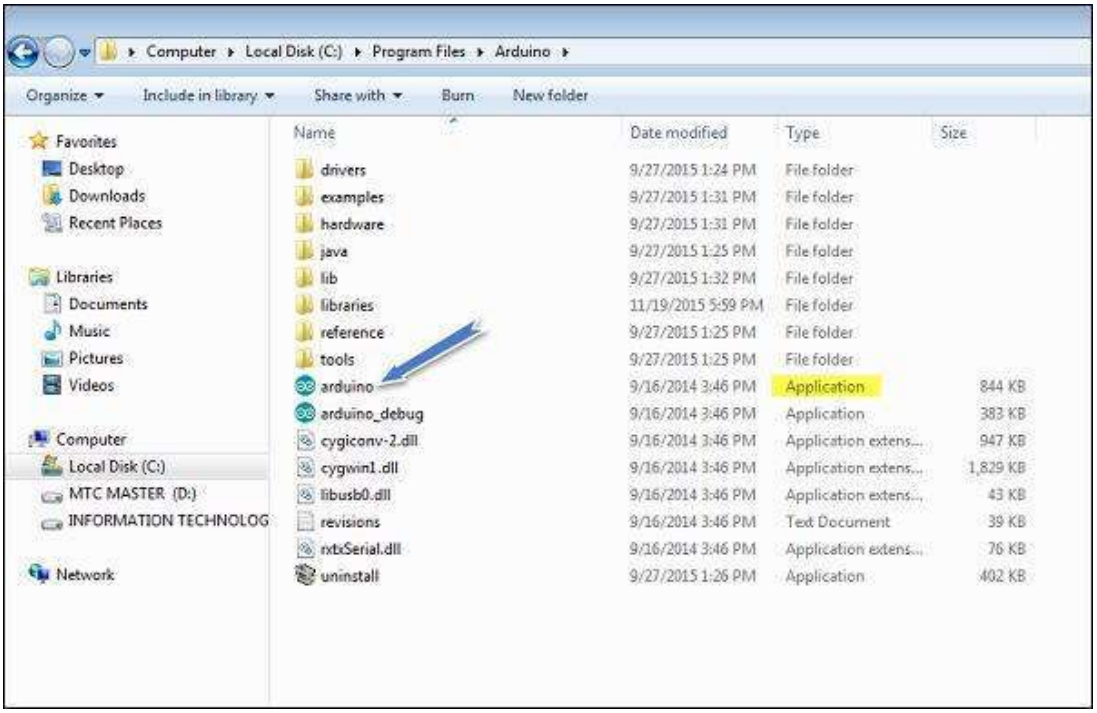
Step 3 – Power up your board.

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.

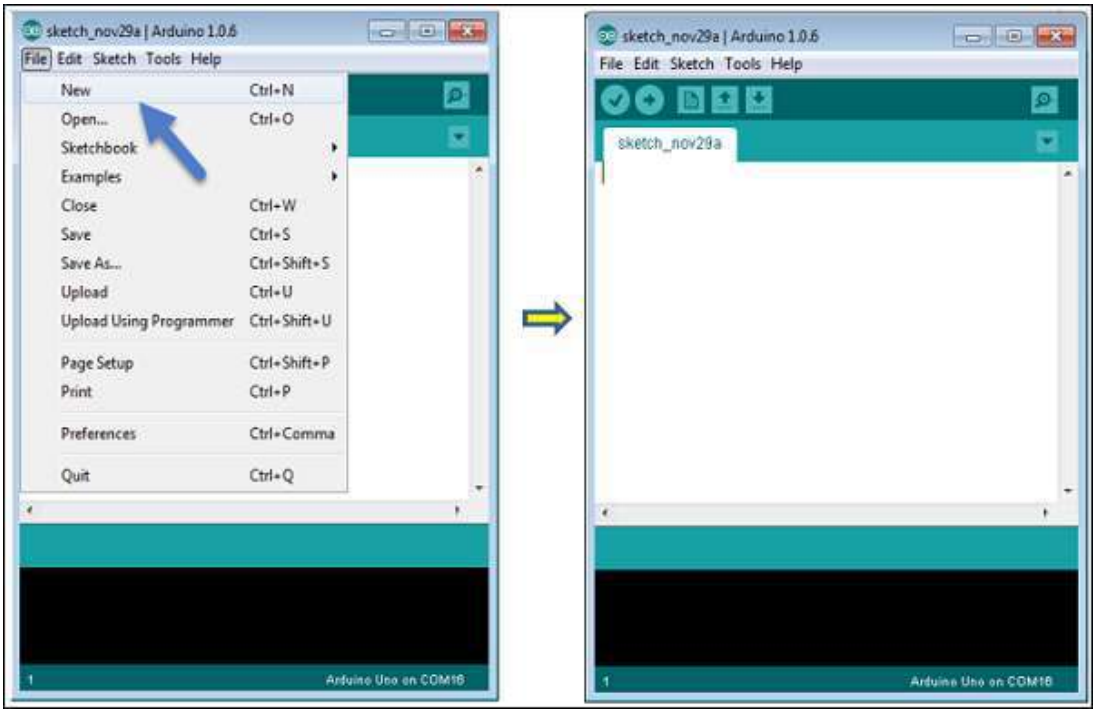


Step 5 – Open your first project.

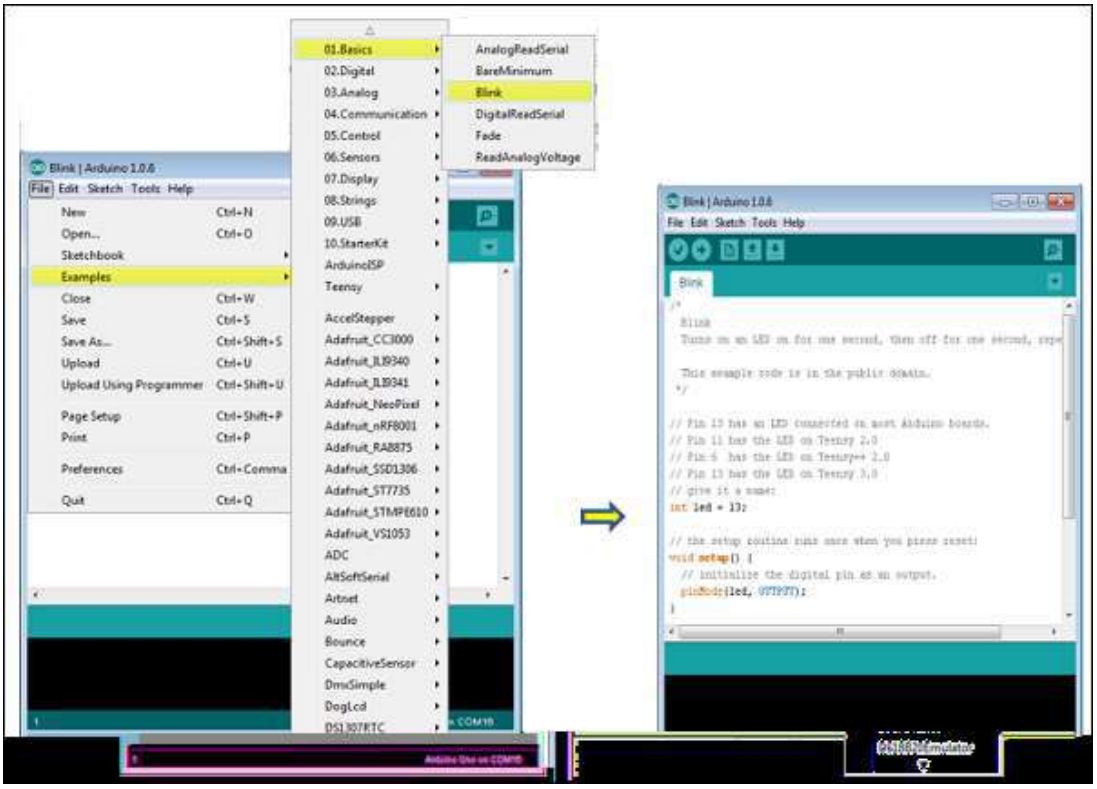
Once the software starts, you have two options −

- 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.

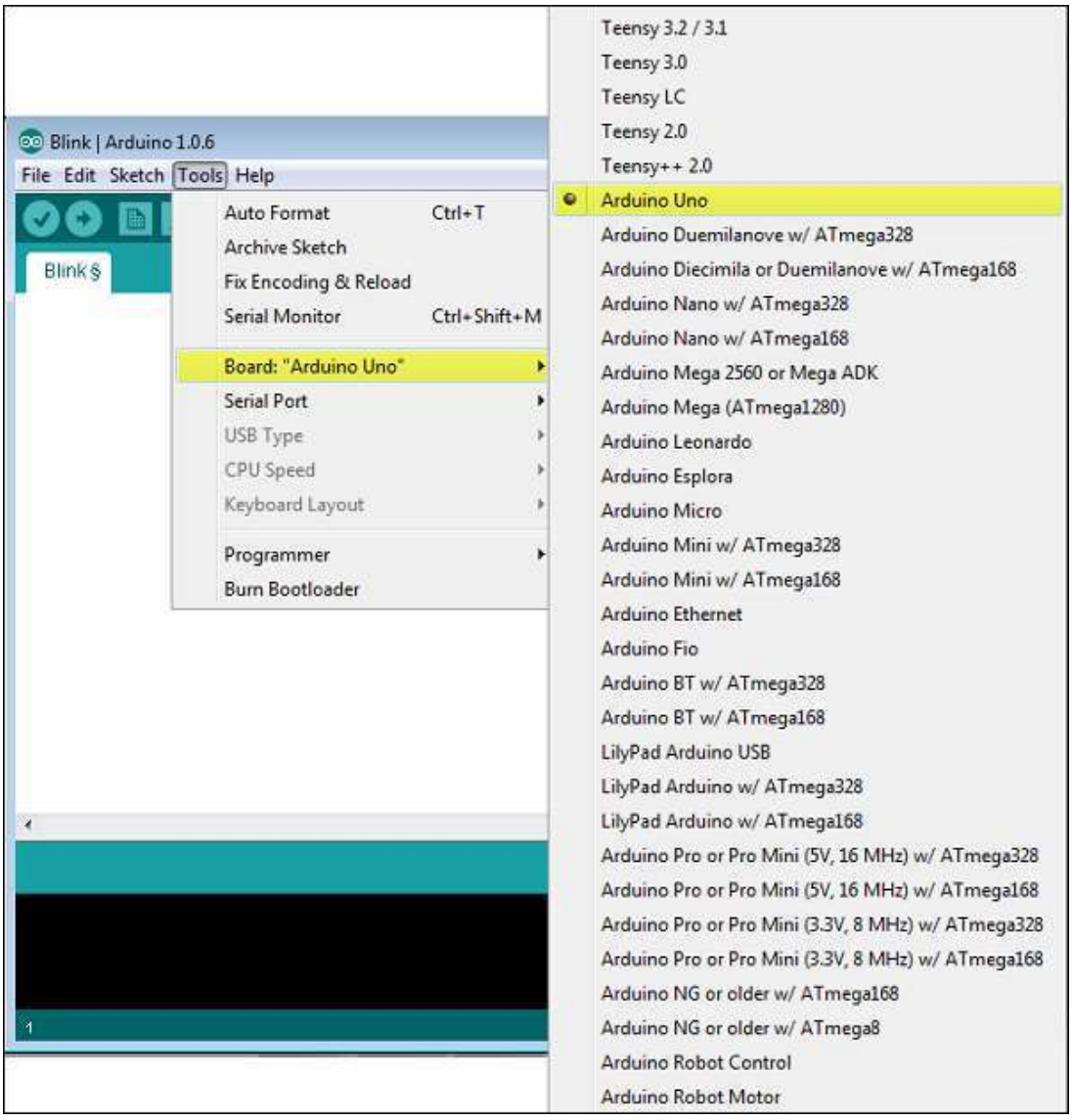


Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

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.

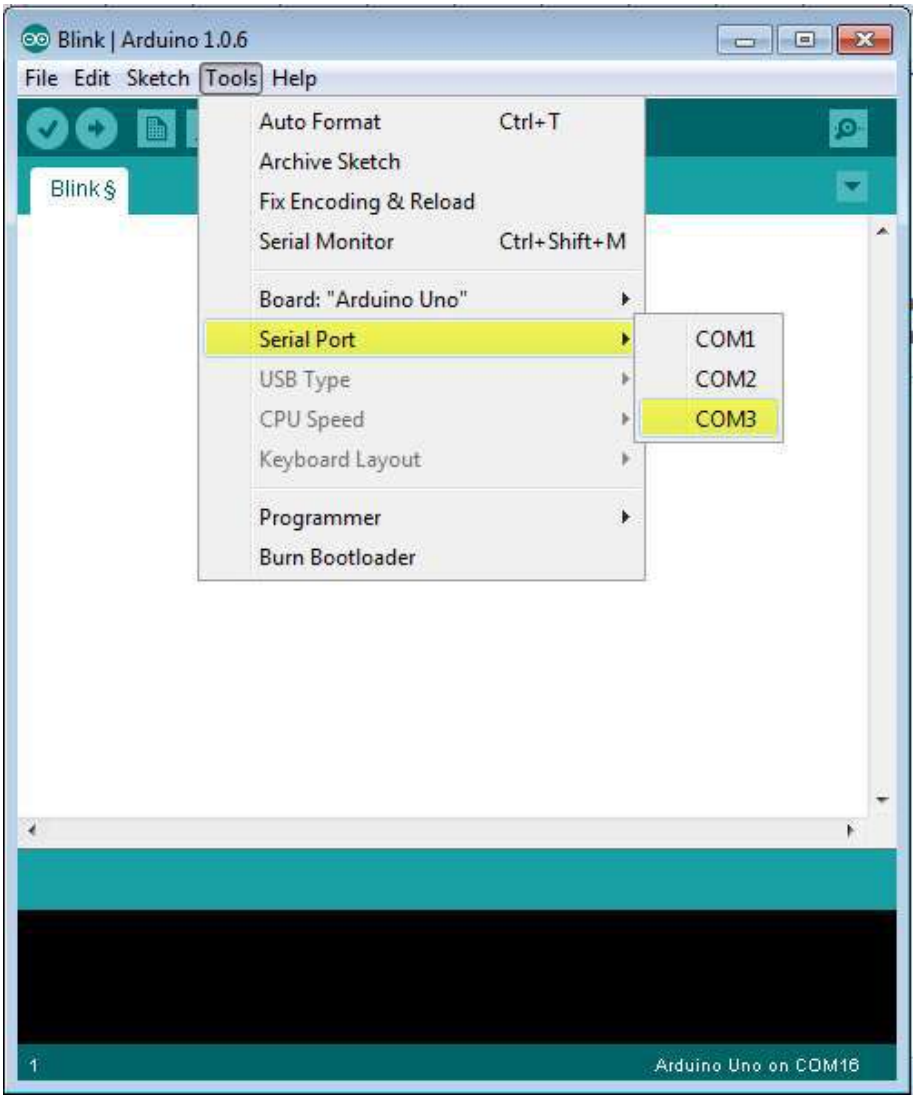
Go to Tools → Board and select your board.



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

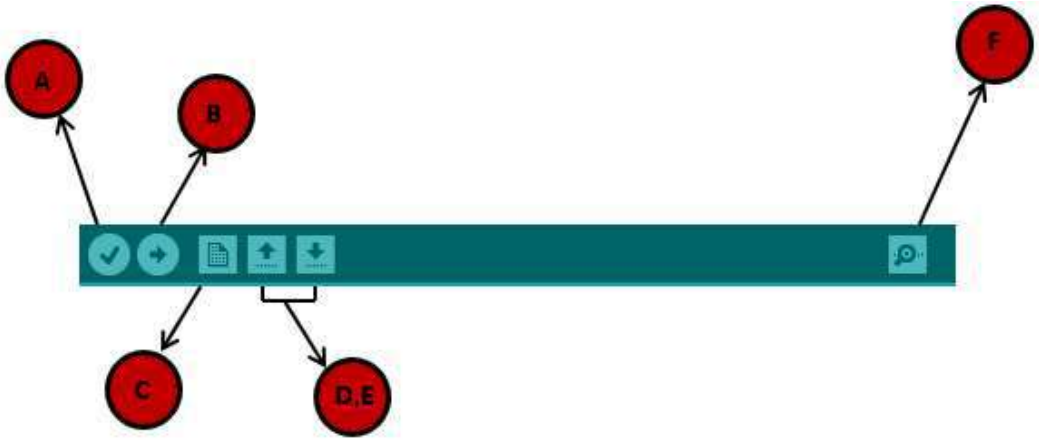
Step 7 – Select your serial port.

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.

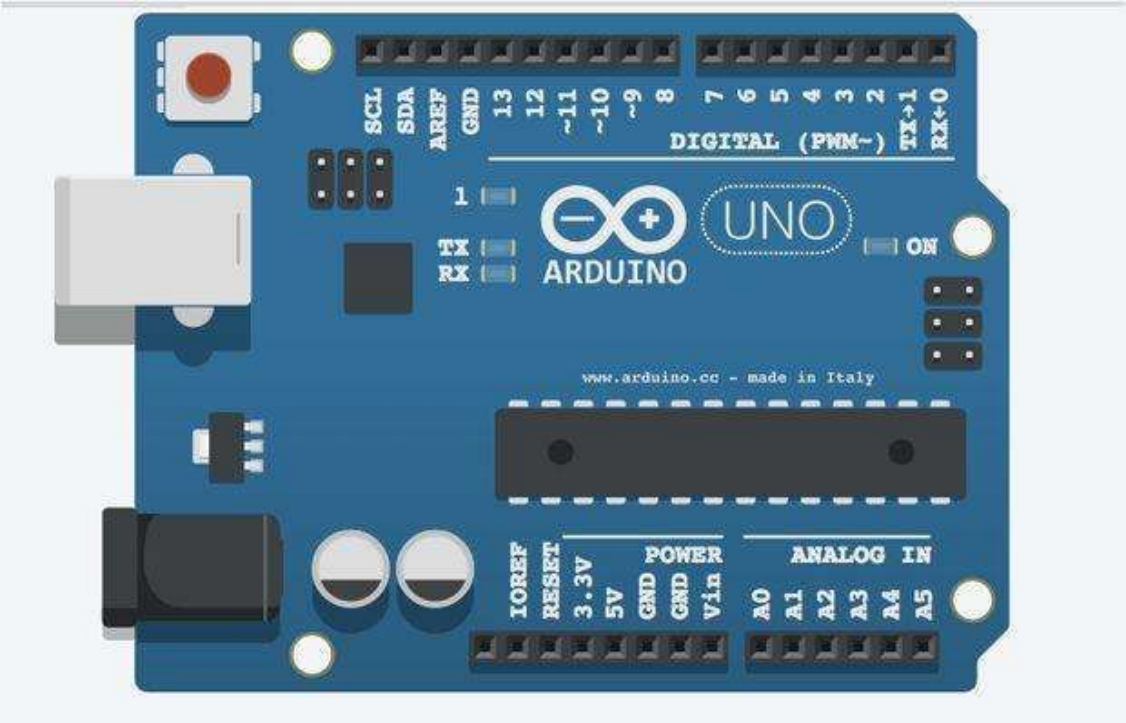
Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

IoT Platforms Overview: Arduino, Raspberry Pi

The IoT concepts imply a creation of network of various devices interacting with each other and with their environment. Interoperability and connectivity wouldn't be possible without hardware platforms that help developers solve issues such as building autonomous interactive objects or completing common infrastructure related tasks.

Let's go through the most popular IoT platforms and see how they work and benefit IoT software developers.

Arduino



The Arduino platform was created back in 2005 by the Arduino company and allows for open source prototyping and flexible software development and back-end deployment while providing significant ease of use to developers, even those with very little experience building IoT solutions.

Arduino is sensible to literally every environment by receiving source data from different external sensors and is capable to interact with other control elements over various devices, engines and drives. Arduino has a built-in micro controller that operates on the Arduino software.

Projects based on this platform can be both standalone and collaborative, i.e. realized with use of external tools and plugins. The integrated development environment (IDE) is composed of the open source code and works equally good with Mac, Linux and Windows OS. Based on a *processing* programming language, the Arduino platform seems to be created for new users and for experiments. The processing language is dedicated to visualizing and building interactive apps using animation and Java Virtual Machine (JVM) platform.

Let's note that this programming language was developed for the purpose of learning basic computer programming in a visual context. It is an absolutely free project available to every interested person. Normally, all the apps are programmed in C/C++, and are wrapped with *avr-gcc* (WinAVR in OS Windows).

Arduino offers analogue-to-digital input with a possibility of connecting light, temperature or sound sensor modules. Such sensors as *SPI* or *I2C* may also be used to cover up to 99% of these apps' market.

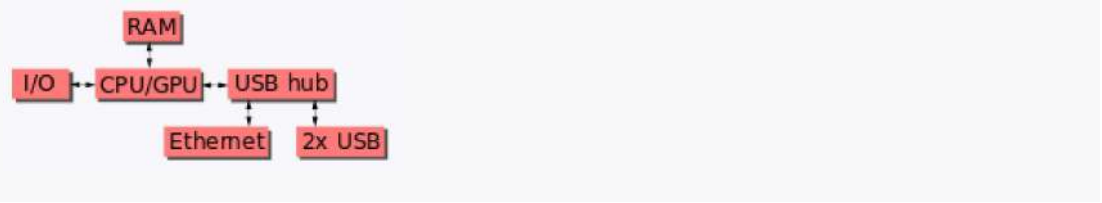
Arduino is a microcontroller (generally it is the 8-bit ATmega microcontroller), but not a mini-computer, which makes Arduino somehow limited in its features for advanced users. Arduino provides an excellent interactivity with external devices and offers a wide range of user manuals, project samples as well as a large community of users to learn from / share knowledge with.

Raspberry Pi

Raspberry Pi (/paɪ/) is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. Early on, the Raspberry Pi project leaned towards the promotion of teaching basic computer science in schools and in developing countries. Later, the original model became far more popular than anticipated,selling outside its target market for uses such as robotics. It is now widely used in many areas, such as for weather monitoring,because of its low cost, modularity, and open design.

After the release of the second board type, the Raspberry Pi Foundation set up a new entity, named Raspberry Pi Trading, and installed Eben Upton as CEO, with the responsibility of developing technology.The Foundation was rededicated as an educational charity for promoting the teaching of basic computer science in schools and developing countries.The Raspberry Pi is one of the best-selling British computers.

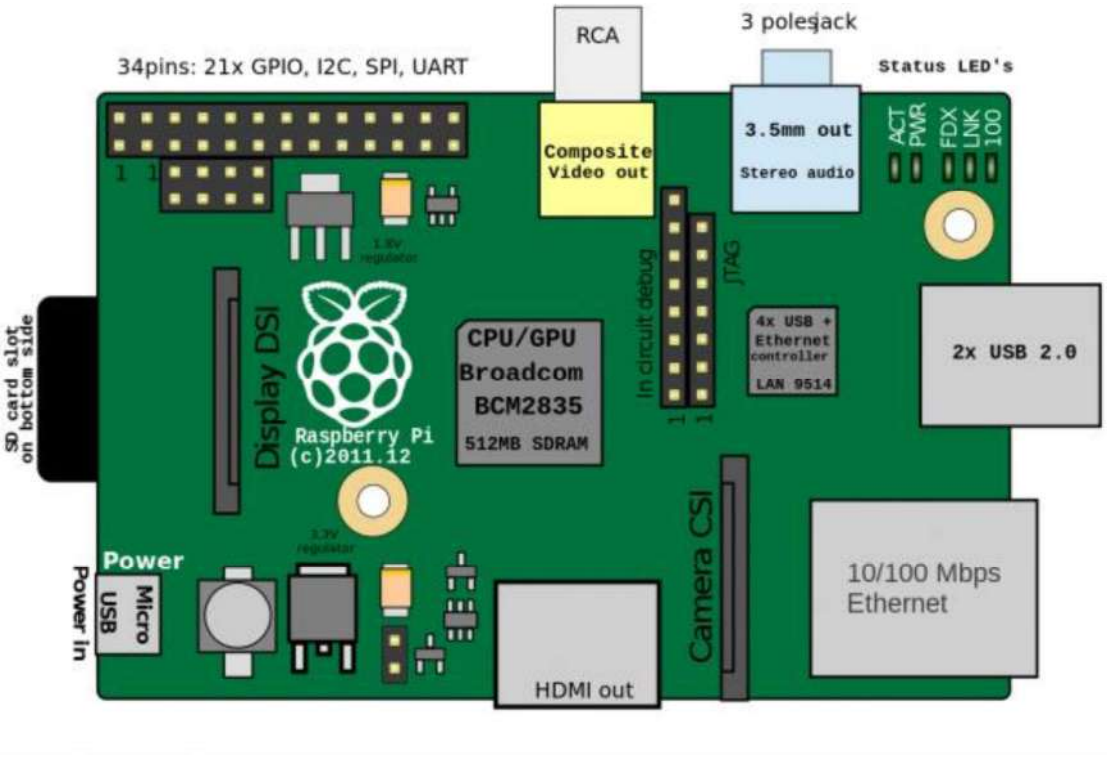
The Raspberry Pi hardware has evolved through several versions that feature variations in the type of the central processing unit, amount of memory capacity, networking support, and peripheral-device support.



This block diagram describes Model B and B+; Model A, A+, and the Pi Zero are similar, but lack the Ethernet and USB hub components. The Ethernet adapter is internally connected to an additional USB port. In Model A, A+, and the Pi Zero, the USB port is connected directly to the system on a chip (SoC). On the Pi 1 Model B+ and later models the USB/Ethernet chip contains a five-port USB hub, of which four ports are available, while the Pi 1 Model B only provides two. On the Pi Zero, the USB port is also connected directly to the SoC, but it uses a micro USB (OTG) port. Unlike all other Pi models, the 40 pin GPIO connector is omitted on the Pi Zero, with solderable through-holes only in the pin locations. The Pi Zero WH remedies this.

Processor speed ranges from 700 MHz to 1.4 GHz for the Pi 3 Model B+ or 1.5 GHz for the Pi 4; on-board memory ranges from 256 MiB to 1 GiB random-access memory (RAM), with up to 8 GiB available on the Pi 4. Secure Digital (SD) cards in MicroSDHC form factor (SDHC on early models) are used to store the operating system and program memory. The boards

have one to five USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm tip-ring-sleeve jack for audio output. Lower-level output is provided by a number of GPIO pins, which support common protocols like I²C. The B-models have an 8P8C Ethernet port and the Pi 3, Pi 4 and Pi Zero W have on-board Wi-Fi 802.11n and Bluetooth.



Raspberry Pi is a mono-board computing platform that's as tiny as a credit card. Initially it was developed for computer science education with later on progress to wider functions.

Since the inception of Raspberry, the company sold out more than 8 million items. Raspberry Pi 3 is the latest version and it is the first 64-bit computing board that also comes with built-in Wi-Fi and Bluetooth functions. According to Raspberry Pi Foundation CEO Eben Upton, "*it's been a year in the making*". The Pi3 version is replaced with a quad-core 64-bit 1.2 GHz ARM Cortex A53 chip, 1GB of RAM, VideoCore IV graphics, Bluetooth 4.1 and 802.11n Wi-Fi. The developers claim the new architecture delivers an average 50% performance improvement over the Pi 2.

Another peculiarity of Raspberry Pi is the *GPIO* (General Purpose Input-Output), which is a low-level interface of self-operated control by input-output ports. Raspberry has it as a 40-pin connector.

Raspberry Pi uses Linux as its default operating system (OS). It's also fully Android compatible. Using the system on Windows OS is enabled through any virtualization system like *XenDesktop.* If you want to develop an application for Raspberry Pi on your computer, it

is necessary to download a specific toolset comprised of ARM-compiler and some libraries complied down to ARM-target platform like *glibc.*

UNIT-V

CLOUD COMPUTING FUNDAMENTALS:

"The cloud" is a metaphor that refers to the delivery of computing services - including servers, data storage, networks, databases, analytics, intelligence, and software - over the Internet (hence "the cloud") to facilitate faster innovation, flexible resources, and economies of scale. Basically, cloud computing is an on-demand computing service that you can avail over the Internet to host and run your applications. Of course, the term "cloud" does not allude to that white, puffy, and cotton-looking thing in the sky. The physical servers are not hovering above the troposphere either. These servers are actually hosted on data centers around the world and possibly could be situated in one of the buildings in the city that you live in. Cloud computing is a type of Internet-based computer technology in which shared resources, software, and information are delivered to computers and other devices on demand.

In the past, before you could launch a website or an enterprise application, you needed to procure and set up your own physical servers first to deploy your applications. You are also responsible for managing, patching, and troubleshooting your servers and network devices. The problem here is that it takes a lot of time, effort and money just to make your solutions available online.

But with **cloud computing**, all you need to do is avail of the computing services over the Internet and the cloud service provider will be responsible for managing the underlying infrastructure that runs your websites. It's like you are 'renting' a server and after you are done using it, you have the option to end your subscription to stop accumulating unnecessary costs. This empowers you, as well other businesses, to focus on building solutions rather than spending a lot of time setting up and managing servers.

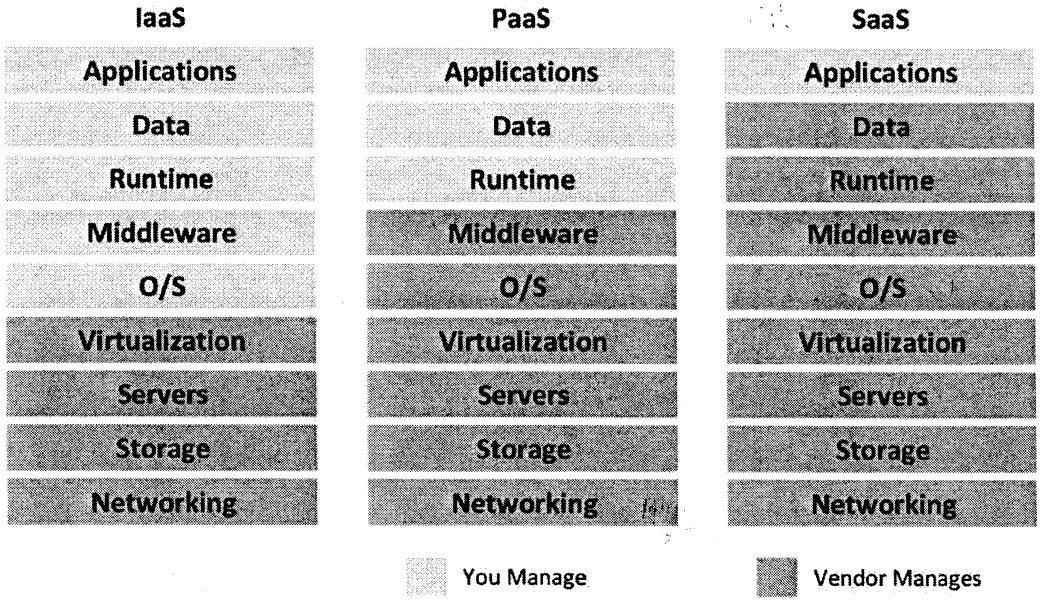
Cloud Computing provides a plethora of helpful services that small and big companies can leverage on. Its services include domain registration, Internet of Things (IoT), data analytics, machine learning, gaming, mobile development, Desktop-as-a-Service (DaaS), quantum computing and many more. This is why there are so many companies and even startups leveraging its power to launch their products faster, save on operating costs, and scale globally with ease.

Benefits of Cloud Computing

1. **Agility :** You have the ability to quickly adapt and deploy services in the most cost effective way in response to changes in business requirements.
2. **Elasticity :**In the case of varying workloads, it allows your resources to adjust quickly and to scale back and forth as the business capacity and needs change.

3. **Cost Savings :** Instead of building and managing your own data center and physical servers, you can trade your Capital Expenses for Operational Expenses and pay only the amount you consume.
4. **Deploy Globally in Minutes:** With just a few clicks, you can quickly deploy your application to different locations and enhance the experience of your users with reduced latency.
5. **Reliable:** Cloud computing authorises businesses to have data backup, disaster recovery, and data replication services.
6. **Security:** Cloud service providers strictly implements policies, controls, and technologies that strengthen data, apps, and infrastructure protection from threats.

Types of Cloud Computing: After considering the deployment model for the company's cloud environment, most cloud computing services fall into four broad categories: Often they are called the "stack" cloud computing because of how they build on top of one another.



1. **Infrastructure as a Service (IaaS) :** Full control of your infrastructure without the maintenance and operating costs of the servers. IaaS provides access to servers, storage, networking, and operating systems. IaaS is the simplest option for businesses. Businesses migrate its their hardware by rents the infrastructure and storage they need from the cloud vendor and then use that cloud infrastructure to build their application instead of purchasing and maintaining its own infrastructure. Example: Digital Ocean.
2. **Platform as a Service (PaaS) :** In this model, you can focus on the deployment and management of your applications. PaaS eliminates the need to manage the underlying infrastructure. For businesses that want to build unique apps without significant financial investments through vendors in terms of building your applications like development tools, infrastructure, operating system, PaaS is a popular choice. Example: Microsoft Azure, Heroku, Google App Engine.

   Function as a service (FaaS): Overlapping with PaaS, FaaS is also known as the severless computer. The main idea of FaaS focuses on building app functionality so

that it doen't continually spend time in managing the servers and infrastructure. It breaks down the application into seperate functions that run when triggered by some action.

3. **Software as a Service (SaaS) :** The software is ready to be used and operated by the service provider. SaaS is also known as end-user applications. SaaS is the most frequently used cloud computing infrastructure which became the dominant way to access software applications for businesses. SaaS delivers its software applications over the Internet, on demand, and applies a pay-as-you-go service. Example: Google Apps, Dropbox, Salesforce.

Cloud services are usually classified by their general use cases that make up an IT infrastructure. Some of the fundamental blocks of Cloud Computing are **Compute**, **Storage**, **Database**, **Networking**, and **Security**.

- **Compute:** Compute is the processing power required by applications and systems to process data and carry out computational tasks. Instead of provisioning your server in a local data center, you can outsource the computing power needed by your server from a cluster of virtual machines in the Cloud. The compute capacity of a virtual machine solely depends on the hardware resources attributed to its host computer. The compute resources refer to the CPU, Storage, Memory, and the Network Bandwidth available to the machine where your VM is running. Today, most cloud service providers have done a pretty good job designing a wide selection of pre-configured machines that can accommodate a variety of workloads.
- **Storage:** The main benefit of storing data in the cloud is the convenience of increasing your storage capacity without maintaining and buying more local hard drives. You cannot prevent data corruption from happening in the event of a hard disk failure. In the cloud, your data is stored persistently across logical pools in physical storage hosted by your cloud service provider. You can store different types of data such as objects, files, and backups.
- **Database:** The database is a system that stores and manages structured and unstructured information. Databases in the cloud are typically managed and offered as a service by a cloud service provider. This means that maintaining and updating the underlying components of your database instance, such as OS updates and software patches, are no longer your responsibility. Databases in the cloud are also scalable and highly available in nature.
- **Networking:** The cloud is a large ecosystem of computers that communicate and integrate with each other to deliver a specific service to customers. Cloud service providers make sure that they always maintain a high-speed network connection within their infrastructures to support the needs of their end-users. You can use the cloud to provide a global link to distribute your application all over the world.
- **Security:** In the cloud, data is stored in secured remote data center facilities. This means that threats like theft and data breach are unlikely going to happen. As a cloud user, your responsibility is more on data management. The cloud has sets of tools to

help you enforce high levels of security. For example, you have control on the encryption and decryption of your data. You can also choose to authenticate and authorize selected users and services to access your applications.

Varieties / Types of Cloud Computing: Cloud Deployment Models: Clouds are established in different models, types, and services that help companies offer the right solution depending on what the business need. The first thing to consider when determining the type of cloud deployment or cloud computing architecture that is going to be implemented in the cloud services, is the three different ways to deploy cloud services: on a public cloud, private cloud, or hybrid cloud.

- **Public Cloud:** A service run by an third-party cloud service provider that may include servers in one or multiple data centers. Using virtual machines, individual servers are shared by multiple organisations. Major hypescalers for public cloud include: Microsoft (Azure), Amazon (Amazon Web Services or AWS), Alibaba (Alicloud), and Google (Google Cloud).
- **Private Cloud:** Refers to a cloud environment in a data centre that is dedicated exclusively for companies or organisations who are concerned in sharing their resources to the public cloud. Private clouds are implemented on the servers owned, managed, and accessible by the company through the Internet or private internal network.
- **Hybrid Cloud:** Combines the public and private cloud. Hybrid cloud facilitates companies greater flexibility, more deployment options, and helps optimize your existing infrastructure, security, and compliance.

History of Cloud Computing: Let's have a quick walkthrough of cloud computing history and evolution all these years-

- **1960's :** One of the renowned names in Computer Science, John McCarthy, enabled enterprises to use expensive mainframe and introduced the whole concept of time-sharing. This turned out to be a huge contribution to the pioneering of Cloud computing concept and establishment of Internet.
- **1969:** With the vision to interconnect the global space, J.C.R. Licklider introduced the concepts of "Galactic Network" and "Intergalactic Computer Network" and also developed Advanced Research Projects Agency Network- ARPANET.
- **1970 :** By this era, it was possible to run multiple Operating Systems in isolated environment.
- **1997 :** Prof. Ramnath Chellappa introduced the concept of "Cloud Computing".
- **1999 :** Salesforce.com started the whole concept of enterprise applications through the medium of simple websites. Along with that, the services firm also covered the way to help experts deliver applications via the Internet.
- **2003:** The Virtual Machine Monitor (VMM), that allows running of multiple virtual guest operating systems on single device, a way ahead for other huge inventions.

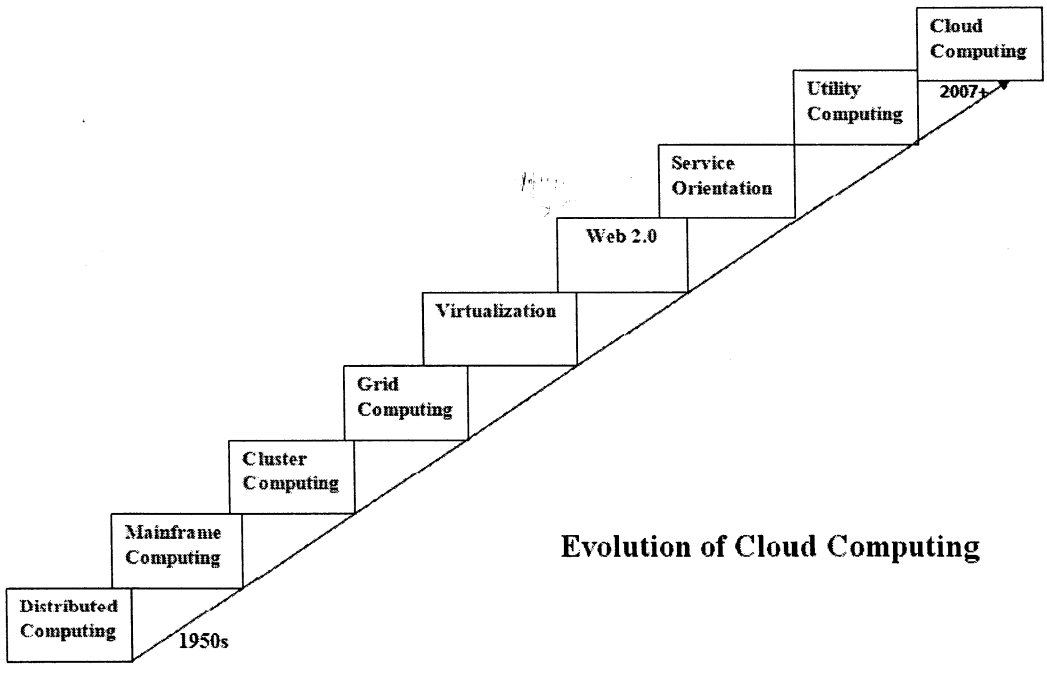
- **2006:** Amazon also started expanding in cloud services. From EC2 to Simple Storage Service S3, they introduced pay-as-you-go model, which has become a standard practice even today.
- **2013:** With IaaS, (Infrastructure-as-a-Service), the Worldwide Public Cloud Services Market was totalled at £78bn, which turned out to be the fastest growing market services of that year.

The evolution of cloud computing can be bifurcated into three basic phases:

- **The Idea Phase-** This phase incepted in the early 1960s with the emergence of utility and grid computing and lasted till pre-internet bubble era. Joseph Carl Robnett Licklider was the founder of cloud computing.
- **The Pre-cloud Phase-** The pre-cloud phase originated in 1999 and extended to 2006. In this phase the internet as the mechanism to provide Application as Service.
- **The Cloud Phase-** The much talked about real cloud phase started in the year 2007 when the classification of IaaS, PaaS, and SaaS development got formalized.

Technologies that played a vital role in the evolution of cloud computing

Cloud computing is all about renting computing services. In making cloud computing what it is today, five technologies played a vital role. These are distributed systems and its peripherals, virtualization, web 2.0, service orientation, and utility computing.



Evolution of Cloud Computing

- **Distributed Systems:** It is a composition of multiple independent systems but all of them are depicted as a single entity to the users. The purpose of distributed systems is to share resources and also use them effectively and efficiently. Distributed systems possess characteristics such as scalability, concurrency, continuous availability,

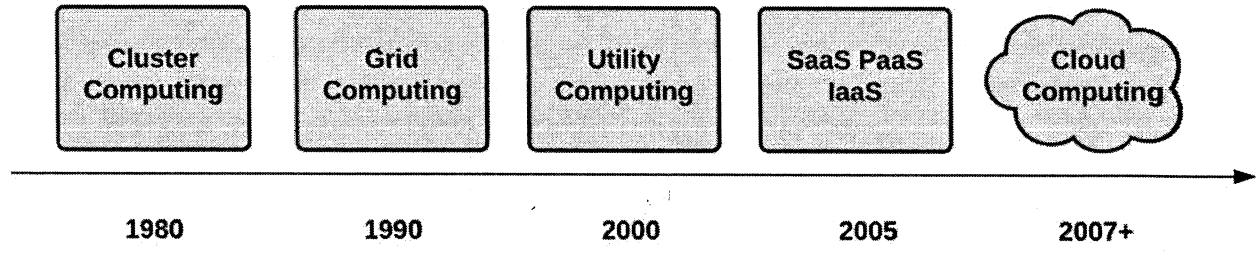
heterogeneity, and independence in failures. But the main problem with this system was that all the systems were required to be present at the same geographical location. Thus to solve this problem, distributed computing led to three more types of computing and they were-Mainframe computing, cluster computing, and grid computing.

- **Mainframe computing:** Mainframes which first came into existence in 1951 are highly powerful and reliable computing machines. These are responsible for handling large data such as massive input-output operations. Even today these are used for bulk processing tasks such as online transactions etc. These systems have almost no downtime with high fault tolerance. After distributed computing, these increased the processing capabilities of the system. But these were very expensive. To reduce this cost, cluster computing came as an alternative to mainframe technology.
- **Cluster computing:** In 1980s, cluster computing came as an alternative to mainframe computing. Each machine in the cluster was connected to each other by a network with high bandwidth. These were cheaper than those mainframe systems. These were equally capable of high computations. Also, new nodes could easily be added to the cluster if it was required. Thus, the problem of the cost was solved to some extent but the problem related to geographical restrictions still pertained. To solve this, the concept of grid computing was introduced.
- **Grid computing:** In 1990s, the concept of grid computing was introduced. It means that different systems were placed at entirely different geographical locations and these all were connected via the internet. These systems belonged to different organizations and thus the grid consisted of heterogeneous nodes. Although it solved some problems but new problems emerged as the distance between the nodes increased. The main problem which was encountered was the low availability of high bandwidth connectivity and with it other network associated issues. Thus. cloud computing is often referred to as "Successor of grid computing".
- **Virtualization:** It was introduced nearly 40 years back. It refers to the process of creating a virtual layer over the hardware which allows the user to run multiple instances simultaneously on the hardware. It is a key technology used in cloud computing. It is the base on which major cloud computing services such as Amazon EC2, VMware vCloud, etc work on. Hardware virtualization is still one of the most common types of virtualization.
- **Web 2.0:** It is the interface through which the cloud computing services interact with the clients. It is because of Web 2.0 that we have interactive and dynamic web pages. It also increases flexibility among web pages. Popular examples of web 2.0 include Google Maps, Facebook, Twitter, etc. Needless to say, social media is possible because of this technology only.
- **Service orientation:** It acts as a reference model for cloud computing. It supports low-cost, flexible, and evolvable applications. Two important concepts were introduced in this computing model. These were Quality of Service (QoS) which also includes the SLA (Service Level Agreement) and Software as a Service (SaaS).

- **Utility computing:** It is a computing model that defines service provisioning techniques for services such as compute services along with other major services such as storage, infrastructure, etc which are provisioned on a pay-per-use basis.

Thus, the above technologies contributed to the making of cloud computing. In the latest years, the development of cloud computing has resulted in numerous new technologies. Cloud computing is a computing method in which dynamically scalable and numerous virtualized resources are delivered as a service over the Internet.

Cloud computing is evolving at a breakneck pace, with businesses of all sizes adopting this new technology. According to industry experts, the evolution of cloud computing will only continue to grow and develop in the coming years. Before the advent of cloud computing, there was Client/Server computing, which is essentially centralized storage in which all software applications, data, and controls are stored on the server-side. Following that, distributed computing emerged, in which all computers are networked together and share resources as needed.



Cloud computing has recently evolved from Web2.0 technology, which caters to web applications that facilitate participatory information sharing, interoperability, and user-centered design, among other things.

The upcoming trends in cloud computing are going to empower industries with multiple cloud offerings and accelerated growth. As a result, the cloud adoption will grow by 20-30% of all enterprise IT sending. "The **pandemic** served as a catalyst for rapid cloud adoption and digital innovation in 2020, especially to empower remote work, collaboration and digitalization for hybrid work models".

BUSINESS ADVANTAGES

Since the inception of cloud computing, the limitations of traditional IT infrastructure are becoming more apparent year on year. With its proven scalability, resiliency, speed and flexibility, the cloud environments are reaching ubiquitous stages. The use of multicloud, Hybrid and edge environments are giving push to wireless communications advancements naming 5G R17 and is transforming industries experiences such as healthcare, mobile banking experiences and more.

Many businesses today are struggling to adapt to marketplace changes and new trends as their technological environments are inefficient at sensing and responding to these. Cloud-based services offer a much more scalable and reliable IT infrastructure that is specifically designed to streamline business performance and support development and growth.Some of the key advantages of cloud computing below:

Flexibility : There is a high level of flexibility provided to companies who invest in cloud-based services. Remote cloud servers offer almost unlimited bandwidth and storage space, which allows businesses to instantly scale up and down their capacities to support growth and cope when website traffic increases. This removes the need to purchase and install equipment and upgrades on site. Cloud computing also allows for improved work place flexibility, as employees can access applications and data on a remote server off-site, anywhere and at any time, as long as internet connection is available.

Business Continuity : By investing in cloud computing, businesses can guarantee reliable disaster recovery and backup solutions without the hassle of setting them up on a physical device. For many businesses investing in complex disaster recovery plans can be a costly venture and backing up data is time-consuming. The cloud itself is designed in such a way that data stored in it is mirrored across the servers, so that if one fails, data is instantly backed up. Being able to access data again quickly after a failure minimizes website downtime and loss of productivity.

Cost Efficiency : Perhaps the most significant advantage of cloud computing is the IT operational cost savings. Using remote servers removes the need for in-house storage equipment and application requirements, as well as overhead costs such as software updates, management and data storage. Cloud-based services are also much cheaper to use, as they are typically deployed on a pay-per-use basis, which means businesses can rent exactly what they need and guarantee a return on investment. Many small and medium-sized businesses with limited budgets are recognizing the benefits of cloud computing.

Improved Collaboration : The cloud environment has been shown to significantly increase collaboration between groups and communities who have access to the same files. It removes the communication limitations of traditional IT models and makes it much quicker and easier for employees working in different locations to access information and collaborate with team members and key personnel. This helps to streamline processes and means more work gets done in less time.

Scalability and Performance: Cloud technology is designed to be scaled to meet business's changing IT requirements. As a company grows, it is inevitable that more storage space and bandwidth will be required to cope with increasing traffic to the website. Cloud servers can be deployed automatically to help businesses scale up and down and ensure optimum performance under heavy loads. Cloud technology also improves website speed and minimizes downtime.

Automatic Software Updates : Many cloud service providers offer regular system updates to ensure IT requirements are consistently met. They ensure round the clock maintenance of cloud servers – including security updates – freeing up time and money that businesses spend doing this in-house.

Environmentally Friendly: For businesses with a CSR, keen to maintain a small carbon footprint, cloud-based services are very environmentally friendly. Using a pay-per-use virtual environment for data storage and running web applications means less energy consumption and carbon emissions in the workplace. Cloud computing also reduces physical hardware needs, which means less IT equipment is required in the office.

Automatic Software Integration: Another key benefit of cloud computing is software integration, which occurs automatically in the cloud. This takes away the need for businesses to manually integrate their applications. Using cloud technology, software applications and services can be quickly and easily customized, allowing businesses to handpick the services that will best suit their requirements.

Mobility : Cloud computing allows mobile access to corporate data via smartphones and devices. Staff with busy schedules, or who live a long way away from the corporate office, can use this feature to keep instantly up to date with clients and co-worker. Through the cloud, you can offer conveniently accessible information to sales staff who travel, freelance employees, or remote employees, for better work-life balance.

Insight : As we move ever further into the digital age, it's becoming clearer and clearer that the old saying "knowledge is power" has taken on the more modern and accurate form: "Data is money." Hidden within the millions of bits of data that surround your customer transactions and business process are nuggets of invaluable, actionable information just waiting to be identified and acted upon. Of course, sifting through that data to find these kernels can be very difficult, unless you have access to the right cloud-computing solution. Many cloud-based storage solutions offer integrated cloud analytics for a bird's-eye view of your data. With your information stored in the cloud, you can easily implement tracking mechanisms and build customized reports to analyze information organization wide. From those insights, you can increase efficiencies and build action plans to meet organizational goals.

Quality Control : There are few things as unfavorable to the success of a business as poor quality and inconsistent reporting. In a cloud-based system, all documents are stored in one place and in a single format. With everyone accessing the same information, you can maintain consistency in data, avoid human error, and have a clear record of any revisions or

updates. Conversely, managing information in silos can lead to employees accidentally saving different versions of documents, which leads to confusion and diluted data.

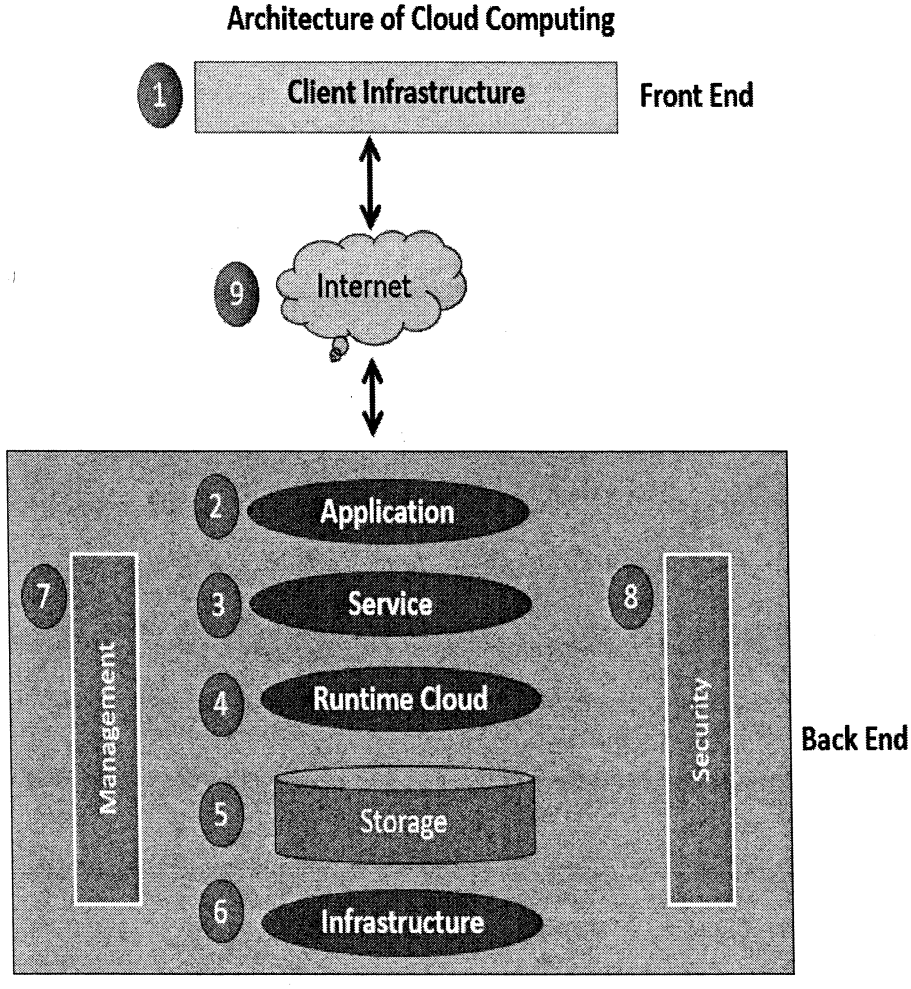
Competitive Edge : If you implement a cloud-based solution before your competitors, you'll be further along the learning curve by the time they catch up. A recent Verizon study showed that 77% of businesses feel cloud technology gives them a competitive advantage, and 16% believe this advantage is significant.

Sustainability : Given the current state of the environment, it's no longer enough for organizations to place a recycling bin in the break room and claim that they're doing their part to help the planet. Real sustainability requires solutions that address wastefulness at every level of a business. Hosting on the cloud is more environmentally friendly and results in less of a carbon footprint. Cloud infrastructures support environmental proactivity, powering virtual services rather than physical products and hardware, and cutting down on paper waste, improving energy efficiency, and (given that it allows employees access from anywhere with an internet connection) reducing commuter-related emissions.

CLOUD COMPUTING ARCHITECTURE AND ITS COMPONENTS

All kinds of businesses, let it be small scale large-scale, are shifting to cloud-based services. The adoption of both public and private clouds has increased in the past few years. The reason behind this shift is lower operating costs and increased flexibility. Cloud computing technology utilizes remote servers to store, manage, and access data on the internet rather than have a local infrastructure.

The cloud computing architecture consists of two parts front end and the back end, the front end part is the one that is used by the user, and the back end part is managed by the host. Both ends are connected to each other via the internet. The frontend includes applications and interfaces that help the user access the cloud services. The company offering cloud services manages the back end and has data storage facilities, virtual machines, security systems, and servers.



Front End : The client uses the front end, which contains a client-side interface and application. Both of these components are important to access the Cloud computing platform. The front end includes web servers (Chrome, Firefox, Opera, etc.), clients, and mobile devices.

Back End: The backend part helps you manage all the resources needed to provide Cloud computing services. This Cloud architecture part includes a security mechanism, a large amount of data storage, servers, virtual machines, traffic control mechanisms, etc.

The Architecture of Cloud computing contains many different components.

1. **Client Infrastructure:** Client Infrastructure is a front-end component that provides a GUI. It helps users to interact with the Cloud.
2. **Application:** The application can be any software or platform which a client wants to access.
3. **Service:** The service component manages which type of service you can access according to the client's requirements. Three Cloud computing services are:
    a. Software as a Service (SaaS)
    b. Platform as a Service (PaaS)
    c. Infrastructure as a Service (IaaS)
4. **Runtime Cloud:** Runtime cloud offers the execution and runtime environment to the virtual machines.
5. Storage: Storage is another important Cloud computing architecture component. It provides a large amount of storage capacity in the Cloud to store and manage data. Cloud storage is generally in the form of three basic configurations: public cloud, private cloud, and hybrid cloud.
6. **Infrastructure:** It offers services on the host level, network level, and application level. Cloud infrastructure includes hardware and software components like servers, storage, network devices, virtualization software, and various other storage resources that are needed to support the cloud computing model.
7. **Management:** This component manages components like application, service, runtime cloud, storage, infrastructure, and other security matters in the backend. It also establishes coordination between them.
8. **Security:** Security in the backend refers to implementing different security mechanisms for secure Cloud systems, resources, files, and infrastructure to the end-user.
9. **Internet:** Internet connection acts as the bridge or medium between frontend and backend. It allows you to establish the interaction and communication between the frontend and backend.

SOFTWARE-AS-A-SERVICE (SAAS),

SaaS is also known as "On-Demand Software". It is a software distribution model in which services are hosted by a cloud service provider. These services are available to end-users over the internet so, the end-users do not need to install any software on their devices to access these services. There are the following services provided by SaaS providers -

- **Business Services** - SaaS Provider provides various business services to start-up the business. The SaaS business services include ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), billing, and sales.
- **Document Management** - SaaS document management is a software application offered by a third party (SaaS providers) to create, manage, and track electronic documents. Example: Slack, Samepage, Box, and Zoho Forms.
- **Social Networks** - As we all know, social networking sites are used by the general public, so social networking service providers use SaaS for their convenience and handle the general public's information.
- **Mail Services** - To handle the unpredictable number of users and load on e-mail services, many e-mail providers offering their services using SaaS.

Advantages of SaaS cloud computing layer

1. **SaaS is easy to buy** : SaaS pricing is based on a monthly fee or annual fee subscription, so it allows organizations to access business functionality at a low cost, which is less than licensed applications. Unlike traditional software, which is sold as a licensed based with an up-front cost (and often an optional ongoing support fee), SaaS providers are generally pricing the applications using a subscription fee, most commonly a monthly or annually fee.
2. **One to Many** : SaaS services are offered as a one-to-many model means a single instance of the application is shared by multiple users.
3. **Less hardware required for SaaS** : The software is hosted remotely, so organizations do not need to invest in additional hardware.
4. **Low maintenance required for SaaS** : Software as a service removes the need for installation, set-up, and daily maintenance for the organizations. The initial set-up cost for SaaS is typically less than the enterprise software. SaaS vendors are pricing their applications based on some usage parameters, such as a number of users using the application. So SaaS does easy to monitor and automatic updates.
5. **No special software or hardware versions required** All users will have the same version of the software and typically access it through the web browser. SaaS reduces IT support costs by outsourcing hardware and software maintenance and support to the IaaS provider.
6. **Multidevice support** : SaaS services can be accessed from any device such as desktops, laptops, tablets, phones, and thin clients.
7. **API Integration** : SaaS services easily integrate with other software or services through standard APIs.

8. **No client-side installation :** SaaS services are accessed directly from the service provider using the internet connection, so do not need to require any software installation.

Disadvantages of SaaS cloud computing layer

1. **Security :** Actually, data is stored in the cloud, so security may be an issue for some users. However, cloud computing is more secure than in-house deployment.
2. **Latency issue :** Since data and applications are stored in the cloud at a variable distance from the end-user, there is a possibility that there may be greater latency when interacting with the application compared to local deployment. Therefore, the SaaS model is not suitable for applications whose demand response time is in milliseconds.
3. **Total Dependency on Internet :** Without an internet connection, most SaaS applications are not usable.
4. **Switching between SaaS vendors is difficult :** Switching SaaS vendors involves the difficult and slow task of transferring the very large data files over the internet and then converting and importing them into another SaaS also.

The below table shows some popular SaaS providers and services that are provided by them -

Provider	Services
Salseforce.com	On-demand CRM solutions
Microsoft Office 365	Online office suite
Google Apps	Gmail, Google Calendar, Docs, and sites
NetSuite	ERP, accounting, order management, CRM, Professionals Services Automation (PSA), and e-commerce applications.
GoToMeeting	Online meeting and video-conferencing software
Constant Contact	E-mail marketing, online survey, and event marketing
Oracle CRM	CRM applications
Workday, Inc	Human capital management, payroll, and financial management.

PLATFORM-AS-A-SERVICE (PAAS)

Platform as a Service (PaaS) provides a runtime environment. It allows programmers to easily create, test, run, and deploy web applications. You can purchase these applications from a cloud service provider on a pay-as-per use basis and access them using the Internet connection. In PaaS, back end scalability is managed by the cloud service provider, so end-users do not need to worry about managing the infrastructure.

PaaS includes infrastructure (servers, storage, and networking) and platform (middleware, development tools, database management systems, business intelligence, and more) to support the web application life cycle. Example: Google App Engine, Force.com, Joyent, Azure.

PaaS providers provide the following services.

- **Programming languages:** PaaS providers provide various programming languages for the developers to develop the applications. Some popular programming languages provided by PaaS providers are Java, PHP, Ruby, Perl, and Go.
- **Application frameworks:** PaaS providers provide application frameworks to easily understand the application development. Some popular application frameworks provided by PaaS providers are Node.js, Drupal, Joomla, WordPress, Spring, Play, Rack, and Zend.
- **Databases:** PaaS providers provide various databases such as ClearDB, PostgreSQL, MongoDB, and Redis to communicate with the applications.
- **Other tools:** PaaS providers provide various other tools that are required to develop, test, and deploy the applications.

There are the following advantages of PaaS -

1. **Simplified Development :** PaaS allows developers to focus on development and innovation without worrying about infrastructure management.
2. **Lower risk:** No need for up-front investment in hardware and software. Developers only need a PC and an internet connection to start building applications.
3. **Prebuilt business functionality:** Some PaaS vendors also provide already defined business functionality so that users can avoid building everything from very scratch and hence can directly start the projects only.
4. **Instant community:** PaaS vendors frequently provide online communities where the developer can get the ideas to share experiences and seek advice from others.
5. **Scalability:** Applications deployed can scale from one to thousands of users without any changes to the applications.

Disadvantages of PaaS cloud computing layer

1. **Vendor lock-in:** One has to write the applications according to the platform provided by the PaaS vendor, so the migration of an application to another PaaS vendor would be a problem.

2. **Data Privacy:** Corporate data, whether it can be critical or not, will be private, so if it is not located within the walls of the company, there can be a risk in terms of privacy of data.
3. **Integration with the rest of the systems applications:** It may happen that some applications are local, and some are in the cloud. So there will be chances of increased complexity when we want to use data which in the cloud with the local data.

The below table shows some popular PaaS providers and services that are provided by them -

Providers	Services
Google App Engine (GAE)	App Identity, URL Fetch, Cloud storage client library, Logservice
Salesforce.com	Faster implementation, Rapid scalability, CRM Services, Sales cloud, Mobile connectivity, Chatter.
Windows Azure	Compute, security, IoT, Data Storage.
AppFog	Justcloud.com, SkyDrive, GoogleDocs
Openshift	RedHat, Microsoft Azure.
Cloud Foundry from VMware	Data, Messaging, and other services.

INFRASTRUCTURE-AS-A-SERVICE (IAAS)

Iaas is also known as Hardware as a Service (HaaS). It is one of the layers of the cloud computing platform. It allows customers to outsource their IT infrastructures such as servers, networking, processing, storage, virtual machines, and other resources. Customers access these resources on the Internet using a pay-as-per use model.

In traditional hosting services, IT infrastructure was rented out for a specific period of time, with pre-determined hardware configuration. The client paid for the configuration and time, regardless of the actual use. With the help of the IaaS cloud computing platform layer, clients can dynamically scale the configuration to meet changing requirements and are billed only for the services actually used. IaaS cloud computing platform layer eliminates the need for every organization to maintain the IT infrastructure.

IaaS is offered in three models: public, private, and hybrid cloud. The private cloud implies that the infrastructure resides at the customer-premise. In the case of public cloud, it is located at the cloud computing platform vendor's data center, and the hybrid cloud is a combination of the two in which the customer selects the best of both public cloud or private cloud.

IaaS provider provides the following services -

- **Compute:** Computing as a Service includes virtual central processing units and virtual main memory for the Vms that is provisioned to the end- users.
- **Storage:** IaaS provider provides back-end storage for storing files.
- **Network:** Network as a Service (NaaS) provides networking components such as routers, switches, and bridges for the Vms.
- **Load balancers:** It provides load balancing capability at the infrastructure layer.

There are the following advantages of IaaS computing layer -

1. **Shared infrastructure:** IaaS allows multiple users to share the same physical infrastructure.
2. **Web access to the resources:** Iaas allows IT users to access resources over the internet.
3. **Pay-as-per-use model :** IaaS providers provide services based on the pay-as-per-use basis. The users are required to pay for what they have used.
4. **Focus on the core business:** IaaS providers focus on the organization's core business rather than on IT infrastructure.
5. **On-demand scalability:** On-demand scalability is one of the biggest advantages of IaaS. Using IaaS, users do not worry about to upgrade software and troubleshoot the issues related to hardware components.

Disadvantages of IaaS cloud computing layer

1. **Security :** Security is one of the biggest issues in IaaS. Most of the IaaS providers are not able to provide 100% security.
2. **Maintenance & Upgrade :** Although IaaS service providers maintain the software, but they do not upgrade the software for some organizations.
3. **Interoperability issues :** It is difficult to migrate VM from one IaaS provider to the other, so the customers might face problem related to vendor lock-in.

Some important point about IaaS cloud computing layer

- IaaS cloud computing platform cannot replace the traditional hosting method, but it provides more than that, and each resource which are used are predictable as per the usage.
- IaaS cloud computing platform may not eliminate the need for an in-house IT department. It will be needed to monitor or control the IaaS setup. IT salary expenditure might not reduce significantly, but other IT expenses can be reduced.
- Breakdowns at the IaaS cloud computing platform vendor's can bring your business to the halt stage. Assess the IaaS cloud computing platform vendor's stability and finances.
- The IaaS cloud computing platform vendor can get access to your sensitive data. So, engage with credible companies or organizations. Study their security policies and precautions.

Top Iaas Providers who are providing IaaS cloud computing platform

IaaS Vendor	Iaas Solution	Details
Amazon Web Services	Elastic, Elastic Compute Cloud (EC2) MapReduce, Route 53, Virtual Private Cloud, etc.	The cloud computing platform pioneer, Amazon offers auto scaling, cloud monitoring, and load balancing features as part of its portfolio.
Netmagic Solutions	Netmagic IaaS Cloud	Netmagic runs from data centers in Mumbai, Chennai, and Bangalore, and a virtual data center in the United States. Plans are underway to extend services to West Asia.
Rackspace	Cloud servers, cloud files, cloud sites, etc.	The cloud computing platform vendor focuses primarily on enterprise-level hosting services.
Reliance Communications	Reliance Internet Data Center	RIDC supports both traditional hosting and cloud services, with data centers in Mumbai, Bangalore, Hyderabad, and Chennai. The cloud services offered by RIDC include IaaS and SaaS.
Sify Technologies	Sify IaaS	Sify's cloud computing platform is powered by HP's converged infrastructure. The vendor offers all three types of cloud services: IaaS, PaaS, and SaaS.
Tata Communications	InstaCompute	InstaCompute is Tata Communications' IaaS offering. InstaCompute data centers are located in Hyderabad and Singapore, with operations in both countries.

MULTI-CLOUD

In cloud computing, a cloud is a collection of servers that cloud customers access over the Internet. Typically, each cloud is managed by a cloud provider – a company that offers cloud services. A public cloud is a cloud that more than one customer shares. **"Multi-cloud"** means multiple public clouds. A company that uses a multi-cloud deployment incorporates multiple public clouds from more than one cloud provider. Instead of a business using one vendor for cloud hosting, storage, and the full application stack, in a multi-cloud configuration they use several.

Multi-cloud deployments have a number of uses. A multi-cloud deployment can leverage multiple IaaS (Infrastructure-as-a-Service) vendors, or it could use a different vendor for IaaS, PaaS (Platform-as-a-Service), and SaaS (Software-as-a-Service) services. Multi-cloud can be purely for the purpose of redundancy and system backup, or it can incorporate different cloud vendors for different services.

Most businesses that move to the cloud will end up with some kind of multi-cloud deployment. A multi-cloud deployment can even come about unintentionally, as a result of shadow IT.

A multi-cloud can also be a hybrid cloud, and a hybrid cloud can also be a multi-cloud, but these terms represent two distinct concepts.

"Hybrid cloud" describes the mixing of two or more distinct types of infrastructure: it combines a private cloud, an on-premises data center, or both with at least one public cloud. Multi-cloud refers to several different public clouds being deployed, and it doesn't necessarily include a private cloud, although it can.

Pros/ Advantages:

- **Reliability and/or redundancy:** By using a multicloud deployment, a business avoids putting all their eggs in one basket. If one cloud goes down, some functionality will still be available to users from the other deployed clouds. In addition, one public cloud could be used as backup to another cloud.
- **Reduced vendor lock-in:** Moving to the cloud means relying on external cloud providers, and as companies use these vendors more and more, it can become difficult to move away from them. However, if a multi-cloud strategy is used, systems and storage are spread out across multiple vendors. Therefore it's easier to migrate away from using one of these vendors, because the majority of the infrastructure still remains in place during the migration.
- **Potential cost savings:** If a business doesn't commit to using one cloud vendor for all its infrastructure needs, it is free to pick and choose the most affordable services from different vendors.

Cons / Disadvantages:

- **Complexity of management:** A multicloud deployment means interfacing with several different vendors, each with different processes and technology. In addition, it becomes harder to have complete visibility into the technology stack with data stored and processes running in multiple clouds.
- **Increased latency:** If services in multiple clouds need to talk to one another in order to fulfill user requests, that can introduce latency, depending on how tightly the clouds are integrated, how far apart the data centers are geographically, and how often multiple clouds need to interact.
- **Greater attack surface:** The more pieces of software and hardware that are integrated, the more vulnerabilities there likely are.
- **Performance and reliability:** It can be difficult to balance loads across different clouds, especially if the data centers are very far apart geographically.

INTERCLOUD

Intercloud or 'cloud of clouds' is a common term used for in cloud computing. It's a theoretical model that combines individual clouds to create a seamless mass network. Intercloud enables a cloud to take advantage of all pre-existing resources available with other cloud providers.

The Intercloud cloud computing concept proposes that every single cloud does not have infinite physical resources or a ubiquitous geographic footprint. If a cloud saturates the computational and storage resources of its infrastructure or is requested to use resources in a geography where it has no footprint, it would still be able to satisfy such requests for service allocations sent from its clients. The Intercloud scenario would address such situations where each cloud would use the computational, storage, or any kind of resource of the infrastructures of other clouds.

This is analogous to the way the Internet works, in that a service provider, to which an endpoint is attached, will access or deliver traffic from/to source/destination addresses outside of its service area by using Internet routing protocols with other service providers with whom it has a pre-arranged exchange or peering relationship. It is also analogous to the way mobile operators implement roaming and inter-carrier interoperability.

Intercloud systems enable organizations to take complete control of their application on the cloud and use corporate data anywhere, anytime. It offers seamless connectivity and addresses common organizational issues, like security, performance, and flexibility. The extension is beneficial for companies that deliver cloud services to multiple users globally.

CLOUD COMPUTING SERVICE MANAGEMENT AND SECURITY

Cloud Service Management or CSM is worried about adjusting the two worlds, that is, the universe of cloud computing and service management and presenting great cloud management practices among supplier, consumer and customer businesses.

The expression "cloud services" refers to a wide scope of services conveyed on-request to organizations and users over the web. These services are intended to give simple, reasonable admittance to resources and applications without the requirement for hardware or infrastructure. From browsing email to teaming up on reports, most representatives use cloud services all through the workday, if they're aware of it or not. Users can get to cloud services with just a virtual private network, internet connectivity, operating system, and computer. Cloud services encourage the progression of client data from front-end customers, through the web, to the supplier's frameworks and back.

Cloud use is rapidly smoothing out and moving operational requests. Its expanded nimbleness is more expense effective and empowers capacities inaccessible in Traditional Information Technology Service Management. To meet this move, we should adjust the service delivery and give CSM. Understanding Cloud Service Management in the more extensive business setting, and results will give collaboration, guarantee elite all through the delivery, and satisfy the needs of a changing business world.

SERVICE DELIVERY

The center part of Cloud Service Management is to move service management to the executives to accomplish results requested by organizations. The modern method of IT requires a fast and powerful help to satisfy these needs. Cloud Service Management has been created to exploit the speed, adaptability, and inventive platform the Cloud provides, by utilizing robotization, fast distribution, and self-service without interruption.

Traditional IT centers around staying away from errors, while in Cloud Service Management, we understand that mistakes will happen, but instead centre around having their effect as immaterial as inconsequential be expected. To utilize the speed, anticipate and limit the effect of mix-ups, manual impedance should be diminished.

In traditional IT, there can be a lot of cycles that hinder the conveyance to the customer. In modern IT, there is greater efficacy and agility, and we can abuse this potential through Cloud Service Management.

Cloud Service Management and Operations involve all the exercises that an association does to operate, deliver, design, plan, and control the cloud and IT services that it offers to users.

Cloud Service Management incorporates the operational parts of your services and applications. After an application is shifted to production, it should be overseen. Applications are checked to guarantee accessibility and execution as per Service Level Objectives (SLOs) and Service Level Agreements (SLAs).

CHARACTERISTICS OF CLOUD SERVICE MANAGEMENT

- On-Demand Self-Service
- Resources Pooling
- Easy Maintenance
- Measured Service
- Pay as you go
- Security
- Economical
- Automatic System
- Availability
- Large Network Access

Cloud is a way of computing where versatile and flexible IT-related capacities are delivered as a service to customers utilizing web innovations. Cloud isn't characterized as a bunch of innovations yet rather a model for consuming, managing and delivering data information technology services and resources.

CASE STUDIES: AMAZON ELASTIC COMPUTE CLOUD (EC2)

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic. Amazon EC2 (Elastic Compute Cloud) is a web service interface that provides resizable compute capacity in the AWS cloud. It is designed for developers to have complete control over web-scaling and computing resources.

EC2 instances can be resized and the number of instances scaled up or down as per our requirement. These instances can be launched in one or more geographical locations or regions, and Availability Zones (AZs). Each region comprises of several AZs at distinct locations, connected by low latency networks in the same region.

EC2 Components : In AWS EC2, the users must be aware about the EC2 components, they are operating systems support, security measures, pricing structures, etc.

- **Operating System Support :** Amazon EC2 supports multiple OS in which we need to pay additional licensing fees like: Red Hat Enterprise, SUSE Enterprise and Oracle Enterprise Linux, UNIX, Windows Server, etc. These OS needs to be implemented in conjunction with Amazon Virtual Private Cloud (VPC).
- **Security :** Users have complete control over the visibility of their AWS account. In AWS EC2, the security systems allow create groups and place running instances into it as per the requirement. You can specify the groups with which other groups may communicate, as well as the groups with which IP subnets on the Internet may talk.
- **Pricing :** AWS offers a variety of pricing options, depending on the type of resources, types of applications and database. It allows the users to configure their resources and compute the charges accordingly.
- **Fault tolerance :** Amazon EC2 allows the users to access its resources to design fault-tolerant applications. EC2 also comprises geographic regions and isolated locations known as availability zones for fault tolerance and stability. It doesn't share the exact locations of regional data centers for security reasons. When the users launch an instance, they must select an AMI that's in the same region where the instance will run. Instances are distributed across multiple availability zones to provide continuous services in failures, and Elastic IP (EIPs) addresses are used to quickly map failed instance addresses to concurrent running instances in other zones to avoid delay in services.
- **Migration :** This service allows the users to move existing applications into EC2. This service suits those users having large amount of data to move.

List of some of the prominent features of EC2 –

- **Reliable** – Amazon EC2 offers a highly reliable environment where replacement of instances is rapidly possible. Service Level Agreement commitment is 99.9% availability for each Amazon EC2 region.
- **Designed for Amazon Web Services** – Amazon EC2 works fine with Amazon services like Amazon S3, Amazon RDS, Amazon DynamoDB, and Amazon SQS. It provides a complete solution for computing, query processing, and storage across a wide range of applications.
- **Secure** – Amazon EC2 works in Amazon Virtual Private Cloud to provide a secure and robust network to resources.
- **Flexible Tools** – Amazon EC2 provides the tools for developers and system administrators to build failure applications and isolate themselves from common failure situations.
- **Inexpensive** – Amazon EC2 wants us to pay only for the resources that we use. It includes multiple purchase plans such as On-Demand Instances, Reserved Instances, Spot Instances, etc. which we can choose as per our requirement.
- Virtual computing environments, known as *instances*
- Preconfigured templates for your instances, known as *Amazon Machine Images (AMIs)*, that package the bits you need for your server (including the operating system and additional software)
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as *instance types*
- Secure login information for your instances using *key pairs* (AWS stores the public key, and you store the private key in a secure place)
- Storage volumes for temporary data that's deleted when you stop, hibernate, or terminate your instance, known as *instance store volumes*
- Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as *Amazon EBS volumes*
- Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as *Regions* and *Availability Zones*
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using *security groups*
- Static IPv4 addresses for dynamic cloud computing, known as *Elastic IP addresses*
- Metadata, known as *tags*, that you can create and assign to your Amazon EC2 resources
- Virtual networks you can create that are logically isolated from the rest of the AWS Cloud, and that you can optionally connect to your own network, known as *virtual private clouds* (VPCs)

MICROSOFT AZURE.

Microsoft Azure, formerly known as Windows Azure, is a cloud computing platform, designed by Microsoft to successfully build, deploy, and manage applications and services through a global network of datacenters. It provides a range of cloud services, including compute, analytics, storage and networking. Users can pick and choose from these services to develop and scale new applications, or run existing applications in the public cloud.

The Azure platform aims to help businesses manage challenges and meet their organizational goals. It offers tools that support all industries -- including e-commerce, finance and a variety of Fortune 500 companies -- and is compatible with open source technologies. This provides users with the flexibility to use their preferred tools and technologies. In addition, Azure offers 4 different forms of cloud computing: infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS) and serverless.

Microsoft charges for Azure on a pay-as-you-go basis, meaning subscribers receive a bill each month that only charges them for the specific resources they have used.

How does Microsoft Azure work?

Once customers subscribe to Azure, they have access to all the services included in the Azure portal. Subscribers can use these services to create cloud-based resources, such as virtual machines (VM) and databases. In addition to the services that Microsoft offers through the Azure portal, a number of third-party vendors also make software directly available through Azure. The cost billed for third-party applications varies widely but may involve paying a subscription fee for the application, plus a usage fee for the infrastructure used to host the application.

Microsoft provides five different customer support options for Azure:

- Basic
- Developer
- Standard
- Professional Direct
- Premier

These customer support plans vary in terms of scope and price. Basic support is available to all Azure accounts, but Microsoft charges a fee for the other support offerings.

Because Microsoft Azure consists of numerous service offerings, its use cases are extremely diverse. Running virtual machines or containers in the cloud is one of the most popular uses for Microsoft Azure. These compute resources can host infrastructure components, such as domain name system (DNS) servers; Windows Server services -- such as Internet Information Services (IIS); or third-party applications. Microsoft also supports the use of third-party operating systems, such as Linux.

Azure is also commonly used as a platform for hosting databases in the cloud. Microsoft offers serverless relational databases such as Azure SQL and non-relational databases such as NoSQL. In addition, the platform is frequently used for backup and disaster recovery. Many organizations use Azure storage as archive in order to meet their long-term Data retention requirements.

Microsoft sorts Azure cloud services into nearly two dozen categories, including:

1. **Compute.** These services enable a user to deploy and manage VMs, containers and batch jobs, as well as support remote application access. Compute resources created within the Azure cloud can be configured with either public IP addresses or private IP addresses, depending on whether the resource needs to be accessible to the outside world.
2. **Mobile.** These products help developers build cloud applications for mobile devices, providing notification services, support for back-end tasks, tools for building application program interfaces (APIs) and the ability to couple geospatial context with data.
3. **Web.** These services support the development and deployment of web applications. They also offer features for search, content delivery, API management, notification and reporting.
4. **Storage.** This category of services provides scalable cloud storage for structured and unstructured data. It also supports big data projects, persistent storage and archival storage.
5. **Analytics.** These services provide distributed analytics and storage, as well as features for real-time analytics, big data analytics, data lakes, machine learning (ML), business intelligence (BI), internet of things (IoT) data streams and data warehousing.
6. **Networking.** This group includes virtual networks, dedicated connections and gateways, as well as services for traffic management and diagnostics, load balancing, DNS hosting and network protection against distributed denial-of-service (DDoS) attacks.
7. **Media and content delivery network (CDN).** These CDN services include on-demand streaming, digital rights protection, encoding and media playback and indexing.
8. **Integration.** These are services for server backup, site recovery and connecting private and public clouds.
9. **Identity.** These offerings ensure only authorized users can access Azure services and help protect encryption keys and other sensitive information in the cloud. Services include support for Azure Active Directory and multifactor authentication (MFA).
10. **Internet of things.** These services help users capture, monitor and analyze IoT data from sensors and other devices. Services include notifications, analytics, monitoring and support for coding and execution.
11. **DevOps.** This group provides project and collaboration tools, such as Azure DevOps -- formerly Visual Studio Team Services -- that facilitate DevOps software

development processes. It also offers features for application diagnostics, DevOps tool integrations and test labs for build tests and experimentation.

12. **Development.** These services help application developers share code, test applications and track potential issues. Azure supports a range of application programming languages, including JavaScript, Python, .NET and Node.js. Tools in this category also include support for Azure DevOps, software development kits (SDKs) and blockchain.
13. **Security.** These products provide capabilities to identify and respond to cloud security threats, as well as manage encryption keys and other sensitive assets.
14. **Artificial intelligence (AI) and machine learning.** This is a wide range of services that a developer can use to infuse artificial intelligence, machine learning and cognitive computing capabilities into applications and data sets.
15. **Containers.** These services help an enterprise create, register, orchestrate and manage huge volumes of containers in the Azure cloud, using common platforms such as Docker and Kubernetes.
16. **Databases.** This category includes Database as a Service (DBaaS) offerings for SQL and NoSQL, as well as other database instances -- such as Azure Cosmos DB and Azure Database for PostgreSQL. It also includes Azure SQL Data Warehouse support, caching and hybrid database integration and migration features. Azure SQL is the platform's flagship database service. It is a relational database that provides SQL functionality without the need for deploying a SQL server.
17. **Migration.** This suite of tools helps an organization estimate workload Migration costs and perform the actual migration of workloads from local data centers to the Azure cloud.
18. **Management and governance.** These services provide a range of backup, recovery, compliance, automation, scheduling and monitoring tools that can help a cloud administrator manage an Azure deployment.
19. **Mixed reality.** These services are designed to help developers create content for the Windows Mixed Reality environment.
20. **Blockchain.** The Azure Blockchain Service allows you to join a blockchain consortium or to create your own.
21. **Intune.** Microsoft Intune can be used to enroll user devices, thereby making it possible to push security policies and mobile apps to those devices. Mobile apps can be deployed either to groups of users or to a collection of devices. Intune also provides tools for tracking which apps are being used. A remote wipe feature allows the organization's data to be securely removed from devices without removing a user's mobile apps in the process.