

Internet of Things

Unit I

History of IoT

- 1970- The actual idea of connected devices was proposed
- 1990- 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

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

Four fundamental components of an IoT system:

1) **Sensors/Devices:** Sensors or devices are a key component that helps to collect live data from the surrounding environment. All this data may have various levels of complexities. It could be a simple temperature monitoring sensor, or it may be in the form of the video feed.

A device may have various types of sensors which performs multiple tasks apart from sensing. Example, A mobile phone is a device which has multiple sensors like GPS, camera but smartphone is not able to sense these things.

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

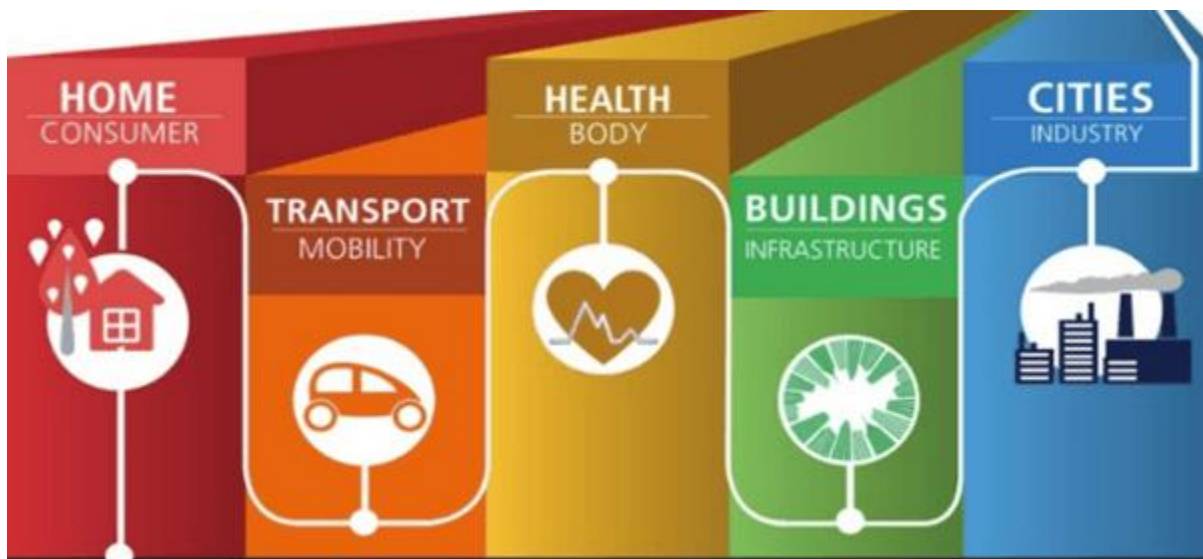
3) **Data Processing:** Once that data is collected and it gets to the cloud, the software performs processing on the gathered data. This process can be just checking the temperature, reading on devices like AC or heaters. However, it can sometimes also be very complex like identifying objects, using 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 be able to adjust the temperature with the help of their mobile phone.

IoT Applications



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

Application type	Description
Smart Thermostats	Helps to save resource on heating bills by knowing the usage patterns.
Connected Cars	IoT helps automobile companies handle billing, parking, insurance and other related stuff automatically.
Activity Trackers	Helps to capture heart rate pattern, calorie expenditure, activity levels, and skin temperature on wrist of the human body.
Smart Outlets	Remotely turn any device on or off. It also allows tracking a device's energy level and getting custom notifications directly into smartphone.
Parking Sensors	IoT technology helps users to identify the real-time availability of parking spaces on their phone.
Connect 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	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 the homes. It includes smoke detectors, home appliances, light bulbs, windows, door locks, etc.
Smart supply chain	Helps in real time tracking of goods while they are on the road, or getting suppliers to exchange inventory information.

IoT Protocols

Introduction

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

IoT protocol architecture

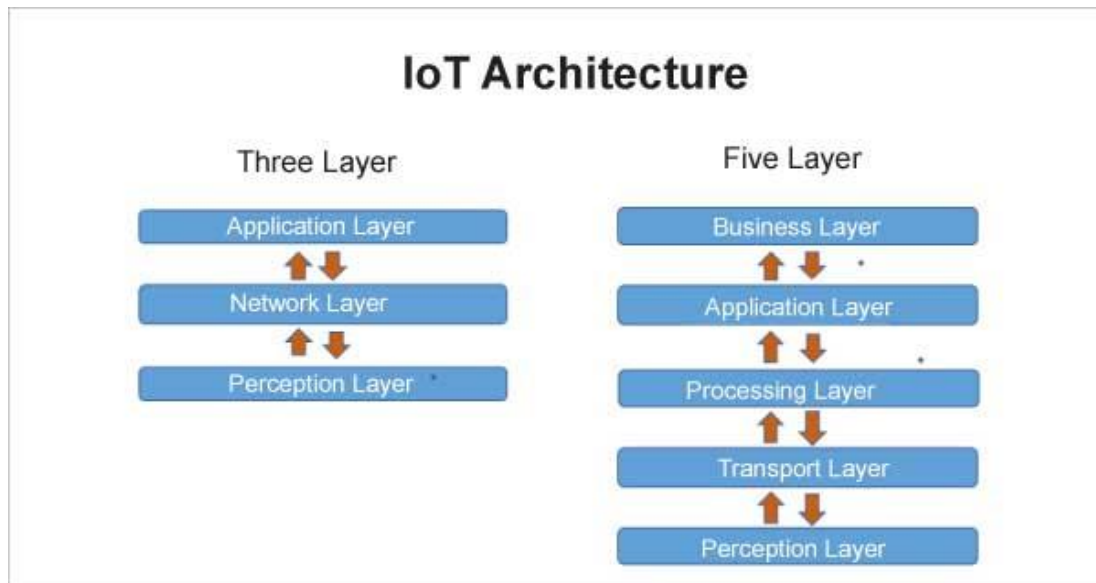
IoT architecture depends on its functionality and implementation in different sectors. The basic process flow, on which IoT is built, has two important architectures — the 3-layer architecture and 5-layer architecture.

3-layer IoT architecture: The 3-layer architecture is the most basic one. It comprises three layers, namely, the **perception, network and application layers**.

- The perception layer is the physical layer, which includes all the smart sensor-based devices that collect the data from the environment.
- The network layer includes all the wireless and the wired communication technologies, and is responsible for providing connections between the devices and the applications of the IoT ecosystem. The data is then passed on to the application layer.
- The application layer is accountable for delivering application-specific services to the user. It defines various applications in which IoT can be deployed, like smart homes, cities, and health.

5-layer IoT architecture: The 5-layer architecture is an extension of the three-layer architecture with the addition of two more layers – the **processing and business layers**. The perception and application layers work in a similar manner as in the 3-layer architecture.

- The transport layer carries the sensor data from the perception to the processing layer and vice versa using networks such as wireless, Bluetooth, 3G, RFID, and NFC.
- The processing layer or middleware layer stores, analyses, and processes large amounts of data that come from the transport layer, utilising many technologies such as databases, cloud computing, and Big Data processing modules.
- The business layer manages the whole IoT system, including applications, businesses, and user privacy.

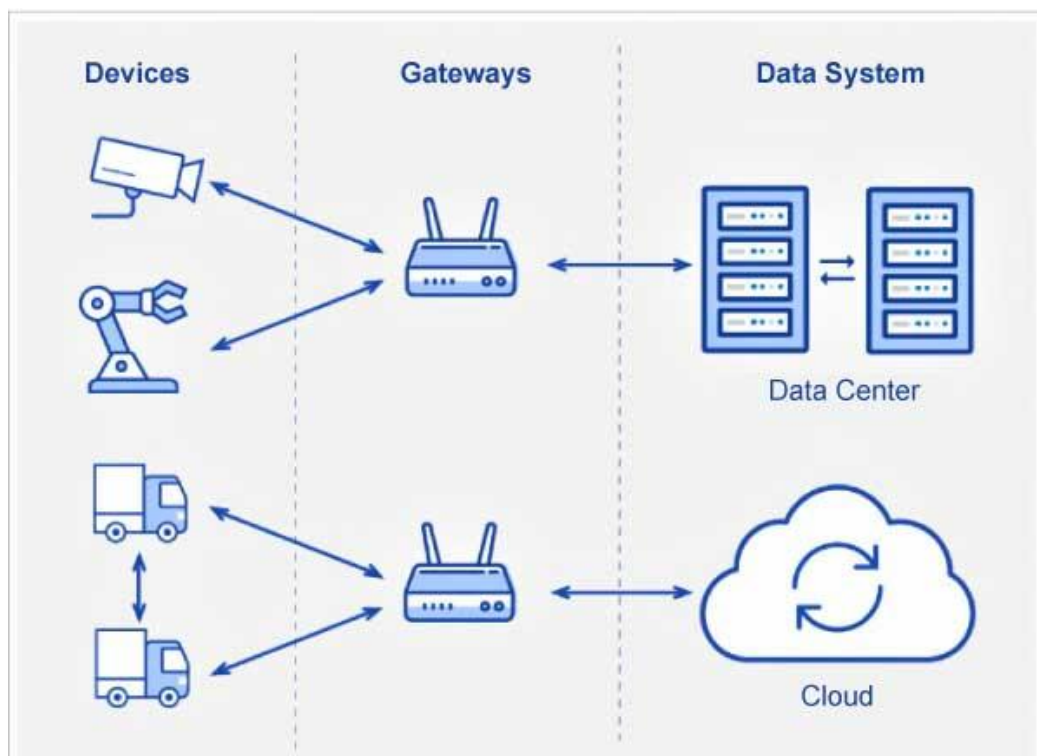


IoT architecture

Types of IoT connections

An IoT system has four types of transmission channels for data communication.

Device-to-device (D2D) communication allows the devices located in close proximity to communicate with each other using Bluetooth, ZigBee, or Z-Wave protocols. To establish a connection without a network is possible through a D2D connection.



Types of IoT connections

Device-to-gateway communication connects devices using an intermediary platform. Gateways serve two main functions – first, to consolidate data from sensors and route it to the relevant data system, and second, to analyse data and if any problems are found, return it back to the device.

Gateway-to-data systems communication is the data transmission from a gateway to the appropriate data system.

Communication between data systems is within data centres or clouds. For this type of connection, protocols should be easy to deploy and integrate with existing apps. They should have high availability, capacity, and reliable disaster recovery.



MQTT publish/subscribe architecture

Types of IoT protocols

Network layer protocols: IoT network protocols connect medium to high power devices over the network. End-to-end data communication within the network is allowed using this protocol. HTTP, LoRaWAN, Bluetooth, Zigbee are a few popular IoT network protocols.

IoT data protocols: IoT data protocols connect low power IoT devices. Without any Internet connection, these protocols can provide end-to-end communication with the hardware. Connectivity in IoT data protocols can be done via a wired or cellular network. MQTT, CoAP, AMQP, XMPP, DDS are some popular IoT data protocols.

IoT protocols and network standards

There are many IoT protocols available for different applications and requirements. However, each has its own advantages and disadvantages for different IoT scenarios. Some of the most widely used IoT protocols are discussed here.

Message queue telemetry transport (MQTT) protocol

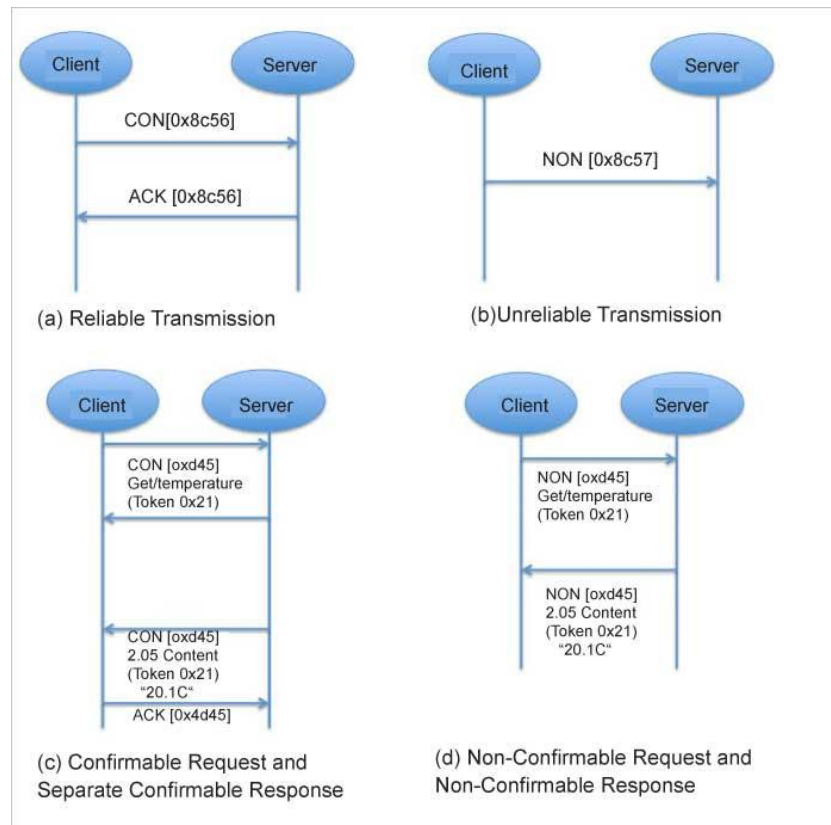
This open source publish/subscribe messaging transport protocol is very lightweight and ideal for connecting small devices to constrained networks. It was designed to work in low-bandwidth conditions, such as sensors and mobile devices, on unreliable networks. This capability makes it the most preferred protocol for connecting devices with small code footprint, as well as for wireless networks with varying levels of latency due to bandwidth constraints or unreliable connections. It works on top of transmission control protocol/Internet protocol (TCP/IP) to provide reliable delivery of data. MQTT has three main components:

- Subscriber
- Publisher
- Broker

The basic workflow of this protocol involves the publisher being responsible for generating and transmitting information to subscribers through a broker. The main function of the broker is to ensure security by checking the authorisation of subscribers and publishers. This protocol is preferred for IoT devices because it provides well-organised information routing functions to small, cheap, low memory, power devices and low bandwidth networks. To ensure message reliability, MQTT uses three levels of quality of service (QoS).

- *QoS0 (At most once)*: This is the least reliable but the quickest mode. The message is sent without any confirmation received.
- *QoS1 (At least once)*: Ensures that the message is delivered at least once, even if duplicate messages may be received.
- *QoS2 (Exactly once)*: This is the most reliable but also the most bandwidth-consuming mode. Duplicate messages are controlled to ensure that it gets delivered only once.

MQTT is a bi-directional communication protocol, where the clients can produce and consume data by publishing messages and subscribing to topics. Two-way communication enables IoT devices to send sensor data and simultaneously receive configuration information and control commands. Through MQTT, encrypting messages using TLS and verifying clients using modern authentication protocols becomes much easier.



CoAP message flows

Constrained application protocol (CoAP)

CoAP is a Web transfer protocol for constrained devices and networks in IoT. It can be implemented over a user datagram protocol (UDP), and is designed for applications with limited capacity to connect using light-weight machine-to-machine (LWM2M) communication — such as smart energy and building automation. LWM2M allows remote management of IoT devices, and provides interfaces to securely monitor and regulate them. CoAP architecture is based on the famous REST model. The architecture of CoAP is divided into two main categories: messaging, which is responsible for the reliability and duplication of messages; and request/ response, which is responsible for communication.

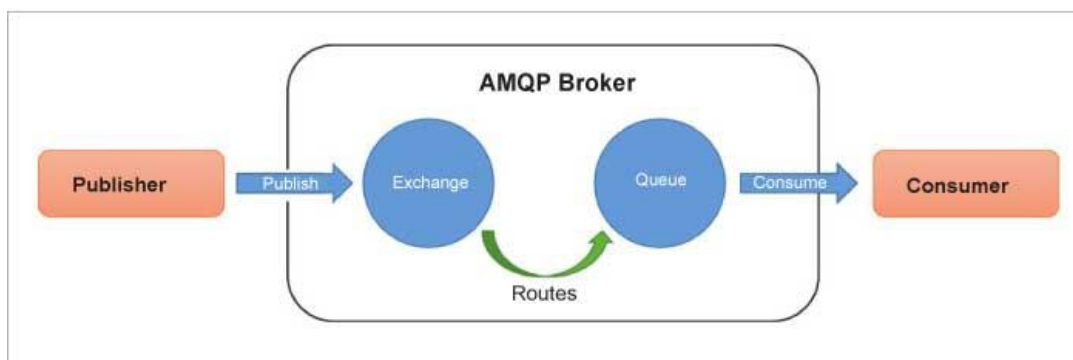
The message layer is on top of the UDP, and is responsible for exchanging messages between the IoT devices and Internet. CoAP has four different types of messages — confirmable, non-confirmable, acknowledgment and reset. A confirmable message (CON) is a reliable message when exchanged between two endpoints. It is sent over and over again until the other end sends an acknowledge message (ACK). The ACK message has the same message ID as that of a CON. If the server faces issues managing the incoming request, it can send back a reset

message (RST) instead of an ACK. For exchanging non-critical messages, unreliable NON messages can be used, where the server does not acknowledge the message. NON messages are assigned message IDs to detect duplicate messages.

Request/response is the second layer of the CoAP abstraction layer, which uses CON or NON messages to send requests. The request and the response both have the same token, different from the message ID. Once the response is ready, a new CON message containing the response is sent back to the client and the client acknowledges the response received.

Advanced message queuing protocol (AMQP)

AMQP is an open standard application layer protocol designed for higher security, reliability, easy provisioning, and interoperability. It is a connection-oriented protocol, which means the client and the broker need to establish a connection before they transfer data because TCP is used as a transport protocol. AMQP offers two levels of QoS for reliable message delivery — unsettle format (similar to MQTT QoS0) and settle format (similar to MQTT QoS1). In AMQP, the broker is divided into two main components — exchange and queues. Exchange is responsible for receiving publisher messages and sending them to queues. Subscribers connect to those queues, which basically represent the topics, and receive the sensory data whenever they are available.



AMQP architecture

Data Distribution Service (DDS)

DDS is a middleware protocol for data-centric connectivity from the object management group, which provides low-latency, data connectivity, extreme reliability, and a scalable architecture for business and mission-critical IoT applications. This protocol supports multicasting techniques in data transmission and high-quality QoS in small memory

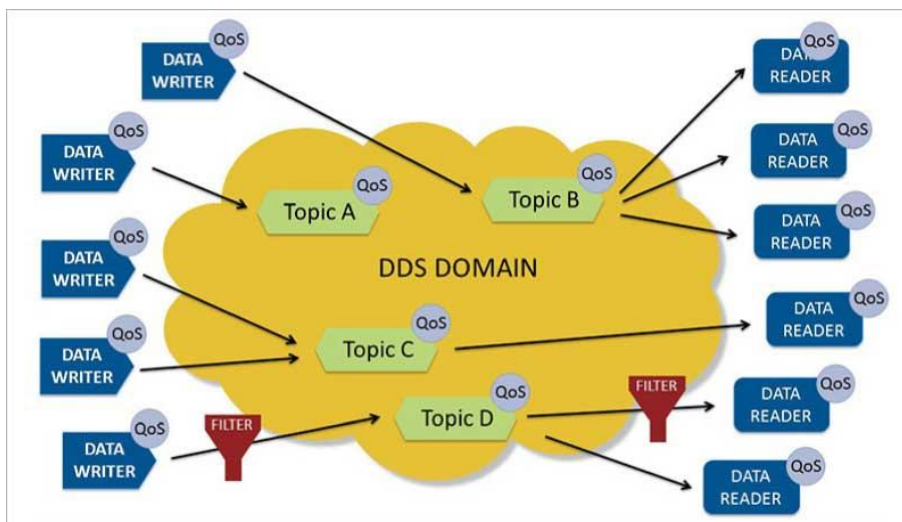
footprint devices and applications. The messaging model in DDS consists of two interface layers — data-centric, publish-subscribe (DCPS) and data local, reconstruction layer (DLRL). The DCPS layer is responsible for binding the values of data objects within an application during the publish/subscribe process.

Features	CoAP	MQTT	AMQP	DDS
Messaging pattern	request/ response	publish- subscribe	request/response; publish-subscribe	publish- subscribe
Architecture	tree	tree	star	bus
Transport	UDP	TCP, MQTT-S: UDP	TCP	UDP or TCP
Network layer	IPv6	IPv4 or IPv6	IPv4 or IPv6	IPv6
QoS level	2 levels	3 levels	3 levels	23 levels
Communication scope	Device-to- device	Device-to-cloud	Device-to-device, Device-to-cloud, Cloud-to-cloud	Device-to- device, Device-to- cloud, Cloud-to- cloud
Addressing	URI	topic only	queue, topic/Routing Key	topic/key
Security	DTLS, IPsec	TLS	SASL/TLS	TLS, DTLS, DDS security
Interoperability	Semantic	Foundational	Structural	Semantic
[Broker/server] implementation language	Java, C, C#, C++, Erlang, Go, Python, JavaScript,	Java, C, C#, C++, JavaScript, Erlang, Go, Lua, Python	Java, C, C#, C++, Python, Ruby	Ada, Java, C, C#, C++, Python, Scala, Lua,

	Ruby, Rust, Swift			Pharo, Ruby
Header size	4 bytes	2 bytes	8 bytes	16 bytes
Governing body	IETF	OASIS	OASIS	OMG

Table 1: A comparative analysis of IoT protocols

There are two main constructs in DCPS — publisher with DataWriter and subscriber with DataReader. A publisher uses the DataWriter to bind values of data objects for each defined data type. It is responsible for data distribution while adhering to the QoS policies, if any. An application uses a DataReader attached to a subscriber to retrieve the data from the latter. It subscribes to data described by a DataReader, which is provided by a known subscriber.



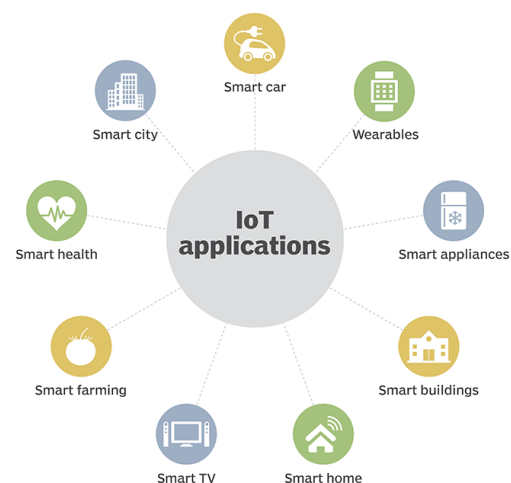
DDS data-centric model

Publishing and subscribing to data objects is done by using topics. DDS provides 23 different QoS levels with features such as security, durability, reliability, and many others.

Real time Examples of IoT

Medical: IoT devices can be used for medical data collection, monitoring and analysis. Sometimes referred to as smart healthcare, the internet of medical things aims to create a digitized healthcare system that connects medical resources and healthcare services.

Some examples IoT applications in this field include:



- Heart rate monitors and pacemakers that monitor a patient's vital functions and can send alerts through an emergency notification system.
- Advanced hearing aids that adjust their level of sensitivity to suit the user.
- Fitbit or smart watches that measure biometrics.
- Smart beds that sense when a patient is moving to alert a healthcare professional or automatically adjust settings to improve comfort.
- End-to-end health monitoring applications that help patients -- with chronic illnesses, in particular -- monitor their vitals and medication requirements.

Military: The military uses smart technology and IoT to prepare for warfare and to conduct surveillance and reconnaissance. Examples include smart drones and the DARPA (Defense Advanced Research Projects Agency) Ocean of Things project, which aims to establish a network of passive sensors at sea to record the presence and activity of military and commercial vessels.

Manufacturing: IoT in the industrial and manufacturing sector aids in various sensing, identification, processing and communication processes in the factory and elsewhere. For example, digital control systems can automate these processes and help optimize plant safety, security and efficiency.

Retail: Retailers and distributors use smart packaging with a QR code or NFC tag that contains a unique identifier with digital information about products to enable digital interactions. Similar technology has been used during the COVID-19 pandemic to enable contactless interaction with publicly used objects, such as restaurant menus or smart water fountains.

Infrastructure: IoT devices can be used to improve public infrastructure. Some examples include:

- Transportation. Smart traffic control systems, smart parking systems, electronic toll collection systems and vehicle road assistance help make transportation more efficient.
- Home and building automation. Smart energy management systems monitor and control various infrastructural components.

- Infrastructure monitoring. IoT devices can monitor infrastructure like bridges or railway tracks to detect significant structural changes to improve emergency management and incident response processes.
- Urban development. Smart cities outfitted with IoT sensors provide citizens with services like environmental monitoring data and parking applications for their smartphones by way of smart meters.
- Agriculture. IoT is used in farming to monitor and collect agricultural data such as rainfall level, temperature, wind speed, pest infestation and soil content. Farmers can use the insights from IoT devices in their fields to improve the quality of their product and minimize waste.

Advantages of IoT

Key benefits of IoT technology are as follows:

- 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.
- Improved Data Collection: Traditional data collection has its limitations and its design for passive use. IoT facilitates immediate action on data.
- 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.
- 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:

Security: IoT technology creates an ecosystem of connected devices. However, during this process, the system may offer little authentication control despite sufficient security measures.

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

Challenges of IoT technology

Despite its potential, IoT faces several challenges, including:

IoT security: IoT devices are meant to automate processes, and so humans don't interact with them as frequently as consumer devices like smartphones (itself a type of IoT device). For example, an administrator of an IoT device like a smart camera is more likely to neglect to change the default password set by the manufacturer. The result is an external-facing IoT device with a simple default password to crack.

Data privacy: There are also concerns about the data rights and privacy of consumer data as IoT becomes more prevalent. With more networked devices sharing data autonomously, being accountable for all that data becomes difficult; for example, billboards with hidden cameras that track demographic information of passers-by who stop to read it (without their knowledge or consent), and securing patient data smart medical devices collect in and out of the hospital.

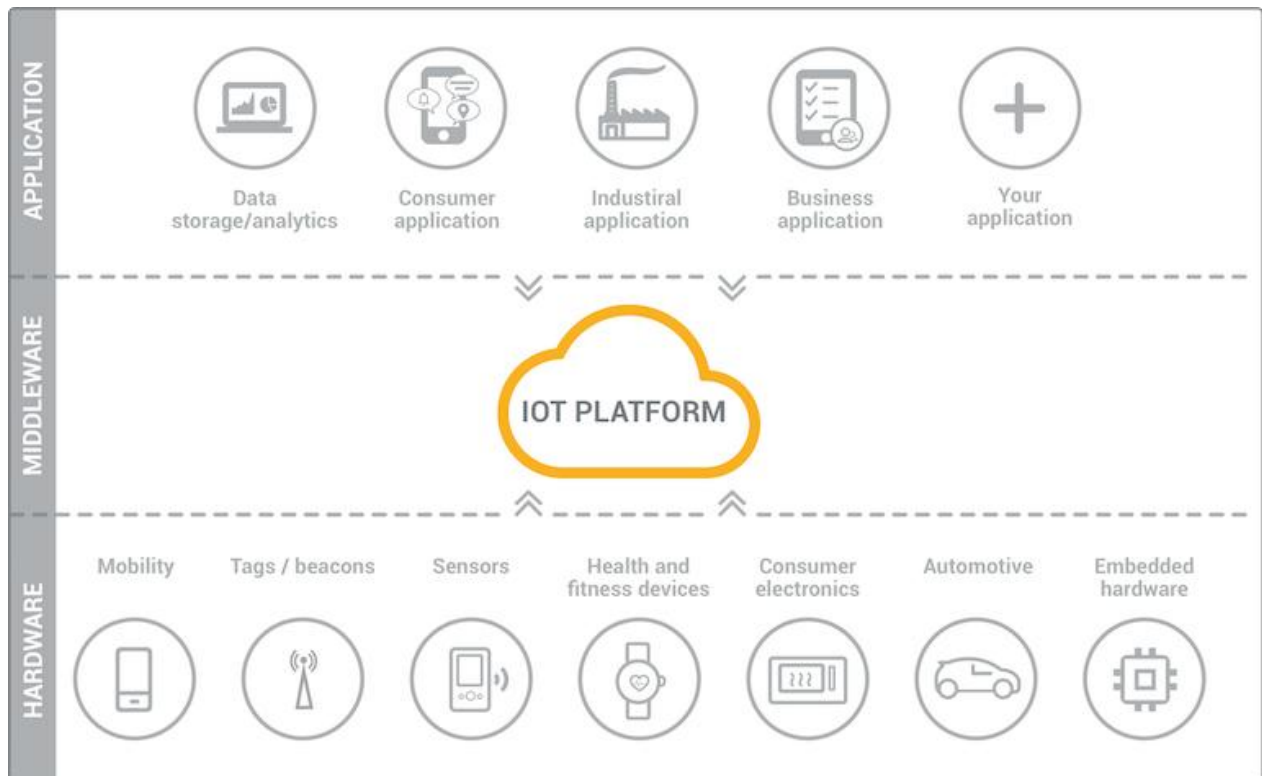
Safety: IoT devices -- especially those used in industrial, medical, transportation and infrastructural applications -- are often tasked with jobs that, if performed incorrectly, could put lives at risk. If a smart car's warning system malfunctions, it could cause the driver to neglect an obstacle or pedestrian. A malfunctioning sensor at an industrial plant can be catastrophic if a key warning sign is missed.

Interoperability: Many IoT devices have unique or niche protocols or proprietary services that they run on, and can't interact with other devices or services without considerable tweaking. There is also no universal standard set of terminology for talking about IoT, or a common set of regulations for when these devices see widespread adoption in the public sphere.

Environmental impact: Heavy metals used in many of IoT devices make it difficult to manufacture, dispose and recycle them without substantial environmental and human costs. Paired with this concern is how some IoT vendors intentionally brick (render useless) their products by disabling proprietary services that their devices need to run.

IoT - Platform

All the IoT devices are connected to other IoT devices and application to transmit and receive information using protocols. There is a gap between the IoT device and IoT application. An IoT Platform fills the gap between the devices (sensors) and application (network). **IoT platform is an integrated service that fulfils the gap between the IoT device and application and offers you to bring physical object online.**



There are several IoT Platforms available that provides facility to deploy IoT application actively. Some of them are listed below:

Amazon Web Services (AWS) IoT platform: Amazon Web Service IoT platform offers a set of services that connect to several devices and maintain the security as well. This platform collects data from connected devices and performs real-time actions.

Microsoft Azure IoT platform: Microsoft Azure IoT platform offers strong security mechanism, scalability and easy integration with systems. It uses standard protocols that support bi-directional communication between connected devices and platform. Azure IoT platform has an Azure Stream Analytics that processes a large amount of information in real-time generated by sensors. Some common features provided by this platform are:

- Information monitoring
- A rules engine
- Device shadowing
- Identity registry

Google Cloud Platform IoT: Google Cloud Platform is a global cloud platform that provides a solution for IoT devices and applications. It handles a large amount of data using Cloud IoT Core by connecting various devices. It allows to apply BigQuery analysis or to

apply Machine learning on this data. Some of the features provided by Google Cloud IoT Platform are:

- Cloud IoT Core
- Speed up IoT devices
- Cloud publisher-subscriber
- Cloud Machine Learning Engine

IBM Watson IoT platform: The IBM Watson IoT platform enables the developer to deploy the application and building IoT solutions quickly. This platform provides the following services:

- Real-time data exchange
- Device management
- Secure Communication
- Data sensor and weather data services

Artik Cloud IoT platform: Arthik cloud IoT platform is developed by Samsung to enable devices to connect to cloud services. It has a set of services that continuously connect devices to the cloud and start gathering data. It stores the incoming data from connected devices and combines this information. This platform contains a set of connectors that connect to third-party services.

Bosch IoT Suite:

Bosch cloud IoT Suit is based on Germany. It offers safe and reliable storing of data on its server in Germany. This platform supports full app development from prototype to application development.

Needs of IoT platform:

- IoT Platform connects sensors and devices.
- IoT platform handles different software communication protocol and hardware.
- IoT platform provides security and authentication for sensors and users.
- It collects, visualizes, and analyzes the data gathered by the sensor and device.

IoT Data Link Communication Protocol

The IoT Data Link communication protocol provides service to the Network Layer. There are various protocols and standard technologies specified by the different organization for data link protocols.

Bluetooth:

Bluetooth is a short-range wireless communication network over a radio frequency. Bluetooth is mostly integrated into smartphones and mobile devices. The Bluetooth communication network works within 2.4 ISM band frequencies with data rate up to 3Mbps.

There are three categories of Bluetooth technology:

1. Bluetooth Classic
2. Bluetooth Low Energy
3. Bluetooth SmartReady

The features of Bluetooth 5.0 version is introduced as Bluetooth 5 which have been developed entirely for the Internet of Things.

Properties of Bluetooth network

- **Standard:** Bluetooth 4.2
- **Frequency:** 2.4GHz
- **Range:** 50-150m
- **Data transfer rates:** 3Mbps

Advantages of Bluetooth network

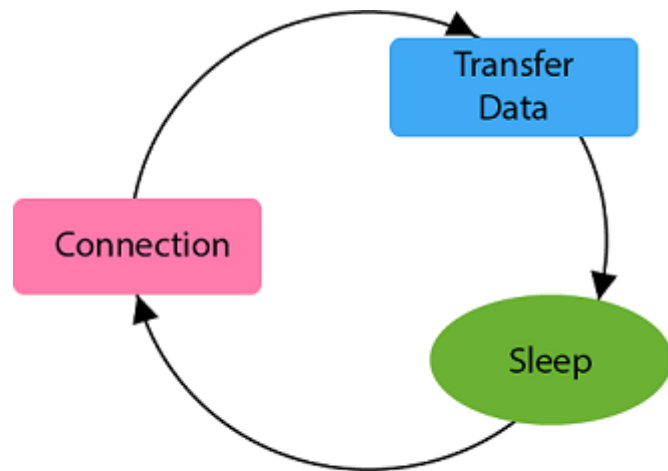
- It is wireless.
- It is cheap.
- It is easy to install.
- It is free to use if the device is installed with it.

Disadvantages of Bluetooth network

- It is a short-range communication network.
- It connects only two devices at a time.

Bluetooth Low Energy

Bluetooth low energy (BLE) is a short-range communication network protocol with PHY (physical layer) and MAC (Medium Access Control) layer. It is designed for low-power devices which uses less data. BLE always remain in sleep mode except when the connection between devices is initiated and data transmission occurs, due to this it conserves power of the device. Bluetooth low energy



BLE Technology

follows the master/slave architecture and offers two types of frames that are advertising and data frames. Slave node sent the advertising frame to discover one or more dedicated advertisement channels. Master nodes sense this advertisement channels to find slaves and connect them.

Z-Wave

Z-Wave is a wireless communication protocol with the frequency of 900MHz. The ranges of Z-Wave lies between 30 meters to 100 meters with the data transfer rate of 100kbps so that it is suitable for small messages in IoT applications for home automation. This communication protocol operates on mesh network architecture with one and several secondary controllers.



Properties of Z-Wave protocol

- **Standard:** Z-Wave Alliance ZAD12837 / ITU-T G.9959
- **Frequency:** 908.42GHz
- **Range:** 30-100m
- **Data transfer rate:** 100kbps

Advantages of Z-Wave protocol

- Low power consumption
- Remote or local control
- Simple installation
- Interoperability

Application of Z-Wave protocol

- Smart product and IoT based application
- Energy saving
- Home security

ZigBee Smart Energy

ZigBee is a low power, low data rate wireless personal area network communication protocol. It is mostly used in home automation and industrial settings. Since ZigBee is a low power communication protocol, the IoT power devices used with ZigBee technology. The ZigBee communication protocol is based on the IEEE 802.15.4 standard operating at the 2.4GHz frequency. The ZigBee protocol supports star, cluster or wireless mesh technology topology.

ZigBee uses the following devices in its network:

- Zigbee Coordinator
- Zigbee End Device
- Zigbee Router

Properties of ZigBee protocol

- **Standard:** ZigBee 3.0 based on IEEE802.15.4

- **Frequency:** 2.4GHz
- **Range:** 10-100m
- **Data transfer rate:** 250kbps

Advantages of ZigBee protocol

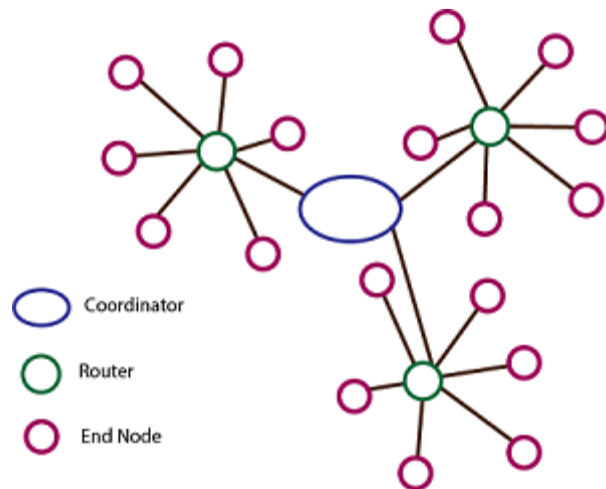
- Wireless
- Mesh networking
- Direct communication
- Low power consumption

Disadvantages of ZigBee protocol

- Costly
- Works with low speed within a small distance

Application of ZigBee protocol

- Commercial and residential control
- Personal and healthcare
- Home networking
- Industrial control and management
- Consumer electronics



LoRaWAN

LoRaWAN refers to Long Range Wide Area Network which is a wide area network protocol. It is an optimized low-power consumption protocol design to support large-scale public networks with millions of low-power devices. A single operator operates the LoRaWAN. The LoRaWAN network is a bi-directional communication for IoT application with low cost, mobility, and security.

Properties of LoRaWAN protocol

- **Standard:** LoRaWAN
- **Frequency:** Various
- **Range:** 2-5km (urban environment), 15km (suburban environment)
- **Data Rates:** 0.3-50 kbps.

Unit II - Arduino

Introduction:



Arduino is a project, open-source hardware, and software platform used to design and build electronic devices. It designs and manufactures microcontroller kits and single-board interfaces for building electronics projects.

The Arduino board consists of sets of analog and digital I/O (Input / Output) pins, which are further interfaced to **breadboard**, **expansion boards**, and other **circuits**. Universal Serial Bus (USB), and **serial communication interfaces** are used for loading programs from the computers.

Arduino Definition:

The Arduino is a single circuit board, which consists of different interfaces or parts. The board consists of the set of digital and analog pins that are used to connect various devices and components, which want to use for the functioning of the electronic devices. Most of the Arduino consists of 14 digital I/O pins.

Features

The features of Arduino are listed below:

- ✓ Arduino programming is a simplified version of C++, which makes the learning process easy.
- ✓ The Arduino IDE is used to control the functions of boards. It further sends the set of specifications to the microcontroller.
- ✓ Arduino does not need an extra board or piece to load new code.
- ✓ Arduino can read analog and digital input signals.
- ✓ The hardware and software platform is easy to use and implement.

Microcontroller



The most essential part of the Arduino is the Microcontroller. Microcontroller is small and low power computer. Most of the microcontrollers have a RAM (Random Access Memory), CPU (Central

Processing Unit), and memory storage like other computer systems.

- It has very small memory of 2KB (two Kilobytes). Due to less memory, some microcontrollers are capable of running only one program at a time.
- It is a single chip that includes memory, Input/Output (I/O) peripherals, and a processor.
- The GPIO (General Purpose Input Output) pins present on the chip help us to control other electronics or circuitry from the program.

Arduino IDE

The Arduino IDE is open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as **Windows, Mac OS X, and Linux**. It supports the programming languages C and C++. Here, IDE stands for **Integrated Development Environment**.

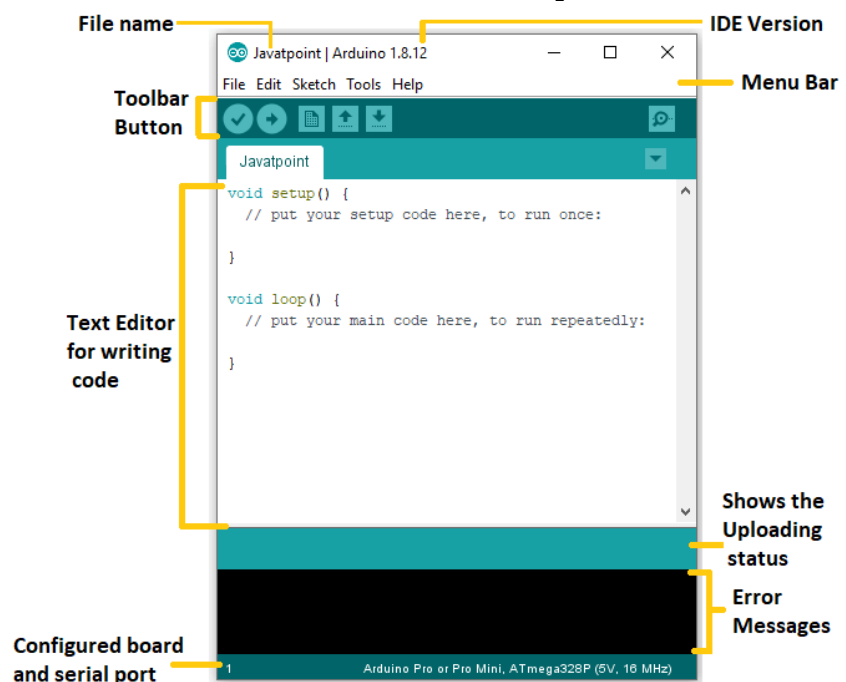


The program or code written in the Arduino IDE is often called as sketching. We need to connect the Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

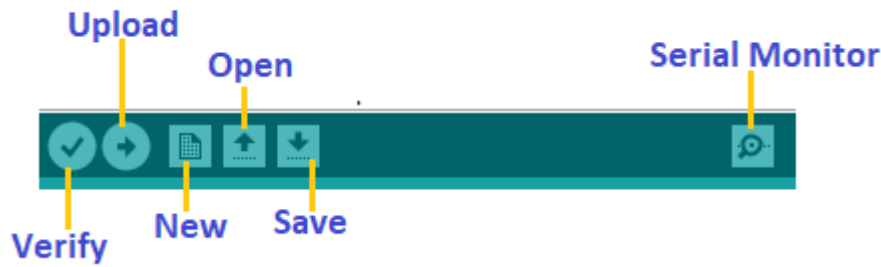
The Arduino IDE will appear as:

Toolbar Button

The icons displayed on the toolbar are **New, Open, Save, Upload,** and **Verify**.



It is shown below:



Upload

The Upload button compiles and runs our code written on the screen. It further uploads the code to the connected board. Before uploading the sketch, make sure that the correct board and ports are selected.

If the uploading is failed, it will display the message in the error window.

A **Bootloader** is defined as a small program, which is loaded in the microcontroller present on the board. The LED will blink on PIN 13.

Open: The Open button is used to open the already created file. The selected file will be opened in the current window.

Save: The save button is used to save the current sketch or code.

New: It is used to create a new sketch or opens a new window.

Verify: The Verify button is used to check the compilation error of the sketch or the written code.

Get Board Info: It gives the information about the selected board. Programmer need to select the appropriate port before getting information about the board.

Programmer: To select the hardware programmer while programming the board. It is required when programmer are not using the onboard USB serial connection. It is also required during the burning of the Bootloader.

Burn Bootloader: The Bootloader is present on the board onto the microcontroller. The option is useful when we have purchased the microcontroller without the bootloader. Before burning the bootloader, we need to make sure about the correct selected board and port.

Arduino Boards

Arduino is an easy-to-use open platform to create electronics projects. Arduino boards play a vital role in creating different projects. It makes electronics accessible to non-engineers, hobbyists, etc.

The various components present on the Arduino boards are **Microcontroller, Digital Input/output pins, USB Interface and Connector, Analog Pins, Reset Button, Power button, LED's, Crystal Oscillator, and Voltage Regulator**. Some components may differ depending on the type of board.

Types of Arduino Boards

- **Arduino UNO**

Arduino UNO is based on an ATmega328P microcontroller. It is easy to use compared to other boards, such as the Arduino Mega board, etc. The Arduino UNO includes 6 analog pin inputs, 14 digital pins, a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header.

It is the most used and of standard form from the list of all available Arduino Boards. It is also recommended for beginners as it is easy to use.



- **Arduino Nano**

The Arduino Nano is a small Arduino board based on ATmega328P or ATmega628 Microcontroller. The connectivity is the same as the Arduino UNO board.

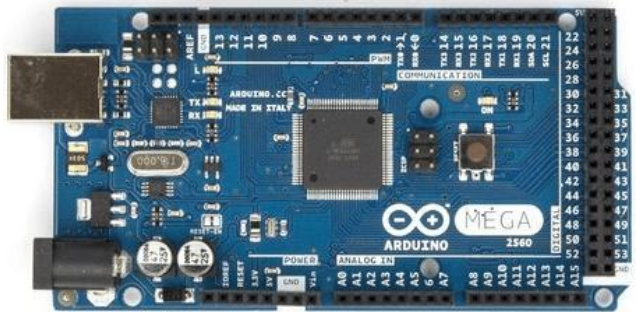


The Nano board is defined as a sustainable, small, consistent, and flexible microcontroller board. It is small in size compared to the UNO board. The devices required to start our projects using the Arduino Nano board are Arduino IDE and mini USB.

The Arduino Nano includes an I/O pin set of 14 digital pins and 8 analog pins. It also includes 6 Power pins and 2 Reset pins.

- **Arduino Mega**

The Arduino Mega is based on ATmega2560 Microcontroller. The ATmega2560 is an 8-bit microcontroller. We need a simple USB cable to connect to the computer and the AC to DC adapter or battery to get started with it. It has the advantage of working with more memory space.



The Arduino Mega includes 54 I/O digital pins and 16 Analog Input/Output (I/O), ICSP header, a reset button, 4 UART (Universal Asynchronous Receiver/Transmitter) ports, USB connection, and a power jack.

- **Arduino Micro**

The Arduino Micro is based on the ATmega32U4 Microcontroller. It consists of 20 sets of pins. The 7 pins from the set are PWM (Pulse Width Modulation) pins, while 12 pins are analog input pins. The other components on board are reset button, 16MHz crystal oscillator, ICSP header, and a micro USB connection.



- **Arduino Bluetooth**

The Arduino Bluetooth board is based on ATmega168 Microcontroller. It is also named as **Arduino BT board**. The components present on the board are 16 digital pins, 6 analog pins, reset button, 16MHz crystal oscillator, ICSP header, and screw terminals. The screw terminals are used for power.



- **Arduino Ethernet**

The Arduino Ethernet is based on the ATmega328 Microcontroller. The board consists of 6 analog pins, 14 digital I/O pins, crystal oscillator, reset button, ICSP header, a power jack, and an RJ45 connection.



Arduino UNO

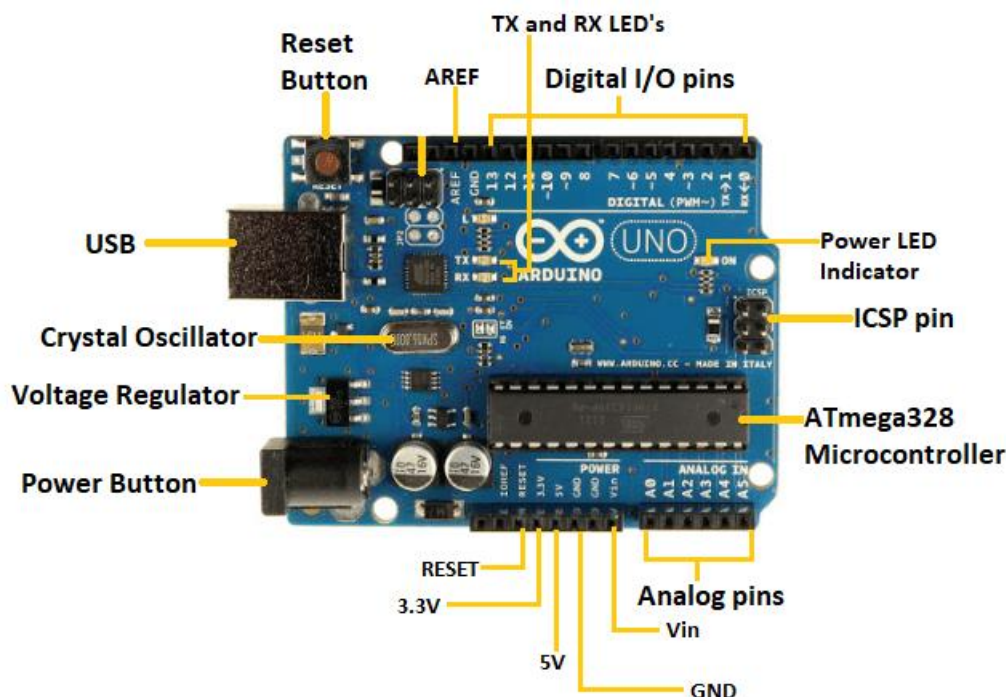
The Arduino UNO is a standard board of Arduino. Here UNO means 'one' in Italian. It was named as UNO to label the first release of Arduino Software. It was also the first USB board released by Arduino. It is considered as the powerful board used in various projects. Arduino.cc developed the Arduino UNO board.

Arduino UNO is based on an ATmega328P microcontroller. It is easy to use compared to other boards, such as the Arduino Mega board, etc. The board consists of digital and analog Input/Output pins (I/O), shields, and other circuits.

The Arduino UNO includes 6 analog pin inputs, 14 digital pins, a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header. It is programmed based on IDE, which stands for Integrated Development Environment. It can run on both online and offline platforms.

The IDE is common to all available boards of Arduino.

The components of Arduino UNO board are shown below:



Let's discuss each component in detail.

- **ATmega328 Microcontroller**- It is a single chip Microcontroller of the ATmel family. The processor code inside it is of 8-bit. It combines **Memory (SRAM, EEPROM, and Flash), Analog to Digital**

Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator.

- **ICSP pin** - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.
- **Power LED Indicator**- The ON status of LED shows the power is activated. When the power is OFF, the LED will not light up.
- **Digital I/O pins**- The digital pins have the value HIGH or LOW. The pins numbered from D0 to D13 are digital pins.
- **TX and RX LED's**- The successful flow of data is represented by the lighting of these LED's.
- **AREF**- The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- **Reset button**- It is used to add a Reset button to the connection.
- **USB**- It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.
- **Crystal Oscillator**- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- **Voltage Regulator**- The voltage regulator converts the input voltage to 5V.
- **GND**- Ground pins. The ground pin acts as a pin with zero voltage.
- **Vin**- It is the input voltage.
- **Analog Pins**- The pins numbered from A0 to A5 are analog pins. The function of Analog pins is to read the analog sensor used in the connection. It can also act as GPIO (General Purpose Input Output) pins.

Technical Specifications of Arduino UNO

The technical specifications of the Arduino UNO are listed below:

- There are 20 Input/Output pins present on the Arduino UNO board. These 20 pins include 6 PWM pins, 6 analog pins, and 8 digital I/O pins.
- The PWM pins are Pulse Width Modulation capable pins.

- The crystal oscillator present in Arduino UNO comes with a frequency of 16MHz.
- It also has a Arduino integrated WiFi module. Such Arduino UNO board is based on the Integrated WiFi ESP8266 Module and ATmega328P microcontroller.
- The input voltage of the UNO board varies from 7V to 20V.
- Arduino UNO automatically draws power from the external power supply. It can also draw power from the USB.

Arduino UNO Pinout

The Arduino UNO is a standard board of Arduino, which is based on an **ATmega328P** microcontroller. It is easier to use than other types of Arduino Boards.

The Arduino UNO Board, with the specification of pins, is shown below:

- **ATmega328 Microcontroller**- It is a single chip Microcontroller of the ATmel family. The processor core inside it is of 8-bit. It is a low-cost, low powered and a simple microcontroller. The Arduino UNO and Nano models are based on the ATmega328 Microcontroller.
- **Voltage Regulator:** The voltage regulator converts the input voltage to 5V. The primary function of voltage regulator is to regulate the voltage level in the Arduino board. For any changes in the input voltage of the regulator, the output voltage is constant and steady.
- **GND** - Ground pins. The ground pins are used to ground the circuit.
- **TXD and RXD:** TXD and RXD pins are used for serial communication. The TXD is used for transmitting the data, and RXD is used for receiving the data. It also represents the successful flow of data.
- **USB Interface:** The USB Interface is used to plug-in the USB cable. It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.
- **RESET:** It is used to add a Reset button to the connection.
- **SCK:** It stands for **Serial Clock**. These are the clock pulses, which are used to synchronize the transmission of data.
- **MISO:** It stands for **Master Input/ Slave Output**. The save line in the MISO pin is used to send the data to the master.

- **VCC** : It is the modulated DC supply voltage, which is used to regulate the IC's used in the connection. It is also called as the primary voltage for IC's present on the Arduino board.
- **Crystal Oscillator**: The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- **ICSP** : It stands for **In-Circuit Serial Programming**. The users can program the Arduino board's firmware using the ICSP pins.

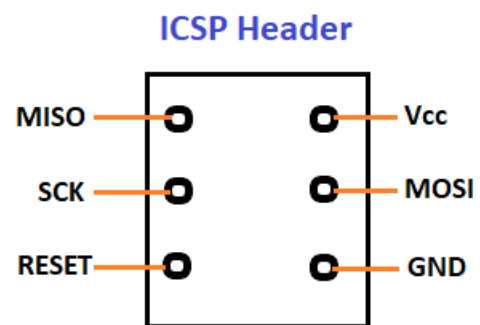
The program or firmware with the advanced functionalities is received by microcontroller with the help of the ICSP header.

The ICSP header consists of 6 pins.

The structure of the ICSP header is shown below:

It is the top view of the ICSP header.

- **SDA**: It stands for **Serial Data**. It is a line used by the slave and master to send and receive data.
- **SCL**: It stands for **Serial Clock**. It is defined as the line that carries the clock data. It is used to synchronize the transfer of data between the two devices. The Serial Clock is generated by the device and it is called as master.



- **SPI**: It stands for **Serial Peripheral Interface**. It is popularly used by the microcontrollers to communicate with one or more peripheral devices quickly. It uses conductors for data receiving, data sending, synchronization, and device selection (for communication).
- **MOSI**: It stands for Master Output/ Slave Input. The MOSI and SCK are driven by the Master.
- **SS**: It stands for **Slave Select**. It is the Slave Select line, which is used by the master. It acts as the enable line.
- **I2C**: It is the two-wire serial communication protocol. It stands for Inter Integrated Circuits. The I2C is a serial communication protocol that uses SCL (Serial Clock) and SDA (Serial Data) to receive and send data between two devices. 3.3V and 5V are the operating voltages of the board.

Arduino Coding Basics

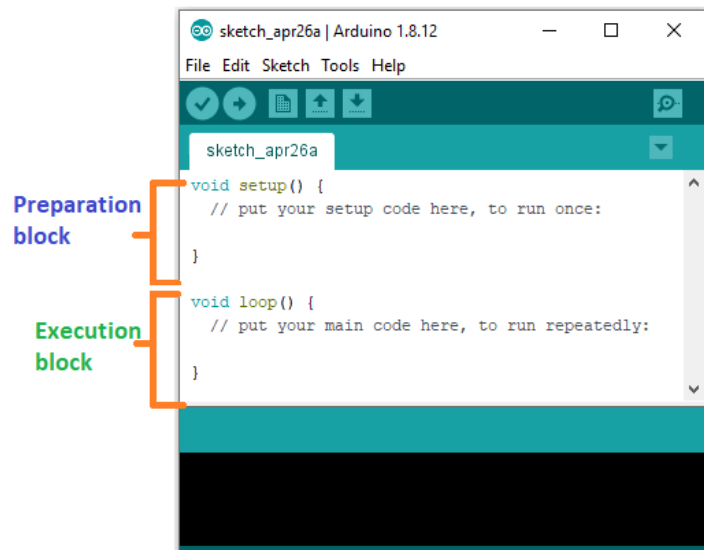
Arduino IDE (Integrated Development Environment) allows to draw the sketch and upload it to the various Arduino boards using code. The code is written in a simple programming language similar to C and C++.

The initial step to start with Arduino is the IDE download and installation.

Coding Screen

The coding screen is divided into two blocks. The **setup** is considered as the preparation block, while the **loop** is considered as the execution block.

The set of statements in the setup and loop blocks are enclosed with the curly brackets. Programmer can write multiple statements depending on the coding requirements for a particular project.



For example:

1. **void** setup ()
2. {
3. Coding statement 1;
4. Coding statement 2;
5. .
6. .
7. .
8. Coding statement n;
9. }
10. **void** loop ()
11. {
12. Coding statement 1;
13. Coding statement 2;
14. .
15. .
16. .
17. Coding statement n;

18. }

Time in Arduino

The time in Arduino programming is measured in a millisecond.

Where, 1 sec = 1000 milliseconds

One can adjust the timing according to the milliseconds.

For example, for a 5-second delay, the time displayed will be 5000 milliseconds.

- The void setup () would include pinMode as the main function.

pinMode (): The specific pin number is set as the INPUT or OUTPUT in the pinMode () function.

The Syntax is: **pinMode (pin, mode)**

Where,

pin: It is the pin number. We can select the pin number according to the requirements.

Mode: We can set the mode as INPUT or OUTPUT according to the corresponding pin number.

Let' understand the pinMode with an example.

Example: pinMode (12, OUTPUT);

digitalWrite(): The digitalWrite () function is used to set the value of a pin as HIGH or LOW.

Where,

HIGH: It sets the value of the voltage. For the 5V board, it will set the value of 5V, while for 3.3V, it will set the value of 3.3V.

LOW: It sets the value = 0 (GND).

The syntax is: **digitalWrite(pin, value HIGH/LOW)**

pin: specify the pin number or the declared variable.

Let's understand with an example.

Example:

1. digitalWrite (13, HIGH);

2. `digitalWrite (13, LOW);`

The HIGH will ON the LED and LOW will OFF the LED connected to pin number 13.

difference between `digitalRead ()` and `digitalWrite ()`

The `digitalRead ()` function will read the HIGH/LOW value from the digital pin, and the `digitalWrite ()` function is used to set the HIGH/LOW value of the digital pin.

`delay ()`

The `delay ()` function is a blocking function to pause a program from doing a task during the specified duration in milliseconds.

For example, - `delay (2000)`

Where, 1 sec = 1000millisecond

Hence, it will provide a delay of 2 seconds.

Example: To light the LED connected to pin number 13. We want to ON the LED for 4 seconds and OFF the LED for 1.5 seconds.

Code:

```
1. void setup ()
2. {
3.   pinMode (13, OUTPUT); // to set the OUTPUT mode of pin number 13.
4. }
5. void loop ()
6. {
7.   digitalWrite (13, HIGH);
8.   delay (4000); // 4 seconds = 4 x 1000 milliseconds
9.   digitalWrite (13, LOW);
10.  delay (1500); // 1.5 seconds = 1.5 x 1000 milliseconds
11. }
```

Blinking an LED

LED (Light Emitting Diode) is an electronic device, which emits light when the current passes through its terminals. The LED will work as a simple light that can be turned ON and OFF for a specified duration.

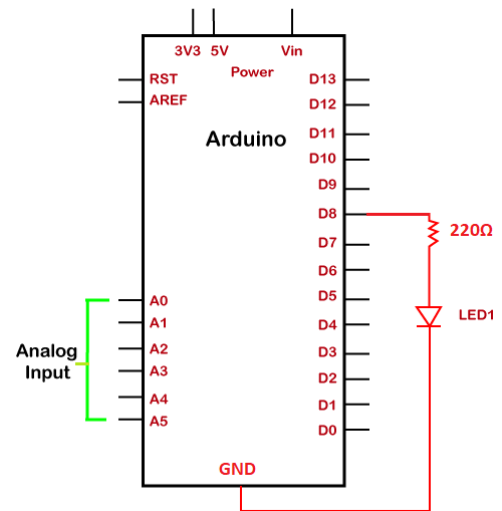
The structure clearly shows the pinout of the UNO board. It also displays the LED and resistance connected to the board. It is shown below:

Open the IDE and start with the coding, which is given below:

```

1. void setup ()
2. {
3.   pinMode ( 8, OUTPUT);
// to set the OUTPUT mode of pin number 8.
4. }
5. void loop ()
6. {
7.   digitalWrite (8, HIGH);
8.   delay(1000); // 1 second = 1 x 1000 milliseconds
9.   digitalWrite (8, LOW);
10.  delay(500); // 0.5 second = 0.5 x 1000 milliseconds
11. }

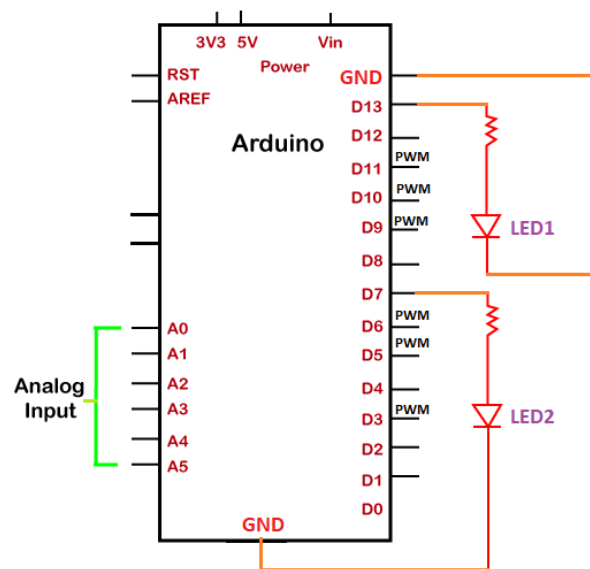
```



Blinking Two LED

The concept of blinking two LED's is similar to the blinking of a single LED. The resistors reduce the amount of current reaching the LED, which saves the LED from being burnt.

Here, The digital output pin number 13 and 7 are used. The positive terminal of the red LED is connected to the PIN 13, and the negative terminal (anode) is connected to the ground.



Similarly, the positive terminal (cathode) of the green LED is connected to PIN 7 and the negative terminal is connected to the ground.

Sketch

```

1. void setup ()
2. {

```

```

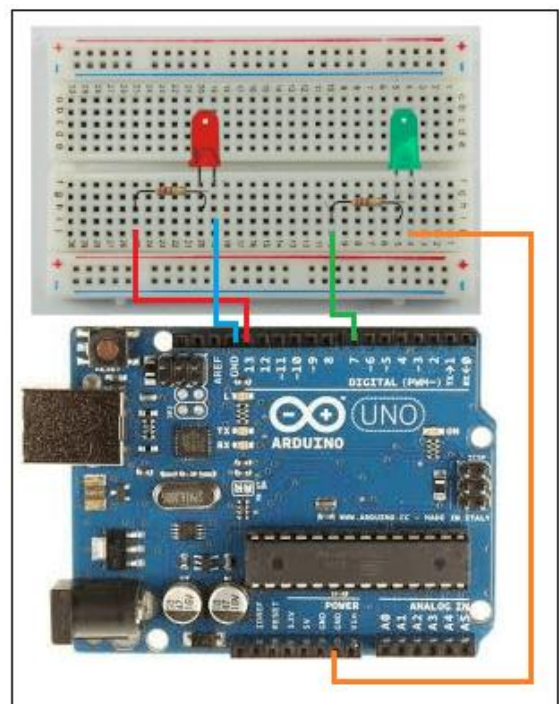
3. pinMode ( 13, OUTPUT); // to set the OUTPUT mode of pin number 13.
4. pinMode ( 7, OUTPUT); // to set the OUTPUT mode of pin number 7.
5. }
6. void loop ()
7. {
8. digitalWrite (13, HIGH);
9. digitalWrite (7, LOW);
10. delay(1500); // 1.5 second = 1.5 x 1000 milliseconds
11. digitalWrite (13, LOW);
12. digitalWrite (7, HIGH);
13. delay(1000); // 1 second = 1 x 1000 milliseconds
14. }

```

Click on the Verify button present on the toolbar to compile the code.

The RX and TX LED on the board will light up after the successful uploading of the code.

- Connect the left leg of the resistor (connected in series with **red** LED) to the digital output pin of the UNO board, i.e., PIN **13**.
- Connect the left leg of the resistor (connected in series with **green** LED) to the digital output pin of the UNO board, i.e., PIN **7**.
- Connect the negative/shorter terminal (Cathode) of the red and green LED to the **GND** pin of the UNO board using the wire.



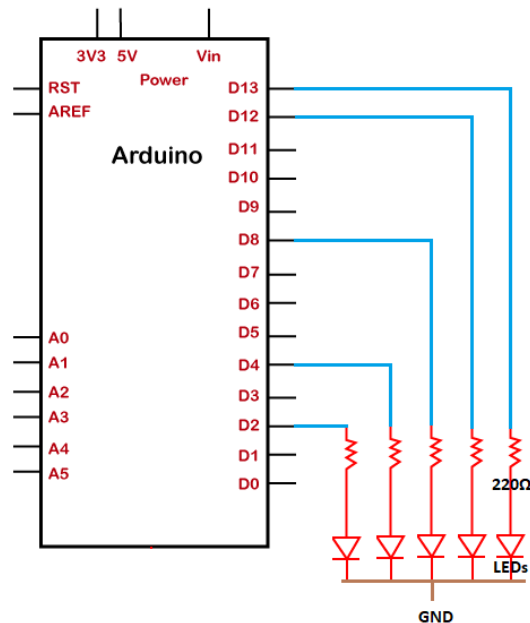
Blinking various LEDs using Arrays

Five LEDs are connected to pins 13, 12, 8, 4, and 2 of the Arduino board. The required resistance of the resistors is enough to light up an LED without damaging the board and other components.

- Connect the resistor of 220 Ohm in series with the five LEDs. Now connect it to the pin number 13, 12, 8, 4, and 2 of the Arduino board.
- Connect the negative terminal of the five LEDs to the GND (Ground).

Sketch

The code to light the five LEDs is given below:



```
1. int timer = 500;
2. int LEDPins[] = {13, 12, 8, 4, 2};    // an array of declared pin numbers on the board
3. int countOfpin = 5;                  // the number of arrays
4. void setup()
5. {
6.    // we have declared an array to initialize the LED pins as OUTPUT
7.    for (int PIN = 0; PIN < countOfpin; PIN= PIN + 1)
8.    {
9.        pinMode(LEDPins[PIN], OUTPUT);
10.    }
11. }
12. void loop()
13. {
14.    // loop starting from the lowest pin in the array to the highest:
15.    for (int PIN = 0; PIN < countOfpin; PIN++) {
16.        // turns the pin ON:
17.        digitalWrite(LEDPins[PIN], HIGH);
18.        delay(timer);
19.        // turns the pin OFF:
20.        digitalWrite(LEDPins[PIN], LOW);
21.    }
22.    // loop from the highest pin in the array to the lowest:
```



```

23.      // It means the LEDs will light in the reverse direction as used
        above
24.      for (int PIN = countOFpin - 1; PIN >= 0; PIN- -)
25.      {
26.          digitalWrite(LEDpins[PIN], HIGH);
27.          delay(timer);
28.          digitalWrite(LEDpins[PIN], LOW);
29.      }
30.  }

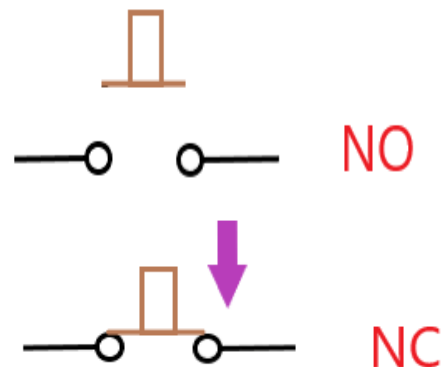
```

Arduino button

The buttons are similar to switches that create and break electrical connections in the circuits. A single press turns the state ON, while another press turns the state OFF. It means that the button connects the two points in a circuit.

There are two types of button, which are listed below:

- **NO (Normally Open)** : In this type, the state of the button is in rest. It means that a terminal in such a condition is not connected. When push the button, the terminals become electrically connected.
- **NC (Normally Closed)** : It is defined as the working state of the button. It connects the terminals of the circuit and allows current to flow through the load.



NC and NO are also defined as the momentary type of switches.

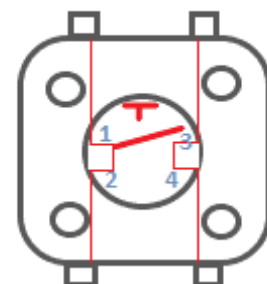
Pushbutton

Structure of pushbutton

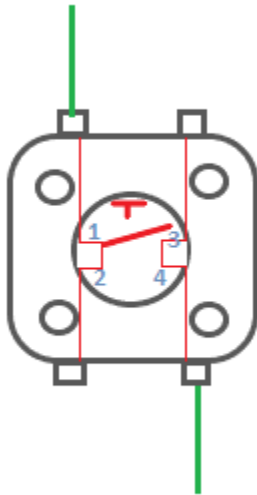
Let's understand the structure of pushbutton.

The pushbutton is a square shape button with four terminals, as shown below:

The two pins are next to each other on one side and another two pins on the other side. The pins across



to each other are connected. The pins next to each other can only be connected, when we press the button.



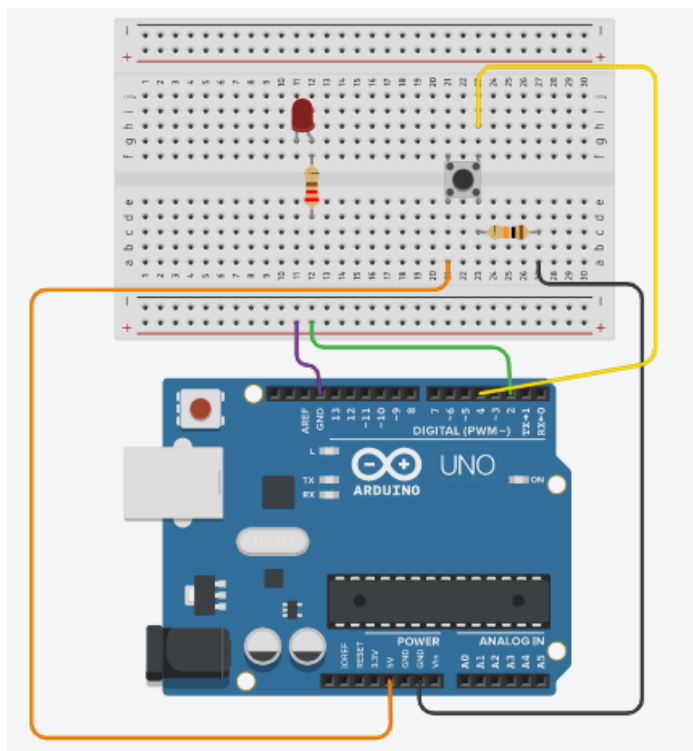
connect two opposite terminals of the pushbutton, as shown below:

Let's understand buttons with an example.

The steps for such an example are listed below:

1. Attach the red LED on the breadboard board.
2. Connect a resistor in series with the LED and connect it to PIN 2 of the breadboard.
3. Connect the negative terminal of the LED to the GND pin.
4. Attach the pushbutton on the breadboard.
5. Connect a 10 kohm resistor in series with the lower right corner of the pushbutton and connect it to the GND pin.
6. Connect the upper right corner of the pushbutton to PIN 4.
7. Connect lower left corner of the pushbutton to 5V.

The circuit is shown below:



Code

The code for the upper circuit is shown below:

```
1. const int ledpin = 2; // initializing pin number 2 to the LED
2. const int buttonpin = 4; // initializing pin number 4 to the button
3. int buttonState = 0;
4. void setup()
5. {
6.   Serial.begin(9600);
7.   pinMode(buttonpin, INPUT);
8.   pinMode(ledpin, OUTPUT);
9. }
10.    void loop()
11.    {
12.        // read the state of the pushbutton value
13.        buttonState = digitalRead(buttonpin);
14.        // check if pushbutton is pressed.  if it is, the
15.        // buttonState is HIGH
16.        if (buttonState == HIGH) {
17.            // turn LED on
18.            digitalWrite(ledpin, HIGH);
19.            Serial.println("LED is ON");
20.            //When we press the button, it will print LED is ON.
21.            delay ( 500);
22.        }
23.        else
24.        {
25.            // turn LED off
26.            digitalWrite(ledpin, LOW);
27.            Serial.println("LED is OFF"); // When we press the button, it
            will print LED is OFF.
28.        }
29.        delay ( 500);
30.    }
```

Arduino PWM

The PWM (**Pulse Width Modulation**) is a method of controlling the average voltage. It is a **stream of voltage pulses** that reduces the electric power supplied by the electrical signal. The effective voltage is controlled by the width of individual pulses in a stream of voltage pulses of a PWM signal.

The common use of PWM pins includes controlling **LEDs** and **DC Motors**.

The PWM in LED controls the frequency of the light. It means the LED will be ON/OFF at a frequency detectable by our eyes.

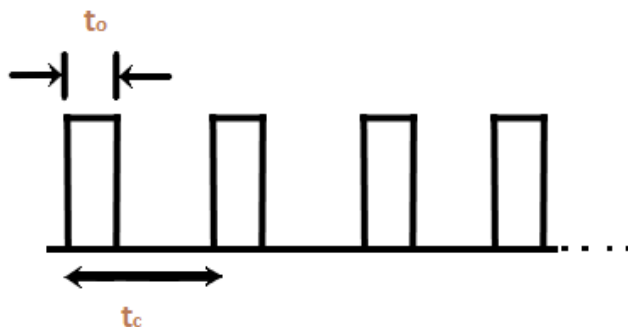
The PWM in DC Motors acts like a pulse train of a DC signal. The DC motors receive a high or low electrical power input based on the width of the PWM pulses.

The application of PWM is **voltage regulation, audio signal generation, devices control** (pump, hydraulics, etc.), **servo motor**, etc.

Principle of PWM

The state of the Digital Input/Output pins in Arduino is either HIGH (1) or LOW (0). Here, HIGH means the voltage is approx to 5V. LOW means the voltage is equivalent to 0 volts.

The PWM is a square wave signal, which is represented as:



The duty cycle of the rectangular pulse is shown below:

$$\text{duty cycle} = \frac{t_o}{t_c}$$

Here,

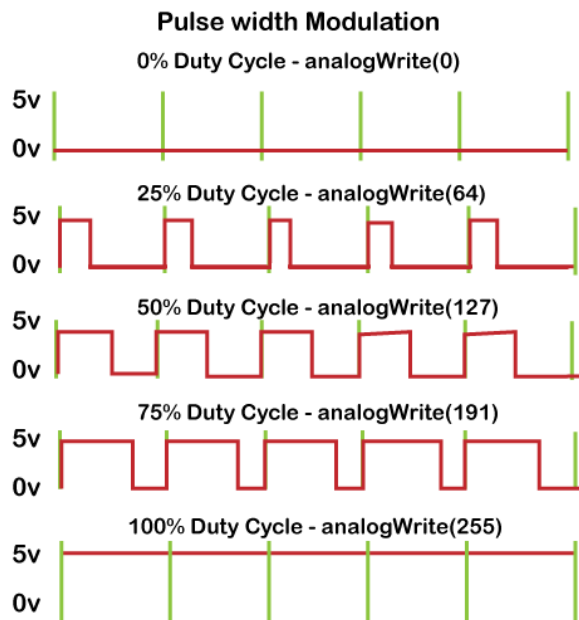
t_o: It is the duration of the signal when the signal is HIGH.

t_c: It is the total duration of the signal as the sum of HIGH and LOW.

Duty cycle of a PWM wave

As defined above, the duty cycle is the ratio of the pulse width to the total width of a signal.

Consider the below image:



The above image displays the wave at different duty cycles.

Control the effective voltage of the DC motor in Arduino by regulating the PWM duty cycle.

For example,

Arduino UNO

Arduino UNO board consists of 14 digital Input/Output pins, where pin 11, 10, 9, 6, 5, and 3 are PWM pins. The `pinMode()`, `digitalRead()`, `digitalWrite()` functions control the operation of non-PWM pins.

The **`pinMode()`** function is used to declare the specific pin as input/output. The `digitalRead` is used to read the HIGH or LOW state of a pin.

Programmer need to use the `analogWrite()` to set the duty cycle of a PWM (Pulse Width Modulation) pulse.

Let's discuss `analogWrite()` in detail.

`analogWrite()` : It writes a PWM value or analog value to a pin. Programmer can light an LED with varying brightness with the help of `analogWrite()`. It is also used to drive a motor at varying speeds.

When an `analogWrite()` function is called, a stable rectangular wave of particular duty cycle is generated by the specified PWM pin until the next `analogWrite()` is called on that same pin.

For example,

The PWM pins on the Arduino Leonardo/Micro are 3, 5, 6, 9, 10, 11, and 13. The frequency on pin 3 and 11 will be 980Hz, while other PWM pins have 490Hz of frequency.

The syntax is:

1. `analogWrite(pin, value)`

where,

pin: Specified PWM pin on the board

value: It determines the value of the duty cycle between 0 and 255.

The data type used here is **int**.

Note: The `analogWrite()` function is not related to the `analogRead()` or analog pins.

What is the difference between `analogRead()` and `analogWrite()`?

The main differences between `analogRead()` and `analogWrite()` are listed below:

- The `analogRead()` is used to read the analog value, while `analogWrite()` function is used to write the PWM value.
- The value of `analogRead()` ranges from 0 to 1023, while `analogWrite()` ranges from 0 to 255.

Let's understand with an example.

Consider the below code:

1. **void** setup()
2. {
3. `pinMode(10, OUTPUT);` // the declared pin must be among the PWM pins.
4. }
5. **void** loop()
6. {
7. `analogWrite(10, 255);` // 255 is the highest value.
8. // modify the value as per the required brightness.

```

9.  delay(1000);
10. analogWrite(10, 0);
11. delay(1000); // time delay of 1 second or 1000 milliseconds
12. }

```

Here, the LED will light at full brightness.

Let's discuss an example to control the brightness of the LED.

calculate Arduino PWM

The `analogWrite()` function discussed above is used to generate a PWM signal in Arduino.

The value associated with the analog signal is from 0 to 255. It means 256 levels of values.

The maximum voltage read by the Arduino is 5V.

PWM voltage = (Duty cycle/ 256) x 5V



Code Example

Let's discuss a method to control the brightness of an LED connected to the PWM pin.

Here, we have connected the LED to the PWM pin 6.

Consider the below code.

```

1. void setup()
2. {
3.   pinMode(6, OUTPUT); // the declared pin must be among the PWM p
   ins.
4. }
5. void loop()
6. {
7.   analogWrite(6, 255); // brightness increases as value increases
8.   delay(1000);
9.   analogWrite(6, 180); // brightness level

```



```
10.      delay(1000);
11.      analogWrite(6, 80);
12.      delay(1000);
13.      analogWrite(6, 20); // brightness decreases as value decreases
14.      delay(1000);
15.      }
```

In the above example, the brightness of the LED will decrease according to the specified value of brightness.

Arduino Library

The Library is considered as the advanced feature, which extends the capabilities of the Arduino IDE. It means that the libraries provide extra functionality to the programming platform of Arduino.

The libraries in Arduino are written in [C](#) or [C++](#) (.cpp). These libraries allow us to manipulate data and work with the hardware. To implement any Library in the [Arduino IDE](#), go to the **Sketch -> Import Library**. There are several libraries available for download. Programmer can also create our own library. Let's discuss some of the libraries.

Standard Libraries

The standard libraries are listed below:

EEPROM

It stands for **Electronic Erasable Programmable Read Only Memory**. The EEPROM is associated with the microcontroller present on the AVR or Arduino Boards. The EEPROM library allows us to read the bytes stored in the memory when the power of the board is off.

The size of EEPROM varies in different boards, such as 1KB or 1024 bytes on the ATmega328P. It can also be of 4KB or 4096 bytes on the Microcontroller ATmega2560, etc.

The library is declared as:

```
1. #include <EEPROM.h>
```

For example, EEPROM Read, EEPROM Clear, EEPROM Write, EEPROM Get, EEPROM Update, EEPROM Put, EEPROM Iteration, etc.

Ethernet Library

The Ethernet library works with the Arduino Ethernet shield and other related devices. The Ethernet library allows us to connect the Arduino board to the Internet.

The SPI bus acts as an intermediate between the board and the shield.

The associated library is:

1. `#include <Ethernet.h>`
2. `#include <SPI.h>`

For example, **TelnetClient**, **WebServer**, **WebClientRepeating**, **WebClient**, **ChatServer**, **DnsWebClient**, **UdpNtpClient**, **UdpSendReceiveString**, etc.

Firmata Library

For the programming environment, Programmer can create custom firmware without producing their own objects and protocols. It is used to implement the firmware protocol, which communicates with the software on the host computer.

The associated library is:

1. `#include <Firmata.h>`

GSM Library

The GSM library exists on the IDE version 1.0.4 and up. The GSM library allows us to perform the operations on the Arduino board similar to the GSM phone, such as internet connection, send and receive messages, and to place voice calls.

The library is declared as:

1. `#include <GSM.h>`

Liquid Crystal Library

It is a library that permits Arduino to communicate with LCDs, which are based on a compatible chipset called Hitachi HD44780. Such chipsets are found on most types of text-based LCDs. It works with either an 8-bit mode or 4-bit mode. Here, the bit mode signifies the data lines in addition to the enable, rs, and rw control lines (optional).

The library is declared as:

1. `#include <LiquidCrystal.h>`

The examples are **Hello World**, **Cursor**, **Blink**, etc.

SD Library

It allows writing to or reading from SD cards. For example, Arduino Ethernet Shield. The file names include the paths separated by the forward slashes, which are passed to the SD Library. But, SPI is used for the communication between the SD card and the Arduino.

The library is declared as:

1. `#include <SPI.h>`
2. `#include <SD.h>`

The examples are Dump files, List Files, Read Write, etc.

Servo Library

The Servo library permits Arduino to work with servo motors. It allows controlling the integrated shaft and gears. Programmer can also position shaft at different angles between 0 and 180 degrees. The servo library on Arduino boards can support upto 12 motors, while on Arduino Mega board, it can support upto 48 motors.

The library is declared as:

1. `#include <Servo.h>`

SPI Library

The SPI (**S**erial **P**eripheral **I**nterface) is a serial data protocol. The microcontrollers use the serial protocol to communicate over short distances with one or more peripheral devices quickly. The required connection of SPI is a full-duplex that allows devices to simultaneously sent and receive data.

The library is declared as:

1. `#include <SPI.h>`

The examples are Dump files, List Files, Read Write, etc.

Stepper Library

The Stepper library in Arduino permits to control of bipolar or unipolar stepper motors.

The library is declared as:

1. `#include <Stepper.h>`

The Stepper includes stepper speed control, stepper one revolution, etc.

WiFi Library

The WiFi library permits Arduino to establish a connection with the internet. It can either be a server to receive the incoming connections or a client to perform outgoing connections.

The personal encryptions supported by the WiFi library are WPA2 and WEP except for WPA2 Enterprise. Arduino uses the SPI bus to communicate with the WiFi shield.

The library is declared as:

1. `#include <WiFi.h>`

The examples include WiFiWebClient, WiFiWebServer, etc.

Audio Library

The Audio library is compatible with the Arduino Due board only. It enables the board to playback .wav files from the specific storage devices, such as the SD card.

It plays sounds by using the DAC0 and DAC1 pins.

The library is declared as:

1. `#include <Audio.h>`

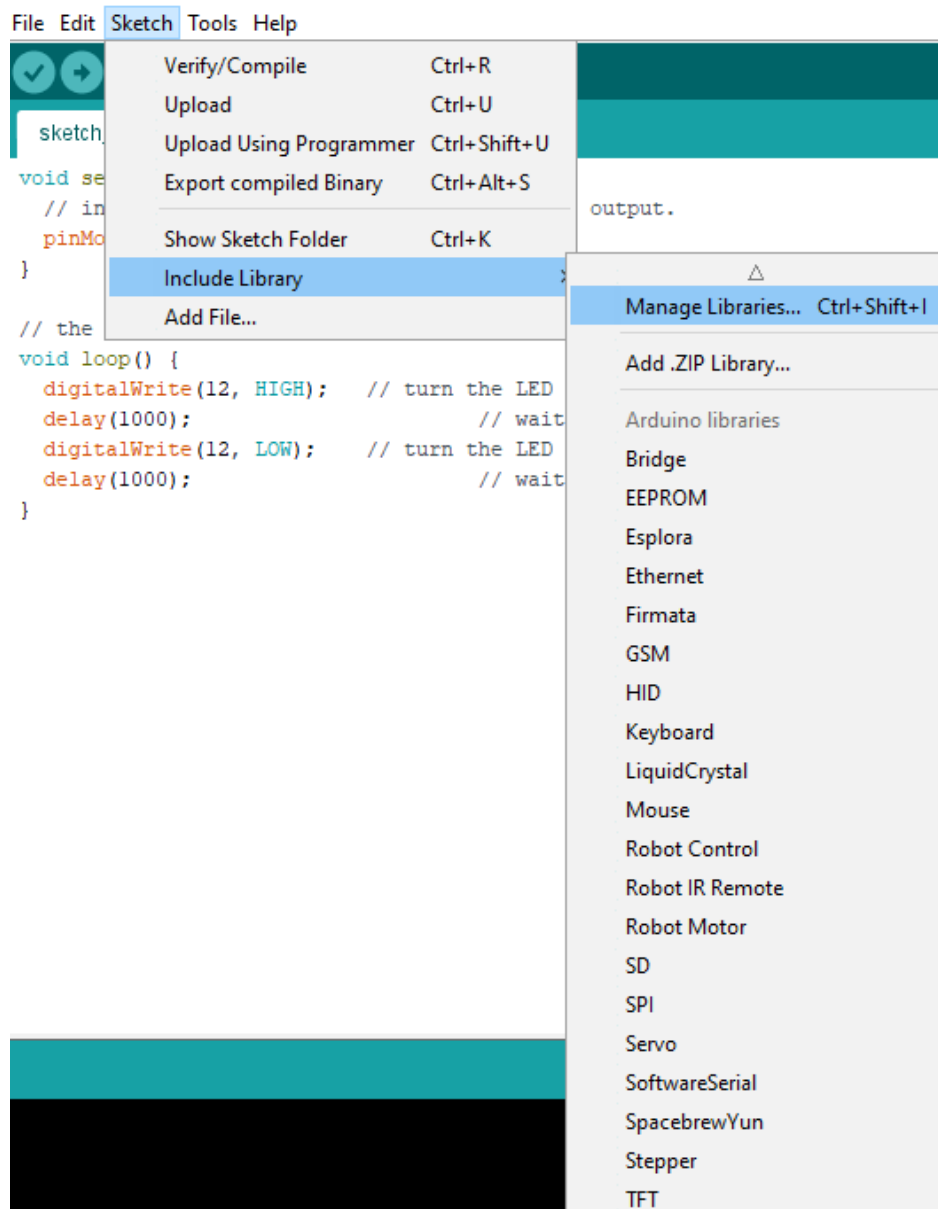
The example is a Simple Audio Player.

Steps involve to install a library in Arduino

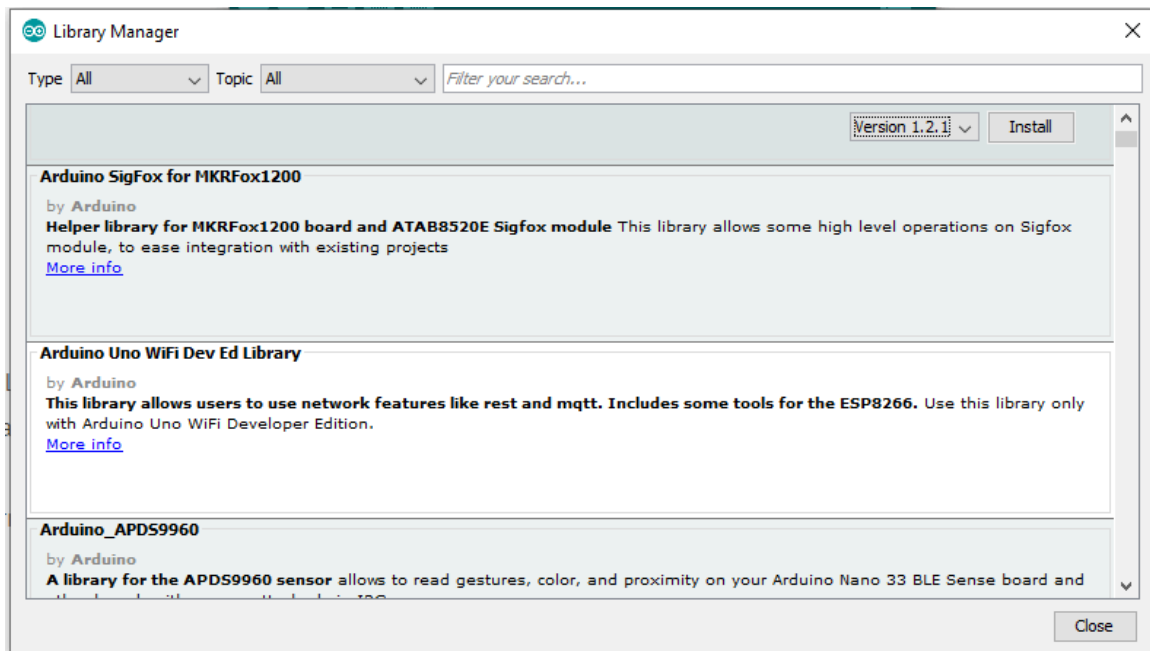
The steps are listed below:

Arduino Library Manager

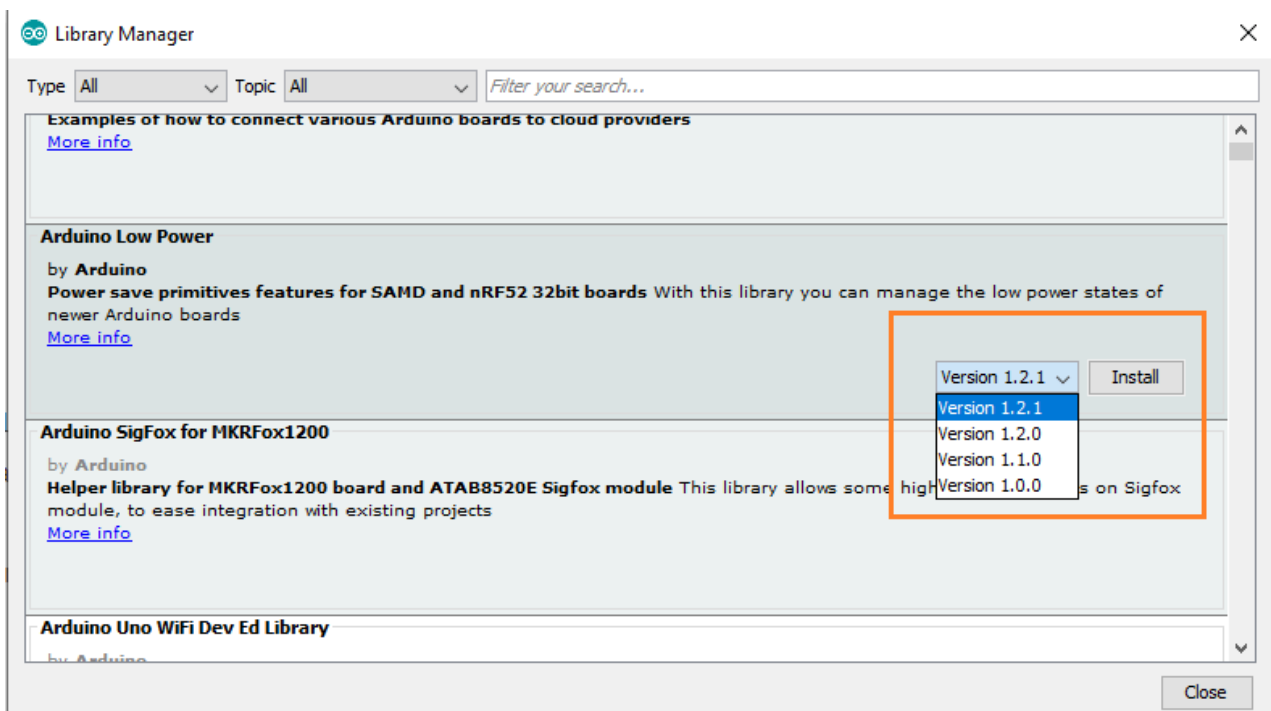
- Open the library manager to install a new library in Arduino. Click on **Sketch -> Include Library -> Manage Libraries**, as shown below:



- A dialog box containing various libraries will appear, as shown below:



- A list of libraries will appear that are ready to install. Need to select the specific library -> select the **version** -> click on **Install** button, as shown below:

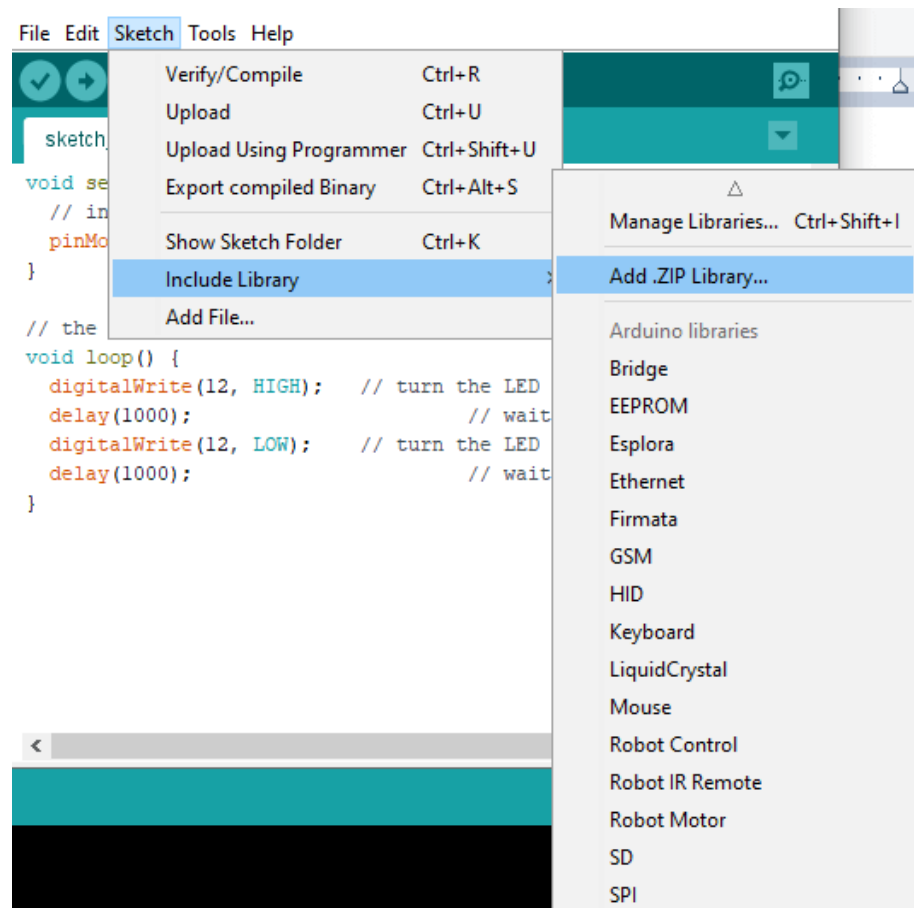


Sometimes there is only one version available for download. In such case, the dropdown of the version will not appear. Hence, it's normal. The 'INSTALLED' tag will appear in front of the library that is already installed on the computer.

Importing a .zip Library

If Programmer wants to add their own library, they can select the desired folder from their computer. The particular zip file containing the library can be imported in the Arduino.

It is shown below:



Programmer can also recheck from the option **Sketch -> Include library** to see that the added libraries are present or not on the list.

Arduino LCD Display

The LCD (**Liquid Crystal Display**) is a type of display that uses the liquid crystals for its operation. Here, programmer will accept the serial input from the computer and upload the sketch to the Arduino. The characters will be displayed on the LCD.

The library that allows us to control the LCD display is called **Liquid Crystal Library**, which is discussed below:

The library is declared as:

1. `#include <LiquidCrystal.h>`

The library is based on a compatible chipset called **Hitachi HD44780**. It is found on most of the LCDs that are based on text. It works with either an 8-bit mode or 4-bit mode. Here, the bit mode signifies the data lines in addition to the enable, rs, and rw control lines (optional).

LCD Structure

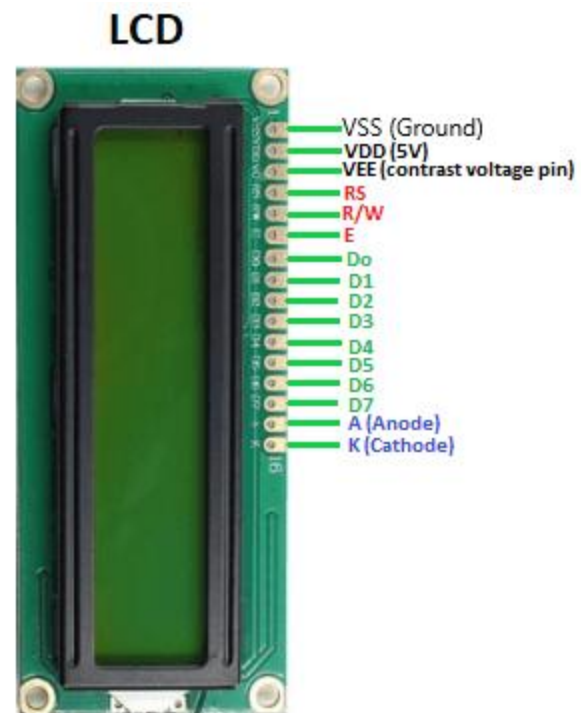
The LCD display has a 16-pin interface.

The structure of the LCD is shown below:

The Liquid Crystal Display has a parallel interface. It means that the microcontroller operates several pins at once to control the LCD display.

The 16-pins present on the LCD display are discussed below:

- **RS** : The **Register Select (RS)** pin controls the memory of the LCD in which we write the data. Programmer can select either the **data register** or the **instruction register**.
- **R/W** : The Read/Write pin selects the reading or writing mode.
- **E** : The **Enable (E)** mode is used to enable the writing to the registers. It sends the data to the data pins when the mode is HIGH.
- **D0 to D7** : These are eight data pins numbered as D0, D1, D3, D3, D4, D5, D6, and D7. We can set the state of the data pin either HIGH or LOW.



Pin 1 of the LCD is the **Ground** pin, and pin 2 is the **Vcc** or the voltage source pin.

The pin 3 of the LCD is the **V_{EE}** or the **contrast pin**. For example, we can connect the potentiometer's output to the VEE and can adjust the contrast of the LCD.

The A and K pins are also called as **Backlight pins** (Bklt+ and Bklt-).

LCD Interfacing with Arduino

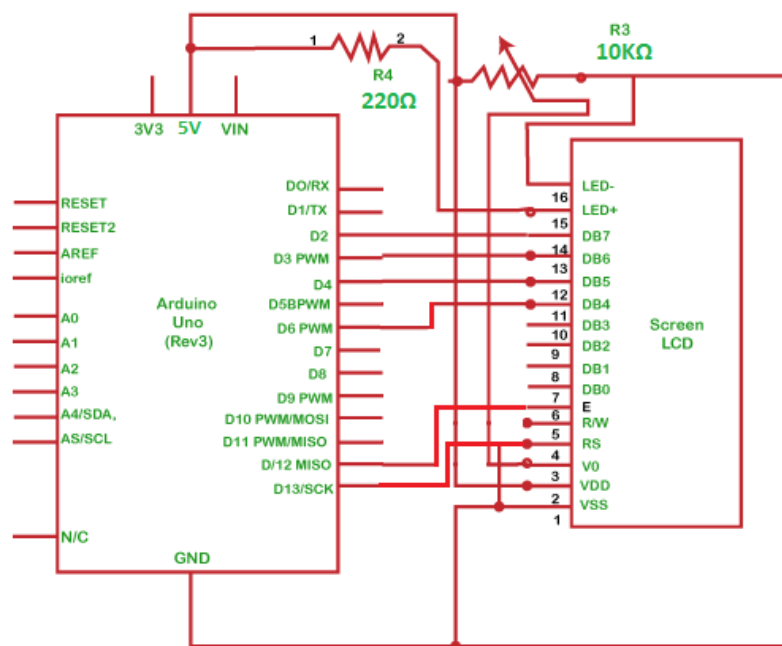
Hardware Required

The components required for the project are listed below:

- LCD Screen (Hitachi HD44780 compatible driver display)
- 1 x 220 Ohm Resistor
- 1 x 10K Ohm Resistor
- Arduino UNO board or Genuino board
- Jump wires
- Pin header required to solder the LCD display pins
- breadboard

Structure of the project

The structure of the project is shown below:



Procedure

The connection is explained below:

first connect the data pins of LCD to the digital pins.

- Connect the RS pin of LCD to pin 13 of the Arduino board.
- Connect the Enable pin of LCD to pin 12 of the Arduino board.
- Connect the D4 pin of LCD to pin 6 of the Arduino board.
- Connect the D5 pin of LCD to pin 4 of the Arduino board.

- Connect the D6 pin of LCD to pin 3 of the Arduino board.
- Connect D7 pin of LCD to pin 2 of the Arduino board.
- Connect the middle terminal of the potentiometer to the VEE (contrast pin).
- Connect the two ends of the potentiometer to the Ground and 5V.
- Connect one end of a resistor to the A and K of the LCD and another end to 5V.

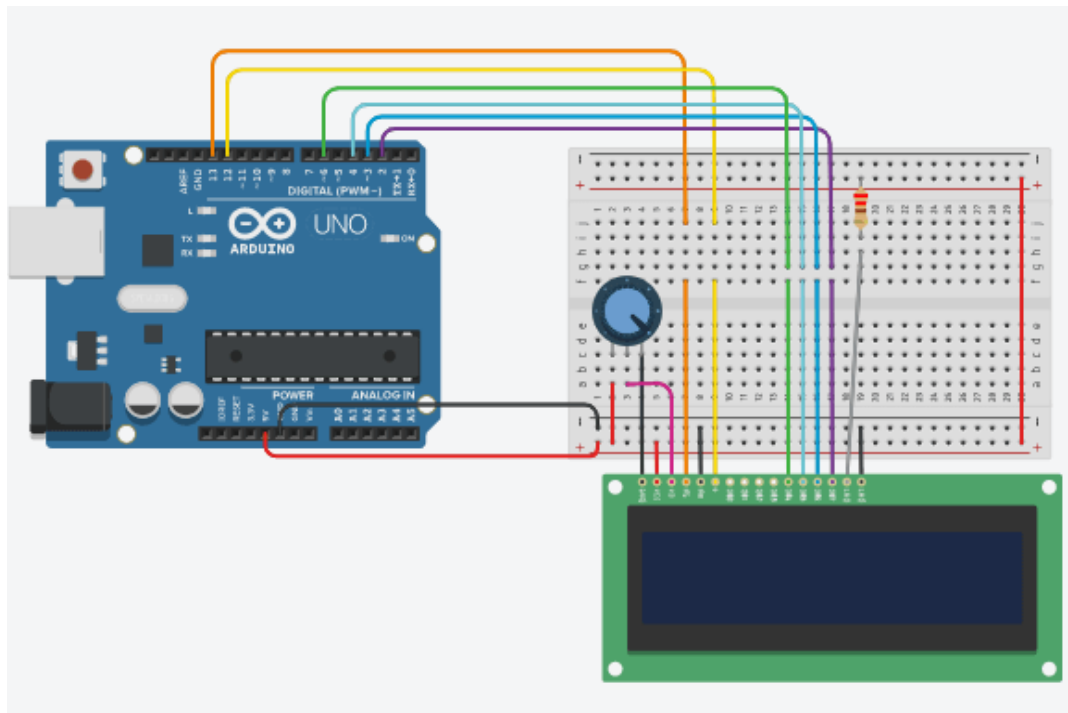
Sketch

The code to display the specified message on the LCD display is given below:

```

1. // here, we will include the liquid crystal library:
2. #include <LiquidCrystal.h>
3. // initialize the library with the pins on the Arduino board
4. LiquidCrystal lcd(13, 12, 6, 4, 3, 2);
5. void setup() {
6.   // Here, 16 and 2 are the columns and rows of the LCD
7.   lcd.begin(16, 2);
8.   // It prints the message on the LCD.
9.   lcd.print("hello Arduino");
10.    // We can modify the message as per our choice.
11. }
12. void loop() {
13.   // It sets the cursor to column 0, line 1
14.   // Since counting begins with 0, line 1 is the second row
15.   lcd.setCursor(0, 1);
16.   // print the number of seconds
17.   lcd.print(millis() / 1000);
18.   // Here, millis() is the return type of the timer in milliseconds
19. }
```

Connection Diagram



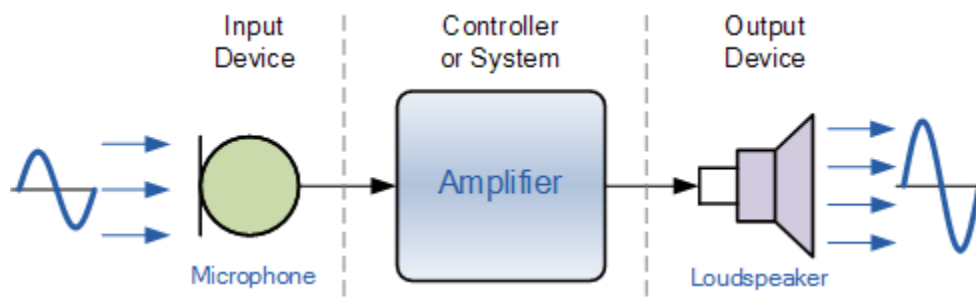
Sensors and Transducers

Sensors and transducers are input and output devices respectively that can be incorporated into an electronic circuit or system allowing it to measure or change its surrounding environment.

The word “Transducer” is the collective term used for both Sensors which can be used to sense a wide range of different energy forms such as movement, electrical signals, radiant energy, thermal or magnetic energy etc, and Actuators which can be used to switch voltages or currents.

Electrical Transducers are used to convert energy of one kind into energy of another kind, so for example, a microphone (input device) converts sound waves into electrical signals for the amplifier to amplify (a process), and a loudspeaker (output device) converts these electrical signals back into sound waves and an example of this type of simple Input/Output (I/O) system is given below.

Simple Input/Output System using Sound Transducers



Common Sensors and Transducers

Quantity being Measured	Input Device (Sensor)	Output (Actuator) Device
Light Level	Light Dependant Resistor (LDR) Photodiode Photo-transistor Solar Cell	Lights & Lamps LED's & Displays Fibre Optics
Temperature	Thermocouple Thermistor Thermostat Resistive Temperature Detectors	Heater Fan
Force/Pressure	Strain Gauge Pressure Switch Load Cells	Lifts & Jacks Electromagnet Vibration

Position	Potentiometer Encoders Reflective/Slotted Opto-switch LVDT	Motor Solenoid Panel Meters
Speed	Tacho-generator Reflective/Slotted Opto-coupler Doppler Effect Sensors	AC and DC Motors Stepper Motor Brake
Sound	Carbon Microphone Piezo-electric Crystal	Bell Buzzer Loudspeaker

All types of sensors can be classed as two kinds, either Passive Sensors or Active Sensors. Active sensors require an external power supply to operate, called an *excitation signal* which is used by the sensor to produce the output signal.

A passive sensor does not need any additional power source or excitation voltage. Instead a passive sensor generates an output signal in response to some external stimulus. For example, a thermocouple which generates its own voltage output when exposed to heat. Then passive sensors are direct sensors which change their physical properties, such as resistance, capacitance or inductance etc.

But as well as analogue sensors, Digital Sensors produce a discrete output representing a binary number or digit such as a logic level "0" or a logic level "1".

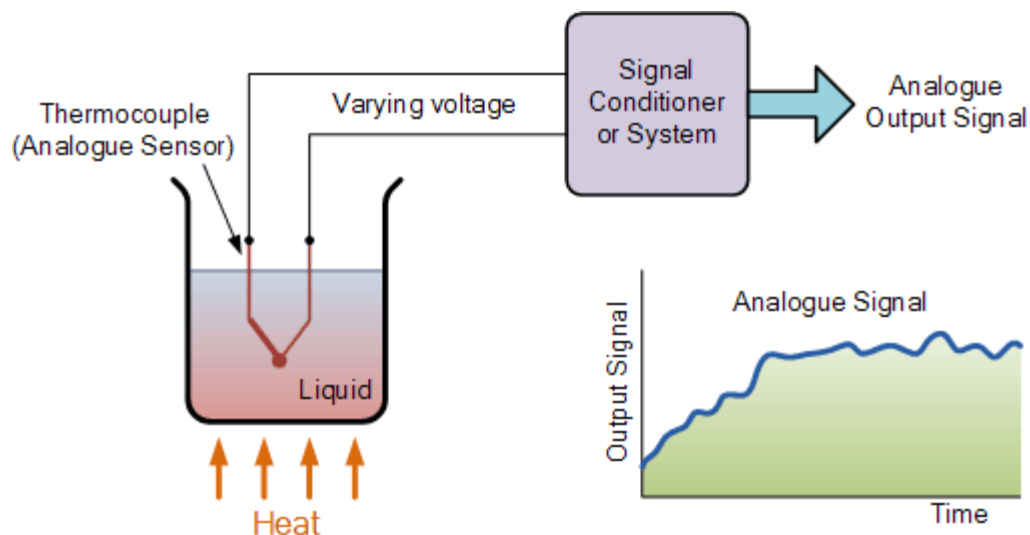
Analogue and Digital Sensors and Transducers

Analogue Sensors

Analogue Sensors produce a continuous output signal or voltage which is generally proportional to the quantity being measured. Physical quantities such as Temperature, Speed, Pressure, Displacement, Strain etc are all analogue quantities as they tend to be continuous in nature.

For example, the temperature of a liquid can be measured using a thermometer or thermocouple which continuously responds to temperature changes as the liquid is heated up or cooled down.

Thermocouple used to produce an Analogue Signal



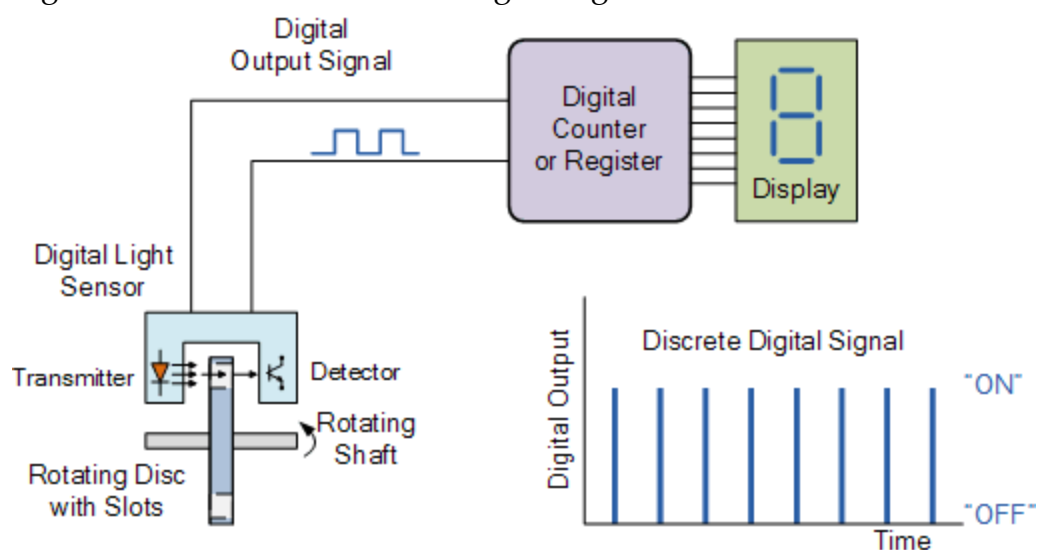
Analogue sensors tend to produce output signals that are changing smoothly and continuously over time. These signals tend to be very small in value from a few micro-volts (μV) to several milli-volts (mV), so some form of amplification is required.

Then circuits which measure analogue signals usually have a slow response and/or low accuracy. Also analogue signals can be easily converted into digital type signals for use in micro-controller systems by the use of analogue-to-digital converters, or ADC's.

Digital Sensors

As its name implies, Digital Sensors produce a discrete digital output signals or voltages that are a digital representation of the quantity being measured. Digital sensors produce a Binary output signal in the form of a logic "1" or a logic "0", ("ON" or "OFF").

Light Sensor used to Produce a Digital Signal



In simple example above, the speed of the rotating shaft is measured by using a digital LED/Opto-detector sensor. The disc which is fixed to a rotating shaft (for

example, from a motor or robot wheels), has a number of transparent slots within its design. As the disc rotates with the speed of the shaft, each slot passes by the sensor in turn producing an output pulse representing a logic “1” or logic “0” level.

These pulses are sent to a register of counter and finally to an output display to show the speed or revolutions of the shaft. By increasing the number of slots or “windows” within the disc more output pulses can be produced for each revolution of the shaft.

Compared to analogue signals, digital signals or quantities have very high accuracies and can be both measured and “sampled” at a very high clock speed. The accuracy of the digital signal is proportional to the number of bits used to represent the measured quantity.

For example, using a processor of 8 bits, will produce an accuracy of 0.390% (1 part in 256). While using a processor of 16 bits gives an accuracy of 0.0015%, (1 part in 65,536) or 260 times more accurate. This accuracy can be maintained as digital quantities are manipulated and processed very rapidly, millions of times faster than analogue signals.

Sensors

Sensor measuring very small changes must have very high sensitivity. Sensors are designed in such a way that they cause only a small effect on what is measured. That is why most of the sensors are small in size. Analog Sensors produce a continuous output signal or voltage which is generally proportional to the quantity being measured. Digital Sensors produce a discrete output signal or voltage that is a digital representation of the quantity being measured.

Voltage Sensor:



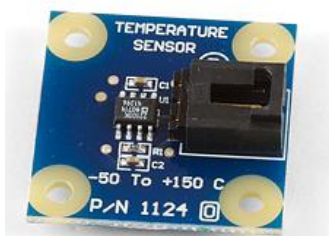
Voltage sensor converts potential difference measured between two points of an electrical circuit into proportional electrical signal. This signal can be stored for further analysis or can be utilized for control purpose.

Current Sensor:

Current sensor can be used to measure current of an electrical circuit. It converts current of an electrical circuit into proportional electrical signal. This signal can be utilized for further analysis or control purpose.



Temperature Sensor:

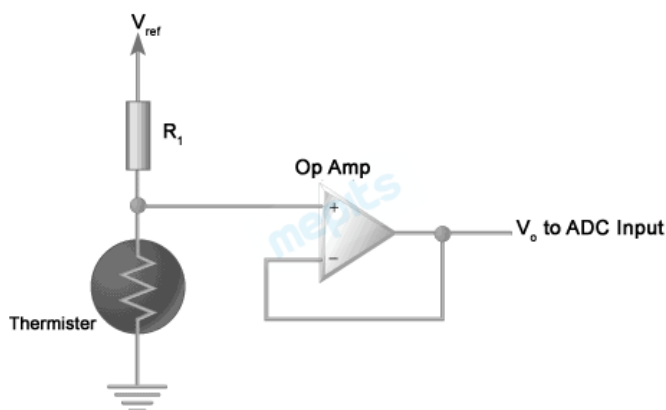
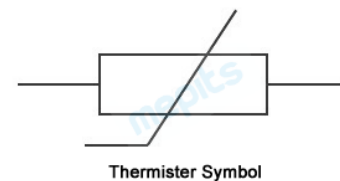


Temperature sensors are electronic devices used to measure temperature of a medium. It provides a signal proportional to the temperature of the medium. Mainly there are two

types of temperature sensors contact type sensors and non-contact type. Contact type sensors measure the temperature of an object or liquid by physical contact. These types of sensors are mainly used for low temperature measurements. Non-contact type sensors use the thermal radiation of the heat source to monitor change in temperature. Non-contact sensors are used when the object is moving or when contact with the object changes its temperature. For very high temperature measurement this type of sensors are used. Thermistor and thermocouple are two commonly using temperature sensors.

Thermistor:

Thermistor is a resistor whose resistance change proportionally with change in temperature. In most of the thermistors, resistance decreases with increase in temperature. Thermistors have very high speed of response to any changes in temperature. Generally thermistors are constructed using ceramic or polymer



material.

This figure shows the circuit for measuring temperature using thermistor. Resistor R_1 pulls thermistor to reference voltage V_{ref} . Resistor and thermistor combination forms a voltage divider. According to temperature, thermistor resistance will change. This results in varying voltage at the junction. This voltage is

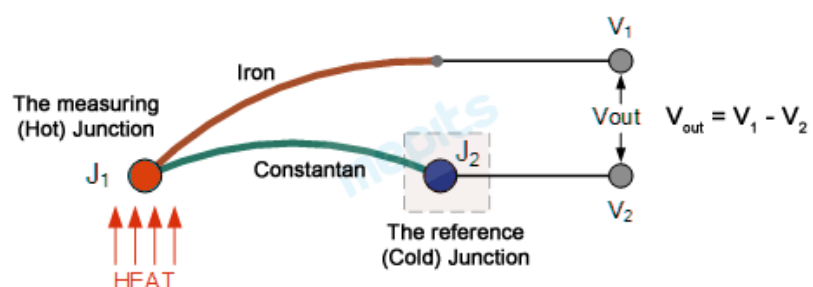
amplified by an operational amplifier.

Thermocouple:

Thermocouples are temperature sensors that consist of two junctions of dissimilar metals, such as copper and constantan that are welded or crimped together. One junction (J_1) is connected to the object whose temperature is to be measured. This junction is called measuring junction. Other is connected to a known temperature. This junction is known

as reference junction.

Due to the temperature difference in two junctions, current will flow through



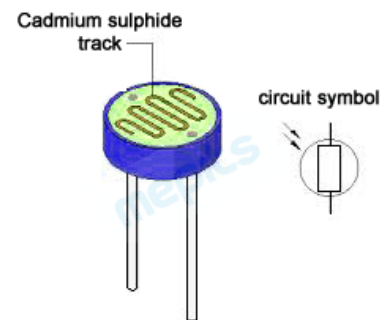
thermocouple. We can measure the potential difference and by processing this voltage we can calculate the temperature of the body. Thermocouple is the most commonly used temperature sensor. It has widest temperature range from -200°C to 2000°C .

Light Sensor:

Light sensor means a passive electronics device which can be used for measuring light. It produces a signal proportional to the intensity of light falls on it. Light sensors are commonly known as "Photoelectric Devices" or "Photo Sensors" because they convert light energy into electricity.

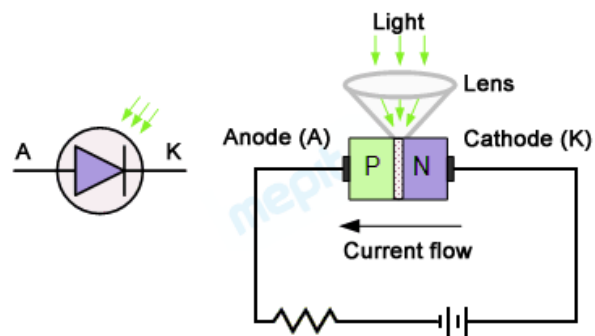
Light Dependent Resistor (LDR):

LDR normally have very high resistance about 1000000 ohms. As light fall on it, its resistance decreases to a great extent. When this material is exposed to light by creating hole-electron pairs its resistance will drop from several thousands of ohms to only a few hundred ohms.



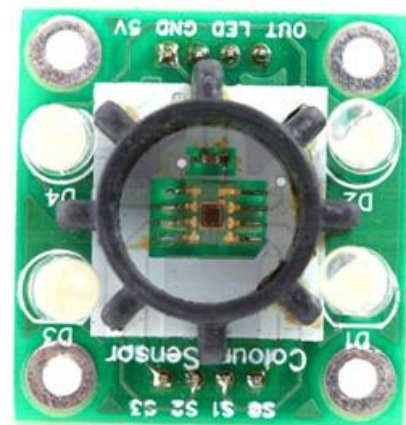
Photodiode:

Photo diode is a device which converts light into current or voltage, according to the mode of operation. It is similar to conventional PN-junction diode, because it is a regular semiconductor diodes whose outer casing is either transparent or has a lens to focus the light onto the PN junction for increased sensitivity. Photo-diode varies the current conducted through it according to the intensity of light falls on it. As light fall in it more charge carriers are created. Photodiodes have a more linear response than LDRs. Photo diodes are very accurate and stable than LDR. Commonly used materials to produce photo diodes are silicon, germanium, lead sulfide etc.



Colour Sensor:

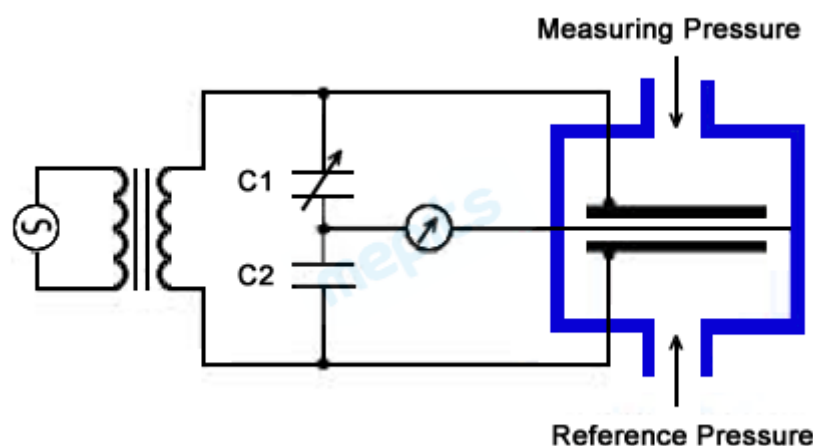
Colour sensor is used to detect colours. Colour sensors mainly consist of an array of photo detectors. Each of photodiodes will have red, blue or green filter or no filter. Photo diodes with each filter are present in equal numbers. They are also distributed evenly throughout the array. Colour filter only allow the particular Colour to pass through it and block the rest. So using this array



intensity of light can be easily calculated. An oscillator present in the sensor will produce a square wave with frequency proportional to the intensity of the chosen colour.

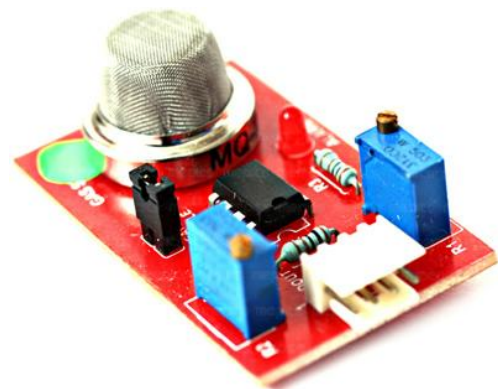
Pressure Sensor:

Electronics pressure sensors are used to measure pressure, mainly of liquids and gases. They record the pressure and convert it into proportional output electrical signal. Capacitive pressure sensor consists of two thin metal plates as capacitor. One side of this capacitor is exposed to measuring pressure and other to a reference pressure. Change in pressure changes the gap between the plates. So capacitance varies with difference in pressure. By processing this capacitance we can measure pressure. Pressure sensors have wide variety of application in different fields such as automobile, manufacturing, aviation, bio medical measurements, air conditioning, hydraulic measurements etc.

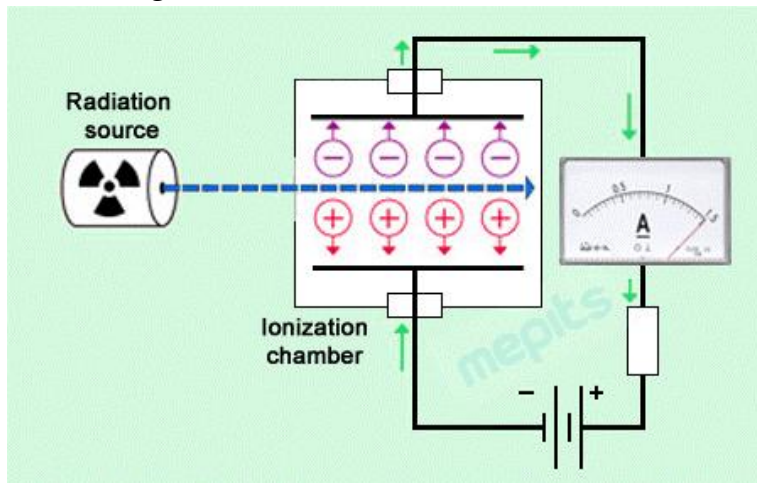


Smoke Sensor:

Smoke sensors are used to detect smoke. It is widely used for many industrial applications as well as commercial application. Smoke sensors work either by optical detection or by ionization process. Some sensors use both the detection methods for higher sensitivity to smoke. In optical detection method sensor have a light source producing a light beam and also a photodiode. When smoke enters in to the sensor some light is scattered by smoke particle. Scattered light is detected by the photodiode. When smoke particles enter



the chamber it combines with ion and neutralizes it and drop the current. By monitoring this current we can detect smoke.



Humidity Sensor

It is an electronic sensor which measures atmospheric humidity by changing its electrical characteristics such as resistance or capacitance according to the amount of atmospheric humidity. Output voltage or current is produced proportional to the fluctuations in humidity. Most common humidity sensors use capacitive measurements. They consist of two metal plates with a non-conductive humidity-sensing film between them. This film is to collect moisture from the air. Moisture cause minute changes in voltage between two plates. This change in voltage is proportional to the amount of humidity.



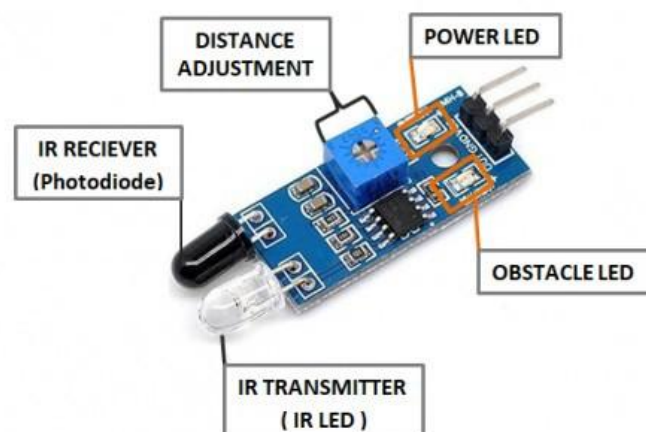
www.mepits.com

Arduino sensors:

Sensors are responsible for the conversion of physical quantities like temperature or light into electrical quantities. Arduino sensors are made for fluent and efficient output. The digital signals are generated through IR obstacles or ultrasonic modules. Such as Motion sensors, object detection transfer the environment changes as an input signal to the electronic device. Arduino include different type of sensors such as light sensors, temperature sensors, motion or object detector, ultrasonic sensors, knock sensors, tracking or metal detector etc.

IR obstacle avoidance sensor:

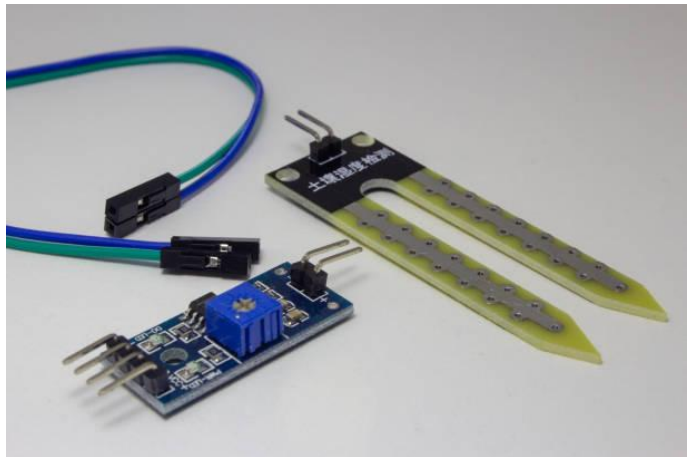
IR transmitter and receiver work together to generate electric signal under the influence of any object coming into the way of IR light. The range of the module can be



adjusted as required by using its potentiometer. The principle of the IR sensor is the reflection of IR light that is detected through the receiver.

Soil moisture sensor:

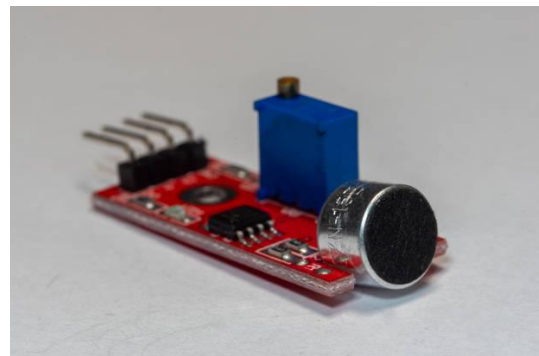
Also referred to as soil humidity detector. Soil moisture sensor detect the water content in the soil based on the alteration of soil properties that are depending on the moisture. It works best when completely surrounded by soil. Different types of moisture content are available such as a point soil moisture detector that tells the water content of a single point.



Multiple single-point sensors are used to measure the water content of more areas. There are soil water profiling probes are being used to detect water on vertical soil horizons.

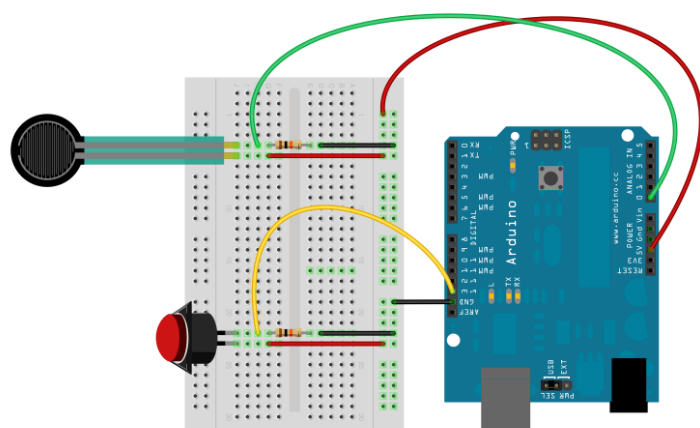
Microphone sound sensor:

The sound sensor detects the loudness of sound. The sensitivity of the digital output pin can be adjusted by a built-in potentiometer. It is used to detect the surrounding environment sound as if it accedes the limit. LED light turns on if the sound intensity gets higher than a specific threshold.



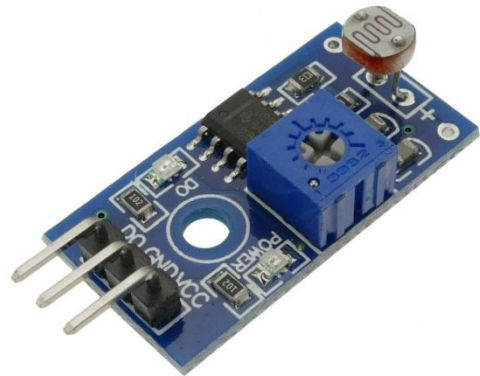
Barometric pressure sensor:

Barometer/barometric pressure sensors are used to detect atmospheric pressure alterations in the surrounding environment. Mainly it is used to interpret short-term weather changes. The miniaturized barometers can be used in microcontroller boards (Arduino) or smartphones. In the past mercury, barometers were used but now in modern pressure sensors, no liquid is required. MEMS and Aneroid barometers are widely used nowadays.



Photoresists:

Light-sensitive resistors alter the resistance value of the component under exposure to light. There is an inverse relationship between resistance and light i.e. when light intensity increases the resistance decreases. These are also known as Light Dependent Resistors. LDR has wide applications influencing daily life. LDR or photoresists are used in Street lights, automatic doors, alarm clocks, light intensity meters, etc.

**Knock sensor:**

This sensor consists of an LED light that provides a visual output for the input of knock sound or vibration. The intensity of the signal is depicted by turning on the LED light. Knock sensor is widely used in car engines to detect ignition timing and damage control. The sound or vibration signal of the engine is altered into an electrical signal directed to the car's ECU (engine control unit). CEL (check engine light) of cars can also be monitored through a knock sensor.

**Heart rate sensor:**

For determining and tracking heartbeats, a heart rate pulse sensor is used under a voltage of 3 to 5V. An optical heart rate sensor assesses pulse waves through blood pressure alterations in vessels. Optical sensors use alterations in volume as a signal and a green LED light to make an output accordingly.

**Alcohol sensor/MQ3 sensor:**

It is a gas sensor that assesses the presence of ethanol in the air. The concentration of ethanol/alcohol determines the output through an LED light. The range of detection is 10-1000 ppm of alcohol or ethanol.

**Reed switch/magnetic proximity sensor:**

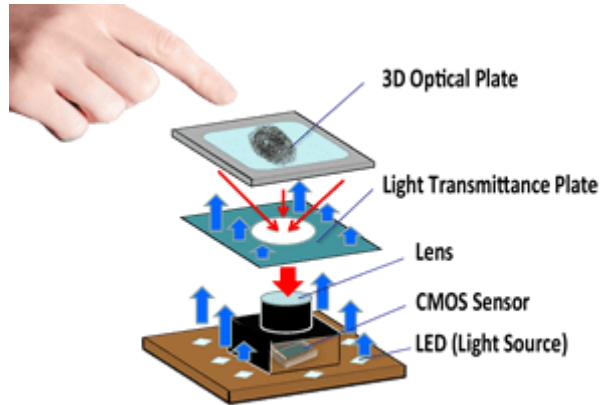
A reed switch is used to detect magnetic fields and control electricity flow. In the simplest reed switch, a pair of ferromagnetic blades is used on a board. Under a magnetic field, the blades tend to pull towards each other and give the signal of



CLOSE, and in absence of a magnetic field as they keep their distance the signal is found in ON for electricity flow. On Arduino board magnetic sensor is connected through 5 volts pin to the board along with series of resistors to a 3.3volts pin. Transistor output can be seen when the magnetic field is nearby to the sensor and vice versa.

Optical Fingerprint sensor:

Ridges and grooves of fingerprint determined through light and dark area identified by the scanner of fingerprint sensor which acquires finger tip's low-resolution snap shorts and generates a 2D image of the fingerprint. For illumination LED light is used. It can be connected to TTL serial system, microcontroller, and FLASH memory.



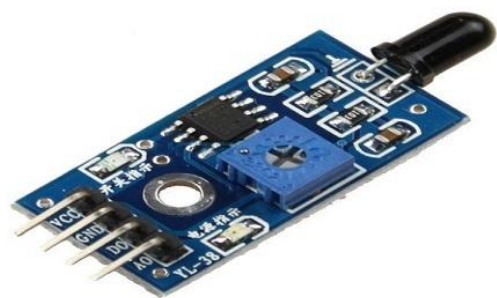
The ultrasonic fingerprint sensor is more accurate than the optical sensor as it gives 3D fingerprint model acquired through ridges, pores, and pulse.

Speed sensor/Tachometer:

Speed of rotating or moving objects like motor or pulse can be determined by Arduino speed sensor. It determines the rotational speed and changes in the speed of the object. There are many kinds of tachometers used like magneto-resistive speed sensor, IR dependent speed sensor, and hall-effect based speed sensor.

Flame sensor:

Flame and light can be detected through a flame sensor. The light wavelength of 760-1100 nm within 0.8 m can be detected through this sensor. Flame of Light Intensity determines the distance of detection. These sensors are used in houses, factories, or offices to avoid possible damages and fire breakout.



Interfacing DHT11 Humidity & Temperature Sensor with Arduino & LCD

Introduction

The DHT11 Temperature and Humidity Sensor is an advanced piece of equipment that boasts a calibrated **digital signal output** by its integrated temperature and humidity sensor complex. At the heart of this sensor lies an efficient **8-bit microcontroller**. The DHT11 sensor is designed featuring a **resistive element** and a wet **NTC temperature** measuring component.

DHT11 Humidity Temperature Sensor

The **DHT11** is a commonly used and cost-effective sensor that provides readings of both **temperature** and **humidity**. Compact and cost-effective, it has become a popular choice for its **relative accuracy** and ease of integration. Operating on a simple **one-wire communication protocol**, the DHT11 can be seamlessly interfaced with a variety of **microcontrollers**.

Features/Specifications of DHT11 Sensor

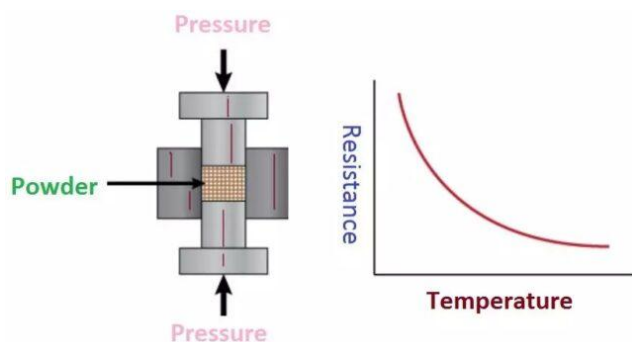
- **Power Supply:** Typically operates from 3.3 to 5V DC, making it suitable for interfacing with most microcontrollers.
- **Temperature Range:** 0°C to 50°C with $\pm 2^\circ\text{C}$ accuracy.
- **Humidity Range:** 20% to 90% RH (Relative Humidity) with $\pm 5\%$ accuracy.
- **Output:** Calibrated digital signal. It employs a single-wire communication protocol.
- **Sampling Period:** Suggested minimum time of 1 second between readings to ensure sensor accuracy.
- **Dimensions:** Compact size, Available in a 4-pin single-row package.
- **Longevity:** Offers excellent long-term stability.

Construction of DHT11 Sensor

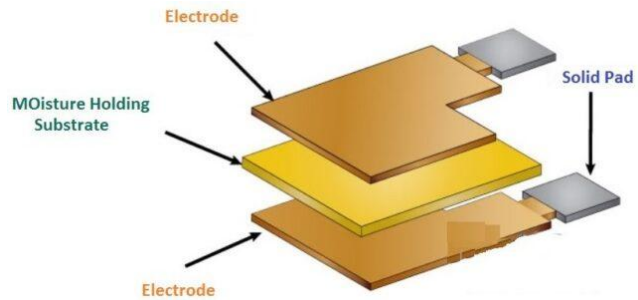
- **Sensing Elements:** The DHT11 integrates two primary components for its operations:
 1. A thermistor for temperature measurements.
 2. A capacitive humidity sensor for gauging atmospheric moisture.
- **IC Integration:** An 8-bit microcontroller reads the outputs from the two sensors and translates them into a format suitable for digital systems.
- **Packaging:** The sensor elements are often encased in a plastic casing, which ensures protection against environmental factors while still allowing for accurate readings.

Working of DHT11 Sensor

- **Temperature Measurement:** The core of the temperature sensing capability is the thermistor, a type of resistor whose resistance changes with temperature. As the temperature fluctuates, so does the resistance of the thermistor. This variance is then read and converted into temperature values by the on-board microcontroller.

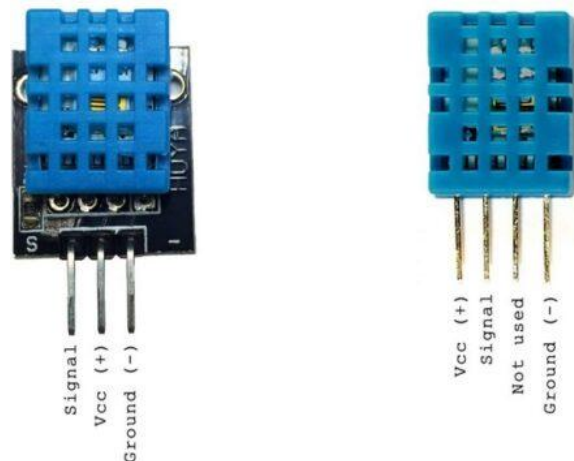


- **Humidity Measurement:** The capacitive humidity sensor operates by having a dielectric material between two plates. As humidity changes, the dielectric constant of the material changes, leading to a variance in capacitance. This change in capacitance is then read and interpreted as a humidity percentage by the on-board microcontroller.
- **Signal Output:** After the microcontroller processes the readings, the data is sent as a digital signal through a single-wire communication protocol. This digital output can then be easily read by microcontrollers such as Arduino, making integration and data interpretation straightforward.
- **Pinout of DHT11 Sensor**



The DHT11 typically comes with a 4-pin package, although only three of them are functionally used:

1. **VCC (Pin 1):** This is the power supply pin, which can accept voltages from 3.3V to 5V, making it compatible with most microcontroller systems.
2. **Data (Pin 2):** This is the pin through which the DHT11 communicates. It uses a proprietary single-wire protocol to transmit temperature and humidity data to the connected microcontroller.
3. **NC (Pin 3):** Not connected or used.
4. **GND (Pin 4):** Ground pin, used to complete the circuit.



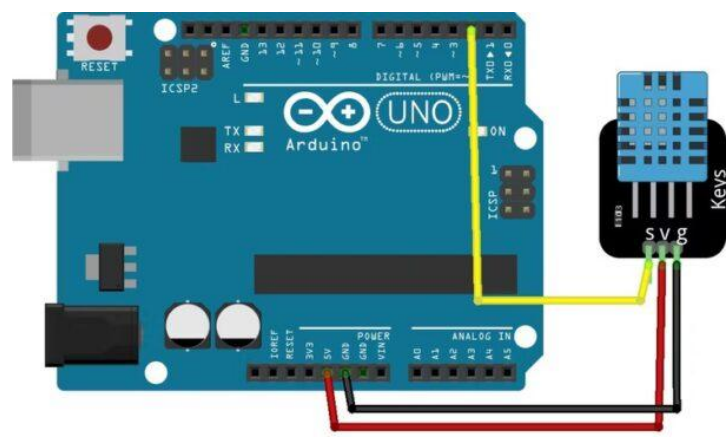
Interfacing DHT11 Sensor with Arduino

Let us interface the DHT11 Humidity Temperature Sensor with Arduino UNO.

Hardware Connection

The connection diagram is pretty simple as shown in the image below.

Connect the VCC & GND Pin of DHT11 Sensor Module to 3.3V & GND pin of Arduino respectively. Similarly connect



the DHT11 Data pin to Arduino Digital Pin 2.

Source Code/Program

First programmer need to install the DHT11 Sensor library. Download the DHT11 Sensor library from following link and add it to the Arduino Library Folder.

Download: [DHT11 Sensor Library](#)

Here is a complete code for interfacing DHT11 Sensor with Arduino.

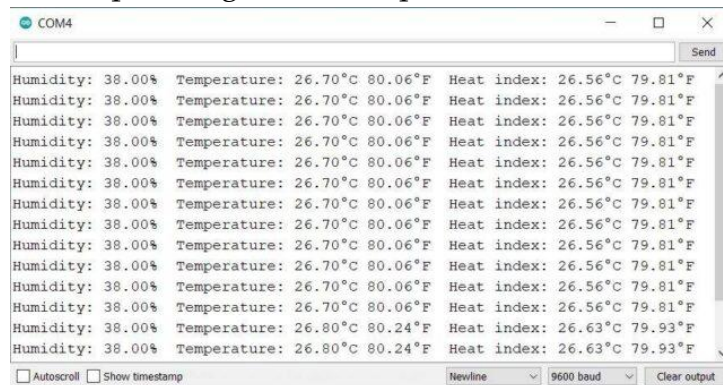
```
#include "DHT.h"
#define DHTPIN 2    // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop()
{
  // Wait a few seconds between measurements.
  delay(2000);
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
```

```

Serial.print(f);
Serial.print(F("°F Heat index: "));
Serial.print(hic);
Serial.print(F("°C "));
Serial.print(hif);
Serial.println(F("°F"));
}

```

To upload this code, Select Arduino UNO from Board Manager and the COM port. After uploading the code, open the Serial Monitor.

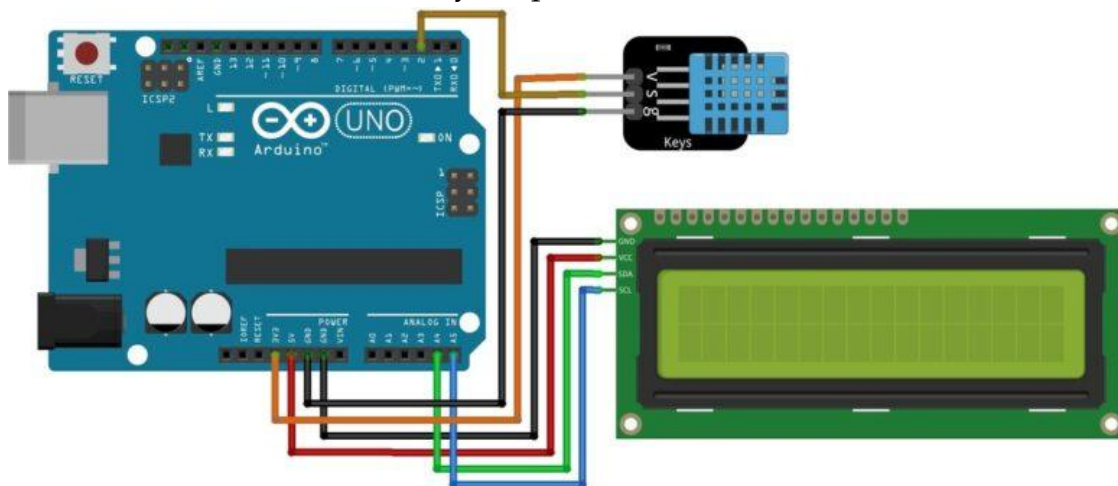


The Serial Monitor will display the temperature readings in degree Celcius as well as Fahrenheit. It will also show the value of Humidity and Heat Index.

Interfacing DHT11 Sensor with Arduino & LCD Display

Hardware Connection

The hardware connection is very simple as shown in the schematic below.



Connect the SDA & SCL pin of LCD Display to Arduino A4 & A5 Pin respectively. Similarly connect the VCC & GND of LCD Display to 5V & GND of Arduino.

Source Code/Program

After LCD connection is done, you can move to the coding part of this project. Programmer need to add LCD I2C Library to the Arduino library folder again

Here is a complete code for interfacing DHT11 Sensor with Arduino & 16×2 I2C LCD Display. Copy the following code and paste it on your Arduino IDE editor window.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "DHT.h"
LiquidCrystal_I2C lcd(0x27, 16, 2);
#define DHTPIN 2    // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);
void setup()
{
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
  dht.begin();
}
void loop()
{
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("% Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
```

```

Serial.print(f);
Serial.print(F("°F Heat index: "));
Serial.print(hic);
Serial.print(F("°C "));
Serial.print(hif);
Serial.println(F("°F"));
lcd.setCursor(0, 0);
lcd.print("Temp: ");
lcd.print(t);
lcd.print("°C");
lcd.setCursor(0, 1);
lcd.print("Humi: ");
lcd.print(h);
lcd.print("%");
delay(2000);
lcd.clear();
}

```

Upload the code to Arduino Board. After the code is uploaded, the LCD Display will show the value of Temperature and Humidity. The reading changes after every 2 seconds.

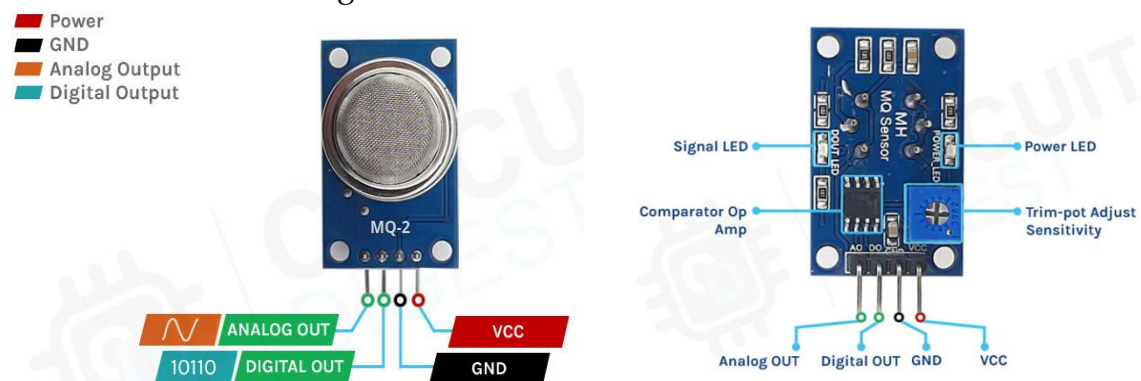
MQ-2 Flammable Gas and Smoke Sensor interfacing with Arduino

MQ-2 Gas Sensor

The **MQ-2 Combustible Gas and Smoke Sensor** is commonly used in smoke and gas detectors. This sensor can not only **detect gas or smoke, but can also detect LPG, Alcohol, Propane, Hydrogen, Methane, and Carbon Monoxide.**

MQ-2 Gas Sensor Pinout

The MQ-2 Gas detection sensor module has four pins VCC, GND, A_{out} and D_{out} that can be used to get the needful information out of the sensor, The **pinout of the MQ-2 Gas detection sensor** is given below:



VCC is the power supply pin of the Gas Detection Sensor that can be connected to 5V of the supply.

GND is the ground pin of the board and it should be connected to the ground pin of the Arduino.

D_{OUT} is the Digital output pin of the board, output low indicates gas or smoke is not present in the atmosphere and output high indicates gas or smoke is present in the atmosphere.

A_{OUT} is the Analog output pin of the board that will give us an analog signal which will vary between vcc and ground based on the gas level detected

MQ-2 Gas Sensor Working

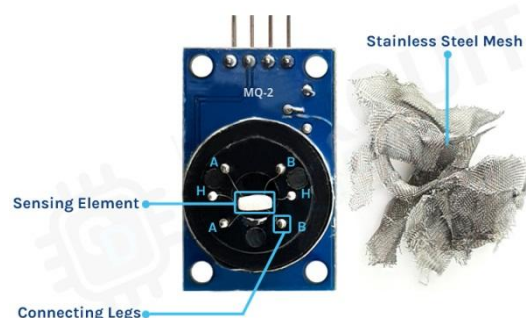
The MQ-2 gas sensor needs a heating element in order to properly detect combustible gasses but, a heating element close to combustible gasses could be disastrous, so the sensor is manufactured with an anti-explosion network made out of two thin layers of stainless steel mesh. The heating element is placed inside this stainless steel mesh



This mesh structure also provides resistance against dust and other suspended particulars and it only lets in the gaseous elements from the atmosphere. First one is the **heating element** which is made out of nichrome wire and other is the **sensing element** that is made out of a platinum wire with a

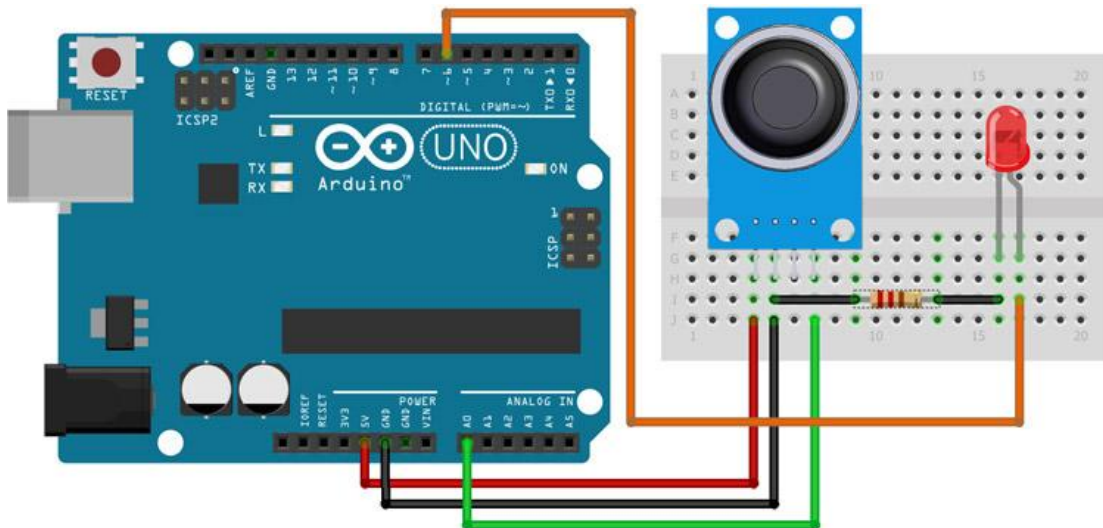
coating of tin dioxide. Now we don't want you to cut and damage your sensor, so we have done that for you, the below image shows the mesh decapped from the actual sensor

The sensor looks like something like this when the mesh is removed. The star-shaped pins of the sensor are formed because of the structure of the actual sensing and heating element and it's connected to the six legs of the sensor. The black base of the sensor is made with Bakelite to improve thermal conductivity.



MQ-2 Combustible Gas Sensor with Arduino UNO – Connection Diagram

To work with the sensor we need to power the sensor first, for that we are using the 5V and GND pin of the Arduino UNO Board and we are connecting the output pin of the sensor to A0 pin of the Arduino.



As shown in the above schematic we have connected an LED to digital PIN 6 of the Arduino and the analog pin is connected to the A0 pin of the arduino, and the ground pin is common in between the led and the sensor. We will **program the Arduino** so that the brightness of the LED will change depending on the concentration of the gas present in the atmosphere.

Arduino Code for Interfacing MQ-2 Gas Sensor Module

The code for the **Arduino mq-2 gas sensor module** is very simple and easy to understand. Programmer is just reading the analog data out of the sensor and changing the brightness of the LED according to the received data. Programmer is only processing the analog data coming out of the sensor for the digital data.

```
// Sensor pins pin D6 LED output, pin A0 analog Input
#define ledPin 6
#define sensorPin A0
void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
}
void loop() {
  Serial.print("Analog output: ");
  Serial.println(readSensor());
  delay(500);
}
// This function returns the analog data to calling function
int readSensor() {
```



```

unsigned int sensorValue = analogRead(sensorPin); // Read the analog value from
sensor
unsigned int outputValue = map(sensorValue, 0, 1023, 0, 255);
// map the 10-bit data to 8-bit data
if (outputValue > 65)
    analogWrite(ledPin, outputValue); // generate PWM signal
else
    digitalWrite(ledPin, LOW);
return outputValue; // Return analog moisture value
}

```

Arduino Motion Sensor

All objects (having temperature higher than absolute zero) emit radiations from the generated heat. These radiations cannot be detected by a human eye. Hence, electronic devices such as motion sensors, etc. are used for the detection of these radiations.

PIR sensor

The Passive Infra-Red sensors or PIR sensors detect motion or movement of an object that detect infrared radiations, such as the human body. Hence, the use of sensors is very common.

The advantages of using a PIR sensor are listed below:

- Inexpensive
- Adjustable module
- Efficient
- Small in size
- Less power consumption
- It can detect motion in the dark as well as light.

The PIR sensor is shown below:

The PIR sensor has three terminals, which are listed below:

- VCC
- Digital Output
- GND (Ground)

Connect the Vcc terminal of the sensor to the 5V on the Arduino board. The PIR's sensor output can be connected to any of the digital pins on the Arduino board.

The applications of the PIR sensor are automation, security systems, etc. Such sensors work great in detecting the entrance of a person in an area and leaving it. The detection range of PIR sensors is from 5m to 12m.

Working of PIR Sensors



The working of the PIR sensor is entirely based on detecting the IR (Infra-Red) radiations, which are either emitted or reflected by the objects. The infrared radiations are detected by the crystalline material present at the centre of the sensor.

Structure of PIR Sensor

A round metal can is mounted on the center with the rectangular crystal that detects the IR radiations.

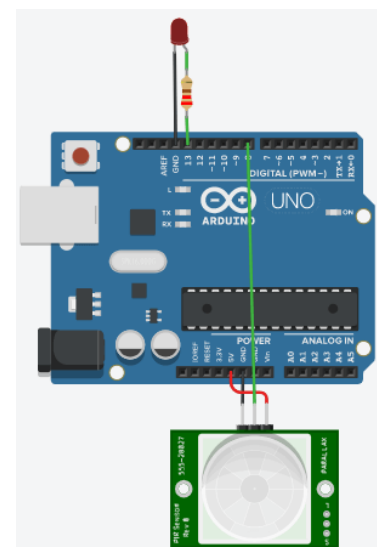
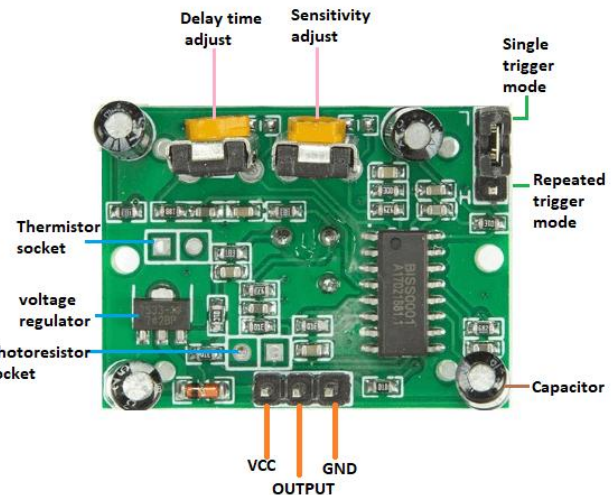
A ball like a lens present on some sensors helps in enhancing the viewing angle.

The bottom part of the sensor contains many circuits mounted on it, which is shown below:

Hardware Required

The components required for the project are listed below:

- 1 x PIR motion sensor
- Arduino UNO R3 board (We can take any Arduino board).
- Jump wires
- 1 x red LED (we can take LED of any color)
- 1 x 220 Ohm resistor



Principle

The movement of jumper present on the sensor on the L side will cause a change in the state of the sensor whenever the motion is detected. Such a condition is defined as a single trigger mode.

When the sensor resets the timer after every detection of motion, it is defined as repeated trigger mode. The two potentiometers present on the sensor are called as **Sensitivity** Potentiometer and **Time** Potentiometer. It should be restricted for atleast 15 seconds in front of the PIR sensor for proper calibration in the output. After 15 seconds, the sensor can easily detect movements.

If any movement is detected, the LED will be HIGH. If there is no such movement, the output will be LOW.

Connection

The steps to set up the connection are listed below:

- Connect the Vcc terminal of the PIR sensor to the 5V pin of the Arduino board.
- Connect the Output terminal of the PIR sensor to pin 8 of the Arduino board.
- Connect the GND terminal of the PIR sensor to the Ground pin of the Arduino board.
- Connect the positive leg of the LED in series with 220 Ohm resistor to pin 13 of the Arduino board.
- Connect the negative terminal of the LED to the Ground pin of the Arduino board.

Sketch

Consider the below code:

```

1. int LEDpin = 13; // LED pin
2. int PIRpin = 8; // The pin of Arduino connected to the PIR output
3. int PIRvalue = 0; // It specifies the status of PIR sensor
4. void setup() {
5.   pinMode(LEDpin, OUTPUT);
6.   pinMode(PIRpin, INPUT);
7.   // the output from the sensor is considered as input for Arduino
8.   Serial.begin(9600);
9. }
10. void loop()
11. {
12.   PIRvalue = digitalRead(PIRpin);
13.   if (PIRvalue == HIGH)
14.   {
15.     digitalWrite(LEDpin, HIGH);
16.     // turn ON LED if the motion is detected
17.     Serial.println("hello, I found you...heyyy..");
18.   }
19.   else
20.   {
21.     digitalWrite(LEDpin, LOW);
22.     // LED will turn OFF if we have no motion
23.     Serial.println("I cannot find you");

```

```

24. delay(1000);
25. }
26. }

```

The output will be based on the detection.

Arduino Light Sensor

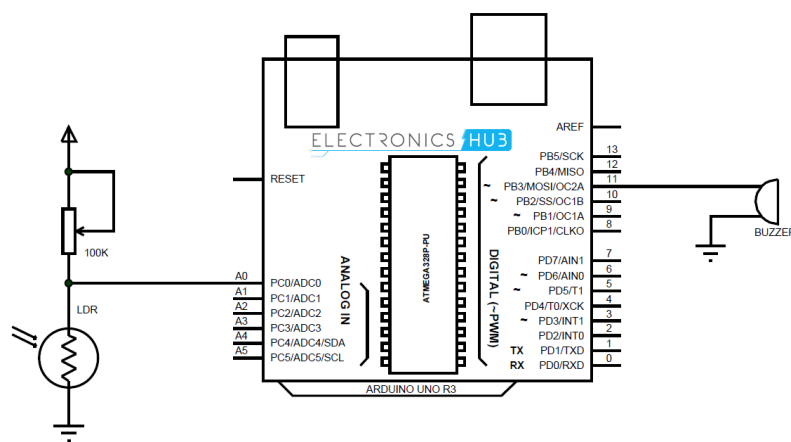
A Light Sensor is a device that detects light. It generates an output signal that is proportional to the intensity of light. A light sensor measures the radiant energy present in the wide range of frequencies in the light spectrum. Some of the common frequencies are infrared, visible and ultraviolet.

A Light Sensor is also called as Photo Sensor or Photo electric Sensor as it converts light energy or photons in to electrical signals.

A photo resistor changes its resistance when light is incident on it. Hence, a photo resistor is also called as Light Dependent Resistor or LDR.

When there is no light, the resistance of LDR is very high. When there is light incident on the LDR, its resistance decreases.

Circuit Diagram



Hardware Required

- Arduino UNO
- Light Dependent Resistor (LDR)
- 100 KΩ POT
- Buzzer

Working of Arduino Light Sensor

Light Sensors are very useful devices in wide range of applications. One of the common application is an automatic night lamp, where a light bulb is automatically turned on as soon as the sun sets down. Another good application is solar tracker, which tracks the sun and rotates the solar panel accordingly.

All these applications use a simple photo resistor or an LDR as the main sensing device. Hence, in this project, we designed a simple light sensor that indicates when the light is indicated. The working of the project is very simple and is explained below.

All the connections are made as per the circuit diagram. The code for Arduino is written and dumped in the board. When the LDR detects a light over certain intensity, the Arduino will trigger the buzzer. When the intensity of light decreases, the buzzer is turned off.

The 100 K Ω POT used in the voltage divider network can be used to adjust the intensity levels at which the buzzer is triggered.

Code

```
int sensorPin = A0; // select the input pin for the potentiometer
//int ledPin = 13; // select the pin for the LED
int sensorValue = 0; // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
  pinMode(11,OUTPUT);
}

void loop()
{
  sensorValue=analogRead(sensorPin);
  if(sensorValue <= 14)
    digitalWrite(11,HIGH);
  else
    digitalWrite(11,LOW);
  Serial.println(sensorValue);
  delay(2);
}
```

Applications

- Light Sensors are used in variety of applications.
- They can be used in security systems like burglar alarm systems where an alarm is triggered when the light falling on the sensor is interrupted.

- Another common application of light sensor is night lamp. As long as the sun light falls on the light sensor, the lamp will be switched off. When the sun light starts decreasing and is completely off, the lamp will be turned on automatically.
- One of the important applications of light sensors is in generation of efficient solar energy. Light sensors are often used in Solar Tracking systems. The solar panel will be rotated according to the movement of the sun and its intensity.

Arduino - DC Motor

There are three different type of motors –

- DC motor
- Servo motor
- Stepper motor

A DC motor (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative.

Do not drive the motor directly from Arduino board pins. This may damage the board. Use a driver Circuit or an IC.



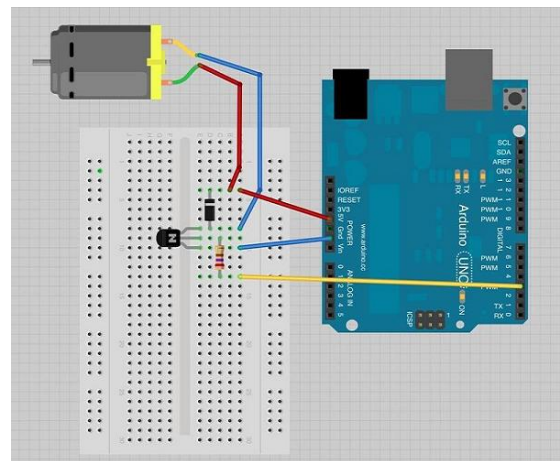
Procedure

Follow the circuit diagram and make the connections as shown in the image given below.

Precautions

Take the following precautions while making the connections.

- First, make sure that the transistor is connected in the right way. The flat side of the transistor should face the Arduino board as shown in the arrangement.
- Second, the striped end of the diode should be towards the +5V power line according to the arrangement shown in the image.



Spin Control Arduino Code

```
int motorPin = 3;
void setup() {
}
void loop() {
```

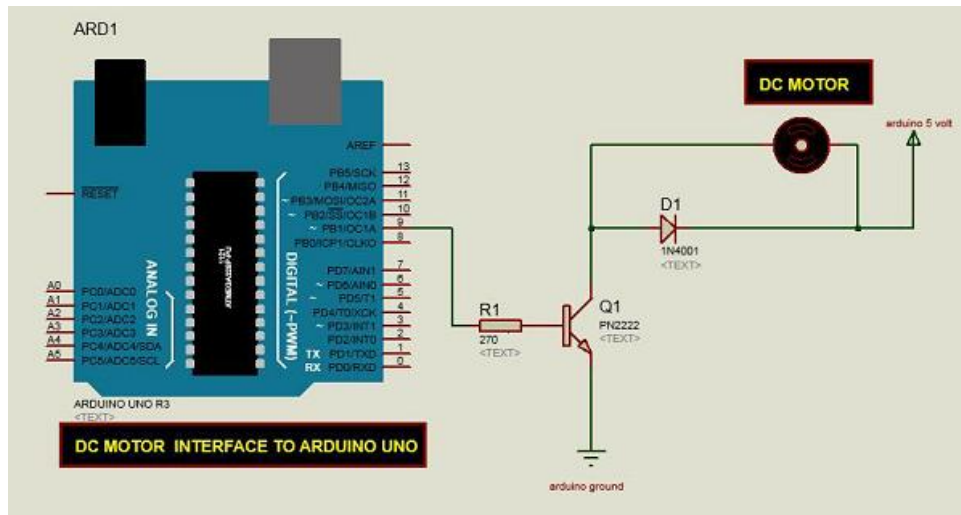
```
digitalWrite(motorPin, HIGH);
}
```

Code to Note

The transistor acts like a switch, controlling the power to the motor. Arduino pin 3 is used to turn the transistor on and off and is given the name 'motorPin' in the sketch. Motor will spin in full speed when the Arduino pin number 3 goes high.

Motor Speed Control

Following is the schematic diagram of a DC motor, connected to the Arduino board.



Arduino Code

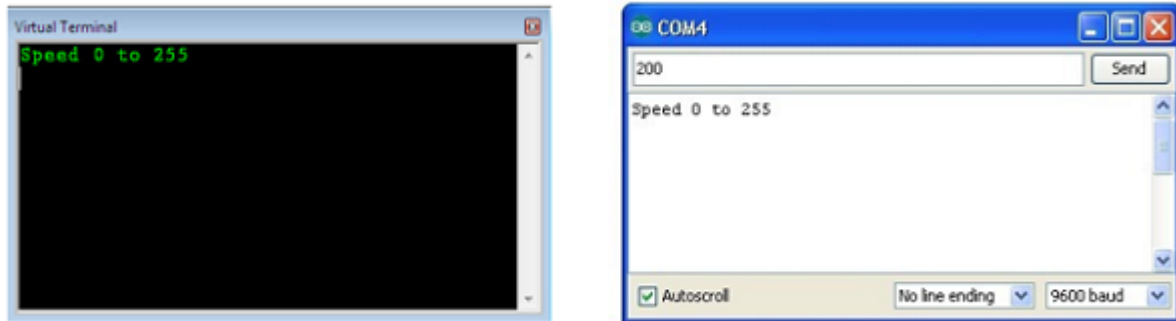
```
int motorPin = 9;
```

```
void setup() {
  pinMode(motorPin, OUTPUT);
  Serial.begin(9600);
  while (! Serial);
  Serial.println("Speed 0 to 255");
}

void loop() {
  if (Serial.available()) {
    int speed = Serial.parseInt();
    if (speed >= 0 && speed <= 255) {
      analogWrite(motorPin, speed);
    }
  }
}
```

Code to Note

The transistor acts like a switch, controlling the power of the motor. Arduino pin 3 is used to turn the transistor on and off and is given the name 'motorPin' in the sketch. When the program starts, it prompts you to give the values to control the speed of the motor. You need to enter a value between 0 and 255 in the Serial Monitor.



The DC motor will spin with different speeds according to the value (0 to 250) received via the serial port.

Interface L298N DC Motor Driver Module with Arduino

The L298N motor driver can control the speed and spinning direction of two DC motors. In addition, it can control a bipolar stepper motor.

Controlling a DC Motor

Full control over a DC motor if one can control its speed and spinning direction. This is possible by combining these two techniques.

- PWM – to control speed
- H-Bridge – to control the spinning direction

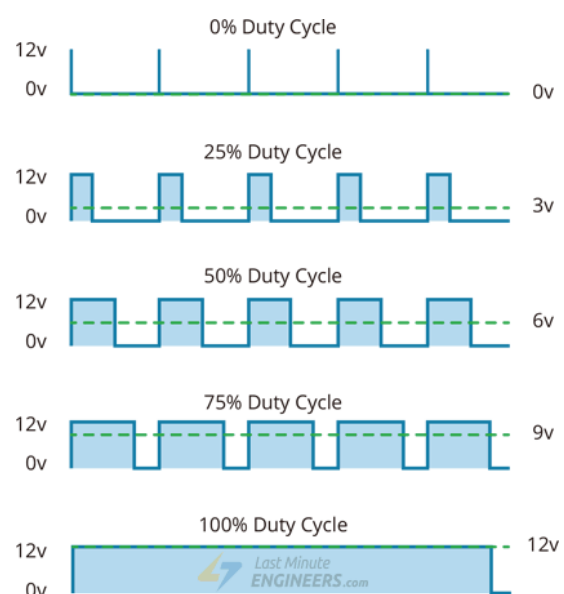
PWM – to control speed

The speed of a DC motor can be controlled by changing its input voltage. A widely used technique to accomplish this is Pulse Width Modulation (PWM).

PWM is a technique in which the average value of the input voltage is adjusted by sending a series of ON-OFF pulses. This average voltage is proportional to the width of the pulses, which is referred to as the Duty Cycle.

The higher the duty cycle, the higher the average voltage applied to the DC motor, resulting in an increase in motor speed. The shorter the duty cycle, the lower the average voltage applied to the DC motor, resulting in a decrease in motor speed.

The image below shows PWM technique with various duty cycles and average



voltages.

Pulse Width Modulation (PWM) Technique

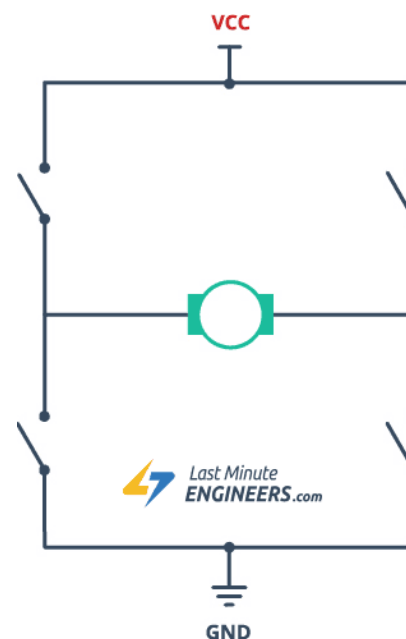
H-Bridge – to control the spinning direction

The spinning direction of a DC motor can be controlled by changing the polarity of its input voltage. A widely used technique to accomplish this is to use an H-bridge.

An H-bridge circuit is made up of four switches arranged in a H shape, with the motor in the center.

Closing two specific switches at the same time reverses the polarity of the voltage applied to the motor. This causes a change in the spinning direction of the motor.

The following animation shows the working of the H-bridge circuit.



L298N Motor Driver Chip

At the center of the module is a big, black chip with a chunky heat sink – the L298N.

The L298N chip contains two standard H-bridges capable of driving a pair of DC motors, making it ideal for building a two-wheeled robotic platform.

The L298N motor driver has a supply range of 5V to 35V and is capable of 2A continuous current per channel, so it works very well with most of our DC motors.

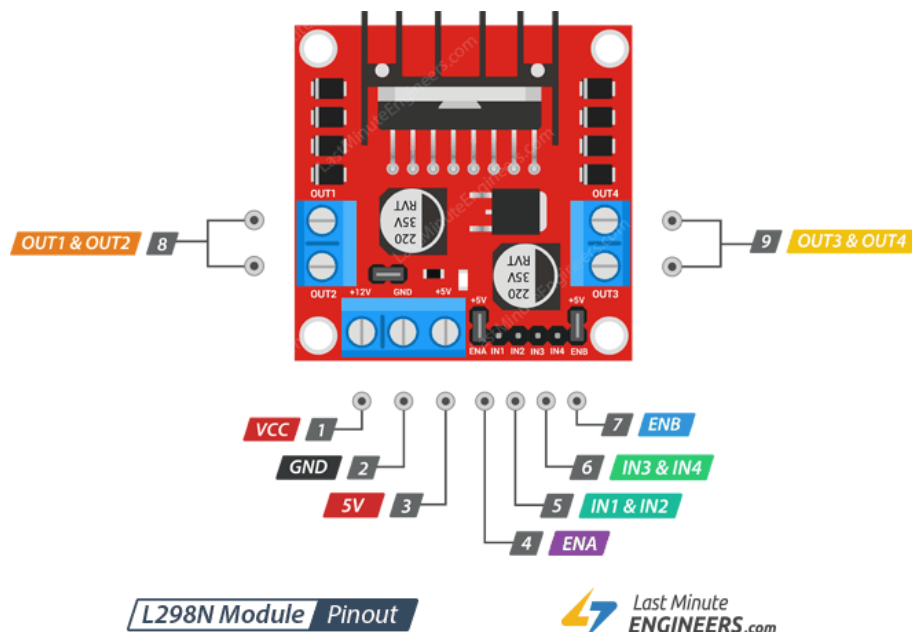
Technical Specifications

Motor output voltage	5V – 35V
Motor output voltage (Recommended)	7V – 12V
Logic input voltage	5V – 7V
Continuous current per channel	2A
Max Power Dissipation	25W

L298N Motor Driver Module Pinout

The L298N module has 11 pins that allow it to communicate with the outside world.

The pinout is as follows:



Power Pins: The L298N motor driver module receives power from a 3-pin, 3.5mm-pitch screw terminal.

The L298N motor driver has two input power pins: VS and VSS.

VS pin powers the IC's internal H-Bridge, which drives the motors. This pin accepts input voltages ranging from 5 to 12V.

VSS is used to power the logic circuitry within the L298N IC, and can range between 5V and 7V.

GND is the common ground pin.

Output Pins : The output channels of the L298N motor driver, OUT1 and OUT2 for motor A and OUT3 and OUT4 for motor B, are broken out to the edge of the module with two 3.5mm-pitch screw terminals. You can connect two 5-12V DC motors to these terminals.

Each channel on the module can supply up to 2A to the DC motor. The amount of current supplied to the motor, however, depends on the capacity of the motor power supply.

Direction Control Pins

The direction control pins allow you to control whether the motor rotates forward or backward. These pins actually control the switches of the H-Bridge circuit within the L298N chip.

Input1	Input2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

Arduino Example Code

The sketch below will show you how to control the speed and spinning direction of a DC motor using the L298N Motor Driver. The sketch moves the motor in one direction for one revolution, then in the opposite direction. There is also some acceleration and deceleration involved.

When accelerating or decelerating the motor, you may hear it humming, especially at lower PWM values. This is normal; there is nothing to be concerned about. This happens because the DC motor requires a minimum amount of voltage to operate.

// Motor A connections

```
int enA = 9;
```

```
int in1 = 8;
```

```
int in2 = 7;
```

// Motor B connections

```
int enB = 3;
```

```
int in3 = 5;
```

```
int in4 = 4;
```

```
void setup() {
```

```
    // Set all the motor control pins to outputs
```

```
    pinMode(enA, OUTPUT);
```

```
    pinMode(enB, OUTPUT);
```

```
    pinMode(in1, OUTPUT);
```

```
    pinMode(in2, OUTPUT);
```

```
    pinMode(in3, OUTPUT);
```

```
    pinMode(in4, OUTPUT);
```

```
    // Turn off motors - Initial state
```

```
    digitalWrite(in1, LOW);
```

```
    digitalWrite(in2, LOW);
```

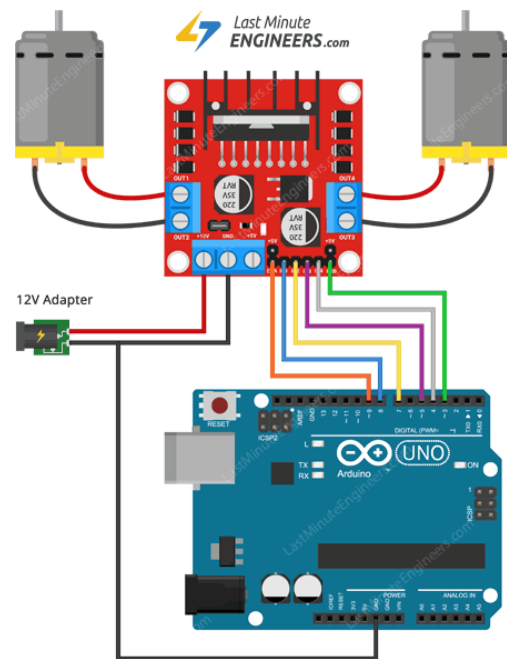
```
    digitalWrite(in3, LOW);
```

```
    digitalWrite(in4, LOW);
```

```
}
```

```
void loop() {
```

```
    directionControl();
```



```

        delay(1000);
        speedControl();
        delay(1000);
    }
    // This function lets you control spinning direction of motors
    void directionControl() {
        // Set motors to maximum speed
        // For PWM maximum possible values are 0 to 255
        analogWrite(enA, 255);
        analogWrite(enB, 255);
        // Turn on motor A & B
        digitalWrite(in1, HIGH);
        digitalWrite(in2, LOW);
        digitalWrite(in3, HIGH);
        digitalWrite(in4, LOW);
        delay(2000);
        // Now change motor directions
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
        digitalWrite(in3, LOW);
        digitalWrite(in4, HIGH);
        delay(2000);
        // Turn off motors
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, LOW);
    }
    // This function lets you control speed of the motors
    void speedControl() {
        // Turn on motors
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
        digitalWrite(in3, LOW);
        digitalWrite(in4, HIGH);
        // Accelerate from zero to maximum speed
        for (int i = 0; i < 256; i++) {
            analogWrite(enA, i);
            analogWrite(enB, i);
            delay(20);
        }
    }
}

```

```

    }
    // Decelerate from maximum speed to zero
    for (int i = 255; i >= 0; --i) {
        analogWrite(enA, i);
        analogWrite(enB, i);
        delay(20);
    }
    // Now turn off motors
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}

```

Code Explanation:

The Arduino code is fairly simple. It does not require any libraries to work. The sketch starts by declaring the Arduino pins that are connected to the L298N's control pins.

```
// Motor A connections
```

```
int enA = 9;
```

```
int in1 = 8;
```

```
int in2 = 7;
```

```
// Motor B connections
```

```
int enB = 3;
```

```
int in3 = 5;
```

```
int in4 = 4;
```

In the setup section of the code, all of the motor control pins, including the direction and speed control pins, are configured as digital OUTPUT. And the direction control pins are pulled LOW to initially disable both motors.

```

void setup() {
    // Set all the motor control pins to outputs
    pinMode(enA, OUTPUT);
    pinMode(enB, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);

    // Turn off motors - Initial state
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
}

```

```

        digitalWrite(in3, LOW);
        digitalWrite(in4, LOW);
    }

```

In the loop section of the code, Programmer call two user-defined functions with a one-second delay.

```

void loop() {
    directionControl();
    delay(1000);
    speedControl();
    delay(1000);
}

```

These functions are:

- directionControl() – This function causes both motors to spin at full speed for two seconds. It then reverses the spinning direction of the motors and spins for two seconds. Finally, it stops the motors.

```

void directionControl() {
    // Set motors to maximum speed
    // For PWM maximum possible values are 0 to 255
    analogWrite(enA, 255);
    analogWrite(enB, 255);

    // Turn on motor A & B
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
    delay(2000);

    // Now change motor directions
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);
    delay(2000);

    // Turn off motors
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}

```

```
}
```

- speedControl() – This function uses the analogWrite() function to generate a PWM signal that accelerates both motors from zero to maximum speed before decelerating them back to zero. Finally, it stops the motors.

```
void speedControl() {  
    // Turn on motors  
    digitalWrite(in1, LOW);  
    digitalWrite(in2, HIGH);  
    digitalWrite(in3, LOW);  
    digitalWrite(in4, HIGH);  
  
    // Accelerate from zero to maximum speed  
    for (int i = 0; i < 256; i++) {  
        analogWrite(enA, i);  
        analogWrite(enB, i);  
        delay(20);  
    }  
  
    // Decelerate from maximum speed to zero  
    for (int i = 255; i >= 0; --i) {  
        analogWrite(enA, i);  
        analogWrite(enB, i);  
        delay(20);  
    }  
  
    // Now turn off motors  
    digitalWrite(in1, LOW);  
    digitalWrite(in2, LOW);  
    digitalWrite(in3, LOW);  
    digitalWrite(in4, LOW);  
}
```

Arduino Servomotor using Potentiometer

The Potentiometer will be used to control the position of the servo motor. The connection will be similar to the last servo motor project, except the added Potentiometer.

Mini Servo Motor: It is defined as a tiny motor that can approximately rotate up to 180 degrees. It works similar to the usual servo motor, but smaller in size.

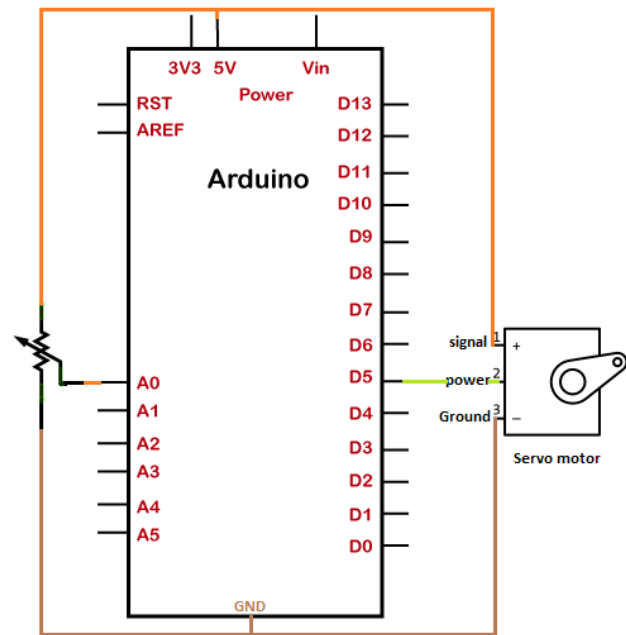
Principle

The project allows us to control the shaft at angles between 0 and 180 degrees. Programmer can also set the rotation of the shaft at different speeds. Servo motor has three terminals signal, power, and ground.

The power pin of the servo motor is connected to the PWM pin of the Arduino board. Here, we have connected the power terminal to pin 9 of the Arduino UNO R3 board.

Structure of the project

The structure of the connection or project is shown below:



Consider the below code:

1. `#include <Servo.h>`
2. `Servo myservo;`
3. `// It creates a servo object, which is used to control the servo`
4. `int potentiometerPIN = A0; // specified analog pin used to connect the potentiometer`
5. `int value; // value initialized to the variable to read the value from the analog pin`
6. `void setup()`
7. `{`
8. `myservo.attach(9); // servo connected to pin 9 of the Arduino board to the servo object`
9. `}`
10. `void loop()`
11. `{`
12. `value = analogRead(potentiometerPIN);`
13. `// reads the value of the potentiometer (value between 0 and 1023)`
14. `value = map(value, 0, 1023, 0, 180);`
15. `// scale it to use it with the servo (value between 0 and 180)`
16. `myservo.write(value);`

```

17. delay(1000); // it will wait for 1 second for the
18. // It will set the position of the motor according to the scaled value
19. value = map(value, 1023, 0, 180, 0);
20. // reads the value of the potentiometer (value between 1023 and 0)
21. myservo.write(value);
22. // scale it to use it with the servo (value between 180 and 0)
23. // the motor will rotate in reverse direction
24. delay(1000);
25. // delay time in milliseconds
26. //after 1500 millisecond it will again rotate from 0 to 180 degree
27. }

```

Connection

The steps to set up the connection are listed below:

- Connect the signal terminal of the servo motor to the 5V pin of the Arduino board.
- Connect the power terminal of the servo motor to pin 9 of the Arduino board. We can connect the power terminal of the motor to any digital PWM pin on the Arduino board.
- Connect the ground terminal of the servo motor to the GND pin of the Arduino board.
- One outer pin of the Potentiometer is connected to the ground (**GND**), and other external pin is connected to **5V** of the Arduino board.
- The middle terminal of the Potentiometer is connected to the analog input pin A0 of the board.

Output

The shaft will rotate at angles between 0 and 180 degrees and again in the reverse direction. Programmer can also modify the code by specifying it only in one direction from 0 to 180 degrees. Hence, we can make changes according to the requirements.

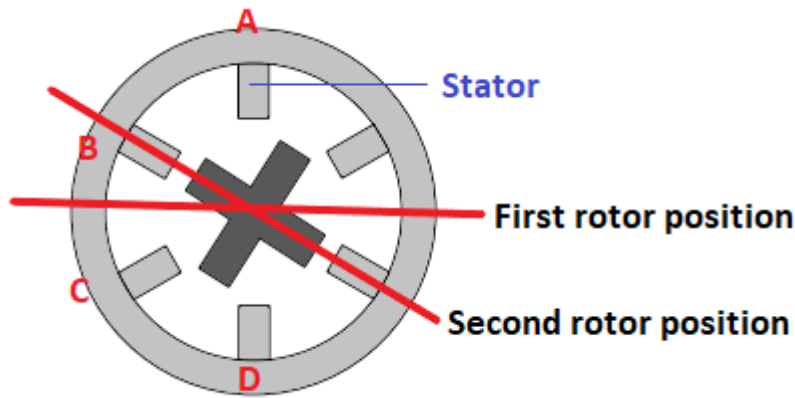
Arduino Stepper motor

The stepper motor does not require any feedback for its operation. It can be controlled with high accuracy due to its design.

The series of magnets mounted on the shaft of the stepper motor are controlled by the electromagnetic coils. These coils are negatively and positively charged in a sequence, which makes the shaft to move in forward and backward in little steps.

Stepper motor working

The stepper motor can control the angular position of the rotor without a closed feedback loop. For example, Consider a motor with six stator teeth and a rotor. It is shown below:

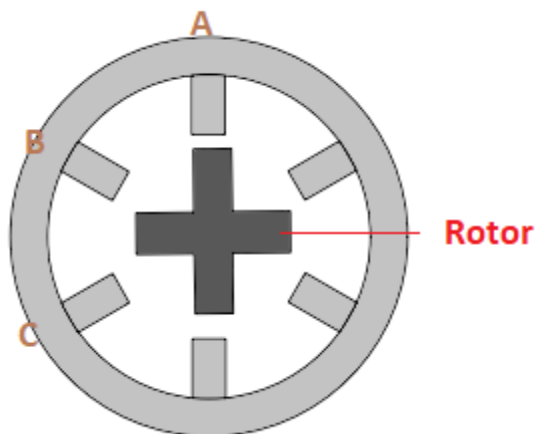


A stepper motor with six stator teeth can be triggered with three different DC power sources. The rotor in the stepper motor is made up of a stack of steel laminations. It has different teeth compared to the rotor, which is four.

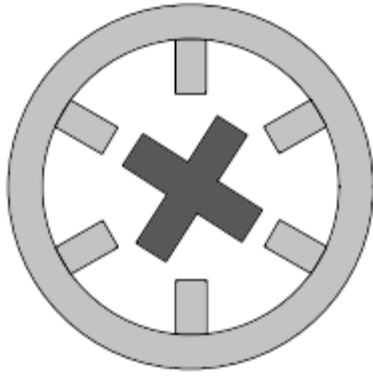
It is done so that one pair of rotor teeth at a time can be aligned easily with the stator.

If we trigger or energize the coil A and B, the rotor would rotate. The above figure signifies the step size is 30 degrees. We will energize coil B and C. After that, the coil A will energize again. It means that the rotor moves to the position with the least reluctance.

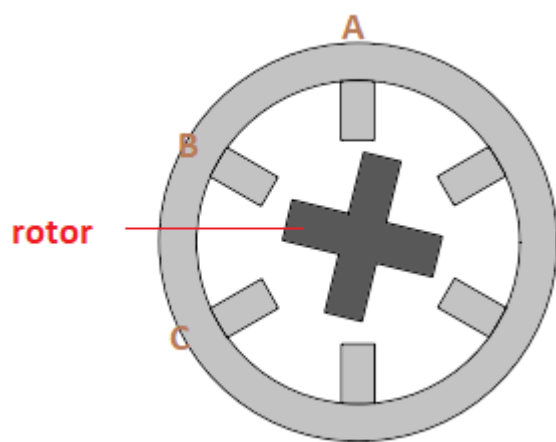
The position of the rotor, when coil A is energized is shown below:



The position of the rotor (moves 30 degrees), when coil B is energized is shown below:



When both the coils are excited, the position of the rotor (in between) is shown below:



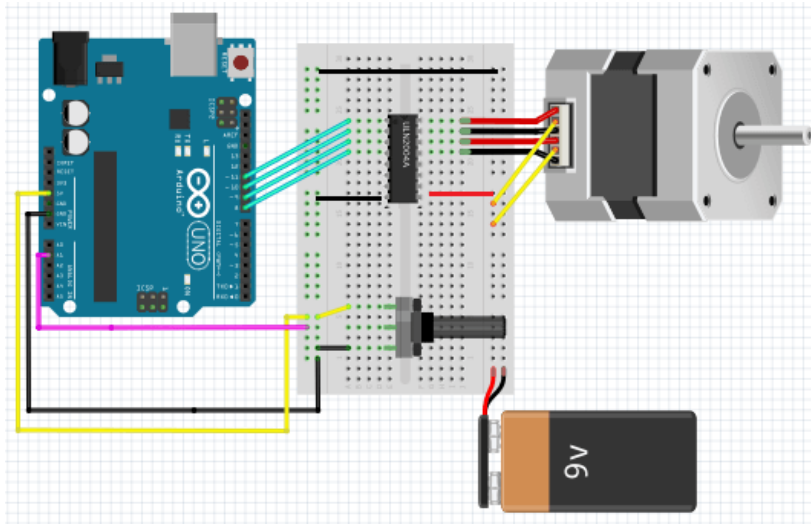
The energizing of both the coils change the accuracy of the rotor from 30 degrees to 15 degrees. The common stepper motor type is the hybrid motor type.

Hardware Required

The required components are listed below:

- 1 x Arduino UNO R3 (We can use any Arduino board)
- 1 x Breadboard
- Jump Wires
- 1 x 10K Ohm Potentiometer
- 1 x Stepper motor
- 1 x power supply (according to the stepper)
- U2004 Darlington Array (For a Unipolar stepper)
- SN754410ne H-Bridge (for a bipolar stepper)

Connection diagram



Sketch

Consider the below code:

```

1. #include <Stepper.h> //library declared for the operation of stepper motor
2. const int stepsPERrevolution = 200; // We can change it according to the req
   uired steps per revolution
3. // for our motor
4. // the initialization of pins 8 to 11 of stepper library
5. Stepper myStepper(stepsPERrevolution, 8, 9, 10, 11);
6. int CountofSTEP = 0; // number of steps the motor has taken
7. void setup()
8. {
9.   Serial.begin(9600);
10. }
11. void loop()
12. {
13.   // read the sensor value:
14.   int ReadingINSensor = analogRead(A1);
15.   // we can map it to a range from 0 to 100:
16.   int SpeedOFmotor = map(ReadingINSensor, 0, 1023, 0, 100);
17.   // to set the speed of the motor
18.   if (SpeedOFmotor > 0)
19.   {
20.     myStepper.setSpeed(SpeedOFmotor);
21.     // step 1/100 of a revolution
22.     myStepper.step(stepsPERrevolution/ 100);
23.   }
24. }

```

Procedure

The steps to establish the above connection are listed below:

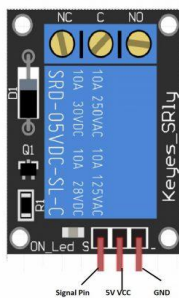
- Connect the negative and positive terminal of the battery to the GND and 5V pin of the Arduino board.
- One outer pin of the potentiometer is connected to ground (**GND**), and the other external pin is connected to **5V** of the Arduino board.
- The middle pin of the potentiometer is connected to the analog input pin A1 of the board.
- Connect the 8 to 11 digital pins of the Arduino board to the U2004 Darlington Array, which is further connected to the motor.
- Connect other pins of the U2004 Darlington Array to the stepper motor, as shown in the connection diagram.

Interfacing Relay with Arduino Uno

Relay Module

A relay is generally an electrically operated switch. The principle used by the relays is an electromagnet to mechanically operate the switch. So basically it operates a switch using an electromagnet which needs only less power like 5V, 12V or 24V. Different kinds of relays are available in the market like SPDT, DPDT, SPST, 5V, 12V, 24V and with various high current/voltage driving capacity. In this tutorial we are using a 5V relay module.

Pinouts



Relay Module

As shown in the above image :

- **NO : Normally Open** – This is the normally open terminal of the relay, means if we don't energise the relay there won't be any contact with Common terminal. But it will establish electrical contact with Common terminal once the relay is energized.
- **C** : Common terminal
- **NC : Normally Closed** – This is the normally closed terminal of the relay, means it will have electrical contact with common terminal whenever the relay is not energised. And there won't be electrical contact when the relay is energised.
- **GND** : Ground Pin
- **Signal** : Actuation signal to control the relay.
- **5V VCC** : Operating voltage for the relay.

Working

The principle behind the relay is electromagnetism, a switch operated by an electromagnet. So, a low voltage is enough for an electromagnet to get activated. This small voltage will be given to the relay by Arduino Uno or using some intermediate driver if required.

NO, NC and COM Terminals

Usually SPDT (Single Pole Double Throw) relays have 3 output terminals, these are the 3 terminals of internal SPDT electromagnetic switch.

Common (COM) : This is the commonly terminal. This terminal will be connected to either of other 2 terminals (NO or NC) based on the state of relay.

Normally Open (NO) : As the name indicates this is normally open terminal, ie. if the relay is not energized (not ON), this pin will be open. We can say that the switch is OFF by default and when the relay is energized it will become ON.

Normally Closed (NC) : As the name indicates it is normally closed terminal, ie. if the relay is not energized (not ON), this pin will be closed. We can say that the switch is ON by default and when the relay is energized it will become OFF.

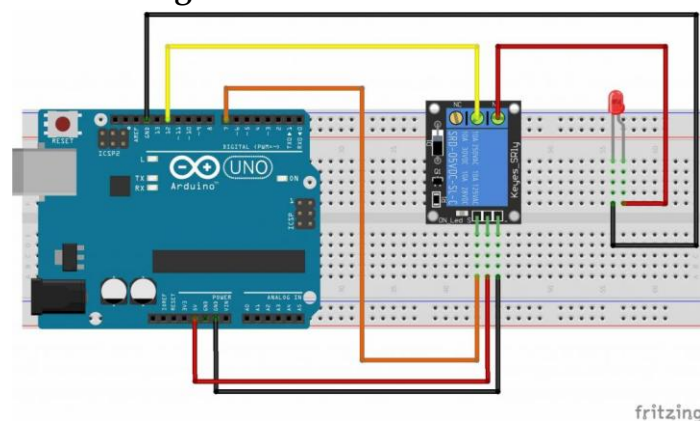
Interfacing Relay with Arduino Uno

In the first phase of this tutorial we are controlling a normal LED for testing the functionality of the relay as playing directly with AC needs to be very careful.

Components Required

- Arduino Uno
- LED
- 5V Relay Module
- Bread Board
- Jumper Wires

Circuit Diagram



Interfacing Relay Module with Arduino Uno

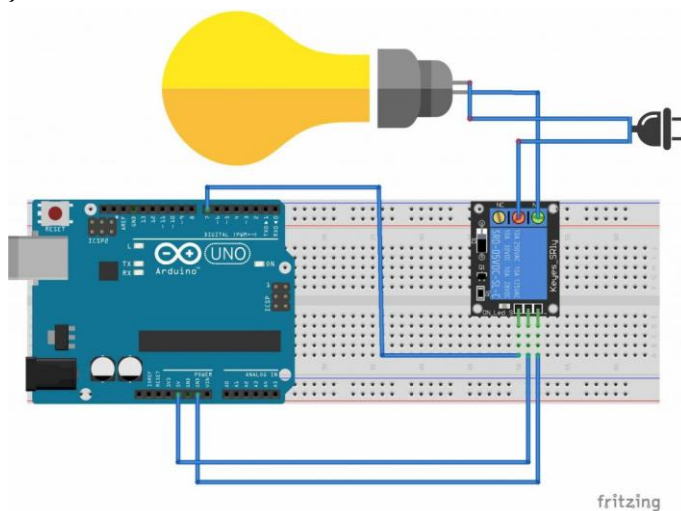
Description

- GND pin of 5V Relay – GND pin of Arduino
- Signal (Input) pin of 5V Relay – pin 7 of Arduino
- VCC pin of 5V Relay – 5V pin of Arduino

- Common pin of 5V Relay – pin 12 of Arduino
- NO pin of 5V Relay – Positive pin of the LED
- GND pin of LED – GND pin of Arduino

Program

```
int relay_pin = 7;
int led_pin = 12;
void setup()
{
  pinMode(relay_pin,OUTPUT);
  pinMode(led_pin,OUTPUT);
  digitalWrite(led_pin,HIGH);
}
void loop()
{
  digitalWrite(relay_pin,HIGH);
  delay(2000);
  digitalWrite(relay_pin,LOW);
  delay(2000);
}
```



Unit IV

ESP8266 NodeMCU

The ESP8266 is a Wi-Fi System on a Chip (SoC) produced by Espressif Systems. It's great for IoT and Home Automation projects.

Introducing the ESP8266 NodeMCU

The ESP8266 is a low-cost Wi-Fi chip developed by Espressif Systems. It can be used as a standalone device, or as a UART to Wi-Fi adaptor to allow other microcontrollers to connect to a Wi-Fi network. For example, ESP8266 connected to an Arduino to add Wi-Fi capabilities to Arduino board.

Advantages of ESP8266:

- **Low-cost:** ESP8266 boards starting at \$3 (or less) depending on the model.
- **Low-power:** the ESP8266 consumes very little power and can even go into deep sleep mode to consume less power;
- **Wi-Fi:** the ESP8266 can generate its own Wi-Fi network (access point) or connect to other Wi-Fi networks (station) to get access to the internet. It can also act as a web server.
- **Compatible with the Arduino “programming language”:** those that are already familiar with programming the Arduino board, they can program the ESP8266 in the Arduino style.
- **Compatible with MicroPython:** one can program the ESP8266 with MicroPython firmware, which is a re-implementation of Python 3 targeted for microcontrollers and embedded systems.

ESP8266 Technical Details:

For more details about the specs of the ESP8266, check the following list:

ESP8266-12E Wi-Fi chip

- Processor: L106 32-bit RISC microprocessor core based on the Tensilica Diamond Standard 106Micro running at 80 or 160 MHz
- Memory:
 - 32 KiB instruction RAM
 - 32 KiB instruction cache RAM
 - 80 KiB user-data RAM



- 16 KiB ETS system-data RAM
- External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
- IEEE 802.11 b/g/n Wi-Fi
- Integrated TR switch, balun, LNA, power amplifier, and matching network
- WEP or WPA/WPA2 authentication or open networks
- 17 GPIO pins
- Serial Peripheral Interface Bus (SPI)
- I²C (software implementation)
- I²S interfaces with DMA (sharing pins with GPIO)
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 10-bit ADC (successive approximation ADC)

ESP8266 Versions

There are several versions of the ESP8266 modules as shown in the picture below. The ESP-01 and ESP-12E are the most popular versions.



ESP8266 NodeMCU Development Boards

ESP8266 NodeMCU come with all the needed circuitry to apply power, upload code, easy access to the GPIOs to connect sensors and actuators, an antenna for the Wi-Fi signal, and other useful features.

- **USB-to-UART interface and voltage regulator circuit.** Most full-featured development boards have these two features. This is important to easily connect the ESP8266 to computer to upload code and apply power.
- **BOOT and RESET/EN buttons** to put the board in flashing mode or reset (restart) the board.
- **Pin configuration and the number of pins.** To properly use the ESP8266 in your projects, you need to have access to the board pinout (like a map that shows which pin corresponds to which GPIO and its features). So make sure you have access to the pinout of the board you're getting. Additionally, some boards have more accessible GPIOs than others. That's a factor you should take into account depending on your project features.
- **Size.** There is a wide variety of ESP8266 development boards with different sizes. Some boards benefit from a small form factor, which might be very practical depending on your project features. Usually, smaller boards have a small number of available GPIOs like the ESP-01.
- **Antenna connector.** Most boards come with an onboard antenna for the Wi-Fi signal. Some boards come with an antenna connector to optionally connect an external antenna. Adding an external antenna increases your Wi-Fi range.

ESP8266 NodeMCU Pinout

The most widely used ESP8266 NodeMCU development boards are the ESP8266-12E NodeMCU Kit. A pinout is like a map that shows which pin corresponds to which GPIO and its features.

Power Pins

Usually, all boards come with power pins: 3V3, GND, and V_{IN} . Programmer can use these pins to power the board, or to get power for other peripherals.

General Purpose Input Output Pins (GPIOs)

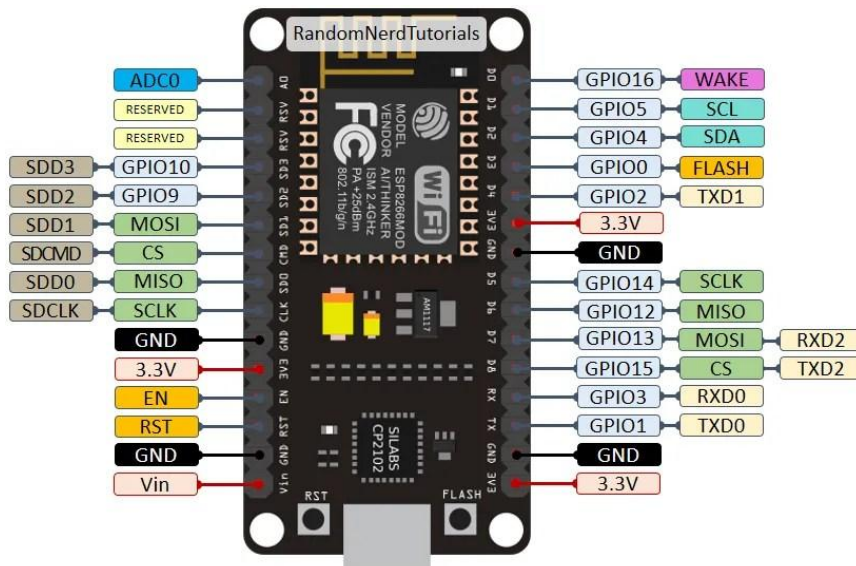
One important thing to notice about the ESP8266 is that the GPIO number doesn't match the label on the board silkscreen. For example, D0 corresponds to GPIO16 and D1 corresponds to GPIO5. When programming boards using Arduino IDE, Programmer must use the GPIO number and not the number on the silkscreen. This applies to most ESP8266 boards.

The ESP8266 peripherals include:

- 17 GPIOs (usually not all GPIOs are accessible on the ESP8266 development boards)
- SPI
- I²C (implemented on software)
- I²S interfaces with DMA
- UART
- 10-bit ADC

ESP-12E NodeMCU Kit Pinout

The following picture shows an overview of the ESP-12E NodeMCU Kit pinout:



Program the ESP8266

There are many different ways to program the ESP8266 using different programming languages: Arduino C/C++ using the Arduino core for the ESP32, Micropython, LUA, and others.

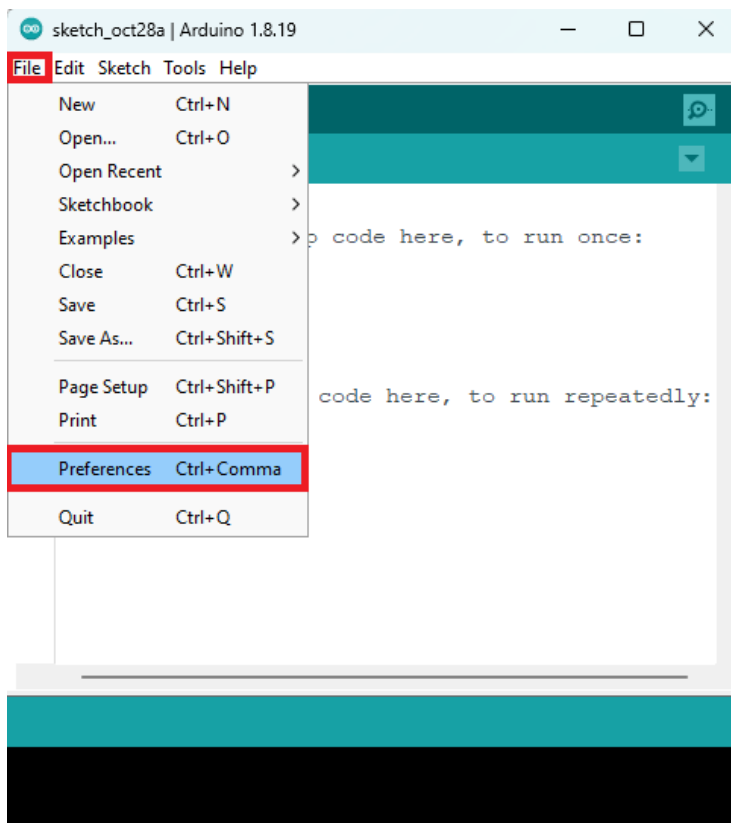
Programming ESP8266 with Arduino IDE

To program the boards, Programmer need an IDE to write the code. Arduino IDE works well and is simple and intuitive to use for beginners.

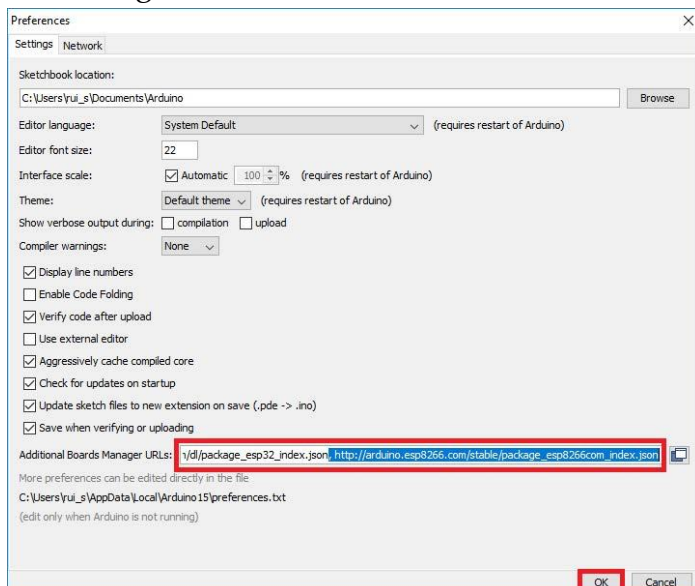
Installing the ESP8266 NodeMCU in Arduino IDE

To be able to program the ESP8266 NodeMCU using Arduino IDE, programmer need to add support for the ESP8266 boards. Follow the next steps:

1. Go to **File > Preferences**.



2. Enter the following into the “*Additional Board Manager URLs*” field.
https://arduino.esp8266.com/stable/package_esp8266com_index.json
See the figure below. Then, click the “**OK**” button.



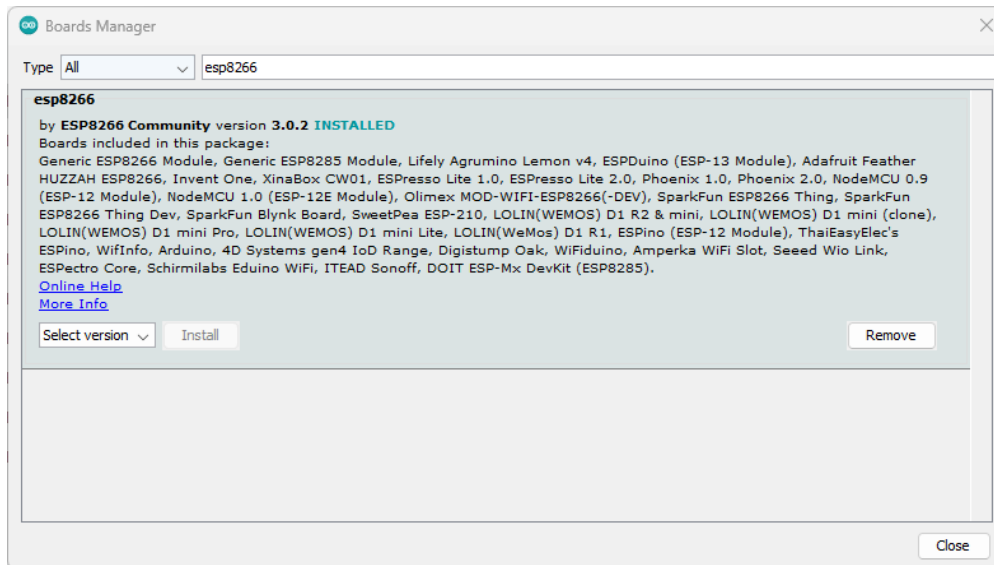
Note: if programmer already has the ESP32 boards URL, they can separate the URLs with a comma as follows:

https://dl.espressif.com/dl/package_esp32_index.json,

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

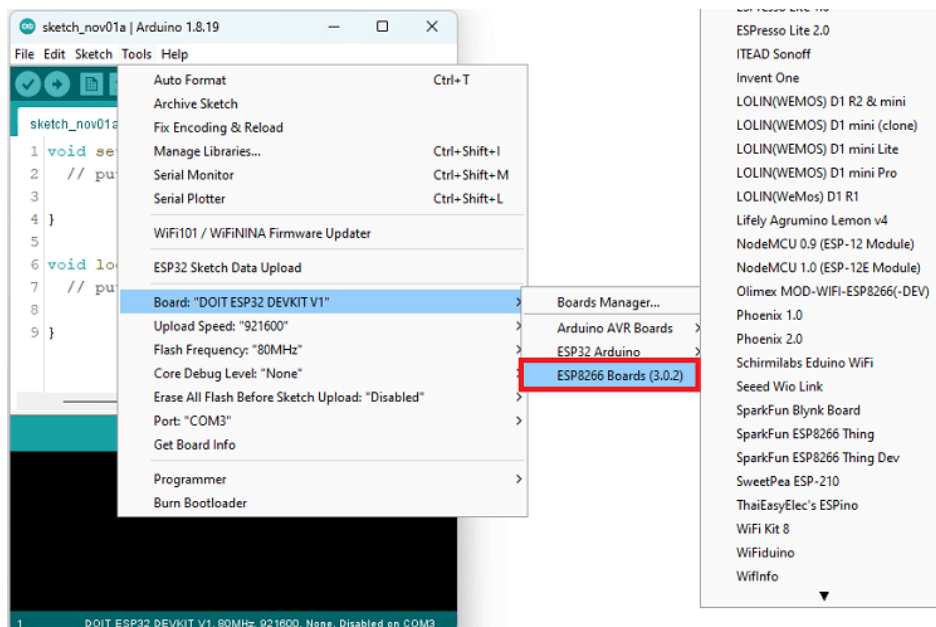
3. Open the **Boards Manager**. Go to **Tools > Board > Boards Manager...**
4. Search for **ESP8266** and install the “**ESP8266 by ESP8266 Community**”.

That's it. It will be installed after a few seconds.



After this, restart the Arduino IDE.

Then, go to **Tools > Board** and check that you have ESP8266 boards available.



Now, programmer is ready to start programming your ESP8266 using Arduino IDE.

ESP8266WiFi library

ESP8266 is all about Wi-Fi. If you are eager to connect your new ESP8266 module to a Wi-Fi network to start sending and receiving data, this is a good place to start. If you are looking for more in depth details of how to program specific Wi-Fi networking functionality, you are also in the right place.

Introduction

The Wi-Fi library for ESP8266 has been developed based on ESP8266 SDK, using the naming conventions and overall functionality philosophy of the Arduino WiFi library. Over time, the wealth of Wi-Fi features ported from ESP8266 SDK to esp8266 / Arduino outgrew Arduino WiFi library and it became apparent that we would need to provide separate documentation on what is new and extra.

This documentation will walk you through several classes, methods and properties of the ESP8266WiFi library. If you are new to C++ and Arduino, don't worry. We will start from general concepts and then move to detailed description of members of each particular class including usage examples.

The scope of functionality offered by the ESP8266WiFi library is quite extensive and therefore this description has been broken up into separate documents marked with :arrow_right:.

Quick Start

Hopefully, you are already familiar how to load the [Blink.ino](#) sketch to an ESP8266 module and get the LED blinking. If not, please use this tutorial by Adafruit or another great tutorial developed by Sparkfun.

To hook up the ESP module to Wi-Fi (like hooking up a mobile phone to a hot spot), you need only a couple of lines of code:

```
#include <ESP8266WiFi.h>

void setup()
{
    Serial.begin(115200);

    Serial.println();

    WiFi.begin("network-name", "pass-to-network");

    Serial.print("Connecting");

    while (WiFi.status() != WL_CONNECTED)
```

```

{
  delay(500);

  Serial.print(".");
}

Serial.println();

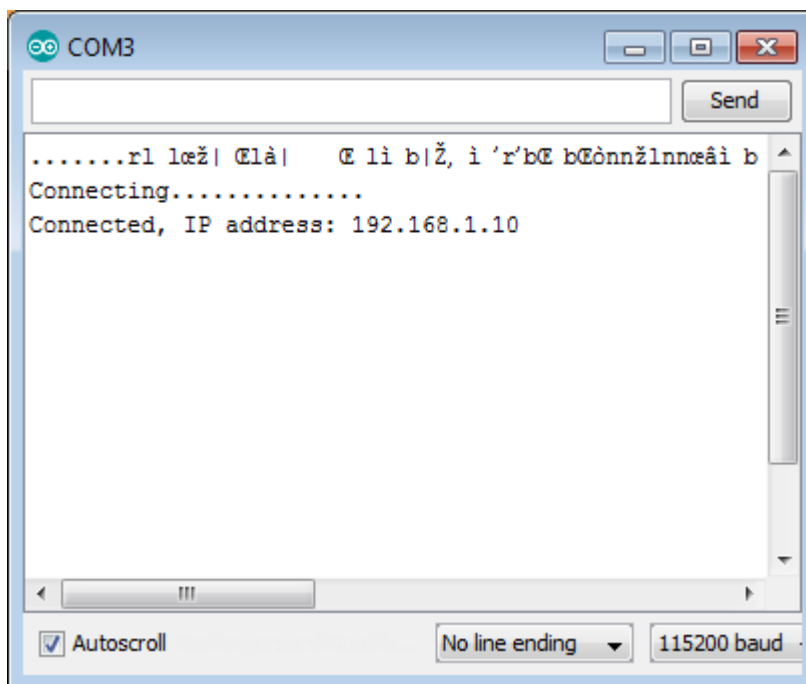
Serial.print("Connected, IP address: ");

Serial.println(WiFi.localIP());
}

void loop() {}

```

In the line `WiFi.begin("network-name", "pass-to-network")` replace network-name and pass-to-network with the name and password of the Wi-Fi network you would like to connect to. Then, upload this sketch to ESP module and open the serial monitor. You should see something like:



How does it work? In the first line of the sketch, `#include <ESP8266WiFi.h>` we are including the ESP8266WiFi library. This library provides ESP8266 specific Wi-Fi routines that we are calling to connect to the network.

The actual connection to Wi-Fi is initialized by calling:

```
WiFi.begin("network-name", "pass-to-network");
```

The connection process can take couple of seconds and we are checking for whether this has completed in the following loop:

```
while (WiFi.status() != WL_CONNECTED)
{
    delay(500);

    Serial.print(".");
}
```

The while() loop will keep looping as long as WiFi.status() is other than WL_CONNECTED. The loop will exit only if the status changes to WL_CONNECTED.

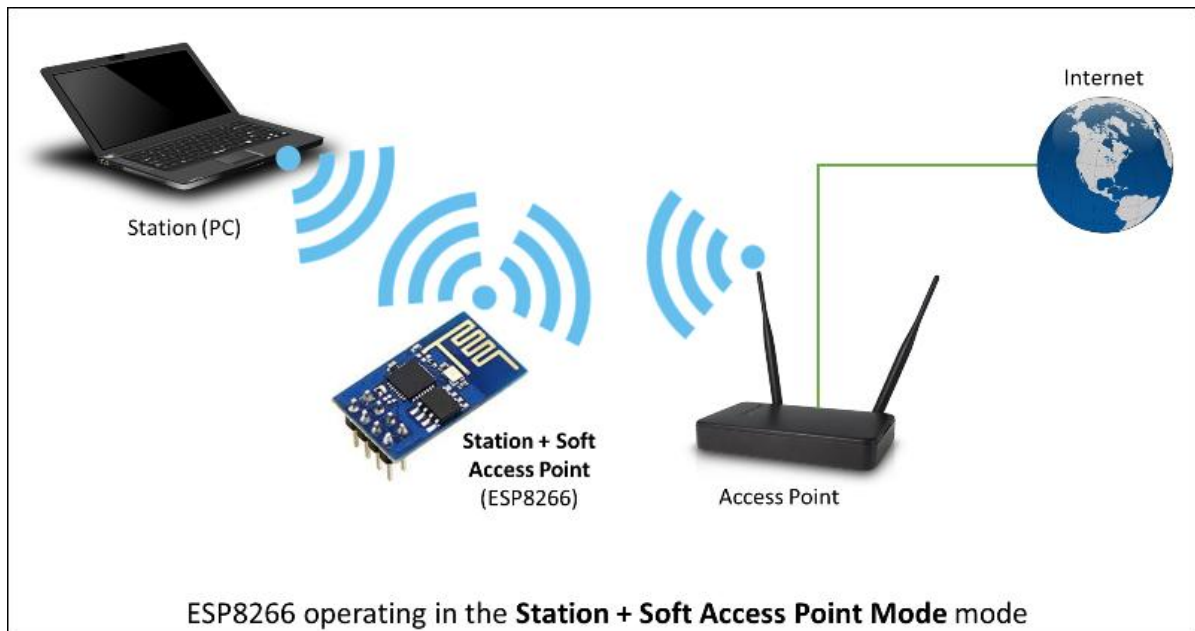
The last line will then print out the IP address assigned to the ESP module by DHCP:

```
Serial.println(WiFi.localIP());
```

Note: if connection is established, and then lost for some reason, ESP will automatically reconnect to the last used access point once it is again back on-line. This will be done automatically by Wi-Fi library, without any user intervention.

Devices that connect to Wi-Fi networks are called stations (STA). Connection to Wi-Fi is provided by an access point (AP), that acts as a hub for one or more stations. The access point on the other end is connected to a wired network. An access point is usually integrated with a router to provide access from a Wi-Fi network to the internet. Each access point is recognized by a SSID (**S**ervice **S**et **I**Dentifier), that essentially is the name of network you select when connecting a device (station) to the Wi-Fi.

ESP8266 modules can operate as a station, so we can connect it to the Wi-Fi network. It can also operate as a soft access point (soft-AP), to establish its own Wi-Fi network. When the ESP8266 module is operating as a soft access point, we can connect other stations to the ESP module. ESP8266 is also able to operate as both a station and a soft access point mode. This provides the possibility of building e.g. mesh networks.



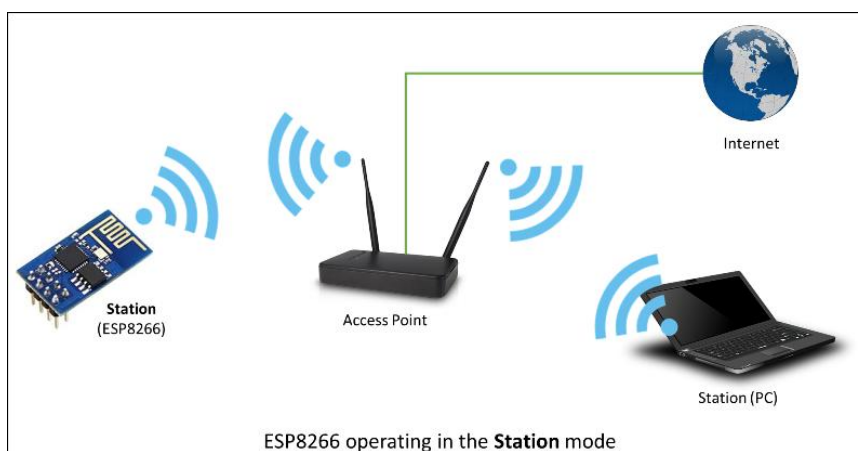
The ESP8266WiFi library provides a wide collection of C++ methods (functions) and properties to configure and operate an ESP8266 module in station and / or soft access point mode. They are described in the following chapters.

Class Description

The ESP8266WiFi library is broken up into several classes. In most of cases, when writing the code, the user is not concerned with this classification. We are using it to break up description of this library into more manageable pieces.

Station

Station (STA) mode is used to get the ESP module connected to a Wi-Fi network established by an access point.



Station class has several features to facilitate the management of a Wi-Fi connection. In case the connection is lost, the ESP8266 will automatically reconnect to the last

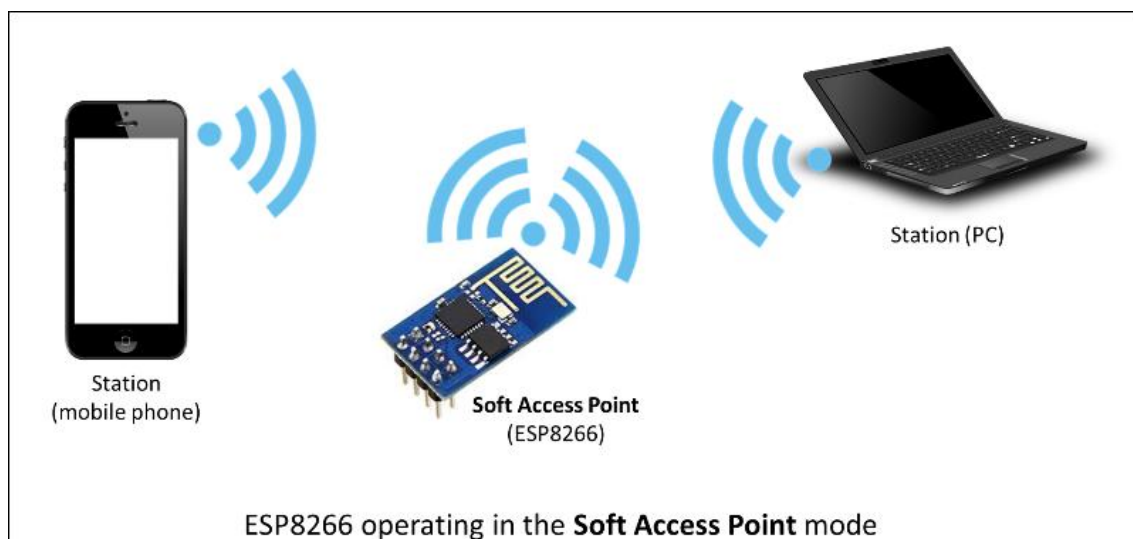
used access point, once it is available again. The same happens on module reboot. This is possible since ESP saves the credentials to the last used access point in flash (non-volatile) memory. Using the saved data ESP will also reconnect if sketch has been changed but code does not alter the Wi-Fi mode or credentials.

Station Class documentation

Check out separate section with examples.

Soft Access Point

An access point (AP) is a device that provides access to a Wi-Fi network to other devices (stations) and connects them to a wired network. The ESP8266 can provide similar functionality, except it does not have interface to a wired network. Such mode of operation is called soft access point (soft-AP). The maximum number of stations that can simultaneously be connected to the soft-AP can be set from 0 to 8, but defaults to 4.



The soft-AP mode is often used as an intermediate step before connecting ESP to a Wi-Fi in a station mode. This is when SSID and password to such network is not known upfront. ESP first boots in soft-AP mode, so we can connect to it using a laptop or a mobile phone. Then we are able to provide credentials to the target network. Then, the ESP is switched to the station mode and can connect to the target Wi-Fi.

Another handy application of soft-AP mode is to set up mesh networks. The ESP can operate in both soft-AP and Station mode so it can act as a node of a mesh network.

Soft Access Point Class documentation

Check out the separate section with examples.

Scan

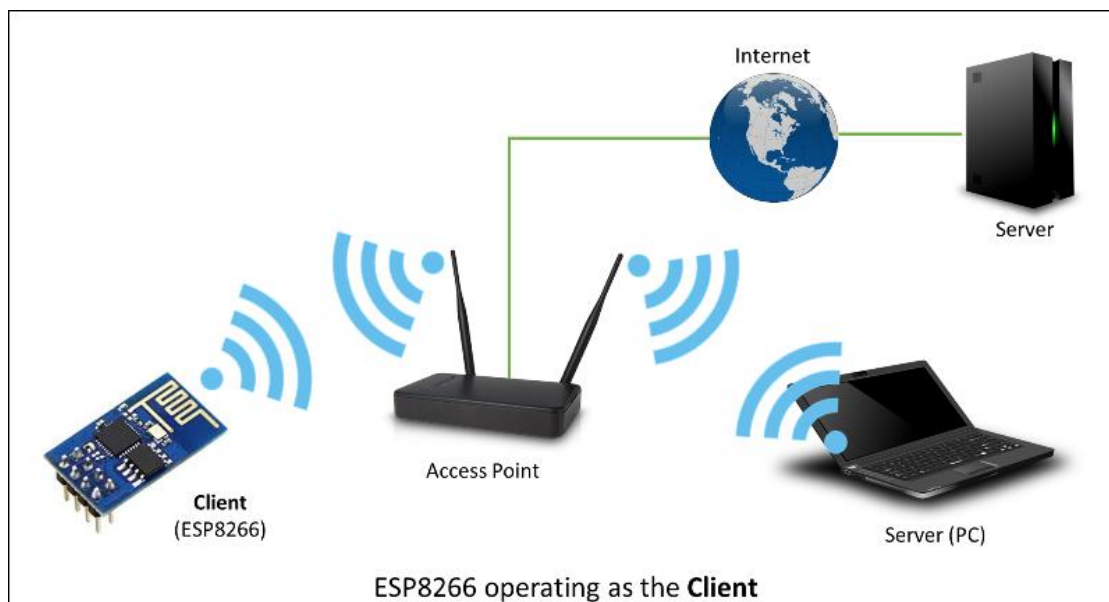
To connect a mobile phone to a hot spot, you typically open Wi-Fi settings app, list available networks and pick the hot spot you need. Then enter a password (or not) and you are in. You can do the same with the ESP. Functionality of scanning for, and listing of available networks in range is implemented by the Scan Class.

Scan Class documentation

Check out the separate section with examples.

Client

The Client class creates clients that can access services provided by servers in order to send, receive and process data.



Check out the separate section with list of functions

WiFi Multi

ESP8266WiFiMulti.h can be used to connect to a WiFi network with strongest WiFi signal (RSSI). This requires registering one or more access points with SSID and password. It automatically switches to another WiFi network when the WiFi connection is lost.

Example:

```
#include <ESP8266WiFiMulti.h>
```

```
ESP8266WiFiMulti wifiMulti;
```

```

//WiFi connect timeout per AP. Increase when connecting takes longer.

const uint32_t connectTimeoutMs = 5000;

void setup()
{
    // Set in station mode
    WiFi.mode(WIFI_STA);

    // Register multi WiFi networks
    wifiMulti.addAP("ssid_from_AP_1", "your_password_for_AP_1");
    wifiMulti.addAP("ssid_from_AP_2", "your_password_for_AP_2");
    wifiMulti.addAP("ssid_from_AP_3", "your_password_for_AP_3");
}

void loop()
{
    // Maintain WiFi connection

    if (wifiMulti.run(connectTimeoutMs) == WL_CONNECTED) {
        ...
    }
}

```

BearSSL Client Secure and Server Secure

BearSSL::WiFiClientSecure and *BearSSL::WiFiServerSecure* are extensions of the standard Client and Server classes where connection and data exchange with servers and clients using secure protocol. It supports TLS 1.2 using a wide variety of modern ciphers, hashes, and key types.



Secure clients and servers require significant amounts of additional memory and processing to enable their cryptographic algorithms. In general, only a single secure client or server connection at a time can be processed given the little RAM present on the ESP8266, but there are methods of reducing this RAM requirement detailed in the relevant sections.

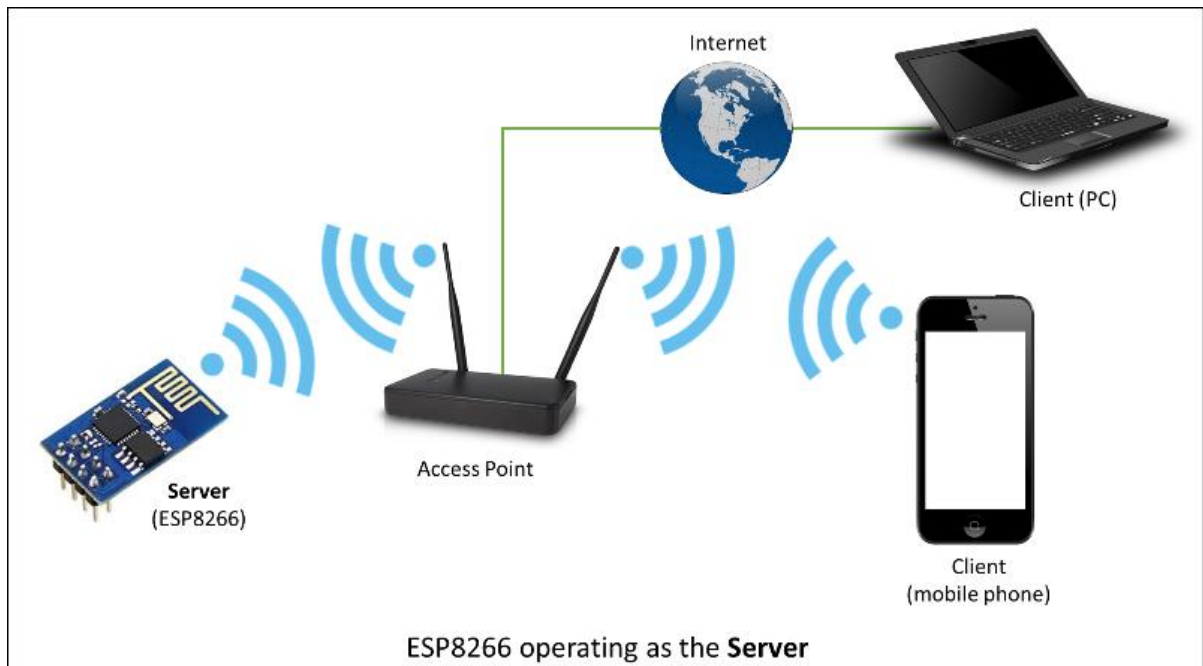
[BearSSL::WiFiClientSecure](#) contains more information on using and configuring TLS connections.

[BearSSL::WiFiServerSecure](#) discusses the TLS server mode available. Please read and understand the [BearSSL::WiFiClientSecure](#) first as the server uses most of the same concepts.

Check out the separate section with examples .

Server

The `Server` Class creates servers that provide functionality to other programs or devices, called clients.



Clients connect to sever to send and receive data and access provided functionality.

UDP

The UDP Class enables the User Datagram Protocol (UDP) messages to be sent and received. The UDP uses a simple “fire and forget” transmission model with no guarantee of delivery, ordering, or duplicate protection. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram.

Generic

There are several functions offered by ESP8266’s SDK and not present in Arduino WiFi library. If such function does not fit into one of classes discussed above, it will likely be in Generic Class. Among them is handler to manage Wi-Fi events like connection, disconnection or obtaining an IP, Wi-Fi mode changes, functions to manage module sleep mode, hostname to an IP address resolution, etc.

Diagnostics

There are several techniques available to diagnose and troubleshoot issues with getting connected to Wi-Fi and keeping connection alive.

Check Return Codes

Almost each function described in chapters above returns some diagnostic information.

Such diagnostic may be provided as a simple boolean type true or false to indicate operation result. You may check this result as described in examples, for instance:

```
Serial.printf("Wi-Fi mode set to WIFI_STA %s\n", WiFi.mode(WIFI_STA) ? "" : "Failed!");
```

Some functions provide more than just a binary status information. A good example is `WiFi.status()`.

```
Serial.printf("Connection status: %d\n", WiFi.status());
```

This function returns following codes to describe what is going on with Wi-Fi connection:

- 0 : WL_IDLE_STATUS when Wi-Fi is in process of changing between statuses
- 1 : WL_NO_SSID_AVAIL in case configured SSID cannot be reached
- 3 : WL_CONNECTED after successful connection is established
- 4 : WL_CONNECT_FAILED if connection failed
- 6 : WL_CONNECT_WRONG_PASSWORD if password is incorrect
- 7 : WL_DISCONNECTED if module is not configured in station mode

It is a good practice to display and check information returned by functions. Application development and troubleshooting will be easier with that.

Use `printDiag`

There is a specific function available to print out key Wi-Fi diagnostic information:

```
WiFi.printDiag(Serial);
```

A sample output of this function looks as follows:

Mode: STA+AP

PHY mode: N

Channel: 11

AP id: 0

Status: 5

Auto connect: 1

SSID (10): sensor-net

Passphrase (12): 123!\$#0&*esP

BSSID set: 0

Use this function to provide snapshot of Wi-Fi status in these parts of application code, that you suspect may be failing.

Enable Wi-Fi Diagnostic

By default the diagnostic output from Wi-Fi libraries is disabled when you call `Serial.begin`. To enable debug output again, call `Serial.setDebugOutput(true)`. To redirect debug output to `Serial1` instead, call `Serial1.setDebugOutput(true)`.

Connecting

state: 0 -> 2 (b0)

state: 2 -> 3 (0)

state: 3 -> 5 (10)

add 0

aid 1

cnt

connected **with** sensor-net, channel 6

dhcp client start...

chg_B1:-40

...ip:192.168.1.10,mask:255.255.255.0,gw:192.168.1.9

.Connected, IP address: 192.168.1.10

The same sketch without `Serial.setDebugOutput(true)` will print out only the following:

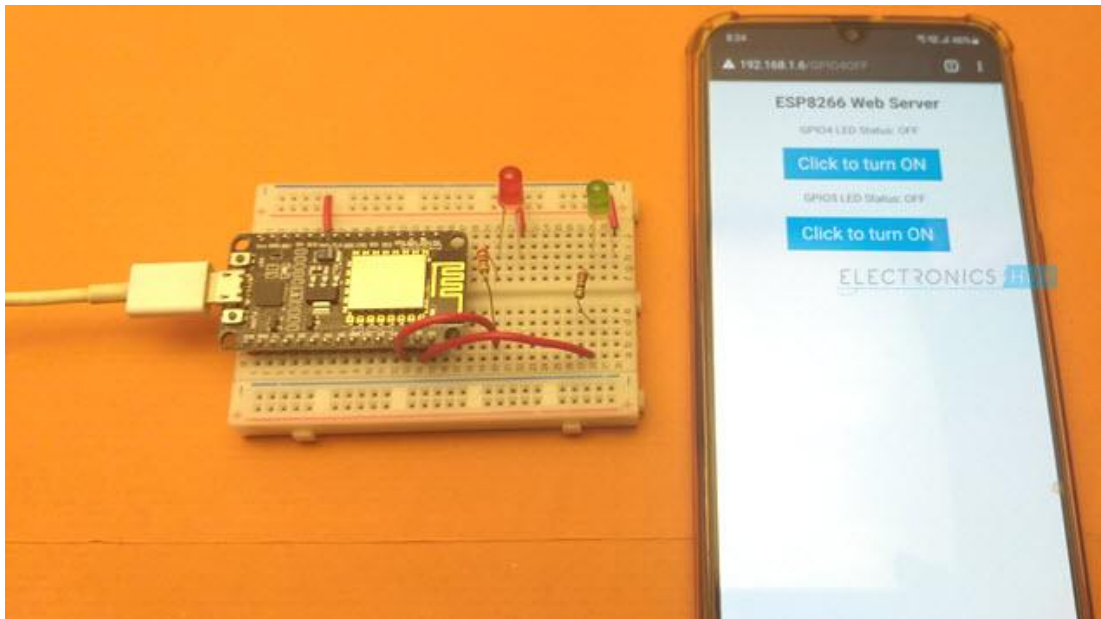
Connecting....

Connected, IP address: 192.168.1.10

To Create ESP8266 Web Server

In this tutorial, I will show How to Build a Simple ESP8266 Web Server. This ESP8266 NodeMCU standalone Web Server can be accessed by any device in the

local network that has a web browser (Mobiles, Laptops, Tablets). To demonstrate the working of the web server is ESP8266, we will create a web page which controls two LEDs.

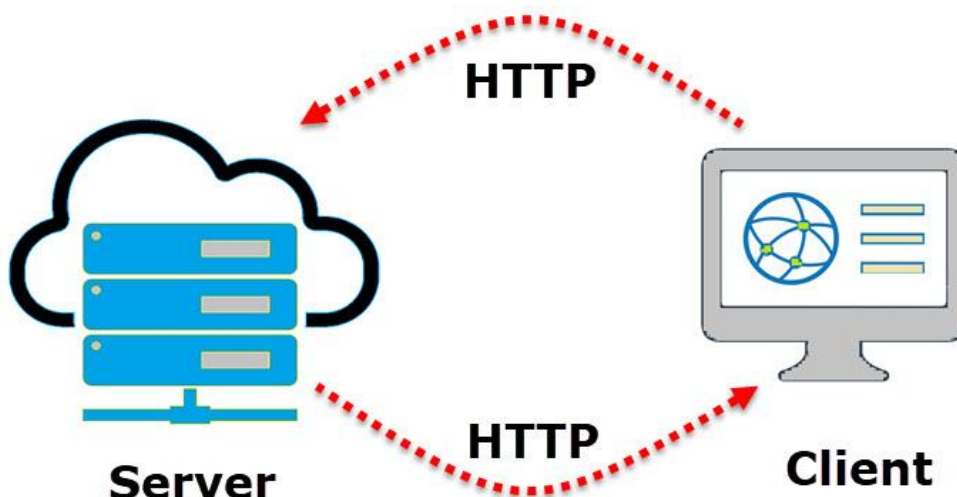


Outline

What is a Web Server?

A Web Server is combination of Hardware and Software which is responsible for maintaining, fetching and serving web pages to Web Clients. The information in the web pages can be Text in the form of HTML Documents, Images, Video, Applications etc.

Web Browsers in your laptops and mobile phones are web clients. If you observed the terms Web Server and Web Client then yes, this type of communication is called as Client - Server Model.

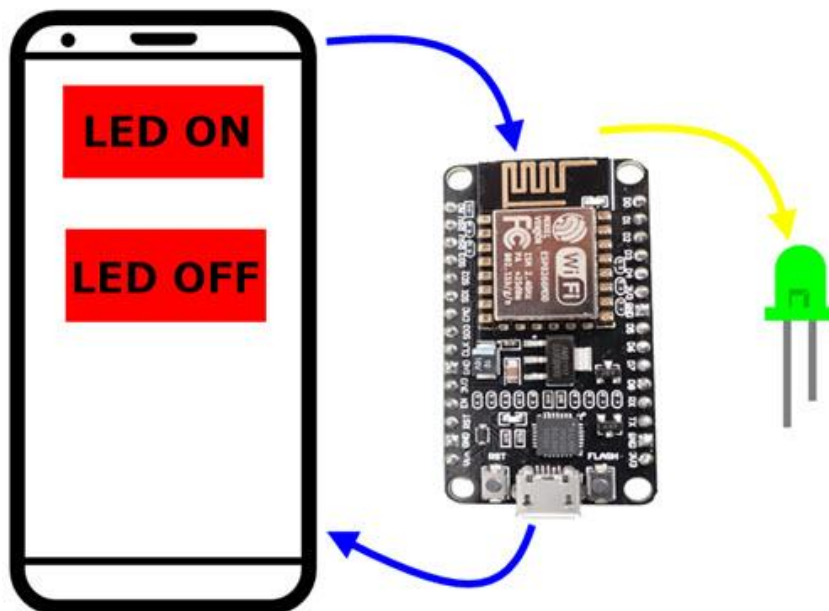


The communication between Client and Server is implemented using a special protocol called HTTP or Hyper Text Transfer Protocol. In this type of communication, the Web Client makes a request for information from the server using HTTP. The Web Server, which is always waiting (listening) for a request, responds to the client's request with appropriate web page.

If the requested page is not found, then the server responds with HTTP 404 Error.

Requirements of ESP8266 Web Server

With the brief introduction of Web Servers in general, we will now understand what are the requirement of a standalone ESP8266 Web Server. An ESP8266 Web Server must contain a web page in the form of HTML Text.



When a client, like a web browser in a mobile phone, sends a request for that web page over HTTP, the web server in ESP8266 must respond with the web page. Additionally, when the client performs any operations, like clicking on a button, the server should respond with appropriate actions (like turning ON / OFF an LED).

Wi-Fi Modes of Operation of ESP8266

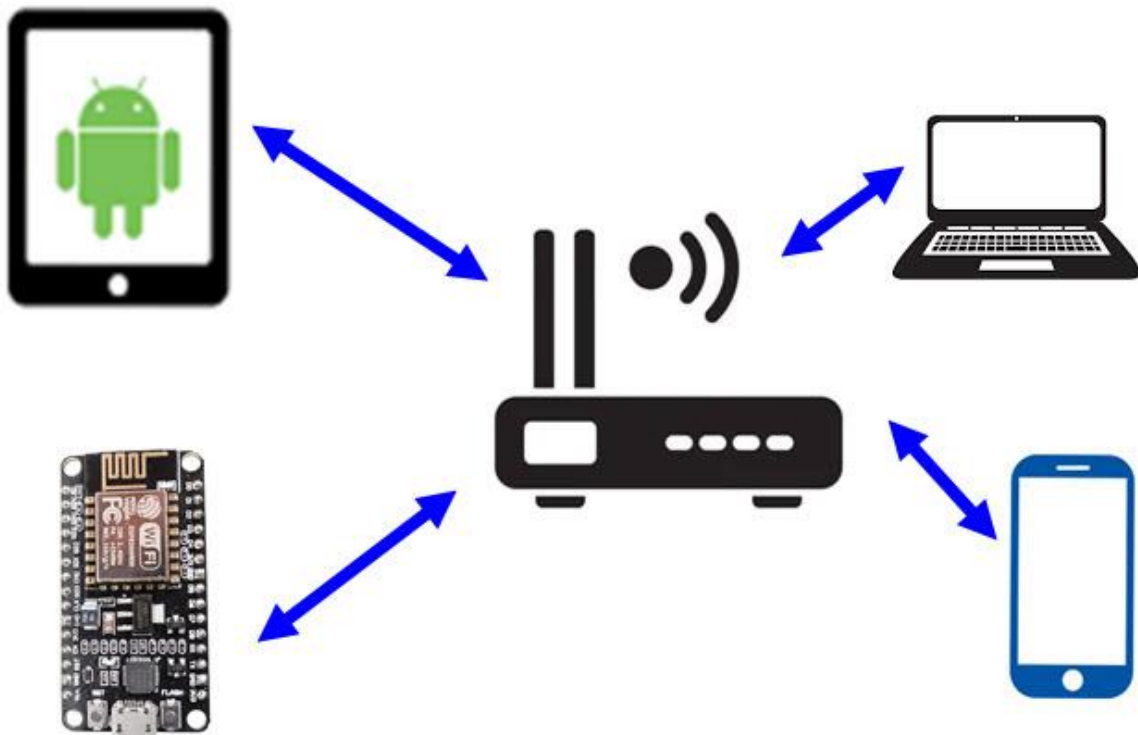
Before proceeding with creating a Web Server for ESP8266, we will take a look at the different operating modes of Wi-Fi in ESP8266. If you remember the How to Connect ESP8266 to WiFi tutorial, I already discussed about these modes. But for this web server tutorial, we will revise them once again.

Basically, the ESP8266 Wi-Fi Module operates in three WiFi operating modes. They are:

- **Station Mode (STA)**
- **Soft Access Point Mode (AP)**
- **Station + Soft AP Mode**

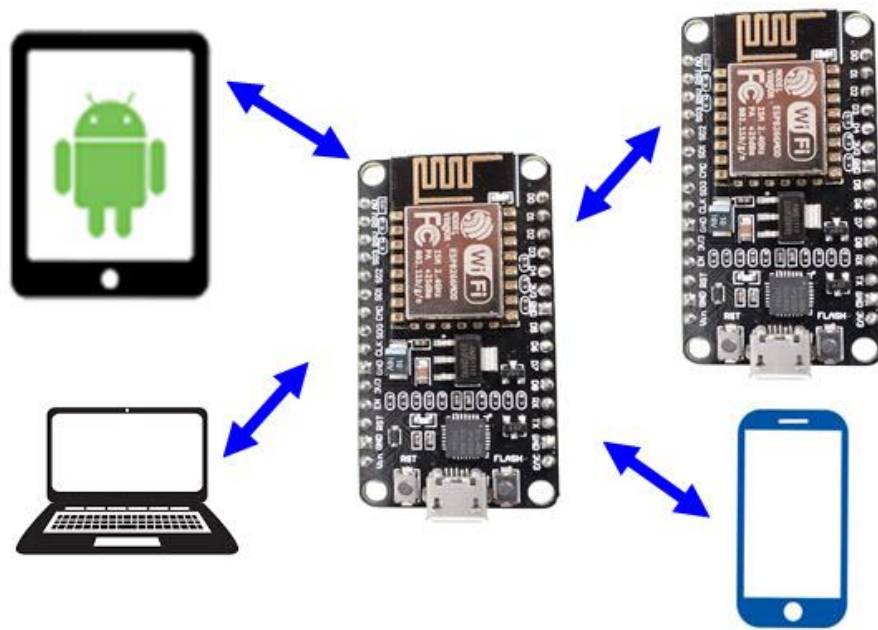
In station mode, the ESP8266 Module connects to an existing WiFi Network, which is setup by a Wireless Router, just like our Mobile Phones and Laptops.

The ESP8266 Wi-Fi Module connects to a Wi-Fi Network of Router using the router's SSID and Password and the router assigns the local IP Address for ESP8266.



Coming to Access Point Mode, the ESP8266 Module creates its own WiFi Network like a Wireless Router, so that other stations like Mobile Phones, Laptops and even other ESP8266 modules (in STA Mode) can connect to that network.

Since ESP8266 doesn't have a Wired Ethernet connectivity to internet, this AP Mode is called as Soft AP Mode. While configuring ESP8266 in AP Mode, you have to set the SSID and Password for the network, so that other devices can connect to that network using those credentials.



Station + Soft AP is a combination of Station Mode and Soft AP Mode. In this, the ESP8266 acts as both the Station as well as an Access Point.

Which Mode to use for Creating Web Server?

You can configure ESP8266 Wi-Fi Module either in Station Mode or in Access point Mode to create a web server. The difference is that in station mode, all the devices (Mobiles, laptops, ESP8266, etc.) are connected to Wireless Router's WiFi Network and IP Address to all the devices (including the Web Server of ESP8266) is assigned by the router.

Using this IP Address, clients can access the Web Page. Additionally, the clients do not lose internet connectivity from the Router.

But if we create Web Server for ESP8266 in AP Mode, then clients must connect to the network provided by ESP8266 using its own SSID and Password in order to access the Web Pages. Since it is a soft AP Mode, clients do not have internet connectivity.

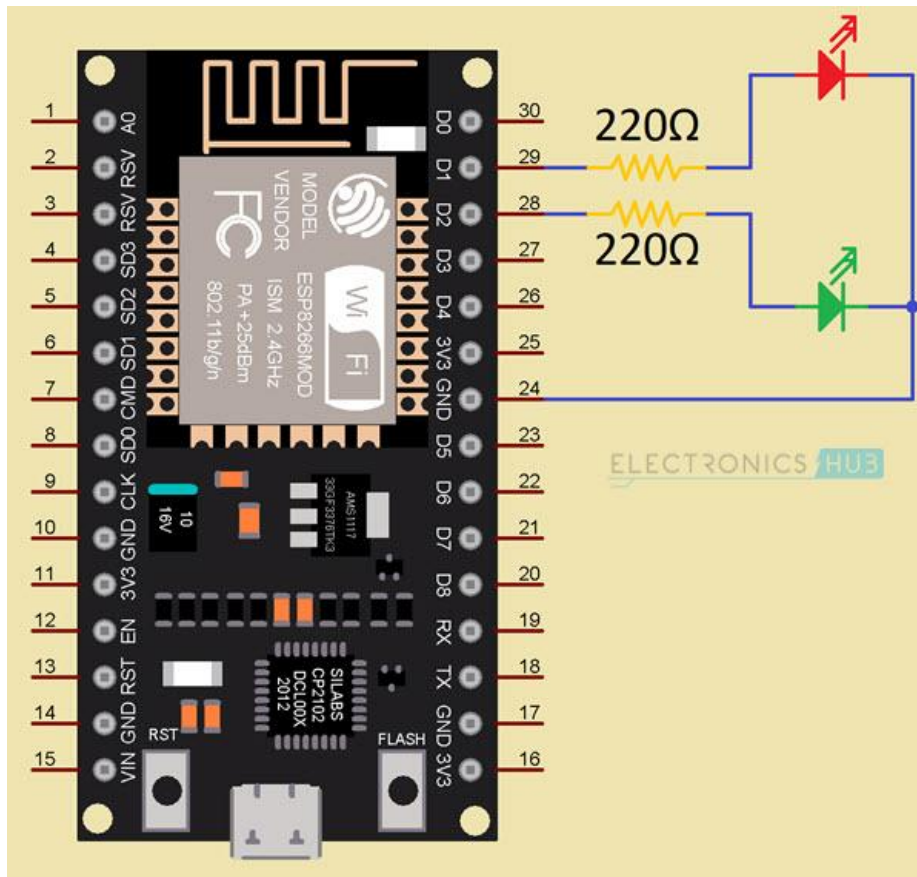
Creating ESP8266 Web Server either in Station Mode or in Soft AP Mode is very similar except the configuration part of the ESP8266.

In this tutorial, I will show you how to create a Web Server on ESP8266 configured in Station Mode (STA).

NodeMCU ESP8266 Web Server

Apart from creating the web server on ESP8266 and accessing it on clients, we will also see how Server responds to different requests for clients by controlling two LEDs connected to GPIO Pins of ESP8266 NodeMCU Board.

To demonstrate this, I connected two 5mm LEDs to GPIO4 and GPIO5 of ESP8266 through respective current limiting resistors (220Ω). GPIO4 is labelled D2 and GPIO5 is labelled D1 on NodeMCU.



Code

Coming to the important and interesting stuff, the actual code for Web Server on ESP8266. It is just an HTML Code with some text, couple of buttons and some stylization.

The following block shows the complete code for ESP8266 Web Server. I will explain the code in the next section.

```
#include <ESP8266WiFi.h>
#define gpio4LEDPin 4 /* One LED connected to GPIO4 - D2 */
#define gpio5LEDPin 5 /* One LED connected to GPIO5 - D1 */
const char* ssid = "ESP8266-WiFi"; /* Add your router's SSID */
const char* password = "12345678"; /*Add the password */
```

```

int gpio4Value;
int gpio5Value;
WiFiServer espServer(80); /* Instance of WiFiServer with port number 80 */
/* 80 is the Port Number for HTTP Web Server */
void setup()
{
  Serial.begin(115200); /* Begin Serial Communication with 115200 Baud Rate */
  /* Configure GPIO4 and GPIO5 Pins as OUTPUTs */
  pinMode(gpio4LEDPin, OUTPUT);
  pinMode(gpio5LEDPin, OUTPUT);
  /* Set the initial values of GPIO4 and GPIO5 as LOW*/
  /* Both the LEDs are initially OFF */
  digitalWrite(gpio4LEDPin, LOW);
  digitalWrite(gpio5LEDPin, LOW);

  Serial.print("\n");
  Serial.print("Connecting to: ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA); /* Configure ESP8266 in STA Mode */
  WiFi.begin(ssid, password); /* Connect to Wi-Fi based on above SSID and
Password */
  while(WiFi.status() != WL_CONNECTED)
  {
    Serial.print("*");
    delay(500);
  }
  Serial.print("\n");
  Serial.print("Connected to Wi-Fi: ");
  Serial.println(WiFi.SSID());
  delay(100);
  /* The next four lines of Code are used for assigning Static IP to ESP8266 */
  /* Do this only if you know what you are doing */
  /* You have to check for free IP Addresses from your Router and */
  /* assign it to ESP8266 */
  /* If you are comfortable with this step, please un-comment the next four lines */
  /* if not, leave it as it is and proceed */
  //IPAddress ip(192,168,1,6);
  //IPAddress gateway(192,168,1,1);
  //IPAddress subnet(255,255,255,0);
  //WiFi.config(ip, gateway, subnet);

```

```

//delay(2000);
Serial.print("\n");
Serial.println("Starting ESP8266 Web Server...");
espServer.begin(); /* Start the HTTP web Server */
Serial.println("ESP8266 Web Server Started");
Serial.print("\n");
Serial.print("The URL of ESP8266 Web Server is: ");
Serial.print("http://");
Serial.println(WiFi.localIP());
Serial.print("\n");
Serial.println("Use the above URL in your Browser to access ESP8266 Web
Server\n");
}
void loop()
{
  WiFiClient client = espServer.available(); /* Check if a client is available */
  if(!client)
  {
    return;
  }
  Serial.println("New Client!!!");
  String request = client.readStringUntil('\r'); /* Read the first line of the request
from client */
  Serial.println(request); /* Print the request on the Serial monitor */
  /* The request is in the form of HTTP GET Method */
  client.flush();
  /* Extract the URL of the request */
  /* We have four URLs. If IP Address is 192.168.1.6 (for example),
  * then URLs are:
  * 192.168.1.6/GPIO4ON and its request is GET /GPIO4ON HTTP/1.1
  * 192.168.1.6/GPIO4OFF and its request is GET /GPIO4OFF HTTP/1.1
  * 192.168.1.6/GPIO5ON and its request is GET /GPIO5ON HTTP/1.1
  * 192.168.1.6/GPIO4OFF and its request is GET /GPIO5OFF HTTP/1.1
  */
  /* Based on the URL from the request, turn the LEDs ON or OFF */
  if (request.indexOf("/GPIO4ON") != -1)
  {
    Serial.println("GPIO4 LED is ON");
    digitalWrite(gpio4LEDPin, HIGH);
    gpio4Value = HIGH;
  }
}

```

```

}
if (request.indexOf("/GPIO4OFF") != -1)
{
    Serial.println("GPIO4 LED is OFF");
    digitalWrite(gpio4LEDPin, LOW);
    gpio4Value = LOW;
}
if (request.indexOf("/GPIO5ON") != -1)
{
    Serial.println("GPIO5 LED is ON");
    digitalWrite(gpio5LEDPin, HIGH);
    gpio5Value = HIGH;
}
if (request.indexOf("/GPIO5OFF") != -1)
{
    Serial.println("GPIO5 LED is OFF");
    digitalWrite(gpio5LEDPin, LOW);
    gpio5Value = LOW;
}
/* HTTP Response in the form of HTML Web Page */
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(); // IMPORTANT
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<head>");
client.println("<meta    name=\"viewport\"    content=\"width=device-width,
initial-scale=1\">");
client.println("<link rel=\"icon\" href=\"data:;\">");
/* CSS Styling for Buttons and Web Page */
client.println("<style>");
client.println("html { font-family: Courier New; display: inline-block; margin:
0px auto; text-align: center;});
client.println(".button {border: none; color: white; padding: 10px 20px; text-
align: center;});
client.println("text-decoration: none; font-size: 25px; margin: 2px; cursor:
pointer;});
client.println(".button1 {background-color: #13B3F0;});
client.println(".button2 {background-color: #3342FF;});
client.println("</style>");

```

```

client.println("</head>");
/* The main body of the Web Page */
client.println("<body>");
client.println("<h2>ESP8266 Web Server</h2>");
if(gpio4Value == LOW)
{
    client.println("<p>GPIO4 LED Status: OFF</p>");
    client.print("<p><a      href=\" /GPIO4ON\"><button      class=\"button
button1\">Click to turn ON</button></a></p>");
}
else
{
    client.println("<p>GPIO4 LED Status: ON</p>");
    client.print("<p><a      href=\" /GPIO4OFF\"><button      class=\"button
button2\">Click to turn OFF</button></a></p>");
}

if(gpio5Value == LOW)
{
    client.println("<p>GPIO5 LED Status: OFF</p>");
    client.print("<p><a      href=\" /GPIO5ON\"><button      class=\"button
button1\">Click to turn ON</button></a></p>");
}
else
{
    client.println("<p>GPIO5 LED Status: ON</p>");
    client.print("<p><a      href=\" /GPIO5OFF\"><button      class=\"button
button2\">Click to turn OFF</button></a></p>");
}

client.println("</body>");
client.println("</html>");
client.print("\n");

delay(1);
/* Close the connection */
client.stop();
Serial.println("Client disconnected");
Serial.print("\n");
}

```

Modify and Upload Code

In lines 6 and 7 of above code, you have to make the modifications as per your Wi-Fi Network Settings. These are the SSID and Password of the Wi-Fi Network.

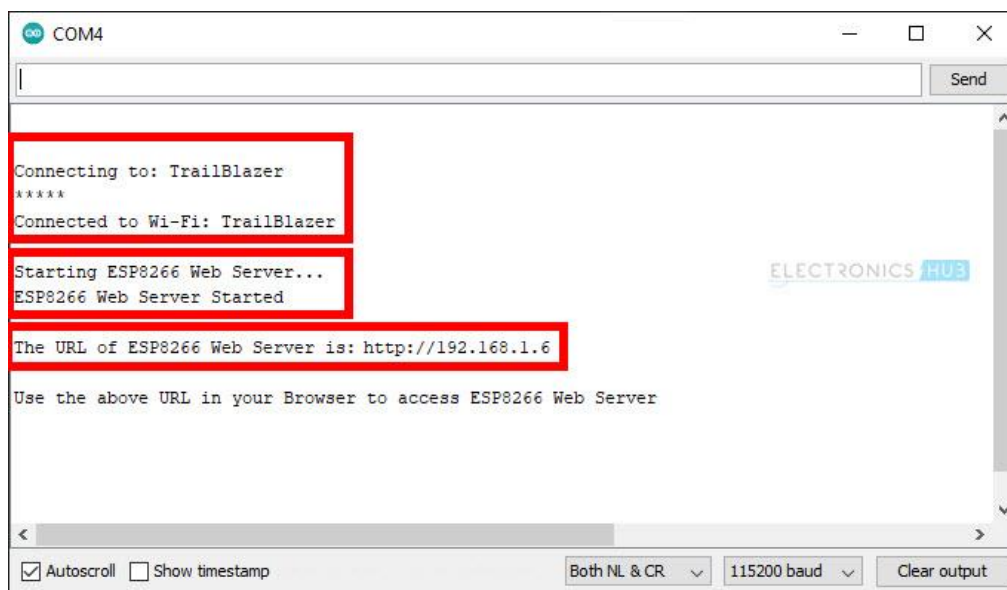
```
const char* ssid = "ESP8266-WiFi"; /* Add your router's SSID */
```

```
const char* password = "12345678"; /*Add the password */
```

After making necessary modifications, make the necessary connections as per the circuit diagram, connect the NodeMCU to the computer, select the right COM Port and upload the code.

If you are new to ESP8266 and NodeMCU, then the Getting Started with NodeMCU tutorial will help you in configuring Arduino IDE.

Open the Serial Monitor and ESP8266 NodeMCU will print some important information like the progress of Wi-Fi Connection, IP Address and URL of Web Server (which is essentially the IP Address of the ESP8266).

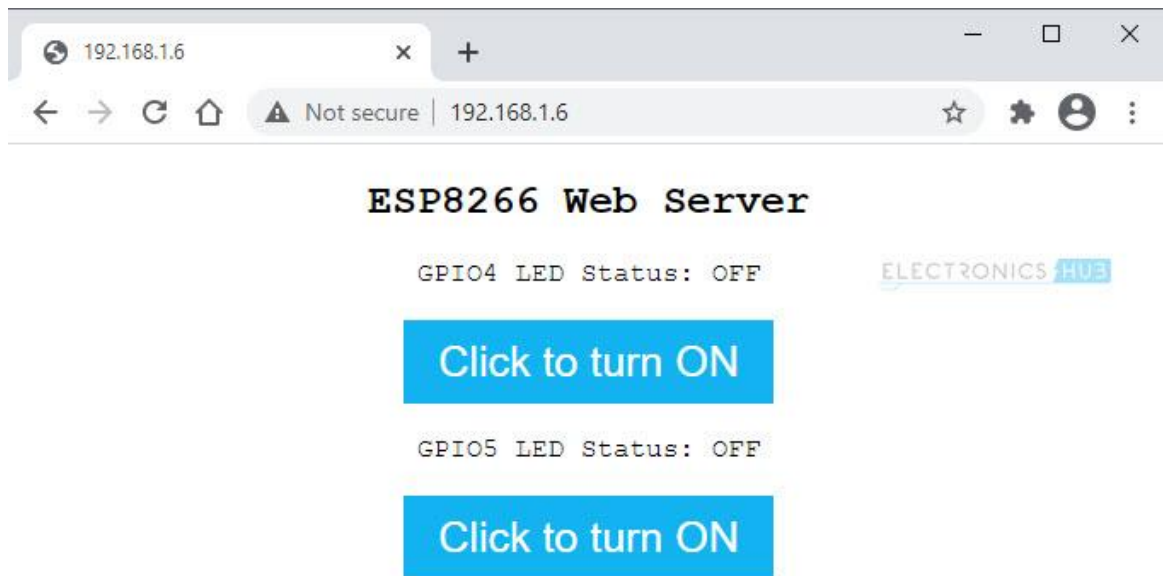


So, in my case the IP Address of ESP8266 is 192.168.1.6.

Accessing the ESP8266 Web Server from Clients

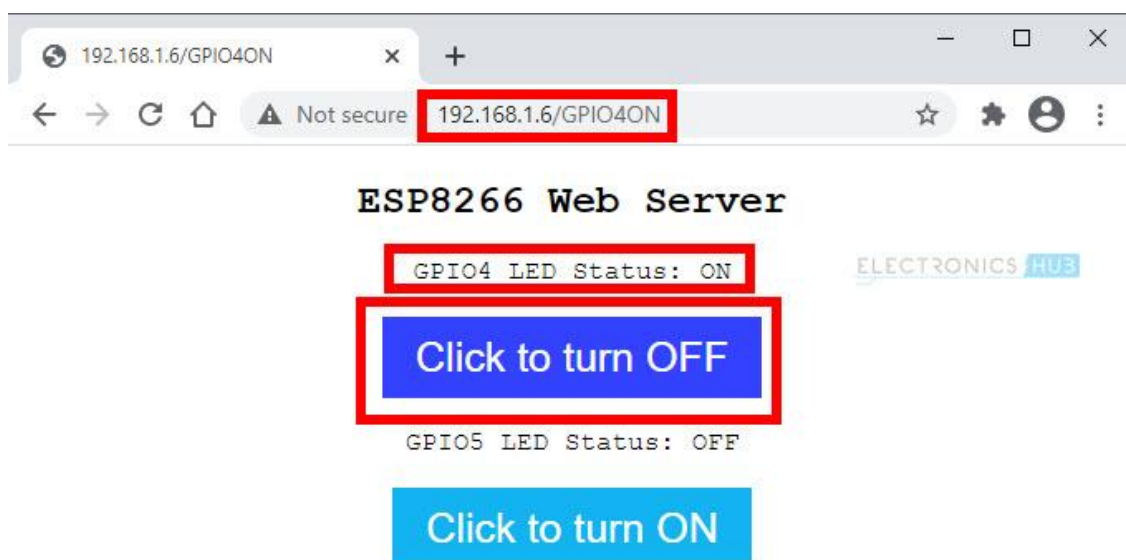
Open a Web Browser either in a laptop or mobile phone and type the IP Address. This is the moment of truth. If everything goes well, then you should be able to see a simple web page hosted by the ESP8266 Web Server.

The following is a screenshot of Chrome Web Browser on a laptop accessing the Web Server of ESP8266.



As you can see from the image, the web page displays a main header text, followed by status of the LED connected to GPIO4. This is followed by a Button, which can be used to turn ON or OFF the LED. The same stuff again for GPIO5 (status followed by Button).

Now, if I click on the first button, the LED connected to GPIO4 will turn ON, the status is updated in the web page, the text and color of the Button is also changed.



If you take a look at the Serial Monitor, every time a client tries to connect (or send a request), some key information is printed on the serial monitor. I will explain about this information (this is actually a part of request from client) in the next section.



Next, I tried the same thing on a Mobile Phone. It works perfectly.

NOTE: All the clients i.e., mobiles, laptops, etc., must be connected to the same network as the ESP8266 Module.

How ESP8266 NodeMCU Web Server Works?

Let us now try to understand How ESP8266 Web Server works by analyzing the code. I will explain all the important parts of the code and leave the simple stuff (like turning ON LED) for your exploration.

Initial Setup

First, you need to include only one header file related to the ESP8266WiFi Library.

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

Next, assign the GPIO pins to connect two LEDs. I used GPIO4 and GPIO5.

```
#define gpio4LEDPin 4
```

```
#define gpio5LEDPin 5
```

As I mentioned earlier, add the SSID and Password of your Wi-Fi Network here.

```
const char* ssid = "ENTER_YOUR_SSID";
```

```
const char* password = "ENTER_YOUR_PASSWORD";
```

Since we want to create an HTTP Server, we have to setup the Web Server with port number 80 (this is the default port for HTTP Servers).

```
WiFiServer espServer(80);
```

Next, in the setup() function, begin serial communication with baud rate of 115200 and configure the GPIO pins as OUTPUT. Also initialize the GPIO pins to LOW.

```
Serial.begin(115200);  
  
pinMode(gpio4LEDPin, OUTPUT);  
  
pinMode(gpio5LEDPin, OUTPUT);  
  
digitalWrite(gpio4LEDPin, LOW);  
  
digitalWrite(gpio5LEDPin, LOW);
```

The next few lines of code are used to begin the Wi-Fi Connection.

```
WiFi.mode(WIFI_STA);  
  
WiFi.begin(ssid, password);  
  
while(WiFi.status() != WL_CONNECTED)  
{  
  
Serial.print("*");  
  
delay(500);  
  
}
```

After successful connection, start the Web Server and print the IP Address of ESP8266. This IP address also acts as URL for the Web Server, which must be entered in the client's web browser.

```
Serial.println("Starting ESP8266 Web Server...");  
  
espServer.begin();  
  
Serial.println("ESP8266 Web Server Started");  
  
Serial.print("\n");  
  
Serial.print("The URL of ESP8266 Web Server is: ");  
  
Serial.print("https://");  
  
Serial.println(WiFi.localIP());
```

This completes the initial setup of the Web Server.

Waiting for Client and Responding

Next, in the loop() function, the Server checks if any client wants its service. If there is no client, check again. If there is a request from a client then proceed with response with request.

```
WiFiClient client = espServer.available();
```

```
if(!client)
```

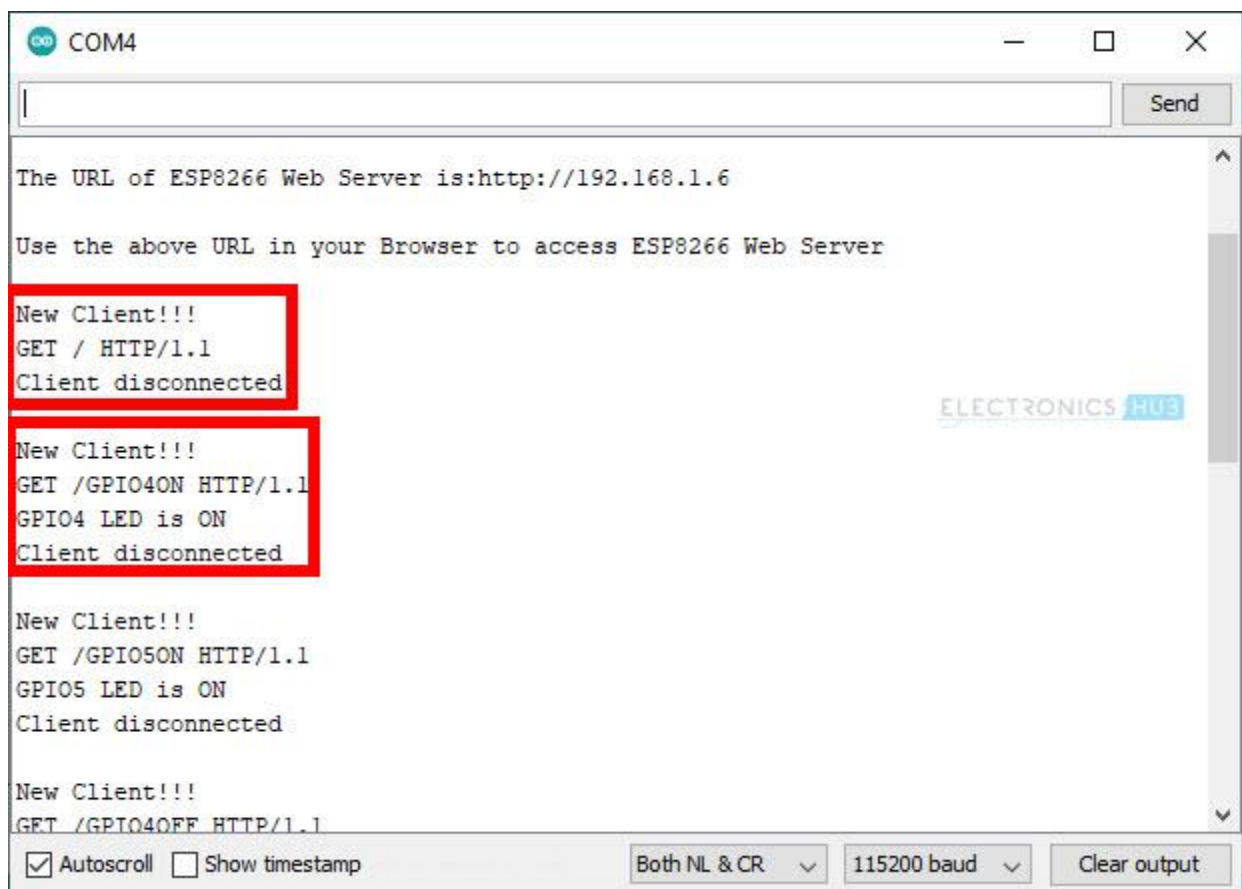
```
{
```

```
return;
```

```
}
```

When a client enters the IP Address of ESP8266 in its browser, the Server responds back with a simple web page. This is the first request – response, so there will not be any GPIO action. The request from the client is in the form of HTTP GET Method.

You can see in the following image the serial monitor output of the first request from the server i.e., the client enters the IP Address in its browser. The request was “GET / HTTP/1.1”. The “1.1” at the end is the version of HTTP.



If the client is able to open the web page successfully, then the web server is working properly and the HTTP Communication between client and server is successful.

Now, depending on the status of the LED, the buttons perform two actions. If the LED is OFF and we click on the button, the client sends a request to turn ON the LED and vice-versa.

To differentiate between the two requests, we used two URLs for the same button click that gets sent to the server based on the status of the LED. So, for both the LEDs, we have a total of four URLs. They are:

- **/GPIO4ON**
- **/GPIO4OFF**
- **/GPIO5ON**
- **/GPIO5OFF**

Using these URLs, the client sends the request and the format of the request will be in the following forms:

- **GET /GPIO4ON HTTP/1.1**
- **GET /GPIO4OFF HTTP/1.1**
- **GET /GPIO5ON HTTP/1.1**
- **GET /GPIO5OFF HTTP/1.1**

The next piece of code resolves this request from the client and performs necessary action i.e., making GPIO4 or GPIO5 LOW or HIGH.

```
if (request.indexOf("/GPIO4ON") != -1)
```

```
{
```

```
Serial.println("GPIO4 LED is ON");
```

```
digitalWrite(gpio4LEDPin, HIGH);
```

```
gpio4Value = HIGH;
```

```
}
```

```
if (request.indexOf("/GPIO4OFF") != -1)
```

```
{
```

```
Serial.println("GPIO4 LED is OFF");
```

```
digitalWrite(gpio4LEDPin, LOW);
```

```

gpio4Value = LOW;

}

if (request.indexOf("/GPIO5ON") != -1)
{
    Serial.println("GPIO5 LED is ON");
    digitalWrite(gpio5LEDPin, HIGH);
    gpio5Value = HIGH;
}

if (request.indexOf("/GPIO5OFF") != -1)
{
    Serial.println("GPIO5 LED is OFF");
    digitalWrite(gpio5LEDPin, LOW);
    gpio5Value = LOW;
}

```

Modifying the Code for ESP8266 AP Mode Web Server

If you want to create a web server in which the ESP8266 is acting as an Access Point (Soft AP), then you can still use the previous code but with slight modifications.

The first important thing to modify is the SSID and Password. In STA Mode (station mode), we enter the SSID and Password of the Router. But in AP mode, we have to create a Wi-Fi Network using ESP8266 with its own SSID and Password so that clients can connect to this network.

For example, you can put the SSID as “ESP8266-WIFI” and password as “12345678”. This SSID will be visible to clients and in order to connect to this network, use the above password.

```

const char* ssid = "ESP8266-WIFI ";
const char* password = "12345678";

```

Next is the IP address, gateway and subnet. This information is necessary for ESP8266 in AP Mode in order to create a Wi-Fi Network.

```
IPAddress ip(192,168,1,1);
```

```
IPAddress gateway(192,168,1,1);
```

```
IPAddress subnet(255,255,255,0);
```

Now, we can initialize the ESP8266 in AP Mode and also configure its IP address using the following two lines.

```
WiFi.softAP(ssid, password);
```

```
WiFi.softAPConfig(ip, gateway, subnet);
```

Difference between IoT and M2M

- 1. Internet of Things:** IOT is known as the Internet of Things where things are said to be the communicating devices that can interact with each other using a communication media. Usually every day some new devices use various sensors and actuators for sending and receiving data over the internet.
- 2. Machine to Machine:** This is commonly known as Machine to machine communication. It is a concept where two or more than two machines communicate with each other without human interaction and internet. M2M communications offer several applications such as security, tracking and tracing, manufacturing and facility management.

Difference between IoT and M2M :

Basis of	IoT	M2M
Abbreviation	Internet of Things	Machine to Machine
Intelligence	Devices have objects that are responsible for decision making	Some degree of intelligence is observed in this.
Connection type used	The connection is via Network and using various communication types.	The connection is a point to point
Communication protocol used	Internet protocols are used such as HTTP, FTP, and Telnet.	Traditional protocols and communication technology techniques are used
Data Sharing	Data is shared between other applications that are used to improve the end-user experience.	Data is shared with only the communicating parties.
Internet	Internet connection is required for communication	Devices are not dependent on the Internet.
Type of	It supports cloud	It supports point-to-point

Basis of	IoT	M2M
Communication	communication	communication.
Computer System	Involves the usage of both Hardware and Software.	Mostly hardware-based technology
Scope	A large number of devices yet scope is large.	Limited Scope for devices.
Business Type used	Business 2 Business(B2B) and Business 2 Consumer(B2C)	Business 2 Business (B2B)
Open API support	Supports Open API integrations.	There is no support for Open APIs
Examples	Smart wearables, Big Data and Cloud, etc.	Sensors, Data and Information, etc.

IoT Communication Protocols—IoT Data Protocols

These data communication protocols are those that work in the low levels of the Open Systems Interconnection (OSI) model, without the need for an Internet connection.

IoT Data Communication Protocols

Some of the different IoT data protocols, namely:

- Message Queue Telemetry Transport (MQTT)
- HyperText Transfer Protocol (HTTP)
- Constrained Application Protocol (CoAP)
- Data Distribution Service (DDS)
- WebSocket

Message Queue Telemetry Transport (MQTT)

Designed to be lightweight, so it can work in very low bandwidth networks, MQTT allows communication between nodes in both reliable and unreliable networks. MQTT follows a publish/subscribe architecture, meaning that there are nodes (brokers) that make the information available,

while others (clients) can read the available information after subscribing by accessing the corresponding URL.

A use case of MQTT is in a smart factory where there is temperature sensors installed along with the production plant. The installed sensors will connect to the MQTT broker and will publish the data within sensor topics, as follows:

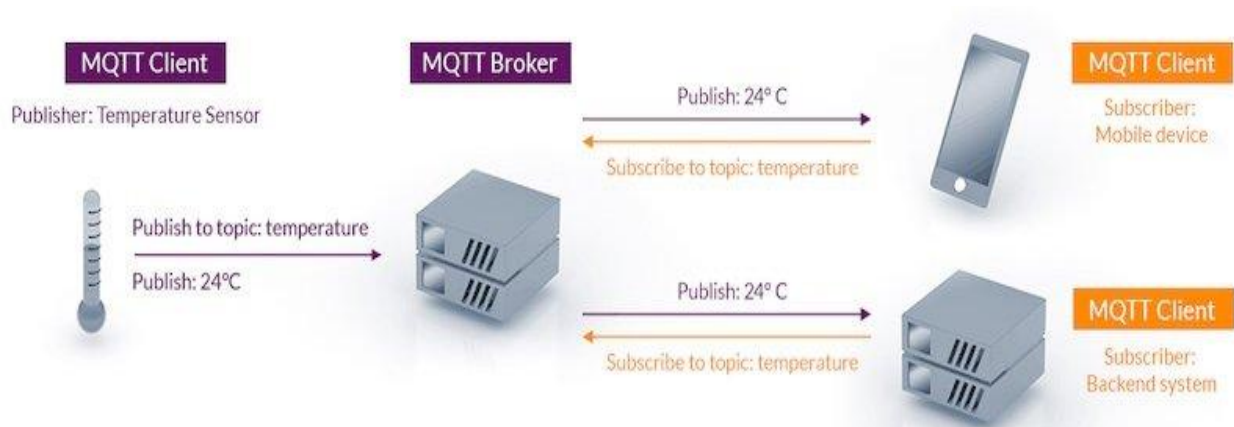


Figure 1. MQTT's publish/subscribe architecture.

In addition, MQTT defines three levels of quality of service, depending upon the reliability, from lowest to highest:

- Level 0: there is no guarantee of the message delivery.
- Level 1: the delivery is guaranteed, but it is possible to receive duplicate messages.
- Level 2: the delivery is guaranteed and there will be no duplicates.

HyperText Transfer Protocol (HTTP)

This protocol has been the origin of data communication for the World Wide Web (WWW), so it is being used in the IoT world. However, it is not optimized for it because of the following:

- The HTTP is made for two systems communicating to each other at a time, not more, so it is time and energy-consuming to connect several sensors to get information.
- The HTTP is unidirectional, made for one system (client) to be sending one message to another one (server). This makes it quite hard to escalate an IoT solution.

- Power consumption: HTTP relies on Transmission Control Protocol (TCP), which requires a lot of computing resources, so it is not suitable for battery-powered applications.

Constrained Application Protocol (CoAP)

CoAP is a web transfer protocol to be used with limited networks with low bandwidth and low availability. It follows client/server architecture and is built similarly to HTTP, supporting the REST model: servers make resources available with an URL, and clients can make requests of types GET, POST, PUT and DELETE.

Data Distribution Service (DDS)

Similar to MQTT, DDS follows publish-subscribe methodology, with the main difference being that there are no brokers. It means that all publishers (i.e., temperature sensors) and subscribers (i.e., mobile phones) are all connected to the same network. This network is known as Global Data Space (GDS) and it interconnects each node with all the other ones to avoid bottlenecks. An example of the DDS GDS can be seen in Figure 2.

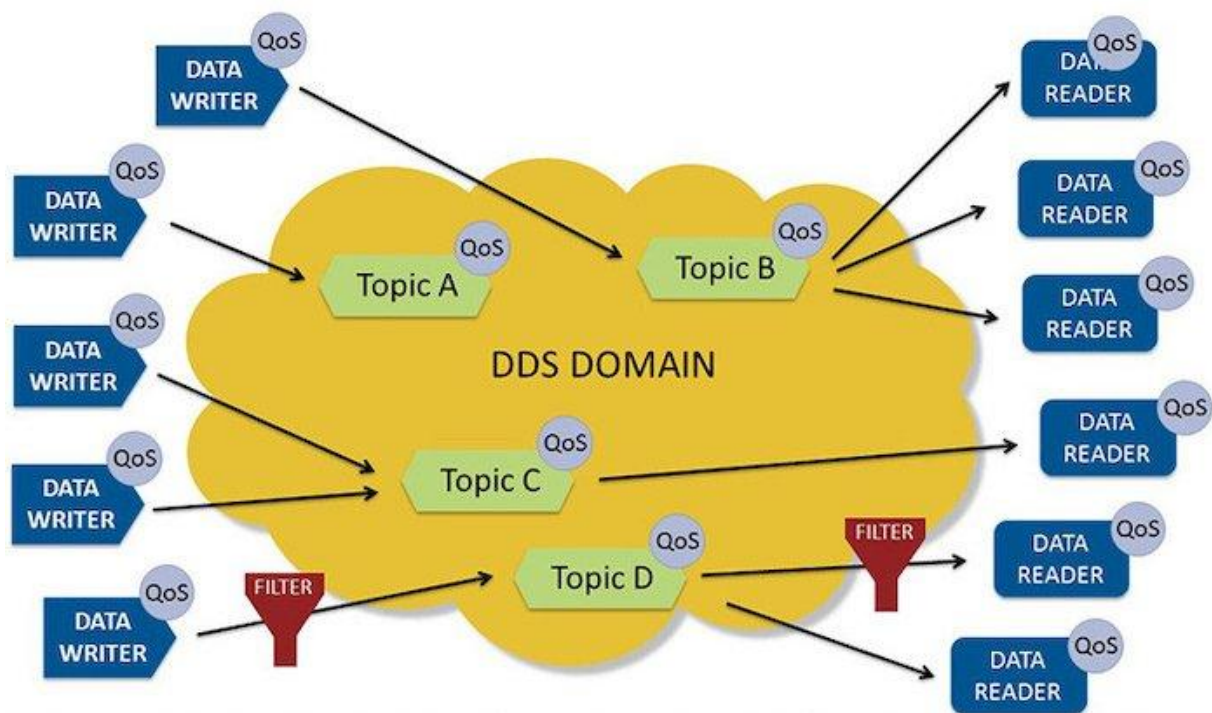


Figure 2. A DDS Global Data Space.

Furthermore, any node can leave or join the network, since they are dynamically discovered.

WebSocket

Linked to the HTTP protocol, the WebSocket technology establishes a TCP connection between a browser and a server, and then both of them exchange information until the connection is closed. Figure 3 shows a high-level comparison between HTTP and WebSocket.

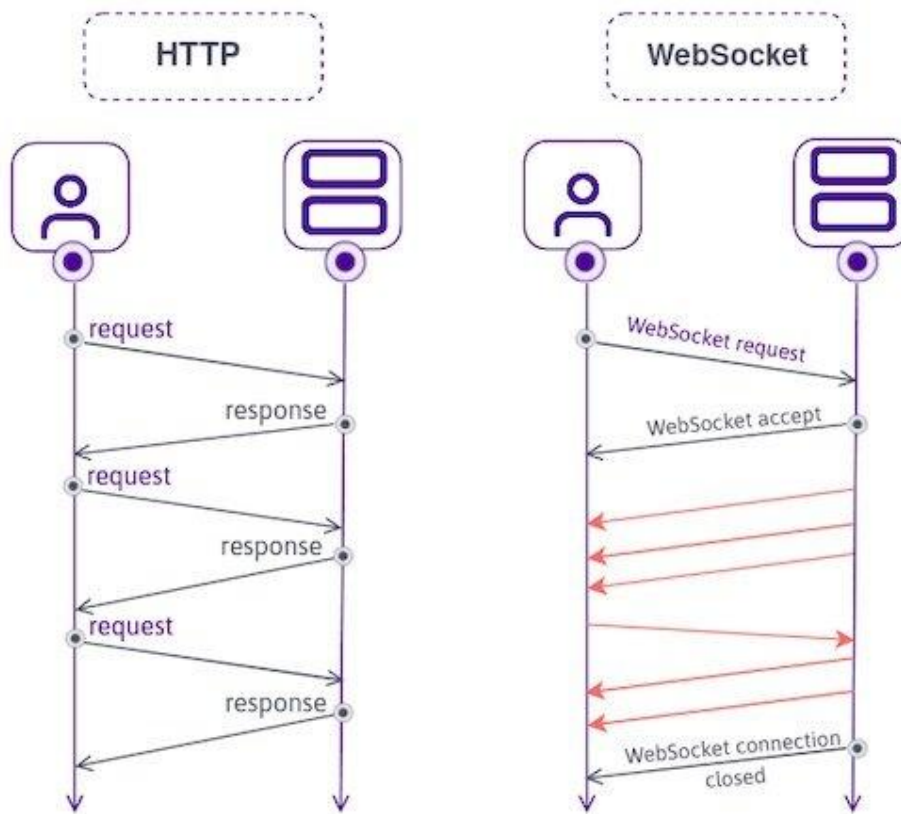
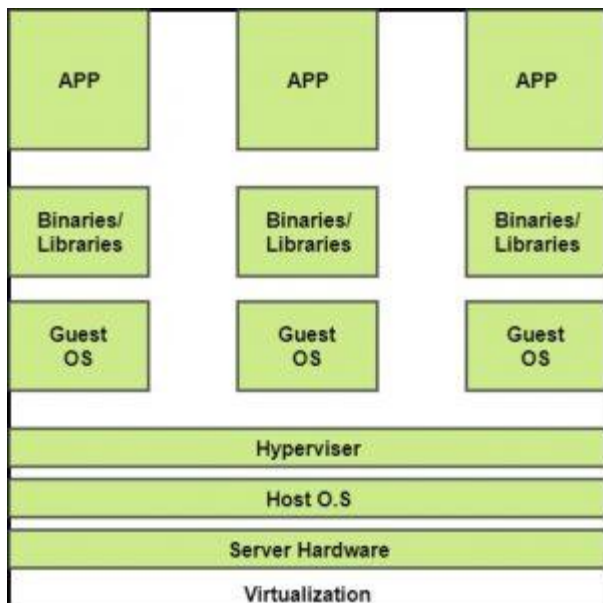


Figure 3. Comparison between HTTP and WebSocket.

Although this protocol can be seen as an improvement of the HTTP connection, the WebSocket is still very overloaded and heavy for IoT applications.

Virtualization in Cloud Computing and Types

Virtualization is a technique of how to separate a service from the underlying physical delivery of that service. It is the process of creating a virtual version of something like computer hardware. It was initially developed during the mainframe era. It involves using specialized software to create a virtual or software-created version of a computing resource rather than the actual version of the same resource.



With the help of Virtualization, multiple operating systems and applications can run on same machine and its same hardware at the same time, increasing the utilization and flexibility of hardware.

In other words, one of the main cost effective, hardware reducing, and energy saving techniques used by cloud providers is virtualization. Virtualization allows sharing a single physical instance of a resource or an application among multiple customers and organizations at one time.

BENEFITS OF VIRTUALIZATION:

1. More flexible and efficient allocation of resources.
2. Enhance development productivity.
3. It lowers the cost of IT infrastructure.
4. Remote access and rapid scalability.
5. High availability and disaster recovery.
6. Pay per use of the IT infrastructure on demand.
7. Enables running multiple operating systems.

Types of Virtualization:

1. Application Virtualization.
2. Network Virtualization.
3. Desktop Virtualization.
4. Storage Virtualization.
5. Server Virtualization.
6. Data virtualization.

- 1. Application Virtualization:** Application virtualization helps a user to have remote access of an application from a server. The server stores all personal information and other characteristics of the application but can still run on a local workstation through the internet. Example of this would be a user who needs to run two different versions of the same software.
- 2. Network Virtualization:** The ability to run multiple virtual networks with each has a separate control and data plan. It co-exists together on top of one physical network. It can be managed by individual parties that potentially confidential to each other. Network virtualization provides a

facility to create and provision virtual networks—logical switches, routers, firewalls, load balancer, Virtual Private Network (VPN), and workload security within days or even in weeks.

- 3. Desktop Virtualization:** Desktop virtualization allows the users' OS to be remotely stored on a server in the data centre. It allows the user to access their desktop virtually, from any location by a different machine. Users who want specific operating systems other than Windows Server will need to have a virtual desktop. Main benefits of desktop virtualization are user mobility, portability, easy management of software installation, updates, and patches.
- 4. Storage Virtualization:** Storage virtualization is an array of servers that are managed by a virtual storage system. The servers aren't aware of exactly where their data is stored. It makes managing storage from multiple sources to be managed and utilized as a single repository. storage virtualization software maintains smooth operations, consistent performance and a continuous suite of advanced functions despite changes, break down and differences in the underlying equipment.
- 5. Server Virtualization:** This is a kind of virtualization in which masking of server resources takes place. Here, the central-server(physical server) is divided into multiple different virtual servers by changing the identity number, processors. It causes an increase in the performance and reduces the operating cost by the deployment of main server resources into a sub-server resource. It's beneficial in virtual migration, reduce energy consumption, reduce infrastructural cost, etc.
- 6. Data virtualization:** The data is collected from various sources and managed that at a single place without knowing more about the technical information like how data is collected, stored & formatted then arranged that data logically. so that its virtual view can be accessed by its interested people and stakeholders, and users through the various cloud services remotely. Many big giant companies are providing their services like Oracle, IBM, At scale, Cdata, etc.

It can be used to performing various kind of tasks such as:

- Data-integration
- Business-integration
- Service-oriented architecture data-services
- Searching organizational data

IoT Cloud Architecture

Some incarnations of these IoT systems are Smart Homes, Connected Cars, Smart Factories, and Smart Grids that consolidate/manage office data. An IoT cloud architecture is introduced when there is a need to process

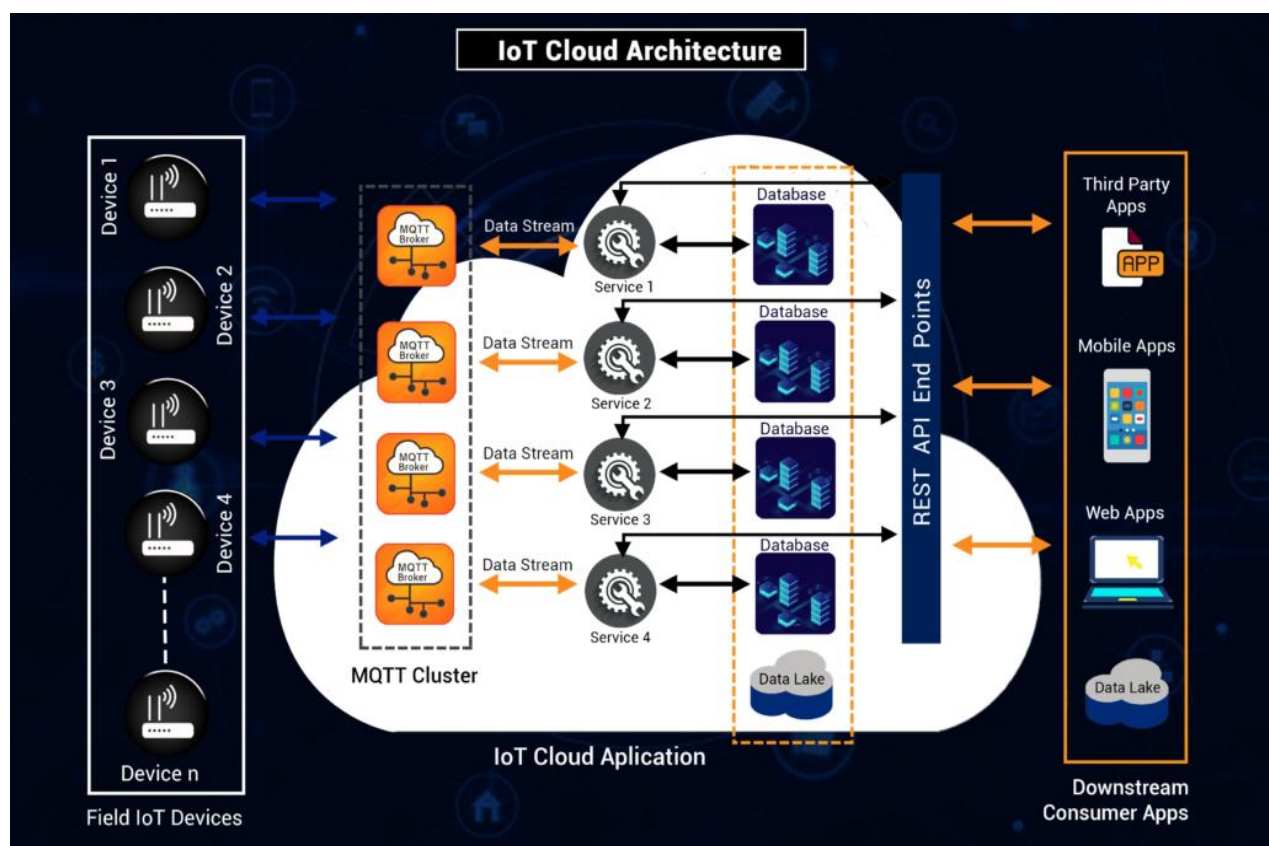
and analyse large data sets. IoT cloud platform should be robust, scalable and agile with impeccable security features.

Layers of Cloud Architecture for IoT

In an IoT cloud architecture, the data flows across many layers. Although the actual cloud platform configuration may differ between organisations, all these solutions accomplish the same basic objective.

This is allowing different devices to connect to the network, processing the data and utilizing the insights gained for automation. A significant part of the data processing takes place in the cloud databases and reporting layer.

Let us take a look at the basic components of an IoT architecture.



- 1. IoT Devices:** The device integrates with sensors or actuators and establishes a connection with the IoT Integration Middleware. In some use cases, multiple devices may be grouped together and connected to an IoT gateway infrastructure that transmits device data to the IoT Integration Middleware. The devices have drivers, i.e., software that enables access to the sensors/actuators.
- 2. IoT Integration Middleware:** The IoT Integration Middleware is an integration layer for various devices when connecting with the cloud. It

receives data from the connected devices, processes it and transmits this information to downstream applications. The processing of data may include the evaluation of condition-action rules, and deployment of commands to the device sensors based on the evaluation. An IoT gateway has to be installed in between for communication with the middleware platform.

The middleware broker also ensures that there is no loss of data, since asynchronous communication is established with the connected device.

1. **Cloud Servers:** Servers are the most important part of the IoT cloud, as these are needed for providing business services to customers. These are virtual machines linked to individual databases.
2. **Databases:** Based on the business requirements for data storage and processing, SQL and No SQL databases can be configured on the IoT cloud. SQL databases store data in the form of two-dimensional tables. The main disadvantage of this kind of database is its performance.
3. **Downstream Applications/BI Tools:** The cloud servers are connected to third-party apps, mobile/web applications or business intelligence tools through REST API endpoints.

When a large number of IoT devices and applications are connected to the IoT cloud platform, the cloud application servers will have to transfer a huge amount of data. In order to streamline this, load balancing is enforced. This ensures even distribution of workload across the backend servers and improves efficiency.

Cloud Computing Architecture

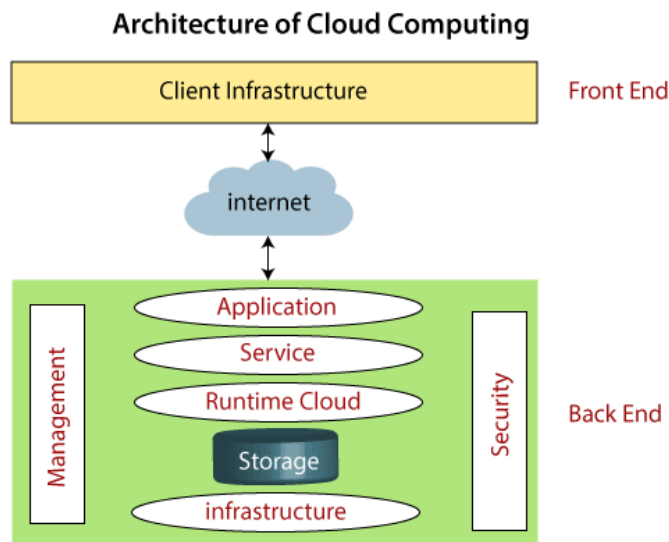
Cloud computing technology is used by both small and large organizations to **store the information** in cloud and **access** it from anywhere at anytime using the internet connection.

Cloud computing architecture is a combination of service-oriented architecture and event-driven architecture.

Cloud computing architecture is divided into the following two parts -

- Front End
- Back End

The below diagram shows the architecture of cloud computing -



Front End: The front end is used by the client. It contains client-side interfaces and applications that are required to access the cloud computing platforms. The front end includes web servers (including Chrome, Firefox, internet explorer, etc.), thin & fat clients, tablets, and mobile devices.

Back End: The back end is used by the service provider.

It manages all the resources that are required to provide cloud computing services. It includes a huge amount of data storage, security mechanism, virtual machines, deploying models, servers, traffic control mechanisms, etc.

Components of Cloud Computing Architecture

There are the following components of cloud computing architecture -

- 1. Client Infrastructure:** Client Infrastructure is a Front end component. It provides GUI (Graphical User Interface) to interact with the cloud.
- 2. Application:** The application may be any software or platform that a client wants to access.
- 3. Service:** A Cloud Services manages that which type of service you access according to the client's requirement.

Cloud computing offers the following three type of services:

i. Software as a Service (SaaS) – It is also known as cloud application services. Mostly, SaaS applications run directly through the web browser means client do not require downloading and installing these applications. Some important example of SaaS is given below –

Example: Google Apps, Salesforce Dropbox, Slack, Hubspot, Cisco WebEx.

ii. Platform as a Service (PaaS) – It is also known as cloud platform services. It is quite similar to SaaS, but the difference is that PaaS provides a platform for software creation, but using SaaS, client can access software over the internet without the need of any platform.

Example: Windows Azure, Force.com, Magento Commerce Cloud, OpenShift.

iii. Infrastructure as a Service (IaaS) – It is also known as cloud infrastructure services. It is responsible for managing applications data, middleware and runtime environments.

Example: Amazon Web Services (AWS) EC2, Google Compute Engine (GCE), Cisco Metapod.

4. Runtime Cloud: Runtime Cloud provides the execution and runtime environment to the virtual machines.

5. Storage: Storage is one of the most important components of cloud computing. It provides a huge amount of storage capacity in the cloud to store and manage data.

6. Infrastructure: It provides services on the host level, application level, and network level. Cloud infrastructure includes hardware and software components such as servers, storage, network devices, virtualization software, and other storage resources that are needed to support the cloud computing model.

7. Management: Management is used to manage components such as application, service, runtime cloud, storage, infrastructure, and other security issues in the backend and establish coordination between them.

8. Security: Security is an in-built back end component of cloud computing. It implements a security mechanism in the back end.

9. Internet: The Internet is medium through which front end and back end can interact and communicate with each other.

Role of Cloud computing in IoT

1. Enables remote computing capabilities: With a large storage capacity, IoT eliminates the dependencies on on-site infrastructure. With continued development and internet-based tech development such as the internet and devices supporting advanced cloud solutions, cloud technology has become main stream.

2. Security & Privacy: Tasks can be handled automatically with cloud tech & IoT, organizations are able to reduce security threats by a considerable amount. A cloud tech-enabled with IoT is a solution that provides preventive, detective and corrective control. With effective authentication and encryption protocols, it also provides users with strong security measures. Protocols such as biometrics in IoT products help manage as well as safeguard user identities along with data.

3. Data Integration: Current tech developments have not only integrated IoT and cloud smoothly but also provide real-time connectivity and communication. This in turn makes the extraction of real-time information about key business processes and performing on-spot data integration with 24/7 connectivity easy. Cloud-based solutions with powerful data integration capabilities are able to handle a large amount of data generated from multiple sources along with its centralized storage, processing and analysis.

4. Minimal Hardware Dependency: Presently, several IoT solutions offer plug-and-play hosting services that are enabled by integrating the cloud with the IoT. With cloud-enabled, IoT hosting providers need not rely on any kind of hardware or equipment to support the agility required by IoT devices. It has become easy for organizations to implement large scale IoT strategies seamlessly across platforms and move to omni channel communication.

5. Business Continuity: Known for their agility and reliability, cloud computing solutions are able to provide business continuity in case of any emergency, data loss or disaster. Cloud services operate via a network of data servers located in multiple geographical locations storing multiple copies of data backup. In case of any emergency, IoT based operations continue to work and data recovery becomes easy.

6. Communication Between Multiple Devices & Touchpoint:

IoT devices and services need to connect with each other and communicate to perform tasks that are enabled using cloud solutions. By supporting several robust APIs, cloud & IoT is able to interact amongst themselves and connected devices. Having a cloud supported communication helps fasten the interaction happen seamlessly.

7. Response Time & Data Processing:

Edge computing combined with IoT solutions usually shortens response time and speeds up data processing capabilities. It requires the deployment of IoT with cloud computing and edge computing solutions for maximum utilization.

Though cloud computing services can accelerate the growth of IoT, there are certain challenges in deploying these services successfully. The combination of IoT and cloud presents a few obstacles that need to be handled beforehand.

Cloud Service Models

There are the following three types of cloud service models -

1. Infrastructure as a Service (IaaS)
2. Platform as a Service (PaaS)
3. Software as a Service (SaaS)



Infrastructure as a Service (IaaS)

IaaS is also known as Hardware as a Service (HaaS). It is a computing infrastructure managed over the internet. The main advantage of using IaaS is that it helps users to avoid the cost and complexity of purchasing and managing the physical servers.

Characteristics of IaaS

There are the following characteristics of IaaS -

- Resources are available as a service
- Services are highly scalable
- Dynamic and flexible
- GUI and API-based access
- Automated administrative tasks

Example: DigitalOcean, Linode, Amazon Web Services (AWS), Microsoft Azure, Google Compute Engine (GCE), Rackspace, and Cisco Metacloud.

Platform as a Service (PaaS)

PaaS cloud computing platform is created for the programmer to develop, test, run, and manage the applications.

Characteristics of PaaS

There are the following characteristics of PaaS -

- Accessible to various users via the same development application.
- Integrates with web services and databases.
- Builds on virtualization technology, so resources can easily be scaled up or down as per the organization's need.
- Support multiple languages and frameworks.
- Provides an ability to "**Auto-scale**".

Example: AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, Magento Commerce Cloud, and OpenShift.

Software as a Service (SaaS)

SaaS is also known as "**on-demand software**". It is a software in which the applications are hosted by a cloud service provider. Users can access these applications with the help of internet connection and web browser.

Characteristics of SaaS

There are the following characteristics of SaaS -

- Managed from a central location
- Hosted on a remote server
- Accessible over the internet
- Users are not responsible for hardware and software updates. Updates are applied automatically.
- The services are purchased on the pay-as-per-use basis

Example: BigCommerce, Google Apps, Salesforce, Dropbox, ZenDesk, Cisco WebEx, ZenDesk, Slack, and GoToMeeting.

Difference between IaaS, PaaS, and SaaS

The below table shows the difference between IaaS, PaaS, and SaaS -

IaaS	Paas	SaaS
It provides a virtual data center to store information and create platforms for app development, testing, and deployment.	It provides virtual platforms and tools to create, test, and deploy apps.	It provides web software and apps to complete business tasks.
It provides access to resources such as virtual machines, virtual storage, etc.	It provides runtime environments and deployment tools for applications.	It provides software as a service to the end-users.
It is used by network architects.	It is used by developers.	It is used by end users.
IaaS provides only Infrastructure.	PaaS provides Infrastructure+Platform.	SaaS provides Infrastructure+Platform +Software.

Infrastructure as a Service | IaaS

IaaS is also known as **Hardware as a Service (HaaS)**. 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.

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 either public cloud or private cloud.

IaaS provider provides the following services -

1. **Compute:** Computing as a Service includes virtual central processing units and virtual main memory for the Vms that is provisioned to the end- users.
2. **Storage:** IaaS provider provides back-end storage for storing files.
3. **Network:** Network as a Service (NaaS) provides networking components such as routers, switches, and bridges for the Vms.
4. **Load balancers:** It provides load balancing capability at the infrastructure layer.



Advantages of IaaS cloud computing 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:** 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.

Top IaaS Providers who are providing IaaS cloud computing platform

IaaS Vendor		IaaS Solution	Details
Amazon Services	Web	Elastic, Elastic Compute (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.
Reliance Communications		Reliance Internet Data Center	RIDC supports both traditional hosting and cloud services, with data centers in Mumbai, Bangalore, Hyderabad, and Chennai.
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.

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. Client

can purchase these applications from a cloud service provider on a pay-as-per use basis and access them using the Internet connection.

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.

1. Programming languages: PaaS providers provide various programming languages for the developers to develop the applications. Programming languages provided by PaaS providers are Java, PHP, Ruby, Perl, and Go.



2. Application frameworks: PaaS providers provide application frameworks to easily understand the application development. Application frameworks provided by PaaS providers are Node.js, Drupal, Joomla, WordPress, Spring, Play, Rack, and Zend.

3. Databases: PaaS providers provide various databases such as ClearDB, PostgreSQL, MongoDB, and Redis to communicate with the applications.

4. Other tools: PaaS providers provide various other tools that are required to develop, test, and deploy the applications.

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.

Popular PaaS Providers

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.

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 - 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.

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. 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. 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.

2) Latency issue: 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 involve 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.

Popular SaaS Providers

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.

Cloud Service Provider Companies

Cloud Service providers (CSP) offers various services such as **Software as a Service, Platform as a service, Infrastructure as a service, network services, business applications, mobile applications,** and **infrastructure** in the cloud. The cloud service providers host these services in a data center and users can access these services through cloud provider companies using an Internet connection.

There are the following Cloud Service Providers Companies -

Amazon Web Services (AWS): AWS (Amazon Web Services) is a **secure cloud service platform** provided by **Amazon**. It offers various services such as database storage, computing power, content delivery, Relational Database, Simple Email, Simple Queue and other functionality to increase the organization's growth.

Features of AWS

AWS provides various powerful features for building scalable, cost-effective, enterprise applications. Some important features of AWS is given below-

- AWS is **scalable** because it has an ability to scale the computing resources up or down according to the organization's demand.
- AWS is **cost-effective** as it works on a **pay-as-you-go** pricing model.
- It provides various flexible storage options.
- It offers various **security services** such as infrastructure security, data encryption, monitoring & logging, identity & access control, penetration testing, and DDoS attacks.
- It can efficiently manage and secure Windows workloads.

2. **Microsoft Azure:** Microsoft Azure is also known as **Windows Azure**. It supports various operating systems, databases, programming languages, frameworks that allow IT professionals to easily build, deploy and manage applications through a worldwide network. It also allows users to create different groups for related utilities.

Features of Microsoft Azure

- Microsoft Azure provides **scalable, flexible, and cost-effective**
- It allows developers to quickly manage applications and websites.
- It managed each resource individually.
- Its IaaS infrastructure allows us to launch a general-purpose virtual machine in different platforms such as Windows and Linux.

- It offers a **Content Delivery System (CDS)** for delivering the Images, videos, audios, and applications.

3. **Google Cloud Platform:** Google cloud platform is a product of **Google**. It consists of a set of physical devices, such as computers, hard disk drives, and virtual machines. It also helps organizations to simplify the migration process.

Features of Google Cloud

- Google cloud includes various **big data services** such as Google BigQuery, Google CloudDataproc, Google CloudDatalab, and Google Cloud Pub/Sub.
- It provides various services related to **networking**, including Google Virtual Private Cloud (VPC), Content Delivery Network, Google Cloud Load Balancing, Google Cloud Interconnect, and Google Cloud DNS.
- It offers various **scalable** and **high-performance**
- GCP provides various **serverless services** such as Messaging, Data Warehouse, Database, Compute, Storage, Data Processing, and Machine learning (ML)
- It provides a free cloud shell environment with Boost Mode.

4. **IBM Cloud Services**

IBM Cloud is an open-source, faster, and more reliable platform. It is built with a suite of advanced data and AI tools. It offers various services such as Infrastructure as a service, Software as a service and platform as a service. Client can access its services like compute power, cloud data & Analytics, cloud use cases and storage networking using internet connection.

Feature of IBM Cloud

- IBM cloud improves operational efficiency.
- Its speed and agility improve the customer's satisfaction.
- It offers Infrastructure as a Service (IaaS), Platform as a Service (PaaS) as well as Software as a Service (SaaS)

5. **VMware Cloud:** VMware cloud is a Software-Defined Data Center (SSDC) unified platform for the Hybrid Cloud. It allows cloud providers to build agile, flexible, efficient, and robust cloud services.

Features of VMware

- VMware cloud works on the **pay-as-per-use** model and **monthly subscription**
- It provides better customer satisfaction by protecting the user's data.
- It can easily create a new VMware **Software-Defined Data Center (SDDC)** cluster on AWS cloud by utilizing a RESTful API.
- It provides flexible storage options.
- It provides a dedicated high-performance network for managing the application traffic and also supports multicast networking.
- It eliminates the time and cost complexity.

Thingspeak IoT analytics Platform

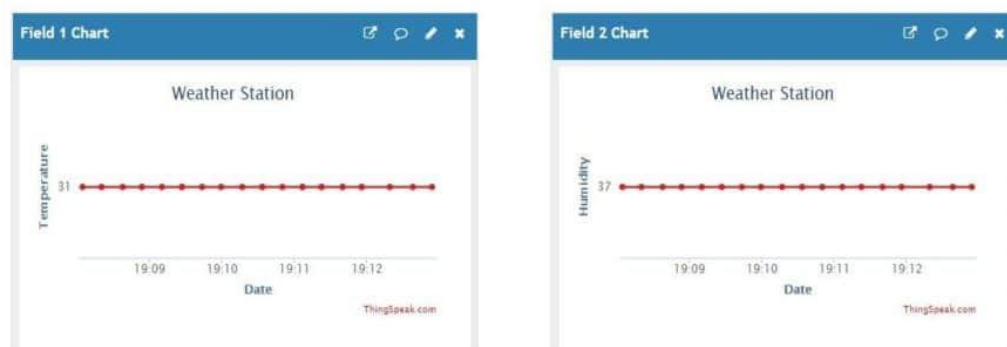
Thingspeak is open source platform made for Internet of Things (IoT) device developers and learners where developers can send and log data to the server, analyse, retrieve and store results using graphs with matlab support.

Client can send up to 8 data simultaneously to the thingspeak account; the data uploaded will be converted to graphical representation automatically as illustrated below:

Channel Stats

Created: 5 days ago
 Updated: 9 minutes ago
 Last entry: less than a minute ago
 Entries: 16

Electronics-project-hub.com



Sending data to Thingspeak

The straight line signifies that there was no temperature or humidity change during data logging. Client can over the mouse cursor on these dots to see the numerical data recorded at a particular local time. Thingspeak updates the data/graph every 15 seconds.

To send data to thingspeak Client need something called API key from Client account which needs to be inserted in the given program code.

API key in Thingspeak

API stands for “Application programming Interface”, on thingspeak it is a string of random character contains alphabets of lower and upper case, numbers and even special characters to identify Client account and ensures that Client sent data doesn’t end on someone else’s account and vice-versa.

Sign up for Thingspeak and get API Key

- Signing up for thingspeak is simple, just click this link: https://thingspeak.com/users/sign_up and fill the necessary fields.
- Once this is done a verification link will be sent to e-mail and click the received link.

Now go to Thingspeak account and click on the API key tab and client can see their read and write API keys. Always keep client write API keys confidential.

Thingspeak™ Channels Apps Community Support Commercial Use How to Buy Account Sign Out

Private View Public View Channel Settings Sharing **API Keys** Data Import / Export

Write API Key

Key PV 0IA28

Generate New Write API Key

Read API Keys

Key 3GK IVO

Note

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Update a Channel Feed

`https://api.thingspeak.com/update?api_key=PV0IA28field1=0`

Getting API key from Thinkgspeak

- Now click the channel settings, Client will now enable the two channels (graphs) and will be writing necessary labels to make our readings easier. Just do the following changes in client account as shown below:

[Private View](#) [Public View](#) [Channel Settings](#) [Sharing](#) [API Keys](#)

Channel Settings

Percentage complete 30%

Channel ID 573678


Name

Description

Field 1	<input type="text" value="Temperature"/>	<input checked="" type="checkbox"/>
Field 2	<input type="text" value="Humidity"/>	<input checked="" type="checkbox"/>
Field 3	<input type="text"/>	<input type="checkbox"/>
Field 4	<input type="text"/>	<input type="checkbox"/>

Editing Thingspeak Channels

- - After making the changes scroll down and press “save”.



Video URL

Show Status ☐

Now let's learn about “APN” which is necessary for enabling the GSM modem to access internet.

APN

APN stands for **Access Point Name**. It is the gateway between carrier's network and internet. APN helps in getting the IP address and decides which security protocols should be used.

For example:

- Airtel: airtelgrps.com (Tested)
- BSNL: bsnlnet (Tested)