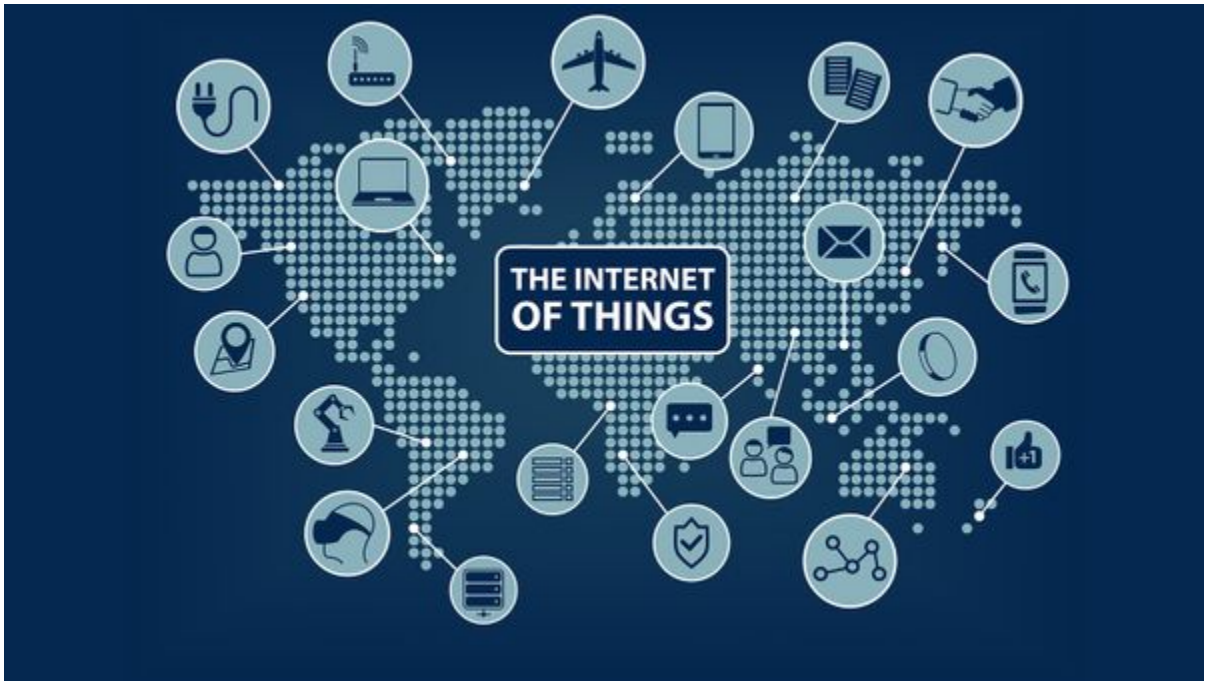
The Internet of Things (IOT)

Table of Contents

The Internet of Things (IoT)

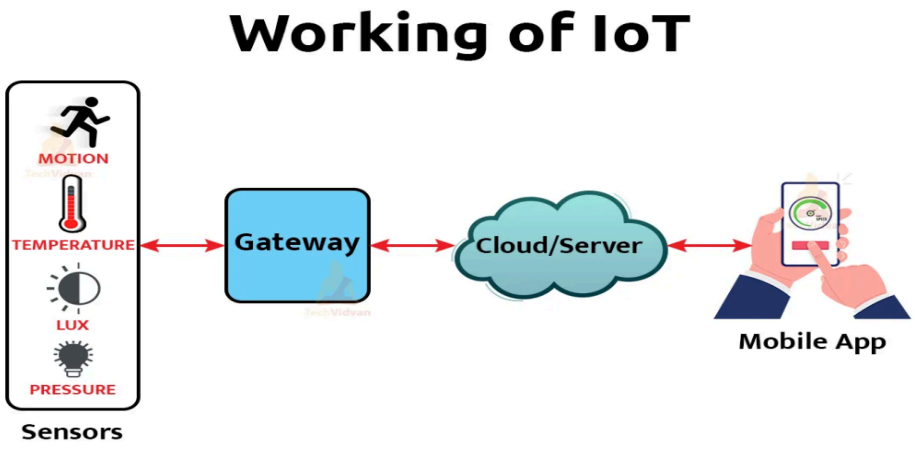


The Internet of Things (IoT) is a term for physical objects with sensors and actuators. These objects can connect to computers through wired or wireless networks, which let computers watch or even control the real world. These gadgets range from commonplace home items to highly advanced industrial instruments. A wireless network linking items like home appliances is called the "Internet of Things," sometimes known as the "Internet of Objects." Typically, this network is wireless and self-configuring.

How does IoT work?

Connected gadgets form the backbone of what is known as the Internet of Things. Sensors are incorporated into IoT devices. These sensors can sense their environment. The information is kept on the devices as data in some format. These devices consist of automobiles, coffee makers, microwaves, mobile phones, geysers, fire detectors, air conditioners, and automobiles.

These devices' built-in sensors constantly beep out information about their surroundings and how they're functioning. The data these devices have gathered can be dumped onto the Internet of Things.



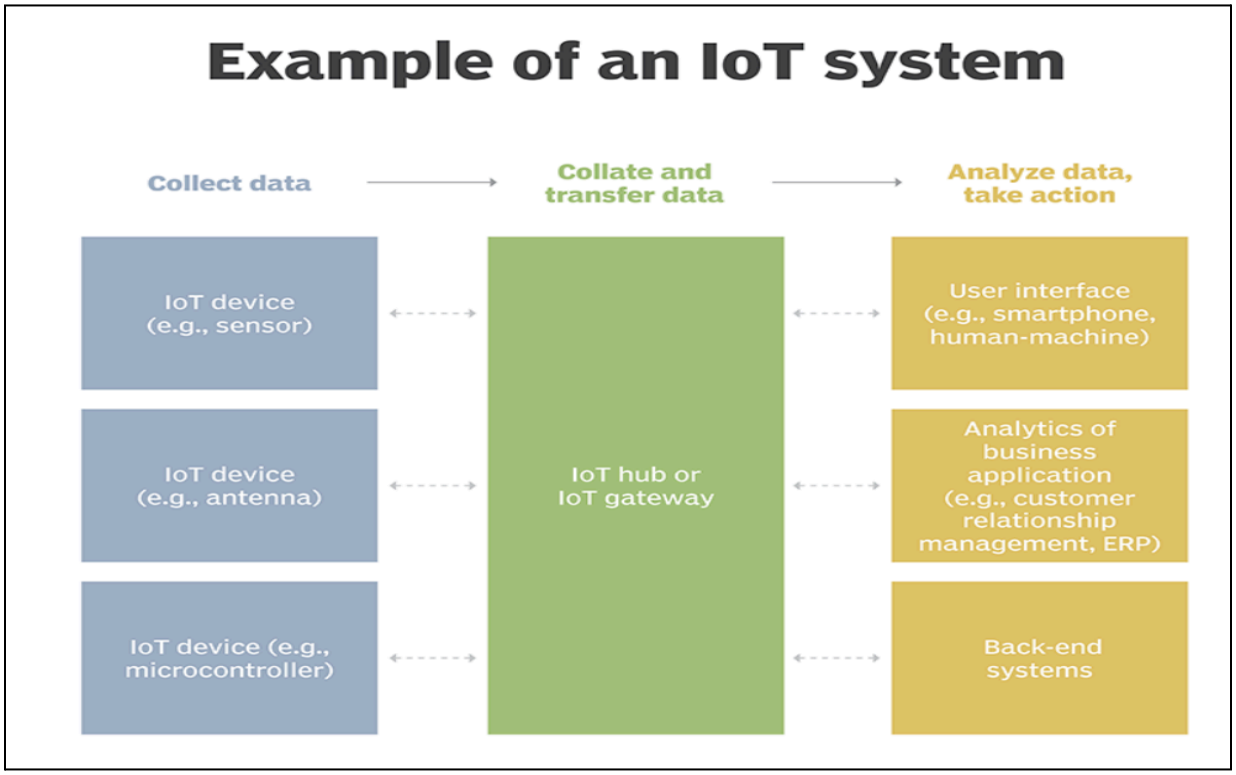
In order to share the data collected by their sensors, Internet of Things devices link to an IoT gateway, which acts as a central hub for data transfer among IoT devices. The data may also be sent to an edge device for local processing before sharing. Analysis of local data reduces cloud data transmission, thereby decreasing bandwidth consumption.

Every once in a while, these devices share information and act on that information. The gadgets can function mostly independently, but they can still be programmed, given instructions, or given access to data by human interaction.

Components of Internet of Things platforms include large databases and servers hosted in the cloud. The IoT platform works with the data. It processes the data after assembling it. The platform analyzes the data thoroughly to uncover essential details.

The platform generates

instructions in response to data. Lastly, other devices receive the collected data in order to improve their performance from now on. It's also done to improve the user experience.



The IoT has a significant and promising future ahead of it. In 2020, 24 billion IoT devices were installed, according to a Business Insider article. ITC forecasts that revenue from the Internet of Things will reach 300 billion dollars in the years to come.

This creates a great deal of employment prospects in the technology sector as well as in other businesses. IoT functions, in short, are as follows:

- Devices make use of hardware, such as sensors, in order to gather information.
- Next, the sensors upload their data to the cloud, where various programs can access and use it.
- After that, the software analyzes the data and sends it to users through a website or an application.

According to the specialists at IoT For All, "the support software that connects everything in an IoT system" is the best way to characterize smart devices' Internet of Things platform. Industry giants like IBM and Oracle, among others, are developing some IoT platforms.

Technologies involved in iot development

Internet/web and networking basics osi model

- Networking technologies enable IoT devices to communicate with other devices, applications, and services running in the cloud.
- The internet relies on standardized protocols to ensure communication between heterogeneous devices is secure and reliable.
- Standard protocols specify rules and formats that devices use to establish and manage networks and transmit data across those networks.
- Networks are built as a “stack” of technologies. A technology such as Bluetooth LE is at the bottom of the stack.
- While others such as such as IPv6 technologies (which is responsible for the logical device addressing and routing of network traffic) are further up the stack. Technologies at the top of the stack are used by the applications that are running on top of those layers, such as message queuing technologies.
- This article describes widely adopted technologies and standards for IoT networking. It also provides guidance for choosing one network protocol over another. It then discusses key considerations and challenges related to networking within IoT: range, bandwidth, power usage, intermittent connectivity, interoperability, and security.

Networking standards and technologies

- The Open Systems Interconnection (OSI) model is an ISO-standard abstract model is a stack of seven protocol layers.

- From the top down, they are: application, presentation, session, transport, network, data link and physical. TCP/IP, or the Internet Protocol suite, underpins the internet, and it provides a simplified concrete implementation of these layers in the OSI model.

The TCP/IP model includes only four layers, merging some of the OSI model layers:

- **Network Access & Physical Layer**

This TCP/IP Layer subsumes both OSI layers 1 and 2. The physical (PHY) layer (Layer 1 of OSI) governs how each device is physically connected to the network with hardware, for example with an optic cable, wires, or radio in the case of wireless network like wifi IEEE 802.11 a/b/g/n). At the link layer (Layer 2 of OSI), devices are identified by a MAC address, and protocols at this level are concerned with physical addressing, such as how switches deliver frames to devices on the network.

- **Internet Layer**

This layer maps to the OSI Layer 3 (network layer). OSI Layer 3 relates to logical addressing. Protocols at this layer define how routers deliver packets of data between source and destination hosts identified by IP addresses. IPv6 is commonly adopted for IoT device addressing.

- **Transport Layer**

The transport layer (Layer 4 in OSI) focuses on end-to-end communication and provides features such as reliability, congestion avoidance, and guaranteeing that packets will be delivered in the same order that they were sent. UDP (User Datagram protocol) is often adopted for IoT transport for performance reasons.

- **Application Layer**

The application layer (Layers 5, 6, and 7 in OSI) covers application-level messaging. HTTP/S is an example of an application layer protocol that is widely adopted across the internet.

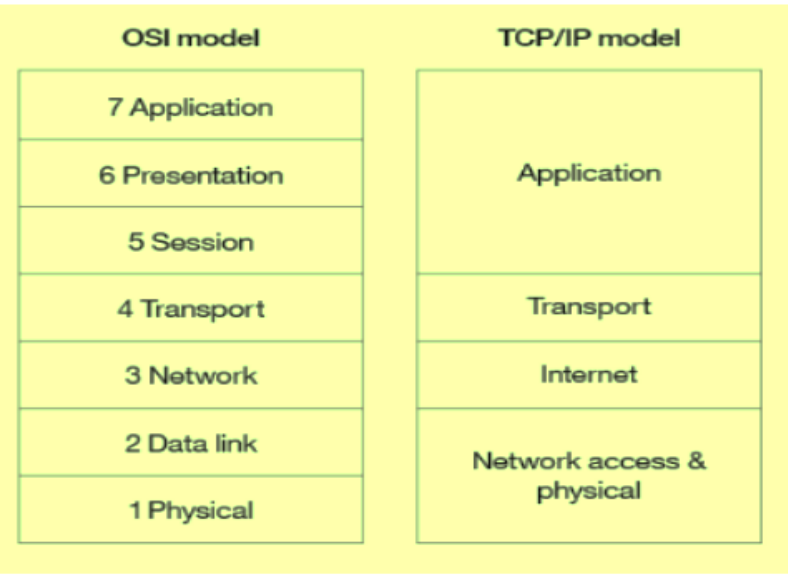


Figure 1. OSI and TCP/IP networking models

Although the TCP/IP and OSI models provide you with useful abstractions for discussing networking protocols and specific technologies that implement each protocol, some protocols don't fit neatly into these layered models and are impractical. For example, the Transport Layer Security (TLS) protocol that implements encryption to ensure privacy and data integrity of network traffic can be considered to operate across OSI layers 4, 5, and 6.

Network access and physical layer iot network technologies

IoT network technologies to be aware of toward the bottom of the protocol stack include cellular, Wifi, and Ethernet, as well as more specialized solutions such as LPWAN, Bluetooth Low Energy (BLE), ZigBee, NFC, and RFID.

NB-IoT is becoming the standard for LPWAN networks, according to Gartner. This IoT for All article tells more about NB-IoT.

The following are network technologies with brief descriptions of each:

- **LPWAN (Low Power Wide Area Network)**

LPWAN is a category of technologies designed for lowpower, long-range wireless communication. They are ideal for large-scale deployments of low-power IoT devices such as wireless sensors. LPWAN technologies include LoRa (LongRange physical layer protocol), Haystack, SigFox, LTE-M, and NB-IoT (Narrow-Band IoT).

- **Cellular**

The LPWAN NB-IoT and LTE-M standards address low-power, low-cost IoT communication options using existing cellular networks. NB-IoT is the newest of these standards and is focused on long-range communication between large numbers of primarily indoor devices. LTE-M and NB-IoT were developed specifically for IoT, however existing cellular technologies are also frequently adopted for long-range wireless communication. While this has included 2G (GSM) in legacy devices (and currently being phased out), CDMA (also being retired or phased out), it also includes 3G, which is rapidly being phased out with several network providers retiring all 3G devices. 4G is still active and will be until 5G becomes fully available and implemented.

- **Bluetooth Low Energy (BLE)**

BLE is a low-power version of the popular Bluetooth 2.4 GHz wireless communication protocol. It is designed for short-range (no more than 100 meters) communication, typically in a star configuration, with a single primary device that controls several secondary devices. Bluetooth operates across both layers 1 (PHY) and 2 (MAC) of the OSI model. BLE is best suited to devices that transmit low volumes of data in bursts. Devices are designed to sleep and save power when they are not transmitting data. Personal IoT devices such as wearable health and fitness trackers, often use BLE.

- **ZigBee**

ZigBee operates on 2.4GHz wireless communication spectrum. It has a longer range

than BLE by up to 100 meters. It also has a slightly lower data rate (250 kbps maximum compared to 270 kbps for BLE) than BLE. ZigBee is a mesh network protocol. Unlike BLE, not all devices can sleep between bursts. Much depends on their position in the mesh and whether they need to act as routers or controllers within the mesh. ZigBee was designed for building and home automation applications. Another closely related technology to ZigBee is Z-Wave, which is also based on IEEE 802.15.4. Z-Wave was designed for home automation. It has been proprietary technology, but was recently released as a public domain specification.

- **NFC**

The near field communication (NFC) protocol is used for very small range communication (up to 4 cm), such as holding an NFC card or tag next to a reader. NFC is often used for payment systems, but also useful for check-in systems and smart labels in asset tracking.

- **RFID**

RFID stands for Radio Frequency Identification. RFID tags store identifiers and data. The tags are attached to devices and read by an RFID reader. The typical range of RFID is less than a meter. RFID tags can be active, passive, or assisted passive. Passive tags are ideal for devices without batteries, as the ID is passively read by the reader. Active tags periodically broadcast their ID, while assisted passive tags become active when RFID reader is present. Dash7 is a communication protocol that uses active RFID that is designed to be used within Industrial IoT applications for secure long-range communication. Similar to NFC, a typical use case for RFID is tracking inventory items within retail and industrial IoT applications.

- **Wifi**

Wifi is standard wireless networking based on IEEE 802.11a/b/g/n specifications.

802.11n offers the highest data throughput, but at the cost of high-power consumption, so IoT devices might only use 802.11b or g for power conservation reasons. Although wifi is adopted within many prototype and current generation IoT devices, as longer-range and lower-power solutions become more widely available, it is likely that wifi will be superseded by lower-power alternatives.

- **Ethernet**

Widely deployed for wired connectivity within local area networks, Ethernet implements the IEEE 802.3 standard. Not all IoT devices need to be stationery wireless . For example, sensor units installed within a building automation system can use wired networking technologies like Ethernet. Power line communication (PLC), an alternative hard-wired solution, uses existing electrical wiring instead of dedicated network cables.

Internet layer iot network technologies

Internet layer technologies (OSI Layer 3) identify and route packets of data. Technologies commonly adopted for IoT are related to this layer, and include IPv6, 6LoWPAN, and RPL.

- **IPv6**

At the Internet layer, devices are identified by IP addresses. IPv6 is typically used for IoT applications over legacy IPv4 addressing. IPv4 is limited to 32-bit addresses, which only provide around 4.3 billion addresses in total, which is less than the current number of IoT devices that are connected, while IPv6 uses 128 bits, and so provides 2 128 addresses (around 3.4 × 10 38 or 340 billion billion billion billion) addresses. In practice, not all IoT devices need public addresses. Of the tens of billions of devices expected to connect via the IoT over the next few years, many will be deployed in private networks that use private address ranges and only communicate out to other devices or services on external networks by using Gateways.

- **6LoWPAN**

The IPv6 Low Power Wireless Personal Area Network (6LoWPAN) standard allows IPv6 to be used over 802.15.4 wireless networks. 6LoWPAN is often used for wireless sensor networks, and the Thread protocol for home automation devices also runs over 6LoWPAN.

- **RPL**

The Internet Layer also covers routing. IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) is designed for routing IPv6 traffic over low-power networks like those networks implemented over 6LoWPAN. RPL (pronounced “ripple”) is designed for routing packets within constrained networks such as wireless sensor networks, where not all devices are reachable at all times and there are high or unpredictable amounts of packet loss. RPL can compute the optimal path by building up a graph of the nodes in the network based on dynamic metrics and constraints like minimizing energy consumption or latency.

Application layer iot network technologies

HTTP and HTTPS are ubiquitous across internet applications, which is true also within IoT, with RESTful HTTP and HTTPS interfaces widely deployed. CoAP (Constrained Application Protocol) is like a lightweight HTTP that is often used in combination with 6LoWPAN over UDP. Messaging protocols like MQTT, AMQP, and XMPP are also frequently used within IoT applications:

MQTT Message Queue Telemetry Transport (MQTT)

MQTT is a publish/subscribe-based messaging protocol that was designed for use in low bandwidth situations, particularly for sensors and mobile devices on unreliable networks.

AMQP Advanced Message Queuing Protocol (AMQP)

AMQP is an open standard messaging protocol that is used for message-oriented middleware. Most notably, AMQP is implemented by RabbitMQ.

The Extensible Messaging and Presence Protocol (XMPP)

XMPP was originally designed for real-time human-to-human communication including instant messaging. This protocol has been adapted for machine-to-machine (M2M) communication to implement lightweight middleware and for routing XML data. XMPP is primarily used with smart appliances.

Your choice of technologies at this layer will depend on the specific application requirements of your IoT project. For example, for a budget home automation system that involves several sensors, MQTT would be a good choice as it is great for implementing messaging on devices without much storage or processing power because the protocol is simple and lightweight to implement.

Iot networking considerations and challenges

When you consider which networking technologies to adopt within your IoT application, be mindful of the following constraints:

- Range
- Bandwidth
- Power usage
- Intermittent connectivity
- Interoperability
- Security

Range

Networks can be described in terms of the distances over which data is typically

transmitted by the IoT devices attached to the network:

- **PAN(PersonalAreaNetwork)**

PAN is short-range, where distances can be measured in meters, such as a wearable fitness tracker device that communicates with an app on a cell phone over BLE.

- **LAN(LocalAreaNetwork)**

LAN is short- to medium-range, where distances can be up to hundreds of meters, such as home automation or sensors that are installed within a factory production line that communicate over wifi with a gateway device that is installed within the same building.

- **MAN (Metropolitan Area Network)**

MAN is long-range (city wide), where distances are measured up to a few kilometers, such as smart parking sensors installed throughout a city that are connected in a mesh network topology.

- **WAN (Wide Area Network)**

WAN is long-range, where distances can be measured in kilometers, such as agricultural sensors that are installed across a large farm or ranch that are used to monitor micro-climate environmental conditions across the property.

Your network should retrieve data from the IoT devices and transmit to its intended destination. Select a network protocol that matches the range is required. For example, do not choose BLE for a WAN application to operate over a range of several kilometers. If transmitting data over the required range presents a challenge, consider edge computing. Edge computing analyzes data directly from the devices rather than from a distant data center or elsewhere.

Bandwidth

Bandwidth is the amount of data that can be transmitted per unit of time. It limits the rate at which data can be collected from IoT devices and transmitted upstream. Bandwidth is affected by many factors, which include:

- The volume of data each device gathers and transmits.
- The number of devices deployed .
- Whether data is being sent as a constant stream or in intermittent bursts, and if any peak periods are notable.

The packet size of the networking protocol should match up with the volume of data typically transmitted. It is inefficient to send packets padded with empty data. In contrast, there are overheads in splitting larger chunks of data up across too many small packets. Data transmission rates are not always symmetrical (that is, upload rates might be slower than download rates). So, if there is two-way communication between devices, data transmission needs to be factored in. Wireless and cellular networks are traditionally low bandwidth, so consider whether a wireless technology is the right choice for high-volume applications.

Consider whether all raw data must be transmitted. A possible solution is to capture less data by sampling less frequently. Thus, you'll capture fewer variables and may filter data from the device to drop insignificant data. If you aggregate the data before you transmit it, you reduce the volume of data transmitted. But this process affects flexibility and granularity in the upstream analysis. Aggregation and bursting are not always suitable for time-sensitive or latency-sensitive data. All of these techniques increase the data processing and storage requirements for the IoT device.

Power usage

Transmitting data from a device consumes power. Transmitting data over long ranges

requires more power than over a short range. You must consider the power source – such as a battery, solar cell, or capacitor – of a device and its total lifecycle. A long and enduring lifecycle will not only provide greater reliability but reduce operating cost. Steps may be taken to help achieve longer power supply lifecycles. For example, to prolong the battery life, you can put the device into sleep mode whenever it is idle. Another best practice is to model the energy consumption of the device under different loads and different network conditions to ensure that the device's power supply and storage capacity matches with the power that is required to transmit the necessary data by using the networking technologies that you adopted.

Intermittent connectivity

IoT devices aren't always connected. In some cases, devices are designed to connect periodically. However, sometimes an unreliable network might cause devices to drop off due to connectivity issues. Sometimes quality of service issues, such as dealing with interference or channel contention on a wireless network using a shared spectrum. Designs should incorporate intermittent connectivity and seek any available solutions to provide uninterrupted service, should that be a critical factor for IoT landscape design.

Interoperability

Devices work with other devices, equipment, systems, and technology; they are interoperable. With so many different devices connecting to the IoT, interoperability can be a challenge. Adopting standard protocols has been a traditional approach for maintaining interoperability on the Internet. Standards are agreed upon by industry participants and avoid multiple different designs and directions. With proper standards, and participants who agree to them, incompatibility issues, hence interoperability issues may be avoided.

However, for the IoT, standardization processes sometimes struggle to keep up with innovation and change. They are written and released based on upcoming versions of

standards that are still subject to change. Consider the ecosystem around the technologies: Are they widely adopted? Are they open versus proprietary? How many implementations are available?

Using these questions to plan your IoT networks help plan better interoperability for a more robust IoT network.

Security

Security is a priority. Selection of networking technologies that implement end-to-end security, including authentication, encryption, and open port protection is crucial. IEEE 802.15.4 includes a security model that provides security features that include access control, message integrity, message confidentiality, and replay protection, which are implemented by technologies based on this standard such as ZigBee.

Consider the following factors in shaping a secure and safe IoT network:

- **Authentication**

Adopt secure protocols to support authentication for devices, gateways, users, services, and applications. Consider using adopting the X.509 standard for device authentication.

- **Encryption**

If you are using wifi, use Wireless Protected Access 2 (WPA2) for wireless network encryption. You may also adopt a Private Pre-Shared Key (PPSK) approach. To ensure privacy and data integrity for communication between applications, be sure to adopt TLS or Datagram Transport-Layer Security (DTLS), which is based on TLS, but adapted for unreliable connections that run over UDP. TLS encrypts application data and ensures its integrity.

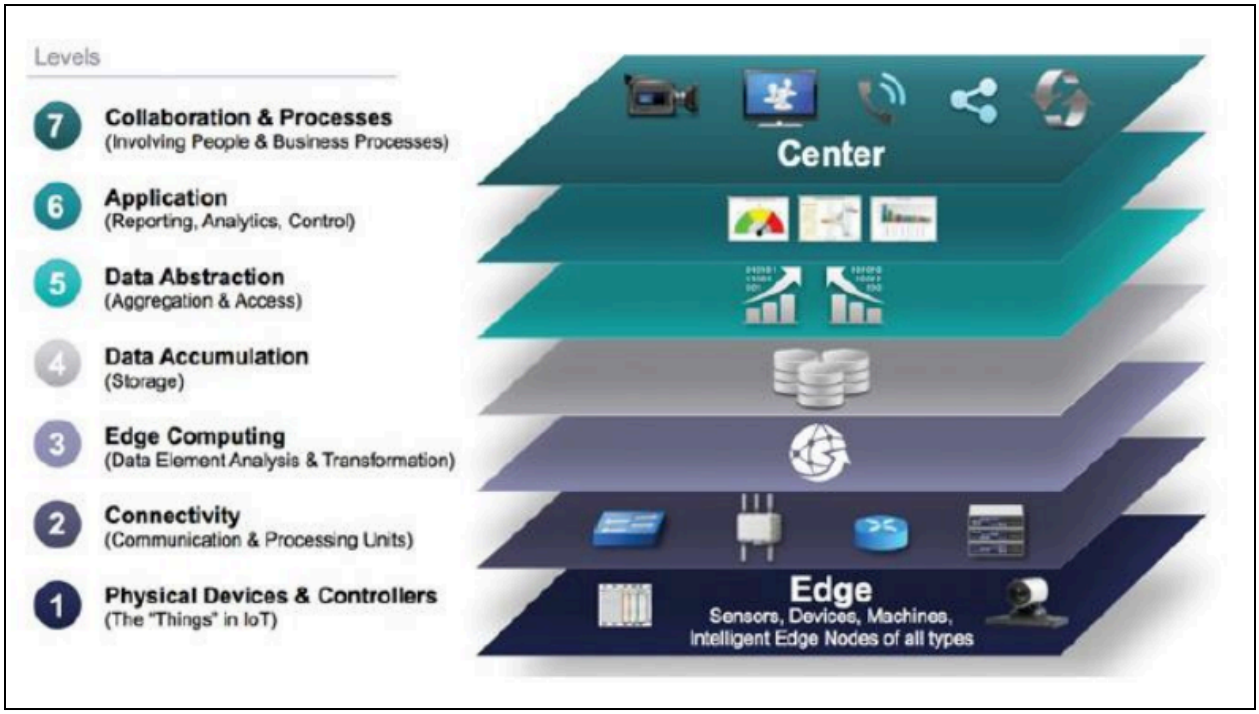
- **Port protection**

Port protection ensures that only the ports required for communication with the gateway or upstream applications or services remain open to external connections. All other ports should be disabled or protected by firewalls. Device ports might be exposed when

exploiting Universal Plug and Play (UPnP) vulnerabilities. Thus, UPnP should be disabled on the router.

The IoT World Forum (IoTWF) Standardized Architecture

In 2014 the IoTWF architectural committee (led by Cisco, IBM, Rockwell Automation, and others)published a seven-layer IoT architectural reference model. While various IoT reference models exist, theone put forth by the IoT World Forum offers a clean,simplified perspective on IoT and includes edgecomputing, data storage, and access. It provides a succinct way of visualizing IoT from a technicalperspective. Each of the seven layers is broken down into specific functions, and security encompassesthe entire model. Figure below details the IoT Reference Model published by the IoTWF.



As shown in Figure , the IoT Reference Model defines a set of levels with control flowing from thecenter (this could be either a cloud service or a dedicated data center), to the edge,

which includessensors, devices, machines, and other types of intelligent end nodes. In

general, data travels up the stack,originating from the edge, and goes northbound to the center. Using this reference model, we are able toachieve the following:

- Decompose the IoT problem into smaller parts.
- Identify different technologies at each layer and how they relate to one another.
- Define a system in which different parts can be provided by different vendors.
- Have a process of defining interfaces that leads to interoperability.
- Define a tiered security model that is enforced at the transition points between levels.

The following sections look more closely at each of the seven layers of the IoT Reference Model.

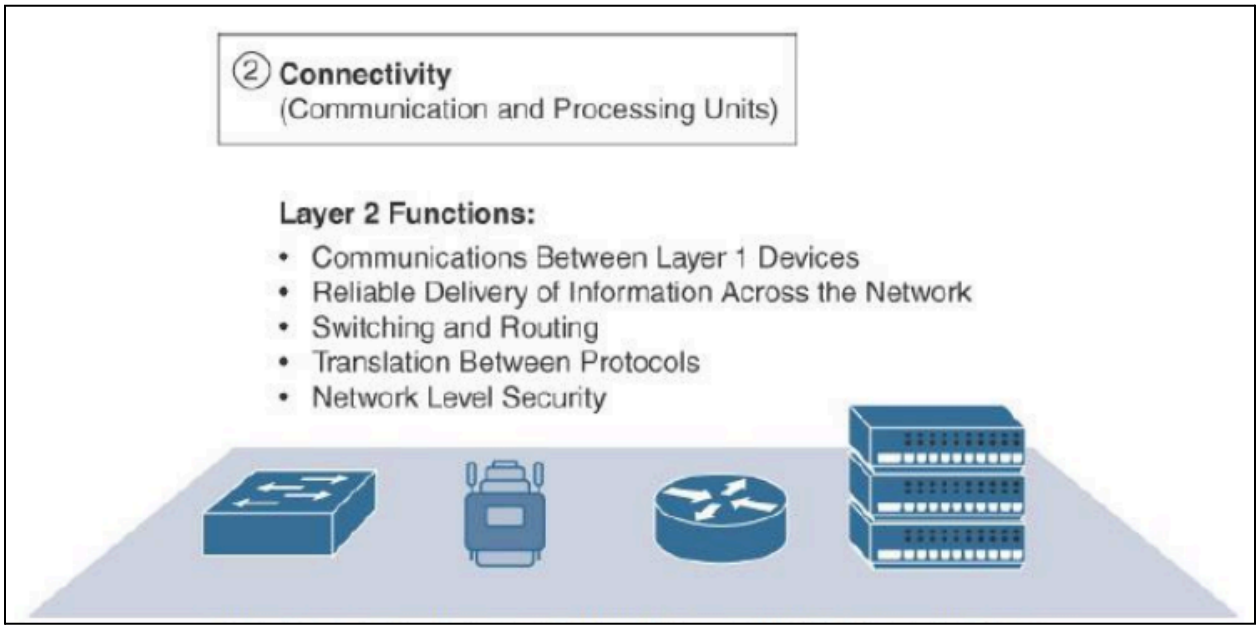
Layer 1: Physical Devices and Controllers Layer

The first layer of the IoT Reference Model is the physical devices and controllers layer. This layer is hometo the “things” in the Internet of Things, including the various endpoint devices and sensors that send andreceive information. The size of these “things” can range from almost microscopic sensors to giantmachines in a factory. Their primary function is generating data and being capable of being queriedand/or controlled over a network.

Layer 2: Connectivity Layer

In the second layer of the IoT Reference Model, the focus is on connectivity. The most important functionof this IoT layer is the reliable and timely transmission of data. More specifically, this includestransmissions between Layer 1 devices and the network and between the network and informationprocessing that occurs at Layer 3 (the edge computing layer).As you may notice, the connectivity layer encompasses all networking elements of IoT and doesn’t reallydistinguish between the last-mile network (the network between the sensor/endpoint and the IoT gateway,discussed later in this

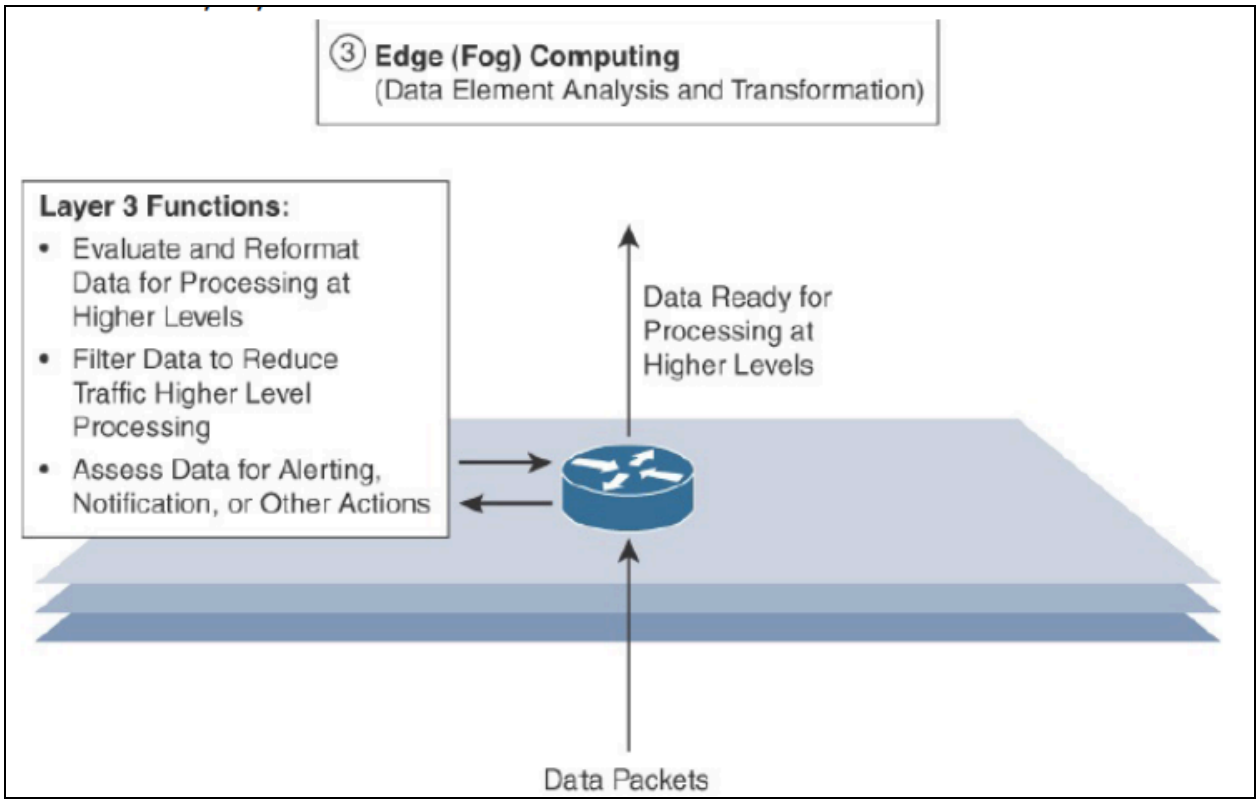
chapter), gateway, and backhaul networks. Functions of the connectivity layer aredetailed in Figure below.



Layer 3: Edge Computing Layer

Edge computing is the role of Layer 3. Edge computing is often referred to as the “fog” layer and is discussed in the section “Fog Computing,” later in this chapter. At this layer, the emphasis is on datareduction and converting network data flows into information that is ready for storage and processing byhigher layers. One of the basic principles of this reference model is that information processing is initiated as early and as close to the edge of the network as possible.

Figure below highlights the functions handledby Layer 3 of the IoT Reference Model.



Another important function that occurs at Layer 3 is the evaluation of data to see if it can be

filtered oraggregated before being sent to a higher layer. This also allows for data to be reformatted or decoded,making additional processing by other systems easier. Thus, a critical function is assessing the data to seeif predefined thresholds are crossed and any action or alerts need to be sent.

Upper Layers: Layers 4–7

The upper layers deal with handling and processing the IoT data generated by the bottom

layer. For thesake of completeness, Layers 4–7 of the IoT Reference Model are summarized in Table .

IoT Reference Model Layer	Functions
Layer 4: Data accumulation layer	Captures data and stores it so it is usable by applications when necessary. Converts event-based data to query-based processing.
Layer 5: Data abstraction layer	Reconciles multiple data formats and ensures consistent semantics from various sources. Confirms that the data set is complete and consolidates data into one place or multiple data stores using virtualization.
Layer 6: Applications layer	Interprets data using software applications. Applications may monitor, control, and provide reports based on the analysis of the data.
Layer 7: Collaboration and processes layer	Consumes and shares the application information. Collaborating on and communicating IoT information often requires multiple steps, and it is what makes IoT useful. This layer can change business processes and delivers the benefits of IoT.

M2M Communication

Machine-to-machine communication, or M2M, is exactly as it sounds: two machines "communicating," or exchanging data, without human interfacing or interaction. This includes serial connection, powerline connection (PLC), or wireless communications in the industrial Internet of Things (IoT). Switching over to wireless has made M2M communication much easier and enabled more applications to be connected.

In general, when someone says M2M communication, they often are referring to cellular communication for embedded devices. Examples of M2M communication in this case would be vending machines sending out inventory information or ATM machines getting authorization to despense cash.

As businesses have realized the value of M2M, it has taken on a new name: the Internet of Things (IoT). IoT and M2M have similar promises: to fundamentally change the way the world operates. Just like IoT, M2M allows virtually any sensor to communicate, which opens up the possibility of systems monitoring themselves and automatically responding to changes in the environment, with a much reduced need for

human involvement. M2M and IoT are almost synonymous—the exception is IoT (the newer term) typically refers to wireless communications, whereas M2M can refer to any two machines—wired or wireless—communicating with one another.

Traditionally, M2M focused on “industrial telematics,” which is a fancy way of explaining data transfer for some commercial benefit. But many original uses of M2M still stand today, like smart meters. Wireless M2M has been dominated by cellular since it came out in the mid-2000’s with 2G cell networks. Because of this, the cellular market has tried to brand M2M as an inherently cellular thing by offering M2M data plans. But cellular M2M is only one subsection of the market, and it shouldn’t be thought of as a cellular-only area.

How M2M Works

As previously stated, machine-to-machine communication makes the Internet of Things possible. According to Forbes, M2M is among the fastest-growing types of connected device technologies in the market right now, largely because M2M technologies can connect millions of devices within a single network. The range of connected devices includes anything from vending machines to medical equipment to vehicles to buildings. Virtually anything that houses sensor or control technology can be connected to some sort of wireless network.

This sounds complex, but the driving thought behind the idea is quite simple. Essentially, M2M networks are very similar to LAN or WAN networks, but are exclusively used to allow machines, sensors, and controls, to communicate. These devices feed information they collect back to other devices in the network. This process allows a human (or an intelligent control unit) to assess what is going on across the whole network and issue appropriate instructions to member devices.

M2M Applications

The possibilities in the realm of M2M can be seen in four major use cases, which we've detailed below:

Manufacturing

Every manufacturing environment—whether it's food processing or general product manufacturing—relies on technology to ensure costs are managed properly and processes are executed efficiently. Automating manufacturing processes within such a fast-paced environment is expected to improve processes even more. In the manufacturing world, this could involve highly automated equipment maintenance and safety procedures.

For example, M2M tools allow business owners to be alerted on their smartphones when an important piece of equipment needs servicing, so they can address issues as quickly as they arise. Sophisticated networks of sensors connected to the Internet could even order replacement parts automatically.

Home appliances

IoT already affects home appliance connectivity through platforms like Nest. However, M2M is expected to take home-based IoT to the next level. Manufacturers like LG and Samsung are already slowly unveiling smart home appliances to help ensure a higher quality of life for occupants.

For example, an M2M-capable washing machine could send alerts to the owners' smart devices once it finishes washing or drying, and a smart refrigerator could automatically order groceries from Amazon once its inventory is depleted. There are many more examples of home automation that can potentially improve quality of life for residents, including systems that allow members of the household to remotely control HVAC systems using their mobile devices. In situations where a homeowner decides to leave work early, he or she could contact the home heating system before leaving work to make sure the temperature at home will be comfortable upon arrival.

Healthcare device management

One of the biggest opportunities for M2M technology is in the realm of health care. With M2M technology, hospitals can automate processes to ensure the highest levels of treatment. Using devices that can react faster than a human healthcare professional in an emergency situation make this possible. For instance, when a patient's vital signs drop below normal, an M2M-connected life support device could automatically administer oxygen and additional care until a healthcare professional arrives on the scene. M2M also allows patients to be monitored in their own homes instead of in hospitals or care centers. For example, devices that track a frail or elderly person's normal movements can detect when he or she has had a fall and alert a healthcare worker to the situation

Smart utility management

In the new age of energy efficiency, automation will quickly become the new normal. As energy companies look for new ways to automate the metering process, M2M comes to the rescue, helping energy companies automatically gather energy consumption data, so they can accurately bill customers. Smart meters can track how much energy a household or business uses and automatically alert the energy company, which supplants sending out an employee to read the meter or requiring the customer to provide a reading. This is even more important as utilities move toward more dynamic pricing models, charging consumers more for energy usage during peak times.

A few key analysts predict that soon, every object or device will need to be able to connect to the cloud. This is a bold but seemingly accurate statement. As more consumers, users, and business owners demand deeper connectivity, technology will need to be continually equipped to meet the needs and challenges of tomorrow. This will empower a wide range of highly automated processes, from equipment repairs and firmware upgrades to system diagnostics, data retrieval, and analysis. Information will be delivered to users, engineers, data scientists, and key decision-makers in real time, and it will eliminate the need for guesswork.

The Value Of M2M

Growth in the M2M and IoT markets has been growing rapidly, and according to many reports, growth will continue. Strategy Analytics believes that low power, wide-area network (LPWAN) connections will grow from 11 million in 2014 to 5 billion in 2022. And IDC says the market for worldwide IoT solutions will go from $1.9 trillion in 2013 to $7.1 trillion in 2020.

Many big cell operators, like AT&T and Verizon, see this potential and are rolling out their own M2M platforms. Intel, PTC, and Wipro are are all marketing heavily in M2M and working to take advantage of this major industry growth spurt. But there is still a great opportunity for new technology companies to engage in highly automated solutions to help streamline processes in nearly any type of industry. We're certain we'll see a huge influx of companies who begin to innovate in this area in the next five years.

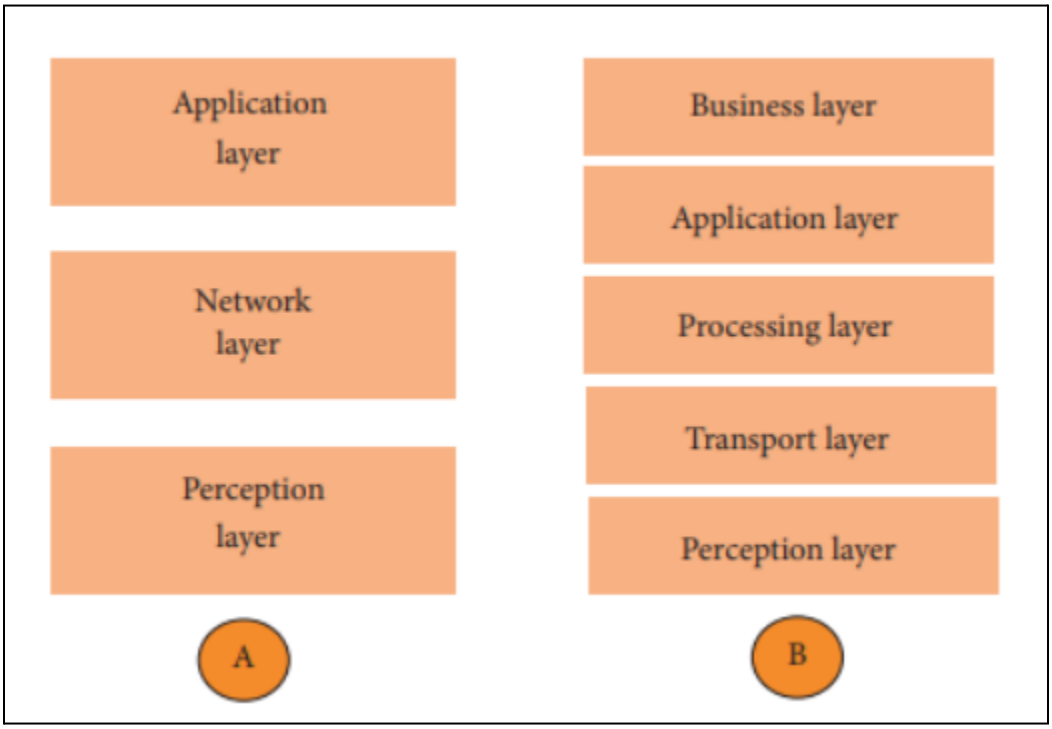
However, as the cost of M2M communication continues to decrease, companies must determine how they will create value for businesses and customers. In our mind, the opportunity and value for M2M doesn't lie in the more traditional layers of the communication world. Cell carriers and hardware manufacturers, for example, are beginning to look into full-stack offerings that enable M2M and IoT product development. We strongly believe value lies in the application side of things, and the growth in this industry will be driven by smart applications from this point forward.

Companies shouldn't think about IoT or M2M for the sake of IoT or M2M. Instead, they should focus on optimizing their business models or providing new value for their customers. For example, if you're a logistics company like FedEx or UPS, you have obvious choices for automated logistics decisions made by machines. But if you're a retailer, the transition to automation may not be as obvious. It's one thing to think of a "cool" automated process—say, creating advertising that is automatically tied to a specific customer through the use of M2M technology—but before you move forward with the process, you have to consider the value you're getting out of it. How much does it cost to implement? Will any company considering a move into the IoT space needs to understand what its business model is, how it will make money, and how it will provide value for customers or internal processes.

Architecture of IoT

Figure below has three layers, namely, the perception, network, and application layers.

- The perception layer is the physical layer, which has sensors for sensing and gathering information about the environment. It senses some physical parameters or identifies other smart objects in the environment.
- The network layer is responsible for connecting to other smart things, network devices, and servers. Its features are also used for transmitting and processing sensor data.
- The application layer is responsible for delivering application specific services to the user. It defines various applications in which the Internet of Things can be deployed, for example, smart homes, smart cities, and smart health.



The three-layer architecture defines the main idea of the Internet of Things, but it is not sufficient for research on IoT because research often focuses on finer aspects of the Internet of Things. That is why, we have many more layered architectures proposed in

the literature. One is the fivelayer architecture, which additionally includes the processing and business layers [3–6]. The five layers are perception, transport, processing, application, and business layers (see Figure 1). The role of the perception and application layers is the same as the architecture with three layers. We outline the function of the remaining three layers.

- The transport layer transfers the sensor data from the perception layer to the processing layer and vice versa through networks such as wireless, 3G, LAN, Bluetooth, RFID, and NFC.
- The processing layer is also known as the middleware layer. It stores, analyzes, and processes huge amounts of data that comes from the transport layer. It can manage and provide a diverse set of services to the lower layers. It employs many technologies such as databases, cloud computing, and big data processing modules.
- The business layer manages the whole IoT system, including applications, business and profit models, and users' privacy. The business layer is out of the scope of this paper. Hence, we do not discuss it further.

Core IoT Functional Stack

The IoT network must be designed to support its unique requirements and constraints. This section provides an overview of the full networking stack, from sensors all the way to the applications layer.

The Core IoT Functional Stack IoT networks are built around the concept of "things," or smart objects performing functions and delivering new connected services. These objects are "smart" because they use a combination of contextual information and configured goals to perform actions. These actions can be self-contained (that is, the smart object does not rely on external systems for its actions); however, in most cases, the "thing" interacts with an external system to report information that the smart object collects, to exchange with other objects, or to interact with a management platform. In this case, the management platform can be used to process data collected from the smart object and also guide the behavior of the smart object. From an architectural

standpoint, several components have to work together for an IoT network to be operational:

“Things” layer

At this layer, the physical devices need to fit the constraints of the environment in which they are deployed while still being able to provide the information needed.

Communications network layer

When smart objects are not self-contained, they need to communicate with an external system. In many cases, this communication uses a wireless technology. This layer has four sublayers: Access network sublayer: The last mile of the IoT network is the access network. This is typically made up of wireless technologies such as 802.11ah, 802.15.4g, and LoRa. The sensors connected to the access network may also be wired.

Gateways and backhaul network sublayer

A common communication system organizes multiple smart objects in a given area around a common gateway. The gateway communicates directly with the smart objects. The role of the gateway is to forward the collected information through a longerrange medium (called the backhaul) to a headend central station where the information is processed. This information exchange is a Layer 7 (application) function, which is the reason this object is called a gateway. On IP networks, this gateway also forwards packets from one IP network to another, and it therefore acts as a router. Network transport sublayer: For communication to be successful, network and transport layer protocols such as IP and UDP must be implemented to support the variety of devices to connect and media to use.

IoT network management sublayer

Additional protocols must be in place to allow the headend applications to exchange data with the sensors. Examples include CoAP and MQTT.

Application and analytics layer

At the upper layer, an application needs to process the collected data, not only to control the smart objects when necessary, but to make intelligent decision based on the information collected and, in turn, instruct the “things” or other systems to adapt to the analyzed conditions and change their behaviors or parameters. The following sections examine these elements and help you architect your IoT communication network.

Layer 1: Things: Sensors and Actuators Layer

Most IoT networks start from the object, or “thing,” that needs to be connected.From an architectural standpoint, the variety of smart object types, shapes, and needs drive the variety of IoT protocols and architectures. There are myriad ways to classify smart objects.

One architectural classification could be:

- **Battery-powered or power-connected:**

This classification is based on whether the object carries its own energy supply or receives continuous power from an external power source. Battery- powered things can be moved more easily than linepowered objects. However, batteries limit the lifetime and amount of energy that the object is allowed to consume, thus driving transmission range and frequency.

- **Mobile or static:**

This classification is based on whether the “thing” should move or always stay at the same location. A sensor may be mobile because it is moved from one object to another (for example, a viscosity sensor moved from batch to batch in a chemical plant) or because it is attached to a moving object (for example, a location sensor on moving goods in a warehouse or factory floor). The frequency of the movement may also vary, from occasional to permanent. The range of mobility (from a few inches to miles away) often drives the possible power source.

- **Low or high reporting frequency:**

This classification is based on how often the object should report monitored parameters. A rust sensor may report values once a month. A motion sensor may report acceleration several hundred times per second. Higher frequencies drive higher energy consumption, which may create constraints on the possible power source (and therefore the object mobility) and the transmission range.

- **Simple or rich data:**

This classification is based on the quantity of data exchanged at each report cycle. A humidity sensor in a field may report a simple daily index value (on a binary scale from 0 to 255), while an engine sensor may report hundreds of parameters, from temperature to pressure, gas velocity, compression speed, carbon index, and many others. Richer data typically drives higher power consumption. This classification is often combined with the previous to determine the object data throughput (low throughput to high throughput). You may want to keep in mind that throughput is a combined metric. A medium-throughput object may send simple data at rather high frequency (in which case the flow structure looks continuous), or may send rich data at rather low frequency (in which case the flow structure looks bursty).

- **Report range:**

This classification is based on the distance at which the gateway is located. For example, for your fitness band to communicate with your phone, it needs to be located a few meters away at most. The assumption is that your phone needs to be at visual distance for you to consult the reported data on the phone screen. If the phone is far away, you typically do not use it, and reporting data from the band to the phone is not necessary. By contrast, a moisture sensor in the asphalt of a road may need to communicate with its reader several hundred meters or even kilometers away.

- **Object density per cell:**

This classification is based on the number of smart objects (with a similar need to communicate) over a given area, connected to the same gateway. An oil pipeline may utilize a single sensor at key locations every few miles. By contrast, telescopes like the SETI Colossus telescope at the Whipple Observatory deploy hundreds, and sometimes thousands, of mirrors over a small area, each with multiple gyroscopes, gravity, and vibration sensors.

Layer 2: Communications Network Layer

Once you have determined the influence of the smart object form factor over its transmission capabilities (transmission range, data volume and frequency, sensor density and mobility), you are ready to connect the object and communicate. Compute and network assets used in IoT can be very different from those in IT environments. The difference in the physical form factors between devices used by IT and OT is obvious even to the most casual of observers. What typically drives this is the physical environment in which the devices are deployed. What may not be as inherently obvious, however, is their operational differences. The operational differences must be understood in order to apply the correct handling to secure the target assets. Temperature variances are an easily understood metric. The cause for the variance is

easily attributed to external weather forces and internal operating conditions. Remote external locations, such as those associated with mineral extraction or pipeline equipment can span from the heat of the Arabian Gulf to the cold of the Alaskan North Slope. Controls near the furnaces of a steel mill obviously require heat tolerance, and controls for cold food storage require the opposite.

In some cases, these controls must handle extreme fluctuations as well. These extremes can be seen within a single deployment. For example, portions of the Tehachapi, California, wind farms are located in the Mojave Desert, while others are at an altitude of 1800 m in the surrounding mountains. As you can imagine, the wide variance in temperature takes a special piece of hardware that is capable of withstanding such harsh environments. Humidity fluctuations can impact the longterm success of a system as well. Well heads residing in the delta of the Niger River will see very different conditions from those in the middle of the Arabian Desert. In some conditions, the systems could be exposed to direct liquid contact such as may be found with outdoor wireless devices or marine condition deployments. Less obvious are the operating extremes related to kinetic forces. Shock and vibration needs vary based on the deployment scenario. In some cases, the focus is on lowamplitude but constant vibrations, as may be expected on a bushing-mounted manufacturing system. In other cases, it could be a sudden acceleration or deceleration, such as may be experienced in peak ground acceleration of an earthquake or an impact on a mobile system such as high-speed rail or heavy-duty earth moving equipment. Solid particulates can also impact the gear. Most IT environments must contend with dust build-up that can become highly concentrated due to the effect of cooling fans. In less-controlled IT environments, that phenomenon can be accelerated due to higher concentrations of particulates.

A deterrent to particulate build-up is to use fanless cooling, which necessitates a higher surface area, as is the case with heat transfer fins. Hazardous location design may also cause corrosive impact to the equipment. Caustic materials can impact connections

over which power or communications travel. Furthermore, they can result in reduced thermal efficiency by potentially coating the heat transfer surfaces. In some scenarios, the concern is not how the environment can impact the equipment but how the equipment can impact the environment. For example, in a scenario in which volatile gases may be present, spark suppression is a critical design criterion. There is another class of device differentiators related to the external connectivity of the device for mounting or industrial function. Device mounting is one obvious difference between OT and IT environments. While there are rack mount environments in some industrial spaces, they are more frequently found among IT type assets. Within industrial environments, many compute and communication assets are placed within an enclosed space, such as a control cabinet where they will be vertically mounted on a DIN (Deutsches Institut für Normung) rail inside. In other scenarios, the devices might be mounted horizontally directly on a wall or on a fence. In contrast to most IT-based systems, industrial compute systems often transmit their state or receive inputs from external devices through an alarm channel. These may drive an indicator light (stack lights) to display the status of a process element from afar. This same element can also receive inputs to initiate actions within the system itself. Power supplies in OT systems are also frequently different from those commonly seen on standard IT equipment. A wider range of power variations are common attributes of industrial compute components. DC power sources are also common in many environments. Given the criticality of many systems, it is often required that redundant power supplies be built into the device itself. Extraneous power supplies, especially those not inherently mounted, are frowned upon, given the potential for accidental unplugging. In some utility cases, the system must be able to handle brief power outages and still continue to operate.

Access Network Sublayer

There is a direct relationship between the IoT network technology you choose and the type of connectivity topology this technology allows. Each technology was designed with a certain number of use cases in mind (what to connect, where to connect, how much

data to transport at what interval and over what distance). These use cases determined the frequency band that was expected to be most suitable, the frame structure matching the expected data pattern (packet size and communication intervals), and the possible topologies that these use cases illustrate. As IoT continues to grow exponentially, you will encounter a wide variety of applications and special use cases. For each of them, an access technology will be required. IoT sometimes reuses existing access technologies whose characteristics match more or less closely the IoT use case requirements. Whereas some access technologies were developed specifically for IoT use cases, others were not. One key parameter determining the choice of access technology is the range between the smart object and the information collector. Figure 2-9 lists some access technologies you may encounter in the IoT world and the expected transmission distances.

- **PAN (personal area network):**

Scale of a few meters. This is the personal space around a person. A common wireless technology for this scale is Bluetooth.

- **HAN (home area network):**

Scale of a few tens of meters. At this scale, common wireless technologies for IoT include ZigBee and Bluetooth Low Energy (BLE).

- **NAN (neighborhood area network):**

Scale of a few hundreds of meters. The term NAN is often used to refer to a group of house units from which data is collected.

- **FAN (field area network):**

Scale of several tens of meters to several hundred meters. FAN typically refers to an outdoor area larger than a single group of house units. The FAN is often seen as “open space” (and therefore not secured and not controlled). A FAN is sometimes viewed as a group of NANs, but some verticals see the FAN as a group of HANs or a group of smaller outdoor cells. As you can see, FAN and NAN may sometimes be used interchangeably. In most cases, the vertical context is clear enough to determine the

grouping hierarchy.

- **LAN (local area network):**

Scale of up to 100 m. This term is very common in networking, and it is therefore also commonly used in the IoT space when standard networking technologies (such as Ethernet or IEEE 802.11) are used. Other networking classifications, such as MAN (metropolitan area network, with a range of up to a few kilometers) and WAN (wide area network, with a range of more than a few kilometers), are also commonly used.

Network Transport Sublayer

The previous section describes a hierarchical communication architecture in which a series of smart objects report to a gateway that conveys the reported data over another medium and up to a central station. However, practical implementations are often flexible, with multiple transversal communication paths. For example, consider the case of IoT for the energy grid. Your house may have a meter that reports the energy consumption to a gateway over a wireless technology. Other houses in your neighborhood (NAN) make the same report, likely to one or several gateways. The data to be transported is small and the interval is large (for example, four times per hour), resulting in a low-mobility, lowthroughput type of data structure, with transmission distances up to a mile. Several technologies (such as 802.11ah, 802.15.4, or LPWA) can be used for this collection segment. Other neighborhoods may also connect the same way, thus forming a FAN.

IoT Network Management Sublayer

IP, TCP, and UDP bring connectivity to IoT networks. Upper-layer protocols need to take care of data transmission between the smart objects and other systems. Multiple protocols have been leveraged or created to solve IoT data communication problems. Some networks rely on a push model (that is, a sensor reports at a regular interval or based on a local trigger), whereas others rely on a pull model (that is, an application queries the sensor over the network), and multiple hybrid approaches are also possible.

Layer 3: Applications and Analytics Layer

Once connected to a network, your smart objects exchange information with other systems. As soon as your IoT network spans more than a few sensors, the power of the Internet of Things appears in the applications that make use of the information exchanged with the smart objects. Analytics Versus Control Applications Multiple applications can help increase the efficiency of an IoT network. Each application collects data and provides a range of functions based on analyzing the collected data. It can be difficult to compare the features offered.

From an architectural standpoint, one basic classification can be as follows:

Analytics application

This type of application collects data from multiple smart objects, processes the collected data, and displays information resulting from the data that was processed. The display can be about any aspect of the IoT network, from historical reports, statistics, or trends to individual system states. The important aspect is that the application processes the data to convey a view of the network that cannot be obtained from solely looking at the information displayed by a single smart object.

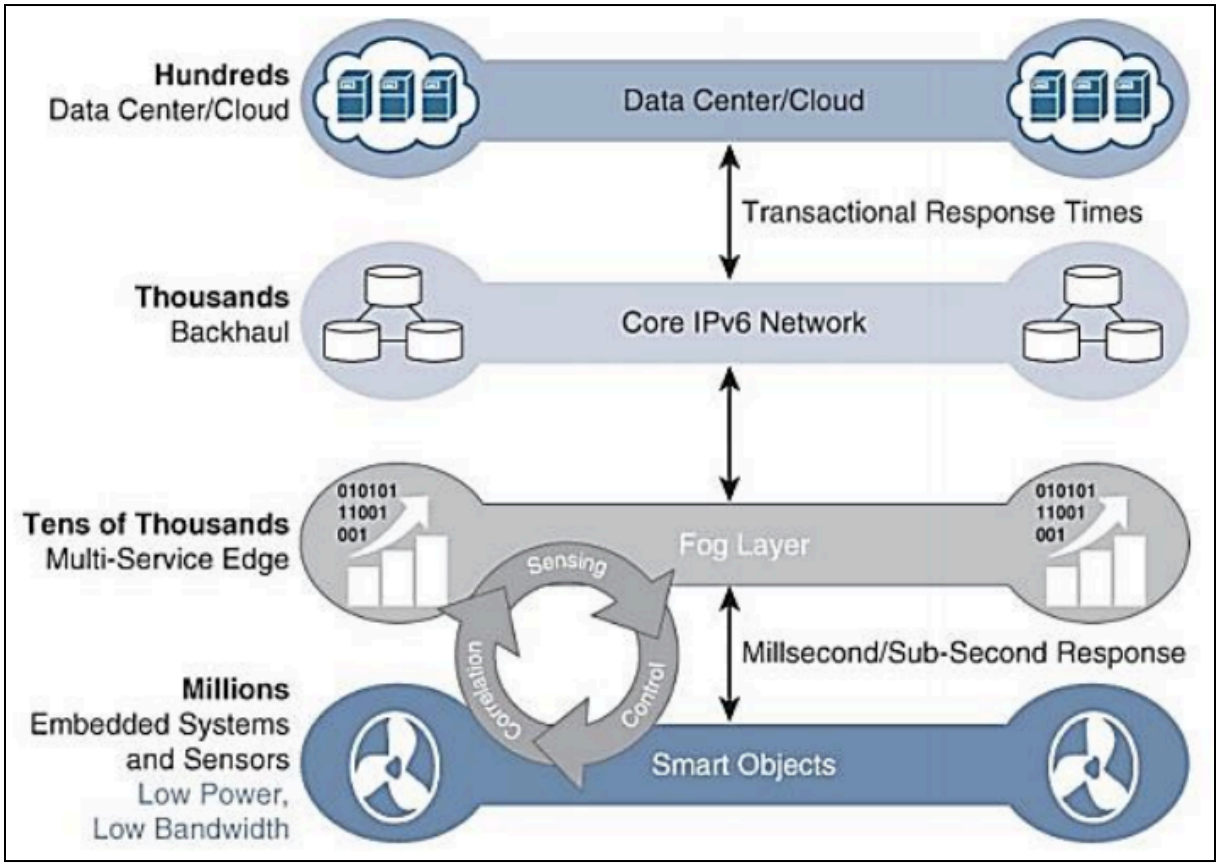
Control application

This type of application controls the behavior of the smart object or the behavior of an object related to the smart object. For example, a pressure sensor may be connected to a pump. A control application increases the pump speed when the connected sensor detects a drop in pressure. Control applications are very useful for controlling complex aspects of an IoT network with a logic that cannot be programmed inside a single IoT object, either because the configured changes are too complex to fit into the local system or because the configured changes rely on parameters that include elements outside the IoT object.

Fog Computing

The solution to the challenges mentioned in the previous section is to distribute data

management throughout the IoT system, as close to the edge of the IP network as possible. The best-known embodiment of edge services in IoT is fog computing. Any device with computing, storage, and network connectivity can be a fog node. Examples include industrial controllers, switches, routers, embedded servers, and IoT gateways. Analyzing IoT data close to where it is collected minimizes latency, offloads gigabytes of network traffic from the core network, and keeps sensitive data inside the local network.



Fog services are typically accomplished very close to the edge device, sitting as close to the IoT endpoints as possible. One significant advantage of this is that the fog node has contextual awareness of the sensors it is managing because of its geographic proximity to those sensors. For example, there might be a fog router on an oil derrick that is monitoring all the sensor activity at that location. Because the fog node is able to analyze information from all the sensors on that derrick, it can provide contextual analysis of the messages it is receiving and may decide to send back only the relevant

information over the backhaul network to the cloud. In this way, it is performing distributed analytics such that the volume of data sent upstream is greatly reduced and is much more useful to application and analytics servers residing in the cloud.

The defining characteristic of fog computing are as follows:

Contextual location awareness and low latency:

The fog node sits as close to the IoT endpoint as possible to deliver distributed computing.

Geographic distribution:

In sharp contrast to the more centralized cloud, the services and applications targeted by the fog nodes demand widely distributed deployments.

Deployment near IoT endpoints:

Fog nodes are typically deployed in the presence of a large number of IoT endpoints. For example, typical metering deployments often see 3000 to 4000 nodes per gateway router, which also functions as the fog computing node.

Wireless communication between the fog and the IoT endpoint:

Although it is possible to connect wired nodes, the advantages of fog are greatest when dealing with a large number of endpoints, and wireless access is the easiest way to achieve such scale.

Use for real-time interactions:

Important fog applications involve real-time interactions rather than batch processing. Preprocessing of data in the fog nodes allows upper-layer applications to perform batch processing on a subset of the data.

Edge Computing

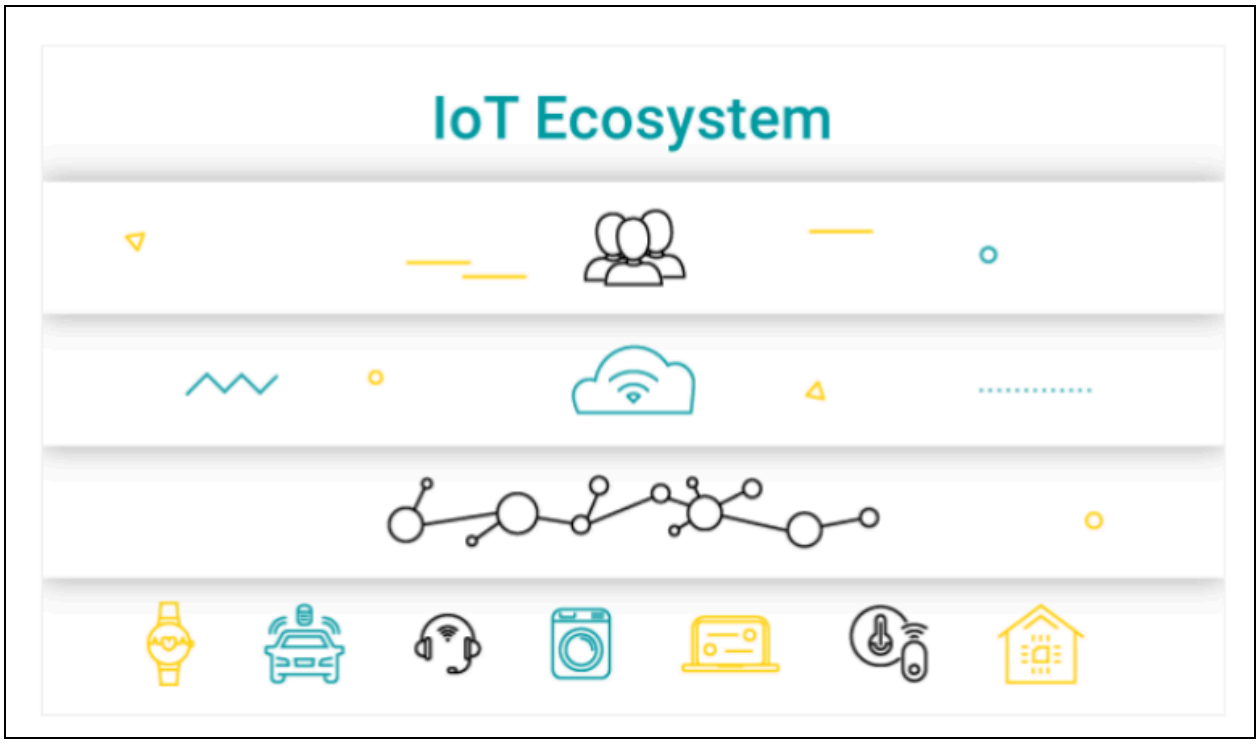
Fog computing solutions are being adopted by many industries, and efforts to develop distributed applications and analytics tools are being introduced at an accelerating pace.

The natural place for a fog node is in the network device that sits closest to the IoT endpoints, and these nodes are typically spread throughout an IoT network. However, in recent years, the concept of IoT computing has been pushed even further to the edge, and in some cases it now resides directly in the sensors and IoT devices.

Functional blocks of an IoT ecosystem

IoTdon’t exist in a void. A lone sensor isn’t really good for anything, nor is a bunch of them, for that matter, unless they are all connected to one another and to platforms that generate data for further use. This is what we call an Internet of Things (IoT) ecosystem – a broad network of connected and interdependent devices and technologies that are applied by specialists towards a specific goal, such as the creation of a smart city.

Obviously, there are limitless applications to the IoT and therefore we can speak of endless coexisting IoT ecosystems. But if you boil what is happening in the ecosystem down to the bare essentials, you will come up with a simple schema: a device collects data and sends it across the network to a platform that aggregates the data for future use by the agent. And so we have the key components to an IoT ecosystem: devices, networks, platforms, and agents. Let’s discuss them in more detail.



Four things form basic building blocks of the IoT system –sensors, processors, gateways, applications. Each of these nodes has to have its own characteristics in order to form an useful IoT system.

Sensors:

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

Processors:

- Processors are the brain of the IoT system. Their main function is to process the

data captured by the sensors and process them so as to extract the valuable data from the enormous amount of raw data collected. In a word, we can say that it gives intelligence to the data.

- Processors mostly work on real-time basis and can be easily controlled by applications. These are also responsible for securing the data – that is performing encryption and decryption of data.
- Embedded hardware devices, microcontroller, etc are the ones that process the data because they have processors attached to it.

Gateways:

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

Applications:

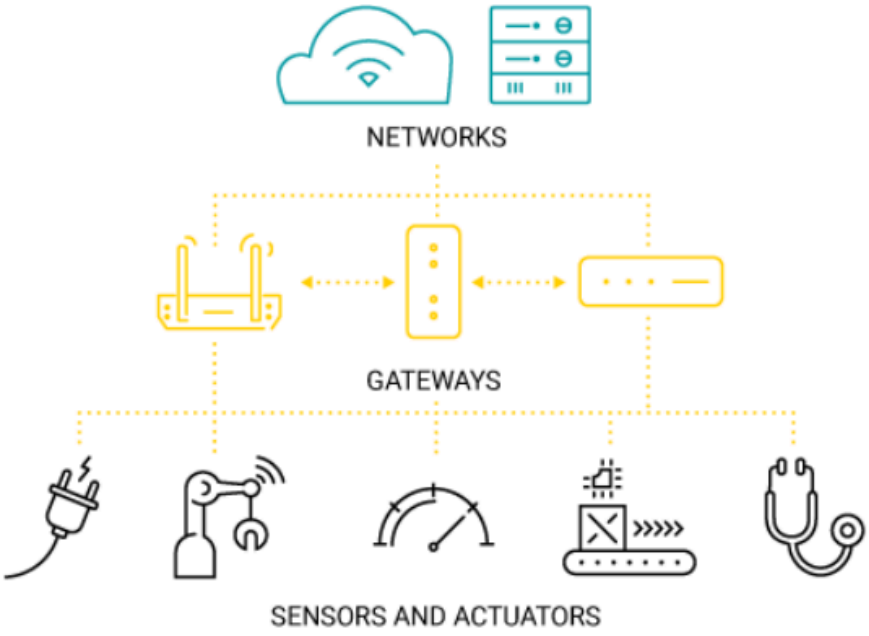
- Applications form another end of an IoT system. Applications are essential for proper utilization of all the data collected.
- These cloud-based applications which are responsible for rendering the effective meaning to the data collected. Applications are controlled by users and are a delivery point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

IoT devices

As we said earlier, there are many scenarios in which IoT can be employed and they all require different devices. Here, at the most basic level, we can speak of sensors (i.e. devices that sense things, such as temperature, motion, particles, etc.) and actuators (i.e. devices that act on things, such as switches or rotors).

Rarely, though, will a smart solution make do with just one type of an IoT sensor or an actuator. If you think of a smart surgical robot, for example, it will require hundreds, if not thousands, of components that measure different parameters and act accordingly. But even apparently less complicated solutions aren't truly that easy. Consider running a smart farm – for a plant to grow, it's not just a matter of measuring the humidity of the soil, but also its fertility; it's also a matter of providing proper irrigation based on insolation, and much more. So you need not just one, but many sensors and actuators that all have to work together.

When speaking of devices essential for the IoT ecosystem, one cannot forget about IoT gateways. They are a piece of hardware that is capable of "translating" and facilitating the essential connection between devices or between devices and the network and work as a kind of relay for the two. Which brings us to the next element of our puzzle…



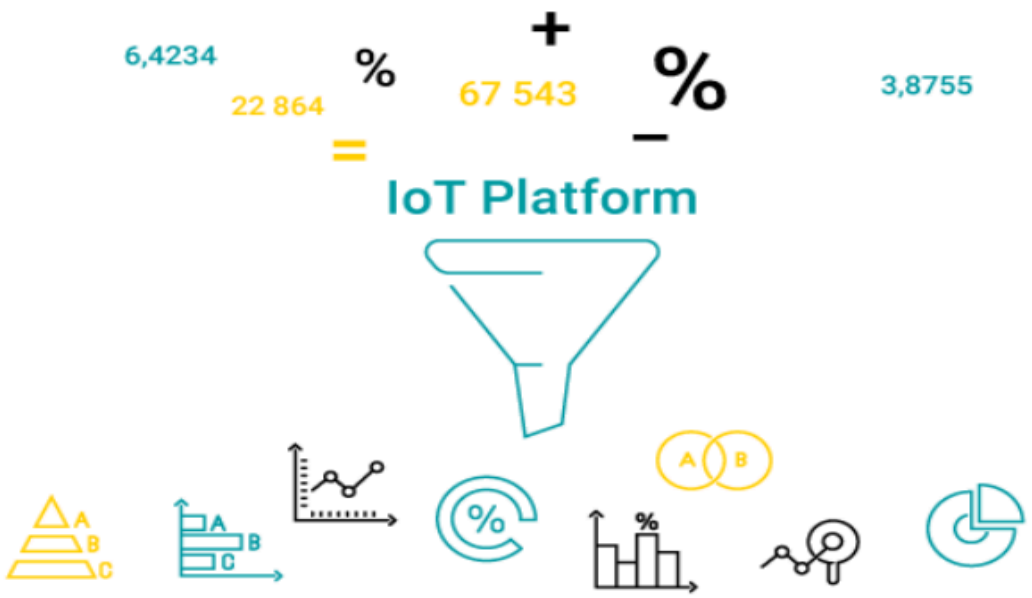
Networks

Based on what you read before, you may think: “Well, if an automatic door senses my presence and opens itself, is that IoT?” Obviously, it is not, because while that door has sensors and actuators, it is not connected to much else. And, as the name suggests, the Internet of Things requires both things and the Internet (although there are cases of data delivery without the use of the Internet Protocol). Arguably, the real power of this concept lies in the connectivity.

Again, based on your deployment needs, there are plenty of different IoT connectivity options, starting with the “classics,” such as WiFi or Bluetooth, to more specialized and fieldoriented technologies, such as Low-Power Wide Area Networks (LPWAN). They all differ in range and speed of data transfer, making them more or less appropriate for particular deployments. Consider, for example, smart cars that require both high data speed and long range and juxtapose them with the smart farms we’ve mentioned that don’t necessarily need either.

IoT platform

Whether they are in the cloud or not, IoT platforms are always the binder for any IoT ecosystem. They are the quiet administrators that take care of device lifecycle management, so that you don't have to worry about them. They are also the hub that collects and aggregates the data, allowing you to make sense of it. With the variety of platforms offered on the market and the breadth of claims their providers make, the choice of the "ideal" IoT platform for a deployment is arguably the most significant, yet also the most difficult to make. It shouldn't be taken lightly, as it determines whether the IoT ecosystem will thrive or wither into oblivion.



The right IoT device management platform should be versatile and adaptable, as the IoT world is very fragmented and constantly shifting and you don't want the core element of your ecosystem to become the stumbling block of your deployment. It should also be scalable, so that your ecosystem can grow naturally, and secure, so it can do so without any threats.

Agents

Agents are all the people whose actions affect the IoT ecosystem. These may be the engineers who devise IoT deployments and design the platforms, it can also be the

platform operators. But probably, most importantly, it's the stakeholders, who ultimately reap the results. After all, IoT deployments aren't just art for art's sake. These complex ecosystems are put in place for a reason: to drive efficiency and improve the quality of life. And it is the agents who decide on how to use the devices, networks and platforms to achieve these results. This is where technology and business converge, because it's business goals that very much shape the IoT ecosystem

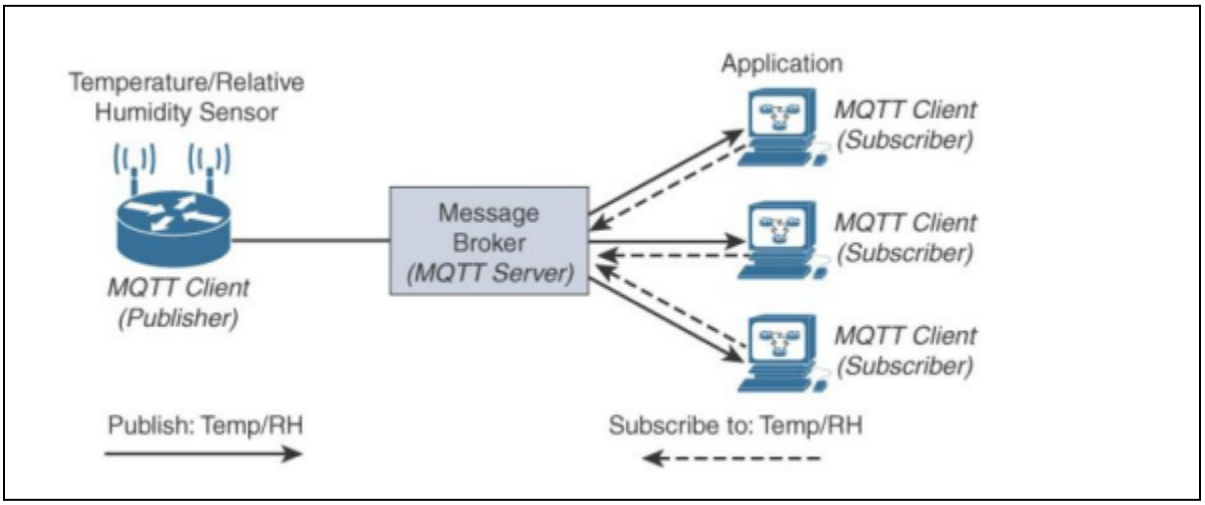
People are an essential part of this equation. Ecosystems are created by us, managed by us and, ultimately, it is our responsibility to realize their full potential. It is the devices that collect the data, but it is the people that make sense of it and put it to use. Similarly with networks and platforms, which are a necessary component of the ecosystem, but wouldn't be of much value if it weren't for the people who create and perfect them to fit their needs.

As was said, an IoT ecosystem is a very complex concept that eludes easy classification, as its characteristics vary from deployment to deployment. Much like our world, the IoT world comprises numerous different ecosystems that evolve and adapt. What they have in common is the idea and the people that make them happen: device manufacturers, service providers, application developers, and enterprises. Yet in this ever changing landscape there still remains a lot of variety – the technology, represented by the devices, networks and platforms, always gets better. This is particularly worth remembering, because the one mistake the inhabitants of an IoT ecosystem should never make, is to take it for granted. There is nothing more toxic for

that landscape than stagnation and lock-in, so you should always be on the lookout for newer, better technologies that will help you flourish.

Message Queuing Telemetry Transport (MQTT)

At the end of the 1990s, engineers from IBM and Arcom (acquired in 2006 by Eurotech) were looking for a reliable, lightweight, and cost-effective protocol to monitor and control a large number of sensors and their data from a central server location, as typically used by the oil and gas industries. Their research resulted in the development and implementation of the Message Queuing Telemetry Transport (MQTT) protocol that is now standardized by the Organization for the Advancement of Structured Information Standards (OASIS). The selection of a client/server and publish/subscribe framework based on the TCP/IP architecture, as shown:



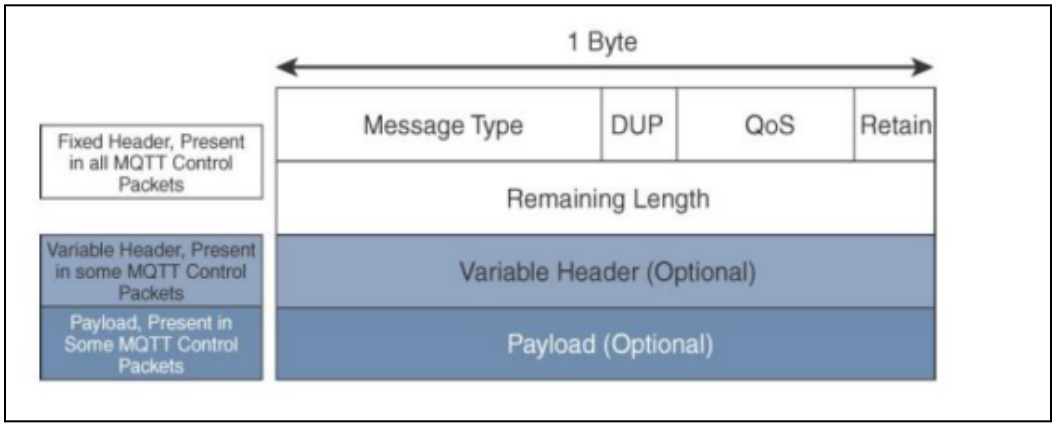
An MQTT client can act as a publisher to send data (or resource information) to an MQTT server acting as an MQTT message broker. In the example illustrated in Figure, the MQTT client on the left side is a temperature (Temp) and relative humidity (RH) sensor that publishes its Temp/RH data. The MQTT server (or message broker) accepts the network connection along with application messages, such as Temp/RH data, from the publishers. It also handles the subscription and unsubscription process and pushes the application data to MQTT clients acting as subscribers.

The application on the right side of Figure 2.22 is an MQTT client that is a subscriber to the Temp/RH data being generated by the publisher or sensor on the left. This model,

where subscribers express a desire to receive information from publishers, is well known. A great example is the collaboration and social networking application Twitter.

With MQTT, clients can subscribe to all data (using a wildcard character) or specific data from the information tree of a publisher. In addition, the presence of a message broker in MQTT decouples the data transmission between clients acting as publishers and subscribers. In fact, publishers and subscribers do not even know (or need to know) about each other. A benefit of having this decoupling is that the MQTT message broker ensures that information can be buffered and cached in case of network failures. This also means that publishers and subscribers do not have to be online at the same time. MQTT control packets run over a TCP transport using port 1883. TCP ensures an ordered, lossless stream of bytes between the MQTT client and the MQTT server. Optionally, MQTT can be secured using TLS on port 8883, and WebSocket (defined in RFC 6455) can also be used.

MQTT is a lightweight protocol because each control packet consists of a 2-byte fixed header with optional variable header fields and optional payload. You should note that a control packet can contain a payload up to 256 MB. Figure provides an overview of the MQTT message format.



Compared to the CoAP message format, MQTT contains a smaller header of 2 bytes compared to 4 bytes for CoAP. The first MQTT field in the header is Message Type, which identifies the kind of MQTT packet within a message. Fourteen different types of control packets are specified in MQTT version 3.1.1. Each of them has a unique value

that is coded into the Message Type field. Note that values 0 and 15 are reserved. MQTT message types are summarized in Table.

Message Type	Value	Flow	Description
CONNECT	1	Client to server	Request to connect
CONNACK	2	Server to client	Connect acknowledgement
PUBLISH	3	Client to server Server to client	Publish message
PUBACK	4	Client to server Server to client	Publish acknowledgement
PUBREC	5	Client to server Server to client	Publish received
PUBREL	6	Client to server Server to client	Publish release
PUBCOMP	7	Client to server Server to client	Publish complete
SUBSCRIBE	8	Client to server	Subscribe request
SUBACK	9	Server to client	Subscribe acknowledgement
UNSUBSCRIBE	10	Client to server	Unsubscribe request
UNSUBACK	11	Server to client	Unsubscribe acknowledgement
PINGREQ	12	Client to server	Ping request
PINGRESP	13	Server to client	Ping response
DISCONNECT	14	Client to server	Client disconnecting

The next field in the MQTT header is DUP (Duplication Flag). This flag, when set, allows the client to notate that the packet has been sent previously, but an acknowledgement was not received. The QoS header field allows for the selection of three different QoS levels. The next field is the Retain flag. Only found in a PUBLISH message, the Retain flag notifies the server to hold onto the message data. This allows new subscribers to instantly receive the last known value without having to wait for the next update from the publisher.

The last mandatory field in the MQTT message header is Remaining Length. This field specifies the number of bytes in the MQTT packet following this field.

MQTT sessions between each client and server consist of four phases: session establishment, authentication, data exchange, and session termination. Each client connecting to a server has a unique client ID, which allows the identification of the MQTT session between both parties. When the server is delivering an application message to more than one client, each client is treated independently.

Subscriptions to resources generate SUBSCRIBE/SUBACK control packets, while unsubscription is performed through the exchange of UNSUBSCRIBE/UNSUBACK control packets. Graceful termination of a connection is done through a DISCONNECT control packet, which also offers the capability for a client to reconnect by re-sending its client ID to resume the operations.

A message broker uses a topic string or topic name to filter messages for its subscribers. When subscribing to a resource, the subscriber indicates the one or more topic levels that are used to structure the topic name. The forward slash (/) in an MQTT topic name is used to separate each level within the topic tree and provide a hierarchical structure to the topic names.

Comparison of CoAP and MQTT

Factor	CoAP	MQTT
Main transport protocol	UDP	TCP
Typical messaging	Request/response	Publish/subscribe
Effectiveness in LLNs	Excellent	Low/fair (Implementations pairing UDP with MQTT are better for LLNs.)
Security	DTLS	SSL/TLS
Communication model	One-to-one	many-to-many
Strengths	Lightweight and fast, with low overhead, and suitable for constrained networks; uses a RESTful model that is easy to code to; easy to parse and process for constrained devices; support for multicasting; asynchronous and synchronous messages	TCP and multiple QoS options provide robust communications; simple management and scalability using a broker architecture
Weaknesses	Not as reliable as TCP-based MQTT, so the application must ensure reliability.	Higher overhead for constrained devices and networks; TCP connections can drain low-power devices; no multicasting support

Iot Platform

An IoT platform is a multi-layer technology that enables straightforward provisioning, management, and automation of connected devices within the Internet of Things universe. It basically connects your hardware, however diverse, to the cloud by using flexible connectivity options, enterprise-grade security mechanisms, and broad data processing powers. For developers, an IoT platform provides a set of ready-to-use features that greatly speed up development of applications for connected devices as well as take care of scalability and cross-device compatibility.

Thus, an IoT platform can be wearing different hats depending on how you look at it. It is commonly referred to as middleware when we talk about how it connects remote devices to user applications (or other devices) and manages all the interactions between the hardware and the application layers. It is also known as a cloud

enablement platform or IoT enablement platform to pinpoint its major business value, that is empowering standard devices with cloud-based applications and services. Finally, under the name of the IoT application enablement platform, it shifts the focus to being a key tool for IoT developers.

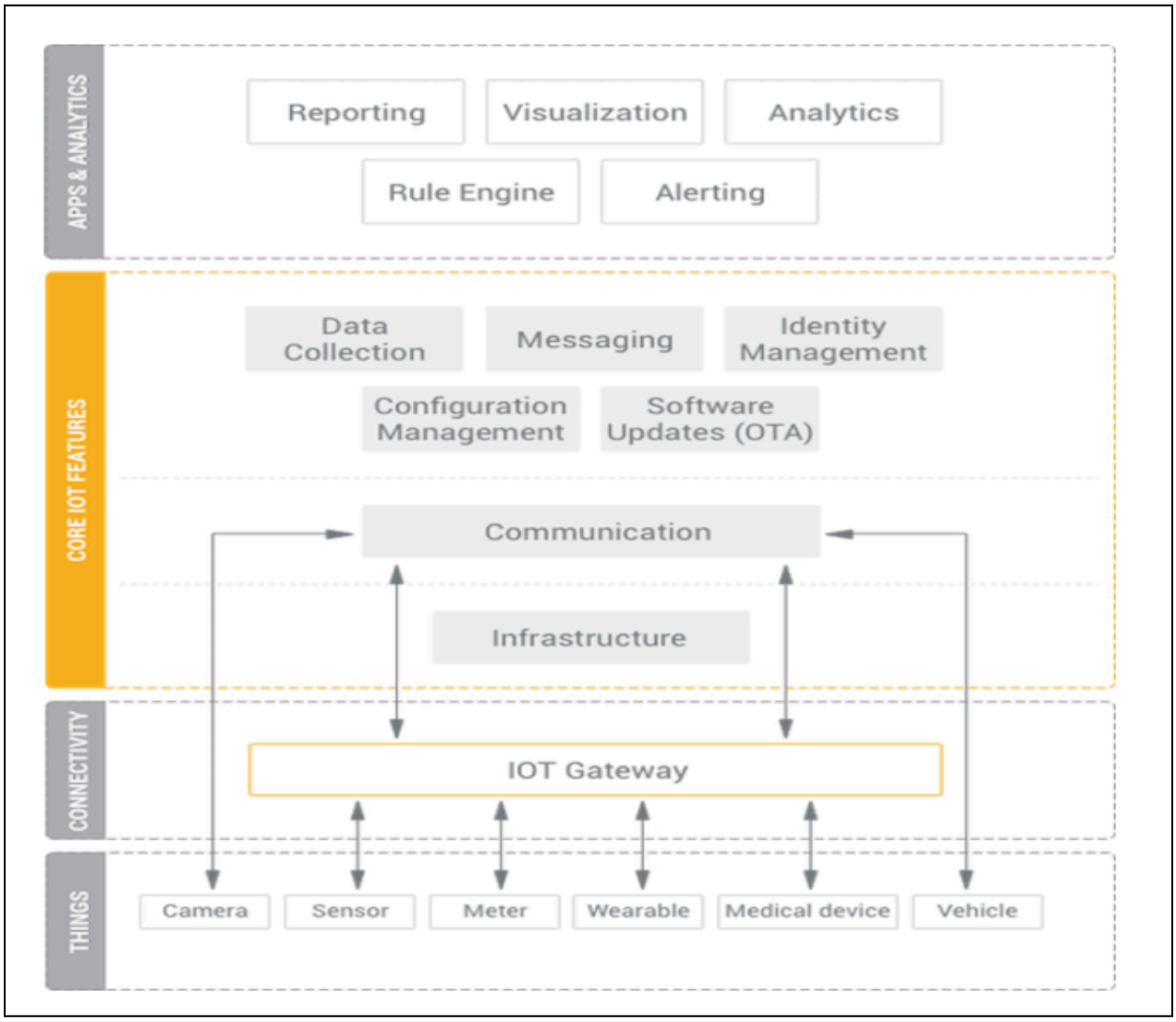
IoT platform as the middleware

IoT platforms originated in the form of IoT middleware, which purpose was to function as a mediator between the hardware and application layers. Its primary tasks included data collection from the devices over different protocols and network topologies, remote device configuration and control, device management, and over-the-air firmware updates.

To be used in real-life heterogeneous IoT ecosystems, IoT middleware is expected to support integration with almost any connected device and blend in with third-party applications used by the device. This independence from underlying hardware and overhanging software allows a single IoT platform to manage any kind of connected device in the same straightforward way.

Modern IoT platforms go further and introduce a variety of valuable features into the hardware and application layers as well. They provide components for frontend and

analytics, on-device data processing, and cloud-based deployment. Some of them can handle end-to-end IoT solution implementation from the ground up.



IoT platform technology stack

In the four typical layers of the IoT stack, which are things, connectivity, core IoT features, and applications & analytics, a top-of-the-range IoT platform should provide you with the majority of IoT functionality needed for developing your connected devices and smart things.

Your devices connect to the platform, which sits in the cloud or in your on-premises data center, either directly or by using an IoT gateway. A gateway comes useful whenever your endpoints aren't capable of direct cloud communication or, for example, you need some computing power on edge. You can also use an IoT gateway to convert protocols, for example, when your endpoints are in LoRaWan network but you need them to communicate with the cloud over MQTT.

An IoT platform itself can be decomposed into several layers. At the bottom there is the infrastructure level, which is something that enables the functioning of the platform. You can find here components for container management, internal platform messaging, orchestration of IoT solution clusters, and others.

The communication layer enables messaging for the devices; in other words, this is where devices connect to the cloud to perform different operations. The following layer represents core IoT features provided by the platform. Among the essential ones are data collection, device management, configuration management, messaging, and OTA software updates.

Sitting on top of core IoT features, there is another layer, which is less related to data exchange between devices but rather to processing of this data in the platform. There is reporting, which allows you to generate custom reports. There is visualization for data representation in user applications. Then, there are a rule engine, analytics, and alerting for notifying you about any anomalies detected in your IoT solution.

Importantly, the best IoT platforms allow you to add your own industry-specific components and third-party applications. Without such flexibility adapting an IoT platform for a particular business scenario could bear significant extra cost and delay the solution delivery indefinitely.

Advanced IoT platforms

There are some other important criteria that differentiate IoT platforms between each other, such as scalability, customizability, ease of use, code control, integration with 3rd

party software, deployment options, and the data security level.

- **Scalable (cloud native)**

  Advanced IoT platforms ensure elastic scalability across any number of endpoints that the client may require. This capability is taken for granted for public cloud deployments but it should be specifically put to the test in case of an on-premises deployment, including the platform's load balancing capabilities for maximized performance of the server cluster.

- **Customizable**

  A crucial factor for the speed of delivery. It closely relates to flexibility of integration APIs, louse coupling of the platform's components, and source code transparency. For small-scale, undemanding IoT solutions good APIs may be enough to fly, while feature-rich, rapidly evolving IoT ecosystems usually require developers to have a greater degree of control over the entire system, its source code, integration interfaces, deployment options, data schemas, connectivity and security mechanisms, etc.

- **Secure**

  Data security involves encryption, comprehensive identity management, and flexible deployment. End-to-end data flow encryption, including data at rest, device authentication, user access rights management, and private cloud infrastructure for sensitive data – this is the basics of how to avoid potentially compromising breaches in your IoT solution.

Cutting across these aspects, there are two different paradigms of IoT solution cluster deployment offered by IoT platform providers: a public cloud IoT PaaS and a self-hosted private IoT cloud.

IoT cloud enablement

An IoT cloud is a pinnacle of the IoT platforms evolution. Sometimes these two terms

are used interchangeably, in which case the system at hand is typically an IoT platform-as-aservice (PaaS). This type of solution allows you to rent cloud infrastructure and an IoT platform all from a single technology provider. Also, there might be ready-to-use IoT solutions (IoT cloud services) offered by the provider, built and hosted on its infrastructure. However, one important capability of a modern IoT platform consists in a private IoT cloud enablement. As opposed to public PaaS solutions located at a provider's cloud, a private IoT cloud can be hosted on any cloud infrastructure, including a private data center. This type of deployment offers much greater control over the new features development, customization, and third-party integrations. It is also advocated for stringent data security and performance requirements.

Arduino

Arduino

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

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

Board Types

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

The differences are based on the number of inputs and outputs (the number of sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, form factor etc. Some boards are designed to be embedded and have no programming

interface (hardware), which you would need to buy separately. Some can run directly from a 3.7V battery, others need at least 5V.

Here is a list of different Arduino boards available.

Arduino boards based on ATMEGA328 microcontroller:

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programmin Interface
Arduino Uno R3	5V	16MHz	14	6	6	1	USB vi ATMega16U
Arduino Uno R3 SMD	5V	16MHz	14	6	6	1	USB vi ATMega16U
Red Board	5V	16MHz	14	6	6	1	USB via FTDI
Arduino Pro 3.3v/8 MHz	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
Arduino Pro 5V/16MHz	5V	16MHz	14	6	6	1	FTDI-Compatible Header
Arduino mini 05	5V	16MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 3.3v/8mhz	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 5v/16mhz	5V	16MHz	14	8	6	1	FTDI-Compatible

Arduino boards based on ATMEGA32u4 microcontroller

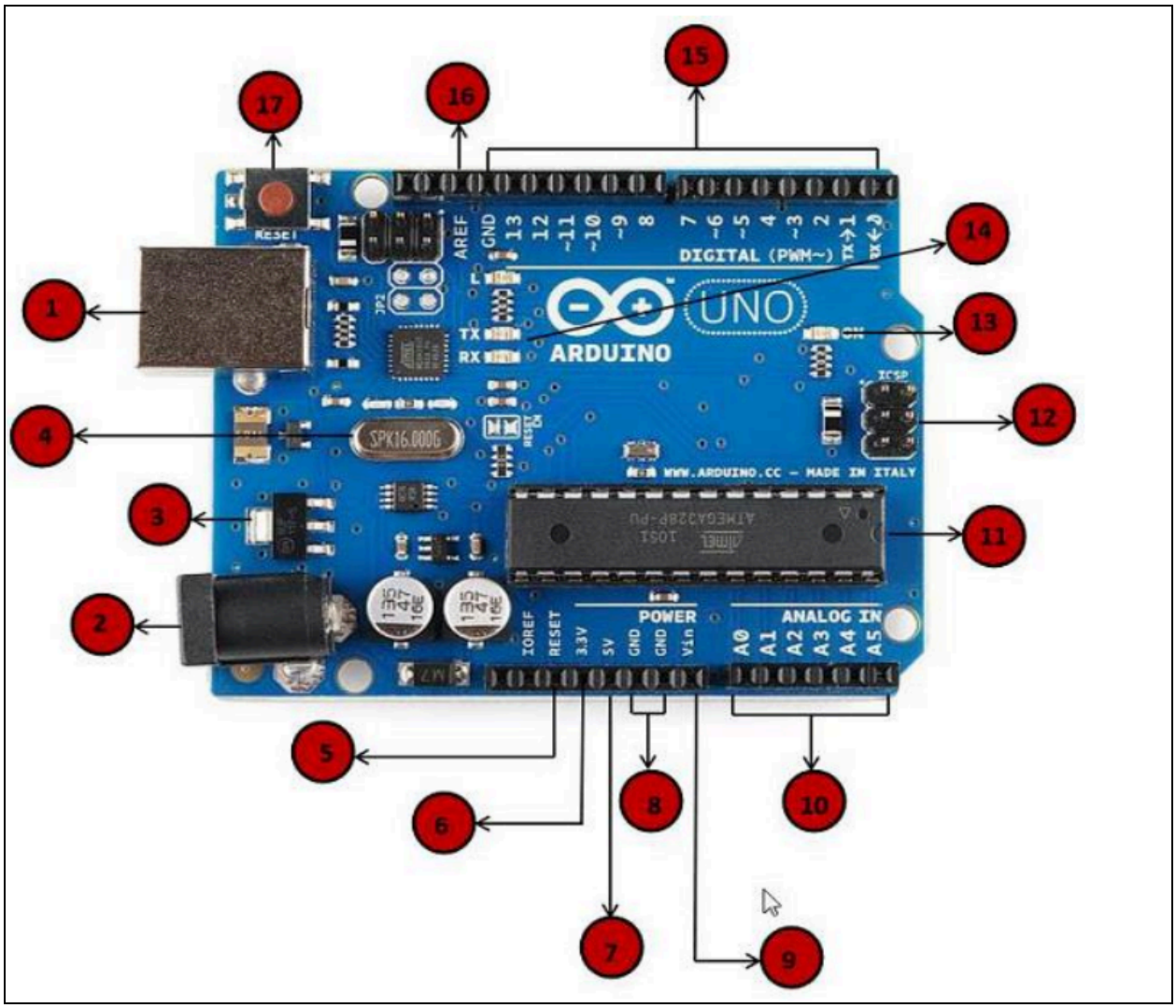
Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programmin Interface
Arduino Leonardo	5V	16MHz	20	12	7	1	Native USB
Pro micro 5V/16MHz	5V	16MHz	14	6	6	1	Native USB
Pro micro 3.3V/8MHz	5V	16MHz	14	6	6	1	Native USB
LilyPad Arduino USB	3.3V	8MHz	14	6	6	1	Native USB

Arduino boards based on ATMEGA2560 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	5V	16MHz	54	16	14	4	USB via ATMega16U2
Mega Pro 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro 5V	5V	16MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro Mini 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header

Arduino boards based on AT91SAM3X8E microcontroller

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



1	**Power USB** Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).

2	**Power (Barrel Jack)** Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).

3	**Voltage Regulator** The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

4	**Crystal Oscillator** The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.

5,17	**Arduino Reset** You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

6,7 8,9	**Pins (3.3, 5, GND, Vin)** • 3.3V (6) – Supply 3.3 output volt • 5V (7) – Supply 5 output volt • Most of the components used with Arduino board works fine with 3.3 volt and 5 volt. • GND (8)(Ground) – There are several GND pins on the Arduino, any of which can be used to ground your circuit. • Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.

10	**Analog pins** The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.
11	**Main microcontroller** Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.
12	**ICSP pin** Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.
13	**Power LED indicator** This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

14	**TX and RX LEDs** On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

15	**Digital I/O** The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled "~" can be used to generate PWM.
16	**AREF** AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

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

how to set up the Arduino IDE

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

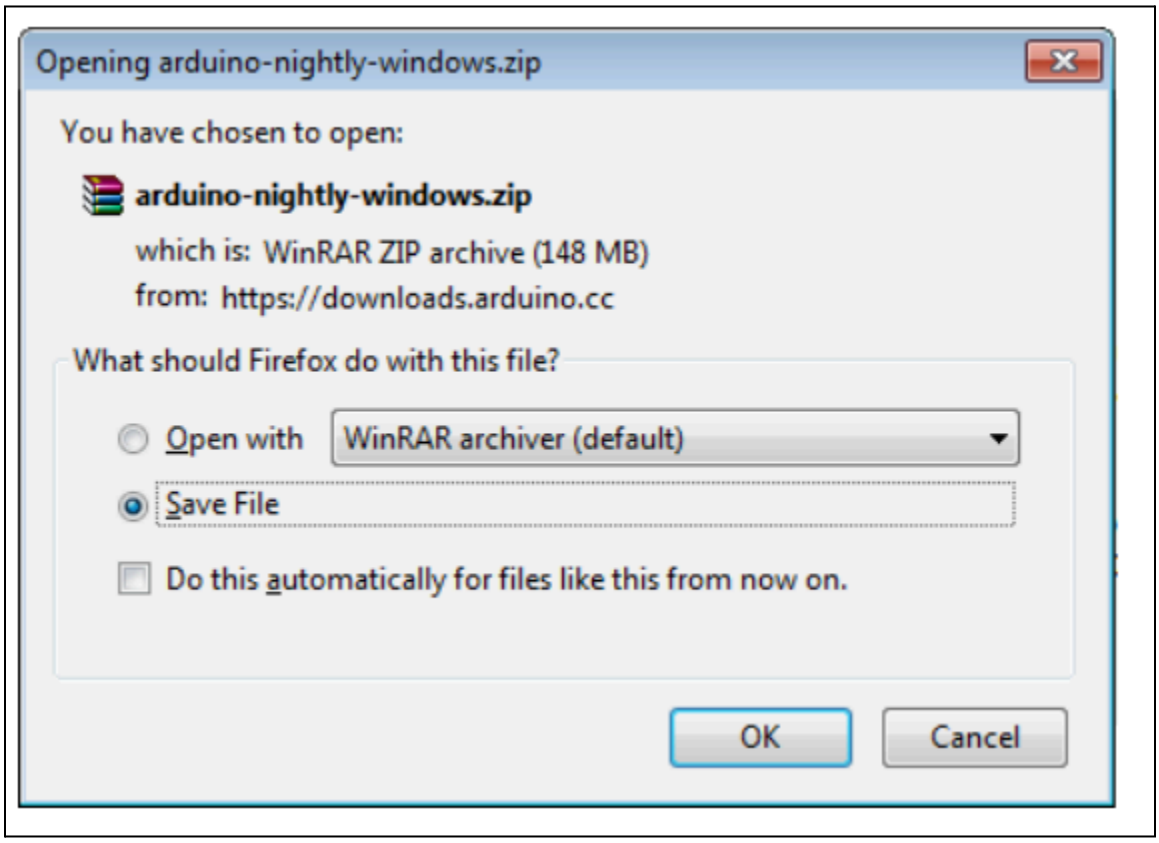
Step 1 −

First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you

would connect to a USB printer as shown in the following image.

Step 2 − Download Arduino IDE Software.

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



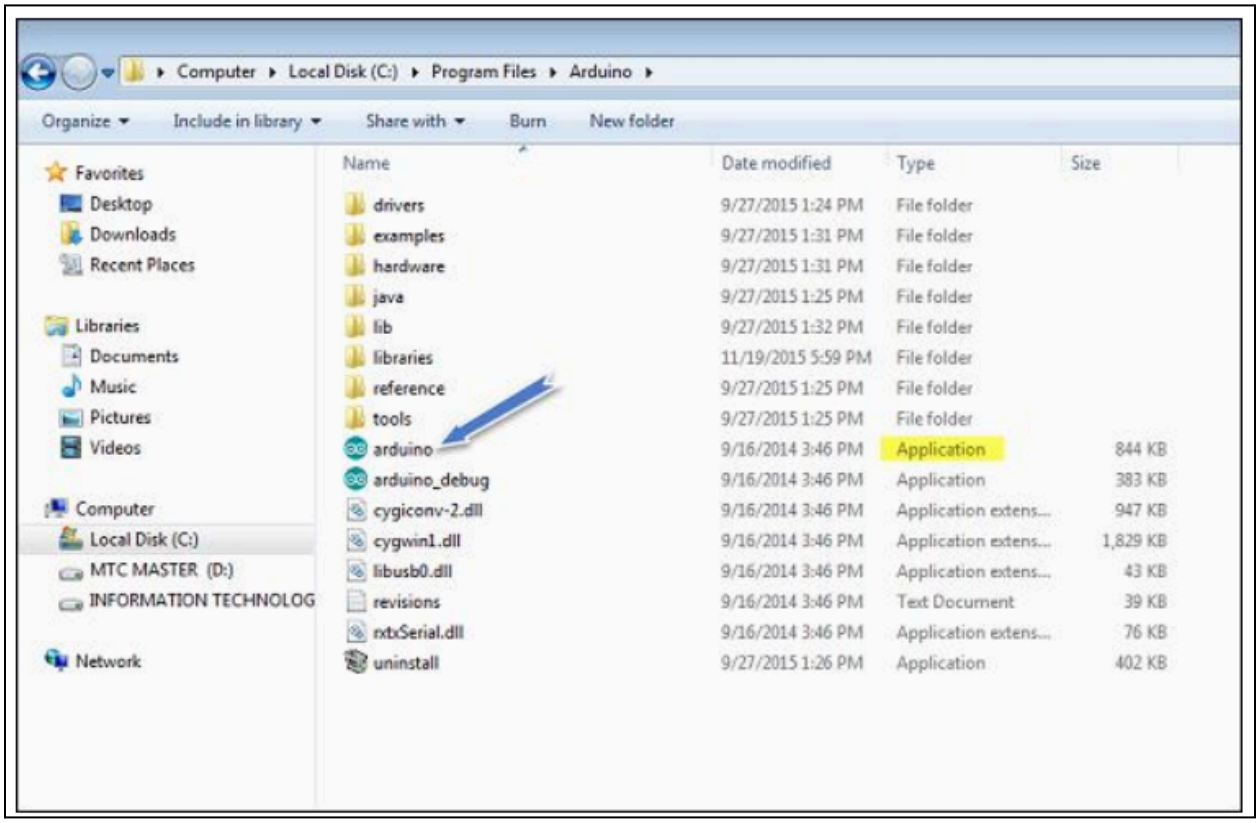
Step 3 −

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

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

Step 4 − Launch Arduino IDE.

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



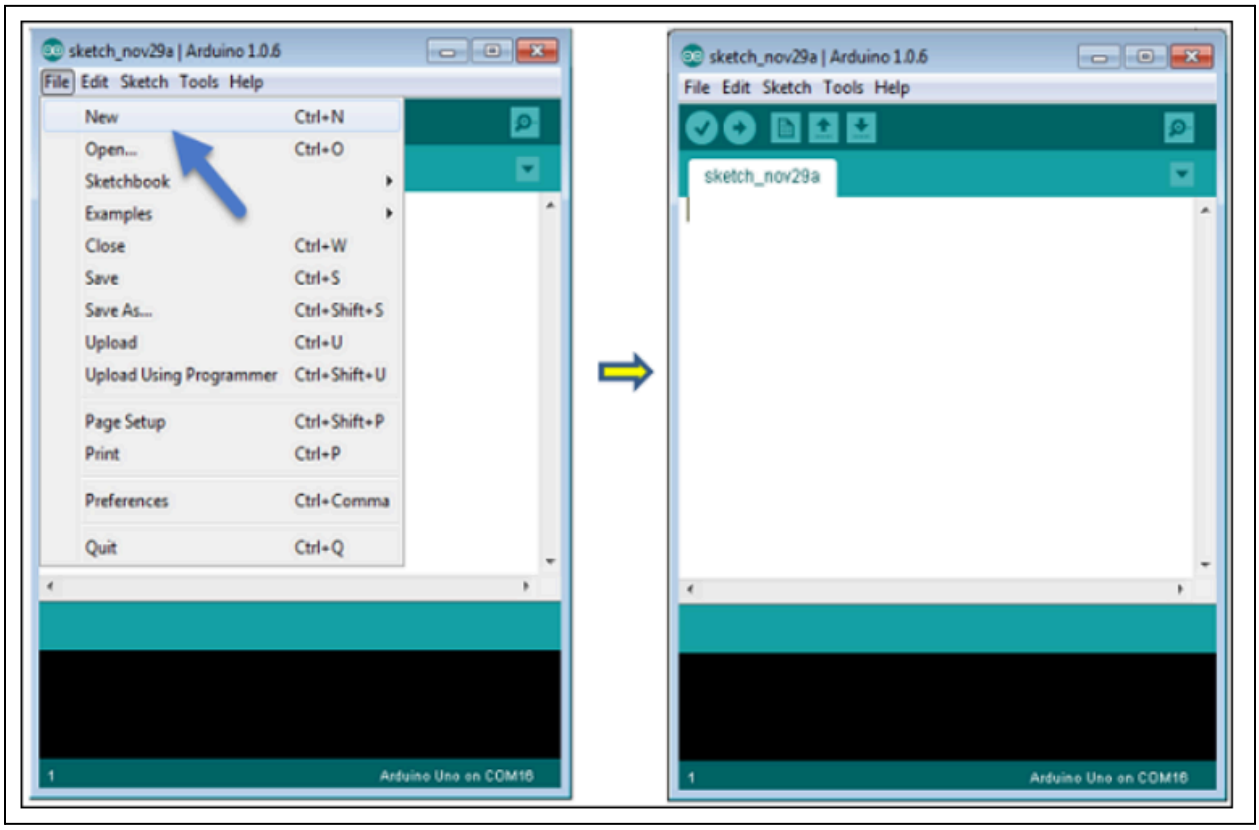
Step 5 −

Open your first project.

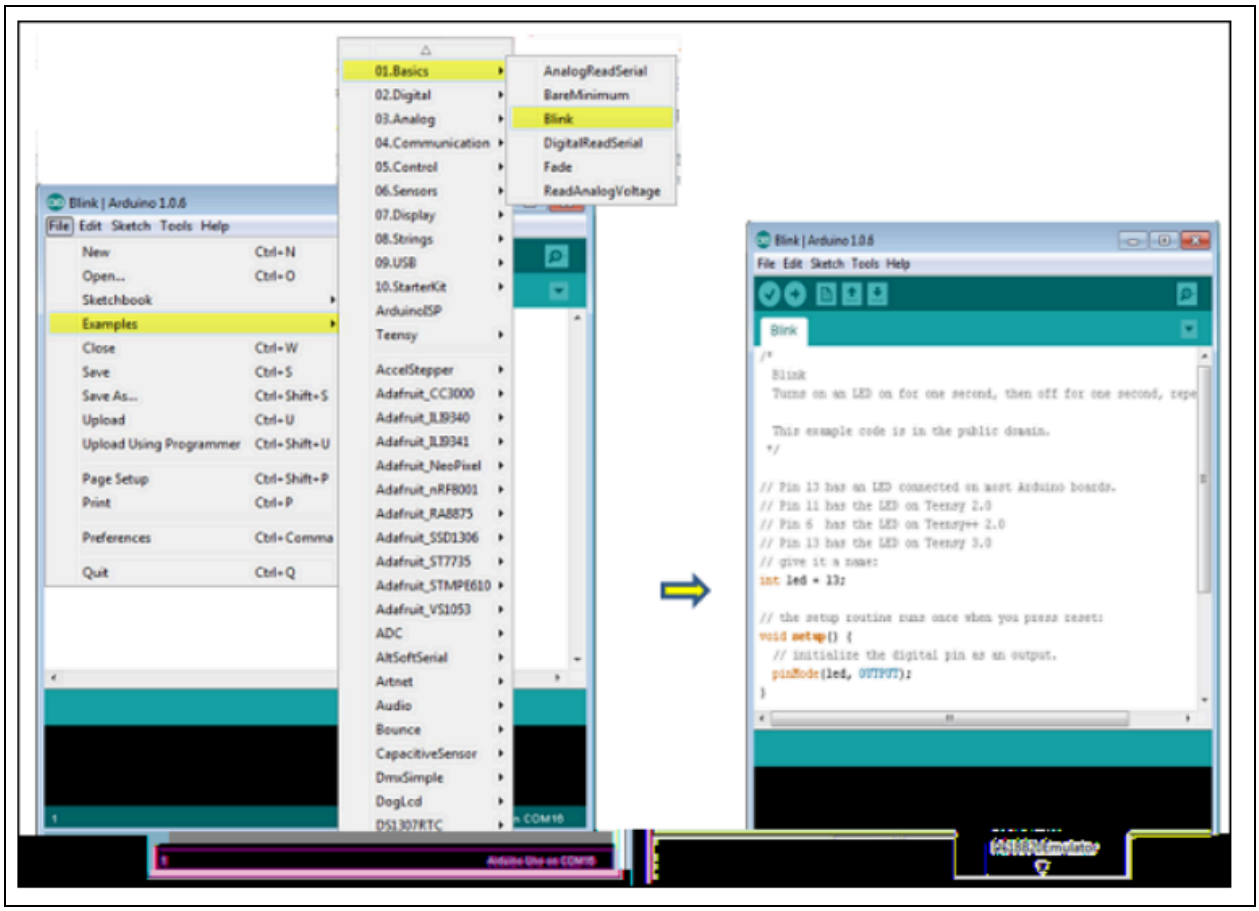
Once the software starts, you have two options −

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

To create a new project, select File → New.



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

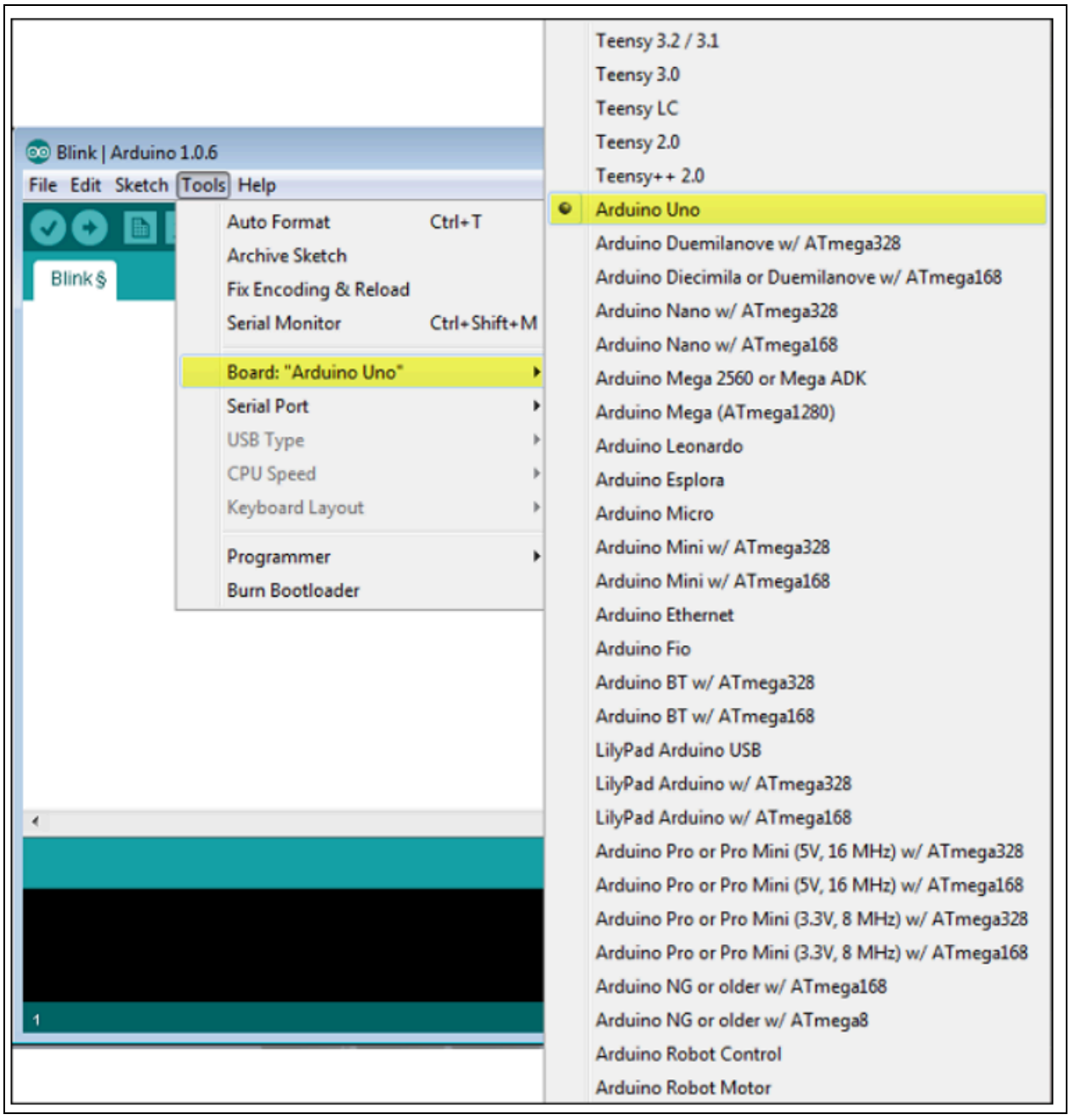


Here, we are selecting just one of the examples with the name Blink. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6 − Select your Arduino board.

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

Go to Tools → Board and select your board.

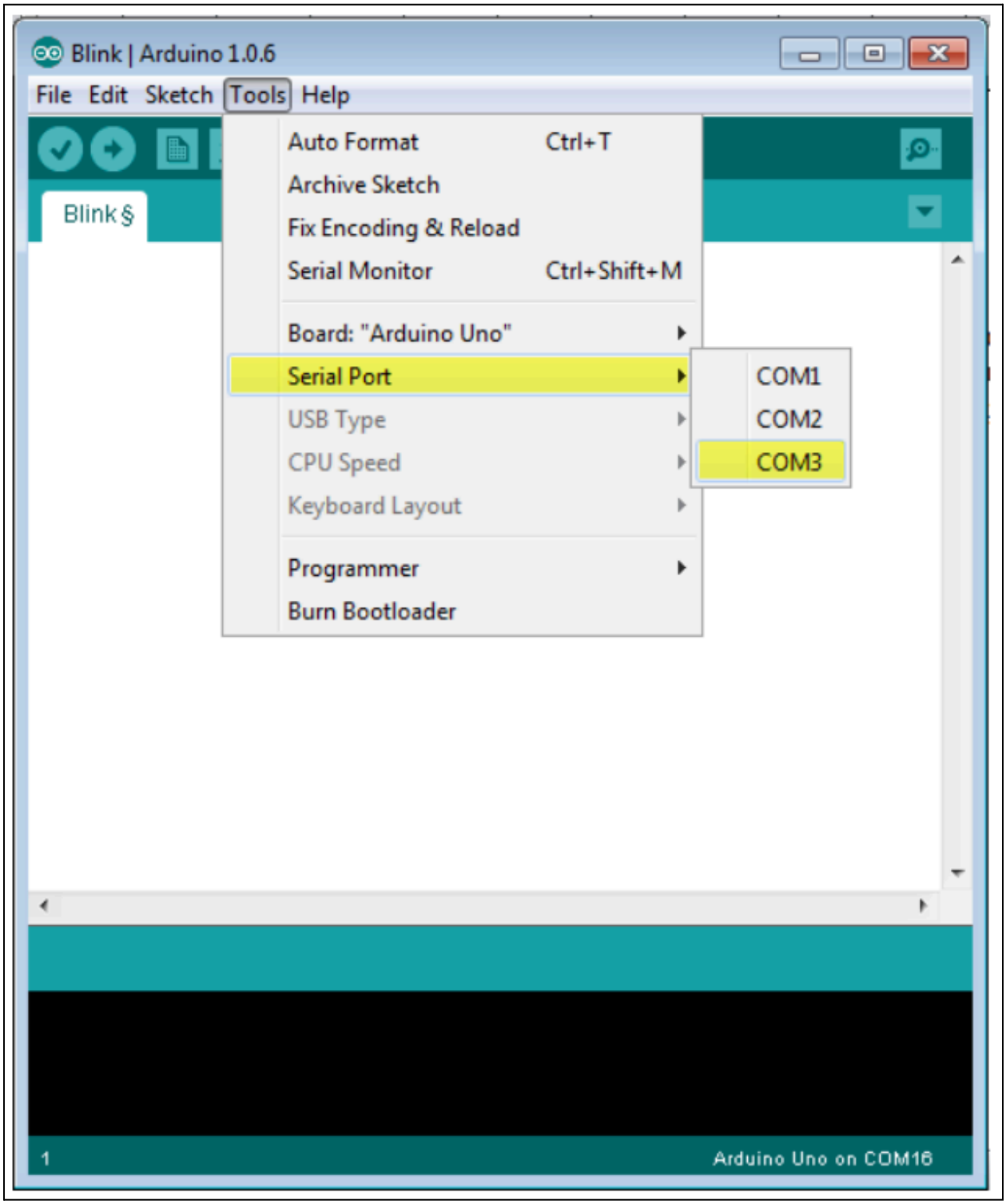


Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

Step 7 − Select your serial port.

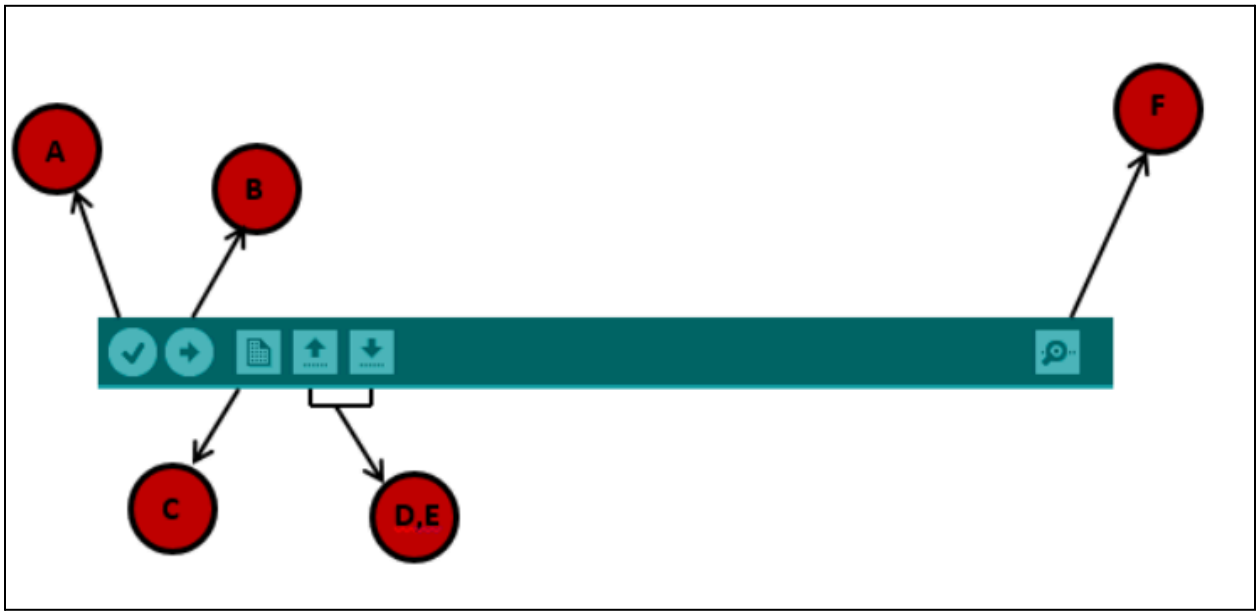
Select the serial device of the Arduino board. Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the

entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 − Upload the program to your board.

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



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

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

C − Shortcut used to create a new sketch.

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

E − Used to save your sketch.

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

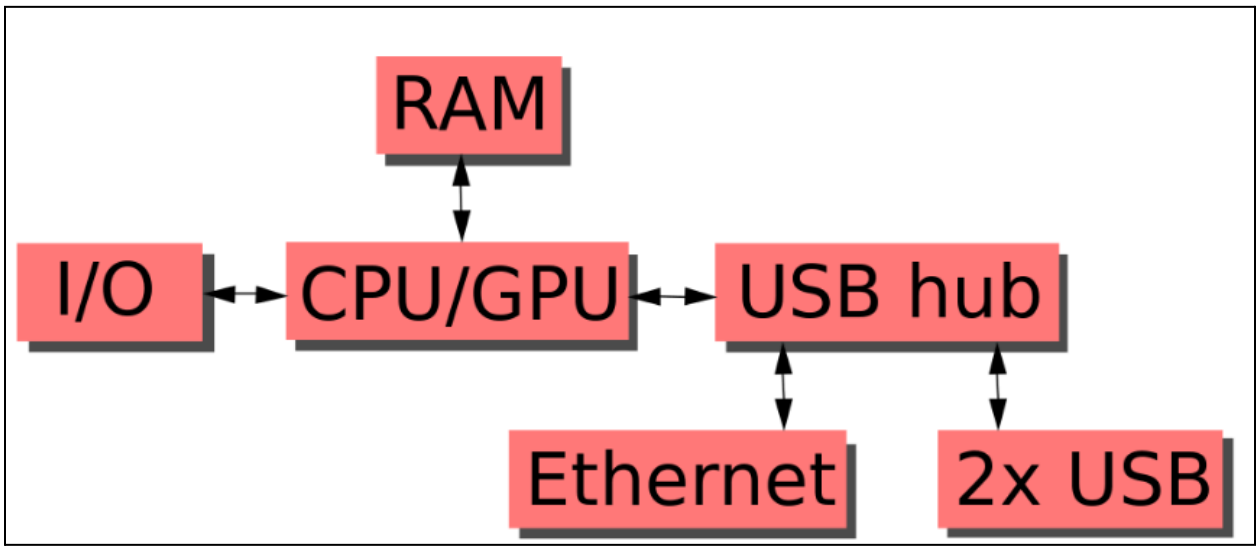
Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Raspberry Pi

Raspberry Pi (/paɪ/) is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. Early on, the Raspberry Pi project leaned towards the promotion of teaching basic computer science in schools and in developing countries. Later, the original model became far more popular than anticipated,selling outside its target market for uses such as robotics. It is now widely used in many areas, such as for weather monitoring,because of its low cost, modularity, and open design.

After the release of the second board type, the Raspberry Pi Foundation set up a new entity, named Raspberry Pi Trading, and installed Eben Upton as CEO, with the responsibility of developing technology.The Foundation was rededicated as an educational charity for promoting the teaching of basic computer science in schools and developing countries.The Raspberry Pi is one of the best-selling British computers.

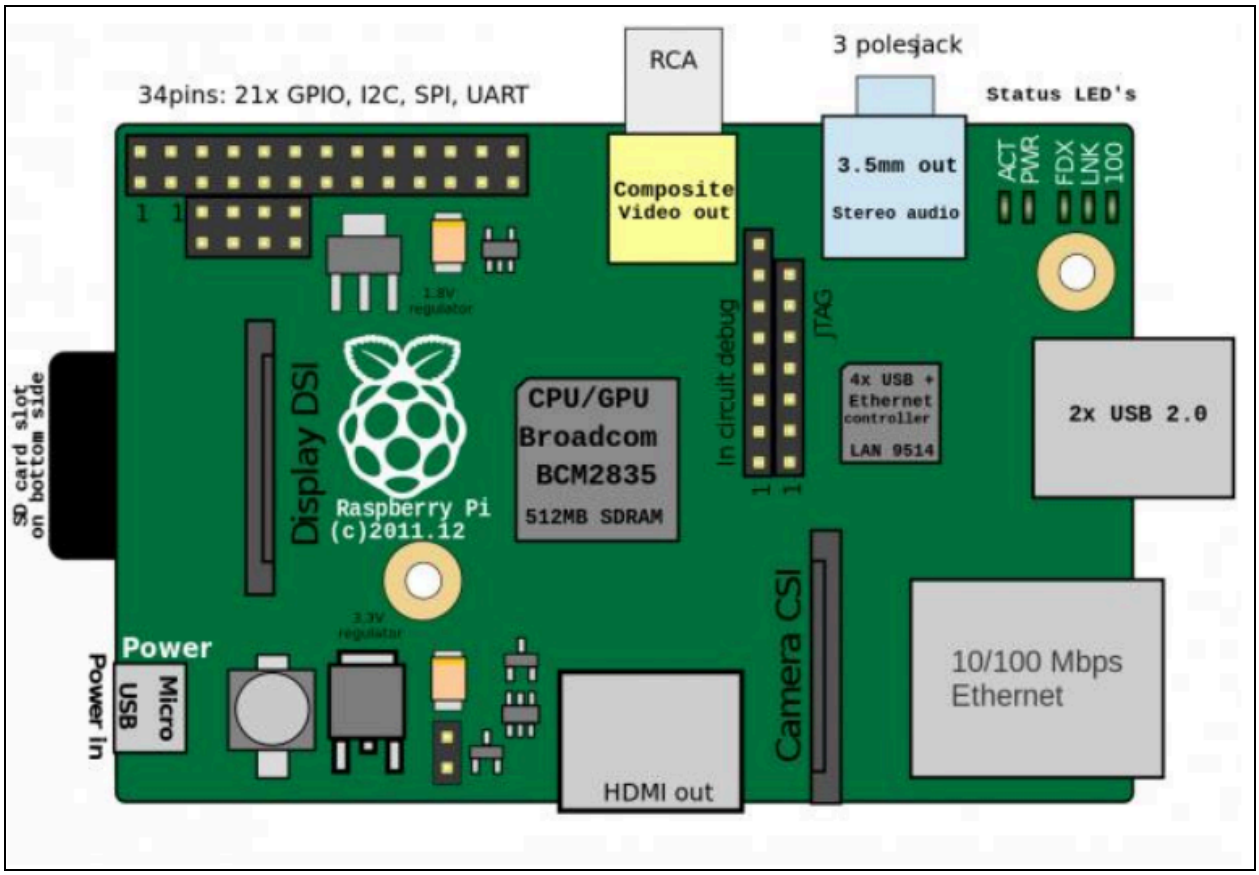
The Raspberry Pi hardware has evolved through several versions that feature variations in the type of the central processing unit, amount of memory capacity, networking support, and peripheral-device support.



This block diagram describes Model B and B+; Model A, A+, and the Pi Zero are similar,

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

Processor speed ranges from 700 MHz to 1.4 GHz for the Pi 3 Model B+ or 1.5 GHz for the Pi 4; on-board memory ranges from 256 MiB to 1 GiB random-access memory (RAM), with up to 8 GiB available on the Pi 4. Secure Digital (SD) cards in MicroSDHC form factor (SDHC on early models) are used to store the operating system and program memory. The boards have one to five USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm tip-ring-sleeve jack for audio output. Lower-level output is provided by a number of GPIO pins, which support common protocols like I²C. The B-models have an 8P8C Ethernet port and the Pi 3, Pi 4 and Pi Zero W have on-board WiFi 802.11n and Bluetooth.



Raspberry Pi is a mono-board computing platform that's as tiny as a credit card. Initially it was developed for computer science education with later on progress to wider functions.

Since the inception of Raspberry, the company sold out more than 8 million items. Raspberry Pi 3 is the latest version and it is the first 64-bit computing board that also comes with built-in Wi-Fi and Bluetooth functions. According to Raspberry Pi Foundation CEO Eben Upton, "it's been a year in the making". The Pi3 version is replaced with a quad-core 64-bit 1.2 GHz ARM Cortex A53 chip, 1GB of RAM, VideoCore IV graphics, Bluetooth 4.1 and 802.11n Wi-Fi. The developers claim the new architecture delivers an average 50% performance improvement over the Pi 2.

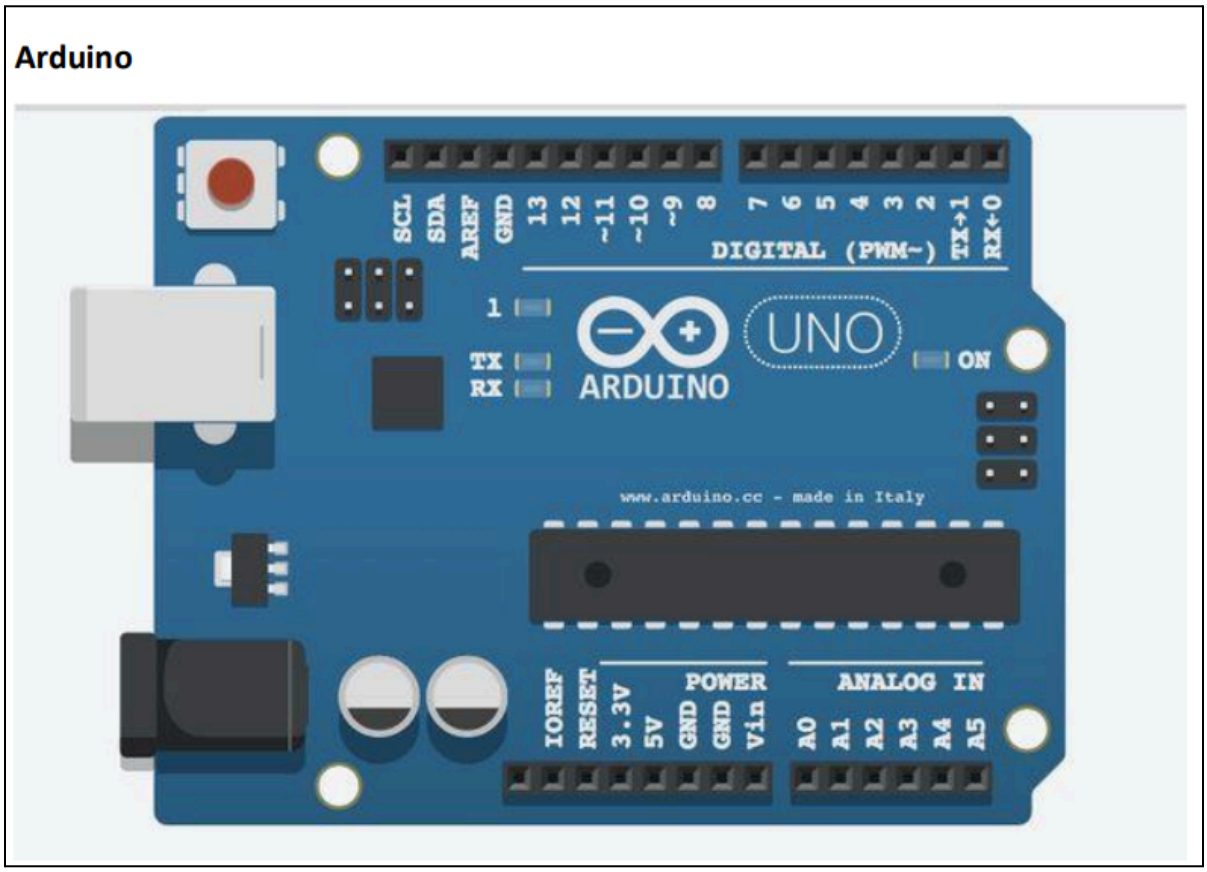
Another peculiarity of Raspberry Pi is the GPIO (General Purpose Input-Output), which is a low-level interface of self-operated control by input-output ports. Raspberry has it as

a 40- pin connector. Raspberry Pi uses Linux as its default operating system (OS). It's also fully Android compatible. Using the system on Windows OS is enabled through any virtualization system like XenDesktop. If you want to develop an application for Raspberry Pi on your computer, it is necessary to download a specific toolset comprised of ARM-compiler and some libraries complied down to ARM-target platform like glibc.

IoT Platforms Overview: Arduino, Raspberry Pi

The IoT concepts imply a creation of network of various devices interacting with each other and with their environment. Interoperability and connectivity wouldn't be possible without hardware platforms that help developers solve issues such as building autonomous interactive objects or completing common infrastructure related tasks.

Let's go through the most popular IoT platforms and see how they work and benefit IoT software developers.



The Arduino platform was created back in 2005 by the Arduino company and allows for open source prototyping and flexible software development and back-end deployment

while providing significant ease of use to developers, even those with very little experience building IoT solutions.

Arduino is sensible to literally every environment by receiving source data from different external sensors and is capable to interact with other control elements over various devices, engines and drives. Arduino has a built-in micro controller that operates on the Arduino software.

Projects based on this platform can be both standalone and collaborative, i.e. realized with use of external tools and plugins. The integrated development environment (IDE) is composed of the open source code and works equally good with Mac, Linux and Windows OS. Based on a processing programming language, the Arduino platform seems to be created for new users and for experiments. The processing language is dedicated to visualizing and building interactive apps using animation and Java Virtual Machine (JVM) platform.

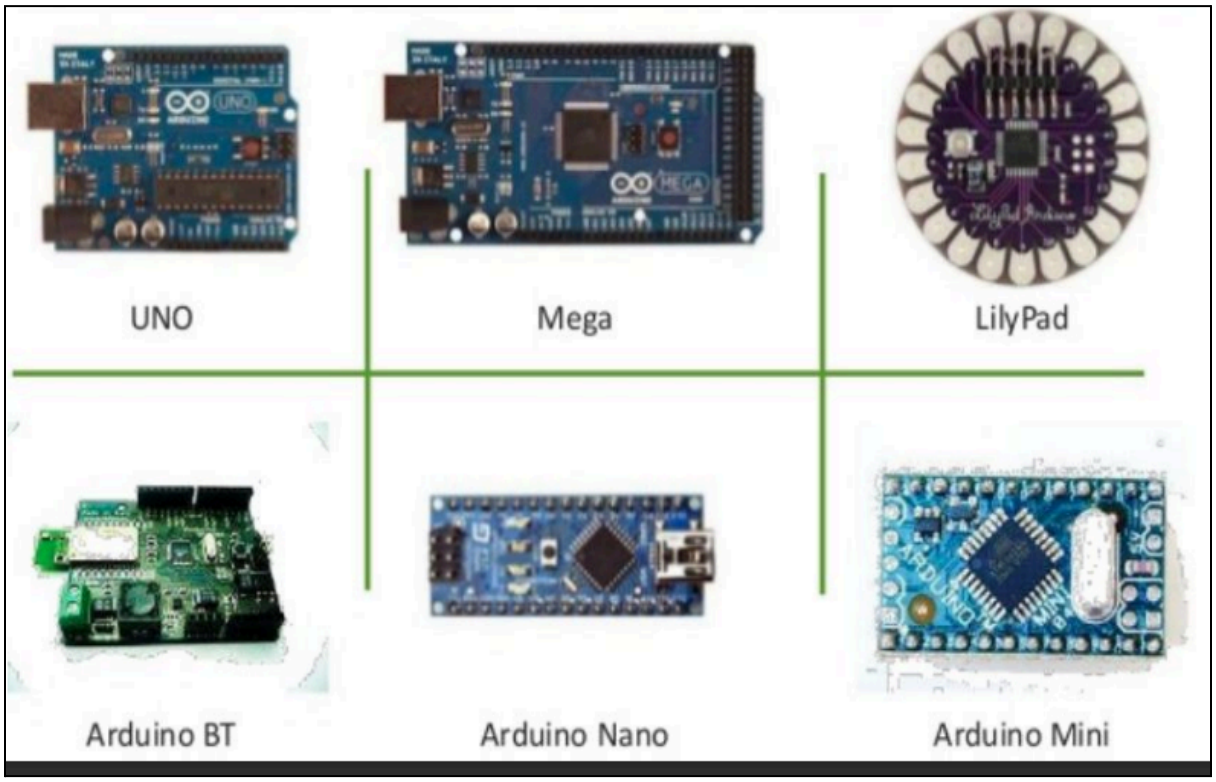
Let's note that this programming language was developed for the purpose of learning basic computer programming in a visual context. It is an absolutely free project available to every interested person. Normally, all the apps are programmed in C/C++, and are wrapped with avr-gcc (WinAVR in OS Windows).

Arduino offers analogue-to-digital input with a possibility of connecting light, temperature or sound sensor modules. Such sensors as SPI or I2C may also be used to cover up to 99% of these apps' market.

Arduino is a microcontroller (generally it is the 8-bit ATmega microcontroller), but not a mini-computer, which makes Arduino somehow limited in its features for advanced users. Arduino provides an excellent interactivity with external devices and offers a wide range of user manuals, project samples as well as a large community of users to learn from / share knowledge with.

Arduino Models

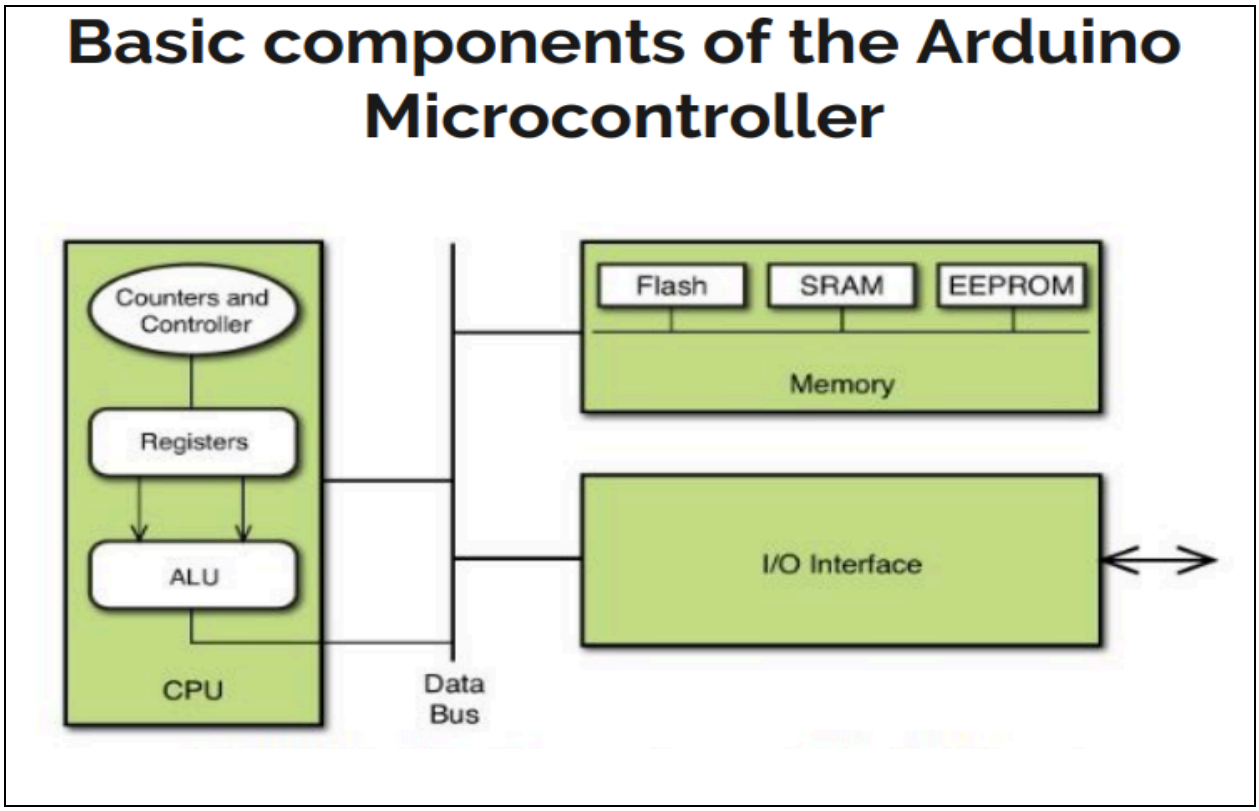
Arduino's are microcontrollers: mini-computers with their own memory, disk, and processor



Lights on an Arduino (Arduino UNO)

4 Lights on an Arduino…..

1. ON Light - shows that it is powered on
2. TX Light - shows that data is being transmitted
3. RX Light - shows that data is being received
4. L Light - an LED you are able to control in your program



How are Arduino Programs Structured

```
void setup() {
Code to run once
}

void loop(){
Code to run repeatedly
}
```

Declaring and Instantiating Variables

Declaring a Variable

dataType variableName;

Example:
int year;

Instantiating a Variable
Add equals sign

Example:
int year;
year = 2017;

Data Type	Size (Bytes)	Value Range
boolean	1	Logic true or false
char	1	–128 to +127
byte	1	0 to 255
int	2	–32,768 to 32,767
word	2	0 to 65,535
long	4	–2,147,483,648 to 2,147,483,647
float	4	–3.4028235E+38 to 3.4028235E+38
double	4	–3.4028235E+38 to 3.4028235E+38

Declaring and Instantiating Simultaneously

Example:

int year = 2017;

For Constants

Add 'const' before dataType

Example:

const float pi = 3.14;

Scope of Variables

- Variable scope determines where the variable can be used in the sketch. There are two variable scopes
    - Local Variables
        - Can only be used inside a function, and only appear inside that function block.
        - You won't see a local variable from the setup function in the loop function
    - Global Variables (Static Variables)
        - Can be used in ALL functions
        - Common to see them at the start of the sketch before the setup function

Math Operators

Standard Operators are built-in for use and can be used with variables.

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement
!	Logical NOT
&&	Logical AND
||	Logical OR
&	Bitwise AND
|	Bitwise OR
<<	Left shift
>>	Right shift

Using the Serial Monitor

The serial monitor allows you to see the output from the program

1.insert Serial.begin(baudRate); to initialize the serial port

baudRate = Speed of Connection

(higher is faster; must match workstation band)

// i want that baud default baud rate is 9600;

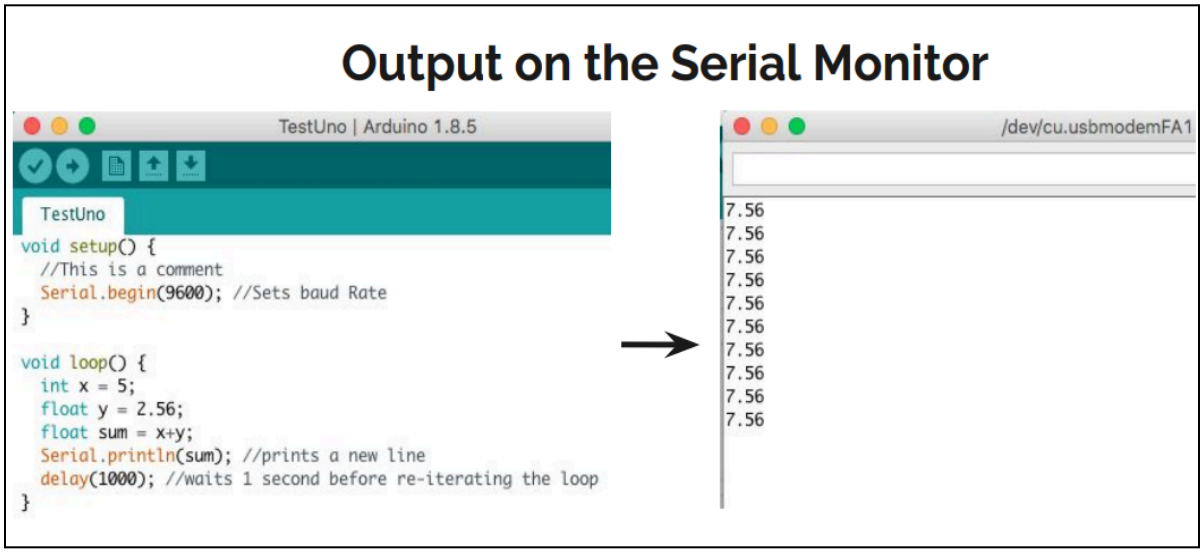
2. Printing to the serial monitor:

Serial.print(sum) // does not start a new line

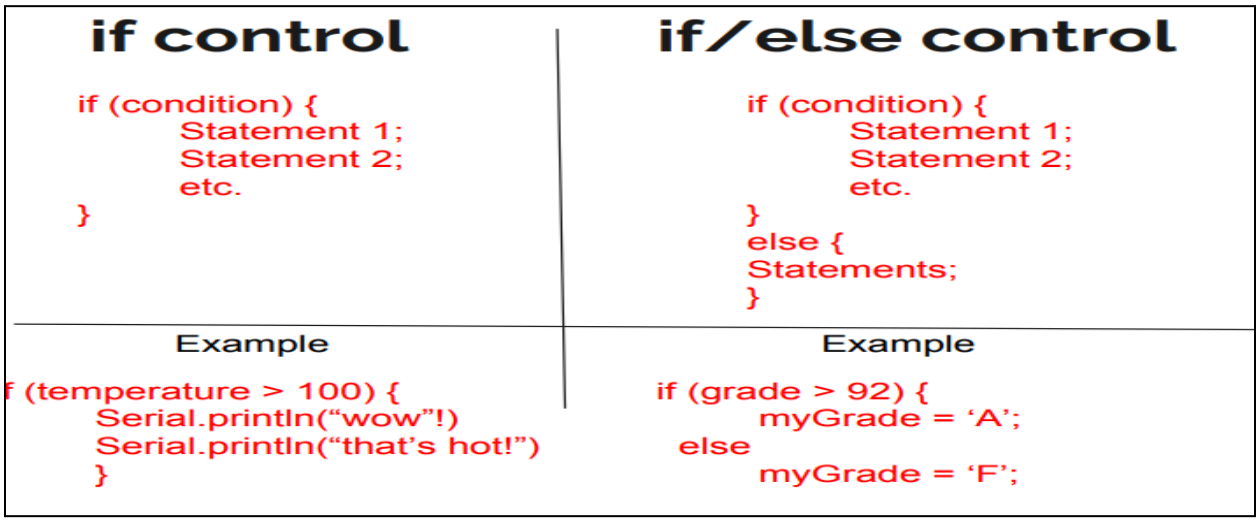
Serial.println(sum) //starts a new line

3. Working with time

- delay(x): pauses the sketch for x milliseconds
- delayMicroseconds(x): pauses the sketch for x microseconds
- micros(): Returns the number of microseconds since the arduino was reset
- millis(): returns the number of milliseconds since the Arduino was reset



Control Flow :



also,

if/else if control

```
if (condition)
        Statement;
else if (condition)
        Statement;
else if (condition)
        Statement;
else
        Statement;
```

Example

```
if (grade > 92) {
        myGrade = 'A';
  else if (grade  > 83)
        myGrade = 'B';
  else
        myGrade = 'C';
```

Operator	Description
==	Equal
!=	Not equal
<>	Not equal
>	Greater than
>=	Greater than or equal
>	Less than
<=	Less than or equal

Numeric Comparisons:

Compound Conditions

Example 1

```
If ((a == 1) || (b == 1)){
   statements;
}
```

Example 2

```
If ((a == 1) && (b == 2)) {
    statements;
}
```

Negating a condition check

```
int a == 1;
If (!(a == 1))
        Serial.println("The 'a' variable is not
equal to 1");
if(!(a ==2))
        Serial.println("The 'a' variable is not
equal to 2");
```

Or just use != in the condition

Using a switch statement:

Instead of doing a lot of if and else statement, it may be useful to do a switch

Format

```
switch (var) {
   case 23:
     //do something when var equals 23
     break;
   case 64:
     //do something when var equals 64
     break;
   default:
     // if nothing else matches, do the default
     // default is optional
    break;
 }
```

Loops:

for

```
for (statement1; condition; statement 2){
        Code statements
}
```

executes until condition is met

Example

```
int values[5] = {10, 20, 30, 40, 50};
for (int counter = 0; counter < 5; counter ++){
        Serial.print("one value in the array is " );
        Serial.println(values[counter]);
}
```

while

```
while (condition){
        Code statements
}
```

executes until condition is met

Example

```
int i = 0;
while (i<3) {//hearts for days
Serial.println(" i is : " i);
i++;
}
```

do while

```
do {
        Code statements
    } while (condition);
```

executes at least once, and until condition is met

Example

```
int i = 0;
Serial.println("Hi");
do {
 Serial.println("my name is");
if(i==0) Serial.println("What");
if(i==1) Serial.println("Who");
 i++;
} while (i < 2);
Serial.println("slicka chika slim shady");
```

Using Multiple Variables

You can initialize multiples variables in a for statement

Example

```
int a,b;
For (a = 0, b = 0; a < 10; a++, b++){
        Serial.print("One value is ");
        Serial.println(a);
        Serial.print(" and the other value is ");
        Serial.println(b);
}
```

Nesting Loops

You can place loops inside of another loop. The trick to using inner loop sis that you must complete the inner loop before you complete the outer loop.

Example

```
int a,b;
for (a = 0; a < 10; a++){
        for (b = 0; b < 10; b++){
        Serial.print("one value is ");
        Serial.print(a);
        Serial.print("and the other value is ");
        Serial.println(b);
        }
}
```

Controlling Loops:

Break Statement

- You can use the break statement when you need to break out of a loop before the condition would normally stop the loop

Example:

```
int i;
for (i = 0; i <= 20; i++) {
        if (i == 15)
        Break;
        Serial.print("currently on iteration:");
        Serial.println(i);
        }
  Serial.println("This is the end of the test");
}
```

Continue Statement

- You can use the continue statement to control loops. Instead of telling the Arduino to jump out of a loop, it tells the Arduino to stop processing code inside the loop, but still jumps back to the start of the loop

Example

```
int i;
for (i = 0; i <= 10; i++){
        If ((i > 5) && (i < 10))
        Continue;

Serial.print("The value of the counter at");
Serial.println(i);
}
Serial.println("this is the end of the test");
```

Arrays

An array stores multiple data values of the same data type in a block of memory, allowing you to reference the variables using the same variable name. It does it through an index value.

Format

datatype variableName[size];

Example 1:

```
int myarray[10]; //able to store 10  values
myarray[0] = 20; //stores values in index 0
myarray[1] = 30; // stores values in index 1
```

Example 2:

```
// assigns values to first 5 data locations (index 0-4)
Int myarray[10] = {20, 30, 40, 50, 100};
```

Example 3

```
int myarray[ ] = {20, 30, 40, 50 100};
```

Determining the size of an Array

- You may not remember how many data points are in your array
- You can use the handy sizeof function

```
size = sizeof(myArray) / sizeof(int);
```

STRINGS:

A string value is a series of characters put together to create a word or sentence

Method	Description
`charAt(n)`	Returns the character at the *n*th position in the string.
`compareTo(string2)`	Returns `0` if the string is equal to *`string2`*, a negative number if the string is less than *`string2`*, or a positive number if the string is greater than *`string2`*.
`concat(string1, string2)`	Appends the *`string2`* value to the end of *`string1`*, and creates a new string value.
`endsWith(string2)`	Returns `true` if the string ends with the *`string2`* value.
`equals(string2)`	Returns `true` if the string is equal to *`string2`*.
`equalsIgnoreCase(string2)`	Returns `true` if the string is equal to *`string2`*, ignoring character case.
`getBytes(buf, len)`	Copies *`len`* string characters into the *`buf`* variable.
`indexOf(val, [,from])`	Returns the index location where the string *`val`* starts in the string. By default, it starts at index 0, or you can specify a starting location using the *`from`* parameter. Returns `-1` if *`val`* is not found in the string.
`lastIndexOf(val [, from])`	Returns the index location where the string *`val`* starts in a string. By default, it starts at the end of the string, working toward the front of the string, or you can specify a starting location using the *`from`* parameter. Returns `-1` if *`val`* is not found in the string.

Format

String name = “my text here”;

Example

String myName = “Jackie Chan”;

Manipulating Strings

```
String myName = “Jackie Chan”;
myName.toUpperCase();

Output: JACKIE CHAN
```

Although you can just create a char array, Strings are much easier to work with! They have many more supported functions ..

`length()`	Returns the number of characters in the string (not counting the terminating null character).
`replace(substring1, substring2)`	Returns a new string with the *substring1* value with *substring2* in the original string value.
`reserve(n)`	Reserves a space of *n* characters in memory for a string value.
`startsWith(string2)`	Returns `true` if the string starts with *string2*.
`substring(from [,to])`	Returns a substring of the original string value, starting at the *from* index location. By default, it returns the rest of the string from that location, or you can specify the *to* index value.
`toCharArray(buf, len)`	Copies *len* characters in the string to the character array variable *buf*.
`toInt()`	Returns an integer value created from the string value. The string must start with a numeric character, and the conversion stops at the first non-numeric character in the string.
`setCharAt(index, c)`	Replaces the character at *index* with the character c.
`toLowerCase()`	Converts the string value to all lowercase letters.
`toUpperCase()`	Converts the string value to all uppercase letters.
`trim()`	Removes any leading and trailing space or tab characters from the string value.

Functions :

You’ll often find yourself using the same code in multiple locations. Doing large chunks of these is a hassle. However, functions make these a lot easier. You can encapsulate your C code into a function and then use it as many times as you want.

Make sure you define your function outside of the setup and loop functions in your arduino code sketch. If you define a function inside another function, the inner function becomes a local function, and you can't use it outside the outer function .

Structure

```
datatype functionName() {
        // code statements
  }
```

To create a function that does not return any data values to the calling program, you use the ***void*** data type for the function definition

```
Void myFunction() {
Serial.println("This is my first function");
}
```

Using the function/Returning a value

Using the function:

- To use a function you defined in your sketch, just reference it by the function name you assigned, followed by parentheses

```
void setup() {
        Serial.begin(9600)
        MyFunction();
        Serial.println("Now we're back to the main program");
        }
```

Returning a Value:

To return a value from the function back to the main program,

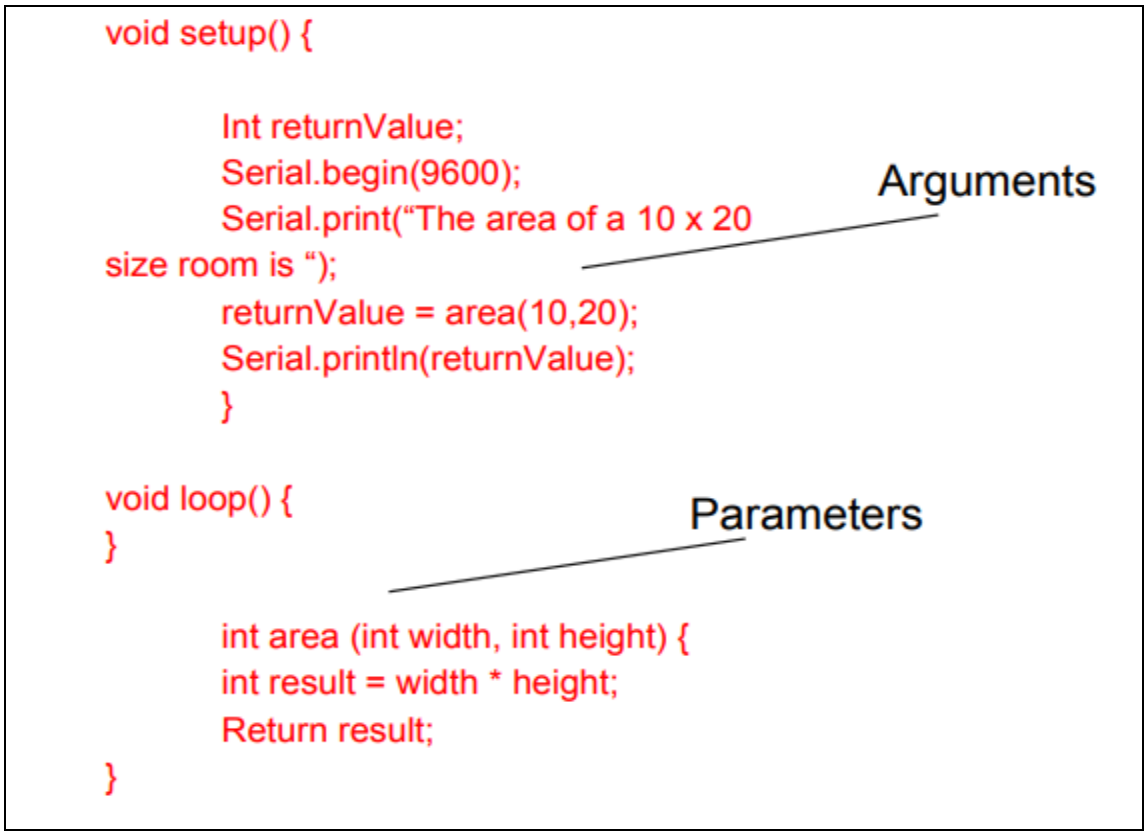
- you end the function with a return statement return value;
- The value can either a constant numeric or string value, or a variable that contains a numeric or string value.
- However, in either case, the data type of the returned value must match the data type that you use to define the function

Passing Values to Functions :

You will most likely want to pass values into function. In the main program code, you specify the values passed to a function in what are called arguments, specific inside the function parenthesis

returnValue = area(10,20);

The 10 and 20 are value arguments separated by a comma. To retrieve the arguments passed to a function, the function definition must declare what are called parameters. You do that in the main function declaration line



Handling Variables inside Functions:

One thing that causes problem for beginning sketch writers is the scope of a variable. He scope is where the variable can be referenced within the sketch. Variables defined in function can have a different scope than regular variables. That is, they can be hidden from the rest of the sketch. Functions use two types of variables:

- Global Variables
- Local Variables

Defining Global Variables:

Write them before the setup() loop.

Ex:

const float pi = 3.14;

Be careful in modifying global variables.

Declaring local variables:

Local variables are declared in the function code itself, separate from the rest of the sketch code. What's interesting is that a local variable can override a global variable (but not good practice)

Calling Functions Recursively

Recursion is when the function calls itself to reach an answer. Usually a recursion function has a base value that it eventually iterates down to. Many advanced algorithms use recursion to reduce a complex equation down one level repeatedly until they get to the level defined by the base value.

```
int factorial (int x) {
        if (x <=1)  return 1;
        else   return x * factorial(x-1);
}
```

Remember you are allowed to call other functions from inside a function.

Structs

Data structures allow us to define custom data types that group related data elements together into a single object.

Format

```
struct name {
        variable list
        };
```

Before you can use a data structure in your sketch, you need to define it. To define a data structure in the Arduino, you can use the struct statement. Here is the generic

format for declaration.

Example of declaration

```
struct sensorinfo {
        char date[9];
        int indoortemp;
        int outdoortemp;
   }morningTemps, noonTemps, eveningTemps;
```

Full Example: Declaring and Calling

```
struct sensorinfo {
        char date[9];
        int indoortemp;
        int outdoortemp;
    }morningTemps

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  strcpy(morningTemps.date, "01/01/14");
  morningTemps.indoortemp = 72;
  morningTemps.outdoortemp = 25;
  Serial.print ("Today's date is ");
  Serial.println(morningTemps.date);
  Serial.print("The morning outdoor temperate
is ");
  Serial.println(morningTemps.outdoortemp);

}
```

Unions

Unions allow you to have a variable take on different data types

Format

```
union {
        //variable list
        };
```

Example

```
Union {
        float analogInput;
        int digitalInput;
} sensorInput;
```

//Full Example:

```
Union{
        float analogInput;
        Int digitalInput;
        }sensorInput;

//Saving a value

sensorInput.analogInput = myFunction1();
sensorInput.digitalInput = myFunction2();


//Calling a value;
Serial.println(sensorInput.analogInput);
Serial.println(sensorInput.DigitalInput);
```

Using Libraries

Libraries allow you to bundle related functions into a single file that the Arduino IDE can compile into your sketches. Instead of having to rewrite the functions in your code, you just reference the library file from your code, and all the library functions become available. This is handy for Arduino Shields .

Defining the library in your sketch

```
#include 'libraryheader'
#include 'EEPROM'
```

Referencing the Library Functions

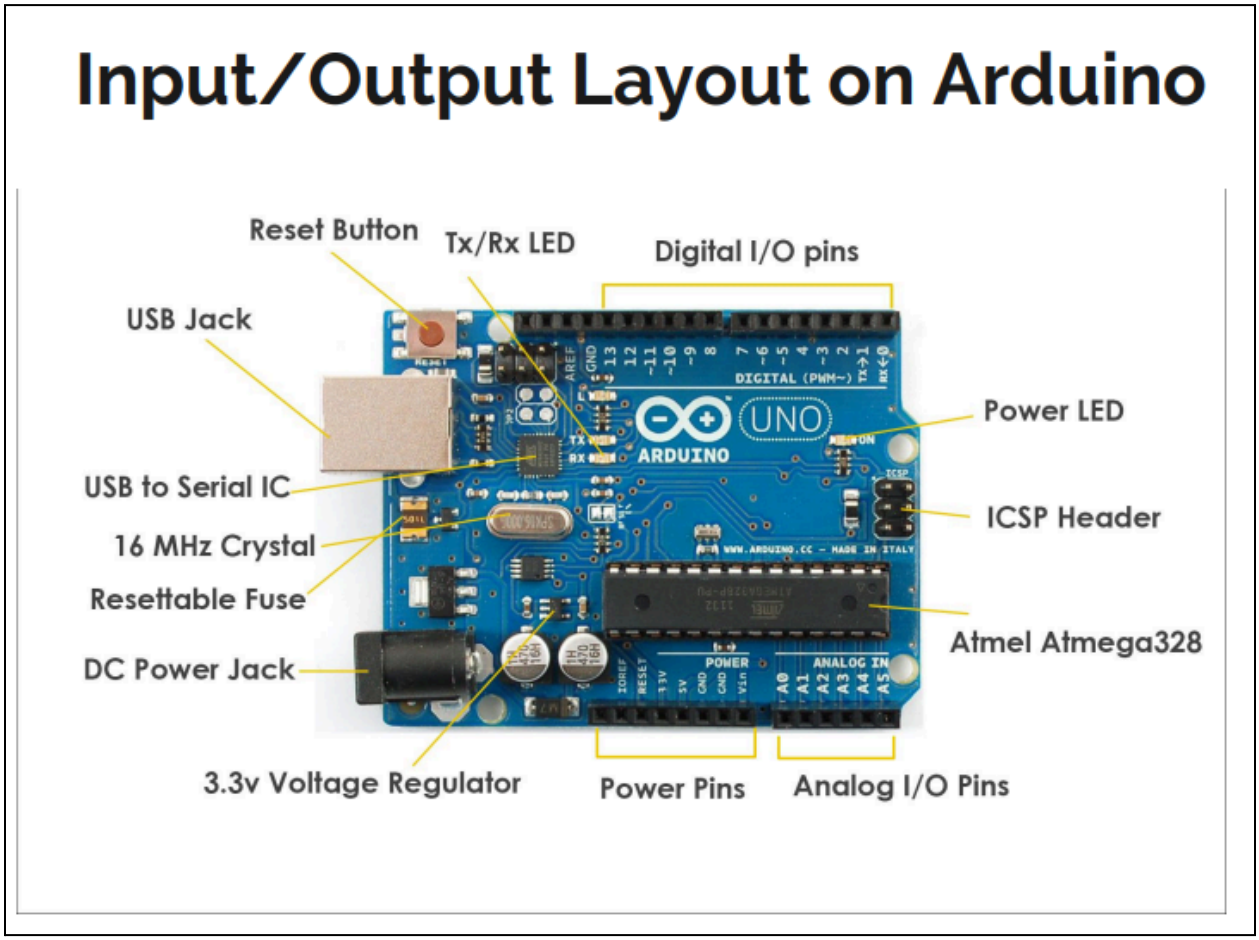
```
Library.function()
```

Ex. for the EEPROM library

```
EEPROM.read(0);
```

Installing your library :

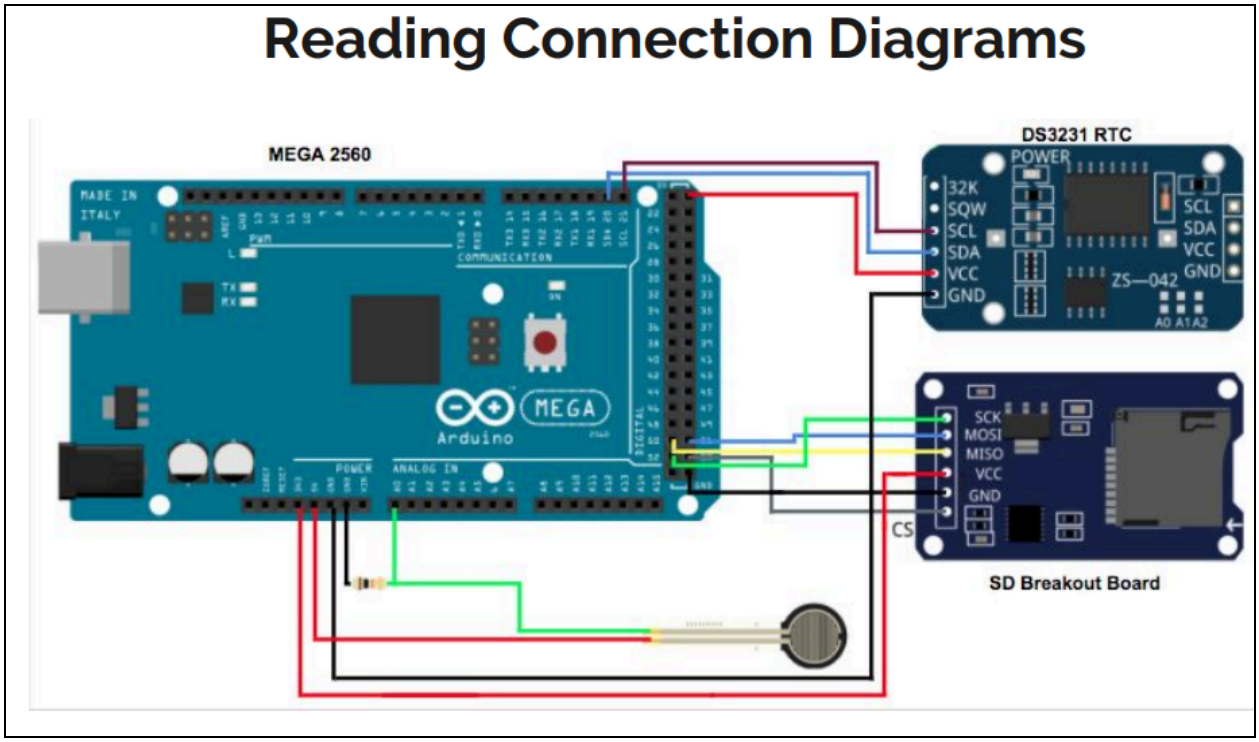
1. Open the Arduino IDE
2. Select the sketch from the menu bar
3. Select Import libraries from the submenu
4. Select Add library from the list of menu options



Writing to EEPROM

EEPROM is the long term memory in the Arduino. It keeps data stored even when powered off, like a USB flash drive. Important Note: EEPROM becomes unreliable after 100,000 writes

You'll first need to include a library file in your sketch:



#include<EEPROM.h>

After you include the standard EEPROM library file, you can use the two standard EEPROM functions in your code:

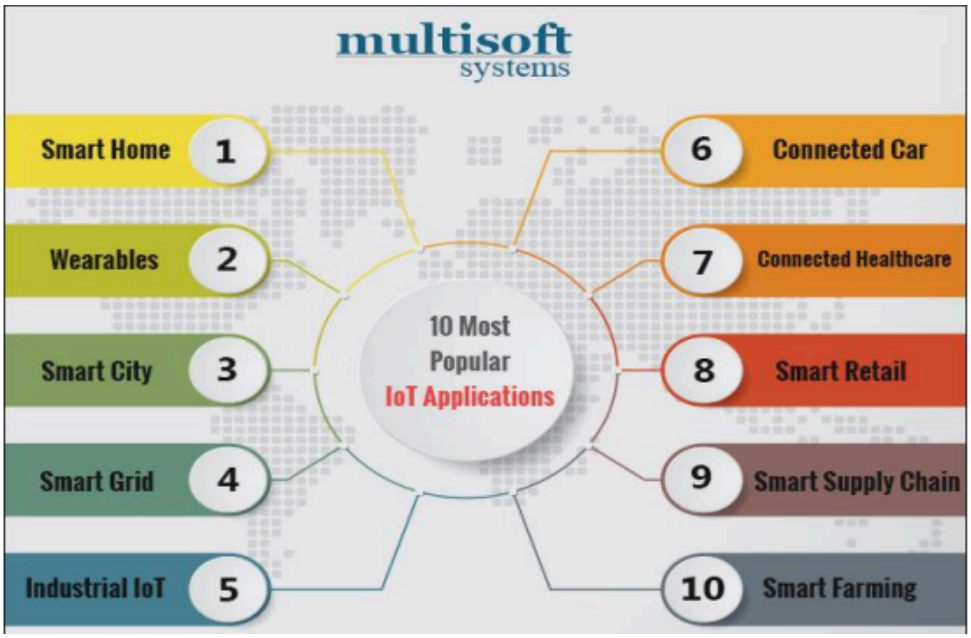
- **read (address)** - to read the data value stored at the EEPROM location specified by address
- **write (address, value)** - to read values to the EEPROM location specified by address.

```
#include <EEPROM.h>

void setup()
{
  for (int i = 0; i < 255; i++)
    EEPROM.write(i, i);
}

void loop()
{
}
```

Debugging Applications

- Compiler will give you the line code(s)
- Compiler will give you the error
- The line causing problems may get highlighted
- Look up errors online

Case Study & IoT Applications



Case Study & IoT Applications

Definition: "A case study is a research strategy and an empirical inquiry that investigates a phenomenon within its real-life context. Case studies are based on an in-depth investigation of a single individual, group or event to explore the causes of underlying principles".

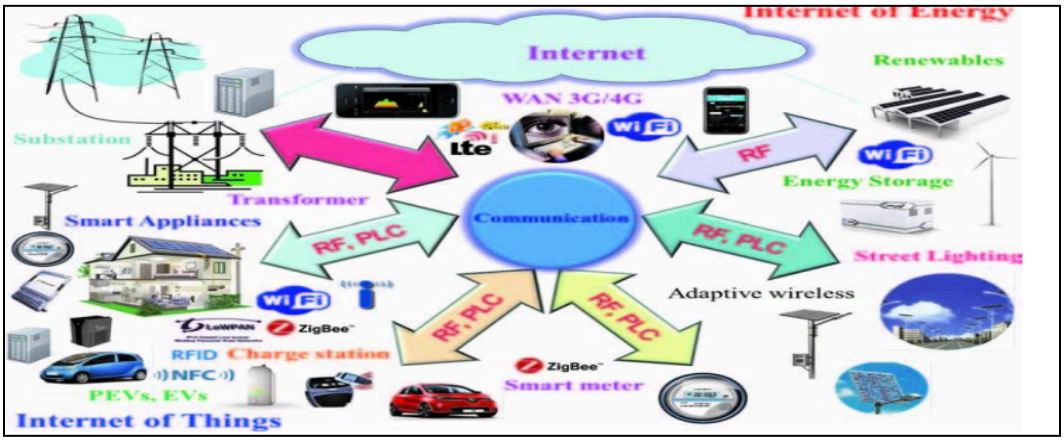
One of the most promising IoT use cases is creating smarter, more efficient cities. Public energy grids can be optimized to balance workloads, predict energy surges, and distribute energy more equitably to customers.Traffic lights could be synced using IoT to adapt to traffic conditions in real-time.

IoT is the next step in the evolution of the internet and is being used in about everything you can think of.

IoT Applications:

IoT applications promise to bring immense value into our lives. With newer wireless networks, superior sensors and revolutionary computing capabilities, the Internet of Things could be the next frontier in the race for its share of the wallet. IoT applications are expected to equip billions of everyday objects with connectivity and intelligence. It is already being deployed extensively, few applications of IoT:

- Wearables
- Smart Home Applications
- Smart Buildings
- Smart Infrastructure
- Securities
- Health Care
- Smart Cities
- Agriculture
- Industrial Automation



Smart Home, Smart Buildings and Infrastructure

IoT home automation is the ability to control domestic appliances by electronically controlled, internet-connected systems. It may include setting complex heating and lighting systems in advance and setting alarms and home security controls, all connected by a central hub and remote-controlled by a mobile app.

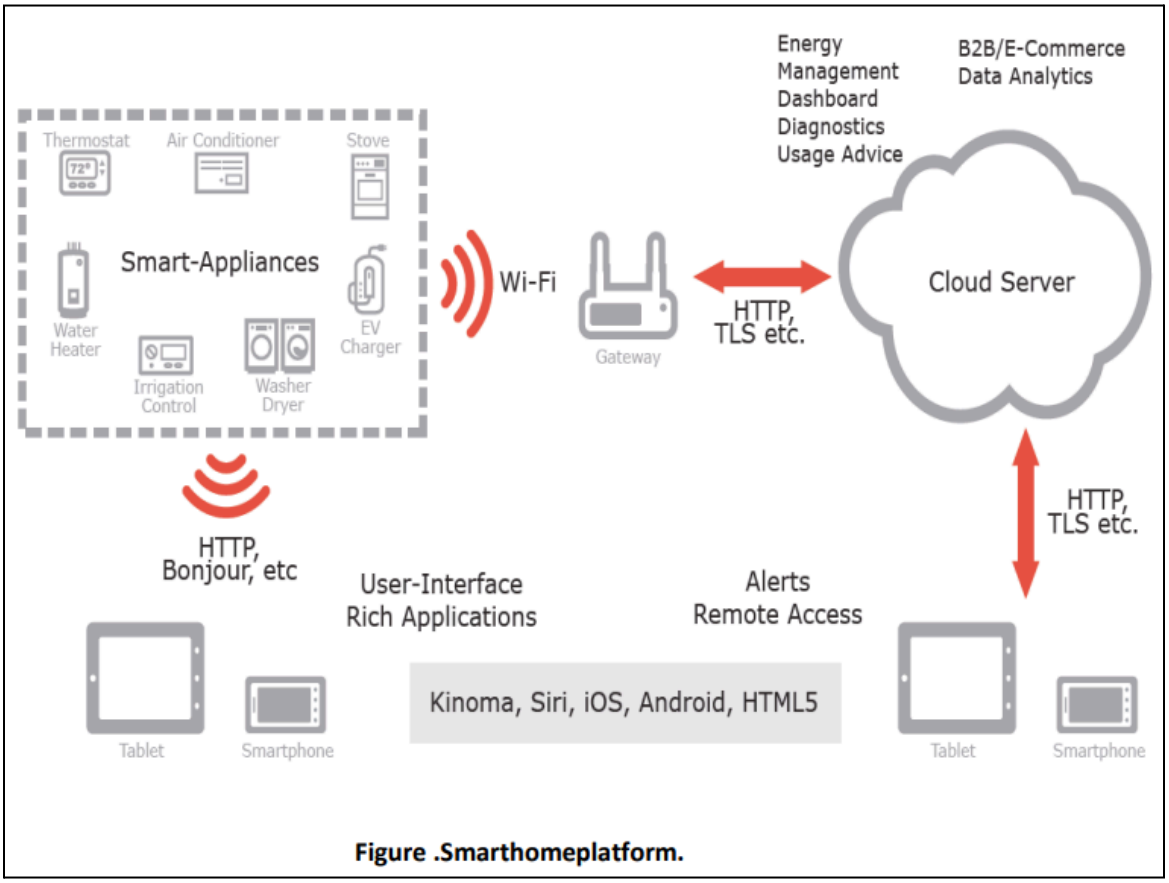


Figure .Smarthomeplatform.

The rise of Wi-Fi's role in home automation has primarily come about due tothe networked nature of deployed electronics where electronic devices (TVsand AV receivers, mobile devices, etc.) have started becoming part of thehome IP network and due the increasing rate of adoption of mobile computingdevices (smartphones, tablets, etc.), see above Figure.

The networking aspects are bringing online streaming services or network playback, while becoming a mean to control of the device functionality over the network. At the

same time mobile devices ensure that consumers have access to a portable ‘controller’ for the electronics connected to the network. Both types of devices can be used as gateways for IoT applications. In this context many companies are considering building platforms that integrate the building automation with entertainment,healthcare monitoring, energy monitoring and wireless sensor monitoring in the home and building environments.

IoT applications using sensors to collect information about the operating conditions combined with cloud hosted analytics software that analyzes disparate data points will help facility managers become far more proactive about managing buildings at peak efficiency.

Issues of building ownership (i.e., building owner, manager, or occupants)challenge integration with questions such as who pays initial system costand who collects the benefits over time. A lack of collaboration between the subsectors of the building industry slows new technology adoption and canvent new buildings from achieving energy, economic and environmental performance targets.

Integration of cyber physical systems both within the building and with external entities, such as the electrical grid, will require stakeholder cooperation to achieve true interoperability. As in all sectors, maintaining security will be a critical challenge to overcome.

Within this field of research the exploitation of the potential of wireless sensor networks (WSNs) to facilitate intelligent energy management in buildings,which increases occupant comfort while reducing energy demand, is highly relevant.

In addition to the obvious economic and environmental gains from the introduction of such intelligent energy management in buildings other positive effects will be achieved. Not least of which is the simplification of building control; as placing monitoring, information feedback equipment and control capabilities in a single location will make a buildings’ energy management system easier to handle for the building owners, building managers,maintenance crews and other users of the building.

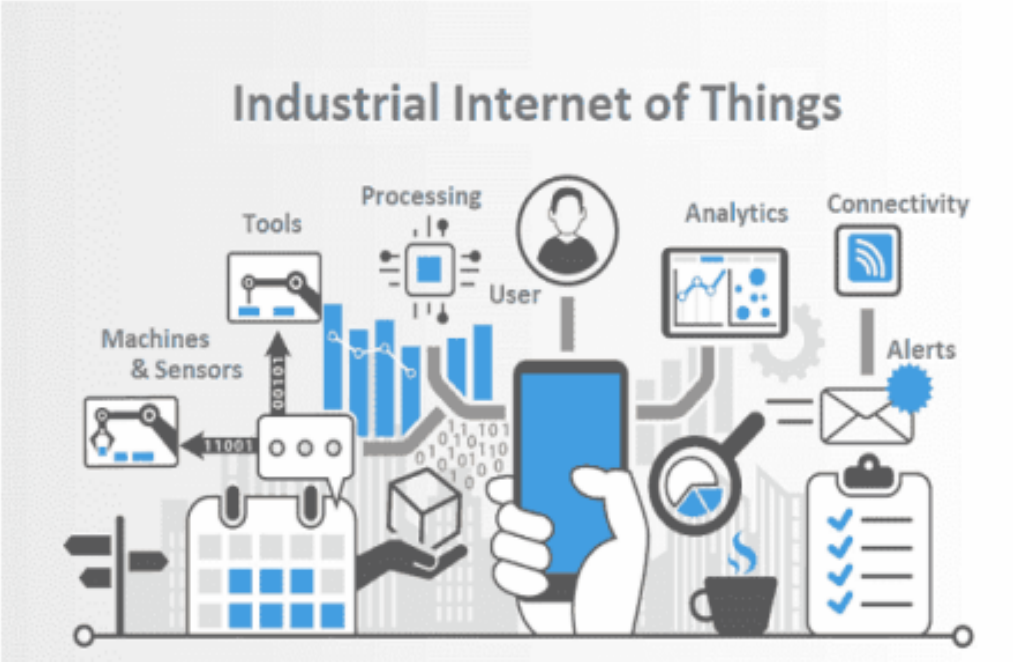
Using the Internet together with energy management systems also offers an

opportunity to access a buildings energy information and control systems from a laptop or a Smartphone Placed anywhere in the world. This has a huge potential for providing the managers,owners and inhabitants of buildings with energy consumption feedback and the ability to act on that information.

In the context of the future Internet of Things, Intelligent Building ManagementSystems can be considered part of a much larger information system.This system is used by facilities managers in buildings to manage energy use and energy procurement and to maintain buildings systems. It is based on the infrastructure of the existing Intranets and the Internet, and therefore utilizes the same standards as other IT devices. Within this context reductions in the cost and reliability of WSNs are transforming building automation, by making the maintenance of energy efficient healthy, productive work spaces in buildings increasingly cost effective.

IoT Application in industries

IoT in industry is a rapidly developing area. Numerous IoT research and application projects have been done by universities or in joint industry- university consortia in recent years.



Internet of things (IoT) has become part of your daily life. The “things connectedto the internet” idea is continuously evolving in content, areas of applications, visions and technology. New real life and industrial projects havebeen done and joint future oriented industry and government initiatives suchas Industry 4.0 in Germany, have been started [1].Since Industrial production is one of the world’s biggest economic factorsone of the major objectives of these initiatives is to bring the paradigms of theIoT to the factories enabling them to cope with the challenges raised by popularmegatrends.

The foremost megatrends relevant for factories are globalization,progressing technological evolution, the dynamization of product life cycles,the aging work force and the shortage of resources. Central effects are theacceleration of innovation cycles and the increasing customer demand forindividualized mass produces with highest quality expectations.Within the context of industrial production IoT projects and applicationsare developing in manufacturing, supply chain, supervision and servicing.A major question in all projects is about the value, the benefit such applicationcan bring to the user, to the owner or to society.

The value question is extremely pertinent in the industry: in the manufacturing industry entire factory related processes, but also in industrial applications where it comes to ensure operation of industrial installations and provide supervision, and improved life service. It is the value which such applications bring which will determine their adoption, acceptance and wide use. However, this value is very difficult to quantify and prove, and it depends on multiple aspects which are strongly application area dependent.

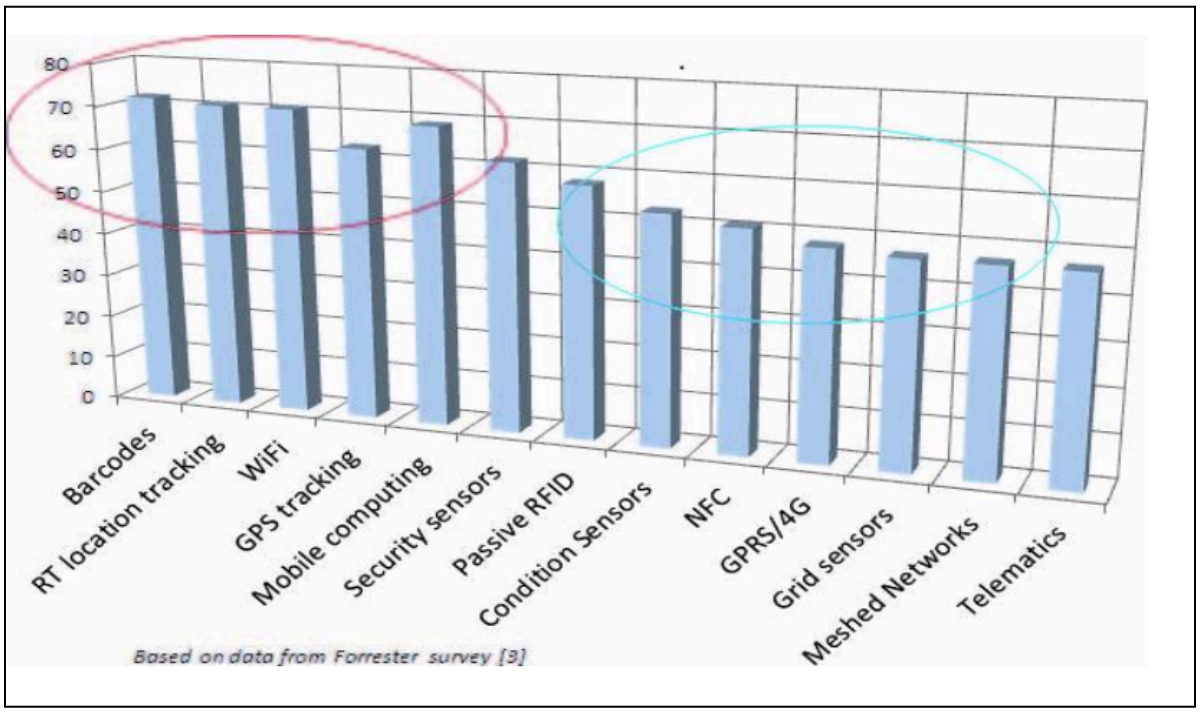
IoT applications form the value creation for industry and brings together expert opinions from academia, research and industry. The industrial application of IoT is multi- facetted and each of the subsections in this paper will highlight an aspect related to industrialapplication, discuss or show a case or the evolution and potential of a specific technology from industry application point of view. The paper is having a holistic manner to industrial challenges and requirements. Also it will refer to factory concepts and applications supported by IoT, including processes and flows taking a view on related technologies and their evolution.

IoT applications benefit and value creation in an industrial environment may have its

origin in different aspects, depending on the application type. There is no value but "values" each contributing to the total benefit such as:

- Value from visibility identification, location tracking
- Value form IoT-supported safety in hard industrial environments
- Value from right information providing or collecting
- Value form improved industrial operation and flows in industry
- Value from reduced production losses
- Value from reduced energy consumption
- Value from new type of processes made possible by IoT applications
- Value form new type of maintenance and lifetime approaches
- Value enabled by smart objects, connected aspects
- Value from sustainability.

View on very important and important perceived IoT technologies expected to bring value in applications:



The status and estimated potential of IoT applications is presented in Figure 5.3 considering three major areas: supply chain, future industry/future actory and over

lifetime applications and activities such as logistics, manufacturing and service/maintenance. A strong potential and additional application is expected in industry operation and industry lifetime applications including lifetime service.

Areas	Supply chain	Industry	Lifetime
Activities	Logistics	Manufacturing	Service
IoT **present** Applications and Value	Many	Some	Few
IoT **additional** Applications **Potential**	Increase	Strong	Strong

Figure.Status and estimated potential of IoT applications.

IoT application requirements and capabilities

The expectations toward IoT applications in industry are high. The capabilities they have to offer are depending strongly on the industrial area and the concrete application. For example the environment where IoT application may be used may range from clean room condition and normal ambient temperatures to heavy and dirty environment, locations with high temperatures, areas withexplosion risk, areas with metallic surroundings, and corrosive environment on sea or underground.

A list of a set of industry related capabilities and requirements is presented below, without claiming completeness. The list items are related to the IoT hardware, software and to serviceability and management aspects. Comments have been added to all items to make the requirement more specific. The IoT application capabilities for industrial application should meet requirements such as:

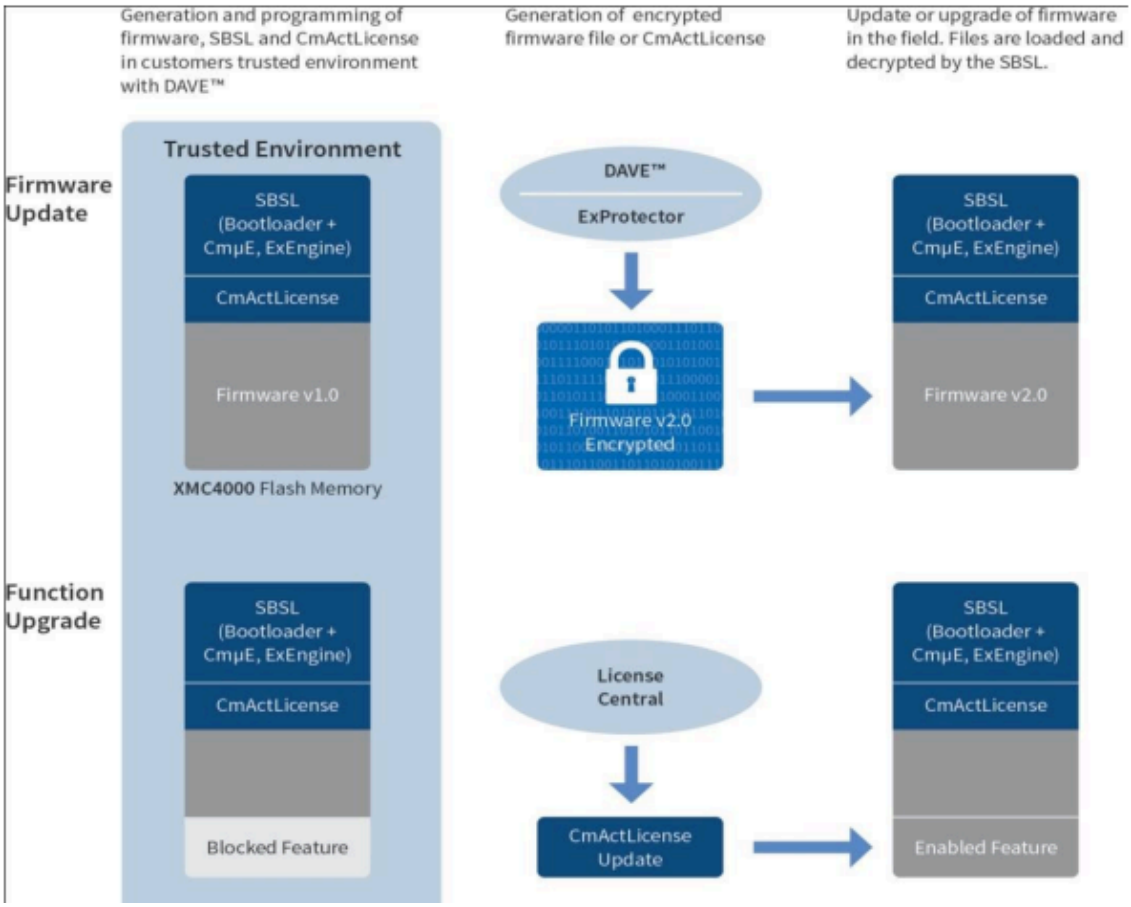
Security

- IoT devices are connected to your desktop or laptop. Lack of security increases the risk of your personal information leaking while the data is collected and transmitted to the IoT device.
- IoT devices are connected with a consumer network. This network is also connected with other systems. So if the IoT device contains any security vulnerabilities, it can be harmful to the consumer's network. This vulnerability can attack other systems and damage them.
- Sometimes unauthorized people might exploit the security vulnerabilities to create risks to physical safety.
-
  - Reliability.
    Reliable IoT devices and systems should allow a continuous operation of industrial processes and perform on-site activities.
  - Robustness.
    The IoT application and devices should be robust and adapted to the task and hard working conditions. This should include also the certifications for the specific work environment where they are used.
  - Reasonable cost.
    Cost aspects are essential and should be fully justifiable and adapted to the benefit. It is basically about the right balance between cost and benefit rather than low cost. Also the costs are related to a more holistic view and life costs and consider the impact on the whole industrial installation in case of a failed IoT device or application.

Privacy Risks:

- In IoT, devices are interconnected with various hardware and software, so there are obvious chances of sensitive information leaking through unauthorized manipulation.
- All the devices are transmitting the user's personal information such as name, address, date of birth, health card information, credit card detail and much more without encryption.

Though there are security and privacy concerns with IoT, it adds values to our lives by allowing us to manage our daily routine tasks remotely and automatically, and more importantly, it is a gamechanger for industries.



IoT Application of home appliances

Internet of Things is a technology that can connect to the internet without the influence of people and send information collected to users through this internet network to which they are connected. Devices in this dynamic are very common today. Many homes, companies and even public organizations benefit from this technology. Used in smart home IoT home appliances is also one of them.

A house must have smart devices to be smart. These smart devices are the building blocks of today's technology. So why are these devices and apps smart? First, these devices have their own Internet. With this internet tool, users can receive information

from the device. With this internet connection, you can get a lot of information from your smart device. This information which receives from smart devices makes safety for your living area.

Smart devices work with technological devices while making you and your home a more secure space. The biggest hero of these technological devices is microprocessors. microprocessors act as the brain for your smart device. There are sensors that allow your smart devices to be classified according to their characteristics and detect the danger or differences in your home.

There are many sensors classified by type. Motion sensors, light sensors, image detection, and processing sensors are one of them. For example, if the position of your belongings changes without your knowledge, there are motion sensors that can detect this position change. The motion sensor detects the position change and sends you information about this.

Home Appliance in Internet of Things

Smart home systems are integrated and enable you to play an active role in every part of your home by surrounding your home. When you're not at home, but your mind stays

at home, it’s behind you. With smart home systems, you can intervene in your home as if you are at home and perform the necessary controls.In addition to these protection systems, smart home appliances have been making human life easier since the day it was developed.

Smart Washing Machine:

It is very important to save time in daily life. we live in a period where we have to keep up. that’s where technology comes in. You can access the developed smart washing machine on your smartphone. you can monitor and control the process at the same time. This smart washing machine can also dry your laundry with the control application.

Smart Refrigerator with Internet of Things:

Internet in this kitchen which makes life easier for you and your family in the kitchen.

With this internet connection, you can transmit a lot of information to your shopping list in the weather. You can also view the inside of your refrigerator with its camera technology.

Shortest Way to Dry Hair:

This time it has infrared technology. With this technology, the device is created wirelessly. Wireless shape so you can dry your hair without connecting the machine .

Smart Doorbell:

The most important thing in smart home applications is known to be secure and protected home. With this smart doorbell designed for security, you can recognize people who come to your home with high quality. The night also has infrared technology added to the smart bell. This will also send the screen to you when it gets dark.

Smart Camera for Safe Home:

Control of your home is in your hands from every part. This smart camera sends records from every part of your home to your smartphone with the Internet of Things

technology. Research on smart camera technology will continue for those who want a safe life.

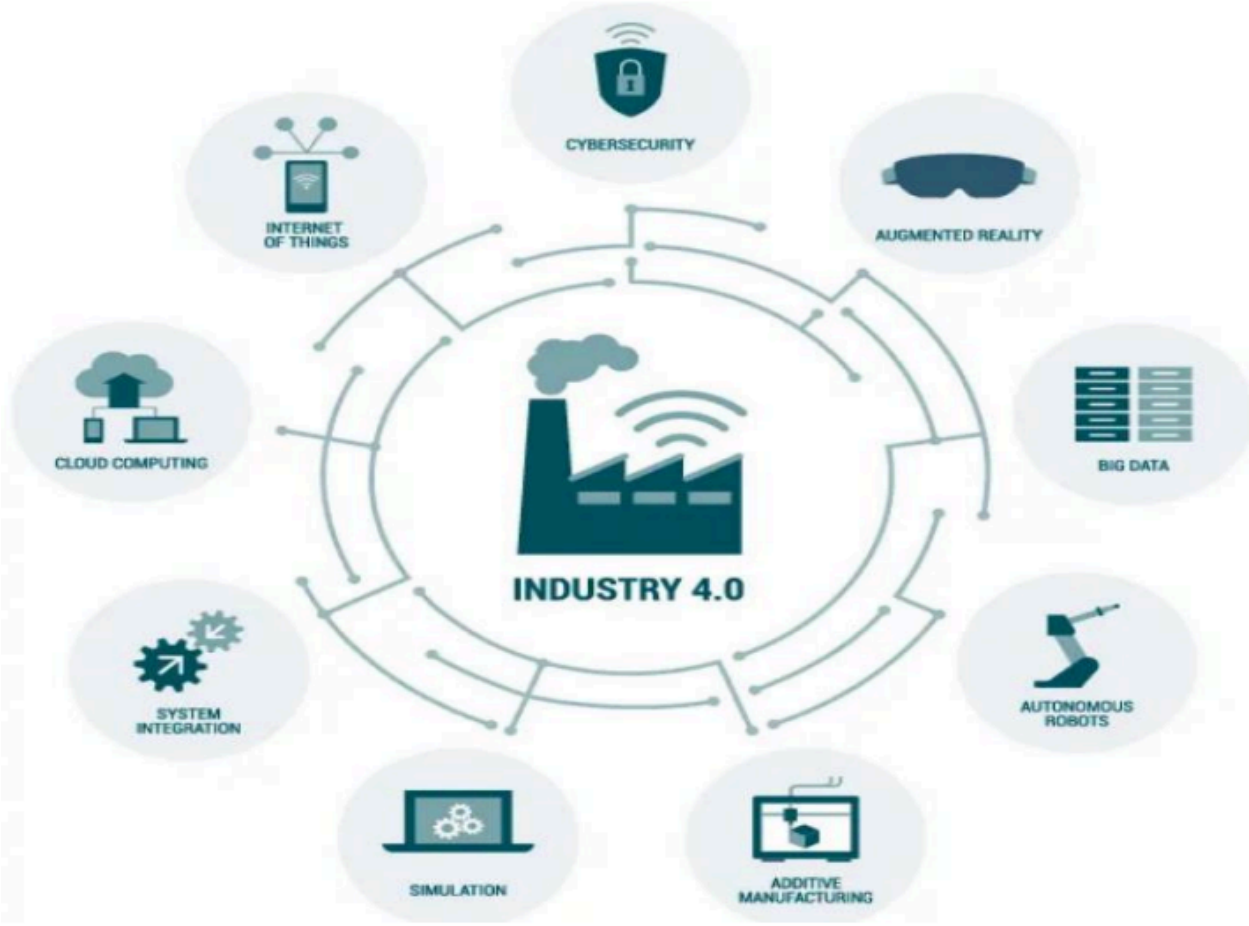
Industry 4.0 concepts

Industry 4.0 refers to a new phase in the Industrial Revolution that focuses heavily on interconnectivity, automation, machine learning, and real-time data. Industry 4.0, also sometimes referred to as IIoT(Industrial Internet of Things) or smart manufacturing which provides physical production and operations with smart digital technology, machine learning and big data to create a more holistic and better connected ecosystem for companies that focus on manufacturing and supply chain management.

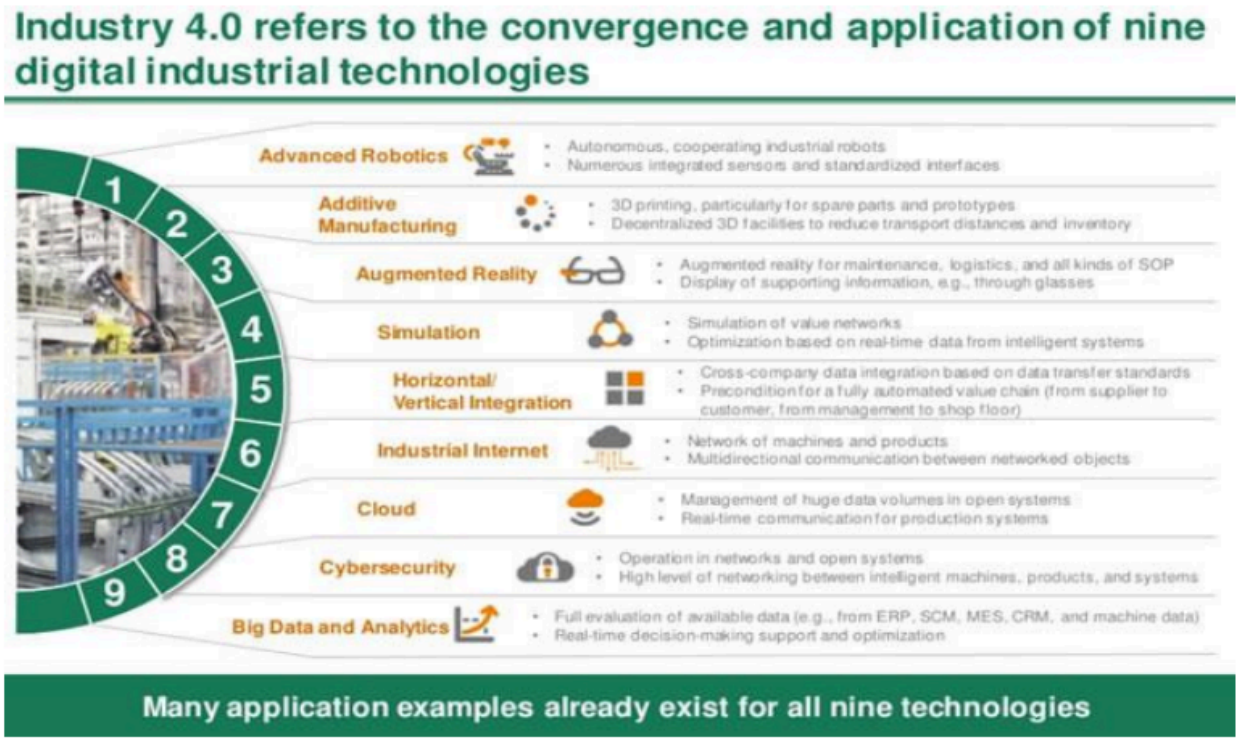
While every company and organization operating today is different, they all face a common challenge—the need for connectedness and access to real-time insights across processes, partners, products, and people. That’s where Industry 4.0 comes into play. Industry 4.0 is not just about investing in new technology and tools to improve manufacturing efficiency but it’s about revolutionizing the way the entire business operates and grows.

Industry 4.0 refers to the use of automation and data exchange in manufacturing. According to the Boston Consulting Group there are nine principal technologies that make up Industry 4.0: Autonomous Robots, Simulation, Horizontal and Vertical System Integration, the Industrial Internet of Things, Cybersecurity, The Cloud, Additive Manufacturing, Data and Analytics, and Augmented Reality. These technologies are used to create a “smart factory” where machines, systems, and humans communicate with each other in order to coordinate and monitor progress along the assembly line. Networked devices provide sensor data and are digitally controlled. The net effect is the ability to rapidly design, modify, create, and customize things in the real world, while lowering costs and reacting to changes in consumer preferences, demand,the supply chain and technology.

The goal is to enable autonomous decision-making processes, monitor assets and processes in real-time, and enable equally real-time connected value creation networks through early involvement of stakeholders, and vertical and horizontal integration.



Today some companies have invested in a few of these technologies; predominantly the traditional pillars of the third platform such as cloud and Big Data / Analytics and increasingly in the Industrial Internet of Things from an integrated perspective and thus overlapping with several of these “technologies” or maybe better: sets of technologies and connected benefits.



Evolution of Industry 4.0

There are four distinct industrial revolutions that the world either has experienced or continues to experience today.

- **The First Industrial Revolution**

The first industrial revolution happened between the late 1700s and early 1800s. During this period of time, manufacturing evolved from focusing on manual labor performed by people and aided by work animals to a more optimized form of labor performed by people through the use of water and steam-powered engines and other types of machine tools.

- **The Second Industrial Revolution**

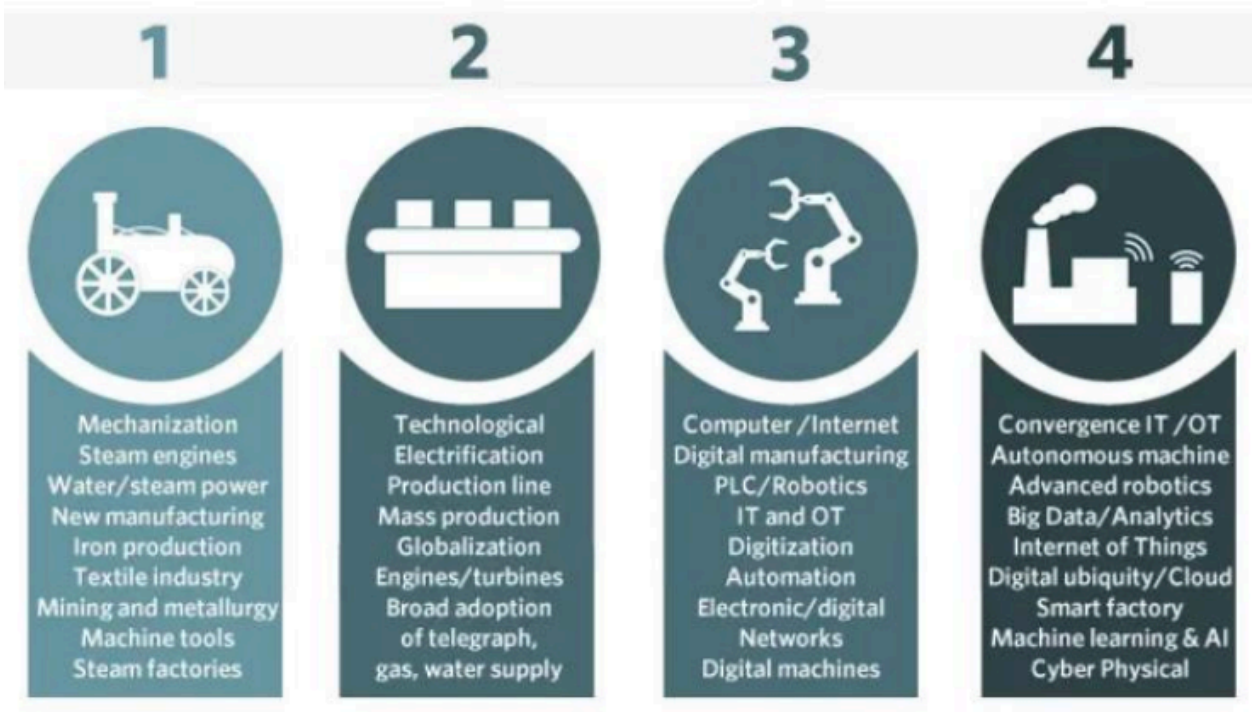
In the early part of the 20th century, the world entered a second industrial revolution with the introduction of steel and use of electricity in factories. The introduction of electricity enabled manufacturers to increase efficiency and helped make factory machinery more mobile. It was during this phase that mass production concepts like the assembly line were introduced as a way to boost productivity.

- **Third Industrial Revolution**

Starting in the late 1950s, a third industrial revolution slowly began to emerge, as manufacturers began incorporating more electronic and eventually computer technology into their factories. During this period, manufacturers began experiencing a shift that put less emphasis on analog and mechanical technology and more on digital technology and automation software.

- **Fourth Industrial Revolution[Industry 4.0]**

Fourth industrial revolution has emerged known as Industry 4.0. Industry 4.0 takes the emphasis on digital technology from recent decades to a whole new level with the help of interconnectivity through the Internet of Things (IoT), access to real-time data, and the introduction of cyber-physical systems. Industry 4.0 offers a more comprehensive, interlinked and holistic approach to manufacturing. It connects physical with digital, and allows for better collaboration and access across departments, partners, vendors, product, and people. An industry 4.0 empowers business owners to control and understand every aspect of their operation, and allows them to leverage instant data to boost productivity, improve processes, and drive growth.



Industry 4.0 is often used interchangeably with the notion of the fourth industrial revolution. It is characterized among others by

1) even more automation than in the third industrial revolution

2) the bridging of the physical and digital world through cyber-physical systems, enabled by Industrial IoT

3) a shift from a central industrial control system to one where smart products define the production steps

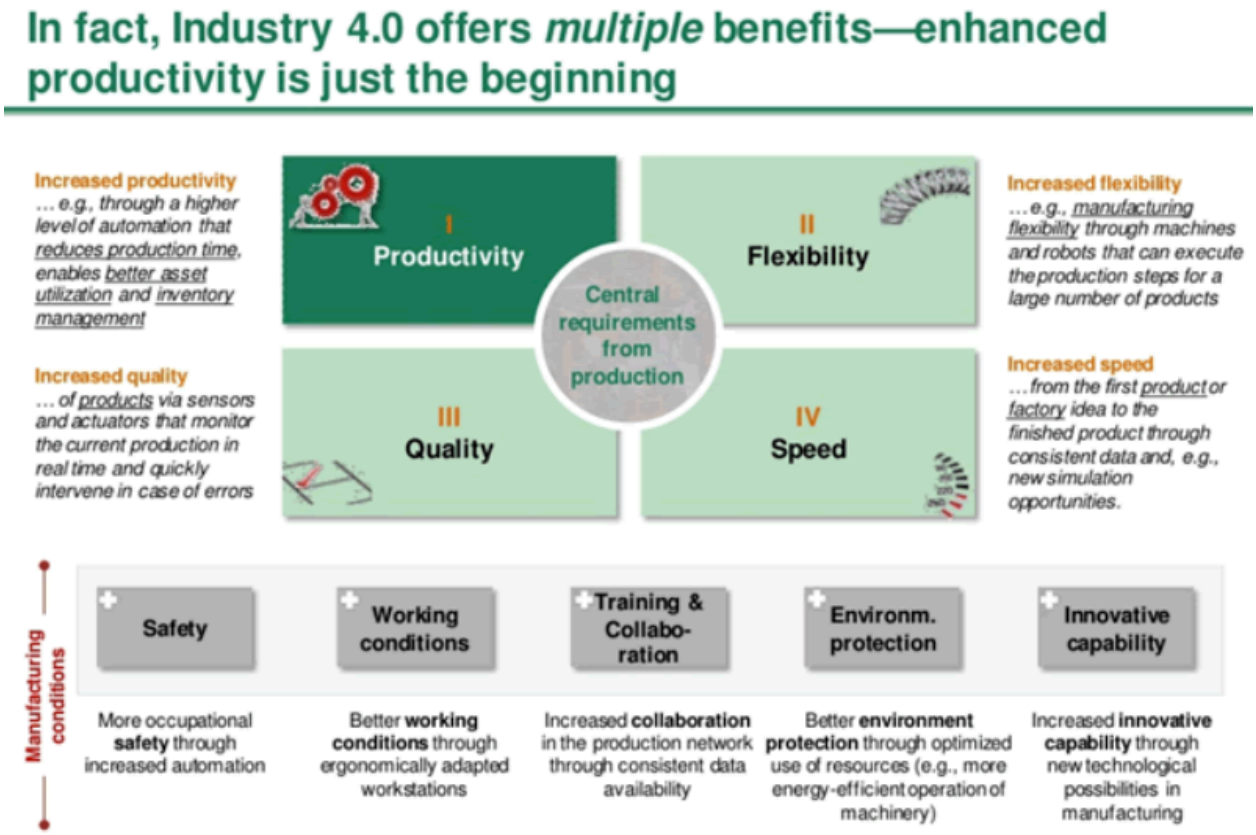
4) closed-loop data models and control systems and

5) personalization/customization of products.

Benefits of Industry 4.0

Industry 4.0 spans the entire product life cycle and supply chain, design, sales, inventory, scheduling, quality, engineering, and customer and field service. Everyone shares informed, up-to-date, relevant views of production and business processes and much richer and more timely analytics.

The essential goal of Industry 4.0 is to make manufacturing and related industries such as logistics faster, more efficient and more customer-centric, while at the same time going beyond automation and optimization and detect new business opportunities and models.



Most of the benefits of Industry 4.0 are obviously similar to the benefits of the digital transformation of manufacturing, the usage of the IoT in manufacturing, operational and

business process optimization, information-powered ecosystems of value, digital transformation overall, the Industrial Internet and many other topics on our website. Few of the key benefits of Industry 4.0 are.

1. Enhanced productivity through optimization and automation
2. Real-time data for a real-time supply chain in a real-time economy
3. Higher business continuity through advanced maintenance and monitoring possibilities
4. Better quality products: real-time monitoring, IoT-enabled quality improvement and cobots
5. Better working conditions and sustainability
6. Personalization and customization for the ‘new’ consumer
7. Improved agility
8. The development of innovative capabilities and new revenue model

ELECTRONICS

Capacitor

An electric circuit element that has an ability of storing electrical energy in the form of electric field is called a capacitor. The property of the capacitor by virtue of which it stores electrical energy is known as capacitance.

In other words, a circuit element whose terminal voltage is directly proportional to integral of current with respect to time is called a capacitor.

A simple capacitor consists of two metallic plates separated by an insulating material. This insulating material is called dielectric and it stores the electrical energy in the form of electric field. Depending on the type of dielectric material used, there are several types of capacitors like paper capacitor, air capacitor, mica capacitor, ceramic capacitor, electrolytic capacitor, etc.

Capacitors are passive electronic components consisting of two or more pieces of conducting material separated by an insulating material. The capacitor is a component which has the ability or “capacity” to store energy in the form of an electrical charge producing a potential difference (*Static Voltage*) across its plates, much like a small rechargeable battery.

There are many different kinds of capacitors available from very small capacitor beads used in resonance circuits to large power factor correction capacitors, but they all do the same thing, they store charge.

In its basic form, a capacitor consists of two or more parallel conductive (metal) plates which are not connected or touching each other, but are electrically separated either by air or by some form of a good insulating material. This insulating material could be waxed paper, mica, ceramic, plastic or some form of a liquid gel as used in electrolytic

capacitors.

As a good introduction to capacitors, it is worth noting that the insulating layer between a capacitors plates is commonly called the **Dielectric**.

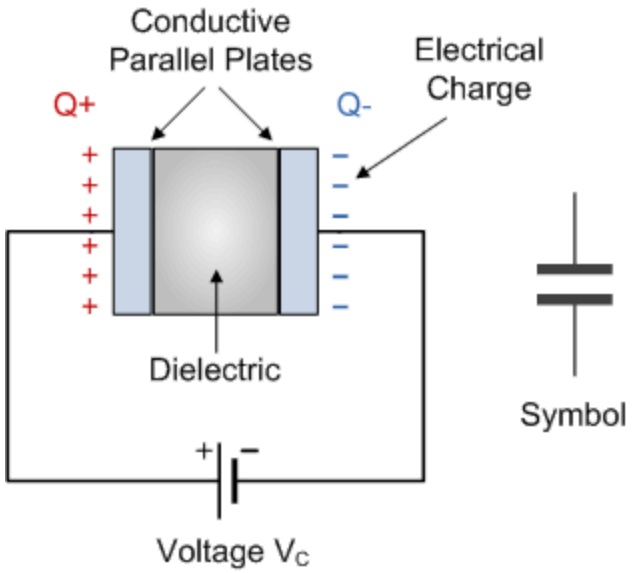
A Typical Capacitor

Due to this insulating layer, DC current can not flow through the capacitor as it blocks it allowing instead a voltage to be present across the plates in the form of an electrical charge.

The conductive metal plates of a capacitor can be either square, circular or rectangular, or they can be of a cylindrical or spherical shape with the general shape, size and construction of a parallel plate capacitor depending on its application and voltage rating.

When used in a direct current or DC circuit, a capacitor charges up to its supply voltage but blocks the flow of current through it because the dielectric of a capacitor is non-conductive and basically an insulator. However, when a capacitor is connected to an alternating current or AC circuit, the flow of the current appears to pass straight through the capacitor with little or no resistance.

There are two types of electrical charge, a positive charge in the form of Protons and a negative charge in the form of Electrons. When a DC voltage is placed across a capacitor, the positive (+ve) charge quickly accumulates on one plate while a corresponding and opposite negative (-ve) charge accumulates on the other plate. For every particle of +ve charge that arrives at one plate a charge of the same sign will depart from the -ve plate.



Resistor

An electric circuit element that introduces electrical friction or resistance in the path of electric current is called a resistor. The characteristic by which it opposes the flow of current is known as resistance. The resistance of a resistor is denoted by symbol R and measured in Ohms (ΩΩ).

There are many different **Types of Resistor** available for the electronics constructor to choose from, from very small surface mount chip resistors up to large wirewound power resistors.

The principal job of a resistor within an electrical or electronic circuit is to “resist” (hence the name **Resistor**), regulate or to set the flow of electrons (current) through them by using the type of conductive material from which they are composed. Resistors can also be connected together in various series and parallel combinations to form resistor networks which can act as voltage droppers, voltage dividers or current limiters within a circuit.

Resistors are what are called “Passive Devices”, that is they contain no source of power or amplification but only attenuate or reduce the voltage or current signal passing through them. This attenuation results in electrical energy being lost in the form of heat as the resistor resists the flow of electrons through it.

Then a potential difference is required between the two terminals of a resistor for current to flow. This potential difference balances out the energy lost. When used in DC circuits the potential difference, also known as a resistors voltage drop, is measured across the terminals as the circuit current flows through the resistor.

Most types of resistor are linear devices that produce a voltage drop across themselves when an electrical current flows through them because they obey Ohm’s Law, and different values of resistance produces different values of current or voltage. This can be very useful in Electronic circuits by controlling or reducing either the current flow or voltage produced across them we can produce a voltage-to-current and current-to-voltage converter.

There are many thousands of different **Types of Resistor** and are produced in a variety of forms because their particular characteristics and accuracy suit certain areas of application, such as High Stability, High Voltage, High Current etc, or are used as general purpose resistors where their characteristics are less of a problem.

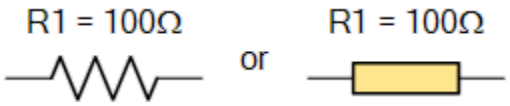
Some of the common characteristics associated with the humble resistor are; **Temperature Coefficient, Voltage Coefficient, Noise, Frequency Response, Power**

as well as a resistors **Temperature Rating, Physical Size** and **Reliability**.

In all Electrical and Electronic circuit diagrams and schematics, the most commonly used symbol for a fixed value resistor is that of a “zig-zag” type line with the value of its resistance given in Ohms, Ω. Resistors have fixed resistance values from less than one ohm, (<1Ω) to well over tens of millions of ohms, (>10MΩ) in value.

Fixed resistors have only one single value of resistance, for example 100Ω, but variable resistors (potentiometers) can provide an infinite number of resistance values between zero and their maximum value.

Standard Resistor Symbols:



The symbol commonly used in schematic and electrical drawings for a Resistor can either be a “zig-zag” type line or a rectangular box.

All modern fixed value resistors can be classified into four broad groups:

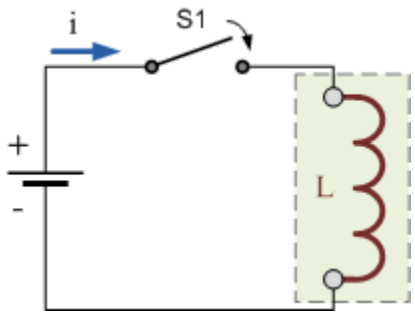
- Carbon Composition Resistor – Made of carbon dust or graphite paste, low wattage values
- Film or Cermet Resistor – Made from conductive metal oxide paste, very low wattage values
- Wire-wound Resistor – Metallic bodies for heatsink mounting, very high wattage ratings
- Semiconductor Resistor – High frequency/precision surface mount thin film technology

There are a large variety of fixed and variable resistor types with different construction styles available for each group, with each one having its own particular characteristics, advantages and disadvantages compared to the others. To include all types would

make this section very large so I shall limit it to the most commonly used, and readily available general purpose types of resistors.

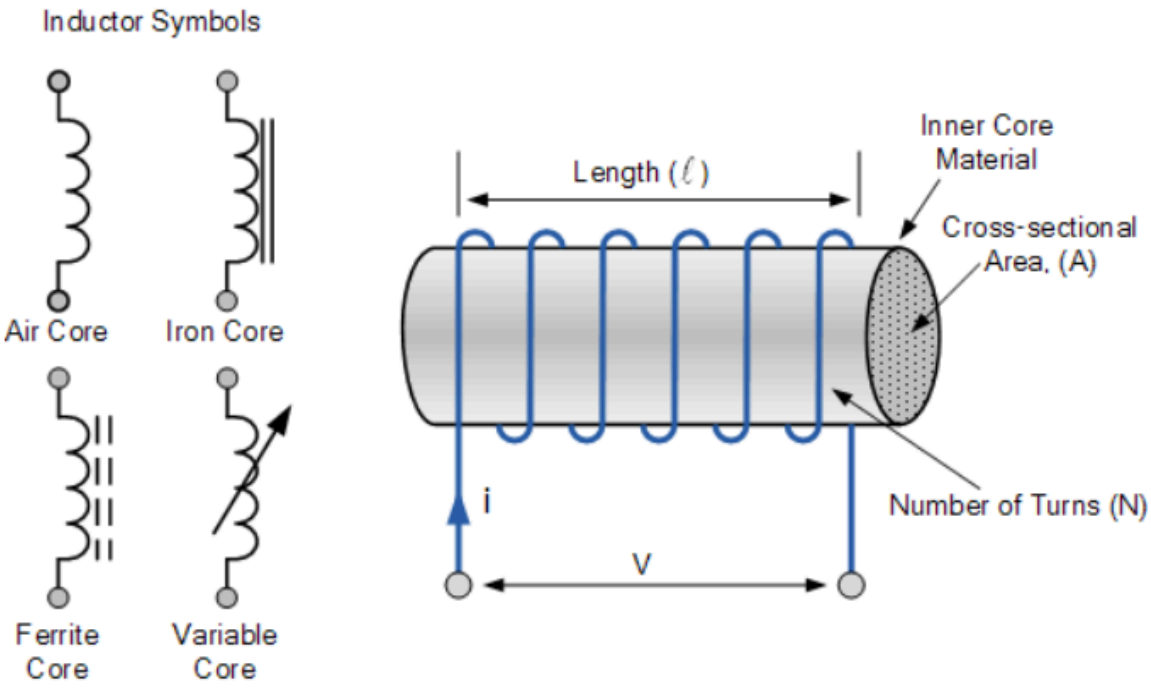
Inductor

An Inductor is a passive electrical component consisting of a coil of wire which is designed to take advantage of the relationship between magentism and electricity as a result of an electric current passing through the coil.

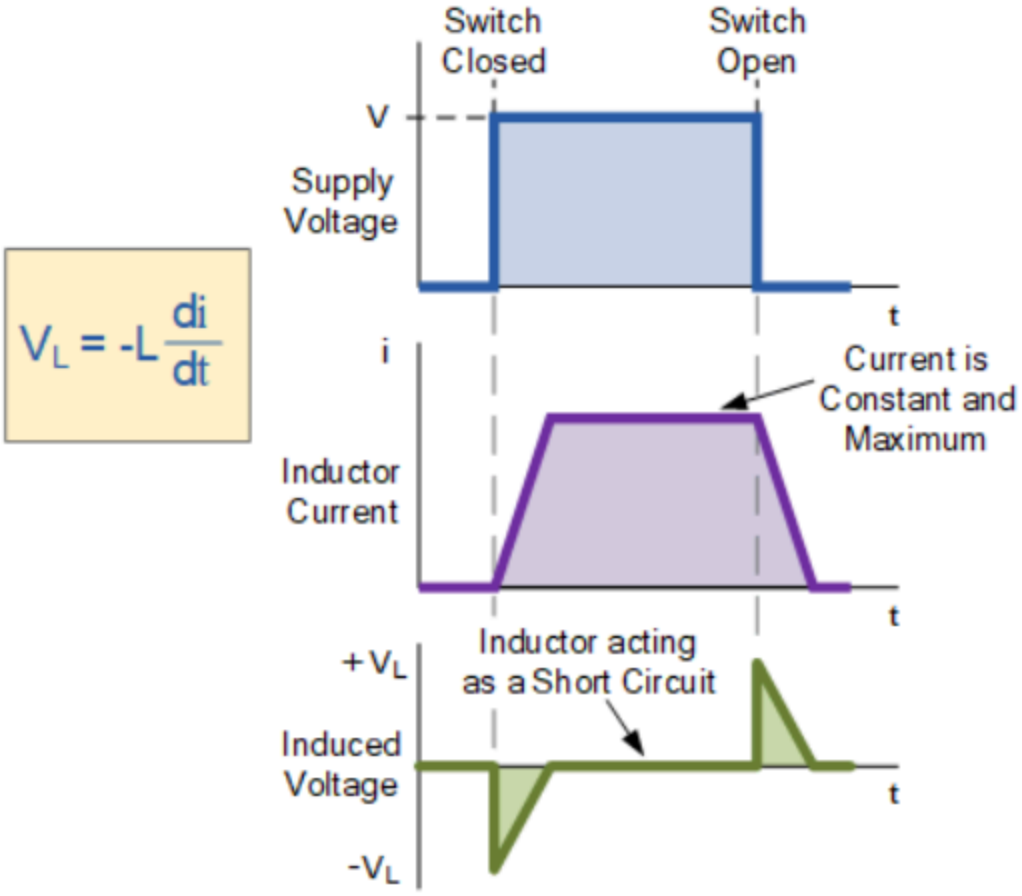


The schematic symbol for a inductor is that of a coil of wire so therefore, a coil of wire can also be called an **Inductor**. Inductors usually are categorised according to the type of inner core they are wound around, for example, hollow core (free air), solid iron core or soft ferrite core with the different core types being distinguished by adding continuous or dotted parallel lines next to the wire coil

Inductor Symbol:



Current and Voltage in an Inductor:



Diode

Diodes are made from a single piece of **Semiconductor** material which has a positive "P-region" at one end and a negative "N-region" at the other, and which has a resistivity value somewhere between that of a conductor and an insulator.

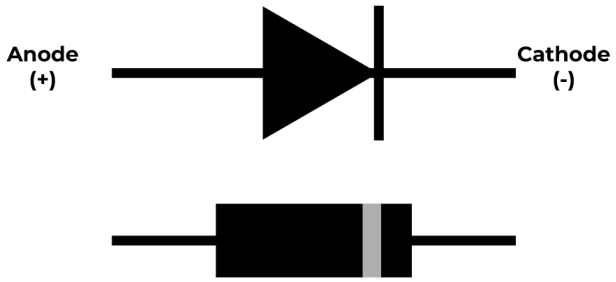
A **diode** is a two-terminal electronic component that conducts current primarily in one direction (asymmetric conductance). It has low (ideally zero) resistance in one direction

and high (ideally infinite) resistance in the other.

A semiconductor diode, the most commonly used type today, is a crystalline piece of semiconductor material with a p–n junction connected to two electrical terminals.[4] It has an exponential current–voltage characteristic. Semiconductor diodes were the first semiconductor electronic devices. The discovery of asymmetric electrical conduction across the contact between a crystalline mineral and a metal was made by German physicist Ferdinand Braun in 1874. Today, most diodes are made of silicon, but other semiconducting materials such as gallium arsenide and germanium are also used.

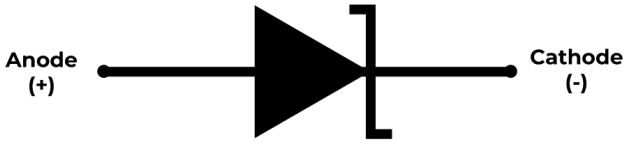
The obsolete thermionic diode is a vacuum tube with two electrodes, a heated cathode and a plate, in which electrons can flow in only one direction, from cathode to plate.

Among many uses, diodes are found in rectifiers to convert alternating current (AC) power to direct current (DC), demodulation in radio receivers, and can even be used for logic or as temperature sensors. A common variant of a diode is a light-emitting diode, which is used as electric lighting and status indicators on electronic devices.



Zener Diode

Zener Diode is a type of diode, that allows the flow of current in a forward direction, and it can also work in reverse conditions, the Zener diode has an application in voltage regulation, The Zener diode is a heavily doped p-n junction diode made to work in



reverse bias condition.

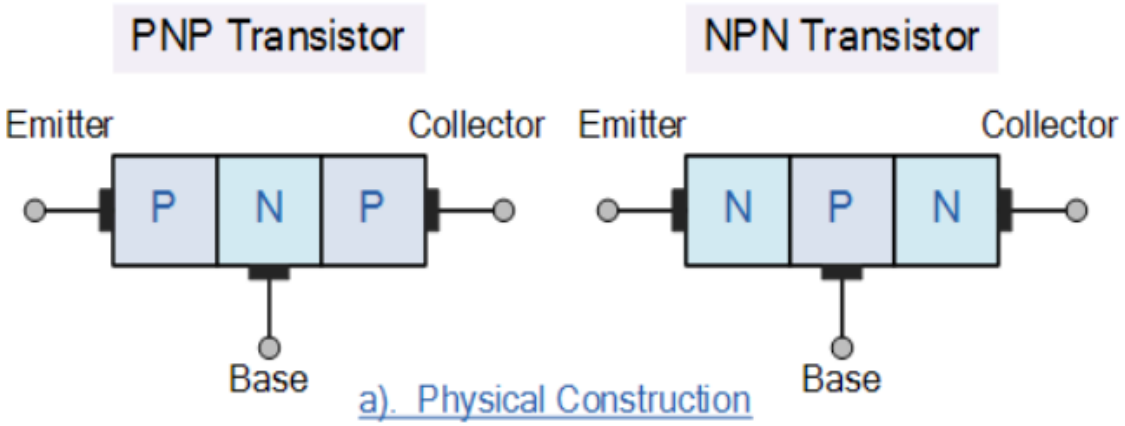
Transistor

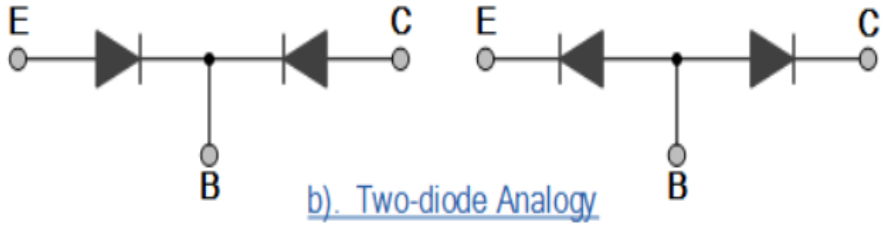
A transistor is a semiconductor device used to amplify or switch electrical signals and power.The word Transistor is a combination of the two words Transfer Varistor which describes their mode of operation way back in their early days of electronics development. There are two basic types of bipolar transistor construction, PNP and NPN, which basically describes the physical arrangement of the P-type and N-type semiconductor materials from which they are made.

The Bipolar Transistor basic construction consists of two PN-junctions producing three connecting terminals with each terminal being given a name to identify it from the other two. These three terminals are known and labelled as the Emitter (E), the Base (B) and the Collector (C) respectively.

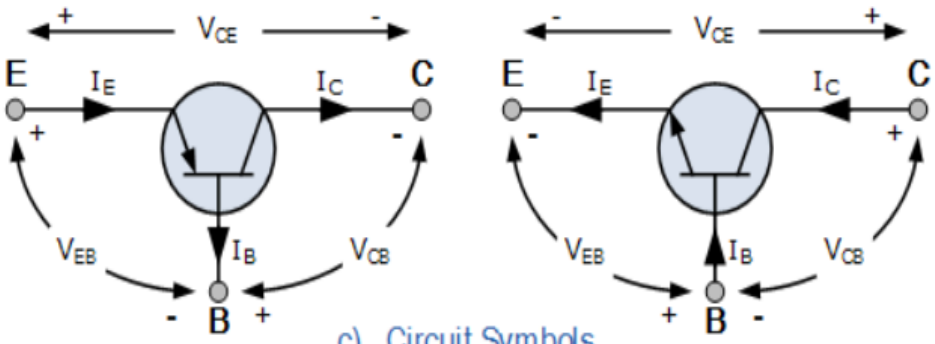
Bipolar Transistors are current regulating devices that control the amount of current flowing through them from the Emitter to the Collector terminals in proportion to the amount of biasing voltage applied to their base terminal, thus acting like a current-controlled switch. As a small current flowing into the base terminal controls a much larger collector current forming the basis of transistor action.

Bipolar Transistor Construction





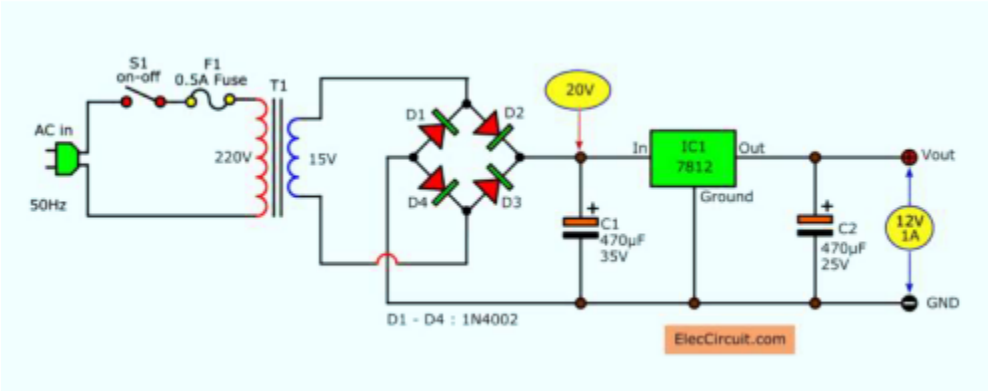
b). Two-diode Analogy



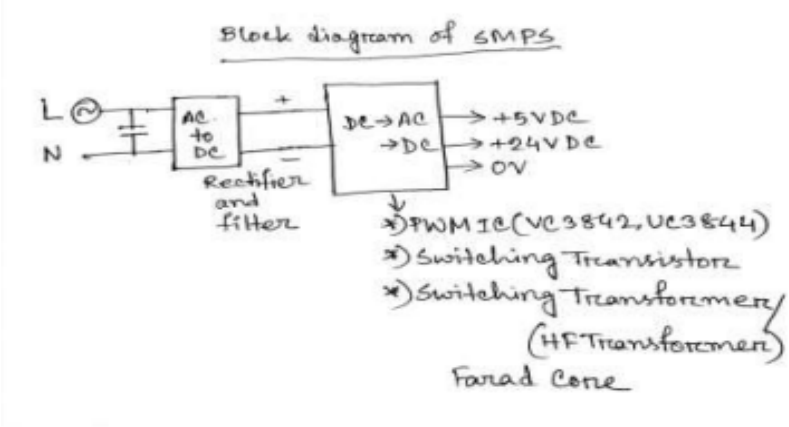
c). Circuit Symbols

Power supply:

Linear PS: T1 is silicon grade steal core transformer .



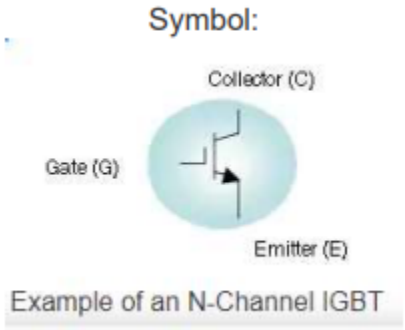
Non Linear PS: Block diagram of SMPS.



Power Electronics:

IGBT:

IGBT stands for insulated-gate bipolar transistor. It is a bipolar transistor with an insulated gate terminal. The IGBT combines, in a single device, a control input with a MOS structure and a bipolar power transistor that acts as an output switch. IGBTs are suitable for high-voltage, highcurrent applications. They are designed to drive high-power applications with a low-power input.



Example of an N-Channel IGBT

SCR:

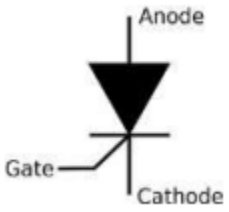
What is Silicon Controlled Rectifier?

- Silicon controlled rectifier is a unidirectional current controlling device. Just like a normal p-n junction diode, it allows electric current in only one direction and

blocks electric current in another direction. A normal p-n junction diode is made of two semiconductor layers namely P-type and Ntype.

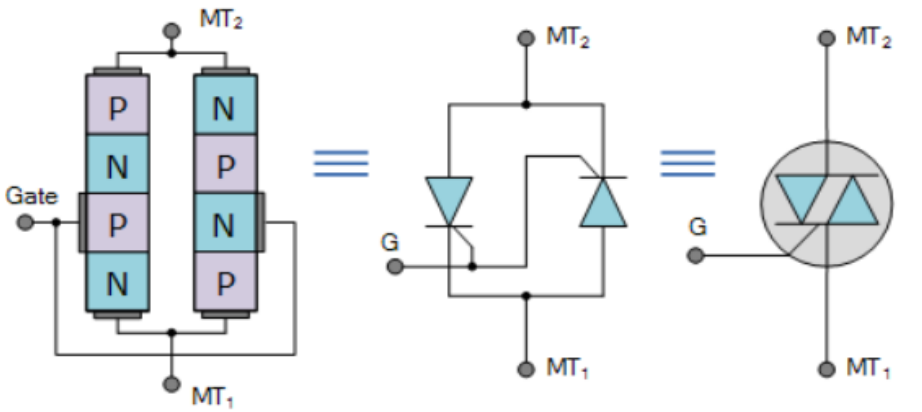
Silicon Controlled Rectifier Symbol

The schematic symbol of a silicon-controlled rectifier is shown in the below figure. A SCR diode consists of three terminals namely anode (A), cathode (K), Gate (G). The diode arrow represents the direction of conventional current.



TRIAC:

A TRIAC behaves just like two conventional thyristors connected together in inverse parallel (back-to-back) with respect to each other and because of this arrangement the two thyristors share a common Gate terminal all within a single three-terminal package.it's converts AC to AC with variable.



Physical Construction Two-Thyristor Analogy Circuit Symbol

SSR: SOLID STATE RELAY

A solid-state relay (SSR) is an electronic switching device that switches on or off when an external voltage (AC or DC) is applied across its control terminals. They serve the same function as an electromechanical relay, but solid-state electronics contain no moving parts and have a longer operational lifetime. It's converts AC to AC and AC to DC with variable.

Analog Output Drive: Output signal: 4mA-20mA, 0-10v

1. Motor Drive:

Digital electronics: There are three types of digital electronics, which available in the market.

- TTL ---74xx, faster, power-7v dc .40drgree
- CMOS-40XX, Slow, power-18v dc
- Military-54XX, power-5v dc,80 degree

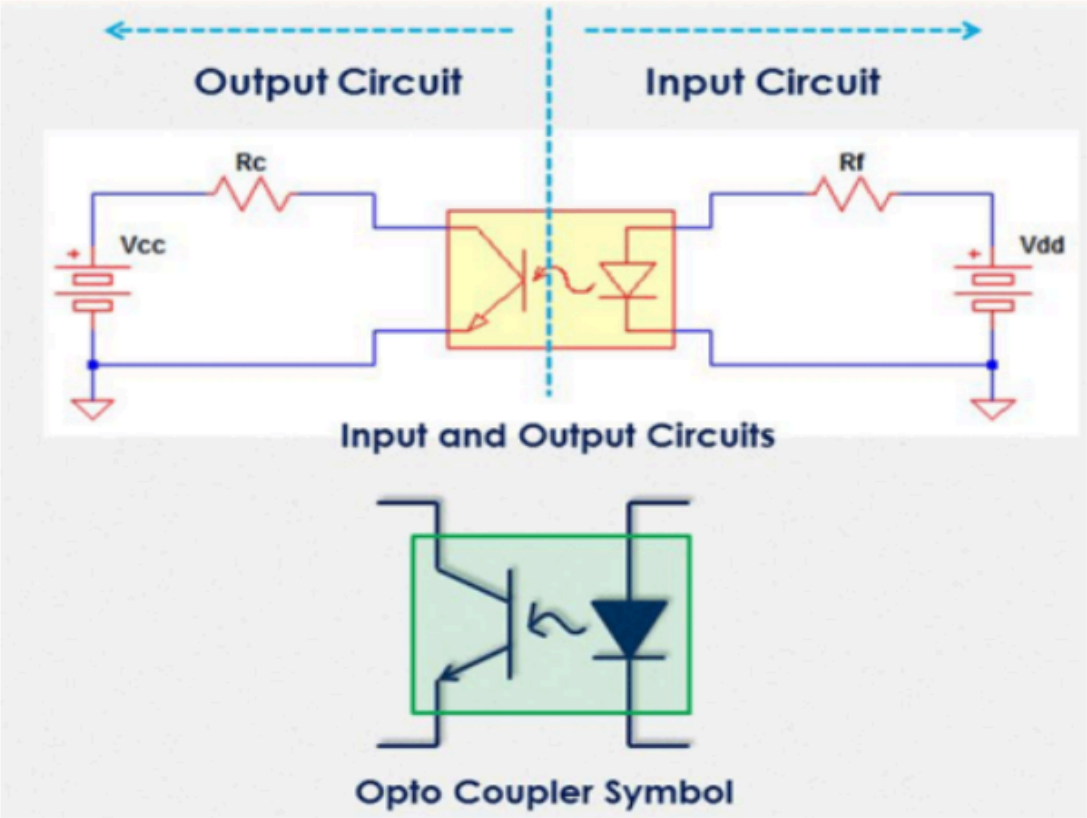
TTL series IC with the number, 7400, 74LS00, 74L00, 7400, 74 H00, 74HC00, 74HCT00

CMOS: IC 4003 , Military: IC 5400

Optocoupler:

Opto-coupler is an electronic component that is used to conduct the electrical signals from one circuit to another circuit without directly being connected between them. In

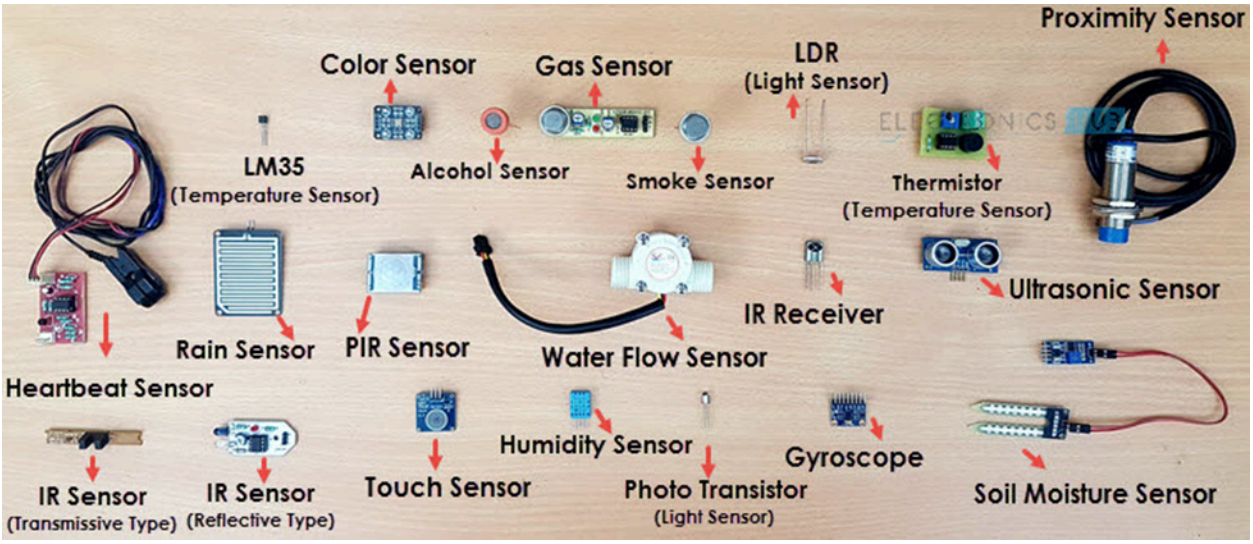
other words, an optocoupler is used to transfer electrical signals between two circuits optically. Here both circuit is electrically isolated from each other. Opto-coupler is also called photocoupler, opto-isolator or optical isolator.



Its function:

- 5v DC to 24 volts
- 5v DC to 220 AC
- 5V DC to 220 AC variables
- 220 to 5v to CPU signal
- IC Model 923, pc929

Sensor And Module



What is a Sensor?

There are numerous definitions as to what a sensor is but I would like to define a Sensor as an input device which provides an output (signal) with respect to a specific physical quantity (input).

The term “input device” in the definition of a Sensor means that it is part of a bigger system which provides input to a main control system (like a Processor or a Microcontroller).

Another unique definition of a Sensor is as follows: It is a device that converts signals from one energy domain to electrical domain. The definition of the Sensor can be better understood if we take an example into consideration.

Classification of Sensors

There are several classifications of sensors made by different authors and experts. Some are very simple and some are very complex. The following classification of sensors may already be used by an expert in the subject but this is a very simple classification of sensors.

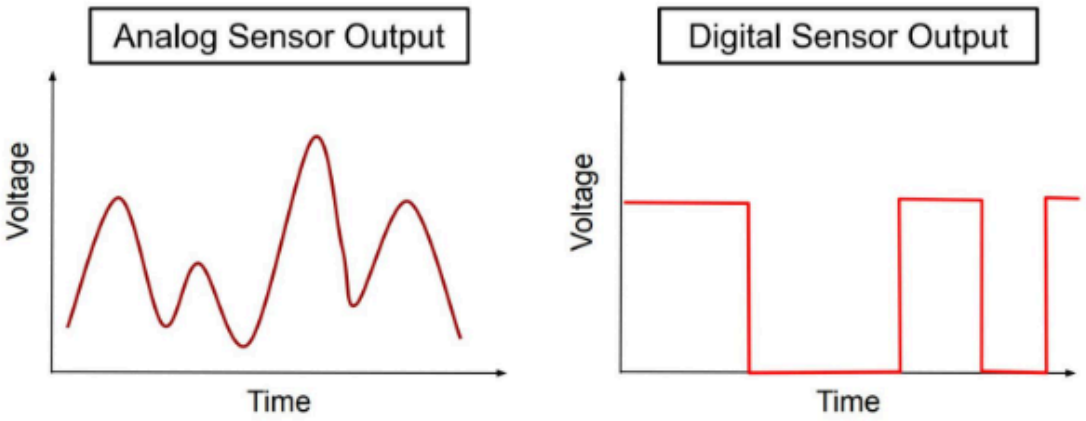
In the **first classification** of the sensors, they are divided into Active and Passive.

- **Active Sensors** are those which require an external excitation signal or a power signal.
- **Passive Sensors**, on the other hand, do not require any external power signal and directly generate output response.

The other type of classification is based on the means of detection used in the sensor. Some of the means of detection are Electric, Biological, Chemical, Radioactive etc.

The next classification is based on conversion phenomenon i.e., the input and the output. Some of the common conversion phenomena are Photoelectric, Thermoelectric, Electrochemical, Electromagnetic, Thermooptic, etc.

The **final classification** of the sensors are **Analog** and **Digital Sensors.**



Analog vs Digital Sensor Output

A *digital sensor* produces a discrete output or binary (0 or 1). These days digital sensors are more popular compared to analog sensors.

Analog sensors produce a continuous output that is proportional to the measured parameter. Examples of analog sensors are accelerometers, pressure sensors, light sensors, and temperature sensors.

Type of Sensor

Temperature Sensor:

Here are some different types of temperature sensors based on their underlying principles of operation and sensing technology.

- **Thermocouples:** Thermocouples are based on the Seebeck effect, where the voltage is generated in a closed loop of two dissimilar metals. They are widely used due to their wide temperature range and durability.
- **RTD (Resistance Temperature Detector):** RTDs are made of pure platinum and their resistance changes with temperature. They are known for their high accuracy and stability.
- **Thermistors:** Thermistors are semiconductor devices with a resistance that

changes significantly with temperature. They are commonly used in consumer electronics and have a high sensitivity.

- **Infrared (IR) Sensors:** IR sensors measure temperature by detecting the electromagnetic radiation emitted from an object. They are commonly used for non-contact temperature measurements.
- **IC Temperature Sensors:** These are integrated circuits that include temperature sensors. They are often used in microcontrollers and other digital systems for temperature monitoring.

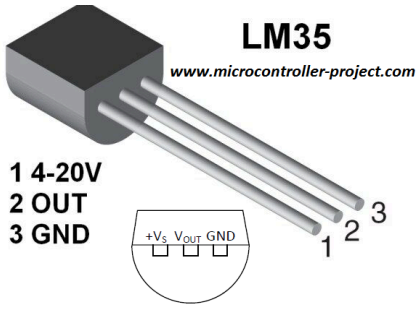
Here are some commonly used temperature sensors compatible with Microcontroller like Arduino ,Esp,STM:

- **LM35:** LM35 is a temperature sensor that outputs an analog signal which is proportional to the instantaneous temperature. The output voltage can easily be interpreted to obtain a temperature reading in Celsius. The advantage of lm35 over thermistor is it does not require any external calibration.

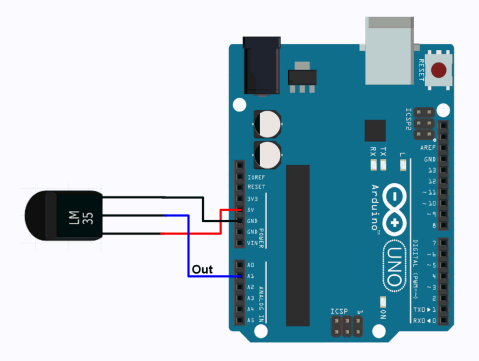
  Feature:

  - Linear + 10-mV/°C Scale Factor
  - 0.5°C Ensured Accuracy (at 25°C)
  - Rated for Full −55°C to 150°C Range

  Figure:

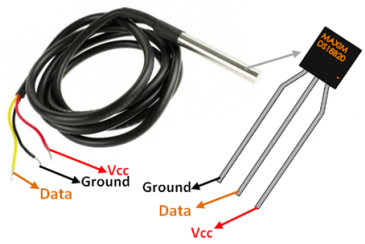


Connection With Arduino:

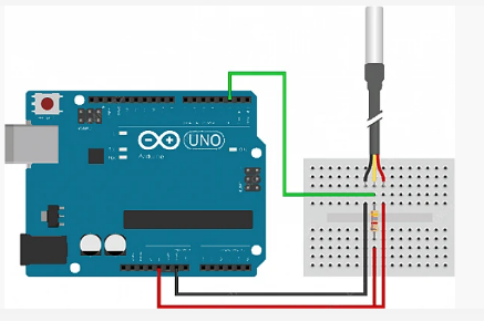


- **DS18B20:** The DS18B20 is a small temperature sensor with a built in 12bit ADC. It can be easily connected to an Arduino digital input. The sensor communicates over a one-wire bus and requires little in the way of additional components. The sensors have a quoted accuracy of +/-0.5 deg C in the range -10 deg C to +85 deg C.

  Figure:



Connection with Arduino:



PT100: Often resistance thermometers are generally called Pt100 sensors, even though in reality they may not be the RTD Pt100 type. Pt refers to the sensor being made from Platinum (Pt). 100 refers to that at 0°C the sensor has a resistance of 100 ohms (Ω). A resistance thermometer is a type of temperature sensor.

Figure:

- **Thermistor:**

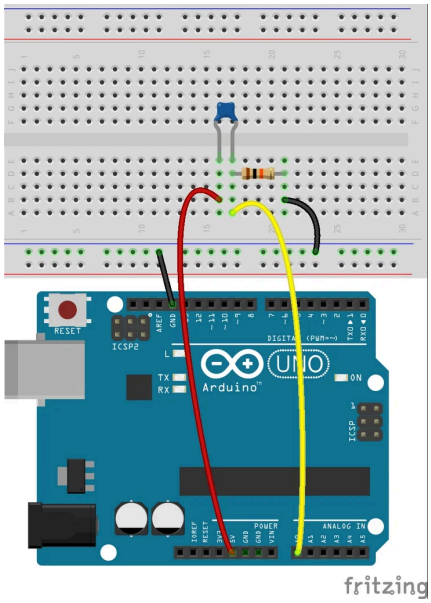
  NTC:NTC stands for "Negative Temperature Coefficient". NTC thermistors are resistors with a negative temperature coefficient, which means that the resistance decreases with increasing temperature. They are primarily used as resistive temperature sensors and current-limiting devices.

  PTC: The PTC thermistor is one type of resistor including a positive temperature coefficient, which means the temperature will be increased once the resistance increases. So the main relationship between resistance (R) & temperature (T) is

linear, as expressed in the below equation.

Figure:

Connection with Arduino:

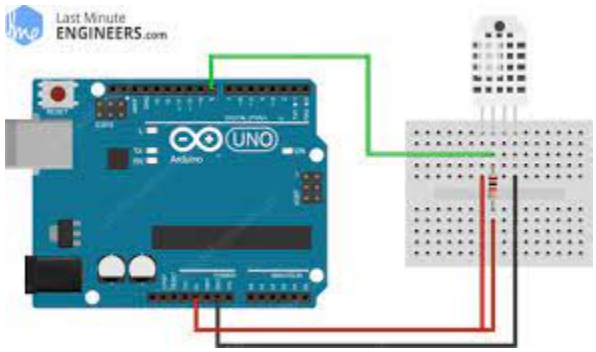


➢ DHT11/22:

These are digital temperature and humidity sensors that use a single-wire digital

communication protocol. They are easy to use and provide temperature and humidity readings.The DHT sensors are made of two parts, a capacitive humidity sensor and a thermistor. There is also a very basic chip inside that does some analog to digital conversion and spits out a digital signal with the temperature and humidity.

DHT11		DHT22
0 - 50°C / ± 2°C	Temperature Range	-40 - 125 °C / ± 0.5 °C
20 - 80% / ± 5%	Humidity Range	0 - 100 % / ± 2-5%
1Hz (one reading every second)	Sampling Rate	0.5 Hz (one reading every two seconds)
15.5mm x 12mm x 5.5mm	Body Size	15.1mm x 25mm x 7.7mm
3 - 5V	Operating Voltage	3 - 5V
2.5mA	Max Current During Measuring	2.5mA



➢ Ultrasonic Sensor (e.g., HC-SR04):

The HC-SR04 Ultrasonic Sensor is a versatile device used for distance measurement and object detection. It operates by emitting ultrasonic waves and measuring the time it takes for the waves to bounce back from an object, allowing it to accurately determine distances within a range of 2 cm to 400 cm. Interfacing with Arduino or ESP microcontrollers is straightforward, making it a popular choice for a wide range of applications, from obstacle avoidance in robotics to water level monitoring. The sensor offers good accuracy, low power requirements, and is equipped with mounting holes for easy installation, making it a valuable tool for various projects.

➢ Pressure Sensor (e.g., BMP180, BMP280):

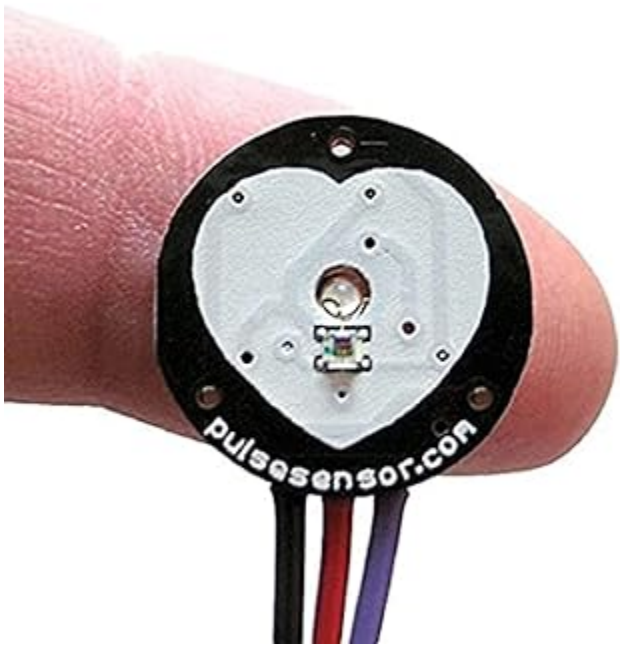
A Pressure Sensor, like the BMP180 or BMP280, is a device designed to measure

atmospheric pressure and, in some cases, altitude. These sensors work based on the piezoresistive principle, where changes in pressure cause variations in the resistance of internal elements. The BMP180 and BMP280 sensors are often used for weather monitoring and environmental data collection. When interfacing with Arduino or ESP microcontrollers, they can provide accurate pressure and temperature data, enabling applications like weather stations, altitude measurement, and even indoor navigation systems. These sensors are known for their high precision, low power consumption, and ease of integration due to available libraries and documentation in the maker and IoT communities.

➢ Heart Rate Sensor:

A Heart Rate Sensor is a specialized device used to measure a person's heart rate or pulse rate in real-time. It typically works by detecting the subtle changes in blood

volume in the skin caused by each heartbeat. Heart rate sensors can be optical or electrical. Optical heart rate sensors often use LEDs to shine light into the skin and photodetectors to measure the amount of light that is absorbed or reflected by blood vessels. Electrical sensors, like ECG (Electrocardiogram) sensors, directly measure the electrical activity of the heart. These sensors are commonly used in wearable fitness devices, medical equipment, and IoT applications to monitor an individual's heart rate for health and fitness tracking. They can be interfaced with Arduino, ESP, or other microcontrollers to integrate heart rate data into various projects or applications, helping users keep track of their cardiovascular health.



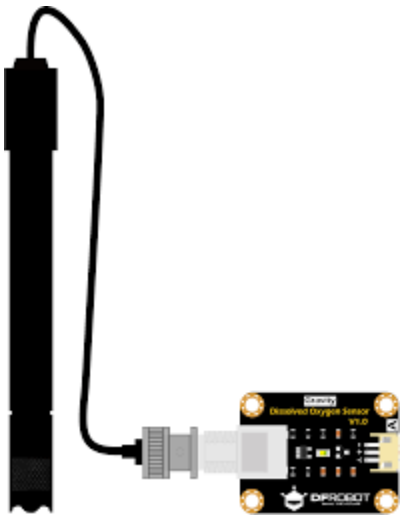
➢ Analog Dissolved Oxygen Sensor:

This is a dissolved oxygen sensor kit. A dissolved oxygen meter is used to measure the dissolved oxygen in water, to reflect the water quality. It is widely applied in many water

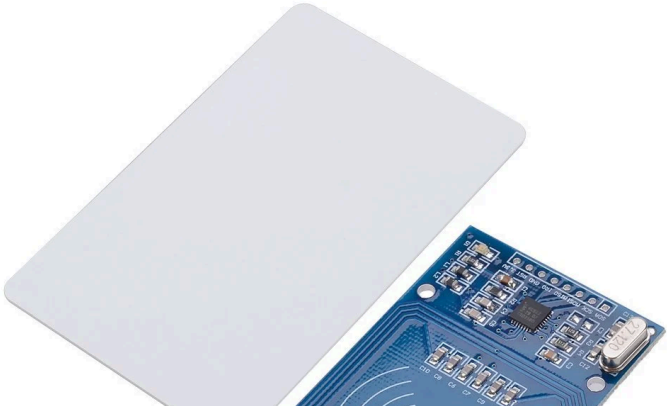
quality applications, such as aquaculture, environment monitoring, natural science, etc.

There's an old saying regarding keeping fish, "Good fish deserves good water". Good water quality is very important to aquatic organisms. Dissolved oxygen is one of the important parameters to reflect water quality. Low dissolved oxygen in the water will lead to difficulty in breathing for aquatic organisms, which may threaten their lives.

The probe is galvanic, does not need polarization time, and stays available at any time. The filling solution and membrane cap are replaceable, leading to the low maintenance cost. The signal converter board is plug-and-play and works from 3.3V - 5V which makes it compatible with most popular microcontrollers such as ESP32, Raspberry Pi and Arduino.



➢ **RFID Reader:**



The reader is a device that has one or more antennas that emit radio waves and receive signals back from the RFID tag. Tags, which use radio waves to communicate their identity and other information to nearby readers, can be passive or active. Passive RFID tags are powered by the reader and do not have a battery.

➢ GPS Module:

GPS receivers are generally used in smartphones, fleet management system, military etc. for tracking or finding location.Global Positioning System (GPS) is a satellite-based system that uses satellites and ground stations to measure and compute its position on Earth.GPS is also known as Navigation System with Time and Ranging (NAVSTAR) GPS.GPS receiver needs to receive data from at least 4 satellites for accuracy purpose. GPS receiver does not transmit any information to the satellites.

This GPS receiver is used in many applications like smartphones, Cabs, Fleet management etc.

➢ Laser Distance Sensor:

Laser distance sensors measure distances and allow it to take measurements at great distances. These distance sensors work on the basis of the Time-Of-Flight (ToF) principle, which means that the sensor emits a laser beam and receives the reflection from it.

➢ Load Cell:

A load cell measures mechanical force, mainly the weight of objects. Today, almost all electronic weighing scales use load cells because of the accuracy with which they can measure weight. Load cells find their application in a variety of fields that demand accuracy and precision.

➢ Barometric Sensor(HX710B):

Biometric sensors are a mechanical or electronic technology that captures biometric data (i.e. the face, palm print, or iris) digitally in a way that can be turned into a biometric template. So for example, the biometric sensor for the face is a device's camera. For fingerprints, it is a fingerprint pad/sensor.

➢ Current Sensor(ACS712):

A current sensor detects and measures the electric current passing through a conductor. It turns the current into a quantifiable output, such as a voltage, current, or digital signal, which may be utilised in a variety of applications for monitoring, control, or protection.

➢ Water Flow Sensor:

A water flow sensor is a specific type of flow sensor designed to measure the flow rate of water. It finds applications in various sectors, including plumbing, irrigation systems, water treatment plants, and industrial processes where water flow monitoring is crucial.

➢ Accelerometer:

Accelerometer sensors are ICs that measure acceleration, which is the change in speed (velocity) per unit time. Measuring acceleration makes it possible to obtain information such as object inclination and vibration.

➢ Gyroscope Sensor:

Gyroscope sensor is a device that can measure and maintain the orientation and angular velocity of an object. These are more advanced than accelerometers. These can measure the tilt and lateral orientation of the object whereas an accelerometer can only measure the linear motion.

Gyroscope sensors are also called Angular Rate Sensors or Angular Velocity Sensors. These sensors are installed in the applications where the orientation of the object is difficult to sense by humans

➢ Moisture Sensor:

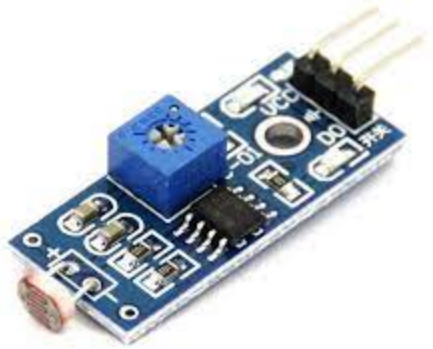
The sensor creates a voltage proportional to the dielectric permittivity, and therefore the water content of the soil.Moisture Sensor uses capacitance to measure dielectric permittivity of the surrounding medium. In soil, dielectric permittivity is a function of the water content.

➢ Vibration Sensor (e.g., SW420):

A vibration sensor is a device that detects mechanical vibrations. It measures the vibration levels in your machine and alerts you to any potential problems, like equipment failure or worn parts that need replacement.

➢ Light Sensor (e.g., LDR):

Light sensors are a type of photodetector (also called photosensors) that detect light. Different types of light sensors can be used to measure illuminance, respond to changes in the amount of light received, or convert light to electricity.

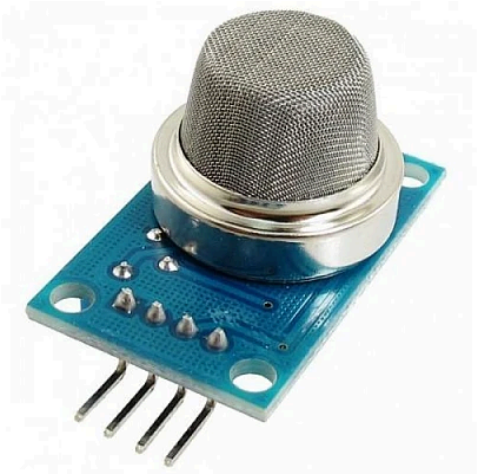


➢ Infrared Sensor(IR):

An infrared sensor (IR sensor) is a radiation-sensitive optoelectronic component with a spectral sensitivity in the infrared wavelength range 780 nm … 50 µm. IR sensors are now widely used in motion detectors, which are used in building services to switch on lamps or in alarm systems to detect unwelcome guests.

➢ Gas Sensor:

A gas sensor is a device which detects the presence or concentration of gases in the atmosphere. Based on the concentration of the gas the sensor produces a corresponding potential difference by changing the resistance of the material inside the sensor, which can be measured as output voltage. Based on this voltage value the type and concentration of the gas can be estimated.



➢ PIR Motion Sensor:

A PIR sensor is an electronic sensor used in motion detectors such as automatically triggered lighting devices and protection systems that measure devices emitting infrared light in their field of view. Each body with a temperature above zero releases heat energy, which is in the form of radiation.

➢ Sound Sensor:

The sound sensor is a module that monitors and detects the sound signals like voice, claps, snaps, knocks, etc. It is also known as an acoustic sensor or sound detector. Used in various applications such as security systems, monitoring systems, radios, telephones, mobile phones, computers, home automation systems, consumer electronic appliances, etc.

➢ Color Sensor(tcs3200):

Colour sensors are employed to recognise/detect the colour of a material in RGB (red, green, blue) scale, while rejecting the unwanted infrared or ultraviolet light. The ultimate challenge with colour sensing has been to detect subtle differences among similar or highly reflective surfaces.

➢ Flame Sensor:

A sensor which is most sensitive to a normal light is known as a flame sensor. That's why this sensor module is used in flame alarms. This sensor detects flame otherwise wavelength within the range of 760 nm – 1100 nm from the light source. This sensor can be easily damaged to high temperatures.

So this sensor can be placed at a certain distance from the flame. The flame detection can be done from a 100cm distance and the detection angle will be 600. The output of this sensor is an analog signal or digital signal. These sensors are used in fire fighting robots like a flame alarm.

➢ Raindrop Sensor:

The Raindrop sensors are used for the detection of rain and also for measuring rainfall intensity. Raindrop sensors can be used for all kinds of weather monitoring. A raindrop sensor is a board on which nickel is coated in the form of lines. It works on the principle of resistance.

➢ Touch Sensor:

A touch sensor is a type of device that captures and records physical touch or embrace on a device and/or object. It enables a device or object to detect touch or near proximity, typically by a human user or operator.

➢ Flex Sensor:

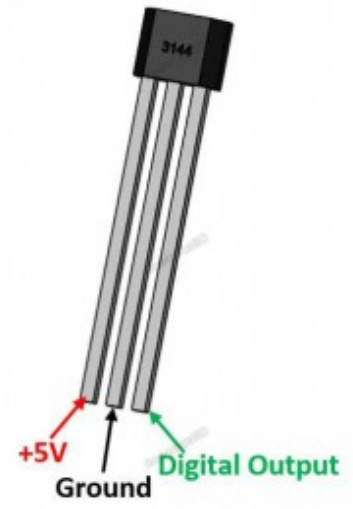
A flex sensor is a kind of sensor which is used to measure the amount of defection otherwise bending. The designing of this sensor can be done by using materials like plastic and carbon. The carbon surface is arranged on a plastic strip as this strip is turned aside then the sensor's resistance will be changed. Thus, it is also named a bend sensor. As its varying resistance can be directly proportional to the quantity of turn thus it can also be employed like a goniometer.

➢ pH Sensor:

pH sensor is one of the most important tools for measuring pH and is commonly used in water quality monitoring. This type of sensor is capable of measuring alkalinity and acidity in water and other solutions. When used properly, pH sensors can ensure the safety and quality of products and processes that occur in wastewater or manufacturing plants

➢ Hall Effect Sensor:

Hall Effect Sensors are devices which are activated by an external magnetic field. We know that a magnetic field has two important characteristics flux density, (B) and polarity (North and South Poles).



The output signal from a Hall effect sensor is the function of magnetic field density around the device. When the magnetic flux density around the sensor exceeds a certain pre-set threshold, the sensor detects it and generates an output voltage called the Hall Voltage, VH.

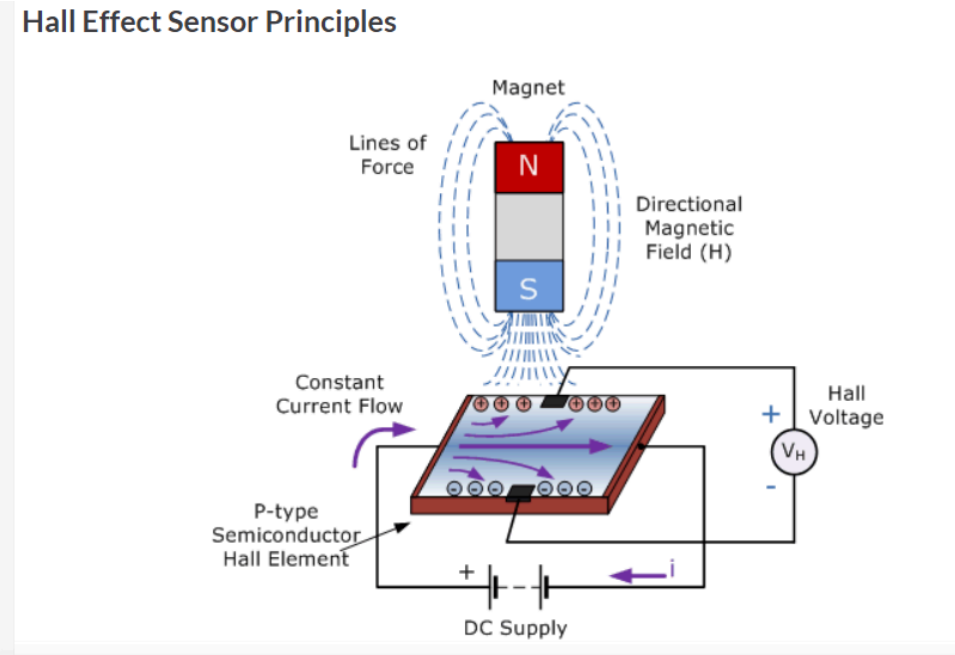
Magnetic sensors convert magnetic or magnetically encoded information into electrical signals for processing by electronic circuits, and in the Sensors and Transducers tutorials we looked at inductive proximity sensors and the LDVT as well as solenoid and relay output actuators.

Magnetic sensors are solid state devices that are becoming more and more popular because they can be used in many different types of application such as sensing position, velocity or directional movement.

They are also a popular choice of sensor for the electronics designer due to their non-contact wear free operation, their low maintenance, robust design and as sealed hall effect devices are immune to vibration, dust and water.

One of the main uses of magnetic sensors is in automotive systems for the sensing of position, distance and speed. For example, the angular position of the crank shaft for the firing angle of the spark plugs, the position of the car seats and seat belts for air-bag control or wheel speed detection for the anti-lock braking system, (ABS).

Magnetic sensors are designed to respond to a wide range of positive and negative magnetic fields in a variety of different applications and one type of magnet sensor whose output signal is a function of magnetic field density around it is called the Hall Effect Sensor.



Hall Effect Sensors consist basically of a thin piece of rectangular p-type semiconductor material such as gallium arsenide (GaAs), indium antimonide (InSb) or indium arsenide (InAs) passing a continuous current through itself.

When the device is placed within a magnetic field, the magnetic flux lines exert a force on the semiconductor material which deflects the charge carriers, electrons and holes, to either side of the semiconductor slab. This movement of charge carriers is a result of the magnetic force they experience passing through the semiconductor material.

➢ Magnetic Reed Switch:

A magnetic reed switch is a type of electrical switch that operates based on the presence or absence of a magnetic field. It consists of two thin, ferrous (iron-containing) reed elements encased in a protective glass or plastic tube. The reed elements are typically shaped like flattened, elongated blades.

When a magnetic field is applied to the proximity of the reed switch, the magnetic forces cause the reed elements to come together and make electrical contact. This closes the circuit and allows current to flow through the switch. Conversely, when the magnetic

field is removed, the reed elements spring back to their original position, breaking the circuit and interrupting the flow of current.

Magnetic reed switches are commonly used in various applications such as door and window sensors in security systems, proximity sensors, and in other devices where the detection of a magnetic field is needed to control the electrical circuit. They are valued for their simplicity, reliability, and low power consumption.

➢ Thermal Imaging Sensor:

A thermal imaging sensor is a device capable of detecting and capturing infrared radiation emitted by objects in the form of heat. Unlike traditional cameras that capture visible light, thermal imaging sensors operate in the infrared spectrum, allowing them to visualize temperature differences and produce images based on thermal emissions.

These sensors use the heat signatures of objects to create a thermal image or thermogram, where warmer areas appear brighter or with different colors than cooler areas. The technology is based on the fact that all objects with a temperature above absolute zero (-273.15 degrees Celsius or -459.67 degrees Fahrenheit) emit infrared

radiation.

➢ Joystick:

A joystick is an input device consisting of a handheld stick that pivots on a base, allowing users to control the movement of an object or cursor on a screen by manipulating the stick in various directions.

➢ Infrared Receiver (IR Receiver):

An infrared receiver is a device that detects and receives infrared signals, commonly used in remote controls to wirelessly transmit commands to electronic devices such as TVs, audio systems, or other appliances.

➢ TDS Sensor:

Total Dissolved Solids (TDS) sensor measures the concentration of dissolved substances in a liquid, providing an indication of water quality by detecting the presence of minerals, salts, and other dissolved particles.

➢ Turbidity Sensor:

A turbidity sensor measures the cloudiness or haziness of a fluid caused by suspended particles. It is commonly used in water quality monitoring to assess the clarity of liquids.

➢ UV Index Sensor:

A UV index sensor measures the intensity of ultraviolet (UV) radiation from the sun, helping to assess the potential harm to human skin and providing information for sun protection.

➢ Hall Effect Proximity Sensor:

A Hall Effect proximity sensor detects the presence or absence of a magnetic field using the Hall Effect principle, making it useful for applications such as position sensing or detecting the opening and closing of covers.

➢ Gesture Recognition Sensor:

A gesture recognition sensor interprets human gestures or movements as commands for electronic devices. It is often used in user interfaces, gaming consoles, and smart devices.

➢ Dust Sensor:

A dust sensor measures the concentration of particulate matter in the air, providing information about air quality by detecting the presence of fine particles.

➢ Wind Speed Sensor:

A wind speed sensor measures the velocity of the wind, providing valuable data for weather monitoring, environmental studies, and various applications.

➢ Lidar Sensor:

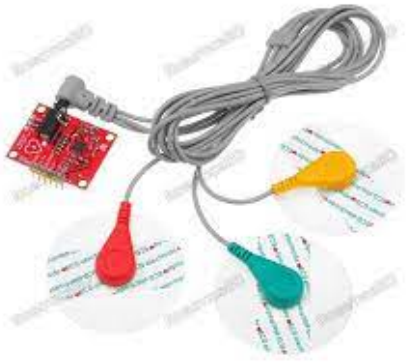
Lidar (Light Detection and Ranging) sensors use laser light to measure distances and create detailed, three-dimensional maps of the surrounding environment. They are commonly used in autonomous vehicles and mapping applications.

➢ Biometric Fingerprint Sensor:

A biometric fingerprint sensor captures and analyzes fingerprint patterns for personal identification and authentication purposes, enhancing security in devices such as smartphones and access control systems.

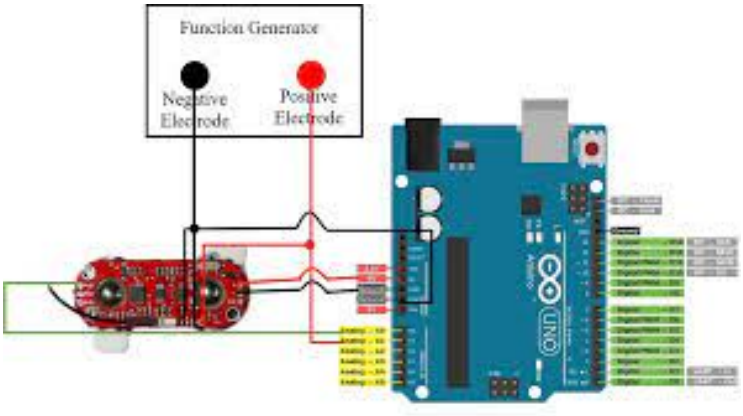
➢ ECG Sensor:

An Electrocardiogram (ECG) sensor measures the electrical activity of the heart, providing information about heart rate and rhythm for medical monitoring and diagnostics.



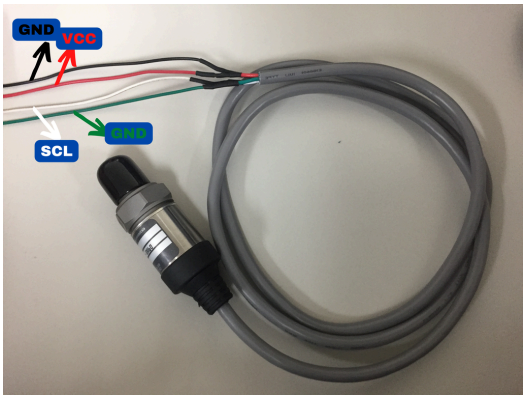
➢ EMG Sensor:

An Electromyography (EMG) sensor measures the electrical activity of muscles, providing information about muscle contractions and activity for applications in medical diagnostics and biomechanics.



➢ Gas Pressure Sensor:

A gas pressure sensor measures the pressure of gases in a system, providing valuable data for applications such as industrial processes, environmental monitoring, and gas appliances.



➢ Doppler Radar Motion Sensor:

A Doppler radar motion sensor detects movement by analyzing changes in the frequency of radar waves reflected off moving objects. It is commonly used in security systems and automatic door openers.

➢ Soil pH Sensor:

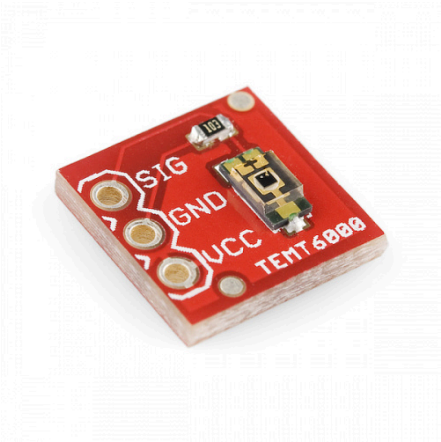
A soil pH sensor measures the acidity or alkalinity of soil, providing information about soil health and suitability for plant growth.

➢ Proximity Sensor:

A proximity sensor detects the presence or absence of an object within its vicinity without physical contact. It is often used in smartphones, elevators, and robotics.

➢ Ambient Light Sensor:

An ambient light sensor measures the level of light in the surrounding environment, allowing devices to adjust screen brightness automatically for optimal visibility.



➢ Metal Detector Sensor:

A metal detector sensor detects the presence of metallic objects in the proximity, commonly used in security screening and treasure hunting.

➢ Thermal Sensor:

A thermal sensor measures temperature by detecting infrared radiation emitted by an object, finding applications in various industries for temperature monitoring.

➢ Force Sensor:

A force sensor measures the force applied to it, providing information about pressure, tension, or compression in various applications such as industrial processes and robotics.

➢ Capacitive Touch Sensor:

A capacitive touch sensor detects touch or proximity by measuring changes in capacitance. It is widely used in touchscreens and touch-sensitive interfaces.

➢ Pressure Transducer:

A pressure transducer converts pressure into an electrical signal, providing precise measurements in applications such as industrial processes, automotive systems, and medical devices.

➢ Reflective Optical Sensor:

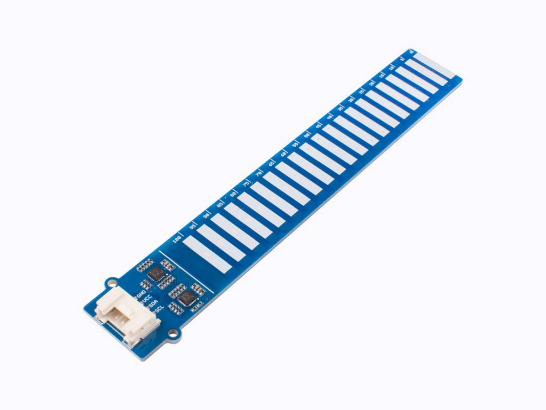
A reflective optical sensor uses light reflection to detect the presence or absence of an object, commonly employed in applications such as object detection and counting.

➢ Pulse Oximeter Sensor:

A pulse oximeter sensor measures the oxygen saturation level in blood by analyzing the absorption of light, providing valuable information for healthcare monitoring.

➢ Water level sensor:

A water level sensor detects and measures the depth of liquid in a container or reservoir, commonly used in applications such as water tanks and industrial processes.



➢ Floating Sensor:

A floating sensor detects the presence or absence of a floating object on a liquid surface, often used in applications such as water level monitoring and alarms.

➢ Limit Sensor:

A limit sensor indicates when a physical limit or boundary has been reached, commonly used in machinery and automation to control the movement of mechanical components.

These sensors cover a wide range of applications, from environmental monitoring to medical devices, industrial automation, and more. The choice of sensor depends on the specific needs

Introduction to ESP8266

ESP8266

The ESP8266 is a low-cost, programmable Wi-Fi module that integrates a microcontroller and Wi-Fi capability. It gained popularity for its affordability and capability to connect to the internet, making it a popular choice for IoT projects, home automation, and various wireless applications.

Arduino IDE:

The Arduino IDE is a user-friendly software environment used to program Arduino microcontrollers. It simplifies the process of writing, compiling, and uploading code to Arduino boards, making it accessible for both beginners and experienced developers.

Integration

When using the ESP8266 with the Arduino IDE, developers can write and upload Arduino-compatible sketches (code) to the ESP8266 module. This allows them to leverage the extensive Arduino libraries and community support while benefiting from the ESP8266's Wi-Fi capabilities. The integration simplifies the development of projects involving wireless communication, sensor data collection, and remote control.

Advantages

- **Ease of Use:** The Arduino IDE provides a familiar and easy-to-use platform for programming the ESP8266, even for those new to IoT development.
- **Community Support:** The extensive Arduino community and wealth of libraries enhance the ESP8266 development experience, offering pre-written code for common tasks.

- **Wi-Fi Capabilities:** Leveraging the ESP8266's built-in Wi-Fi capability allows developers to create connected devices and IoT applications without the need for additional shields or modules.

Introduction to ESP8266 with Arduino IDE

The ESP8266 is a versatile and cost-effective Wi-Fi module that has gained immense popularity in the realm of embedded systems, particularly within the Internet of Things (IoT) community. This module integrates a microcontroller with built-in Wi-Fi capability, making it a powerful tool for creating connected devices and enabling wireless communication.

When paired with the Arduino IDE (Integrated Development Environment), the ESP8266 becomes easily accessible to a broad range of developers, from beginners to experienced makers. The Arduino IDE simplifies the process of programming and developing projects, providing a user-friendly interface and a vast collection of libraries.

Key Features of ESP8266

- **Wi-Fi Connectivity:** The standout feature of the ESP8266 is its ability to connect to Wi-Fi networks, enabling devices to communicate with each other and the internet. This capability is crucial for IoT applications.
- **Low Cost:** The ESP8266 is remarkably affordable, making it an attractive choice for hobbyists, students, and professionals working on projects with budget constraints.
- **Microcontroller Integration**: The module incorporates a microcontroller, allowing developers to execute code locally on the ESP8266. This opens up possibilities for standalone applications without the need for an external microcontroller.
- **Community Support:** The ESP8266 enjoys strong community support, with an active user base contributing to an extensive collection of libraries, tutorials, and projects. This support makes it easier for developers to find solutions to common challenges.

Integrating ESP8266 with Arduino IDE

The Arduino IDE is a well-established environment for programming microcontrollers, known for its simplicity and ease of use. Integrating the ESP8266 with the Arduino IDE allows developers to leverage the familiar Arduino framework while harnessing the power of Wi-Fi connectivity.

Benefits of Using Arduino IDE with ESP8266

- **Code Reusability:** The Arduino IDE offers a vast collection of libraries that can be easily used with the ESP8266, enhancing code reusability and speeding up the development process.
- **Simplified Programming:** The Arduino programming language, based on C/C++, is beginner-friendly, making it accessible to those new to embedded systems. This simplicity is maintained when working with the ESP8266 in the Arduino IDE.
- **Rapid Prototyping:** The combination of the ESP8266 and Arduino IDE allows for quick prototyping of IoT projects, enabling developers to focus on functionality rather than intricate programming details.

In summary, integrating the ESP8266 with the Arduino IDE provides an efficient and approachable platform for developing Wi-Fi-enabled projects. Whether you're a novice or an experienced developer, this combination offers a seamless and productive environment for creating innovative IoT applications and connected devices.

How to setup the Arduino IDE for ESP8299

Setting up the Arduino IDE for ESP8266 involves installing the ESP8266 board package, selecting the appropriate board, and configuring settings.

Step 1: Install Arduino IDE

Ensure that you have the Arduino IDE installed on your computer. If not, download and install the latest version from the official Arduino website.

Step 2: Add ESP8266 Board URL

- Open the Arduino IDE.
- Go to File > Preferences.
- In the "Additional Boards Manager URLs" field, add the following URL: bash"http://arduino.esp8266.com/stable/package_esp8266com_index.json"
- Click OK to close the Preferences window.

Step 3: Install ESP8266 Board Package

- Go to Tools > Board > Boards Manager.
- In the Boards Manager, type "esp8266" into the search bar.
- Find and select "esp8266" by ESP8266 Community. Click on it to highlight.
- Click the Install button.

Step 4: Select ESP8266 Board

- Go to Tools > Board.
- Scroll down and find the section labeled "ESP8266 Boards (2.7.4) or similar" (theversion number may vary).
- Choose your specific ESP8266 board model (e.g., NodeMCU, Wemos D1 Mini)from the list.

Step 5: Configure ESP8266 Settings

- After selecting the board, additional configuration options will appear under Tools > Flash Size, Flash Mode, CPU Frequency, Flash Frequency, Upload Speed, and Port.
- Configure these settings based on the specifications of your ESP8266 board. Refer to the manufacturer's documentation for recommended settings.

Step 6: Install USB Drivers (if required)

If your ESP8266 board requires USB drivers, make sure to install them. Check the manufacturer's website for specific driver information.

Step 7: Verify Setup

- Connect your ESP8266 board to your computer using a USB cable.
- In the Arduino IDE, go to Tools > Port and select the COM port associated with your ESP8266.

Step 8: Test with Blink Example

- Open the "Blink" example sketch: File > Examples > 01.Basics > Blink.
- Make any necessary adjustments in the code (e.g., LED pin number).
- Click the Upload button (right arrow icon) to compile and upload the sketch to your ESP8266.
- Observe the onboard LED (or an external LED connected to the specified pin) to ensure it blinks, indicating a successful upload.

ThingSpeak and IoT Applications

ThingSpeak

ThingSpeak is an IoT (Internet of Things) platform that allows users to collect, analyze, and visualize data from their connected devices.

Creating an Account on ThingSpeak:

To get started, follow these steps to create an account on ThingSpeak:

- Visit the ThingSpeak website (https://thingspeak.com/).
- Click on the "Sign Up" button to create a new account.
- Fill in the required information, including a username, email address, and password.
- Complete the registration process and verify your email address.

Once registered, you gain access to your ThingSpeak account dashboard, where you can create channels to store and analyze data from your IoT devices.

Connect Temperature and Humidity Sensor

To connect a temperature and humidity sensor to ThingSpeak, you need a device capable of reading sensor data and transmitting it to the platform. Commonly, a microcontroller like NodeMCU is used for this purpose. Here's a general outline:

- Connect the temperature and humidity sensor (e.g., DHT11 or DHT22) to the GPIO pins of the NodeMCU.
- Write a program for the NodeMCU to read data from the sensor.

- Use the ThingSpeak API to send the sensor data to your ThingSpeak channel.

Continuously Monitor Sensor Reading through the Internet

Once the hardware is set up, the NodeMCU can continuously monitor the sensor readings and transmit them to ThingSpeak. Here's an overview of the process:

- Program the NodeMCU to read sensor data at regular intervals.
- Establish a Wi-Fi connection on the NodeMCU to connect to the internet.
- Use the ThingSpeak API to send HTTP requests and update your ThingSpeak channel with the latest sensor readings.

This continuous monitoring enables real-time data updates on your ThingSpeak channel, providing insights into environmental conditions.

Generate API and Program NodeMCU

To integrate your NodeMCU with ThingSpeak, follow these steps:

- In your ThingSpeak account, create a new channel for your specific application.
- Obtain the Write API Key from the created channel. This key is used to authorize your NodeMCU to write data to the channel.
- Program your NodeMCU using the Arduino IDE or another compatible development environment.
- Utilize the ThingSpeak library or make HTTP requests directly to update your ThingSpeak channel with the temperature and humidity readings.

By generating an API key and integrating it into your NodeMCU program, you enable seamless communication between the sensor and ThingSpeak, allowing for the continuous transmission of data.

In conclusion, connecting a temperature and humidity sensor to ThingSpeak and programming a NodeMCU for continuous monitoring provides a straightforward approach to building an IoT application. This setup allows you to remotely collect, analyze, and visualize sensor data through the internet, opening up possibilities for various environmental monitoring and automation applications.

BLYNK

Blynk

Blynk is a popular and user-friendly Internet of Things (IoT) platform that allows developers and hobbyists to create interactive and customizable applications for various IoT devices. It simplifies the process of connecting hardware to the internet and enables the creation of smartphone or web-based interfaces to control and monitor IoT devices remotely.

Key Features

Drag-and-Drop Interface:

- Blynk provides a visual, drag-and-drop interface that allows users to quickly design custom interfaces for their IoT projects.

Wide Hardware Support:

- Blynk supports a wide range of hardware platforms, including popular microcontrollers like Arduino, Raspberry Pi, NodeMCU, ESP8266, ESP32, Particle, and more.

Extensive Widget Library:

- Blynk offers a variety of widgets (buttons, sliders, graphs, displays, and more) that users can easily add to their projects to create interactive and dynamic user interfaces.

Cloud Connectivity:

- Blynk uses cloud services to enable communication between IoT devices and mobile applications, making it easy to control and monitor devices from anywhere with an internet connection.

Blynk App:

- The Blynk app is available for both iOS and Android devices, providing a convenient way to control and monitor IoT projects on smartphones or tablets.

Real-Time Communication:

- Blynk facilitates real-time communication between the hardware and the Blynk cloud servers, allowing quick updates and responsiveness in IoT applications.

Energy System:

- Blynk uses an energy system to manage resources. Each widget added to a project consumes a specific amount of energy, and users are allocated a certain amount of free energy. Additional energy can be purchased if needed.

Getting Started with Blynk

Account Creation:

- Users need to create an account on the Blynk platform to get started.

Creating a New Project:

- After signing in, users can create a new project and select the hardware they are using.

Configuring Widgets:

- Add widgets to the project to control and monitor connected devices. Widgets can include buttons, sliders, gauges, displays, and more.

Generating Auth Token:

- Each Blynk project is associated with an authentication token that links the hardware to the Blynk cloud. This token is used in the code flashed to the hardware.

Writing Code:

- Users write code for their hardware platform, incorporating the Blynk library. The code includes the authentication token and instructions for interacting with Blynk widgets.

Uploading Code:

- Upload the code to the hardware device using the respective integrated development environment (IDE) for the chosen platform (e.g., Arduino IDE for Arduino-based devices).

Monitoring and Controlling:

- Open the Blynk app on a mobile device, connect to the project, and start monitoring and controlling the IoT devices through the created interface.

Advanced Blynk Features

Virtual Pins:

- Blynk's virtual pins allow more flexible communication between the hardware and the app, enabling custom data exchange.

Blynk API:

- The Blynk API allows users to send and receive data directly to and from their hardware.

Bridge Widget:

- The Bridge widget allows communication between multiple Blynk devices, enabling complex IoT projects with inter-device communication.

Timer and Notification Widgets:

- Blynk supports timer and notification widgets, allowing users to schedule tasks and receive alerts based on certain conditions.

Use Cases

Home Automation:

- Control smart home devices, lights, and appliances remotely.

Weather Stations:

- Build weather monitoring stations and visualize data on the Blynk app.

IoT Prototyping:

- Quickly prototype and test IoT projects with minimal effort.

Industrial IoT (IIoT):

- Implement remote monitoring and control solutions for industrial applications.

Education:

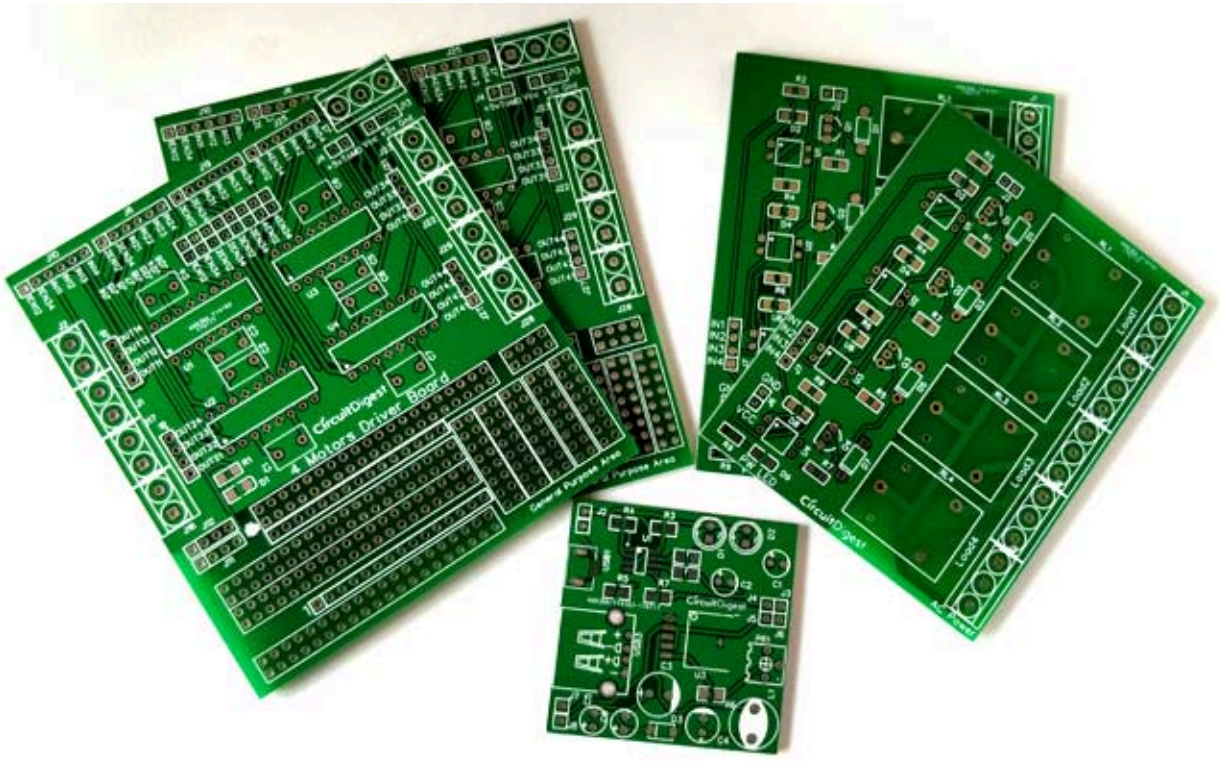
- Blynk is widely used in educational settings to teach IoT concepts and programming.

Blynk simplifies the process of building IoT applications by providing an intuitive platform for creating custom interfaces and connecting hardware devices to the cloud. Its versatility and user-friendly interface make it a popular choice for both beginners and experienced developers in the IoT community

PCB Design

Printed Circuit Board (PCB)

A Printed Circuit Board (PCB) is a flat, rigid, or flexible board made of insulating material that supports and connects electronic components using conductive pathways, traces, or tracks. These pathways are typically copper and are etched onto the surface of the



board in a specific pattern, following the design of the electronic circuit. PCBs are crucial for the assembly and functioning of electronic devices, offering a compact and reliable way to organize and connect various components.

Understanding PCBs involves familiarity with:

- Conceptualization and Schematic Design:

    - Define Requirements: Clearly outline the functionality and specifications of the electronic circuit.
    - Schematic Capture: Use PCB design software to create a schematic diagram that represents the components and their connections in the circuit.
- **Component Selection:**
    - Choose Components: Select electronic components based on the circuit requirements and availability.
    - Check Datasheets: Refer to datasheets to understand the specifications and characteristics of chosen components.
- **Layout Design:**
    - Place Components: Physically arrange components on the PCB, considering factors like signal flow, power distribution, and heat dissipation.
    - Trace Routing: Create electrical connections (traces) between components, adhering to design rules and constraints.
    - Ground and Power Planes: Design ground and power planes to ensure proper signal integrity and power distribution.
- **Layer Stackup Design:**
    - Determine Layers: Decide on the number and order of PCB layers based on design complexity and requirements.
    - Assign Functions to Layers: Allocate specific functions to each layer, such as signal routing, power distribution, and ground planes.
- **Design Rule Check (DRC):**
    - Check Compliance: Use the PCB design software to perform a Design Rule Check to ensure that the layout adheres to specified design rules and constraints.
- **Gerber File Generation:**
    - Generate Gerber Files: Create Gerber files, which contain the information needed for PCB manufacturing, including copper layers, solder mask, and silkscreen.

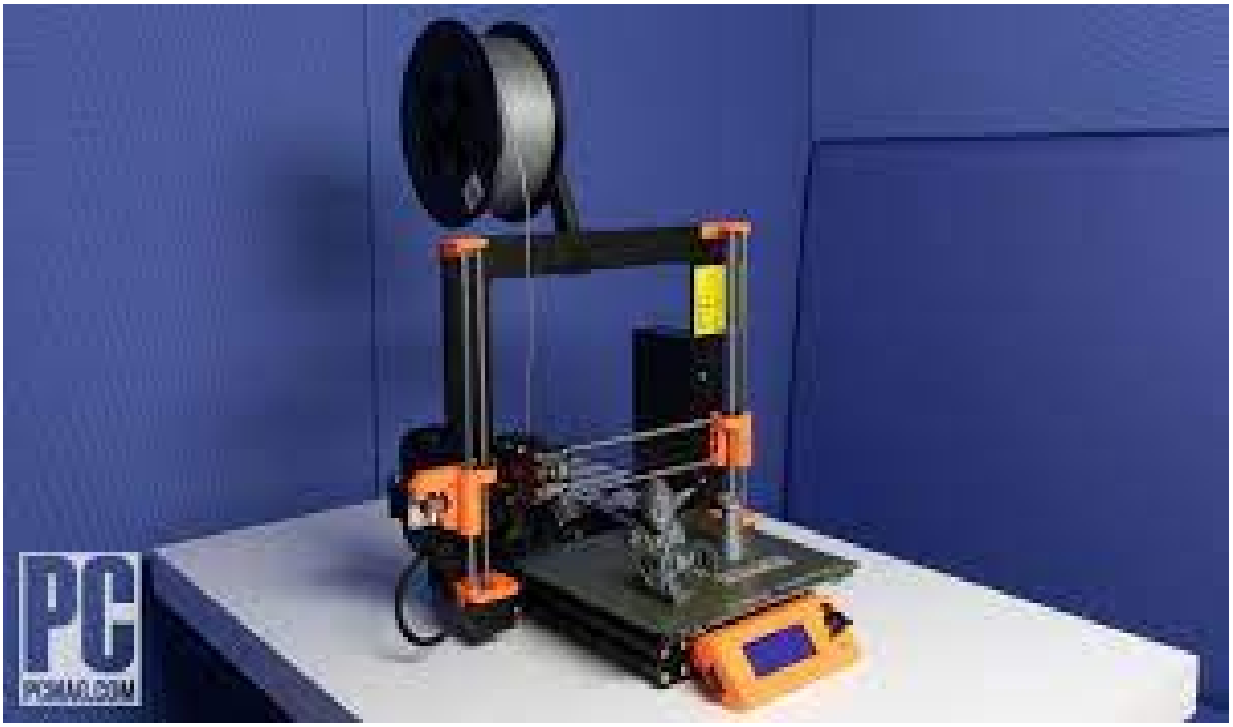
- **Design for Manufacturability (DFM):**
    - Review for DFM: Evaluate the design for manufacturability, considering factors such as panelization, solder mask application, and component placement.
    - Optimize for Production: Make adjustments to improve the efficiency and cost-effectiveness of the manufacturing process.
- **PCB Fabrication:**
    - Substrate Material Selection: Choose the appropriate substrate material (e.g., FR-4) based on thermal, mechanical, and electrical considerations.
    - Layer Preparation: Apply copper layers and substrate material according to the layer stackup design.
    - Etching: Use chemical etching or other methods to remove excess copper, leaving behind the desired traces.
    - Copper Plating and Finishing: Add copper plating to enhance trace thickness and apply surface finishes like HASL or ENIG.
- **Assembly and Component Placement:**
    - Surface Mount Technology (SMT): Place smaller components directly onto the surface of the PCB using automated pick-and-place machines.
    - Through-Hole Components: Insert leads of through-hole components through pre-drilled holes and solder them on the opposite side.
    - Automated Assembly: Use automated assembly processes for efficiency and precision.
- **Testing and Quality Assurance:**
    - In-Circuit Testing (ICT): Verify individual components and connections for electrical faults, opens, and shorts.
    - Functional Testing: Evaluate the overall performance of the assembled PCB to ensure it meets design specifications.
- **Prototyping and Iterative Design:**
    - Build Prototypes: Create prototypes for testing and validation.
    - Iterate on Design: Incorporate feedback from prototypes and testing results into the design for continuous improvement.

- **Documentation:**
    - Create Documentation: Develop comprehensive documentation, including schematics, Gerber files, and assembly instructions for future reference and replication.

3D Printing

Introduction to 3D Printing

3D printing has emerged as a transformative technology, reshaping traditional manufacturing processes and unlocking new possibilities in product development. Also known as additive manufacturing, 3D printing builds objects layer by layer, offering unparalleled flexibility and customization. The origins of 3D printing trace back to the 1980s when Chuck Hull invented stereolithography, the first 3D printing technology. Since then, various techniques and materials have evolved, contributing to the widespread adoption of 3D printing across diverse industries.



Fundamental Principles of 3D Printing

At its core, 3D printing follows a simple yet groundbreaking principle: the layer-by-layer addition of material to create a three-dimensional object. This process stands in stark contrast to subtractive manufacturing methods, where material is removed from a solid block. The key components of 3D printing include:

- **Digital Design:** The journey begins with a digital 3D model created using Computer-Aided Design (CAD) software. This digital blueprint serves as the foundation for the physical object.
- **Slicing:** The CAD model is sliced into thin, horizontal layers by specialized software. Each layer represents a cross-section of the final object.
- **Layer-by-Layer Printing:** The 3D printer interprets these sliced layers and begins the additive manufacturing process. It deposits or solidifies material layer by layer, gradually constructing the complete object.
- **Materials:** 3D printing materials vary widely and include plastics, metals, ceramics, and even biological substances. The choice of material depends on the intended application, from prototyping to end-use products.
- **Techniques:** Different 3D printing techniques exist, such as Fused Deposition Modeling (FDM), Stereolithography (SLA), Selective Laser Sintering (SLS), and more. Each technique has unique advantages and applications.

Applications Across Industries

The versatility of 3D printing has spurred its integration into numerous industries, revolutionizing manufacturing processes and product development. Some key sectors benefiting from 3D printing include:

- **Healthcare:** In the medical field, 3D printing facilitates the production of customized prosthetics, implants, and anatomical models for surgical planning. It has even ventured into bioprinting for creating living tissues and organs.
- **Automotive:** The automotive industry leverages 3D printing for rapid prototyping, creating intricate components with reduced lead times. It also enables the production of lightweight structures, enhancing fuel efficiency.
- **Aerospace:** 3D printing has disrupted aerospace manufacturing by enabling the production of complex and lightweight components. This contributes to fuel efficiency and design optimization in aircraft and spacecraft.
- **Consumer Goods:** From customized smartphone cases to personalized home décor, 3D printing empowers consumers to design and manufacture products tailored to their preferences.
- **Architecture and Construction:** In architecture, 3D printing aids in creating detailed models and prototypes. Innovations in construction involve 3D-printed structures, showcasing the potential for sustainable and cost-effective building methods.
- **Education:** 3D printing has become an invaluable tool in education, allowing students to bring their ideas to life. It promotes hands-on learning and fosters creativity in various disciplines.

Advantages of 3D Printing

The widespread adoption of 3D printing can be attributed to its numerous advantages:

- **Customization:** 3D printing enables the production of highly customized and intricate designs that would be challenging or impossible with traditional manufacturing methods.
- **Rapid Prototyping:** Design iterations can be quickly and cost-effectively implemented, accelerating the product development cycle and reducing time-to-market.

- **Reduced Material Waste:** Unlike subtractive manufacturing, 3D printing adds material selectively, minimizing waste and promoting sustainability.
- **Complex Geometries:** 3D printing excels in creating complex geometries, intricate structures, and interlocking parts that conventional manufacturing struggles to achieve.
- **On-Demand Production:** The ability to produce items on-demand eliminates the need for large inventories, reducing storage costs and enabling just-in-time manufacturing.

Challenges and Future Developments

While 3D printing has made remarkable strides, certain challenges persist, including material limitations, build size constraints, and post-processing requirements. Ongoing research and technological advancements aim to address these challenges and unlock even greater potential for 3D printing. Anticipated developments include:

- **Advanced Materials:** Continued exploration of materials, including composite materials and biodegradable options, will expand the scope of 3D printing applications.
- **Increased Speed and Scale:** Innovations in printing speed and the ability to print larger objects will enhance efficiency and enable the production of more significant components.
- **Integration with Other Technologies:** Combining 3D printing with technologies like artificial intelligence and robotics holds the promise of creating intelligent, automated manufacturing processes.
- **Bioprinting Advancements:** Progress in bioprinting may lead to breakthroughs in regenerative medicine, organ transplantation, and the development of personalized medical treatments.

3D printing stands at the forefront of a manufacturing revolution, reshaping industries and unlocking unparalleled possibilities. Its ability to transform digital designs into tangible objects with speed, precision, and customization has positioned it as a

cornerstone technology in the 21st century. As research and innovation continue to push the boundaries of what is achievable, 3D printing is poised to play an increasingly integral role in shaping the future of manufacturing and design across diverse sectors.