INTRODUCTION TO IoT

Internet of things (IoT) is more than device to device communication, it is a collection of many services, objects, humans and devices that are interconnected that can communicate as well share data and information in order to attain a common goal in different areas and applications. **Kevin Ashton coined INTERNET OF THINGS (IoT) and defined in various ways.**

- A phenomenon which connects a Variety of ***things*** – Everything that Has the ability to communicate
- A global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities
- A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on, existing and evolving, interoperable information and communication technologies

Goal of IoT:

- Connect the unconnected
- Objects that are not currently joined to a computer Network-Internet, will be connected so that they can communicate and interact with people and other objects.

- IoT is a technology transition in which the devices will allow us to sense and control the physical world by making objects smarter and connecting them through an intelligent network.
- When objects and machines can be sensed and controlled remotely by across a network, a tighter integration between physical world and computers are enabled. This allows enablement of advanced applications.

Evolution of IoT:

ARPANET was the first connected network – granddad of the Internet as we know it today. The history of IoT starts with ARPANET. In 1989 Tim Berners Lee proposed the framework of world wide web, which laid the foundation of the Internet.

In 1990 John Romkey developed a toaster that could be turned on and off over the Internet. It was a toaster wired to the computer as there was no Wi-Fi then!! This toaster is considered to be the first IoT device – the first "thing" that began Internet of Things.

The next milestone in development of IoT came in 1999 when Kevin Ashton, current Executive Director of the Auto-ID Labs, coined the term internet of things. In March 2008, the first IoT conference was held in Zurich. It brought together researchers and practitioners from both academia and industry to facilitate sharing of knowledge. In the same year, the US National Intelligence Council included the Internet of Things as one of the six disruptive civil technologies. Cisco Internet Business Solutions Group (CIBSG) said that internet of things can truly be said to be born between 2008 and 2009 when the number of things connected to the internet exceeded the number of people connected to it. CIBSG calculated that the things to people ratio grew from approximately 0.8 in 2003 to 1.84 in 2010. Fig.1.1 depicts the Evolution of IoT.

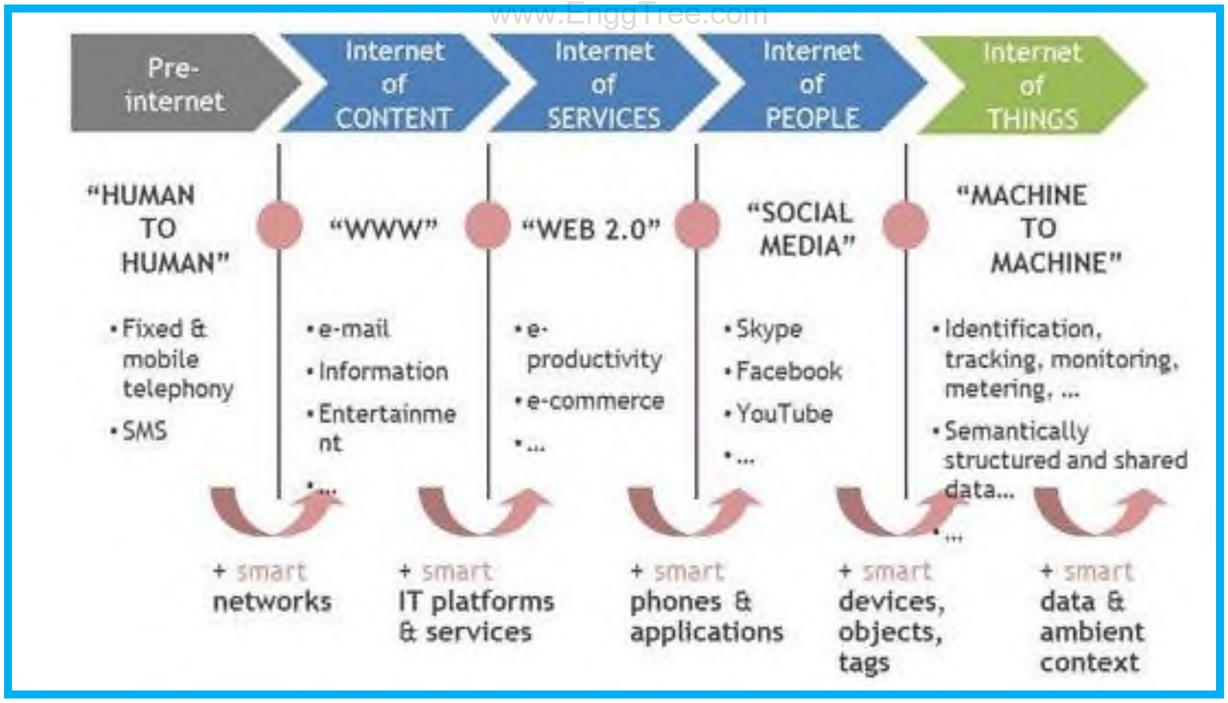


Fig.1.1 Evolution of IoT

Definition of IoT(Internet of Things):

The IoT can be defined in two ways based on

- existing Technology
- Infrastructure

Definition of IoT based on existing technology: IoT is a new revolution to the internet due to the advancement in sensor networks, mobile devices, wireless communication, networking and cloud technologies.

Definition of IoT based on infrastructure: IoT is a dynamic global network infrastructure of physical and virtual objects having unique identities, which are embedded with software, sensors, actuators, electronic and network connectivity to facilitate intelligent applications by collecting and exchanging data.

Characteristics of IoT:

Various characteristics of IoT are:

- Dynamic and self-adapting
- Self-configuring
- Interoperable Communication protocols
- Unique identity
- Integrated into information network

Dynamic and self-adapting: The IoT devices can dynamically adapt with sensed environment, their operating conditions, and user's context and take actions accordingly. For ex: Surveillance System.

Self-configuring:

- IoT devices can be able to upgrade the software with minimal intervention of user, whenever they are connected to the internet.
- They can also setup the network i.e a new device can be easily added to the existing network. For ex: Whenever there will be free wifi access one device can be connected easily.

Interoperable Communication:

IoT allows different devices (different in architecture) to communicate with each other as well as with different network. For ex: MI Phone is able to control the smart AC and smart TV of different manufacturer.

Unique identities:

- The devices which are connected to the internet have unique identities i.e IP address through which they can be identified throughout the network.
- The IoT devices have intelligent interfaces which allow communicating with users. It adapts to the environmental contexts.
- It also allows the user to query the devices, monitor their status, and control them remotely, in association with the control, configuration and management infrastructure.

Integrated into information network:

- The IoT devices are connected to the network to share some information with other connected devices. The devices can be discovered dynamically in the network by other devices. For ex. If a device has wifi connectivity, then that will be shown to other nearby devices having wifi connectivity.
- The devices ssid will be visible though out the network. Due to these things the network is also called as information network.
- The IoT devices become smarter due to the collective intelligence of the individual devices in collaboration with the information network. For Ex: weather monitoring system. Here the information collected from different monitoring nodes (sensors, arduino devices) can be aggregated and analysed to predict the weather.

IoT Enabling Technologies

The technologies which are cooperative with IoT those are as follows.

• Wireless sensor networks

• Cloud computing

• Big Data analytics

• Embedded systems

• Communication protocols

Wireless Sensor networks:

1. Wireless sensor network comprises of distributed devices, wireless sensors. These devices with sensors are used to monitor the environment and physical conditions. Since all the nodes are wireless so they communicate with each other through wifi or Bluetooth.

2. A WSN consists of several end nodes and routers as well as coordinator.

3. Sensors are attached with end nodes. Each router can also be called as end node.

4. Routers are responsible for routing the data packets from end nodes to the coordinator nodes. Coordinator node connects the WSN to the internet. The Coordinator node can be another arduino, raspberry pi or any other IoT DIY device. 5. It collects the data from all the nodes.

6. WSNs are enabled by wireless communication protocols such as IEEE802.15.4. 7. It can also be enabled by ESP 8266 and ZigBee.

8. ZigBEE Bluetooth module is based on IEEE802.15.4. It operates at 2.4 GHz frequency. It offers data rate up to 250 KB/s and ranges from 10 to 100 meters depending upon power output and environmental conditions. In WSN the devices can reconfigure themselves i.e new nodes can be added to the networks and software can be updated automatically whenever they will be connected to the internet.

9. Ex. of Wireless sensor network: Weather monitoring system, Indoor air quality monitoring, soil moisture monitoring, surveillance system, smart grids, machine prognosis and diagnosis.

Cloud Computing:

1. It is an emerging technology which enables on-demand network access to computing resources like network servers, storage, applications and services that can be rapidly provisioned and released.

2. On demand: we invoke cloud services only when we need them, they are not permanent part of IT infrastructure.

3. Pay as you go model: You pay for the cloud services when you use them, either for the short period of time or longer duration (for cloud based storage).

4. Cloud provides various services such as

i. *IAAS: Infrastructure as a service*

Instead of creating a server room we will hire it from a cloud service provider. Here user will not use its local computer, storage and processing resources rather it will use virtual machine and virtual storage, servers, networking of third party. Here the client can deploy the OS (operating system), application of his own choice. User can start, stop, configure and manage the virtual machine instances and virtual storage.

ii. *PAAS: Platform as a service*

User can develop and deploy applications. For ex. We are using various online editors to write codes like online arduino IDE, C IDE, APIs, software libraries. Here we don't need to install anything. The cloud service provider will manage servers, network, OS and storage. The users will develop, deploy, configure and manage applications on the cloud infrastructure.

iii. *SAAS: Software as a service*

It provides complete software application or the user interface to the application itself. The user is not concerned about the underlying architecture of cloud only service provider is responsible for this. It is platform independent and can be accessed from various client devices such as workstation, laptop, tablet and smart phone, running different OS. Ex: The online software we use like online image converter, doc converter etc.

Big data analytics:

Big data refers to large amount of data which cannot be stored, processed and analysed using traditional database like (oracle, mysql) and traditional processing tools. In big data analytics BIG refers to 5 Vs.

• Volume

• Velocity

• Variety

• Veracity

• Value

Volume: volume refers to the massive amount of data generated from the IoT systems. There is no threshold value for generated data. It is difficult to store, process and analyse using traditional database and processing tools. Ex: The volume of data generated by modern IT, industrial and healthcare system.

Velocity: The rate at which the data is generated from the IoT system. This is the primary reason for the exponential growth of data. Velocity refers to how fast the data is generated and how frequently it varies. Ex: Modern IT, industrial and other systems like social networking sites are generating data at increasingly higher speed.

Variety: Variety refers to different forms of data. Since there is various domain of IoT so various type of data is generated from different IoT domain. Those data are called as sparse data. Those data include text, audio, video etc.. The variety of data is mainly divided into 3 types i.e.

- structured
- semi structured
- unstructured

Structured data: The data which has a fixed format to be stored is known as structured data. The data stored in database like oracle, mysql is an example of structured data. With a simple query data can be retrieved from the database.

Semi-structured data: The data which has not a fixed format to be stored but uses some elements and components through which they can be analyzed easily is known as semi structured data. Ex: HTML, XML, JSON data

Unstructured data: The data which has not any fixed format. It is difficult to store and analyse. It can be analyzed after converting into structured data. Ex: Audio, video (gif, audio with lyrics), Text (containing special symbols).

Veracity: The data in doubt is known as veracity. Sometimes what happen it is very difficult accept the data stored in database. This happens due to typical error, corrupted storage or data.

Value: It is efficient to access big data if we can turn it into values i.e we can find greater insights from it so that we can perform some action to get the desired output. This will be beneficial for the organisation. Otherwise it has no use.

Embedded Systems:

- An embedded system is a computer system that has hardware and software embedded to perform specific task.
- The key components of an embedded system include microprocessor or micro controller, memory (RAM, ROM, Cache), networking units (Ethernet, Wi-Fi adapter), input/output units (display, keyboard, etc) and storage (flash memory). They use some special types of processor such as digital signal processor, graphics processor and application specific processor). Embedded system uses embedded OS like RTOS.
- Ex. of embedded systems: digital watch, digital camera, vending machines.

Communication protocols:

- Protocol is nothing but rules and regulations. Communication protocol is the backbone of the IoT system.
- It allows interoperability among various devices. It enables network connectivity and coupling to applications.
- It allows devices to exchange data over the network. These protocols define data exchange format, data encoding, addressing schemes for devices and routing of packets from source to destination. It also includes sequence control, flow control and retransmission of lost packets.

IoT Architectures: oneM2M, IoT World Forum (IoTWF)

The difference between IT and IoT networks is much like the difference between residential architecture and stadium architecture. While traditional network architectures for IT have served us well for many years, they are not well suited to the complex requirements of IoT. Chapter 1, "What Is IoT?" introduces some of the differences between IT and OT, as well as some of the inherent challenges posed by IoT. These differences and challenges are driving fundamentally new architectures for IoT systems.

The key difference between IT and IoT is the data. While IT systems are mostly concerned with reliable and continuous support of business applications such as email, web, databases, CRM systems, and so on, IoT is all about the data generated by sensors and how that data is used. The essence of IoT architectures thus involves how the data is transported, collected, analyzed, and ultimately acted upon.

oneM2M IoT Standardized Architecture

In an effort to standardize the rapidly growing field of machine-to-machine (M2M) communications, the European Telecommunications Standards Institute (ETSI) created the M2M Technical Committee in 2008. The goal of this committee was to create a common architecture that would help accelerate the adoption of M2M applications and devices. Over time, the scope has expanded to include the Internet of Things.

Other related bodies also began to create similar M2M architectures, and a common standard for M2M became necessary. Recognizing this need, in 2012 ETSI and 13 other founding members launched oneM2M as a global initiative designed to promote efficient M2M communication systems and IoT. The goal of oneM2M is to create a common ser- vices layer, which can be readily embedded in field devices to allow communication with application servers.1 oneM2M's framework focuses on IoT services, applications, and platforms. These include smart metering applications, smart grid, smart city automation, e-health, and connected vehicles.

One of the greatest challenges in designing an IoT architecture is dealing with the hetero-geneity of devices, software, and access methods. By developing a horizontal platform architecture, oneM2M is developing standards that allow interoperability at all levels of the IoT stack.

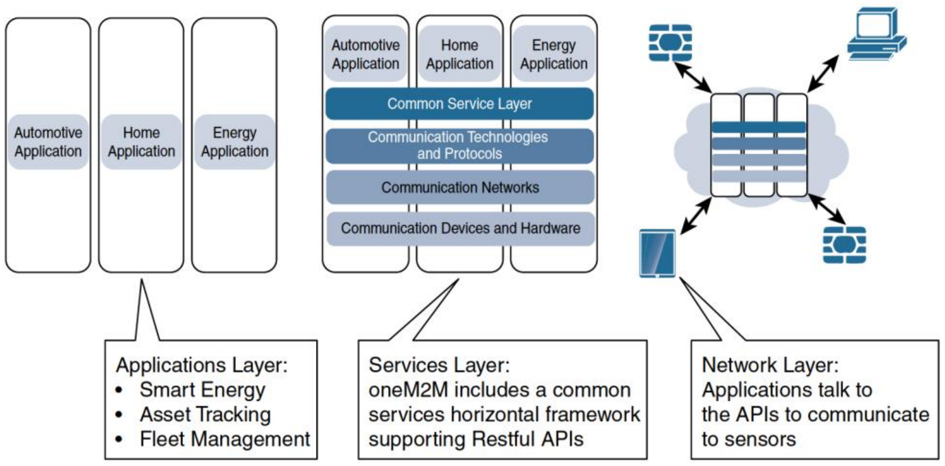


Figure 1.3.1: ***The Main Elements of the oneM2M IoT Architecture***

The oneM2M architecture divides IoT functions into three major domains: the application layer, the services layer, and the network layer. While this architecture may seem simple and somewhat generic at first glance, it is very rich and promotes interoperability through IT-friendly APIs and supports a wide range of IoT technologies.

Applications layer: The oneM2M architecture gives major attention to connectivity between devices and their applications. This domain includes the application-layer protocols and attempts to standardize northbound API definitions for interaction with business intelligence (BI) systems. Applications tend to be industry-specific and have their own sets of data models, and thus they are shown as vertical entities.

Services layer: This layer is shown as a horizontal framework across the vertical industry applications. At this layer, horizontal modules include the physical network that the IoT applications run on, the underlying management protocols, and the hardware. Examples include backhaul communications via cellular, MPLS networks, VPNs, and so on. Riding on top is the common services layer. This conceptual layer adds APIs and middleware supporting third-party services and applications. One of the stated goals of oneM2M is to "develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software nodes, and rely upon connecting the myriad of devices in the field area network to M2M application servers, which typically reside in a cloud or data center."

Network layer: This is the communication domain for the IoT devices and end- points. It includes the devices themselves and the communications network that links them. Embodiments of this communications infrastructure include wireless mesh technologies, such as IEEE 802.15.4, and wireless point-to-multipoint systems, such as IEEE 801.11ah.

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, the one put forth by the IoT World Forum offers a clean, simplified perspective on IoT and includes edge computing, data storage, and access. It provides a succinct way of visualizing IoT from a technical perspective. Each of the seven layers is broken down into specific functions, and security encompasses the entire model.

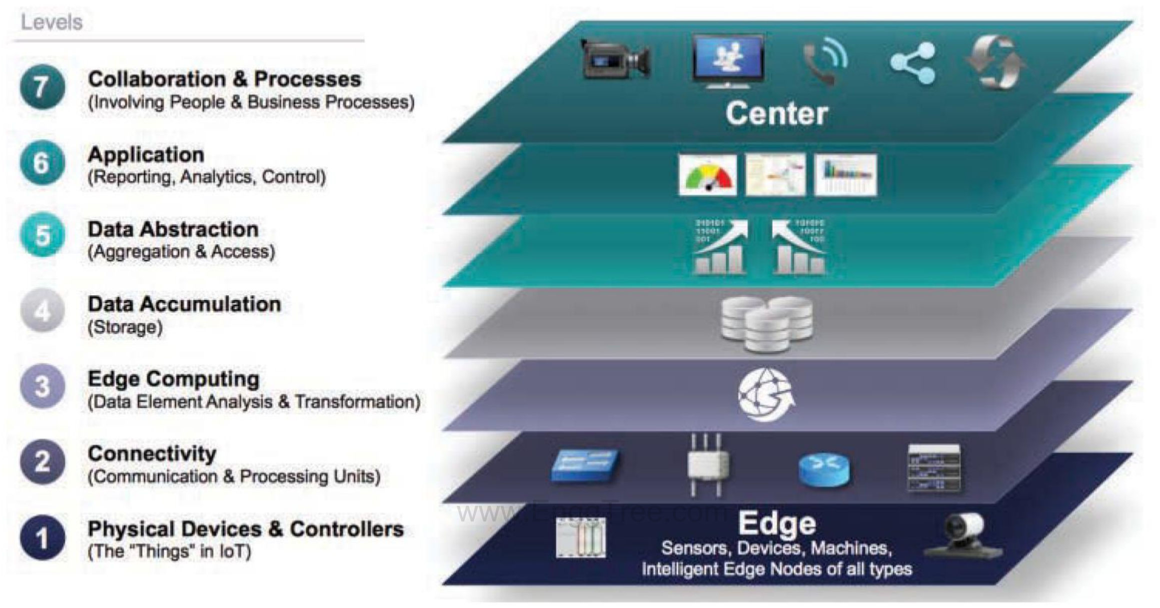

Levels
7 Collaboration & Processes
(Involving People & Business Processes)
6 Application
(Reporting, Analytics, Control)
5 Data Abstraction
(Aggregation & Access)
4 Data Accumulation
(Storage)
3 Edge Computing
(Data Element Analysis & Transformation)
2 Connectivity
(Communication & Processing Units)
1 Physical Devices & Controllers
(The "Things" in IoT)
Center
Edge
Sensors, Devices, Machines,
Intelligent Edge Nodes of all types

Figure 1.3.2 *IoT Reference Model Published by the IoT World Forum*

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

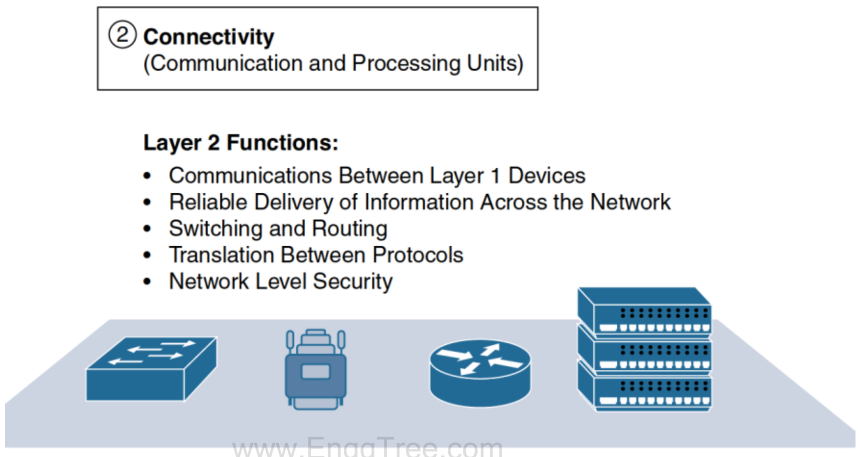
- Decompose the IoT problem into smaller parts
- Identify different technologies at each layer and how they relate to one another
- Define a system in which different parts can be provided by different vendors
- Have a process of defining interfaces that leads to interoperability
- Define a tiered security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

Layer 1: Physical Devices and Controllers Layer

The first layer of the IoT Reference Model is the physical devices and controller's layer. This layer is home to the "things" in the Internet of Things, including the various endpoint devices and sensors that send and receive information. The size of these "things" can range from almost microscopic sensors to giant machines in a factory. Their primary function is generating data and being capable of being queried and/or controlled over a network.

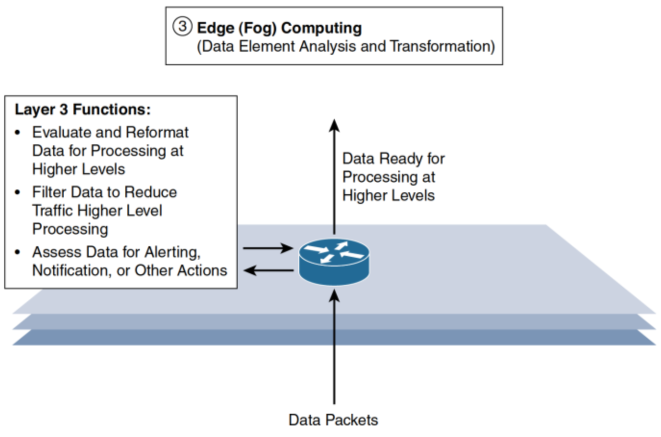
Layer 2: Connectivity Layer

In the second layer of the IoT Reference Model, the focus is on connectivity. The most important function of this IoT layer is the reliable and timely transmission of data.
More specifically, this includes transmissions between Layer 1 devices and the network and between the network and information processing 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 really distinguish between the last-mile network (the network between the sensor/endpoint and the IoT gateway, discussed later in this chapter), gateway, and backhaul networks.


② Connectivity
(Communication and Processing Units)
Layer 2 Functions:
• Communications Between Layer 1 Devices
• Reliable Delivery of Information Across the Network
• Switching and Routing
• Translation Between Protocols
• Network Level Security

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 data reduction and converting network data flows into information that is ready for storage and processing by higher 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.


③ Edge (Fog) Computing
(Data Element Analysis and Transformation)
Layer 3 Functions:
• Evaluate and Reformat Data for Processing at Higher Levels
• Filter Data to Reduce Traffic Higher Level Processing
• Assess Data for Alerting, Notification, or Other Actions
Data Ready for Processing at Higher Levels
Data Packets

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.

Alternative IoT Models

In addition to the two IoT reference models already presented in this chapter, several other reference models exist. These models are endorsed by various organizations and standards bodies and are often specific to certain industries or IoT applications.

Purdue Model for Control Hierarchy

The Purdue Model for Control Hierarchy is a common and well-understood model that segments devices and equipment into hierarchical levels and functions. It is used as the basis for ISA-95 for control hierarchy, and in turn for the IEC- 62443 (formerly ISA-99) cyber security standard. It has been used as a base for many IoT-related models and standards across industry.

Industrial Internet Reference Architecture (IIRA) by Industrial Internet Consortium (IIC)

The IIRA is a standards-based open architecture for Industrial Internet Systems (IISs). To maximize its value, the IIRA has broad industry applicability to drive interoperability, to map applicable technologies, and to guide technology and standard development. The description and representation of the architecture are generic and at a high level of abstraction to support the requisite broad industry applicability. The IIRA distils and abstracts common characteristics, features and patterns from use cases well understood at this time, predominantly those that have been defined in the IIC.

Internet of Things– Architecture (IoT-A)

IoT-A created an IoT architectural reference model and defined an initial set of key building blocks that are foundational in fostering the emerging Internet of Things. Using an experimental para- digm, IoT-A combined top-down reasoning about architectural principles and design guidelines with simulation and prototyping in exploring the technical consequences of architectural design choices.

Simplified IoT Architecture and Core IoT Functional Stack

we present an IoT framework that highlights the fundamental building blocks that are common to most IoT systems and which is intended to help you in designing an IoT network. This framework is presented as two parallel stacks: The IoT Data Management and Compute Stack and the Core IoT Functional Stack. Reducing the framework down to a pair of three-layer stacks in no way suggests that the model lacks the detail necessary to develop a sophisticated IoT strategy. Rather, the intention is to simplify the IoT architecture into its most basic building blocks and then to use it as a foundation to understand key design and deployment principles that are applied to industry-specific use cases.

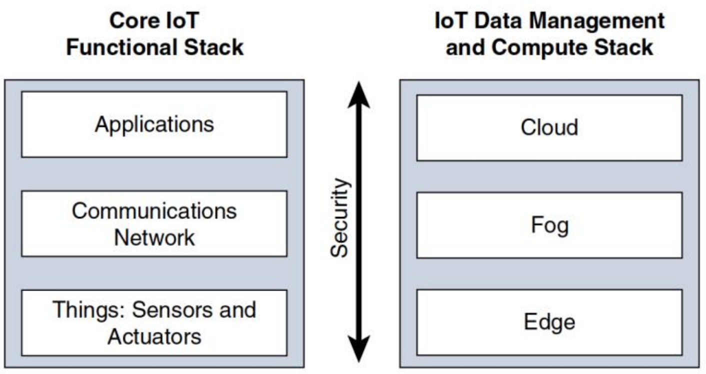

Core IoT Functional Stack
Applications
Communications Network
Things: Sensors and Actuators
Security
IoT Data Management and Compute Stack
Cloud
Fog
Edge

Figure 1.4.1 Simplified IoT Architecture

Nearly every published IoT model includes core layers similar to those shown on the left side of Figure **1.4.1**, including “things,” a communications network, and applications. However, unlike other models, the framework presented here separates the core IoT and data management into parallel and aligned stacks, allowing you to carefully examine the functions of both the network and the applications at each stage of a complex IoT system. This separation gives you better visibility into the functions of each layer. The presentation of the Core IoT Functional Stack in three layers is meant to simplify your understanding of the IoT architecture into its most foundational building blocks. Of course, such a simple architecture needs to be expanded on. The network communications layer of the IoT stack itself involves a significant amount of detail and incorporates a vast array of technologies. Consider for a moment the heterogeneity of IoT sensors and the many different ways that exist to connect them to a network. The network communications layer needs to consolidate these together, offer gateway and backhaul technologies, and ultimately bring the data back to a central location for analysis and processing.

The applications and analytics layer of IoT doesn't necessarily exist only in the data center or in the cloud. Due to the unique challenges and requirements of IoT, it is often necessary to deploy applications and data management throughout the architecture in a tiered approach, allowing data collection, analytics, and intelligent controls at multiple points in the IoT system. In the model presented in this book, data management is aligned with each of the three layers of the Core IoT Functional Stack. The three data management layers are the edge layer (data management within the sensors themselves), the fog layer (data management in the gateways and transit network), and the cloud layer (data management in the cloud or central data center).

The Core IoT Functional Stack can be expanded into sublayers containing greater detail and specific network functions. For example, the communications layer is broken down into four separate sublayers: the access network, gateways and backhaul, IP transport, and operations and management sublayers.
The applications layer of IoT networks is quite different from the application layer of a typical enterprise network. Instead of simply using business applications, IoT often involves a strong big data analytics component.

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.

- **"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 longer-range 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 analysed conditions and change their behaviours or parameters.

Layer 1: Things: Sensors and Actuators Layer

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 line-powered 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.

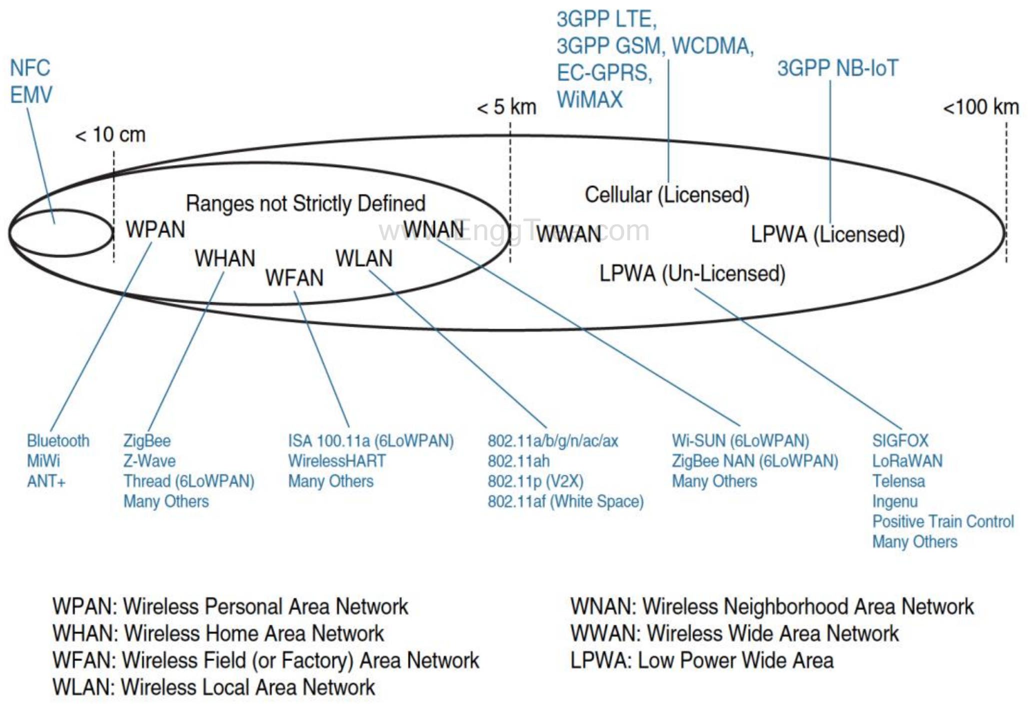
Object density per cell: This classification is based on the number of smart objects (with a similar need to communicate) over a given area, connected to the same gate- way. An oil pipeline may utilize a single sensor at key locations every few miles. By contrast, telescopes like the SETI Colossus telescope 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

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 deter- mined 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.



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.

LAN (local area network): Scale of up to 100 m. This term is very common in net- working, and it is therefore also commonly used in the IoT space when standard net- working 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 kilometre’s) and WAN (wide area network, with a range of more than a few kilometre’s), are also commonly used.

Point-to-point topologies: These topologies allow one point to communicate with another point. This topology in its strictest sense is uncommon for IoT access, as it would imply that a single object can communicate only with a single gateway.
However, several technologies are referred to as “point-to-point” when each object establishes an individual session with the gateway. The “point-to-point” concept, in that case, often refers to the communication structure more than the physical topology.

Point-to-multipoint topologies: These topologies allow one point to communicate with more than one other point. Most IoT technologies where one or more than one gateways communicate with multiple smart objects are in this category. However, depending on the features available on each communicating mode, several subtypes need to be considered. A particularity of IoT networks is that some nodes (for example, sensors) support both data collection and forwarding functions, while some other nodes (for example, some gateways) collect the smart object data, sometimes instruct the sensor to perform specific operations, and also interface with other net- works or possibly other gateways.

Gateways and Backhaul Sublayer
Data collected from a smart object may need to be forwarded to a central station where data is processed. As this station is often in a different location from the smart object, data directly received from the sensor through an access technology needs to be forwarded to another medium (the backhaul) and transported to the central station. The gateway is in charge of this inter-medium communication.

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 neighbourhood (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, low-throughput 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 neighbourhoods may also connect the same way, thus forming a FAN.

Layer 3: Applications and Analytics Layer

Analytics Versus Control Applications

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 behaviour of the smart object or the behaviour 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.

Data Versus Network Analytics

Analytics is a general term that describes processing information to make sense of collected data. In the world of IoT, a possible classification of the analytics function is as follows:

Data analytics: This type of analytics processes the data collected by smart objects and combines it to provide an intelligent view related to the IoT system. At a very basic level, a dashboard can display an alarm when a weight sensor detects that a shelf is empty in a store. In a more complex case, temperature, pressure, wind, humidity, and light levels collected from thousands of sensors may be combined and then processed to determine the likelihood of a storm and its possible path. In this case, data processing can be very complex and may combine multiple changing values over complex algorithms. Data analytics can also monitor the IoT system itself.
For example, a machine or robot in a factory can report data about its own movements. This data can be used by an analytics application to report degradation in the movement speeds, which may be indicative of a need to service the robot before a part breaks.

Network analytics: Most IoT systems are built around smart objects connected to the network. A loss or degradation in connectivity is likely to affect the efficiency of the system. Such a loss can have dramatic effects. For example, open mines use wireless networks to automatically pilot dump trucks. A lasting loss of connectivity may result in an accident or degradation of operations efficiency (automated dump trucks typically stop upon connectivity loss).

Fog, Edge and Cloud in IoT

Fog Computing

The solution to the challenges in IoT 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.

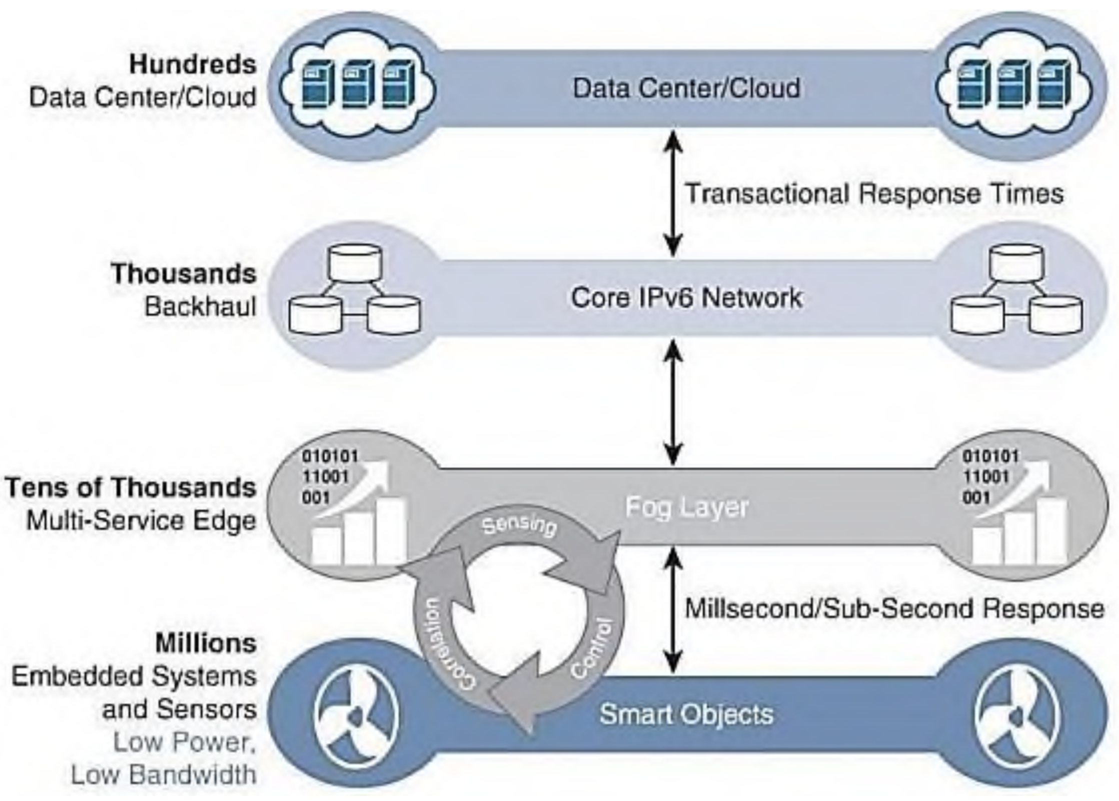


Fig.1.3 Fog computing

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. Pre-processing of data in the fog nodes allows upper-layer applications to perform batch processing on a subset of the data.

Advantages of fog computing in IoT

The fogging approach has many benefits for the Internet of Things, Big Data, and real-time analytics. The main advantages of fog computing over cloud computing are as follows:

- **Low latency -** Fog tends to be closer to users and can provide a quicker response.
- **There is no problem with bandwidth -** pieces of information are aggregated at separate points rather than sent through a channel to a single hub.
- Due to the many interconnected channel**s - loss of connection is impossible**.
- **High Security -** because the data is processed by multiple nodes in a complex distributed system.
- **Improved User Experience -** Quick responses and no downtime make users satisfied.
- **Power-efficiency -** Edge nodes run power-efficient protocols such as Bluetooth, Zigbee, or Z-Wave.

Disadvantages of fog computing in IoT

The technology has no obvious disadvantages, but some shortcomings can be named:

- **Fog is an additional layer in a more complex system -** a data processing and storage system.
- **Additional expenses -** companies must buy edge devices: **routers, hubs, gateways**.
- **Limited scalability -** Fog is not scalable like a cloud.

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.

Some new classes of IoT endpoints have enough compute capabilities to perform at least low-level analytics and filtering to make basic decisions. For example, consider a water sensor on a fire hydrant. While a fog node sitting on an electrical pole in the distribution network may have an excellent view of all the fire hydrants in a local neighborhood, a node on each hydrant would have clear view of a water pressure drop on its own line and would be able to quickly generate an alert of a localized problem.

Another example is in the use of smart meters. Edge compute–capable meters are able to communicate with each other to share information on small subsets of the electrical distribution grid to monitor localized power quality and consumption, and they can inform fog node of events that may pertain to only tiny sections of the grid. Models such as these help ensure the highest quality of power delivery to customers.

Cloud computing

The delivery of on-demand computing services is known as cloud computing. We may use applications to store and process power over the Internet. Without owning any computing infrastructure or data center, anyone can rent access to anything from applications to storage from a cloud service provider.It is a pay-as-you-go service.By using cloud computing services and paying for what we use, we can avoid the complexity of owning and maintaining infrastructure. Cloud computing service providers can benefit from significant economies of scale by providing similar services to customers.

Cloud computing technology provides a variety of services that are classified into three groups:

- **IaaS (Infrastructure as a Service)** - A remote data center with data storage capacity, processing power, and networking resources.
- **PaaS (Platform as a Service)** - A development platform with tools and components to build, test, and launch applications.
- **SaaS (Software as a Service)** - Software tailored to suit various business needs.

By connecting your company to the Cloud, you can access the services mentioned above from any location and through various devices.

Therefore, availability is the biggest advantage. Plus, there's no need to maintain local servers and worry about downtimes - the vendor supports everything for you, saving you money.

Integrating the Internet of Things with the Cloud is an affordable way to do business. Off-premises services provide the scalability and flexibility needed to manage and analyze data collected by connected devices. At the same time, specialized platforms (**e.g., Azure IoT Suite, IBM Watson, AWS, and Google Cloud IoT**) give developers the power to build IoT apps without major investments in hardware and software.

Advantages of Cloud for IoT

Since connected devices have limited storage capacity and processing power, integration with cloud computing comes to the aid of:

- **Improved performance** - faster communication between IoT sensors and data processing systems.
- **Storage** Capacity - Highly scalable and unlimited storage space can integrate, aggregate, and share huge data.
- **Processing Capabilities** - Remote data centers provide unlimited virtual processing capabilities on demand.
- Low Cost - The license fee is less than the cost of on-premises equipment and its ongoing maintenance.

Disadvantages of Cloud for IoT

Unfortunately, nothing is spotless, and cloud technology has some drawbacks, especially for Internet of Things services.

- **High latency** - More and more IoT apps require very low latency, but the Cloud cannot guarantee this due to the distance between client devices and data processing centers.
- **Downtimes** - Technical issues and network interruptions can occur in any Internet-based system and cause customers to suffer from outages; Many companies use multiple connection channels with automatic failover to avoid problems.
- **Security and Privacy** - your data is transferred via globally connected channels along with thousands of gigabytes of other users' information; No wonder the system is vulnerable to cyber-attacks or data loss; the problem can be partially solved with the help of hybrid or private clouds.

Difference between Fog Computing and Cloud Computing:

- In fog computing, data is received from IoT devices using any protocol.
- Cloud computing receives and summarizes data from different fog nodes.

Structure:

- Fog has a decentralized architecture where information is located on different nodes at the source closest to the user.
- There are many centralized data centers in the Cloud, making it difficult for users to access information on the networking area at their nearest source.

Protection:

- Fog is a more secure system with different protocols and standards, which minimizes the chances of it collapsing during networking.
- As the Cloud operates on the Internet, it is more likely to collapse in case of unknown network connections.

Component:

- o Fog has some additional features in addition to the features provided by the components of the Cloud that enhance its storage and performance at the end gateway.
- o Cloud has different parts such as frontend platform (e.g., mobile device), backend platform (storage and servers), cloud delivery, and network (Internet, intranet, intercloud).

Accountability:

- o Here, the system's response time is relatively higher compared to the Cloud as fogging separates the data and then sends it to the Cloud.
- o Cloud service does not provide any isolation in the data while transmitting the data at the gate, increasing the load and thus making the system less responsive.

Application:

- o Edge computing can be used for smart city traffic management, automating smart buildings, visual Security, self-maintenance trains, wireless sensor networks, etc.
- o Cloud computing can be applied to e-commerce software, word processing, online file storage, web applications, creating image albums, various applications, etc.

Reduces latency:

- o Fog computing cascades system failure by reducing latency in operation. It analyzes the data close to the device and helps in averting any disaster.

Flexibility in Network Bandwidth:

- o Large amounts of data are transferred from hundreds or thousands of edge devices to the Cloud, requiring fog-scale processing and storage.
- o For example, commercial jets generate 10 TB for every 30 minutes of flight. Fog computing sends selected data to the cloud for historical analysis and long-term storage.

Wide geographic reach:

- o Fog computing provides better quality of services by processing data from devices that are also deployed in areas with high network density.

- On the other hand, Cloud servers communicate only with IP and not with the endless other protocols used by IoT devices.

Real-time analysis:

- Fog computing analyzes the most time-sensitive data and operates on the data in less than a second, whereas cloud computing does not provide round-the-clock technical support.

Operating Expenses:

- The license fee and on-premises maintenance for cloud computing are lower than fog computing. Companies have to buy edge device routers.

Fog Computing vs. Cloud Computing for IoT

According to Statista, by 2020, there will be 30 billion IoT devices worldwide, and by 2025 this number will exceed 75 billion connected things.

These tools will produce huge amounts of data that will have to be processed quickly and permanently. Fog computing works similarly to cloud computing to meet the growing demand for IoT solutions.

IoT Ecosystem Overview

In IoT ecosystems, computing interactions are driven by smart objects, which are system entities considered the main building blocks of the IoT environment. By putting intelligence into everyday objects (i.e., dedicated embedded systems into everyday physical devices), these devices are turned into smart objects able not only to collect information from the environment and interact/control the things of the physical world, but also to be interconnected to each other through the Internet to autonomously exchange pervasive data and information. The expected huge number of interconnected devices and the significant amount of available data, open new opportunities to create smart services that will bring tangible benefits to the society, environment, economy, and individual citizens.

IoT considers the pervasive presence of a variety of smart objects interacting and cooperating in the physical environment through available ubiquitous services. Thus, the goal of the IoT is to enable things to be interconnected at anytime, anyplace, with anything and anyone, ideally using any path/network and any source.

Let us consider the "city ecosystem" as an example of how the city of the future (i.e the Smart City) will look like in the coming years. Indeed, a smart city is a kind of city that should be able to operate simultaneously on two representation levels, physical and virtual, respectively. These abstractions should imply in the provision of intelligent solutions that ensure efficiency at multiple levels, aiming basically to: (i) a more aware and optimized usage of the resources of the city, (ii) a minimization of environmental impact (e.g.,by reducing CO2 emissions), and (iii) an increase in the life quality of citizens' in terms of safety, health, and wellness. This smart capability is desired due to the fact that, today, half of the global population is concentrated in the cities, and hence, is increasingly consuming the city's resources (e.g., light, water) every day. Besides, quality, sustainability, and security are crucial requirements and unavoidable issues for the city.

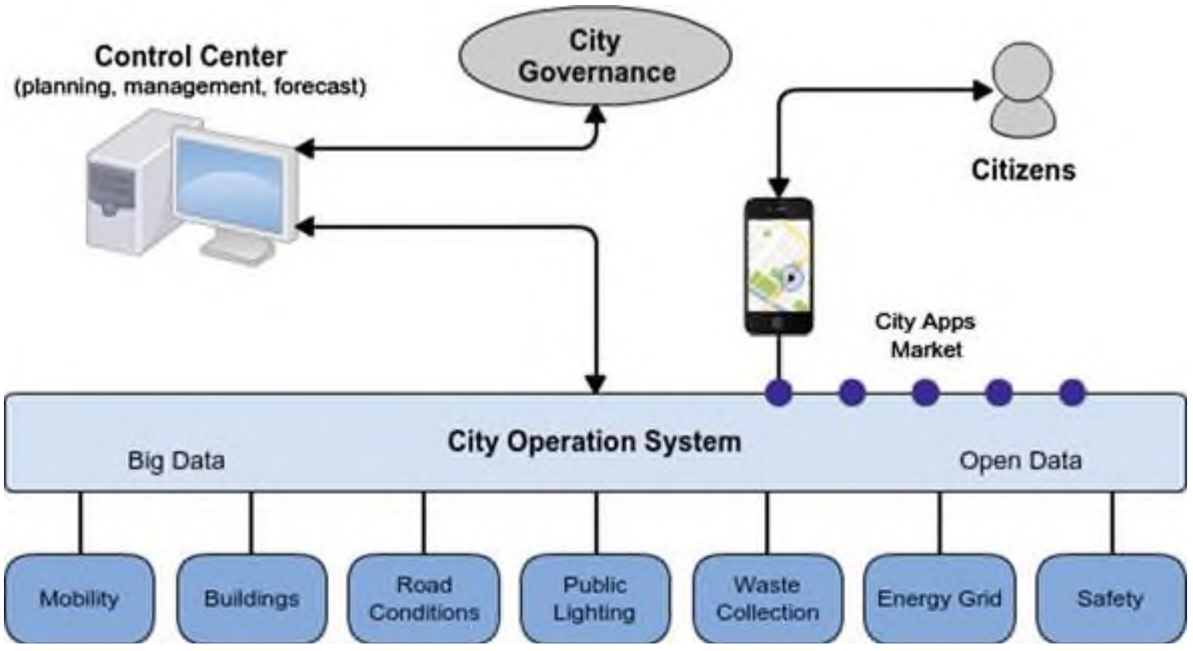


Fig.2.3 Schematic representation of a Smart City ecosystem

[Ref: L.A. Amaral et al. Middleware Technology for IoT Systems: Challenges and Perspectives Toward 5G,2016]

A smart city should provide autonomous management of its public services (e.g., transport, energy, lighting, waste management, health, and entertainment) through the widespread adoption of Information and Communication Technologies (ICT). Such technologies are the basis for the provision of a logical/virtual infrastructure that should be able to control and coordinate the physical infrastructure of the city in order to adapt the city services to the actual citizen needs, while reducing waste and making the city more sustainable. In this way, IoT is essential in this transition process since it has helped conventional cities to be turned into smart cities where traditional and more emerging sectors such as mobility, buildings, energy, living, and governance will also benefit from this transition. For example, smart mobility services can be created to provide effective tools to the citizens to accurately plan their journeys with public/private transportation.

Figure 2.3 provides a schematic representation of a smart city ecosystem. In this perspective, the city will be equipped with physical devices or things (i.e., network of sensors, cameras, speakers, smart meters, and thermostats) that will collect information of the environment. The gathered information, the so-called “Big Data” (the name refers to its large volume and its heterogeneity in terms of content and data representation), will not only be used for the improvement of just a single city service/application, but it will also be shared among different services into the smart city ecosystem. In this sense, a common platform for the operational management of city elements—a sort of City Operating System, will be responsible for managing, storing, analyzing, processing, and forwarding the city data anywhere and anytime, to anyone and anything, helping the city to improve and adapting the city services to users’ needs. This management layer, no longer vertical but horizontal, will ensure interoperability, coordination, and optimization of individual services/applications through the analysis of heterogeneous information flows. Citizens/authorities will access the services offered by the platform through their applications, will consume them and will actively participate by creating additional content (i.e., new data or applications) that will be provided as further input to the City Operating System.

Sensors, Actuators, Smart Objects and Control Units

This section defines sensors, actuators, and smart objects and describes how they are the fundamental building blocks of IoT networks.
Sensor Networks: This section covers the design, drivers for adoption, and deployment challenges of sensor networks.

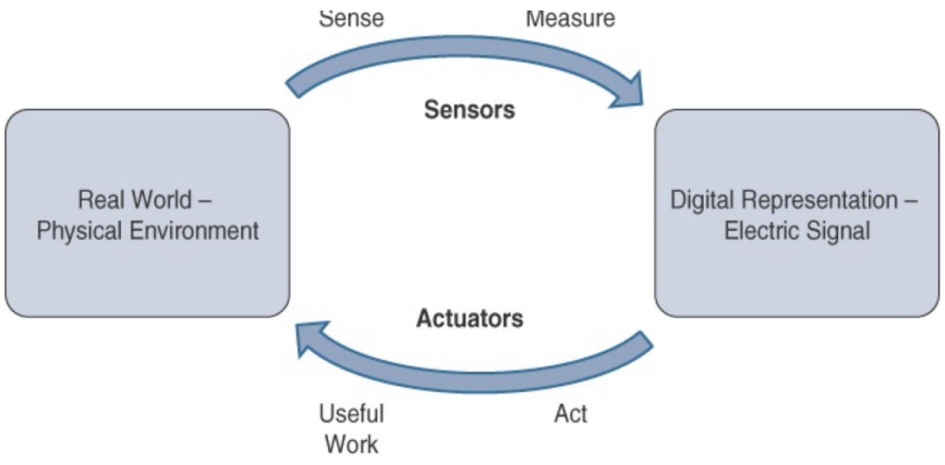
Sensors:

A sensor does exactly as its name indicates: It senses. More specifically, a sensor measures some physical quantity and converts that measurement reading into a digital representation. That digital representation is typically passed to another device for transformation into useful data that can be consumed by intelligent devices or humans.
Active or passive: Sensors can be categorized based on whether they produce an energy output and typically require an external power supply (active) or whether they simply receive energy and typically require no external power supply (passive). Invasive or non-invasive: Sensors can be categorized based on whether a sensor is part of the environment it is measuring (invasive) or external to it (non-invasive). Contact or no-contact: Sensors can be categorized based on whether they require physical contact with what they are measuring (contact) or not (no-contact).

Absolute or relative: Sensors can be categorized based on whether they measure on an absolute scale (absolute) or based on a difference with a fixed or variable reference value (relative).
Area of application: Sensors can be categorized based on the specific industry or vertical where they are being used.
How sensors measure: Sensors can be categorized based on the physical mechanism used to measure sensory input (for example, thermoelectric, electrochemical, piezoresistive, optic, electric, fluid mechanic, photoelastic).
What sensors measure: Sensors can be categorized based on their applications or what physical variables they measure.

Actuators:

Actuators are natural complements to sensors. Figure 3-4 demonstrates the symmetry and complementary nature of these two types of devices. As discussed in the previous section, sensors are designed to sense and measure practically any measurable variable in the physical world. They convert their measurements (typically analog) into electric signals or digital representations that can be consumed by an intelligent agent (a device or a human). Actuators, on the others hand, receive some type of control signal (commonly an electric signal or digital command) that triggers a physical effect, usually some type of motion, force, and so on.


Sense
Measure
Sensors
Real World – Physical Environment
Digital Representation – Electric Signal
Actuators
Useful Work
Act

Some common ways that they can be classified include the following:

Type of motion: Actuators can be classified based on the type of motion they produce (for example, linear, rotary, one/two/three-axes).

Power: Actuators can be classified based on their power output (for example, high power, low power, micro power)

Binary or continuous: Actuators can be classified based on the number of stable-state outputs.

Area of application: Actuators can be classified based on the specific industry or vertical where they are used.

Type of energy: Actuators can be classified based on their energy type.

Smart Objects

Smart objects are, quite simply, the building blocks of IoT. They are what transform everyday objects into a network of intelligent objects that are able to learn from and interact with their environment in a meaningful way. It can't be stressed enough that the real power of smart objects in IoT comes from being networked together rather than being isolated as standalone objects. This ability to communicate over a network has a multiplicative effect and allows for very sophisticated correlation and interaction between disparate smart objects. For instance, recall the smart farming sensors described previously. If a sensor is a standalone device that simply measures the humidity of the soil, it is interesting and useful, but it isn't revolutionary. If that same sensor is connected as part of an intelligent network that is able to coordinate intelligently with actuators to trigger irrigation systems as needed based on those sensor readings, we have something far more powerful.

A smart object, is a device that has, at a minimum, the following four defining characteristics Processing unit: A smart object has some type of processing unit for acquiring data, processing and analyzing sensing information received by the sensor(s), coordinating control signals to any actuators, and controlling a variety of functions on the smart object, including the communication and power systems. The specific type of processing unit that is used can vary greatly, depending on the specific processing needs of different applications. The most common is a microcontroller because of its small form factor, flexibility, programming simplicity, ubiquity, low power consumption, and low cost.

Sensor(s) and/or actuator(s): A smart object is capable of interacting with the physical world through sensors and actuators. As described in the previous sections, a sensor learns and measures its environment, whereas an actuator is able to produce some change in the physical world. A smart object does not need to contain both sensors and actuators. In fact, a smart object can contain one or multiple sensors and/or actuators, depending upon the application.

Communication device: The communication unit is responsible for connecting a smart object with other smart objects and the outside world (via the network). Communication devices for smart objects can be either wired or wireless. Overwhelmingly, in IoT networks smart objects are wirelessly interconnected for a number of reasons, including cost, limited infrastructure availability, and ease of deployment. There are myriad different communication protocols for smart objects.

Power source: Smart objects have components that need to be powered. Interestingly, the most significant power consumption usually comes from the communication unit of a smart object. As with the other three smart object building blocks, the power requirements also vary greatly from application to application. Typically, smart objects are limited in power, are deployed for a very long time, and are not easily accessible. This combination, especially when the smart object relies on battery power, implies that power efficiency, judicious power management, sleep modes, ultra-low power consumption hardware, and so on are critical design elements.

Trends in Smart Objects

Size is decreasing: As discussed earlier, in reference to MEMS, there is a clear trend of ever-decreasing size. Some smart objects are so small they are not even visible to the naked eye. This reduced size makes smart objects easier to embed in everyday objects.

Power consumption is decreasing: The different hardware components of a smart object continually consume less power. This is especially true for sensors, many of which are completely passive. Some battery-powered sensors last 10 or more years without battery replacement.

Processing power is increasing: Processors are continually getting more powerful and smaller. This is a key advancement for smart objects, as they become increasingly complex and connected.

Communication capabilities are improving: It’s no big surprise that wireless speeds are continually increasing, but they are also increasing in range. IoT is driving the development of more and more specialized communication protocols covering a greater diversity of use cases and environments.

Communication is being increasingly standardized: There is a strong push in the industry to develop open standards for IoT communication protocols. In addition, there are more and more opensource efforts to advance IoT

The following are some of the most significant limitations of the smart objects in WSNs:

Limited processing power

Limited memory

Lossy communication

Limited transmission speeds

Limited power

Control Units:

It is a unit of small computer on a single integrated circuit containing microprocessor or processing core, memory and programmable input/output devices/peripherals. It is responsible for major processing work of IoT devices and all logical operations are carried out here.

Bluetooth

Bluetooth is a standard for short range, low power, low cost wireless communication that uses radio technology. Although originally envisioned as a cable-replacement technology. Bluetooth technology can be used at home, in the office, in the car, etc. This technology allows to the users instantaneous connections of voice and information between several devices in real time. The way of transmission used assures protection against interferences and safety in the sending of information.

The Bluetooth is a small microchip that operates in a band of available frequency throughout the world. Communications can realize point to point and pointmultipoint. The standard Bluetooth operates in the band of 2,4 GHz. Though worldwide, this band is available, the width of the band can differ in different countries. This is the frequency of band of the scientific and medical industries 2.45 GHz (ISM*). The ranges of the bandwidth in The United States and Europe are between 2.400 to 2.483,5 MHz and it covers part of France and Spain. The ranges of the bandwidth in Japan are between 2.471 to 2.497 MHz.

User scenarios

Many different user scenarios can be imagined for wireless piconets or WPANs:

● Connection of peripheral devices: Most of the devices are connected to a desktop computer via wires (e.g., keyboard, mouse, joystick, headset, speakers).This type of connection has several disadvantages: each device has its own type of cable, different plugs are needed, and wires block office space. In a wireless network, no wires are needed for data transmission. However, batteries now have to replace the power supply, as the wires not only transfer data but also supply the peripheral devices with power.

● Support of ad-hoc networking: Imagine several people coming together,
Wireless networks can support interactive exchange of data as a group. Small devices might not have WLAN adapters following the IEEE 802.11 standard, but cheaper Bluetooth chips built in.

● Bridging of networks: Using wireless piconets, a mobile phone can be connected to a PDA or laptop in a simple way. Mobile phones will not have full WLAN adapters built in, but could have a Bluetooth chip. The mobile phone can then act as a bridge between the local piconet and, e.g., the global GSM network.

ARCHITECTURE OVERVIEW

Bluetooth link control hardware, integrated as either one chip or a radio module and a baseband module, implements the RF, baseband, and link manager portions of the Bluetooth specification. This hardware handles radio transmission and reception as well asrequired digital signal processing for the baseband protocol. Its functions include establishing

connections, support for asynchronous (data) and synchronous (voice) links, error correction, and authentication. The link manager firmware provided with the baseband CPU performs low-level device discovery, link setup, authentication, and link configuration.
Bluetooth operates on 79 channels in the 2.4 GHz band with 1 MHz carrier spacing. Eachdevice performs frequency hopping with1,600 hops/s in a pseudo random fashion. A piconet is a collection of Bluetooth devices which are synchronized to the same hopping sequence. One device in the piconet can act as master (M), all other devices connected to the master must act as slaves (S).

The master determines the hopping pattern in the piconet and the slaves have to synchronize to this pattern. Each piconet has a unique hopping pattern. If a device wants toparticipate it has to synchronize to this. Two additional types of devices are shown: parked devices (P) cannot actively participate in the piconet (i.e., they do not have a connection), but are known and can be reactivated within some milliseconds.

Devices in stand-by (SB) do not participate in the piconet. Each piconet has exactly one master and up to seven simultaneous slaves. More than 200 devices can be parked. The reason for the upper limit of eight active devices is the 3-bitaddress used in Bluetooth. If a parked device wants to communicate and there are already seven active slaves, one slave has to switch to park mode to allow the parked device to switch to active mode.

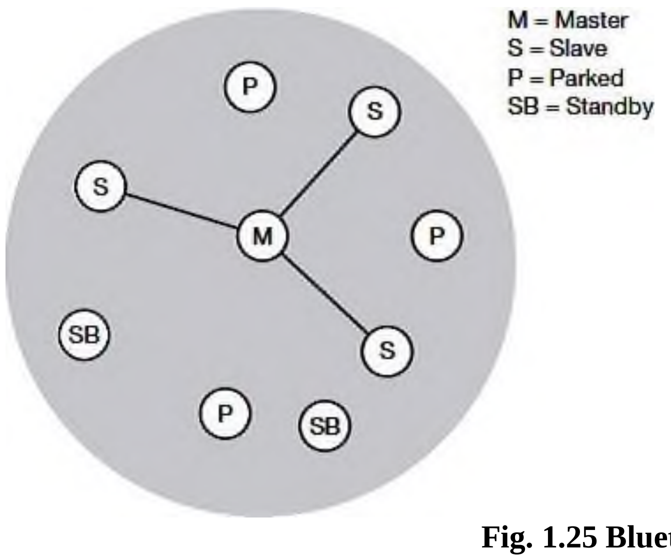


Fig. 1.25 Bluetooth piconet
[Source: Text book -Mobile Communications, Second Edition, Pearson Education by Jochen Schiller]

The Piconet are several devices that are in the same radio of coverage where they share the same channel and that is constituted between two and eight of these units. Every device has the unique direction of 48 bits, based on the standard IEEE 802.11 for WLAN, whereas the Scatternet formed by the connection of a Piconet to other one, with a maximum of interconnections of ten Piconets.

As all active devices have to use the same hopping sequence they must be synchronized. The first step involves a master sending its clock and device ID. All Bluetooth devices have the same networking capabilities, i.e., they can be master or slave. There is no distinction between terminals and base stations, any two or more devices can forma piconet.

The unit establishing the piconet automatically becomes the master, all other devices will be slaves. The phase in the hopping pattern is determined by the master‘s clock. After adjusting the internal clock according to the master a device may participate in the piconet. All active devices are assigned a3-bit active member address (AMA). All parked devices use an 8-bit parked member address (PMA). Devices in stand-by do not need an address.

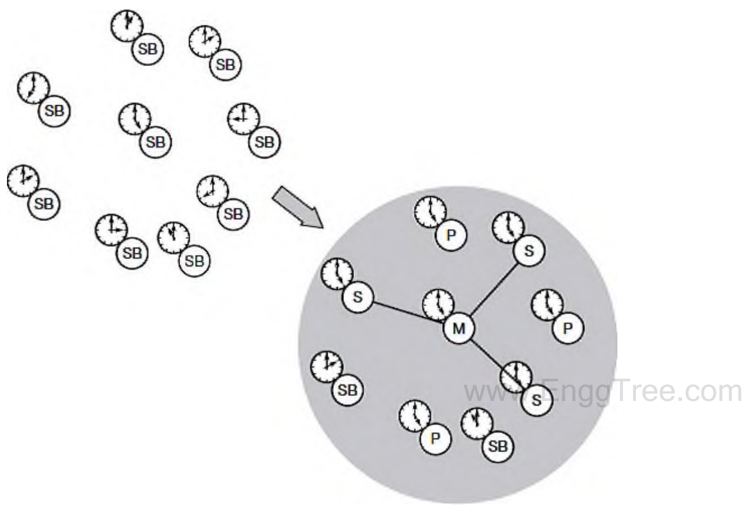


Fig. 1.26 Forming a Bluetooth piconet

[Source: Text book -Mobile Communications, Second Edition, Pearson Education by Jochen Schiller]

All users within one piconet have the same hopping sequence and share the same 1 MHz channel. As more users join the piconet, the throughput per user drops quickly (a single piconet offers less than 1 Mbit/s gross data rate).This led to the idea of forming groups of piconets called scatternet. If a device wants to participate in more than one piconet, it has to synchronize to the hopping sequence of the piconet it wants to take part in.

If a device acts as slave in one piconet, it simply starts to synchronize with the hopping sequence of the piconet it wants to join. After synchronization, it acts as a slave in this piconet and no longer participates in its former piconet. To enable synchronization, a slave has to know the identity of the master that determines the hopping sequence of a piconet. Before leaving one piconet, a slave informs the current master that it will be unavailable for a certain amount of time. There maining devices in the piconet continue to communicate as usual.

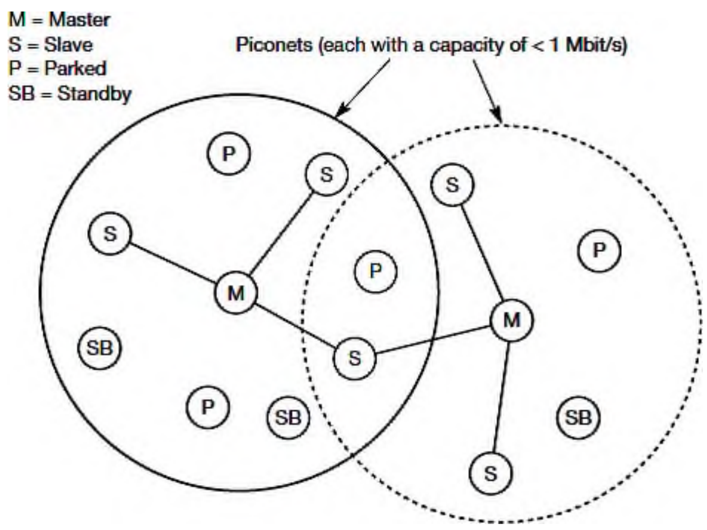


Fig. 1.27 Bluetooth scatternet

[Source: Text book -Mobile Communications, Second Edition, Pearson Education by Jochen Schiller]

Protocols Stack

The protocol architecture of the Bluetooth consists of following in a Bluetooth protocol stack:

• Core protocols consisting 5 layer protocol stack viz. radio, baseband, link managerprotocol, logical link control and adaptation protocol, service discovery protocol.

• Cable replacement protocol, RFCOMM

• Telephony Control Protocols

• Adopted protocols viz. PPP,TCP/UDP/IP,OBEX and WAE/WAPCore protocols

Radio: This protocol specification defines air interface, frequency bands, frequency hopping specifications, modulation technique used and transmits power classes. Baseband: Addressing scheme, packet frame format, timing and power control algorithms rquired for establishing connection between Bluetooth devices within piconet defined in this part of protocol specification.

Link Manager Protocol: It is responsible to establish link between Bluetooth devices and to maintain the link between them. This protocol also includes authentication and encryption specifications. Negotiation of packet sizes between devices can be taken care by this. Logical link control and adaptation protocol: This L2CAP protocol adapts upper layer frame to baseband layer frame format and vice versa. L2CAP take care of both connections oriented and connectionless services.

Service discovery protocol: Service related queries including device information can be taken care at this protocol so that connection can be established between Bluetoothdevices.

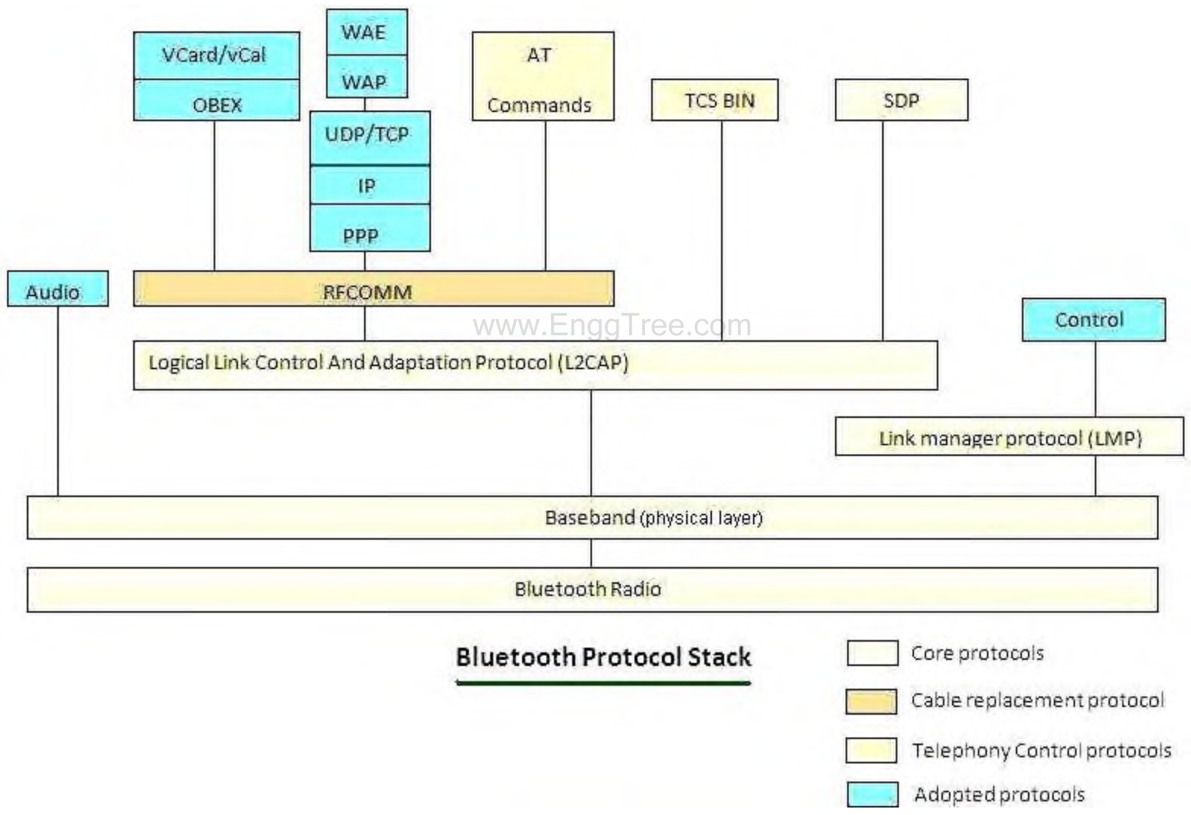

VCard/vCal
OBEX
WAE
WAP
UDP/TCP
IP
PPP
AT Commands
TCS BIN
SDP
Audio
RFCOMM
Control
Logical Link Control And Adaptation Protocol (L2CAP)
Link manager protocol (LMP)
Baseband (physical layer)
Bluetooth Radio
Bluetooth Protocol Stack
Core protocols
Cable replacement protocol
Telephony Control protocols
Adopted protocols

Fig.1.28 Bluetooth Protocol Stack

[Source: Text book -Mobile Communications, Second Edition, Pearson Education by Jochen Schiller]

Radio Layer

• The Bluetooth radio layer corresponds to the physical layer of OSI model. It deals with radio transmission and modulation. The radio layer moves data from master to slave or vice versa. It is a low power system that uses 2.4 GHz ISM band in a range of 10 meters.

• This band is divided into 79 channels of 1MHz each. Bluetooth uses the Frequency Hopping Spread Spectrum (FHSS) method in the physical layer to avoid interference from other devices or networks.

The radio specification defines the carrier frequencies and output power. Bluetooth devices will be integrated into typical mobile devices and rely on battery power. This requires small, low power chips which can be built into handheld devices. The combined use for data and voice transmission has to be reflected in the design, i.e., Bluetooth has to support multi-media data.

Bluetooth uses the license-free frequency band at 2.4 GHz allowing for worldwide operation with some minor adaptations to national restrictions. A frequency-hopping/time- division duplex scheme is used for transmission, with a fast hopping rate of 1,600 hops per second. The time between two hops is called a slot, which is an interval of 625 μs. Each slot uses a different frequency. In order to change bits into a signal, it uses a FSK with Gaussian bandwidth filtering.

Bluetooth transceivers use Gaussian FSK for modulation and are available inthree classes:

- Power class 1: Maximum power is 100 mW and minimum is 1 mW (typ. 100 m range without obstacles). Power control is mandatory.
- Power class 2: Maximum power is 2.5 mW, nominal power is 1 mW, and minimumpower is 0.25 mW (typ. 10 m range without obstacles). Power control is optional.
- Power class 3: Maximum power is 1 mW.

Baseband Layer

Baseband layer is equivalent to the MAC sublayer in LANs.

The baseband layer controls transmission of frames in association with frequency hopping. Master and slave stations communicate with each other using time slots. The master in a piconet takes the channel to transmit in even-numbered hops, and slaves transmit in odd-numbered hops, reflecting a time-division duplex for all devices in apiconet.

A single frame can be transmitted in the duration of one, three, or five hops. Depending on the nature of the logical link between a slave and the master, two types of links are offered. Bluetooth uses a form of TDMA called TDD-TDMA (time division duplex TDMA).The master in each piconet defines the time slot of 625 μsec.

In TDD- TDMA, communication is half duplex in which receiver can send and receive data but not at the same time. If the piconet has only no slave; the master uses even numbered slots (0, 2, 4, ...) and the slave uses odd-numbered slots (1, 3, 5,). Both master and slave communicate in half duplex mode. In slot 0, master sends & secondary receives; in slot 1, secondary sends and primary receives. If piconet has more than one slave, the master uses even numbered slots. The slave sends in the next odd-numbered slotif the packet in the previous slot was addressed to it.

The baseband layer has defined some types of frames that correspond to various purposes of the baseband frames. Different types of frames can carry different sizes of payload data and error-correction schemes. In particular, the access code field in a baseband frame indicates the purpose of the frame in a special state. For example, a frame with the inquiry access code (IAC) will be sent when a device elects to scan for other devices within the radio range in a series of 32 frequency hops.

Bluetooth devices can be configured to periodically hop according to the inquiry scan hopping sequence to scan inquires. When an inquiry is detected, the device, now the slave, will reply with its address and timing information to the master, and then the master and the slave begin the paging process to determine a common hopping sequence to establish a connection. Eventually, both the master and the slave will hop on the same sequence of channels for the duration of the connection.

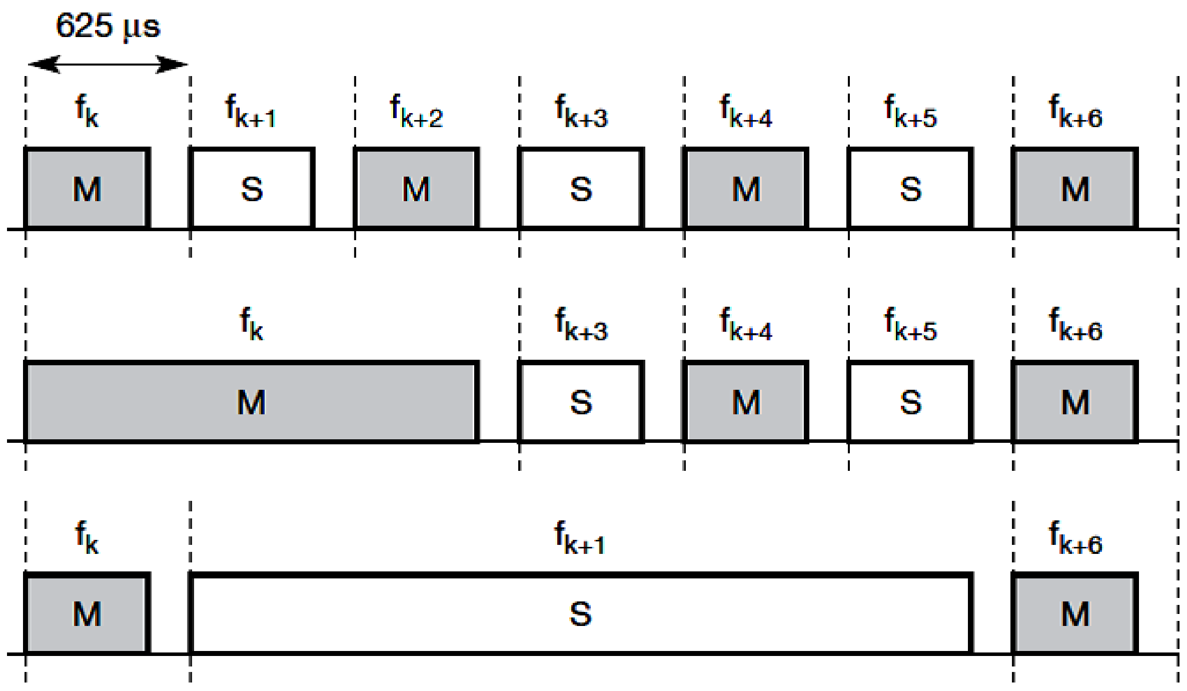


Fig. 1.29 Frequency selection during data transmission using 1, 3, 5 packet slots

[Source: Text book -Mobile Communications, Second Edition, Pearson Education by Jochen Schiller]

Link manager protocol

The Link Manager (LM) translates the commands into operations at the Baseband level, managing the following operations.

1) Attaching slaves to piconets, and allocating their active member addresses.
2) Breaking connections to detach Slaves from a piconet.
3) Configuring the link including Master/Slave switches
4) Establishing ACL and SCO links.
5) Putting connections into Low Power modes: Hold, Sniff and Park.
6) Controlling test modes.

A bluetooth Link Manager communicates with Link Managers on other Bluetooth devices using the Link Management protocol (LMP).

The link can be configured at any time, including at mode changes, quality of service changes, packet type changes and any power level changes. Finally, information about an active link can be retrieved at any time. When the connection is no longer required, LMP can cause disconnection.

The link manager protocol (LMP) manages various aspects of the radio link between a master and a slave and the current parameter setting of the devices. LMP enhances baseband functionality, but higher layers can still directly access the baseband. The following groups of functions are covered by the LMP:

• Authentication, pairing, and encryption: Although basic authentication is handled in the baseband, LMP has to control the exchange of random numbers and signed responses. The pairing service is needed to establish an initial trust relationship between two devices that have never communicated before.

The result of pairing is a link key. This may be changed, accepted or rejected. LMP is not directly involved in the encryption process, but sets the encryption mode (no encryption, point-to-point, or broadcast), key size, and random speed.

• Synchronization: Precise synchronization is of major importance within a Bluetooth network. The clock offset is updated each time a packet is received from the master. Additionally, special synchronization packets can be received. Devices can also exchange timing information related to the time differences (slot boundaries) between two adjacent piconets.

• Capability negotiation: Not only the version of the LMP can be exchanged but also information about the supported features. Not all Bluetoothdevices will support all features that are described in the standard, so devices have to agree the usage of, e.g., multi-slot packets, encryption, SCO links, voice encoding, park/sniff/hold mode, HV2/HV3packets etc.

• Quality of service negotiation: Different parameters control the QoS of a Bluetooth device at these lower layers. The poll interval, i.e., the maximum time between transmissions from a master to a particular slave, controls the latency and transfer capacity.Depending on the quality of the channel, DMor DH packets may be used (i.e., 2/3 FEC protection or no protection). The number of repetitions for broadcast packets can be controlled. A master can also limit the number of slots available for slaves‘ answers to increase its own bandwidth.

• Power control: A Bluetooth device can measure the received signal strength. Dependingon this signal level the device can direct the sender of the measured signal to increase or decrease its transmitting power.

ZIGBEE

Zigbee is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power wireless IoT networks. The Zigbee standard operates on the IEEE 802.15.4 physical radio specification and operates in unlicensed bands including 2.4 GHz, 900 MHz and 868 MHz.

The 802.15.4 specification upon which the Zigbee stack operates gained ratification by the Institute of Electrical and Electronics Engineers (IEEE) in 2003. The specification is a packet-based radio protocol intended for low-cost, battery-operated devices. The protocol allows devices to communicate in a variety of network topologies and can have battery life lasting several years.

The Zigbee 3.0 Protocol

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

ZIGBEE ADVANTAGE

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

Zigbee protocol features include:

- Support for multiple network topologies such as point-to-point, point-to-multipoint and mesh networks
- Low duty cycle – provides long battery life
- Low latency
- Direct Sequence Spread Spectrum (DSSS)
- Up to 65,000 nodes per network

- 128-bit AES encryption for secure data connections
- Collision avoidance, retries and acknowledgements

The Zigbee 3.0 software stack incorporates a ‘base device’ that provides consistent behavior for commissioning nodes into a network. A common set of commissioning methods is provided, including Touchlink, a method of proximity commissioning.

Zigbee 3.0 provides enhanced network security. There are two methods of security that give rise to two types of network:

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

Nodes adopt whichever security method is used by the network they join. Zigbee 3.0 supports the increasing scale and complexity of wireless networks, and copes with large local networks of greater than 250 nodes. Zigbee also handles the dynamic behavior of these networks (with nodes appearing, disappearing and re-appearing in the network) and allows orphaned nodes, which result from the loss of a parent, to re-join the network via a different parent. The self-healing nature of Zigbee Mesh networks also allows nodes to drop out of the network without any disruption to internal routing.

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

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

Mesh Networks

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

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

Zigbee Applications

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

Wi-Fi

Wi-Fi stands for Wireless Fidelity or Frequencies, which allows multiple computers to communicate and provides a means to connect to the Internet from the access point to the computer or laptop. Wi-Fi networking technology combines and transmits Data and Information between devices using different bands of radio waves. Wi-Fi is a widely used technology in today's smartphones and PCs.

Like a mobile phone, a Wi-Fi network uses **Radio Waves** to send data across a network. The computer should include a wireless adaptor that converts data transferred to a radio signal. The identical signals deliver to a router decoder through an **Antenna**. After decoding, the data is sent to the Internet over a connected Ethernet connection.
Because the wireless network is bidirectional, data from the Internet will also transit via the router and coded into a radio signal that the computer's wireless adapter will receive.

Wi-Fi In IoT

Because Wi-Fi is so vulnerable to malicious assaults, a microchip is required for connectivity between devices in the IoT and robust firmware to maintain the device's Wi-Fi credentials. Wi-Fi-enabled IoT devices are frequently massive immovable hubs. There are, however, smaller gadgets that are Wi-Fi capable. If we want to use Wi-Fi, the Wi-Fi IoT device must be reasonably close to the Wi-Fi access point.

WiFi-enabled IoT vs. Bluetooth-enabled IoT.

Speed

In terms of speed, Wi-Fi offers a maximum speed faster than Bluetooth IoT. Wi-Fi IoT devices have a minimum data rate of 54 Mbps, whereas Bluetooth devices have a data rate of just 3 Mbps. The reason is that Bluetooth is better for delivering tiny data files, such as numerical numbers, from a Bluetooth-enabled IoT timepiece. At the same time, Wi-Fi is better for sending essential data files, such as HD films and photographs.

Location detection

Through the Bluetooth IoT and Wi-Fi IoT devices to which they are linked, Wi-Fi and Bluetooth may properly transmit location information. On the other hand, Bluetooth is more dependable due to its closeness. In this scenario, the better alternative is determined by the accuracy and precision required by the equipment in use.

Security

Although Bluetooth does not have a secure IoT protocol, the available security is sufficient for most uses. On the other hand, Wi-Fi offers a safer choice, especially useful when working with sensitive data.

Proximity detection

The proximity data offered by BLE in IoT is substantially more exact than that provided by its Wi-Fi equivalent in terms of proximity detection. It is crucial to note that while neither option guarantees 100 percent accuracy, the Bluetooth option is preferred.

Advantages of Wi-Fi in IoT

1. We can transport a wireless laptop from one location to another.
2. We can reduce the expense of cables by using wireless network communication devices.
3. Wi-Fi setup and configuration are much more straightforward than wiring.
4. It is fully secure and will not disrupt any network.
5. We may also use hot spots to connect to the Internet.
6. Wireless internet access is possible.

Disadvantages of Wi-Fi

1. Wi-Fi emits radiation that is harmful to human health.
2. When we are not utilizing the server, we must terminate the Wi-Fi connection.
3. There are certain limitations to data transfer; we cannot transport data across great distances.
4. When compared to a conventional connection, Wi-Fi deployment is more costly.

Applications of Wi-Fi

1. Apps for smartphones
2. Applications for business
3. Applications for the home
4. Computerized software
5. Automotive industry
6. Video conferencing while surfing the Internet

Global Positioning System (GPS)

The Global Positioning System (GPS) is a navigation system that allows users to determine their exact location on the earth's surface. GPS has become an essential tool for a variety of applications, including navigation, surveying, mapping, and tracking.

GPS uses satellites to monitor the movement of anything equipped with such a GPS tracking device, including automobiles, humans, and even pets. It operates in any weather condition and offers precise location updates in real-time. As one of the earliest ways to track and disseminate digital information from the real environment, GPS has significantly impacted IoT technology. The Internet of Things (IoT) may gather and measure enormous amounts of data on anything from individual health to public transportation; GPS tracking is required to provide location information for such objects.

A more reliable and easily accessible data set can be built using GPS and the Internet of Things. In the same way that GPS pinpoints the precise location of a vehicle, the Internet of Things is able to monitor moving items and collect data on their movements in real time.

How Does GPS Function

GPS satellites complete two accurate orbits around the planet every day. An individual satellite's signal and orbiting parameters can be decoded and used to pinpoint the satellite's location via a GPS receiver. This data, together with triangulation, is used by GPS receivers to pinpoint the precise location of its owners.

The uses of GPS and the Internet of Things

When applied to Internet of Things (IoT) gadgets, GPS technology might provide benefits you might not have anticipated. In the age of the Internet of Things, it is possible to amass vast amounts of data from a wide variety of sources. Information such as medical files and facial mapping is included.

Up until now, we've been able to find our misplaced smartphones by using their GPS systems. We can now locate misplaced suitcases. Symphonia Bags are high-tech backpacks that can be tracked via GPS. When their children are not in sight, parents can use the backpack to track them down. With this bag, you won't have to worry about leaving your belongings behind at the airport or on the road. No one can deny their eager anticipation of the day when they can hop in a driverless automobile and kick their feet up while it whisks them away. There's a chance that today's youth will never use their driving skills. Automobiles equipped with GPS technology will be fully autonomous.

Benefits of GPS Integration in IoT

The integration of GPS with IoT can help organizations to optimize their operations and simplify complex processes, reducing downtime and increasing efficiency. Tracking vehicles in real-time can reduce transport time, simplify logistical planning, and optimize delivery schedules.

Challenges and Limitations of GPS in IoT

Despite the benefits of GPS-enabled IoT devices, there are also challenges and limitations that need to be addressed. The accuracy of GPS tracking is limited and is affected by physical and environmental factors such as signal interference, weather, and the location of the receiver. Issues with privacy and data security must also be addressed when implementing GPS tracking in IoT devices.

Real-World Applications of GPS in IoT

Smart Transportation and Fleet Management

GPS technology is widely used in many transportation and logistics applications, such as tracking cargo shipments and optimizing distribution routes in real-time. Fleet managers can monitor and analyze vehicle performance, fuel consumption, and driver data. This information helps them to increase operational efficiency, reduce costs, and improve safety.

Precision Agriculture and Environmental Monitoring

GPS technology is also useful in precision agriculture and environmental monitoring. GPS receivers on drones, for instance, allow farmers to collect data on crop growth and soil conditions, improving yields and reducing costs. The technology can also be used to monitor environmental factors such as water levels and air or water quality, allowing for informed conservation and sustainability practices.

Personal Tracking and Wearable Devices

GPS technology in wearable devices, such as smartwatches and fitness trackers, can track personal health and activity data. The accurate location information can enable people to monitor exercise routines, sleep patterns, and nutrition, leading to better health outcomes.

Finding the Right Balance

GPS technology in IoT devices is transforming the way we live and work. In instances such as natural disasters, medical emergencies, and search and rescue operations, these devices can help save lives. Despite some limitations and concerns, GPS in IoT has enormous potential to transform industries and provide new benefits. It is therefore essential to find the right balance between utilization and responsible management of the technology to realize its full potential.

Global System for Mobile Communications

GSM stands for Global System for Mobile Communications. It's a standard that specifies how 2G (second generation) cellular networks operate. GSM was a significant improvement over the first generation of cellular networks and represented a transition from analog to digital telecommunications.

GSM is currently the most widely used network technology in Internet of Things (IoT) applications for its simplicity, affordability, and accessibility. But that's likely to change over the next few years.

When the Global System for Mobile Communications was first introduced in Europe in 1991, these 2G networks created faster, more secure wireless connections. For the first time, voice communications became encoded into digital signals before being transmitted through the network.

GSM reigned for years as the world's most widely used standard for mobile communications. But today, 2G networks are significantly slower than other cellular networks, and in several countries 2G networks are being switched off.

Mobile Network Operators (MNOs) are competing to balance the fastest speeds with the best coverage. With decades of built up infrastructure, GSM-based networks can offer good coverage, but they can't compete with the speed, versatility, and security of 3G, 4G, and 5G networks.

Additionally, GSM standards were designed with cell phones in mind—not the Internet of Things (IoT). Today, billions of other devices like parking meters, industrial equipment, car entertainment systems, and security systems rely on cellular networks and use them in different ways than phones do. As a result, specialized networks have emerged to address the modern landscape of cellular connectivity.

Structure of GSM networks

GSM standards divide networks into four distinct parts:

1. Mobile Station
2. Base Station Subsystem (BSS)
3. Network and Switching Subsystem (NSS)
4. Operations Support System (OSS)

Each part of the network contains several components. Together these components form one complete cellular network. Every cellular provider has their own infrastructure with all of these pieces.

Mobile Station

The Mobile Station is essentially the access point someone uses to connect to the network. It's a device (such as an alarm system) with a Subscriber Identity Module (SIM). The SIM associates the device with an individual subscriber, which allows the device to connect to the nearest Base Station Subsystem.

Base Station Subsystem (BSS)
The BSS contains Base Transceiver Stations and a Base Station Controller. The Base Transceiver Stations include components like receivers and antenna, which allows connected devices to send and receive signals, and the Base Station Controller allows the Base Transceiver Stations to relay signals through the network, via the Network and Switching Subsystem.

Network and Switching Subsystem (NSS)
The Network and Switching Subsystem is a term for the major components of a 2G core network. The NSS originally helped facilitate connection-oriented voice calls with the Home Location Register (HLR), Authentication Center (AuC), Message Service Center (MSC), and Visitor Location Register (VLR).

With the introduction of the GPRS core network and its support nodes (GGSN and SGSN), the NSS began playing a role in data connections as well.

Operations Support System (OSS)
The Operations Support System is a conglomeration of processes, data, applications, and tech that allows providers to manage their network. Carriers can use their OSS to:

1. Configure network elements
2. Manage and configure the services they offer
3. Handling system errors and managing the system's state
4. Monitor performance based on quality of service and quality of experience KPIs

More advanced cellular networks have similar structures, but with additional components to improve the networks' security and capabilities.

Is GSM still useful?

GSM networks are now three decades old, and there are three generations of cellular networks with far higher data transfer rates, more secure connections, and advanced networking capabilities. Over the years, telecommunications organizations have implemented upgrades to get more mileage out of GSM-based networks, but in several countries 2G is coming to an end.

This doesn't have much impact on consumers, as phones usually support multiple technologies. But GSM has been one of the most popular connectivity choices in cellular IoT. Modern IoT manufacturers need to evaluate whether 2G connectivity is still a viable option for their application in the region where they want to deploy.

Choose the right connectivity for your application

GSM played a foundational role in modern cellular communications. And while some operators are transitioning to newer networks, this technology is still immensely popular for its global availability and extremely low-cost connectivity. As operators expand their infrastructure for affordable alternatives like LTE-M and NB-IoT, 2G will become less relevant. But until then, it's still an attractive solution for many cellular IoT applications.

PROTOCOLS AND TECHNOLOGIES BEHIND IOT

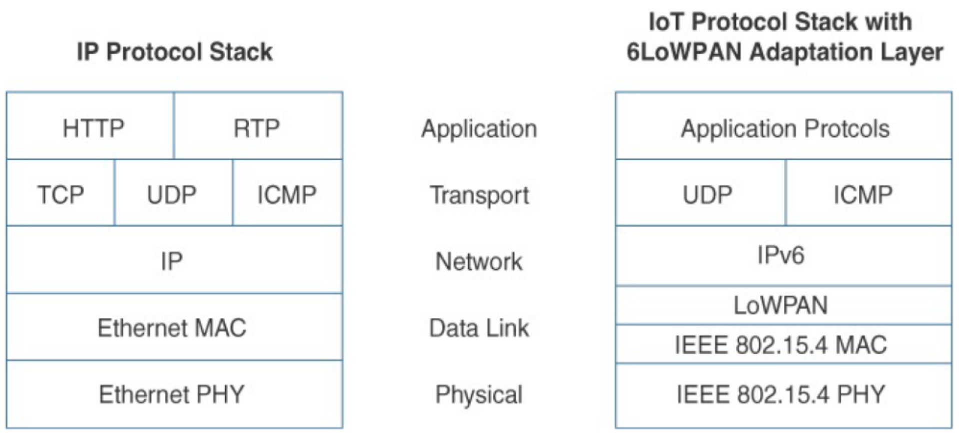
IOT Protocols

Data flowing from or to "things" is consumed, controlled, or monitored by data center servers either in the cloud or in locations that may be distributed or centralized. Dedicated applications are then run over virtualized or traditional operating systems or on network edge platforms (for example, fog computing). These lightweight applications communicate with the data center servers. Therefore, the system solutions combining various physical and data link layers call for an architectural approach with a common layer(s) independent from the lower (connectivity) and/or upper (application) layers.

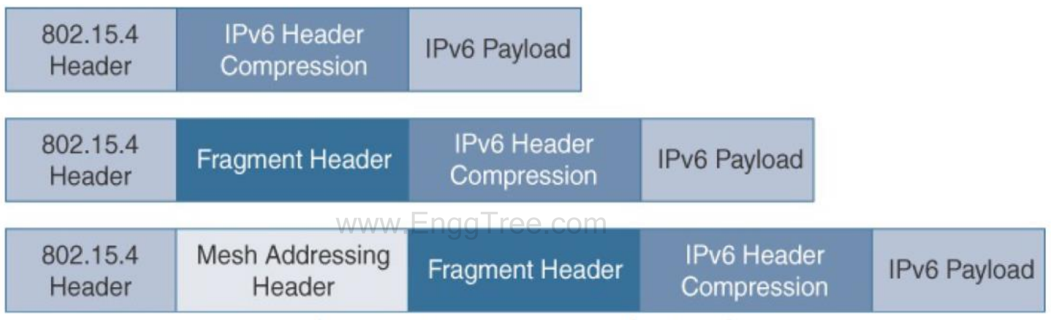
IPv6:

IPv6 Adaptation Layer: IPv6-only adaptation layers for some physical and data link layers for recently standardized IoT protocols support only IPv6. While the most common physical and data link layers (Ethernet, Wi-Fi, and so on) stipulate adaptation layers for both versions, newer technologies, such as IEEE 802.15.4 (Wireless Personal Area Network), IEEE 1901.2, and ITU G.9903 (Narrowband Power Line Communications) only have an IPv6 adaptation layer specified. This means that any device implementing a technology that requires an IPv6 adaptation layer must communicate over an IPv6-only subnetwork. This is reinforced by the IETF routing protocol for LLNs, RPL, which is IPv6 only.

6LoWPAN



Comparison of an IoT Protocol Stack Utilizing6LoWPAN and anIP Protocol Stack



6LoWPAN Header Stacks

Header Compression

IPv6 header compression for 6LoWPAN was defined initially in RFC 4944 and subsequently updated by RFC 6282. This capability shrinks the size of IPv6's 40-byte headers and User Datagram Protocol's (UDP's) 8-byte headers down as low as 6 bytes combined in some cases.

Note that header compression for 6LoWPAN is only defined for an IPv6 header and not IPv4.The 6LoWPAN protocol does not support IPv4, and, in fact, there isno standardized IPv4 adaptation layer for IEEE 802.15.4.

6LoWPAN header compression is stateless, and conceptually it is not too complicated. However, a number offactors affect the amount of compression, such as implementation of RFC 4944 versus RFC 6922, whether UDP is included, and various IPv6 addressing scenarios. It is beyond the scope of this book to cover every use case and how the header fields change for each. However, this chapter provides an example that shows the impact of 6LoWPAN header compression.

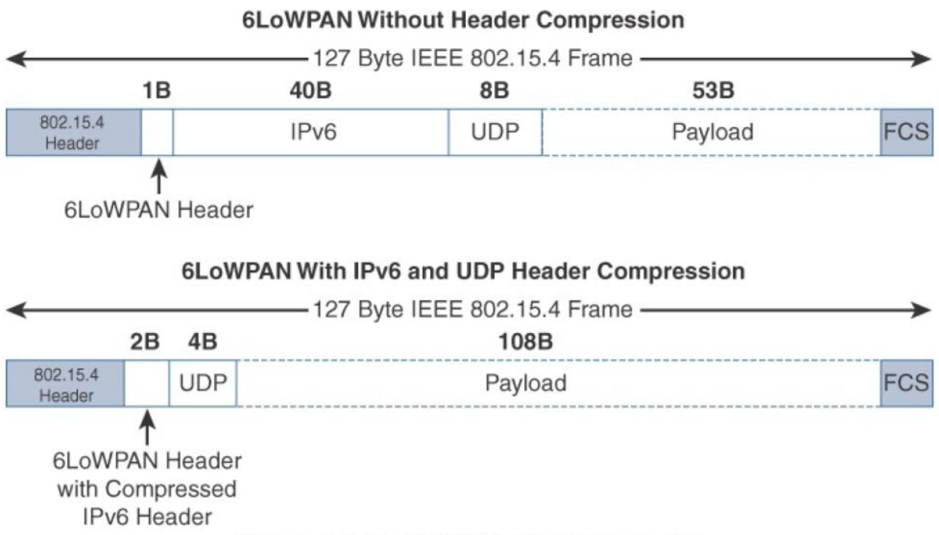

6LoWPAN Without Header Compression
127 Byte IEEE 802.15.4 Frame
1B
40B
8B
53B
802.15.4 Header
IPv6
UDP
Payload
FCS
6LoWPAN Header
6LoWPAN With IPv6 and UDP Header Compression
127 Byte IEEE 802.15.4 Frame
2B
4B
108B
802.15.4 Header
UDP
Payload
FCS
6LoWPAN Header with Compressed IPv6 Header

Figure 5-4 *6LoWPAN Header Compression*

Fragmentation

The maximum transmission unit (MTU) for an IPv6 network must be at least 1280 bytes. The term *MTU* defines the size of the largest protocol data unit that can be passed. For IEEE 802.15.4, 127 bytes is the MTU. You can see that this is a problem because IPv6, with a much larger MTU, is carried inside the 802.15.4 frame with a much smaller one. To remedythis situation, large IPv6 packets must be fragmented across multiple 802.15.4 frames at Layer2.

The fragment header utilized by 6LoWPAN is composed of three primary fields: Datagram Size, Datagram Tag, and Datagram Offset. The 1-byte Datagram Size field specifies the total size of the unfragmented payload. Datagram Tag identifies the set of fragments for a payload. Finally, the Datagram Offset field delineates how far into a payload a particular fragment occurs.

Mesh Addressing

The purpose of the 6LoWPAN mesh addressing function is to forward packets over multiple hops. Three fields are definedfor this header: Hop Limit, Source Address, and Destination Address. Analogous to the IPv6 hop limit field, the hop limit for mesh addressing also provides an upper limit on how manytimes the frame can be forwarded. Each hop decrements this value by 1 as it is forwarded. Once the value hits 0, it is dropped and no longer forwarded.

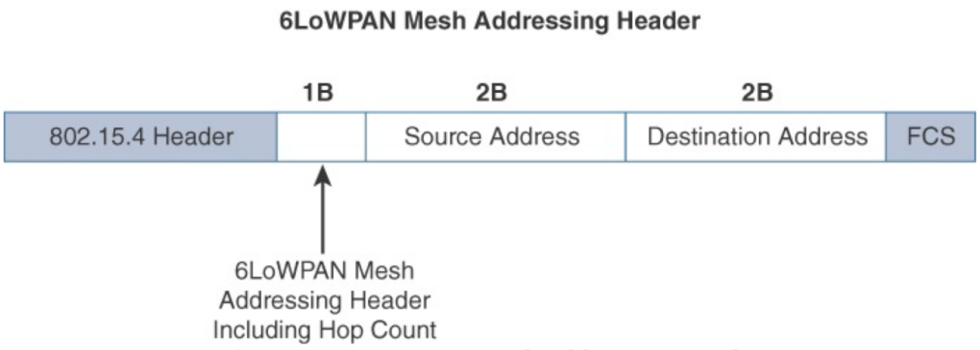

6LoWPAN Mesh Addressing Header
1B
2B
2B
802.15.4 Header
Source Address
Destination Address
FCS
6LoWPAN Mesh Addressing Header Including Hop Count

Figure 5-6 *6LoWPAN Mesh Addressing Header*

6LoWPAN

The 6LoWPAN system is used for a variety of applications including wireless sensor networks. This form of wireless sensor network sends data as packets and using IPv6 - providing the basis for the name - IPv6 over Low power Wireless Personal Area Networks.

6LoWPAN provides a means of carrying packet data in the form of IPv6 over IEEE 802.15.4 and other networks. It provides end-to-end IPv6 and as such it is able to provide direct connectivity to a huge variety of networks including direct connectivity to the Internet.

In this way, 6LoWPAN adopts a different approach to the other low power wireless sensor network solutions.

6LoWPAN and IETF

6LoWPAN is an open standard defined by the Internet Engineering Task Force, IETF in their document RFC 6282. The IETF is the standards body that defines many of the open standards used in the Internet including HTTP, TCP, UDP and many others.

Whilst 6LoWPAN was originally conceived to build on top of IEEE 802.15.4, a standard that set out the lower layers for a 2.4 GHz low power wireless system, it is now being developed and adapted to work with many other wireless bearers including Bluetooth Smart; power line control, PLC, and low power Wi-Fi.

The 6LoWPAN group have then defined the encapsulation and compression mechanisms that enable the IPv6 data to be carried of the wireless network.

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

6LoWPAN application areas

With many low power wireless sensor networks and other forms of ad hoc wireless networks, it is necessary that any new wireless system or technology has a defined area which it addresses. While there are many forms of wireless networks including wireless sensor networks, 6LoWPAN addresses an area that is currently not addressed by any other system, i.e. that of using IP, and in particular IPv6 to carry the data.

The overall system is aimed at providing wireless internet connectivity at low data rates and with a low duty cycle. However there are many applications where 6LoWPAN is being used:

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

- ***Industrial monitoring:*** Automated factories and industrial plants provide a great opportunity for 6LoWPAN and using automation, can enable major savings to be made. The ability of 6LoWPAN to connect to the cloud opens up many different areas for data monitoring and analysis.

6LoWPAN basics

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

In order to send packet data, IPv6 over 6LowPAN, it is necessary to have a method of converting the packet data into a format that can be handled by the IEEE 802.15.4 lower layer system.

IPv6 requires the maximum transmission unit (MTU) to be at least 1280 bytes in length. This is considerably longer than the IEEE802.15.4's standard packet size of 127 octets which was set to keep transmissions short and thereby reduce power consumption.

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

6LoWPAN security

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

Accordingly security is a major issue for any standard like 6LoWPAN, and it uses AES-128 link layer security which is defined in IEEE 802.15.4. This provides link authentication and encryption.

Further security is provided by the transport layer security mechanisms that are also included. This is defined in RFC 5246 and runs over TCP.

For systems where UDP is used the transport layer protocol defined under RFC 6347 can be used, although this may require some specific hardware requirements.

6LoWPAN interoperability

One key issue of any standard is that of interoperability. It is vital that equipment from different manufacturers operates together.

When testing for interoperability, it is necessary to ensure that all layers of the OSI stack are compatible. To ensure that this can be achieved there several different specifications that are applicable.

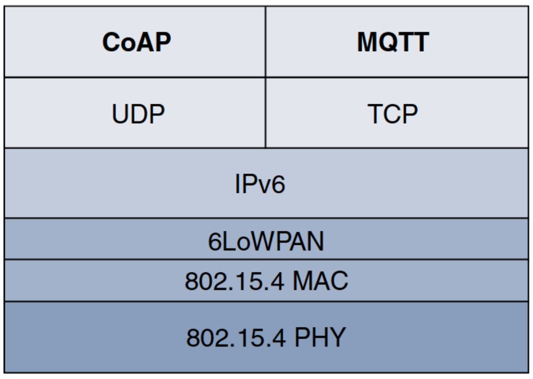
Any item can be checked to conform it meets the standard, and also directly tested for interoperability.

S

6LoWPAN is a wireless / IoT style standard that has quietly gained significant ground. Although initially aimed at usage with IEEE 802.15.4, it is equally able to operate with other wireless standards making it an ideal choice for many applications.

6LoWPAN uses IPv6 and this alone has to set it aside from the others with a distinct advantage. With the world migrating towards IPv6 packet data, a system such 6LoWPAN offers many advantages for low power wireless sensor networks and other forms of low power wireless networks.

Message Queuing Telemetry Transport (MQTT)



Example of a High-Level IoT Protocol Stack for CoAP and 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).

Considering the harsh environments in the oil and gas industries, an extremely simple protocol with only a few options was designed, with considerations for constrained nodes, unreliable WAN backhaul communications, and bandwidth constraints with variable latencies. These were some of the rationales for the selection of a client/server and publish/subscribe framework based on the TCP/IP architecture, as shown in Figure

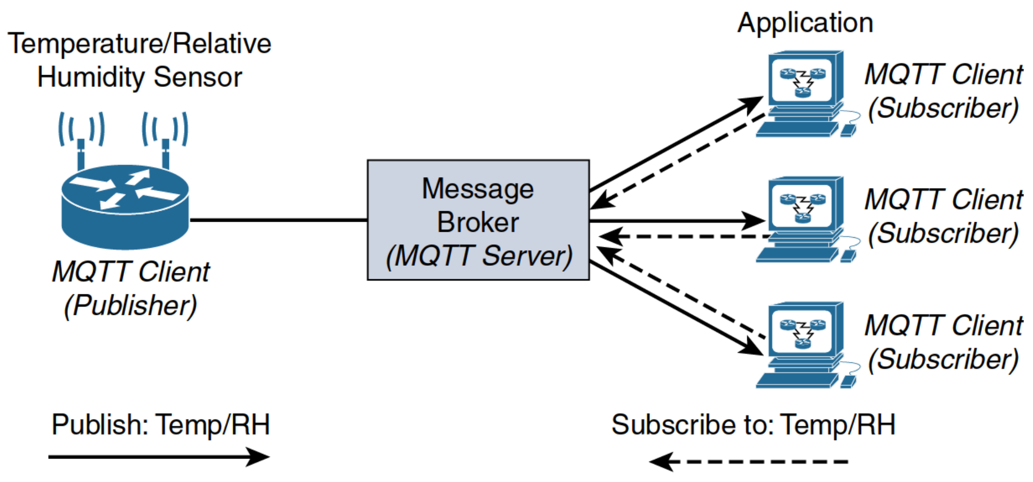


Fig: MQTT Publish/Subscribe Framework

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 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.

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. MQTT is a lightweight protocol because each control packet consists of a 2-byte fixed header with optional variable header fields and optional payload.

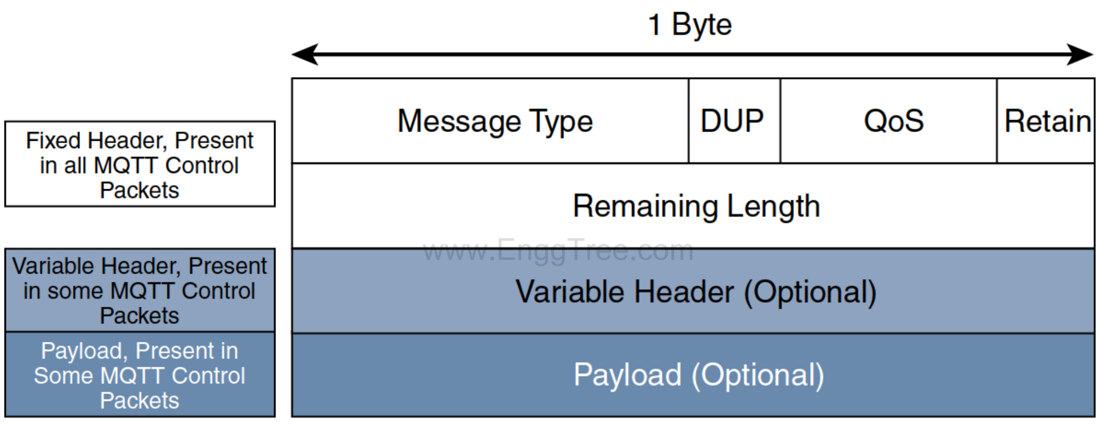

1 Byte
Message Type
DUP
QoS
Retain
Remaining Length
Variable Header (Optional)
Payload (Optional)
Fixed Header, Present in all MQTT Control Packets
Variable Header, Present in some MQTT Control Packets
Payload, Present in Some MQTT Control Packets
www.EnggTree.com

Fig: MQTT Message Format

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 sub- scribers 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.

These are the three levels of MQTT QoS:

- QoS 0: This is a best-effort and unacknowledged data service referred to as "at most once" delivery. The publisher sends its message one time to a server, which transmits it once to the subscribers. No response is sent by the receiver, and no retry is per- formed by the sender. The message arrives at the receiver either once or not at all.
- QoS 1: This QoS level ensures that the message delivery between the publisher and server and then between the server and subscribers occurs at least once. In PUBLISH and PUBACK packets, a packet identifier is included in the variable header. If the message is not acknowledged by a PUBACK packet, it is sent again. This level guarantees "at least once" delivery.
- QoS 2: This is the highest QoS level, used when neither loss nor duplication of messages is acceptable. There is an increased overhead associated with this QoS level because each packet contains an optional variable header with a packet identifier. Confirming the receipt of a PUBLISH message requires a two-step acknowledgement process. The first step is done through the PUBLISH/PUBREC packet pair, and the second is achieved with the PUBREL/PUBCOMP packet pair. This level provides a "guaranteed service" known as "exactly once" delivery, with no consideration for the number of retries as long as the message is delivered once.

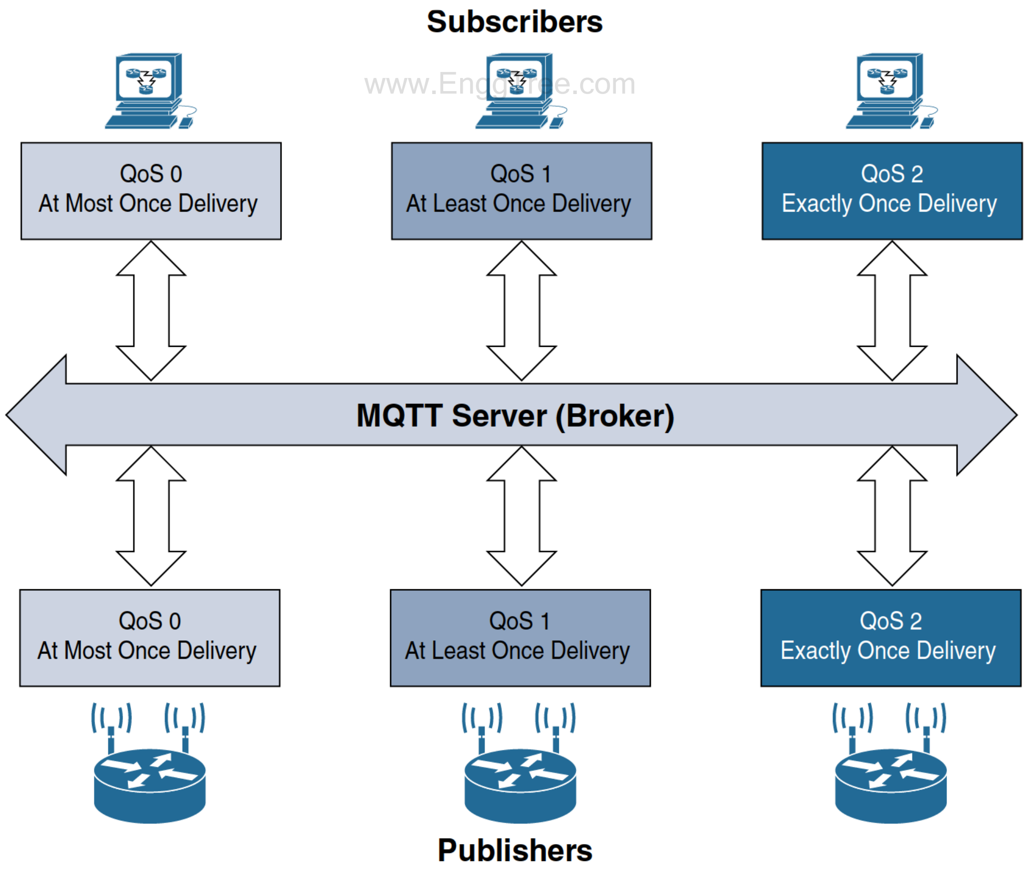


Fig: MQTT QoS Flows

Constrained Application Protocol (CoAP)

The CoAP framework defines simple and flexible ways to manipulate sensors and actuators for data or device management. The IETF CoRE working group has published multiple standards-track specifications for CoAP, including the following:

- **RFC 6690:** Constrained RESTful Environments (CoRE) Link Format
- **RFC 7252:** The Constrained Application Protocol (CoAP)
- **RFC 7641:** Observing Resources in the Constrained Application Protocol (CoAP)
- **RFC 7959:** Block-Wise Transfers in the Constrained Application Protocol (CoAP)
- **RFC 8075:** Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)

The CoAP messaging model is primarily designed to facilitate the exchange of messages over UDP between endpoints, including the secure transport protocol Datagram Transport Layer Security (DTLS). (UDP is discussed earlier in this chapter.) The IETF CoRE working group is studying alternate transport mechanisms, including TCP, secure TLS, and WebSocket. CoAP over Short Message Service (SMS) as defined in Open Mobile Alliance for Lightweight Machine-to-Machine (LWM2M) for IoT device management is also being considered.

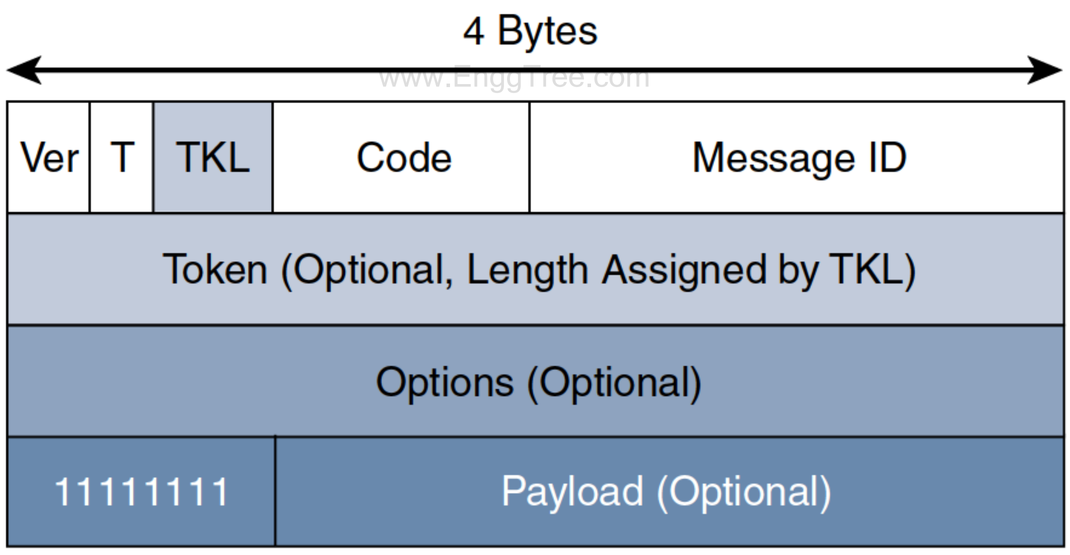


Fig: CoAP Message Format

Table 6-1 *CoAP Message Fields*

CoAP Message Field	Description
Ver (Version)	Identifies the CoAP version.
T (Type)	Defines one of the following four message types: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), or Reset (RST). CON and ACK are highlighted in more detail in Figure 6-9.
TKL (Token Length)	Specifies the size (0–8 Bytes) of the Token field.
Code	Indicates the request method for a request message and a response code for a response message. For example, in Figure 6-9, GET is the request method, and 2.05 is the response code. For a complete list of values for this field, refer to RFC 7252.
Message ID	Detects message duplication and used to match ACK and RST message types to Con and NON message types.
Token	With a length specified by TKL, correlates requests and responses.
Options	Specifies option number, length, and option value. Capabilities provided by the Options field include specifying the target resource of a request and proxy functions.
Payload	Carries the CoAP application data. This field is optional, but when it is present, a single byte of all 1s (0xFF) precedes the payload. The purpose of this byte is to delineate the end of the Options field and the beginning of Payload.

CoAP can run over IPv4 or IPv6. However, it is recommended that the message fit within a single IP packet and UDP payload to avoid fragmentation. For IPv6, with the default MTU size being 1280 bytes and allowing for no fragmentation across nodes, the maximum CoAP message size could be up to 1152 bytes, including 1024 bytes for the payload.

CoAP defines four types of messages: confirmable, non-confirmable, acknowledge- ment, and reset. Method codes and response codes included in some of these messages make them carry requests or responses. CoAP code, method and response codes, option numbers, and content format have been assigned by IANA as Constrained RESTful Environments (CoRE) parameters.

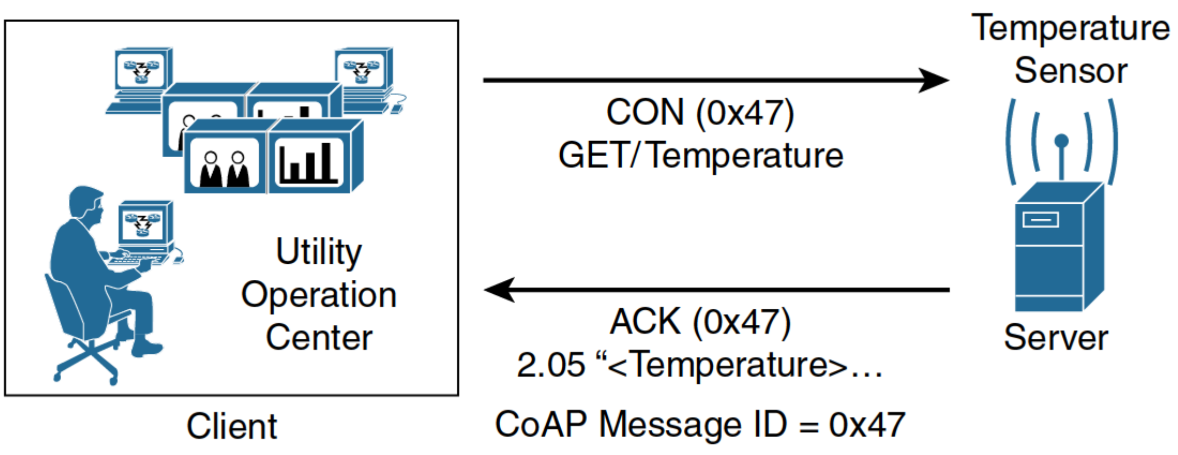


Fig:CoAP Reliable Transmission Example

Figure shows a utility operations center on the left, acting as the CoAP client, with the CoAP server being a temperature sensor on the right of the figure. The communication between the client and server uses a CoAP message ID of 0x47. The CoAP Message ID ensures reliability and is used to detect duplicate messages.
The client in Figure sends a GET message to get the temperature from the sensor.
Notice that the 0x47 message ID is present for this GET message and that the message is also marked with CON. A CON, or confirmable, marking in a CoAP message means the message will be retransmitted until the recipient sends an acknowledgement (or ACK) with the same message ID.

In Figure the temperature sensor does reply with an ACK message referencing the correct message ID of 0x47. In addition, this ACK message piggybacks a successful response to the GET request itself. This is indicated by the 2.05 response code followed by the requested data.

Radio Frequency Identification

RFID is a form of wireless communication that incorporates the use of electromagnetic or electrostatic coupling in the radio frequency portion of the electromagnetic spectrum to uniquely identify an object, animal or person. It uses radio frequency to search,identify, track and communicate with items and people. it is a method that is used to track or identify an object by radio transmission uses over the web. Data digitally encoded in an RFID tag which might be read by the reader. This device work as a tag or label during which data read from tags that are stored in the database through the reader as compared to traditional barcodes and QR codes.

Working of RFID System :

Every RFID system consists of three components: a scanning antenna, a transceiver and a transponder. When the scanning antenna and transceiver are combined, they are referred to as an RFID reader or interrogator. There are two types of RFID readers — fixed readers and mobile readers. The RFID reader is a network-connected device that can be portable or permanently attached. It uses radio waves to transmit signals that activate the tag. Once activated, the tag sends a wave back to the antenna, where it is translated into data.

The transponder is in the RFID tag itself. The read range for RFID tags varies based on factors including the type of tag, type of reader, RFID frequency and interference in the surrounding environment or from other RFID tags and readers. Tags that have a stronger power source also have a longer read range.

Differences between RFID and IoT

- IoT can support any network, while RFID requires a specific radio technology.
- IoT can work over short-, medium- and long-range networks, while RFID only works over a few inches or feet.
- IoT can support any kind of data communications, while RFID is only suited for brief tags or authentication tokens.
- IoT is better for capturing real-time sensor data, while RFID is better suited for recording objects' proximity.
- IoT devices tend to be more complex and expensive, while RFID tags are generally optimized for low cost and simplicity.
- IoT can exchange data across wireless and wired networks, while RFID reads data in one direction from nearby tags.
- In addition, less common types of RFID with more expensive active tags can support longer ranges and two-way communication in some use cases.

How RFID and IoT can work together

RFID and IoT are complementary technologies that users frequently combine.RFID provides dependable connectivity between physical products and RFID readers that are connected to the internet. Most of these implementations take advantage of IoT capabilities for sharing data between physical devices and cloud databases to support various authentication, transaction, analytics and control use cases. Users can sometimes combine these with other technologies such as barcode tags and readers, which can be more cost-effective in some scenarios.

WIRELESS SENSOR NETWORK IN IOT

Wireless Sensor Network in IoT is an infrastructure-less wireless network that is used for deploying a large number of wireless sensors that monitor the system, physical and environmental conditions. Our extremely motivated and professional engineers are very well equipped to provide you with an all round solution if you are looking to incorporate WSN in your business.

NETWORKS CONNECTING WIRELESS SENSORS

To connect Sensors embedded in IoT devices, a communication protocol is used. A low-power wide-area network ,LPWAN, is a type of wireless network designed to allow long-range communications between these IoT devices.Lora based Wireless Sensor network is widely used. Sub-1 GHz, Zigbee,Thread etc are also used to connect sensor networks and gateway and data collected from this sensor network can be sent to cloud using cellular networks such as NBIoT, LTE-M or wifi etc.

WIRELESS SENSOR NETWORK APPLICATIONS

Patient monitoring in hospitals ,

Home security,

Military applications,

Livestock monitoring ,

Server Room monitoring

Wireless sensor network for smart agriculture

Wireless sensor network for forest fire detection

Wireless sensor network for water quality monitoring

Wireless sensor network for office monitoring

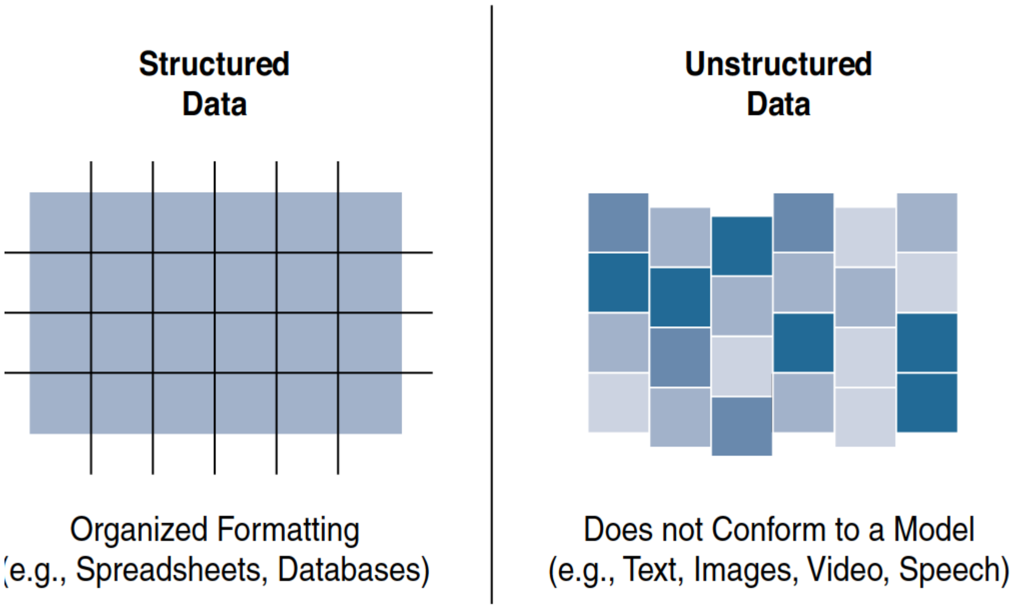
Wireless sensor network for environmental monitoring

Wireless sensor network for landslide detection

Wireless sensor network for IoT security

BigData Analytics

Structured data and unstructured data are important classifications as they typically require different toolsets from a data analytics perspective. Figure provides a high-level comparison of structured data and unstructured data.


Structured
Data
Organized Formatting
(e.g., Spreadsheets, Databases)
Unstructured
Data
Does not Conform to a Model
(e.g., Text, Images, Video, Speech)

Structured data means that the data follows a model or schema that defines how the data is represented or organized, meaning it fits well with a traditional relational database management system (RDBMS).

Structured data can be found in most computing systems and includes everything from banking transaction and invoices to computer log files and router configurations. IoT sensor data often uses structured values, such as temperature, pressure, humidity, and so on, which are all sent in a known format. Structured data is easily formatted, stored, queried, and processed; for these reasons, it has been the core type of data used for making business decisions.

Unstructured data lacks a logical schema for understanding and decoding the data through traditional programming means. Examples of this data type include text, speech, images, and video. As a general rule, any data that does not fit neatly into a predefined data model is classified as unstructured data.

According to some estimates, around 80% of a business's data is unstructured.2 Because of this fact, data analytics methods that can be applied to unstructured data, such as cognitive computing and machine learning, are deservedly garnering a lot of attention.

With machine learning applications, such as natural language processing (NLP), you can decode speech. With image/facial recognition applications, you can extract critical information from still images and video.

IoT Data Analytics Overview

The true importance of IoT data from smart objects is realized only when the analysis of the data leads to actionable business intelligence and insights. Data analysis is typically broken down by the types of results that are produced.

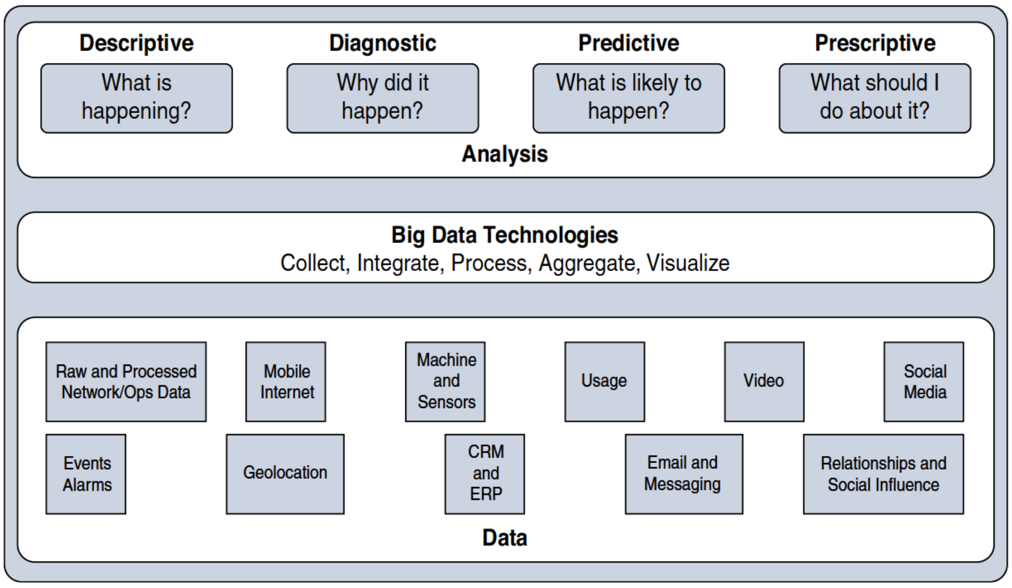


Fig: Types of Data Analysis Results

- **Descriptive:** Descriptive data analysis tells you what is happening, either now or in the past. For example, a thermometer in a truck engine reports temperature values every second. From a descriptive analysis perspective, you can pull this data at any moment to gain insight into the current operating condition of the truck engine.
  If the temperature value is too high, then there may be a cooling problem or the engine may be experiencing too much load.

- **Diagnostic:** When you are interested in the "why," diagnostic data analysis can provide the answer. Continuing with the example of the temperature sensor in the truck engine, you might wonder why the truck engine failed. Diagnostic analysis might show that the temperature of the engine was too high, and the engine overheated. Applying diagnostic analysis across the data generated by a wide range of smart objects can provide a clear picture of why a problem or an event occurred.

- **Predictive:** Predictive analysis aims to foretell problems or issues before they occur. For example, with historical values of temperatures for the truck engine, predictive analysis could provide an estimate on the remaining life of certain components in the engine. These components could then be proactively replaced before failure occurs.
  Or perhaps if temperature values of the truck engine start to rise slowly over time, this could indicate the need for an oil change or some other sort of engine cooling maintenance.

- **Prescriptive:** Prescriptive analysis goes a step beyond predictive and recommends solutions for upcoming problems. A prescriptive analysis of the temperature data from

a truck engine might calculate various alternatives to cost-effectively maintain our truck. These calculations could range from the cost necessary for more frequent oil changes and cooling maintenance to installing new cooling equipment on the engine or upgrading to a lease on a model with a more powerful engine.
Prescriptive analysis looks at a variety of factors and makes the appropriate recommendation.

IoT Data Analytics Challenges

- **Scaling problems**: Due to the large number of smart objects in most IoT networks that continually send data, relational databases can grow incredibly large very quickly. This can result in performance issues that can be costly to resolve, often requiring more hardware and architecture changes.

- **Volatility of data**: With relational databases, it is critical that the schema be designed correctly from the beginning. Changing it later can slow or stop the data- base from operating. Due to the lack of flexibility, revisions to the schema must be kept at a minimum. IoT data, however, is volatile in the sense that the data model is likely to change and evolve over time. A dynamic schema is often required so that data model changes can be made daily or even hourly.

Cloud Computing

Cloud Internet of Things (IoT) uses cloud computing services to collect and process data from IoT devices, and to manage the devices remotely. The scalability of cloud IoT platforms enables the processing of large amounts of data, as well as artificial intelligence (AI) and analytics capabilities.

Cloud IoT is a technology architecture that connects IoT devices to servers housed in cloud data centers. This enables real-time data analytics, allowing better, information-driven decision making, optimization, and risk mitigation. Cloud IoT also simplifies management of connected devices at-scale.

Cloud IoT is different from traditional, or non-cloud-based IoT in a few key ways:

- **Data Storage:** the cloud collects IoT data generated by thousands or millions of IoT sensors, with the data being stored and processed in a central location. While in other types of IoT architectures, data may be stored and processed on-premises
- **Scalability:** cloud IoT is highly scalable, as cloud infrastructure (compute, storage, and networking resources) can easily handle thousands of devices and process their data across large systems
- **Flexibility:** cloud IoT provides a high level of flexibility, as it allows devices to be added or removed as-needed, without having to reconfigure the entire system

- **Maintenance:** in cloud IoT, the maintenance of servers and networking equipment is handled by the cloud service provider (CSP). While in other types of IoT architectures, maintenance may be the responsibility of the end user
- **Cost:** cloud IoT can be more cost-effective over the long-term, as users only pay for the resources they actually consume, and users do not have to invest upfront in their own expensive compute, storage, and networking infrastructure

Cloud IoT connects IoT devices – ***which collect and transmit data*** – to cloud-based servers via communication protocols such as MQTT and HTTP and over wired and wireless networks. These IoT devices can be managed and controlled remotely and integrated with other cloud services.

IoT data is sourced from anywhere and everywhere, including sensors, actuators, operating systems, mobile devices, standalone applications, and analytic systems. By involving the cloud, vast amounts of IoT data can be stored and processed in a central location.

A cloud IoT system typically includes the following elements:

- **IoT Devices:** physical devices, such as sensors and actuators, that generate and transmit data to the cloud
- **Connectivity:** communication protocols and standards used to connect the IoT devices to the cloud. Examples of protocols include MQTT and HTTP, while examples of standards are Wi-Fi, 4G/LTE, 5G, Zigbee, and LoRa (long range)
- **Cloud Platforms:** cloud service providers (CSPs) that offer infrastructure and services to connect to the IoT devices. Examples include AWS IoT and Azure IoT
- **Data Storage:** cloud-based storage for data generated by the IoT devices, which can be housed in repositories such as a database, data warehouse, or data lake
- **Application Layer or API:** cloud IoT platforms typically provide a native application – *for analytics, machine learning (ML), and visualization* – or application programming interface (API) – *for data processing*. Usually, applications offer the ability to manage and monitor the IoT devices for provisioning, software updates, and troubleshooting
- **Security:** measures put in place to secure the data and IoT devices, such as encryption, authentication, and access control

Example – Cloud and IoT System

To illustrate all of the above elements in action, consider the example of a wind farm. A typical wind turbine can have about 108 sensors, and the average wind farm houses roughly 150 turbines, for a total of over 16,000 sensors. The data from these sensors might be sent to the cloud for storage, via 5G cellular broadband.

Once the data is stored on cloud servers, it can be used to monitor wind turbine performance, track turbine health, and adjust operating parameters as needed. Cloud IoT platforms also help with predictive maintenance, which is useful given that wind turbines on such a wind farm would be spread across an area of over 15 square miles (39 square kilometers), and downtime could result in millions of dollars of losses per year.

What are the Cloud Services for IoT?

Cloud platforms deliver a collection of capabilities that allow Internet of Things (IoT) devices to interact with cloud services, other applications, and even other IoT devices. These cloud platforms let users centrally onboard, manage, monitor, and control IoT devices.

In addition, the cloud supports services such as scalable storage, device connectivity, analytics and reporting, and identity and access management (IAM) in IoT.

Scalable Storage

Cloud IoT platforms provide scalable object storage services, such as Amazon Simple Storage Service (Amazon S3), that allow organizations to easily increase or decrease their data storage requirements. This type of flexibility is beneficial for IoT applications, as they often generate large volumes of unstructured data and must be able to store this information without sacrificing device performance.

Device Connectivity

Cloud-based IoT platforms offer straightforward, reliable, and secure connectivity at-scale between physical IoT devices and cloud services. In turn, an organization can connect thousands or millions of IoT devices to the cloud, without the need to provision or manage the requisite servers and networking equipment.

Analytics and Reporting

Cloud-based IoT platforms are equipped with powerful analytics capabilities – *in combination with computing resources* – that enable organizations to gain real-time insights into the large datasets that IoT devices produce. Through sophisticated algorithms, such as predictive modeling, statistical analysis, and machine learning (ML), IoT device data can be used to improve efficiency and make better, information-driven decisions.

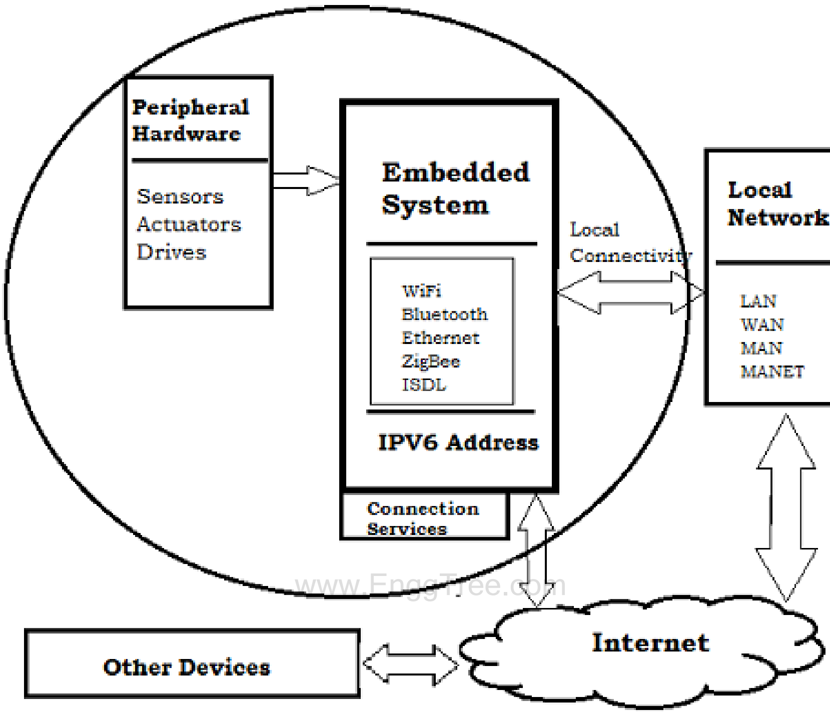
Identity and Access Management (IAM)

Security for the data generated by IoT devices can be protected in the cloud using Identity and Access Management (IAM), which is an authentication and authorization service. IAM enables organizations to grant or deny access to services and resources in the cloud for large numbers of users with different access needs.

Embedded System in (IoT)

It is essential to know about the embedded devices while learning the IoT or building the projects on IoT. The embedded devices are the objects that build the unique computing system. These systems may or may not connect to the Internet.

An embedded device system generally runs as a single application. However, these devices can connect through the internet connection, and able communicate through other network devices.



Internet of Things (IoT)

Embedded System Hardware

The embedded system can be of type microcontroller or type microprocessor. Both of these types contain an integrated circuit (IC).

The essential component of the embedded system is a RISC family microcontroller like Motorola 68HC11, PIC 16F84, Atmel 8051 and many more. The most important factor that differentiates these microcontrollers with the microprocessor like 8085 is their internal read and writable memory. The essential embedded device components and system architecture are specified below.

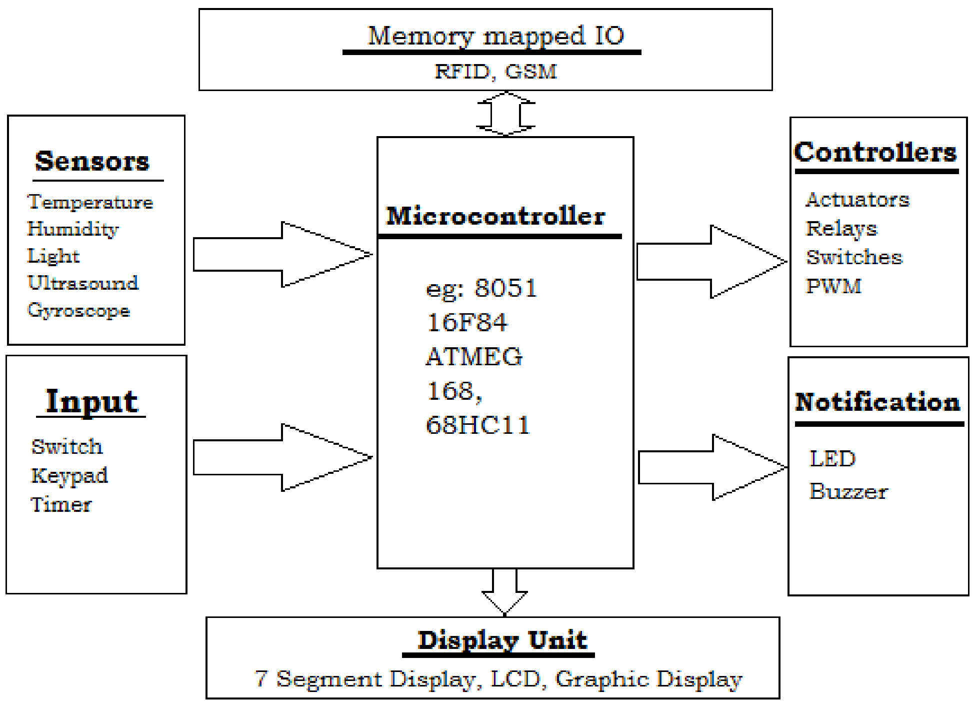


Fig:Basic Embedded System

Embedded System Software

The embedded system that uses the devices for the operating system is based on the language platform, mainly where the real-time operation would be performed. Manufacturers build embedded software in electronics, e.g., cars, telephones, modems, appliances, etc. The embedded system software can be as simple as lighting controls running using an 8-bit microcontroller. It can also be complicated software for missiles, process control systems, airplanes etc.

The Role of Embedded Systems in the IoT:

Embedded systems are at the heart of the Internet of Things. They provide the intelligence that enables devices to communicate with each other and with the cloud. The role of embedded systems in the IoT can be summarized as follows:

Sensor Integration:

Embedded systems are responsible for integrating sensors into devices. Sensors are used to detect and measure physical properties such as temperature, pressure, and humidity. These sensors generate data that is processed by the embedded system and transmitted to other devices or the cloud.

Communication:

Embedded systems are responsible for communication between devices. This communication can be wireless or wired, and can use a variety of protocols such as Wi-Fi, Bluetooth, and Zigbee. Embedded systems also handle the routing of data between devices.

Data Processing:

Embedded systems are responsible for processing the data generated by sensors. This processing can include filtering, normalization, and aggregation. The processed data is then transmitted to other devices or the cloud.

Security:

Embedded systems are responsible for the security of devices in the IoT. This includes securing data transmission, securing access to devices, and protecting against cyber attacks.

Power Management:

Embedded systems are responsible for managing the power consumption of devices in the IoT. This includes managing the power supply, optimizing power usage, and managing battery life.

Applications of Embedded Systems in IoT

Embedded systems in IoT are responsible for collecting, processing, and transmitting data between various devices and systems, and they play a crucial role in the overall functionality of IoT systems. Here are some of the applications of embedded systems in IoT:

- ➢ Smart Homes: Embedded systems in IoT are used in smart home applications to automate various functions such as lighting, temperature control, security, and entertainment. These systems are designed to be energy-efficient and cost-effective, and they can be controlled remotely using a smartphone or other internet-enabled devices.
- ➢ Industrial Automation: In industrial settings, embedded systems in IoT are used to monitor and control various machines and equipment. These systems enable real-time monitoring of production processes, ensuring that they run smoothly and efficiently. They can also detect and report any anomalies, reducing downtime and improving productivity.
- ➢ Healthcare: Embedded systems in IoT are used in healthcare applications to monitor vital signs, track medication schedules, and manage chronic conditions. These systems can transmit data to healthcare providers in real-time, allowing for timely intervention in case of emergencies.
- ➢ Agriculture: Embedded systems in IoT are used in precision agriculture to monitor soil moisture, temperature, and other environmental factors that affect crop growth. These systems enable farmers to optimize irrigation and fertilization, resulting in higher yields and reduced water usage.
- ➢ Transportation: Embedded systems in IoT are used in transportation applications to monitor vehicle performance, track routes, and manage logistics. These systems can also be used to monitor traffic conditions and optimize routes, reducing travel time and fuel consumption.

Examples of Embedded Systems in the IoT:

There are many examples of embedded systems in the IoT. Some examples include:

- Smart Home Devices:

  Embedded systems are used in smart home devices such as thermostats, lighting systems, and security systems. These devices are capable of communicating with each other and with the cloud, and can be controlled by a smartphone or other device.

- Medical Devices:

  Embedded systems are used in medical devices such as pacemakers, insulin pumps, and blood glucose monitors. These devices are capable of monitoring the patient's condition and transmitting data to healthcare providers.

- Industrial Automation:

  Embedded systems are used in industrial automation systems such as assembly lines, robotics, and process control systems. These systems are capable of monitoring and controlling industrial processes, improving efficiency and productivity.

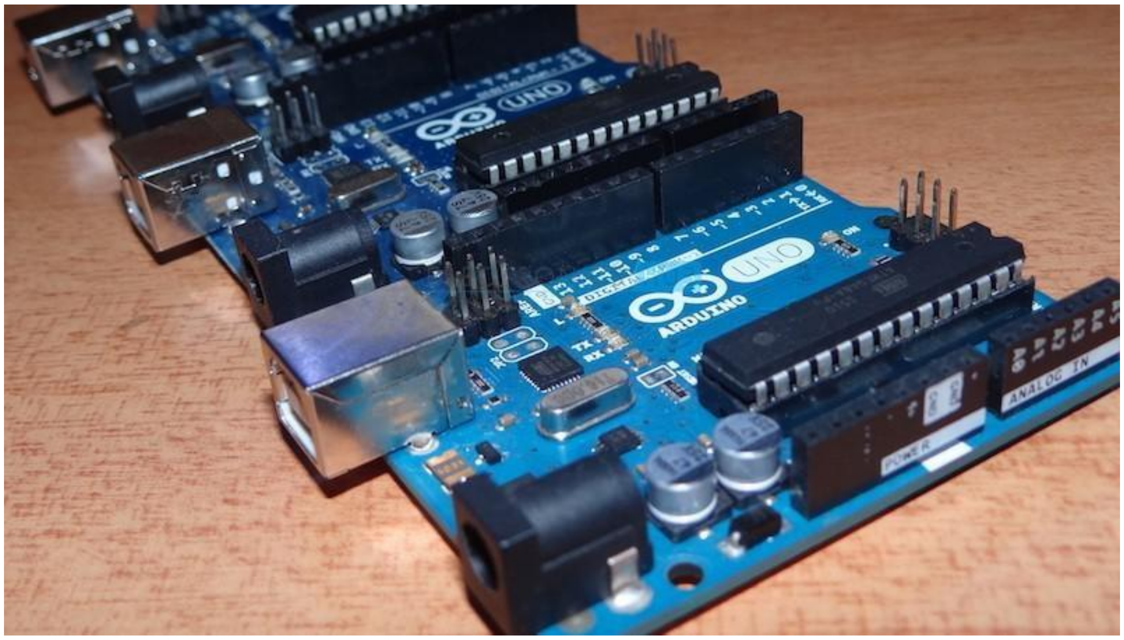
4.1.1 IOT deployment for Raspberry Pi /Arduino platform

A decade ago, working around electronics involved knowledge in physics and math, expensive lab equipment, a laboratory type setup and important of all, love for electronics. But the picture has changed over the decade or so where the above-mentioned factors became irrelevant to work around electronics except for the last part: love for electronics. One such product which made use of the above specified and many other reasons and made electronics be able reach anyone regardless of their background is "Arduino".

Introduction

Arduino is an open-source prototyping platform in electronics based on easy-to-use hardware and software. Subtly speaking, Arduino is a microcontroller based prototyping board which can be used in developing digital devices that can read inputs like finger on a button, touch on a screen, light on a sensor etc. and turning it in to output like switching on an LED, rotating a motor, playing songs through a speaker etc.

The Arduino board can be programmed to do anything by simply programming the microcontroller on board using a set of instructions for which, the Arduino board consists of a USB plug to communicate with your computer and a bunch of connection sockets that can be wired to external devices like motors, LEDs etc. The aim of Arduino is to introduce the world of electronics to people who have small to no experience in electronics like hobbyists, designers, artists etc.



Arduino is based on open source electronics project i.e. all the design specifications, schematics, software are available openly to all the users. Hence, Arduino boards can bought from vendors as they are commercially available or else you can make your own board by if you wish i.e. you can download the schematic from Arduino's official website, buy all the components as per the design specification, assemble all the components, and make your own board.

Hardware and Software

Arduino boards are generally based on microcontrollers from Atmel Corporation like 8, 16 or 32 bit AVR architecture based microcontroller.

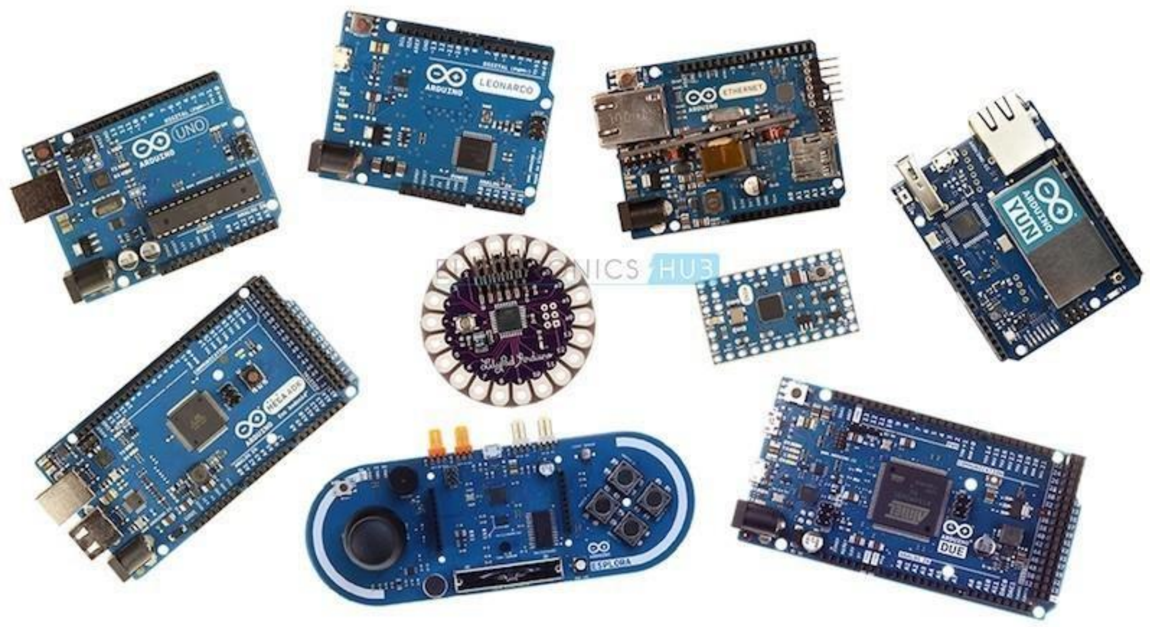
The important feature of the Arduino boards is the standard connectors. Using these connectors, we can connect the Arduino board to other devices like LEDs or add-on modules called Shields. The Arduino boards also consists of on board voltage regulator and crystal oscillator. They also consist of USB to serial adapter using which the Arduino board can be programmed using USB connection.

In order to program the Arduino board, we need to use IDE provided by Arduino. The Arduino IDE is based on Processing programming language and supports C and C++.

Types of Arduino Boards

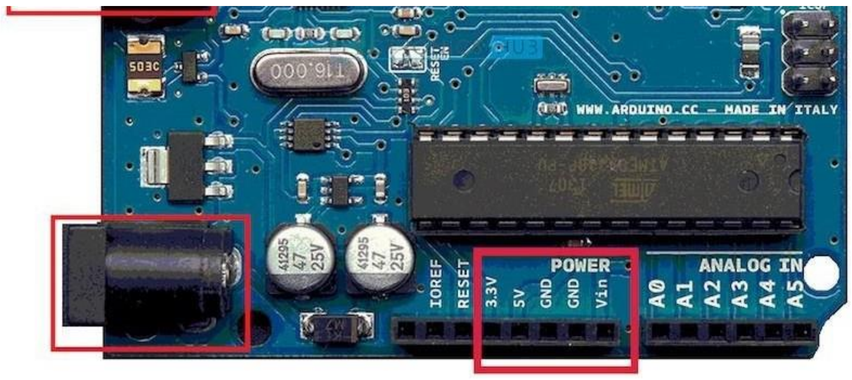
There are many types of Arduino boards available in the market but all the boards have one thing in common: they can be programmed using the Arduino IDE. The reasons for different types of boards are different power supply requirements, connectivity options, their applications etc.

Arduino boards are available in different sizes, form factors, different no. of I/O pins etc. Some of the commonly known and frequently used Arduino boards are Arduino UNO, Arduino Mega, Arduino Nano, Arduino Micro and Arduino Lilypad.



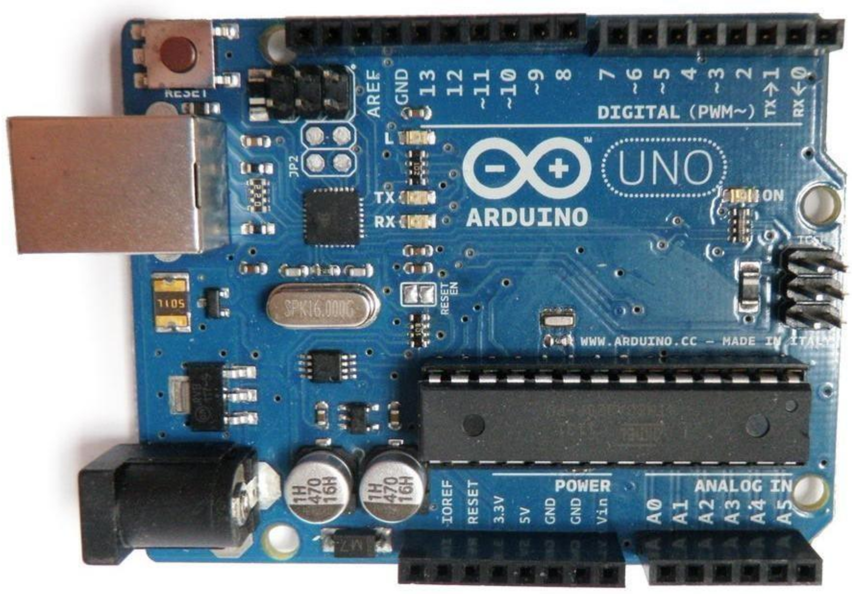
Arduino UNO

The most common version of Arduino is the Arduino Uno. This board is what most people are talking about when they refer to an Arduino. In the next step, there is a more complete rundown of its features.



Arduino Uno Features

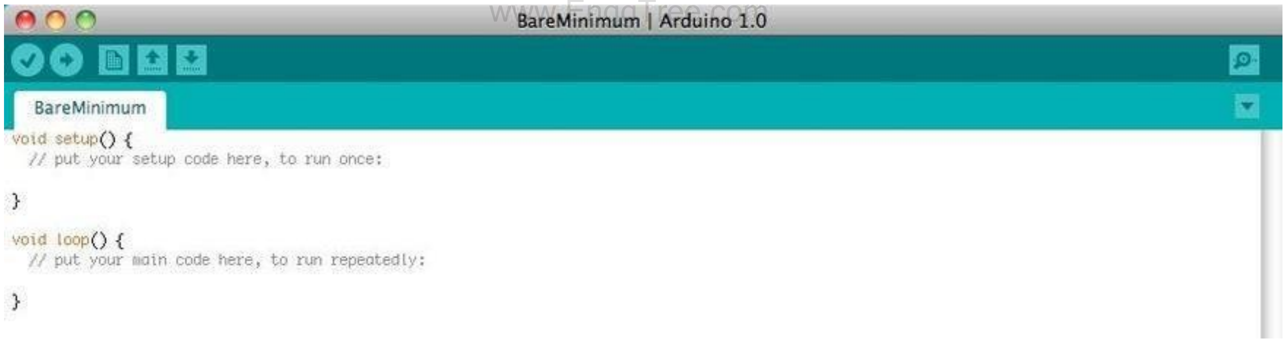
Some people think of the entire Arduino board as a microcontroller, but this is inaccurate. The Arduino board actually is a specially designed circuit board for programming and prototyping with Atmel microcontrollers.
The nice thing about the Arduino board is that it is relatively cheap, plugs straight into a computer's USB port, and it is dead-simple to setup and use (compared to other development boards).
Some of the key features of the Arduino Uno include:
An open source design. The advantage of it being open source is that it has a large community of people using and troubleshooting it This makes it easy to find someone to help you debug your projects.



An easy USB interface . The chip on the board plugs straight into your USB port and registers on your computer as a virtual serial port. This allows you to interface with it as through it were a serial device. The benefit of this setup is that serial communication is an extremely easy (and time-tested) protocol, and USB makes connecting it to modern computers really convenient.

- Very convenient power management and built-in voltage regulation. You can connect an external power source of up to 12v and it will regulate it to both 5v and 3.3v. It also can be powered directly off of a USB port without any external power.
- A 16mhz clock. This makes it not the speediest microcontroller around, but fast enough for most applications.
- 32 KB of flash memory for storing your code.
- 13 digital pins and 6 analog pins. These pins allow you to connect external hardware to your Arduino. These pins are key for extending the computing capability of the Arduino into the real world. Simply plug your devices and sensors into the sockets that correspond to each of these pins and you are good to go.
- An ICSP connector for bypassing the USB port and interfacing the Arduino directly as a serial device. This port is necessary to re-bootload your chip if it corrupts and can no longer talk to your computer.
- An on-board LED attached to digital pin 13 for fast an easy debugging of code.

Step 3: Arduino IDE



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

}

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

}
```

Before you can start doing anything with the Arduino, you need to download and install the Arduino IDE (integrated development environment). From this point on we will be referring to the Arduino IDE as the Arduino Programmer.

The Arduino Programmer is based on the Processing IDE and uses a variation of the C and C++ programming languages.
You can find the most recent version of the Arduino Programmer on this page.

Step 4: Plug It In

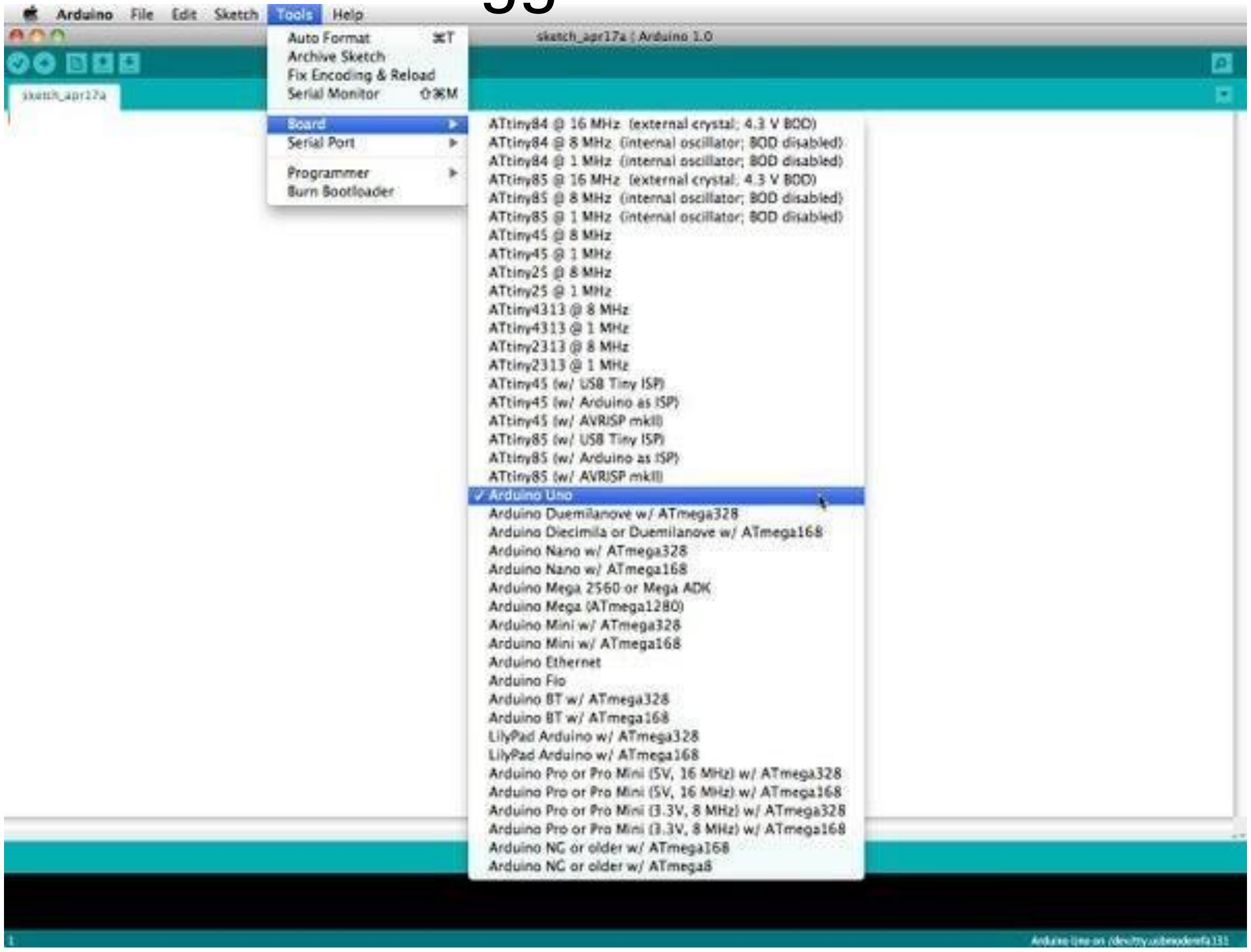
Connect the Arduino to your computer's USB port.

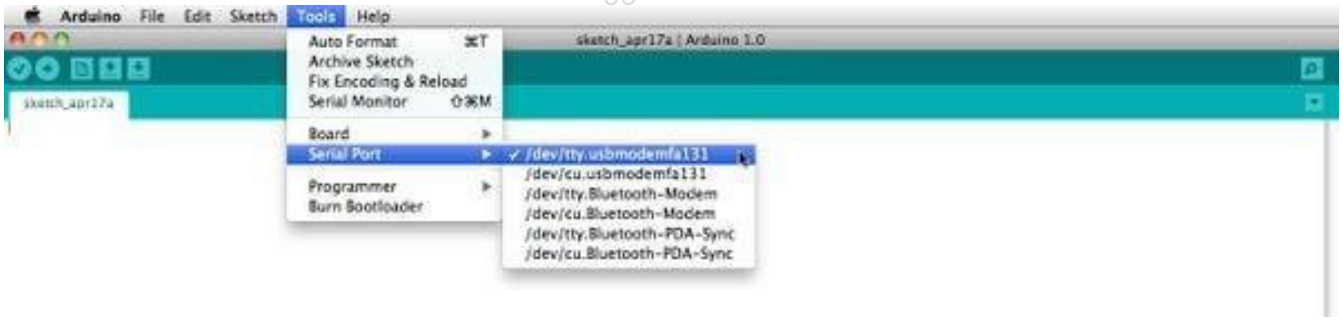
Please note that although the Arduino plugs into your computer, it is not a true USB device. The board has a special chip that allows it to show up on your computer as a virtual serial port whenit is plugged into a USB port. This is why it is important to plug the board in. When the board is
not plugged in, the virtual serial port that the Arduino operates upon will not be present

It is also good to know that every single Arduino has a unique virtual serial port address. This means that every time you plug in a different Arduino board into your computer, you will need toreconfigure the serial port that is in use.

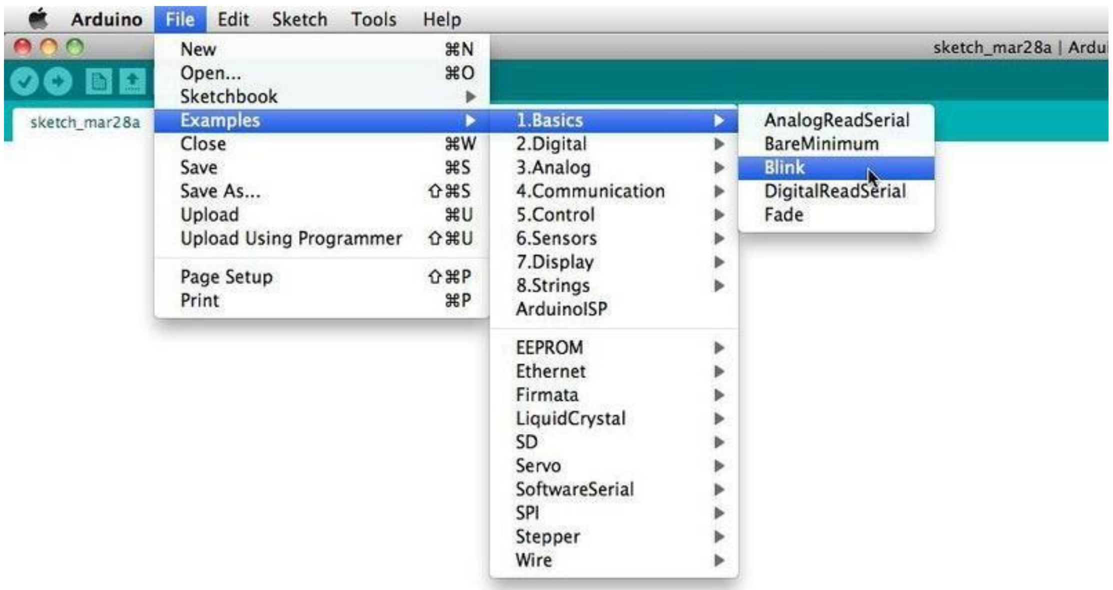
Step 5: Settings

Before you can start doing anything in the Arduino programmer, you must set the board-type
andserial port.
To set the board, go to the following:
Tools --> Boards
Select the version of board that you are using. Since I have an Arduino Uno plugged in, Iobviously selected "Arduino Uno."
To set the serial port, go to the following:Tools --> Serial Port
Select the serial port that looks like:
/dev/tty.usbmodem [random numbers]


Arduino
File
Edit
Sketch
Tools
Help
sketch_apr17a | Arduino 1.0
sketch_apr17a
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Board
Serial Port
Programmer
Burn Bootloader
ATtiny84 @ 16 MHz (external crystal; 4.3 V BOD)
ATtiny84 @ 8 MHz (internal oscillator; BOD disabled)
ATtiny84 @ 1 MHz (internal oscillator; BOD disabled)
ATtiny85 @ 16 MHz (external crystal; 4.3 V BOD)
ATtiny85 @ 8 MHz (internal oscillator; BOD disabled)
ATtiny85 @ 1 MHz (internal oscillator; BOD disabled)
ATtiny45 @ 8 MHz
ATtiny45 @ 1 MHz
ATtiny25 @ 8 MHz
ATtiny25 @ 1 MHz
ATtiny4313 @ 8 MHz
ATtiny4313 @ 1 MHz
ATtiny2313 @ 8 MHz
ATtiny2313 @ 1 MHz
ATtiny45 (w/ USB Tiny ISP)
ATtiny45 (w/ Arduino as ISP)
ATtiny45 (w/ AVRISP mkII)
ATtiny85 (w/ USB Tiny ISP)
ATtiny85 (w/ Arduino as ISP)
ATtiny85 (w/ AVRISP mkII)
✓ Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Uno on /dev/tty.usbmodemfa131


Arduino
File
Edit
Sketch
Tools
Help
sketch_apr17a | Arduino 1.0
sketch_apr17a
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Board
Serial Port
Programmer
Burn Bootloader
✓ /dev/tty.usbmodemfa131
/dev/cu.usbmodemfa131
/dev/tty.Bluetooth-Modem
/dev/cu.Bluetooth-Modem
/dev/tty.Bluetooth-PDA-Sync
/dev/cu.Bluetooth-PDA-Sync

Step 6: Run a Sketch

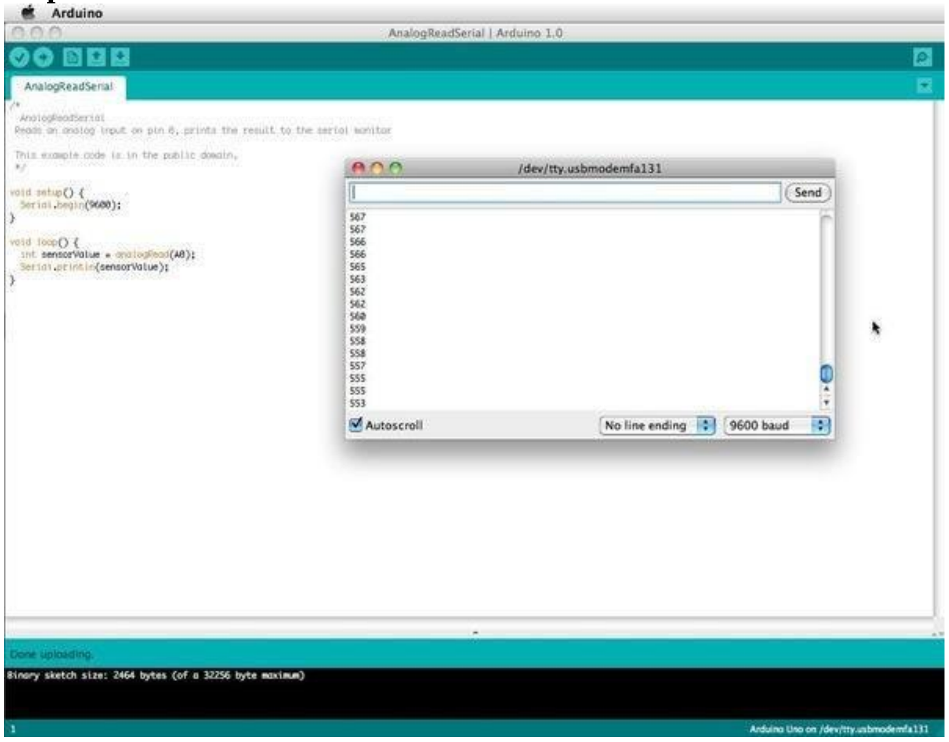


Arduino programs are called sketches. The Arduino programmer comes with a ton of example sketches preloaded. This is great because even if you have never programmed anything in your life, you can load one of these sketches and get the Arduino to do something.

To get the LED tied to digital pin 13 to blink on and off, let's load the blink example.
The blink example can be found here:
Files --> Examples --> Basics --> Blink
The blink example basically sets pin D13 as an output and then blinks the test LED on theArduino board on and off every second.

Once the blink example is open, it can be installed onto the ATMEGA328 chip by pressing theupload button, which looks like an arrow pointing to the right.
Notice that the surface mount status LED connected to pin 13 on the Arduino will start to blink. You can change the rate of the blinking by changing the length of the delay and pressing the upload button again.

Step 7: Serial Monitor



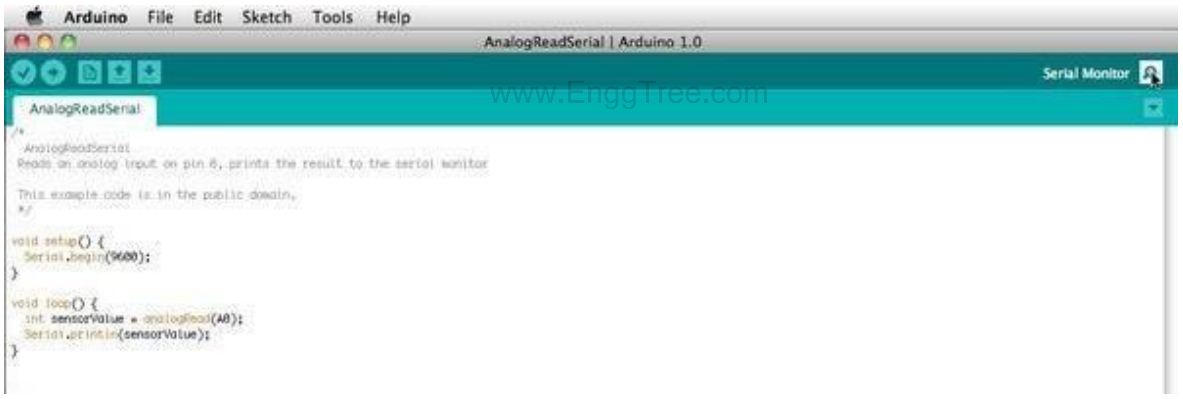
```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor

 This example code is in the public domain.
 */

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
}
```

```
567
567
566
566
565
563
562
562
560
559
558
558
557
555
555
553
```

Autoscroll ✓ | No line ending | 9600 baud

```
Done uploading.
Binary sketch size: 2464 bytes (of a 32256 byte maximum)
```

Arduino Uno on /dev/tty.usbmodemfa131

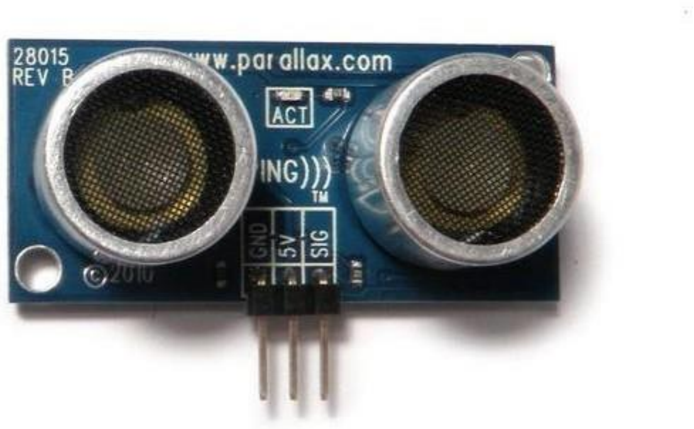


```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor

 This example code is in the public domain.
 */

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
}
```

The serial monitor allows your computer to connect serially with the Arduino. This is important because it takes data that your Arduino is receiving from sensors and other devices and displaysit in real-time on your computer. Having this ability is invaluable to debug your code and understand what number values the chip is actually receiving.

For instance, connect center sweep (middle pin) of a potentiometer to A0, and the outer pins, respectively, to 5v and ground. Next upload the sketch shown below:
File --> Examples --> 1.Basics --> AnalogReadSerial Click the button to engage the serial monitor which looks like a magnifying glass. You can now see the numbers being read by the analog pin in the serial monitor. When you turn the knob the numbers will increase and decrease.
The numbers will be between the range of 0 and 1023. The reason for this is that the analog pinis converting a voltage between 0 and 5V to a discreet number.

Step 8: Digital In



The Arduino has two different types of input pins, those being analog and digital.To begin with, lets look at the digital input pins

Digital input pins only have two possible states, which are on or off. These two on and off states are also referred to as:

- HIGH or LOW
- 1 or 0
- 5V or 0V.

This input is commonly used to sense the presence of voltage when a switch is opened or closed. Digital inputs can also be used as the basis for countless digital communication protocols. By creating a 5V (HIGH) pulse or 0V (LOW) pulse, you can create a binary signal, the basis of all computing. This is useful for talking to digital sensors like a PING ultrasonic sensor, or communicating with other devices.

Step 9: Analog In

Aside from the digital input pins, the Arduino also boasts a number of analog input pins.
Analog input pins take an analog signal and perform a 10-bit analog-to-digital (ADC) conversionto turn it into a number between 0 and 1023 (4.9mV steps).
This type of input is good for reading resistive sensors. These are basically sensors whichprovide resistance to the circuit. They are also good for reading a varying voltage signal between0 and 5V. This is useful when interfacing with various types of analog circuitry.

If you followed the example in Step 7 for engaging the serial monitor, you have already tried using an analog input pin.

Step 10: Digital Out

A digital out pin can be set to be HIGH (5v) or LOW (0v). This allows you to turn things on and off.
Aside from turning things on and off (and making LEDs blink), this form of output is convenientfor a number of applications.
Most notably, it allows you to communicate digitally. By turning the pin on and off rapidly, you are creating binary states (0 and 1), which is recognized by countless other electronic devices asa binary signal. By using this method, you can communicate using a number of differentprotocols.
Digital communication is an advanced topic, but to get a general idea of what can be done, checkout the Interfacing With Hardware page.
If you followed the example in Step 6 for getting an LED to blink, you have already tried using adigital output pin.

Step 11: Analog Out

As mentioned earlier, the Arduino has a number of built in special functions. One of thesespecial functions is pulse width modulation, which is the way an Arduino is able to create an analog-like output.

Pulse width modulation - or PWM for short - works by rapidly turning the PWM pin high (5V) and low (0V) to simulate an analog signal. For instance, if you were to blink an LED on and off rapidly enough (about five milliseconds each), it would seem to average the brightness and only appear to be receiving half the power. Alternately, if it were to blink on for 1 millisecond and then blink off for 9 millisecond, the LED would appear to be 1/10 as bright and only be receiving1/10 the voltage.
PWM is key for a number of applications including making sound, controlling the brightness of lights, and controlling the speed of motors.
To try out PWM yourself, connect an LED and 220 ohm resistor to digital pin 9, in series toground. Run the following example code:
File --> Examples --> 3.Analog --> Fading

4.1.2 Raspberry Pi

Raspberry Pi is a low-cost pocket computer that is very economical to own. It is about the size of an ATM Card and can work as a fully functional computer in certain normal use cases, like working with simple applications, playing low-end games, etc. It was first released in 2012 by the Raspberry Pi foundation with the aim to provide easy access to computing education to everyone. It can cost as less as 5$ to a maximum price of 100$ (which is rare).

Scope

we will be understanding Operating systems that can be installed on a Raspberry Pi.

- We'll learn about What an operating system in general is.
- We'll go through a Variety of Operating Systems that a Raspberry Pi can run.

Introduction

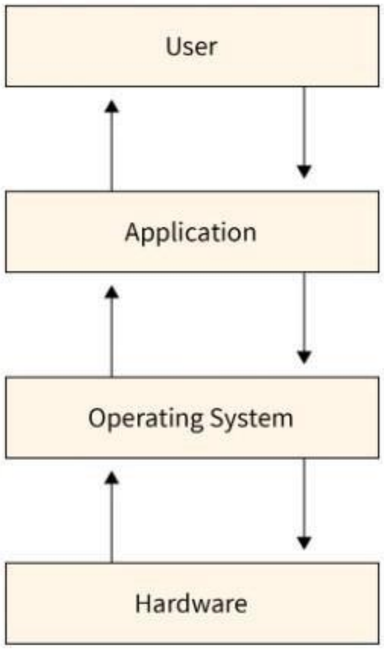
As read above, Raspberry Pi is a very low-cost computer that comes along with the advantage of portability. However, being in such a small form factor, it gets bounded by the type of hardwareto use in making it; hence, it will be significantly tough to run regular operating systems on it.

Due to this, specific operating systems were designed to power a Raspberry Pi; some of them were entirely new, while some originated from existing popular operating systems. Most of the Raspberry Pi OS is Linux based, but it also has windows 10-based Raspberry Pi OS (Windows 10 IoT core) built explicitly for low-powered devices like this.

What is an Operating System?

The technical definition is **An operating system is an interface between the hardware of a machine and the user who is using it**, but what does this really mean? It means it is basically the medium using which we communicate with our computer machine. It does not matter if we have the fastest system in the world; at the end of the day, that is just hardware, an object; andwe do not know how to work with it, so we need some medium that works as an intermediate between us and the computer, i.e., when we press ctrl on our keyboard, it should instruct the computer what to react based on that; when we want to open some application, it should provide us a way to do so, by listing all the available applications on the system.

Meaning an operating system is a software program that helps us to use and to connect with the computer hardware. For example, if we want to use our mouse or keyboard, only with the help ofan OS we can do that; if we want to install some program on our computer, we would be needingan OS; if we want to create a file, we need an OS; we want to delete a file, we would again be needing an OS, i.e., without an operating system we cannot use the computer hardware, wewould be needing some underlying software, i.e., some operating system, using which we would do so.



What is a Raspberry Pi Operating System

Now, what is Raspberry Pi operating system? Before that, let's first try to understand what Raspberry Pi is. Raspberry Pi is a small, low-cost computer, and its size is about the size of an ATM card, which is developed by the Raspberry Pi foundation. The organization's mission is to educate people in computing and to provide easier access to computer education.

The above image is a picture of a Raspberry Pi; we can see that there are various ports available in it on which different devices can be mounted and used.

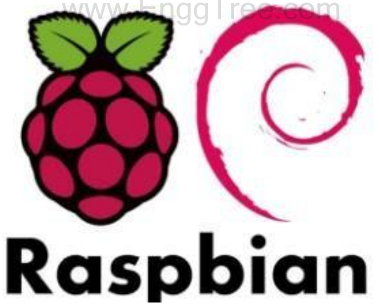
It was first launched in 2012, and from then onwards, various variations of it have been launched. The original Raspberry Pi had a single-core 700mhz CPU and a 256MB of RAM, butit has evolved a lot since then; today, we have a quad-core Raspberry Pi with a clock speed of around 1.5Ghz and up to 4GB of RAM. Surprisingly the cost of Raspberry Pi has always been less than 100 USD. In fact, the Raspberry Pi Zero (an even low-cost version of regular RaspberryPi) costs as less as 5 USD. A full-fledged general-purpose CPU under 5$, that's what theorganization's mission is "Aiming to provide people easier and low-cost access to the computers."

Raspberry Pi is used by people all around the world in learning how to program, build hardware projects, do home automation, and implement Kubernetes clusters, and it is even getting used in some industrial applications. Raspberry Pi is a very economical computer that runs LinuxOperating System.

Now, let's talk about which specific distribution of Linux Raspberry Pi uses. Raspberry Pi officially recommends the use of the Raspbian Operating System. It is a Debian-based OS, explicitly made for Raspberry Pi and hence its name **Raspbian.**

Raspbian

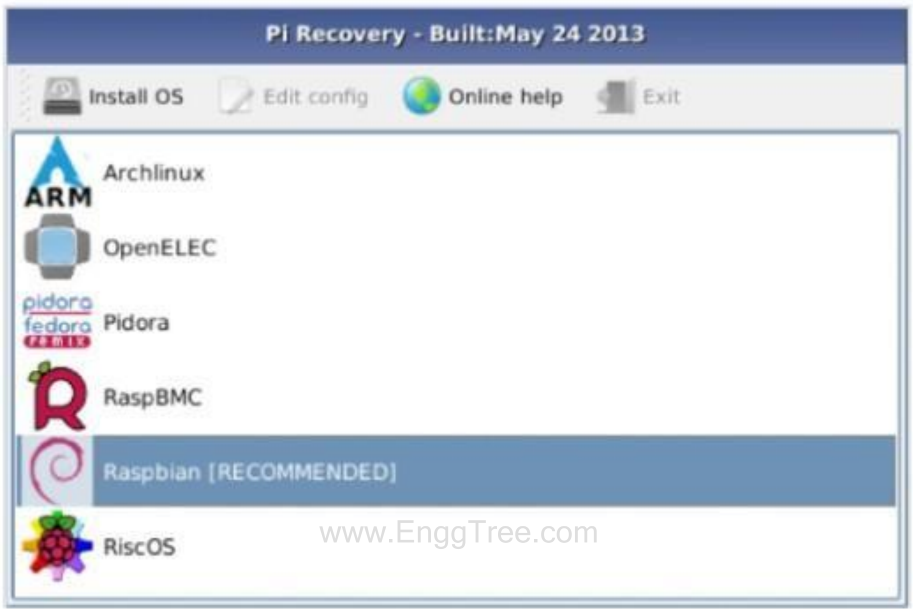
Raspbian or Raspberry Pi OS is a Linux-based operating system built specifically for Raspberry Pi. It is packed with all the necessary tools and features that are required for day-to-day use. It will possibly run on every kind of Raspberry Pi board with a few exceptions, like the Raspberry Pi's pico edition, because of its far smaller form factor and computing power.



NOOBS

New Out Of the Box Software, or simply NOOBS is an operating system installer for Raspberry Pi, delivered primarily on an SD card, which contains a variety of operating systems, out of which we can choose which one we want to install on our Raspberry Pi. It is made for people who are absolutely new to the Raspberry Pi and do not want to deal with the complex setting up process of burning an OS image on an SD card. NOOBS is provided along with every new Raspberry Pi at the time of its purchase.

With NOOBS, the user only needs to connect their Raspberry Pi to a display screen and a keyboard and then power it up; the NOOBs will boot. There we can select which operating system we want to install, and NOOBS will install the respective OS on the same SD card withina few minutes.



4.2.1 Raspberry-Pi Architecture

What is a Raspberry Pi? Raspberry pi is the name of the “credit card-sized computer board” developed by the Raspberry pi foundation, based in the U.K. It gets plugged in a TV or monitor and provides a fully functional computer capability. It is aimed at imparting knowledge about computing to even younger students at the cheapest possible price. Although it is aimed at teaching computing to kids, but can be used by everyone willing to learn programming, thebasics of computing, and building different projects by utilizing its versatility.

Raspberry Pi is developed by Raspberry Pi Foundation in the United Kingdom. The Raspberry Piis a series of powerful, small single board computer

Raspberry Pi is launched in 2012 and there have been several iterations and variations released since then.Various versions of Raspberry Pi have been out till date. All versions consist of a Broadcom system on a chip (SoC) with an integrated ARM-compatible CPU and on-chip graphics processing unit (GPU).

The original device had a single-core Processor speed of device ranges from 700 MHz to 1.2 GHz and a memory range from 256 MB to 1 GB RAM.To store the operating system and program memory Secure Digital (SD) cards are used. Raspbian OS which is a Linux operating system is recommended OS by Raspberry Pi Foundation. Some other third party operating systems like RISC OS Pi. Diet Pi, Kali, Linux can also be run on Raspberry Pi.

Used:
It also provides a set of general purpose input/output pins allowing you to control electronic components for physical computing and explore the Internet of Things (IOT).

Raspberry pi Model

There have been many generations of raspberry Pi from Pi 1 to Pi 4. There is generally a modelA and model B. Model A is a less expensive variant and it trends to have reduced RAM and dualcores such as USB and Ethernet.

List of Raspberry pi models and releases year:

1. pi 1 model B – 2012
2. pi 1 model A – 2013
3. pi 1 model B+ -2014
4. pi 1 model A+ – 2014
5. Pi 2 Model B – 2015
6. Pi 3 Model B- 2016
7. Pi 3 Model B+ -2018
8. Pi 3 Model A+ -2019
9. Pi 4 Model A – 2019

Raspberry pi Diagram:

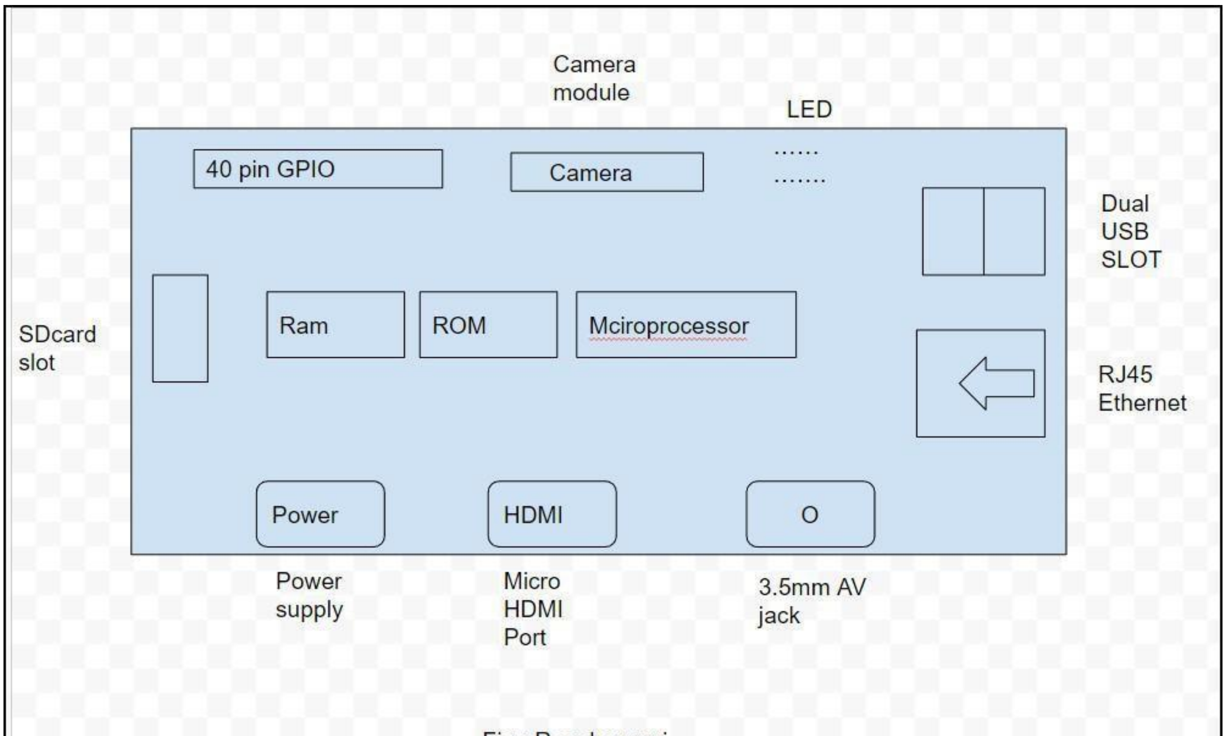


Fig : Raspberry pi

Specs of the Computer: – The computer has a quad-core ARM processor that doesn't support the same instruction as an X86 desktop CPU. It has 1GB of RAM, One HDMI port, four USB ports, one Ethernet connection, Micro SD slot for storage, one combined 3.5mm audio/videoport, and a Bluetooth connection. It has got a series of input and output pins that are used for making projects like – home security camera, encrypted door lock, etc.

Versatility of Raspberry Pi: – It is indeed a versatile computer and can be utilized by people from all age groups, it can be used for watching videos on YouTube, watching movies, and programming in languages like Python, Scratch, and many more. As mentioned above it has a series of I/O pins that give this board the ability to interact with its environment and hence can beutilized to build really cool and interactive projects.

Examples of projects: – It can be turned into a weather station by connecting some instruments to it for check the temperature, wind speed, humidity etc… It can be turned into a home surveillance system due to its small size; by adding some cameras to it the security network will be ready. If you love reading books it can also become a storage device for storing thousands of eBooks and also you can access them through the internet by using this device.

Build Physical Projects With Python on the Raspberry Pi

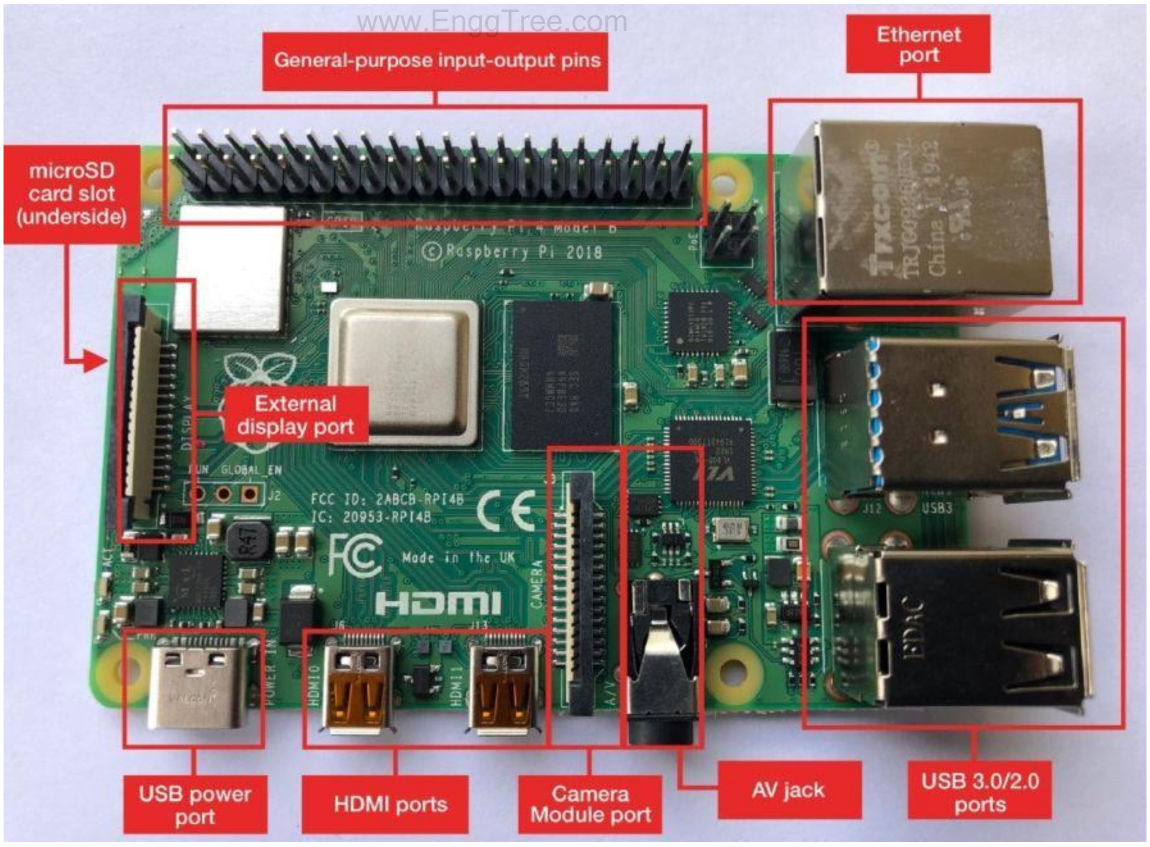
The Raspberry Pi is one of the leading physical computing boards on the market. From hobbyists building DIY projects to students learning to program for the first time, people use the RaspberryPi every day to interact with the world around them. Python comes built in on the Raspberry Pi, so you can take your skills and start building your own Raspberry Pi projects today.

Getting to Know the Raspberry Pi

The Raspberry Pi is a single-board computer developed by the Raspberry Pi Foundation, a UK- based charity organization. Originally designed to provide young people with an affordable computing option to learn how to program, it has developed a massive following in the maker and DIY communities because of its compact size, full Linux environment, and general-purpose input–output (**GPIO**) pins.

With all the features and capabilities that are packed into this small board, there's no shortage of projects and use cases for the Raspberry Pi.If you can think of a project that would benefit from having a credit card–sized computer attached to it, then someone has probably used a Raspberry Pi to do it. The Raspberry Pi is a fantastic way to bring your Python project ideas to life.
Raspberry Pi Board Overview

Below is the board layout of the Raspberry Pi 4. While this layout is slightly different from previous models of the Raspberry Pi, most of the connections are the same. The setup described in the next section should be the same for both a Raspberry Pi 3 and a Raspberry Pi 4:



The Raspberry Pi 4 board contains the following components:

- **General-purpose input–output pins:** These pins are used to connect the Raspberry Pi toelectronic components.
- **Ethernet port:** This port connects the Raspberry Pi to a wired network. The Raspberry Pi also has Wi-Fi and Bluetooth built in for wireless connections.
- **Two USB 3.0 and two USB 2.0 ports:** These USB ports are used to connect peripherals like a keyboard or mouse. The two black ports are USB 2.0 and the two blue ports are USB 3.0.
- **AV jack:** This AV jack allows you to connect speakers or headphones to the Raspberry Pi.
- **Camera Module port:** This port is used to connect the official Raspberry Pi Camera Module, which enables the Raspberry Pi to capture images.
- **HDMI ports:** These HDMI ports connect the Raspberry Pi to external monitors. The Raspberry Pi 4 features two micro HDMI ports, allowing it to drive two separate monitors at the same time.
- **USB power port:** This USB port powers the Raspberry Pi. The Raspberry Pi 4 hasa **USB Type-C** port, while older versions of the Pi have a **micro-USB** port.
- **External display port:** This port is used to connect the official seven-inch RaspberryPi touch display for touch-based input on the Raspberry Pi.
- **microSD card slot (underside of the board):** This card slot is for the microSD card thatcontains the Raspberry Pi operating system and files.

People often wonder what the difference is between a Raspberry Pi and an Arduino. The Arduinois another device that is widely used in physical computing. While there is some overlap in the capabilities of the Arduino and the Raspberry Pi, there are some distinct differences.

The Arduino platform provides a hardware and software interface for programming microcontrollers. A microcontroller is an integrated circuit that allows you to read input from and send output to electronic components. Arduino boards generally have limited memory, so they’re often used to repeatedly run a single program that interacts with electronics.

The Raspberry Pi is a general-purpose, Linux-based computer. It has a full operating system witha GUI interface that is capable of running many different programs at the same time.
The Raspberry Pi comes with a variety of software preinstalled, including a web browser, an office suite, a terminal, and even Minecraft. The Raspberry Pi also has built-in Wi-Fi and Bluetooth to connect to the Internet and external peripherals.

For running Python, the Raspberry Pi is often the better choice, as you get a full-fledged Python installation out of the box without any configuration.

4.2.2 Raspberry Pi Programming

"Hello world" is the beginning of everything when it comes to computing and programming. It's the first thing you learn in a new programming language, and it's the way you test something out or check to see if something's working because it's usually the simplest way oftesting simple functionality.

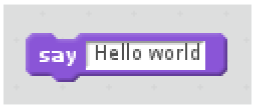
Warriors of programming language wars often cite their own language's "hello world" against that of another, saying theirs is *shorter* or *more concise* or *more explicit* or something. Having a nice simple readable "hello world" program makes for a good intro for beginners learning your language, library, framework, or tool.

I thought it would be cool to create a list of as many different "hello world" programs as possible that can be run on the Raspberry Pi using its Raspbian operating system, but without installing any additional software than what comes bundled when you download it from the Raspberry Pi website. I've created a GitHub repository of these programs, and I've explained 10 of them for you here.

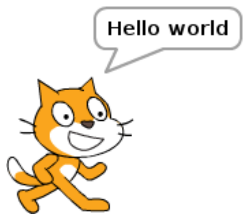
1. Scratch

Scratch is a graphical block-based programming environment designed for kids to learn programming skills without having to type or learn the synax of a programming language. The "hello world" for Scratch is simple—and very visual!

1. Open **Scratch 2** from the main menu
2. Click **Looks**.
3. Drag a **say Hello!** block into the workspace on the right.
4. Change the text to Hello world.



5. Click on the block to run the code.



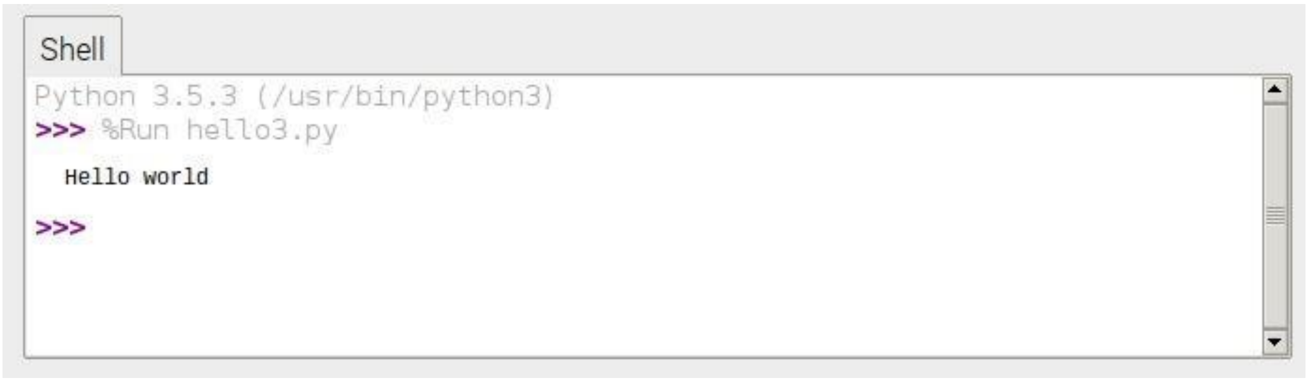
2. Python

Python is a powerful and professional language that's also great for beginners— and it's lots of fun to learn. Because one of Python's main objectives was to be readable and stick to simple English, its "hello world" program is as simple as possible.

1. Open **Thonny Python IDE** from the main menu.
2. Enter the following code:

```
print("Hello world" )
```

3. Save the file as hello3.py.
4. Click the **Run** button.



```
Python 3.5.3 (/usr/bin/python3)
>>> %Run hello3.py

  Hello world

>>>
```

opensource.co

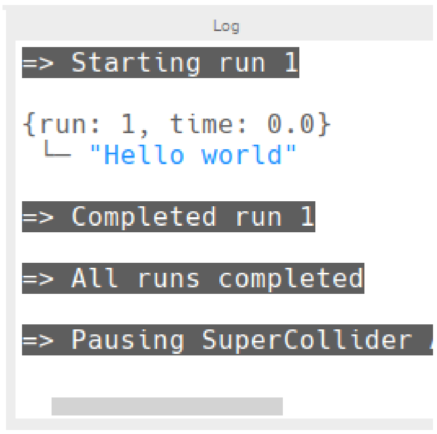
3. Ruby/Sonic Pi

Ruby is another powerful language that's friendly for beginners. Sonic Pi, the live coding musicsynth, is built on top of Ruby, so what users actually type is a form of Ruby.

1. Open **Sonic Pi** from the main menu.
2. Enter the following code:

```
puts "Hello world
```

3. Press **Run**.



```
Log
=> Starting run 1

{run: 1, time: 0.0}
 └─ "Hello world"

=> Completed run 1

=> All runs completed

=> Pausing SuperCollider
```

Unfortunately, "hello world" does not do Sonic Pi justice in the slightest

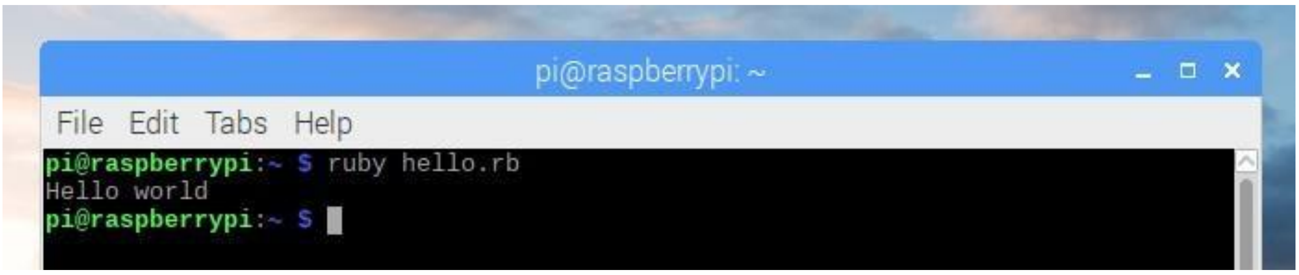
Alternatively, to using the Sonic Pi application for this example, you can write Ruby code in atext editor and run it in the terminal:

1. Open **Text Editor** from the main menu.
2. Enter the following code:

```
puts "Hello world
```

3. Save the file as hello.rb in the home directory
4. Open **Terminal** from the main menu.
5. Run the following command:

```
ruby hello.r
```



```
pi@raspberrypi:~ $ ruby hello.rb
Hello world
pi@raspberrypi:~ $
```

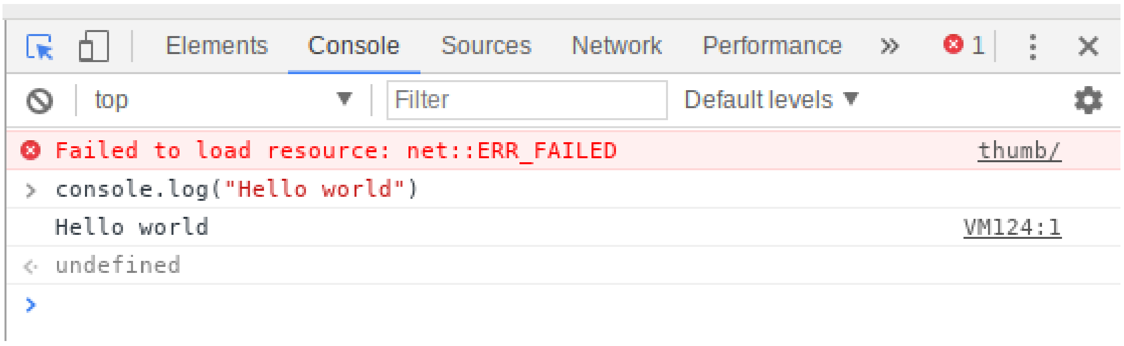
4. JavaScript

This is a bit of a cheat as I just make use of client-side JavaScript within the web browser usingthe Web Inspector console, but it still counts!

1. Open **Chromium Web Browser** from the main menu.
2. Right-click the empty web page and select **Inspect** from the context menu.
3. Click the **Console** tab.
4. Enter the following code:

```
console.log("Hello world" )
```

5. Press **Enter** to run.



```
Failed to load resource: net::ERR_FAILED                    thumb/
> console.log("Hello world")
  Hello world                                              VM124:1
< undefined
>
```

You can also install NodeJS on the Raspberry Pi, and write server-side JavaScript, but that's not available in the standard Raspbian image.

5. Bash

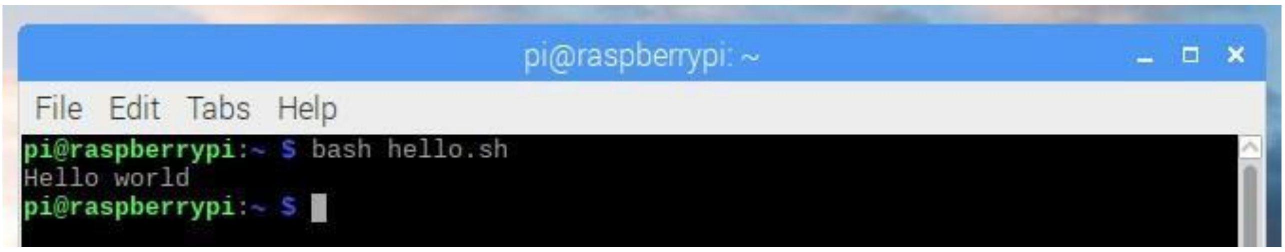
Bash (Bourne Again Shell) is the default Unix shell command language in most Linux distributions, including Raspbian. You can enter Bash commands directly into a terminal window, or script them into a file and execute the file like a programming script.

1. Open **Text Editor** from the main menu
2. Enter the following code:

```
echo "Hello world
```

3. Save the file as hello.sh in the home directory.
4. Open **Terminal** from the main menu.
5. Run the following command:

```
ash
```



```
pi@raspberrypi:~ $ bash hello.sh
Hello world
pi@raspberrypi:~ $
```

Note you'd usually see a "hashbang" at the top of the script (#!/bin/bash), but because I'm calling this script directly using the bash command, it's not necessary (and I'm trying to keep all these examples as short as possible).

You'd also usually make the file executable with chmod +x, but again, this is not necessary as I'm executing with bash.

6. Java

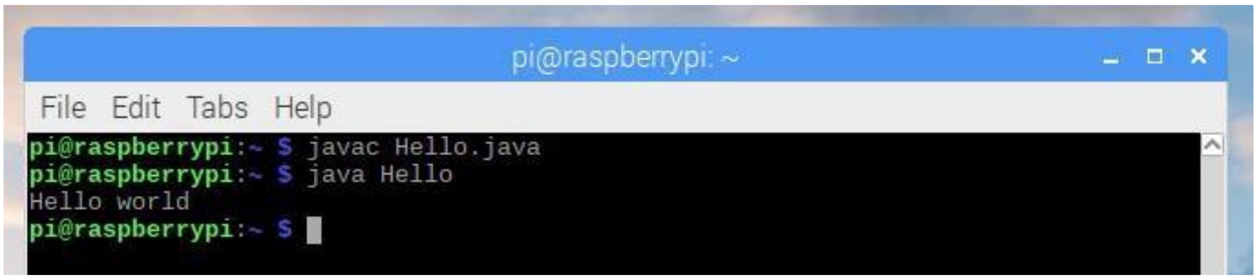
Java is a popular language in industry, and is commonly taught to undergraduates studying computer science. I learned it at university and have tried to avoid touching it since then. Apparently, now I do (very small amounts of) it for fun...

1. Open **Text Editor** from the main menu.
2. Enter the following code:

3. **public class Hello** {
4. **public static void main**(String[] args) {
5. System.out.println("Hello world");
6. }
7. }
8.

9. Save the file as Hello.java in the home directory.
10. Open **Terminal** from the main menu.
11. Run the following commands:

12.
```
javac Hello.java
java Hell o
```



```
pi@raspberrypi:~ $ javac Hello.java
pi@raspberrypi:~ $ java Hello
Hello world
pi@raspberrypi:~ $
```

I could *almost* remember the "hello world" for Java off the top of my head, but not quite.

Ialways forget where the String[] args bit goes, but it's obvious when you think about it...

7. C

C is a fundamental low-level programming language. It's what many programming languages arewritten in. It's what operating systems are written in. See for yourself&mdash:take a look at the source for Python and the Linux kernel. If that looks a bit hazy, get started with "hello world":

1. Open **Text Editor** from the main menu.
2. Enter the following code:
3. #include <stdio.h>

4. int main() {

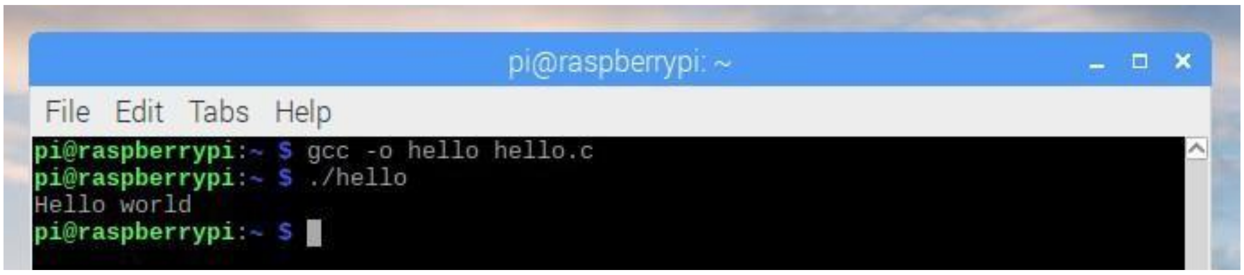
5. printf("Hello world\n");

}

6. Save the file as hello.c in the home directory.
7. Open **Terminal** from the main menu
8. Run the following commands:

```
9. gcc -o hello hello.c
```

```
./hello
```



```
pi@raspberrypi:~ $ gcc -o hello hello.c
pi@raspberrypi:~ $ ./hello
Hello world
pi@raspberrypi:~ $ 
```

Note that in the previous examples, only one command was required to run the code (e.g., python3 hello.py or ruby hello.rb) because these languages are interpreted rather than compiled. (Actually Python is compiled at runtime but that's a minor detail.) C code is compiled into byte code and the byte code is executed.

If you're interested in learning C, the Raspberry Pi Foundation publishes a book Learning to codewith C written by one of its engineers. You can buy it in print or download for free.

8. C++

C's younger bother, C++ (that's C incremented by one...) is another fundamental low-level language, with more advanced language features included, such as classes. It's popular in a

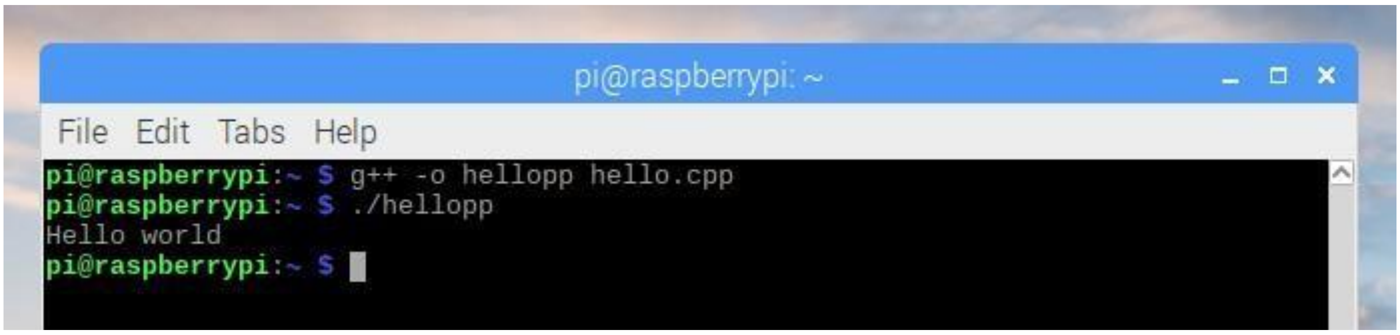
rangeof uses, including game development, and chunks of your operating system will be written in C+
+ too.

Open **Text Editor** from the main menu.Enter the following code:
#include <iostream>using namespace std;int main() {
cout << "Hello world\n";
}

Save the file as hello.cpp in the home directory.Open **Terminal** from the main menu.
Run the following commands:

```
1. g++ -o hellopp
   hello.cpp
```

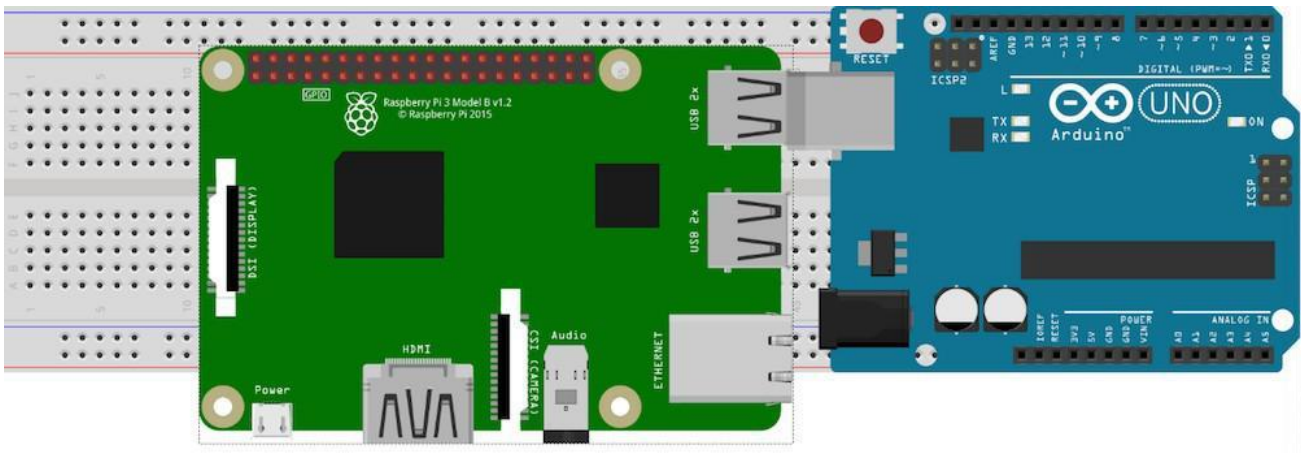
```
./hellocp
```



```
pi@raspberrypi:~ $ g++ -o hellopp hello.cpp
pi@raspberrypi:~ $ ./hellopp
Hello world
pi@raspberrypi:~ $ 
```

Readers familiar with C/C++ will notice I have not included the main function return values inmy examples. This is intentional as to remove boilerplate, which is not strictly necessary.

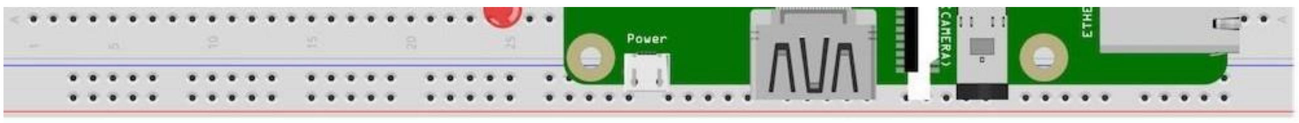
4.3.1 Interface a Raspberry Pi with an Arduino



Interface a Raspberry Pi with an Arduino so the two boards can communicate with one another. Sometimes you may need to connect an Arduino to a Raspberry Pi. For example, if you have sensors, motors, and actuators, you can connect these to the Arduino and make the Arduino send values to and from the Raspberry Pi. This way, we can separate the computing intensive tasks (done by the Raspberry Pi) and controlling tasks (done by the Arduino).

we will connect an Arduino to a Raspberry Pi and have the Arduino send "Hello from Arduino" to the Raspberry Pi, and the Raspberry Pi will blink an LED upon receiving the command from the Arduino.

For communication, we will use simple serial communication over USB cable.So, let's get started! Connect the LED to pin number 11 as shown in the picture below.



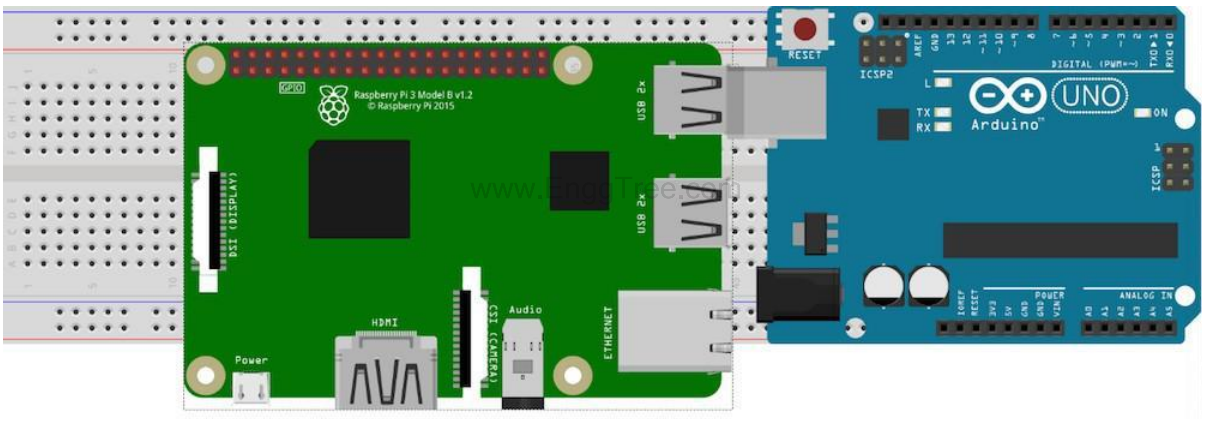
Turn on the Raspberry Pi and open Python 3 in a new window.

Write the following code in the new window and save it. (Save to your desktop so you don't loseit.)

```
import serial
import RPi.GPIO as GPIOimport time

ser=serial.Serial("/dev/ttyACM0",9600)  #change ACM number as found from ls /dev/tty/ACM*
ser.baudrate=9600
def blink(pin):
GPIO.output(pin,GPIO.HIGH)time.sleep(1)
GPIO.output(pin,GPIO.LOW)time.sleep(1)
return
GPIO.setmode(GPIO.BOARD)GPIO.setup(11, GPIO.OUT)
while True:
read_ser=ser.readline()print(read_ser)
if(read_ser=="Hello From Arduino!"):
blink(11)
```

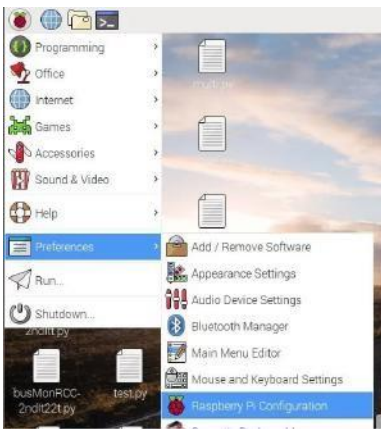
Now open Arduino IDE and upload the following code to your Arduino.String data="Hello From Arduino!";

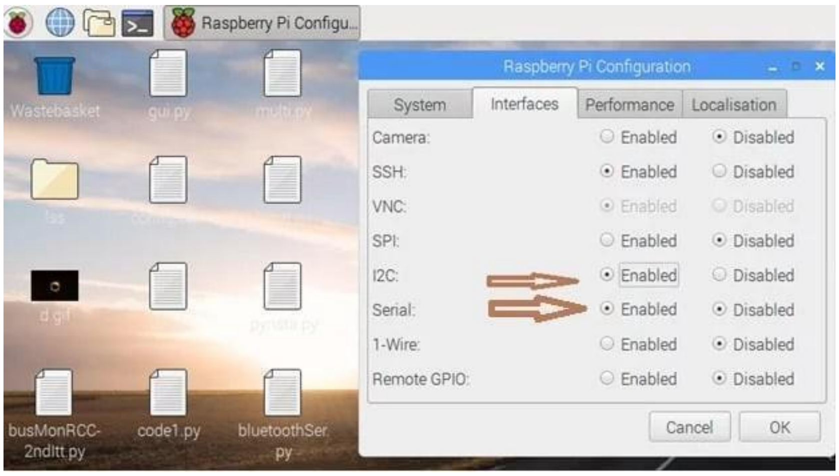
```
void setup()
{
// put your setup code here, to run once:Serial.begin(9600);
}
void loop()
{
// put your main code here, to run repeatedly:Serial.println(data);//data that is being Sent delay(200);
}
```



Make sure the code is uploaded to Arduino.

In your Raspberry Pi interface, be sure to enable Serial and I2C in PiConfig.



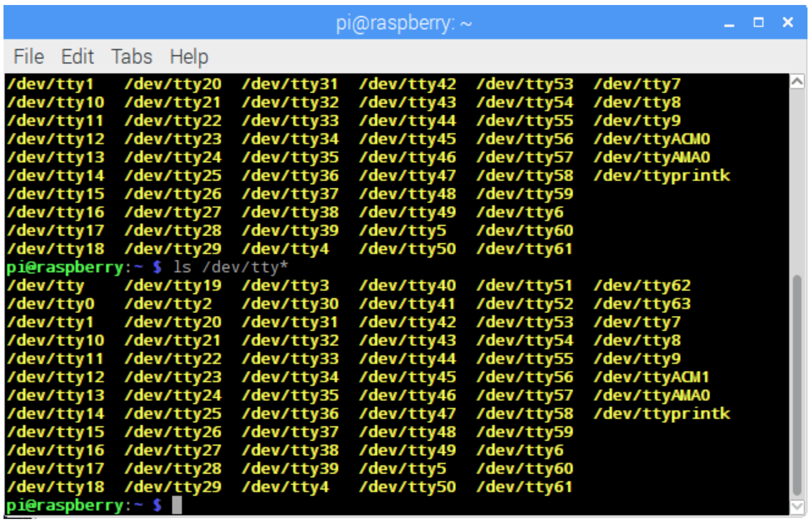


Camera: Enabled [✓ Disabled]
SSH: [✓ Enabled] Disabled
VNC: [✓ Enabled] Disabled
SPI: Enabled [✓ Disabled]
I2C: [✓ Enabled] Disabled
Serial: [✓ Enabled] [✓ Disabled]
1-Wire: Enabled [✓ Disabled]
Remote GPIO: Enabled [✓ Disabled]

Next, you'll need to restart your Raspberry Pi. Open the Terminal and execute these commands:

sudo apt-get install python-serial
sudo pip install pyserial
Connect your Arduino to your Raspberry Pi.Execute.
ls /dev/tty*

Then find a line with /dev/ttyACM0 or something like /dev/ttyACM1 etc. (check for an ACMwith any number 0,1,2 etc.)



```
/dev/tty1   /dev/tty20  /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10  /dev/tty21  /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11  /dev/tty22  /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12  /dev/tty23  /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyACM0
/dev/tty13  /dev/tty24  /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyAMA0
/dev/tty14  /dev/tty25  /dev/tty36  /dev/tty47  /dev/tty58  /dev/ttyprintk
/dev/tty15  /dev/tty26  /dev/tty37  /dev/tty48  /dev/tty59
/dev/tty16  /dev/tty27  /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17  /dev/tty28  /dev/tty39  /dev/tty5   /dev/tty60
/dev/tty18  /dev/tty29  /dev/tty4   /dev/tty50  /dev/tty61
pi@raspberry:~ $ ls /dev/tty*
/dev/tty    /dev/tty19  /dev/tty3   /dev/tty40  /dev/tty51  /dev/tty62
/dev/tty0   /dev/tty2   /dev/tty30  /dev/tty41  /dev/tty52  /dev/tty63
/dev/tty1   /dev/tty20  /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10  /dev/tty21  /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11  /dev/tty22  /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12  /dev/tty23  /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyACM1
/dev/tty13  /dev/tty24  /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyAMA0
/dev/tty14  /dev/tty25  /dev/tty36  /dev/tty47  /dev/tty58  /dev/ttyprintk
/dev/tty15  /dev/tty26  /dev/tty37  /dev/tty48  /dev/tty59
/dev/tty16  /dev/tty27  /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17  /dev/tty28  /dev/tty39  /dev/tty5   /dev/tty60
/dev/tty18  /dev/tty29  /dev/tty4   /dev/tty50  /dev/tty61
pi@raspberry:~ $
```

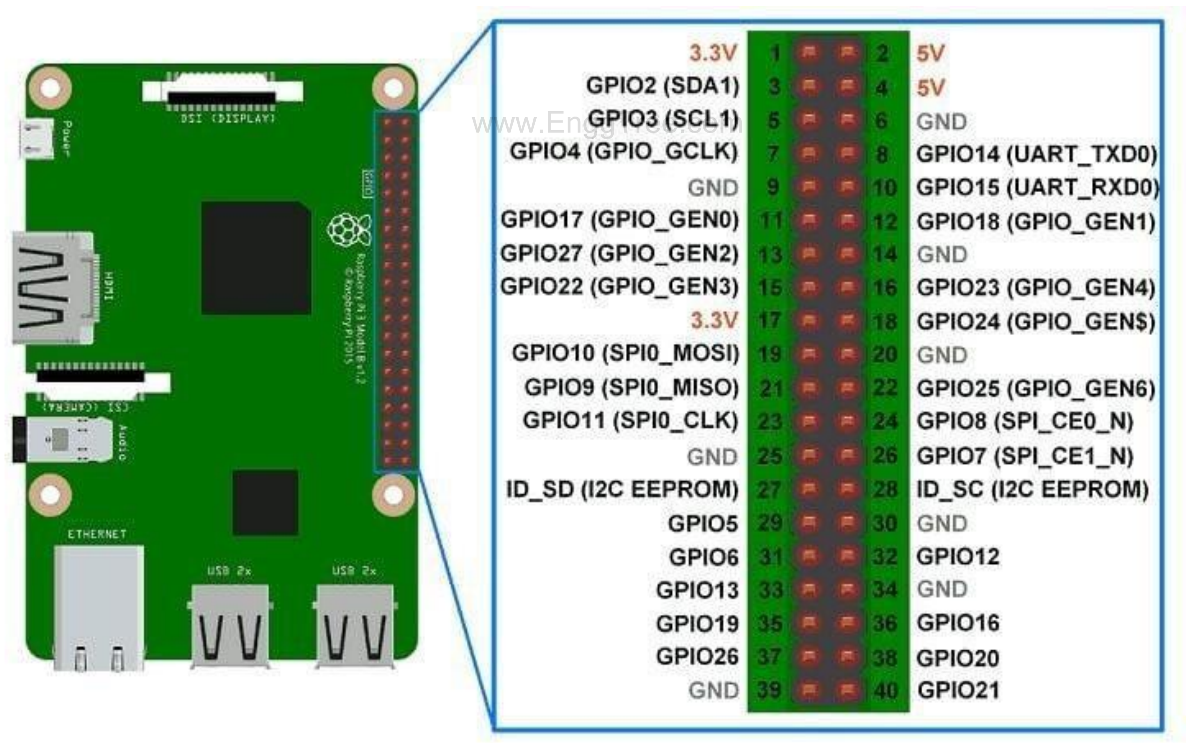
Open Python again and change ser=serial.Serial("dev/ttyACM1",9600) to the ACM number you found. So, if in your case you got ACM0, the line should look like this: ser=serial.Serial("dev/ttyACM0",9600). Now run the program you just created in Python3. You will see "Hello From Arduino!" in the Python terminal, and your LED should be blinking as well!

4.3.2 Raspberry Pi GPIO Access

GPIO (General Purpose Input Output) pins can be used as input or output and allows raspberry pito connect with general purpose I/O devices.

- Raspberry pi 3 model B took out 26 GPIO pins on board.
- Raspberry pi can control many external I/O devices using these GPIO's.
- These pins are a physical interface between the Pi and the outside world.
- We can program these pins according to our needs to interact with external devices. For example, if we want to read the state of a physical switch, we can configure any of the available GPIO pins as input and read the switch status to make decisions. We can also configure any GPIO pin as an output to control LED ON/OFF.
- Raspberry Pi can connect to the Internet using on-board Wi-Fi or Wi-Fi USB adapter. Once the Raspberry Pi is connected to the Internet then we can control devices, which are connected to the Raspberry Pi, remotely.

GPIO Pins of Raspberry Pi 3 are shown in below figure:



Power
DSI (DISPLAY)
HDMI
CSI (CAMERA)
Audio
ETHERNET
USB 2x
USB 2x
3.3V 1 2 5V
GPIO2 (SDA1) 3 4 5V
GPIO3 (SCL1) 5 6 GND
GPIO4 (GPIO_GCLK) 7 8 GPIO14 (UART_TXD0)
GND 9 10 GPIO15 (UART_RXD0)
GPIO17 (GPIO_GEN0) 11 12 GPIO18 (GPIO_GEN1)
GPIO27 (GPIO_GEN2) 13 14 GND
GPIO22 (GPIO_GEN3) 15 16 GPIO23 (GPIO_GEN4)
3.3V 17 18 GPIO24 (GPIO_GEN$)
GPIO10 (SPI0_MOSI) 19 20 GND
GPIO9 (SPI0_MISO) 21 22 GPIO25 (GPIO_GEN6)
GPIO11 (SPI0_CLK) 23 24 GPIO8 (SPI_CE0_N)
GND 25 26 GPIO7 (SPI_CE1_N)
ID_SD (I2C EEPROM) 27 28 ID_SC (I2C EEPROM)
GPIO5 29 30 GND
GPIO6 31 32 GPIO12
GPIO13 33 34 GND
GPIO19 35 36 GPIO16
GPIO26 37 38 GPIO20
GND 39 40 GPIO21

Raspberry Pi 3 Model B GPIO Pin Mapping

Some of the GPIO pins are multiplexed like I2C, SPI, UART etc. We can use any of the GPIO pins for our application.

Pin Numbering

We should define GPIO pin which we want to use as an output or input. But Raspberry Pi hastwo ways of defining pin number which are as follows:

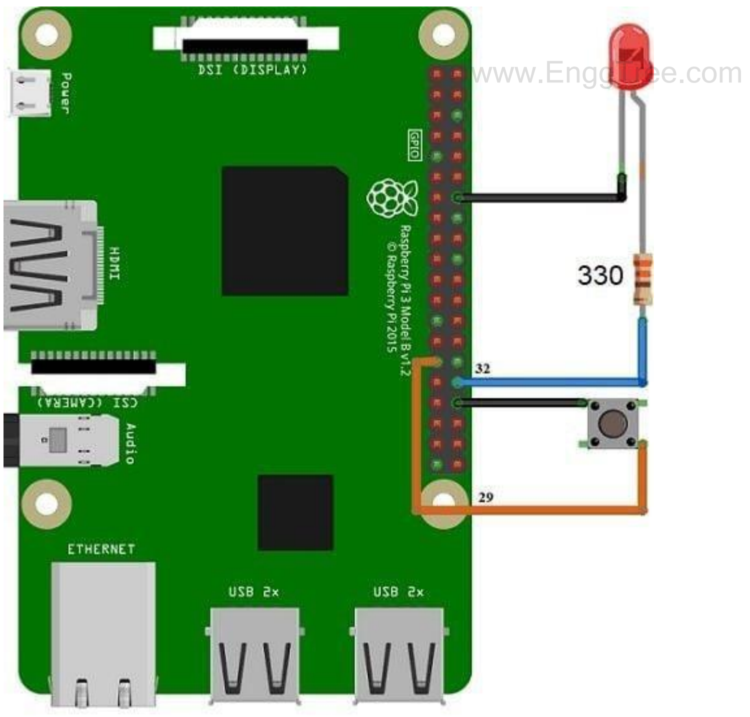
- **GPIO Numbering**
- **Physical Numbering**

In **GPIO Numbering**, pin number refers to number on Broadcom SoC (System on Chip). So, weshould always consider the pin mapping for using GPIO pin.

While in **Physical Numbering**, pin number refers to the pin of 40-pin P1 header on Raspberry PiBoard. The above physical numbering is simple as we can count pin number on P1 header and assign it as GPIO.

But, still we should consider the pin configuration diagram shown above to know which areGPIO pins and which are VCC and GND.

Control LED with Push Button using Raspberry Pi



Control LED using Raspberry Pi Interfacing Diagram

Example

Now, let’s control LED using a switch connected to the Raspberry Pi. Here, we are using Pythonand C (WiringPi) for LED ON-OFF control.

Control LED using Python

Now, let’s turn an ON and OFF LED using Python on Raspberry Pi. The switch is used tocontrol the LED ON-OFF.

Python Program for Raspberry Pi to control LED using Push Button

```
import RPi.GPIO as GPIO            #import RPi.GPIO module
LED = 32                           #pin no. as per BOARD, GPIO18 as per BCMSwitch_input = 29
                                   #pin no. as per BOARD, GPIO27 as per BCM
GPIO.setwarnings(False)            #disable warnings GPIO.setmode(GPIO.BOARD)     #set pin
numbering format GPIO.setup(LED, GPIO.OUT)   #set GPIO as output GPIO.setup(Switch_input,
GPIO.IN, pull_up_down=GPIO.PUD_UP)
while True: if(GPIO.input(Switch_input)):
GPIO.output(LED,GPIO.LOW)
else:
GPIO.output(LED,GPIO.HIGH)
```

Functions Used:

RPi.GPIO

To use Raspberry Pi GPIO pins in Python, we need to import RPi.GPIO package which has classto control GPIO. This RPi.GPIO Python package is already installed on Raspbian OS. So, we don’t need to install it externally. Just, we should include library in our program to use functionsfor GPIO access using Python. This is given as follows.

import RPi.GPIO as GPIO

GPIO.setmode (Pin Numbering System)

This function is used to define Pin numbering system i.e. GPIO numbering or Physicalnumbering.

Pin Numbering System = BOARD/BCM

E.g. If we use pin number 40 of P1 header as a GPIO pin which we have to configure as outputthen,

In BCM,

GPIO.setmode(GPIO.BCM)GPIO.setup(21, GPIO.OUT)

In BOARD,

GPIO.setmode(GPIO.BOARD)GPIO.setup(40, GPIO.OUT)

GPIO.setup (channel, direction, initial value, pull up/pull down)

This function is used to set the direction of GPIO pin as an input/output.**channel** – GPIO pin number as per numbering system. **direction** – set direction of GPIO pin as either Input or Output.

initial value – can provide initial value

pull up/pull down – enable pull up or pull down if requiredFew examples are given as follows,

- GPIO as Output
  GPIO.setup(channel, GPIO.OUT)
- GPIO as Input
  GPIO.setup(channel, GPIO.IN)
- GPIO as Output with initial value
  GPIO.setup(channel, GPIO.OUT, initial=GPIO.HIGH)
- GPIO as Input with Pull up resistor
  GPIO.setup(channel, GPIO.IN, pull_up_down = GPIO.PUD_UP) GPIO.output(channel, state)

This function is used to set the output state of GPIO pin.

channel – GPIO pin number as per numbering system.

state – Output state i.e. HIGH or LOW of GPIO pin

e.g.

GPIO.output(7, GPIO.HIGH)

GPIO.input(channel)

This function is used to read the value of GPIO pin.e.g.

GPIO.input(9)

Control LED using C (WiringPi)

We can access Raspberry Pi GPIO using C. Here, we are using WiringPi library for accessingRaspberry Pi GPIO using C.

Before implementing LED blinking using wiringPi, you can refer How to use WiringPi library.

C (WiringPi) Program for Raspberry Pi to control LED using Push Button

```
#include <wiringPi.h>
#include <stdio.h>

int LED = 26; /* GPIO26 as per wiringPi, GPIO12 as per BCM, pin no.32 */

int switch_input = 21;  /* GPIO21 as per WiringPi, GPIO5 as per BCM, pin no.29 */int main(){

wiringPiSetup();                          /* initialize wiringPi setup */ pinMode(LED,OUTPUT); /* set GPIO

as output */pullUpDnControl(switch_input, PUD_UP);

while (1){

if(digitalRead(switch_input))

digitalWrite(LED,LOW); /* write LOW on GPIO */else


}

}
digitalWrite(LED, HIGH);  /* write HIGH on GPIO */
```

4.4.1 Sending and Receiving Signals Using GPIO Pins

In recent years, the Raspberry Pi has become popular largely as an inexpensive, compact Linux box for media and retro video games, as well as a network device. Some hobbyists go on to use their Pi these ways for years, all without knowing what the pins on the side of their device really do.It's these pins that hold the true power of the Pi. They can control homes, machines, new inventions, and even robots, so why is it so many people don't know what they really are?

What Do These Pins Actually Do?

These 40 (or 26, depending on your Pi model) pins are part of what's known as the "GPIO Header". Within this header, there are four main kinds of pin;

- **Power**: These provide DC power at 3.3 and 5 volts
- **Ground (GND)**: These connect to ground, to close your circuit
- **DNC**: This stands for "do not connect", so don't worry about them
- **GPIO**: These can be set to either send, or receive control voltages

GPIO stands for 'General Purpose Input/Output', and it's these pins that let the Raspberry Pi do its magic. This is because the pins have no specific function, and can be set to a dedicated purpose, such as controlling a signal.

A GPIO pin set to output can provide either 3.3 volts, known as a HIGH signal, or 0 volts, known as a LOW signal. When set to input, it can read these same voltages.

GPIO Pins Don't Provide Much Power

It's important to remember that GPIO pins (and the 3.3-volt power pins) are meant to control and communicate with other components.

You can get about 51mA from all 3.3 volt pins combined, but you'll want to take care when connecting; if your circuit tries to pull too much current through these 3.3 volt pins, you can fry the whole board.

The 5-volt power pins, on the other hand, give you all the power available from the power supply, minus the bit used by the Raspberry Pi itself.

Connecting Your GPIO Pins to a Breadboard

When you first start using these GPIO pins, it's wise to use a breadboard. This makes it easy to build circuits without solder and to modify them.

If you've never used a breadboard before, familiarize yourself with the basics here:

A GPIO extension board also helps immensely. This connects the GPIO header via a ribbon and places the pins directly on the breadboard, in a clearly labeled manner.

It requires some real estate though: 20 rows each side of the breadboard. On a small board, that's nearly the whole thing! A breadboard with 40 rows or so leftover gives plenty of space for beginner projects.

GPIO Pins With Special Uses

Every GPIO pin can be set to send and receive HIGH and LOW signals. Some have special uses too.

Hardware PWM

GPIO pins output either 3.3 or 0 volts: a HIGH or LOW signal. Pulse width modulation, or PWM, is a way of simulating the range of voltages in between by flickering the pin on and off rapidly.

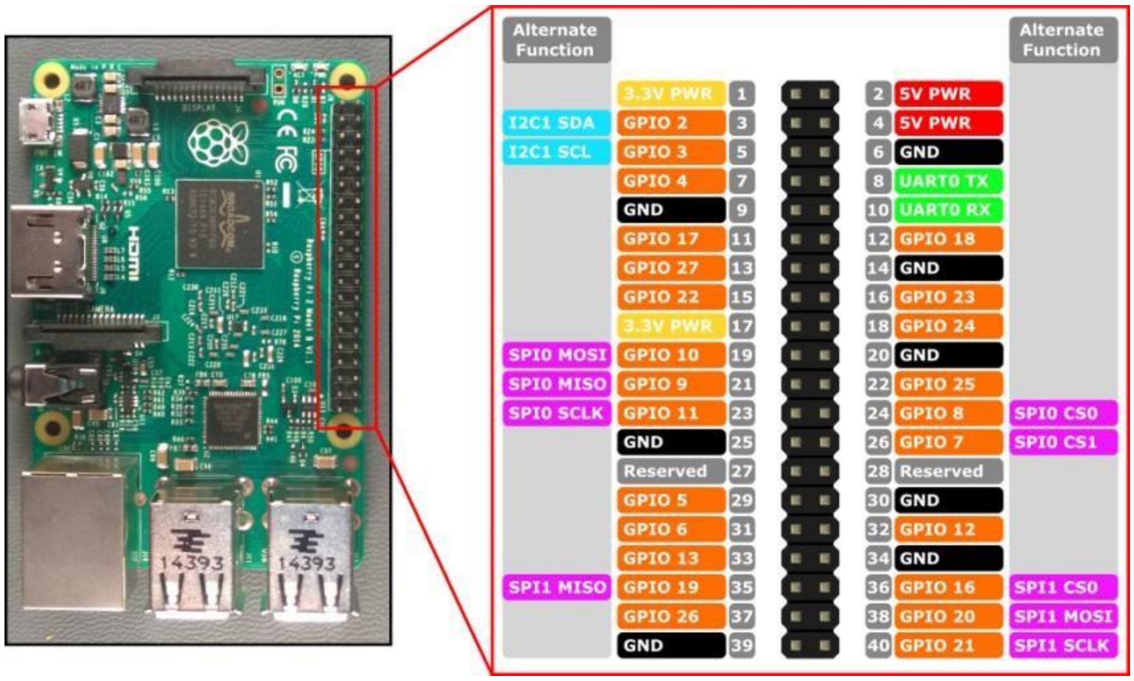
This isn't a true analog signal, but it's fine for something like dimming an LED. It will flicker faster than you can see and simply appear dimmer.

You can also use a low pass filter to smooth a PWM into an analog signal. This can be used for analogue audio, if you aren't fussy about quality. It's fine for a doorbell or toy.

You can generate a PWM signal from any GPIO pin using software, but the operating system juggles this with other tasks, so this signal can jitter.

Serial Bus Pins

If you take a look below at the diagram (known as a Raspberry Pi 'Pinout') you'll see that some pins are I2C, SPI, and UART serial. These are serial bus protocols that can be used to send and receive data from other components.



You can combine these with a digital-analogue converter, or DAC, to output an analogue signal. This can be preferable to PWM for high quality audio, or to control many components.

Pull Up and Pull Down Resistors

Often you will want your Raspberry Pi GPIO pin to read the position of a button or a switch. That's easy to do by wiring it so that it closes a circuit attached to the control voltage to read HIGH, or to ground to read LOW.

The problem is that when this circuit is open and nothing else is attached to the pin, it might return any value. This is known as "floating," and it's extremely unhelpful.

You can prevent floating with "pull up" or "pull down" resistors.

A pull up resistor is wired to your control voltage; when nothing else is attached, the pin will read HIGH. A pull down resistor is wired to ground; the pin will read LOW. Use whichever provides the opposite value to your switch or button.

You don't need to wire these resistors into your circuit. They're inside the Raspberry Pi already and you can control them from software.

Using Software to Control GPIO Pins

Among the easiest ways to control GPIO pins is by using the GPIO Zero library in Python. If you've written any Python before, you'll pick this up easy.
If this is your first time using Python, If you don't, the commands below will still work; you'll just be less able to follow along. The web version of 'Automate the Boring Stuff With Python' is excellent and costs nothing.

GPIO Zero is installed by default on Raspbian Desktop images. If you are using Raspbian Lite or a different operating system, you may need to install it.

Let's Use All This to Switch a Light On

Let's have a go at turning on an LED! A job this simple doesn't really require a computer, but we'll involve the Raspberry Pi in the GPIO pins.

Connecting the Power Rails

If you're using the extension board, connect it to the Raspberry Pi and to the breadboard. Then attach the 3.3-volt power pin to the positive power rail running across the bottom of the breadboard, and the ground pin to the negative power rail.

Connecting and Testing the Button

Now add your button to the middle of the breadboard. Connect one pin of the button to a Raspberry Pi GPIO pin. I'm using 13, because it's my lucky number.

Then, connect the diagonally opposite pin of the button to the negative power rail. When you push this button down, the circuit closes.

If you get a message saying 'ImportError', make sure you capitalized it correctly. If it says 'ModuleNotFoundError', you need to install GPIO Zero. Otherwise, it's time to assign our pin to the button:

```
button = Button(13)
```

This Button class takes care of assigning the pull up resistor. Now let's test that it works by typing the following lines:

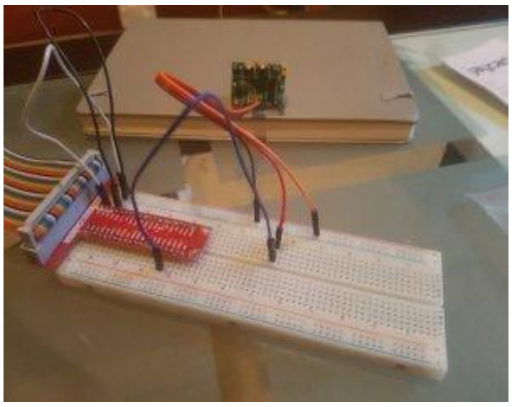
```
while True:
if button.is_pressed:
print('Sweet, the button works!')break
```

Python is fussy about indentation, so be sure to copy the spaces too. Then press Enter again torun the loop.

This loop will run until someone presses the button, so press it. It should produce a message saying the button works. This means you've successfully built a simple circuit that sends a message to your Raspberry Pi. If this doesn't work, check that everything is connected properly and try again.

Connecting and Testing the LED

The D in LED stands for "diode", which means electricity only runs in one direction through it. You will notice that one leg of the LED is slightly longer: this connects to positive. Here, that means connecting to the GPIO pin. I'm using pin 26, for no particular reason. Place the LED in the breadboard, making sure that the legs are spaced horizontally so that you aren't shorting the connection out. Now connect this positive leg to your GPIO pin.An LED should be wired in series with a resistor, so attach one end of your resistor to the short leg of the LED, and the other end to the negative power rail. Resistors can go in either way around.



Now let’s go tell the Raspberry Pi what’s going on. Type: Image: Pi & Breadboard

```
from gpiozero import LEDled = LED(26)
```

If everything’s wired correctly, you can switch it on and off with these commands:

```
led.on()
led.off()
```

Controlling the LED With the Button

Now that everything’s connected and you’ve checked they work, type:

```
button.when_pressed = led.on
```

Now press the button. The LED should switch on and stay on. Now type:

```
button.when_released = led.off
```

Press the button again; the LED should switch off when the button is released.

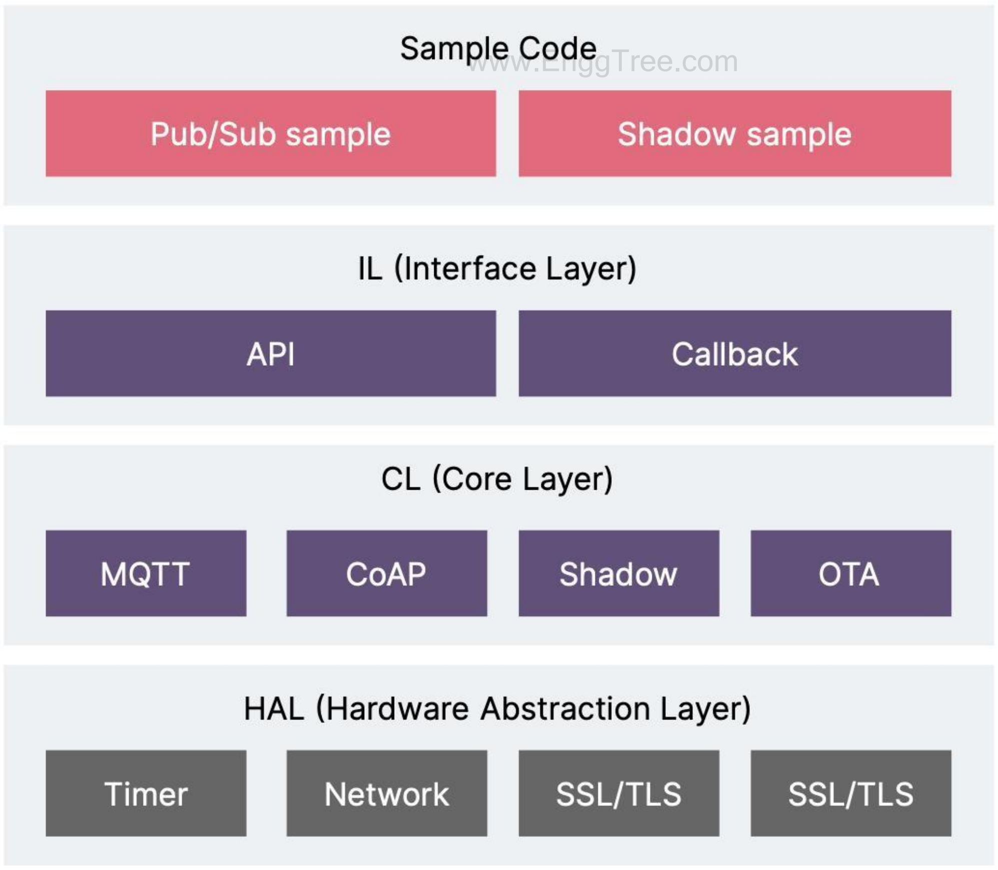
4.5.1 Connecting IOT devices to the cloud

With the development of smart things, sensors and telecommunication, IoT (Internet of Things) technology has greatly developed and become more and more standardized. But fragmented device-side communication connection problems often impede the project implementation process.

There are four best practices to connect different types of devices to the cloud:

- Directly integrate IoT SDK (Software Development Kit) for resource-rich devices
- Rely on a communication module for resource-constrained devices
- Use a local gateway for non-network devices
- Use a cloud gateway for private-protocol devices with network capabilitiesFirst, we should introduce the IoT Device SDK.

IoT Device SDK is used to help us quickly connect hardware devices to the IoT platform. We can download the IoT Device SDK from the corresponding cloud platform, e.g. AWS IoTDevice SDK.



There are 4 layers of the IoT SDK. From bottom to top:

1. The HAL (hardware abstraction layer) abstracts the support function interface of different OS (operating systems) to the SDK. This enables the SDK to be ported to different hardware environments, different OS, and even bare chip environments.

2. The core layer completes the function encapsulation of MQTT/CoAP communication based on the HAL layer interface, including MQTT connection establishment, message sending and receiving; CoAP connection establishment, message sending and receiving; shadow device operation; OTA firmware status query, download and upgrade.

3. Interface layer, providing API and callback function definitions, isolating the core layer and the application.

4. Provide sample programs so that developers can quickly learn how to use the SDK.

When developing applications on a device, we can always choose higher-level SDKs such as Android IoT SDK on an Android device. That's because the hardware environment porting work has already been done by the SDK itself.

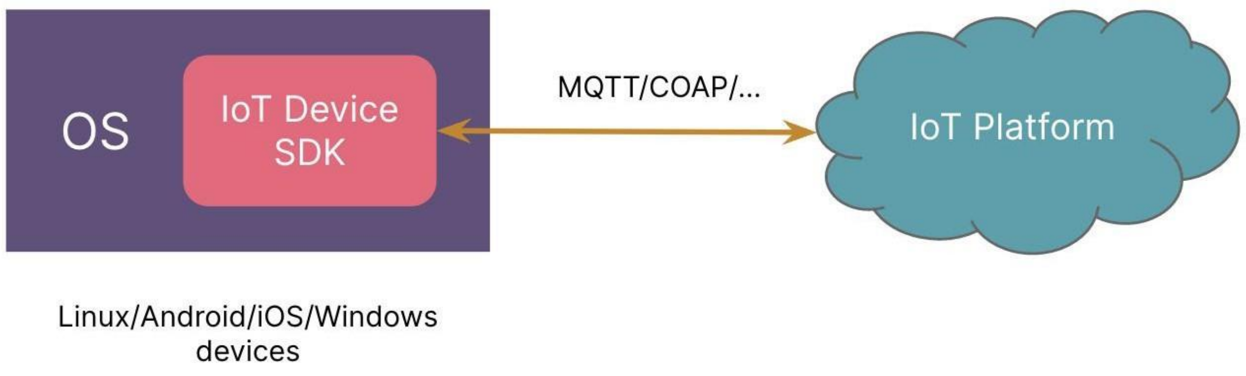
However, when developing applications for an MCU (Micro Controller Unit) which has Linux or RTOS (Real Time Operating System), we should choose Embedded C SDK and port the code to a specific hardware environment.

With the knowledge of IoT SDK, we can discuss how to connect an IoT device to the cloud.

Directly integrate IoT SDK for resource-rich devices

With the development of high-performance hardware, many smart devices have complete OS such as Linux, Android etc. These devices also have a Wi-Fi or cellular network.

At the operating system level, network communication problems have already been resolved. We only need to develop applications which integrate the IoT SDK of the cloud platform and the communication link with the cloud will have been established.



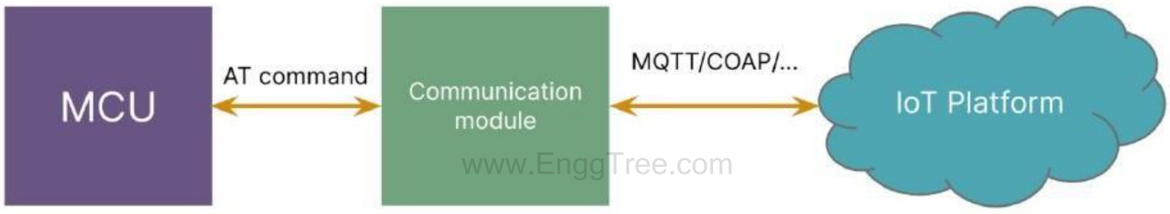
Examples of smart devices include smart phones, tablets, smart wearables, smart POS, computers, industrial gateways and development boards like the Raspberry Pi and ESP32.

Rely on a communication module for resource-constrained devices

In the IoT scenario, a large number of devices are resource-constrained, with RTOS, or even without an operating system, using MCU + communication modules to establish their link to the cloud.

There are many suppliers of cellular modules (NB-IoT/2G/3G/4G) on the market. The AT commands of each company are different, which makes developing device-side applications very difficult.

When we need to connect MCU to an IoT Platform, we should always carefully select the cellular modules and check whether they are suitable for a specific IoT platform.

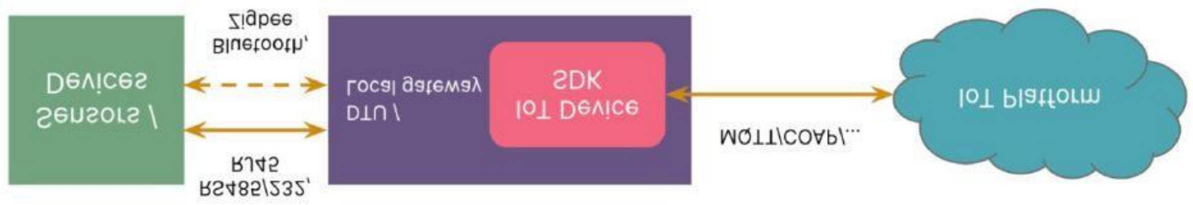


Difference between DTU and industrial gateway:

DTU is a wireless terminal device used to convert serial data into IP data or IP data into serial data and transmit it through a wireless communication network. It has fast and flexible networking, a short construction period and low cost.

The industrial gateway has the functions of collecting data from field devices through serial port or network port. Data collection, protocol analysis, data standardization and uploading to the IoT platform through edge computing functions. Which is more flexible, powerful and customizable than DTU, but it is more expensive and needs more resources to maintain.

Examples of sensors / devices which can be connected to DTU or industrial gateway include sensors, industrial equipment PLCs (Programmable Logic Controller), Bluetooth bracelets

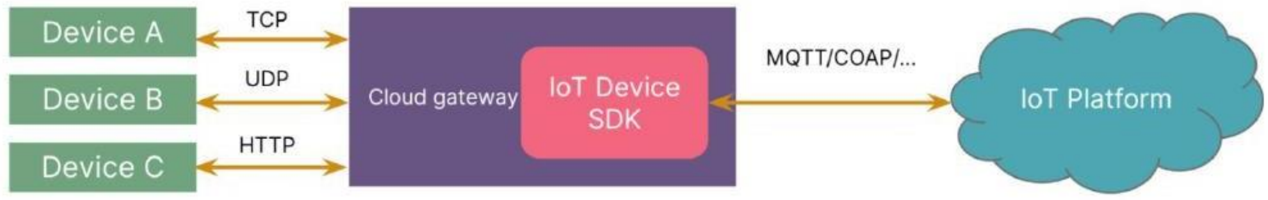


Use a cloud gateway for private-protocol devices with network capabilities

Devices that directly connect to the cloud gateway

For some devices, they already have the ability to connect to the internet, but the protocols vary according to device manufacturers. We don’t want the IoT platform layer to handle the parsing of these protocols directly;an intermediate layer should satisfy the protocol conversion work to make the data meet the unified format of the IoT platform.

This intermediate layer is the cloud gateway. The cloud gateway is at the front of the platform. It receives data from the device side, completes message parsing, and then sends a message to the IoT platform with IoT device SDK.

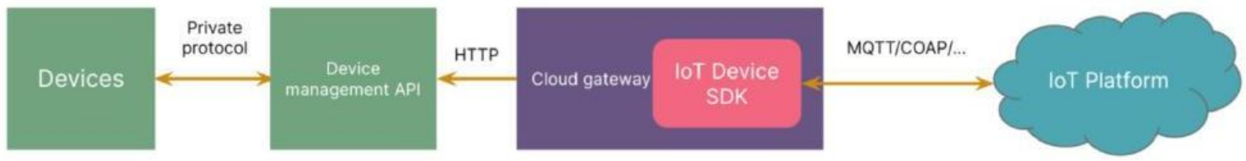


Examples of devices that connect to the internet through private protocol include vehicle GPS.

Devices that connect to the cloud gateway indirectly

Some device vendors already provide a mature system which manages the devices and provides API. In this case, we can have another form of cloud gateway: Cloud to cloud connection. Making full use of the existing system will make the overall system more stable and give clear rights and responsibilities.

Using a mature system will bring us higher development efficiency, but it’ll also introduceanother midware, which will increase communication time.

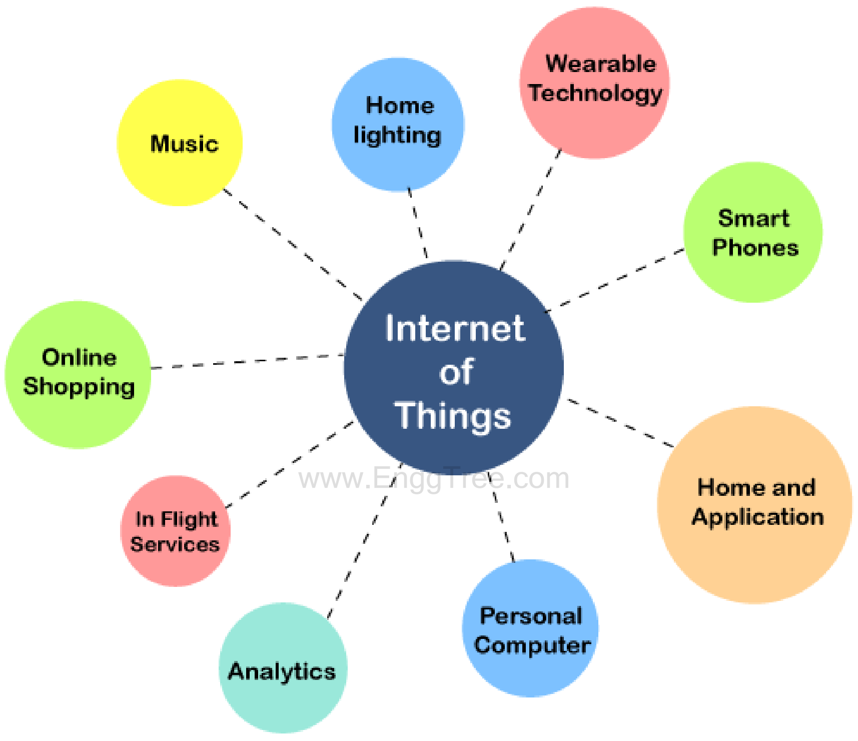


Examples of devices with mature systems include cameras/NVR systems.

UNIT V IOT APPLICATIONS

5.1.1 Business models for the internet of things

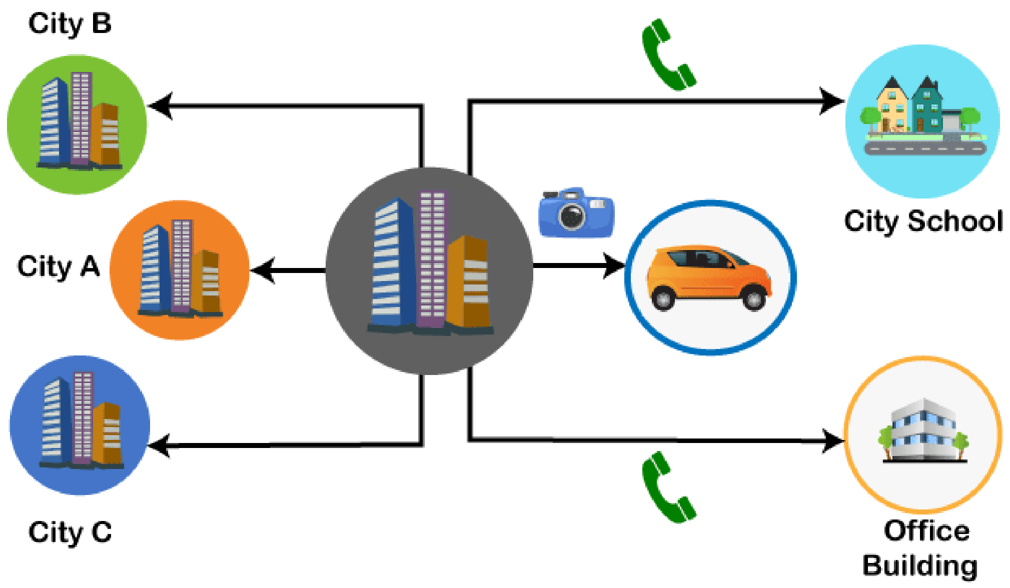
The **Internet of Things (IoT)** provides the ability to interconnect computing devices, mechanical machines, objects, animals or unique identifiers and people to transfer data across a network without the need for human-to-human or human-to-computer is a system of conversation. **IoT applications** bring a lot of value in our lives. The Internet of Things provides objects, computing devices or unique identifiers and people's ability to transfer data across a network without the **human-to-human** or **human-to-computer interaction**.



A traffic camera is an intelligent device. The camera monitors **traffic congestion, accidents** and **weather conditions** and can access it to a common entrance.This gateway receives data from such cameras and transmits information to the city's **traffic monitoring system**.

For example, the municipal corporation has decided to repair a road that is connected to the national highway. It may cause traffic congestion to the national highway. The insight is sent to the traffic monitoring system.

The intelligent system analyzes the situation, estimate their impact, and relay information to other cities connected to the same highway. It generates live instructions to drivers by smart devices and radio channels.



It creates a network of **self-dependent systems** that take advantage of real-time control.

5.1.2 Smart city

A smart city uses information and communication technology to improve the utility, share knowledge with the public, and provide strong sense of community support and local government assistance. Shrewd urban communities are those that make use of brilliant ideas and information as the required resources to address the maintainability issues that urban communities face. Many metropolitan areas are currently becoming more intelligent, utilizing information and innovation to advance transportation, energy consumption, wellness, and air quality, as well as to spur economic growth.

A great city's main objective is to streamline municipal operations, promote economic development, and address resident happiness through clever developments and data analysis. We intended to spend a great amount of time reading up on several shrewd urban groups in this post. As a result, some of the key boundaries that can be built include clever management, clever energy, clever building, clever flexibility, clever structure, clever invention, clever medical care, and clever residence.

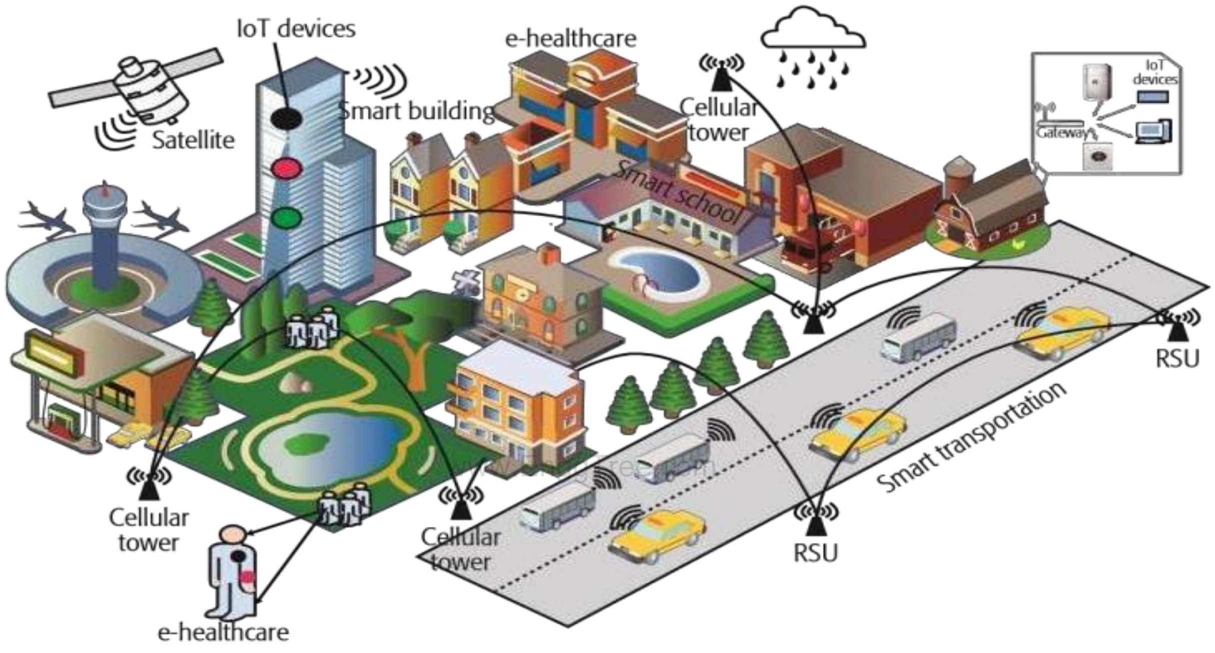
Urban areas collect and analyze information using IoT devices such as connected sensors, lighting, and meters. The foundation, public usage, and administrations, to name just a few, are all progressively developed in urban areas using this knowledge. Smart urban communities focus on improving the lives of their residents in such fundamental areas as strategy effectiveness, reducing waste and everyday problems, improving friendly and financial quality, and enhancing the social consideration of their residents.

Advantages of Smart City:

- Automatic Switching of Street lights.
- Maintenance Cost Reduction.
- Reduction of light pollution.
- Keep the city clean.
- Improve traffic and reduce parking times.
- Reduction of manpower.

IoT-based smart city

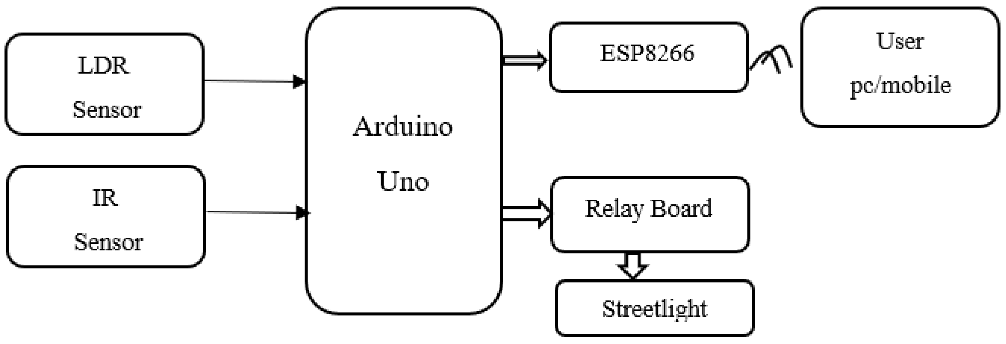
The following Figure provides and illustration of an IoT-based smart city.



An illustration of an IoT-based smart city

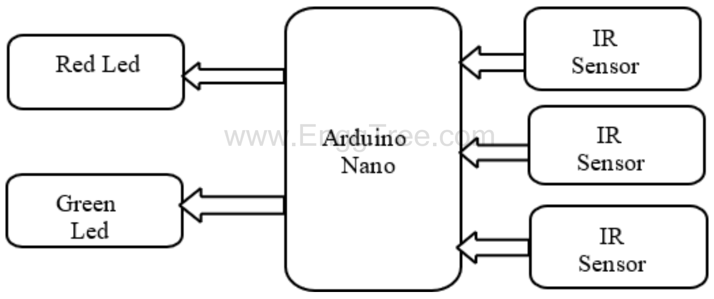
The IR sensors, LDR, PIC16F877A microcontroller, relay, UART, and Wi-Fi module make up the ingenious street lamp's construction. LDRs are light-dependent devices, and their blockage grows in the dark and shrinks when exposed to light. A light-dependent resistor hasa high resistance when maintained dull. The vehicle that is passing the streetlight is recognized by an IR sensor. The streetlight bulb can be turned on and off during the transfer.

The Universal Asynchronous Receiver/Transmitter (UART) software on a microcontroller manages the PC's connection to the associated streetlight framework.



Block diagram of smart street lighting system [5]

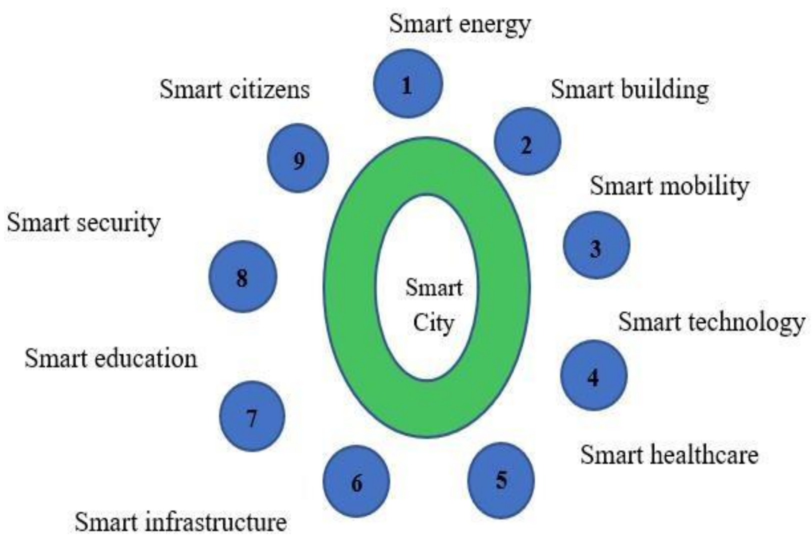
The clever street lamp's structure is made up of IR sensors, LDR, PIC16F877A microcontroller, relay, UART, and Wi-Fi module. LDRs are light-dependent devices whose blockage expands in darkness and decreases when light shines on them. When a light-dependent resistor is kept dull, its resistance is quite high. An IR
sensor identifies the car that is driving past the streetlight. During the transfer, the streetlight bulb can be turned on and off. A microcontroller with software known as a UART (Universal Asynchronous Receiver/Transmitter) controls a PC's connection point to its connected streetlight framework.



Block diagram of smart parking system [8]

It is divided into three areas. The parking area is the first, and it includes an IR sensor and Arduino devices. With the aid of these devices, the client establishes a connection with the halting location. Without the aid of an RFID card, the user is unable to enter the parking space. The cloud-based web administrations, which serve as a go-between for the client and the stopping region, are covered in the following section. Depending on whether a parking space is available, the cloud is updated. The user can view the admin to see if the cloud services are available, and the admin manages the cloud services. The user side is the third section.

Heterogeneous hardware permits the mechanization of comparative and customary workouts using the IoT stage in homes and buildings. The execution of administrations via web interfaces is undoubtedly possible when transforming items into the information of apparatuses that are completely connected via the Internet. Huge numbers of sophisticated home applications use sensor networks . The government (at the municipal, state, and federal levels) should deploy IoT services in all crucial issue areas to enhance government information systems and administration.

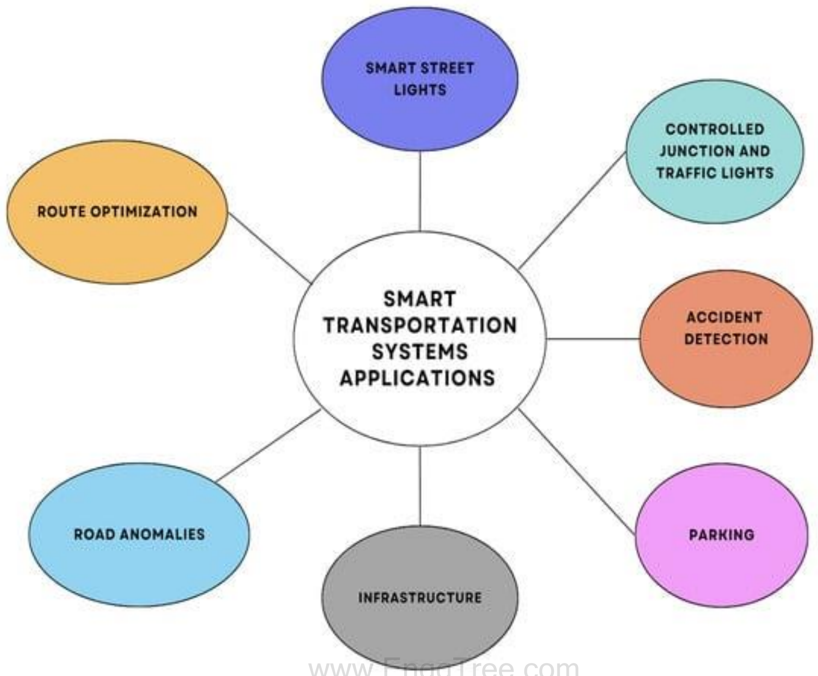


IoT based interconnections

The latest developments in computerized innovations have made shrewd urban regions even more shrewd than previously. A smart city has sensors for transportation systems, road cameras for perception systems, and other electronic components that are used in numerous applications. Additionally, this may increase the use of personal cell phones. This way, different concepts like article highlights, givers, inspirations, and security standards should be investigated while accounting for the diverse climate.

5.2.1 Smart mobility and transport

we examine the main current smart transportation systems. These systems are divided into seven classes based and their functionality as depicted in Figure We divided these systems to provide a structured understanding of the different types of systems and compared their functionalities pertaining to smart transportation systems.



Route Optimization

Urban regions frequently have traffic congestion, which is only worsening as more vehicles are added to the road. In order to reduce traffic congestion, route optimization proposes the optimum path for a given destination. Both the amount of time it takes to travel and vehicle emissions are decreased by reducing traffic congestion. The route optimization problem has been widely challenged and researched in the literature by applying various technical approaches to the IoT infrastructure.
Google was one of the first companies to harness the potential of crowdsourcing for developing new services. All modern mobile devices are compatible with the free Google Maps app. Integrated GPS, accelerometer, and gyroscope sensors are found in mobile devices. In 2009, Google unveiled a brand-new service that would provide users access to traffic data within Google Maps. Fixed location sensors or other monitoring systems did not gather the traffic data. Using the maps application, the end user's mobile device can submit anonymous information about their location and speed. To reduce congestion, Google Maps can now recommend other routes based on traffic data.

Parking

By eliminating the need to hunt parking lots in search of an open spot, making it easier to find available places in advance helps lessen traffic and pollution [29]. Many parking applications are created to monitor parking lot availability efficiently, provide users with reservation options, and even incorporate parking detection and alerting systems. Many IoT devices have

been employed to detect the presence of a car in a parking spot and convey the information to a centralized system. Additionally, other studies apply ML algorithms that use image data to detect free parking slots massively. Saarika et al. [57] proposes a smart parking strategy with the concept of an IoT-supported parking lot and a smart signboard to display pertinent information.
Ultrasonic sensors in the parking lot will determine whether parking spaces are available, and a WiFi module will gather and transfer the data to a cloud server. A user can now utilize a smartphone application or a smart signboard to check parking availability. The signboard is an LCD or LED display powered by a Raspberry Pi that will gather and show data on parking accessibility, weather conditions, travel times to specific locations, etc.
To determine availability, the authors in [108] also place ultrasonic sensors at each parking space. The sensor is linked to an Arduino Uno, which uses an ESP8266-01 WiFi module to transmit data to a cloud server. The MQTT protocol is used for communication. The cloud server runs Thing Speak, an IoT platform that provides customers with various management and monitoring options. Last but not least, customers can download an Android app that enables them to reserve parking spaces and automate parking payments.

Lights

Smart Street Lights (SSL) are a crucial component of a smart city and are included in the category of smart transportation services. Smart lighting can save energy while providing dynamic functionality and manageability. We implements an SSL implementation based on IoT technology. By including a light sensor, an IR sensor, GPS, and a wireless connection module, streetlights acquire smart features. By being aware of congested locations and dynamically adjusting their light intensity, lamps can make densely populated areas safer while simultaneously using less energy.
When the street light breaks, the GPS can let a centralized system tracks its location and condition and expedite maintenance procedures. The NB-IoT network serves as the foundation for the communication between the management system and SSL. The management system is built on fog nodes, which gather information from a number of bulbs and periodically assess their condition.
In addition to the automatic processes that SSLs offer, they can also be remotely administered via the established management platform. Kokilavani and Malathi presents a similar and simpler method for smart lights. This design connects the lamp with a light sensor, an IR sensor, and an IR led using a raspberry pi as the microcontroller.
The sun’s rise and set will be detected by the light sensors, which will then turn on and off the bulb. In order to save energy, the lighting can also recognize passing vehicles or pedestrians and switch the lamps on and off dynamically.

Controlled Junction and Traffic Lights

A controlled junction uses traffic lights to control when vehicles may enter the junction. This is done in an effort to smooth access to a traffic jam on the route. Sensors are frequently used to control traffic signal junctions. These sensors identify areas where traffic accumulates as it approaches the junction and then extend the green light to allow for more vehicles to pass through. Transponders installed in junctions can also be used to prioritize entry to the junction so that emergency vehicles and public transportation can move through the junction more quickly.

By carefully regulating the timing of traffic signals and the speed of approaching cars, intersection control tries to maximize junction throughput and reduce stopping time.
The authors in the research suggest a revolutionary decentralized traffic light control system that utilizes wireless sensor networks.
The wireless sensor network, the localized traffic flow model policy, and the higher-level coordination of the traffic light agents are the three levels of the system architecture. The nearest Intersection Control Agent (ICA) receives data from the wireless sensors, which track the number, speed, and other characteristics of passing cars, and uses it to estimate the intersection's flow model.
The real-time adaptive control of the traffic signals is the key contribution. This will also enhance the movement of cars. By regulating the traffic lights, an intersection control agent controls the intersection. To control a larger area, several intersection agents can communicate with one another.

5.2.2 Industrial IoT

The Industrial Internet of Things (IIoT) is the collection of sensors, instruments and autonomous devices connected through the internet to industrial applications. This network makes it possible to gather data, carry out analyses and optimise production, increasing the efficiency and reducing the costs of the manufacturing process and the provision of services. Industrial applications are complete technological ecosystems that connect devices and these with the people who manage the processes in assembly lines, logistics and large-scale distribution.

Current IIoT applications are primarily concentrated in manufacturing, transport and energy, with **an investment of over 300 billion dollars worldwide in 2019 which is expected to double by 2025.** In the immediate future it is expected that the adoption of the IIoT will result in the implementation of more industrial robots, such as cobots, warehouse and transport control systems, and predictive maintenance systems.
The difference between the Internet of Things (IoT) and its industrial version (IIoT) is that while IoT focuses on services for consumers, **IIoT focuses on increasing safety and efficiency at production sites.** For example, consumer solutions have focused on smart devices for the home, from virtual assistants to temperature sensors or security systems, or for people, such as wearables that monitor health.

CHARACTERISTICS OF THE INDUSTRIAL INTERNET OF THINGS (IIOT)

Not all systems can be classified as IIoT. In general, they need to be networked systems that generate data for analysis and produce concrete actions. The operation of IIoT systems is based on a layered structure:

Devices. The visible part of the system is the devices: sensors, GPS locators, machines, among others.

Network. Above this is the connectivity layer, i.e. the network that is established between these devices and the servers through cloud storage or edge computing.

Services. These are computer applications that analyse the data collected and process them to offer a specific service.

Content. This is the interface with the human operator, which can be a computer, a tablet or even devices such as virtual reality or augmented reality glasses.

APPLICATIONS AND SOLUTIONS OF THE INDUSTRIAL INTERNET OF THINGS (IIOT)

The applications of the Internet of Things in industry are varied, but below we review some of the most relevant:

- **Use of autonomous vehicles**
  The transport of components to the plant or products to the warehouse can be done by autonomous vehicles that are able to move from one side of the factory to the other by detecting obstacles.

- **Optimisation of machine performance**
  An inactive machine represents a loss of revenue. Thanks to sensors and data processing, it is possible to optimise machine utilisation time inside a manufacturing plant.

- **Reduction of human errors**
  Human operators will continue to be essential for many tasks, but the tools they use will be connected to the system to save time and avoid errors.

- **Improvement in logistics and distribution**
  Stored products incorporate sensors that provide real-time data on their location and even on their temperature and surrounding conditions which will be particularly useful during, for example, the distribution of the COVID-19 vaccine.

- **Decrease in the number of accidents**
  Wearables, such as goggles, bracelets and gloves, allow data to be collected from the operator wearing them. Examples of this data range from their location or proximity to machines, to their pulse, temperature and blood pressure, thereby reducing the possibility of accidents.

IBERDROLA

The impact of the Internet of Things
on some industries

ELECTRICITY
- Plant automation
- Checking of the condition of equipment
- Maintenance of sustainable environments

TRANSPORT
- Real-time management of orders and shipments
- Automated vehicle tracking
- On-demand inventory management

GAS AND OIL
- Measurement of crude oil extraction ratios
- Connected oil and gas pipelines
- Remote asset monitoring and management

HOSPITALITY
- Room automation
- Improvements in room service
- Integration between resorts

WATER
- Water flow controllers
- Waste water treatment
- Smart consumption management

Source: DZone.

5.3.1 Smart health

The healthcare monitoring systems has emerged as one of the most vital system and became technology oriented from the past decade. Humans are facing a problem of unexpected death due to various illness which is because of lack of medical care to the patients at right time. The primary goal was to develop a reliable patient monitoring system using IoT so that the healthcare professionals can monitor their patients, who are either hospitalized or at home using an IoT based integrated healthcare system with the view of ensuring patients are cared for better.

A mobile device based wireless healthcare monitoring system was developed which can provide real time online information about physiological conditions of a patient mainly consists of sensors, the data acquisition unit, microcontroller (i.e., Arduino), and programmed with a software (i.e., JAVA). The patient's temperature, heart beat rate, EEG data are monitored, displayed and stored by the system and sent to the doctor's mobile containing the application. Thus, IoT based patient monitoring system effectively monitor patient's health status and save life on time.

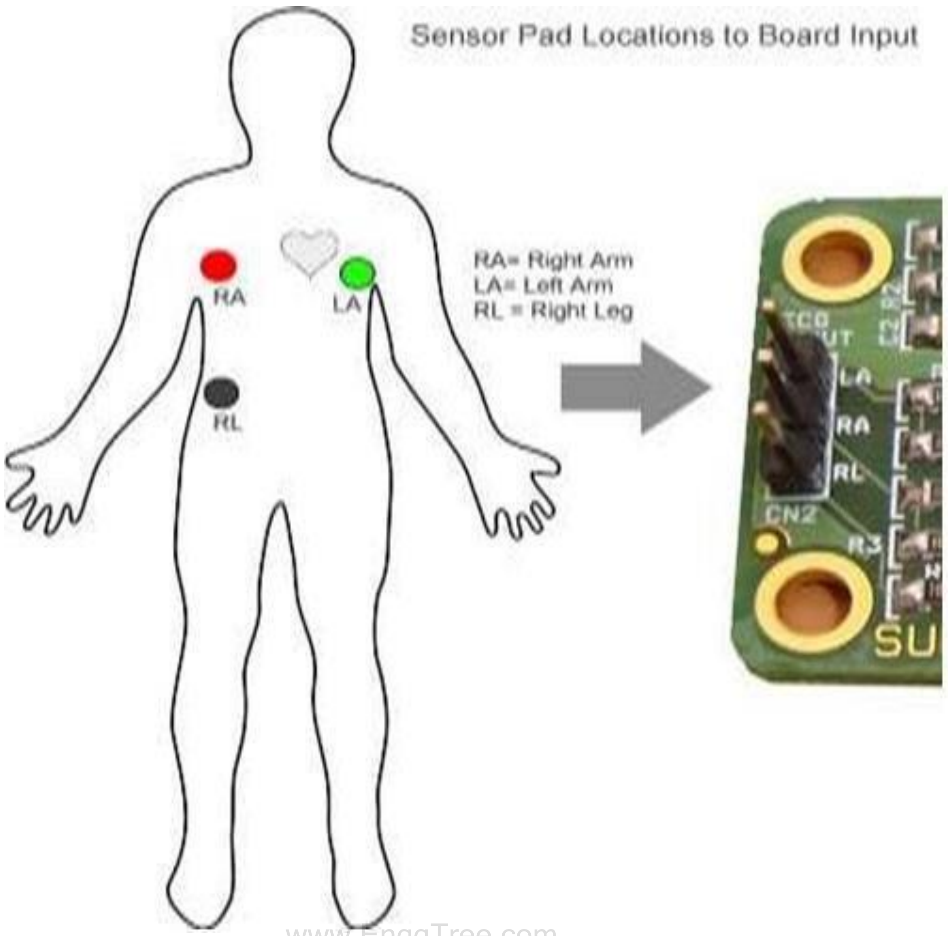
INTRODUCTION

The increased use of mobile technologies and smart devices in the area of health has caused great impact on the world. Health experts are increasingly taking advantage of the benefits these technologies bring, thus generating a significant improvement in health care in clinical settings. Likewise, countless ordinary users are being served from the advantages of the M-Health (Mobile Health) applications and E-Health (health care supported by ICT) to improve, help and assist their health.

According to the constitutions of World Health Organization (WHO) the highest attainable standard of health is a fundamental right for an individual. As we are truly inspired by this, we attempt to propose an innovative system that puts forward a smart patient health tracking system that uses sensors to track patient vital parameters and uses internet to update the doctors so that they can help in case of any issues at the earliest preventing death rates.

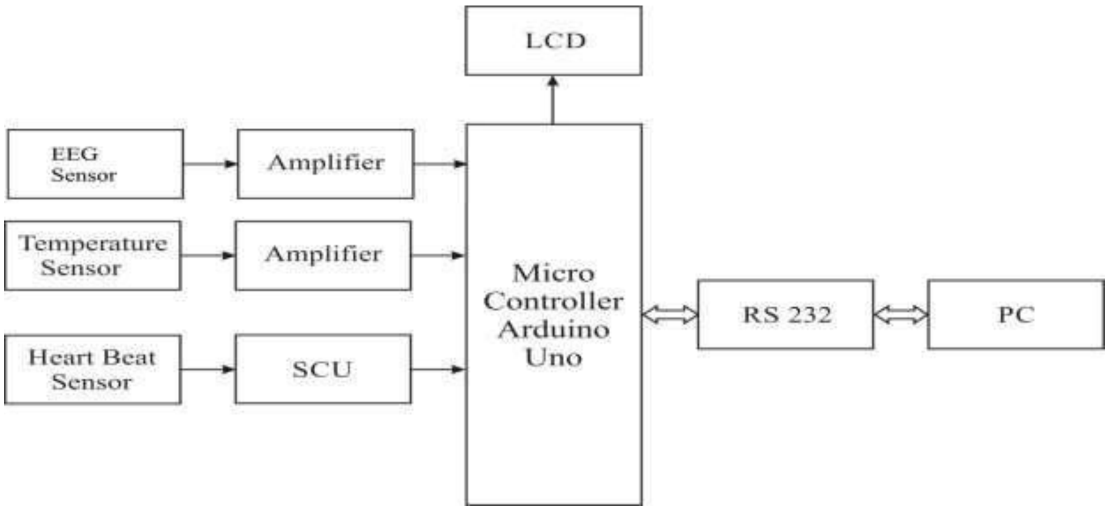
Signal Conditioning Unit

This sensor is a cost-effective board used to measure the electrical activity of the heart. This electrical activity can be charted as an ECG or Electrocardiogram and output as an analog reading. ECGs can be extremely noisy, the AD8232 Single Lead Heart Rate Monitor acts as an op amp to help obtain a clear signal from the PR and QT Intervals easily.

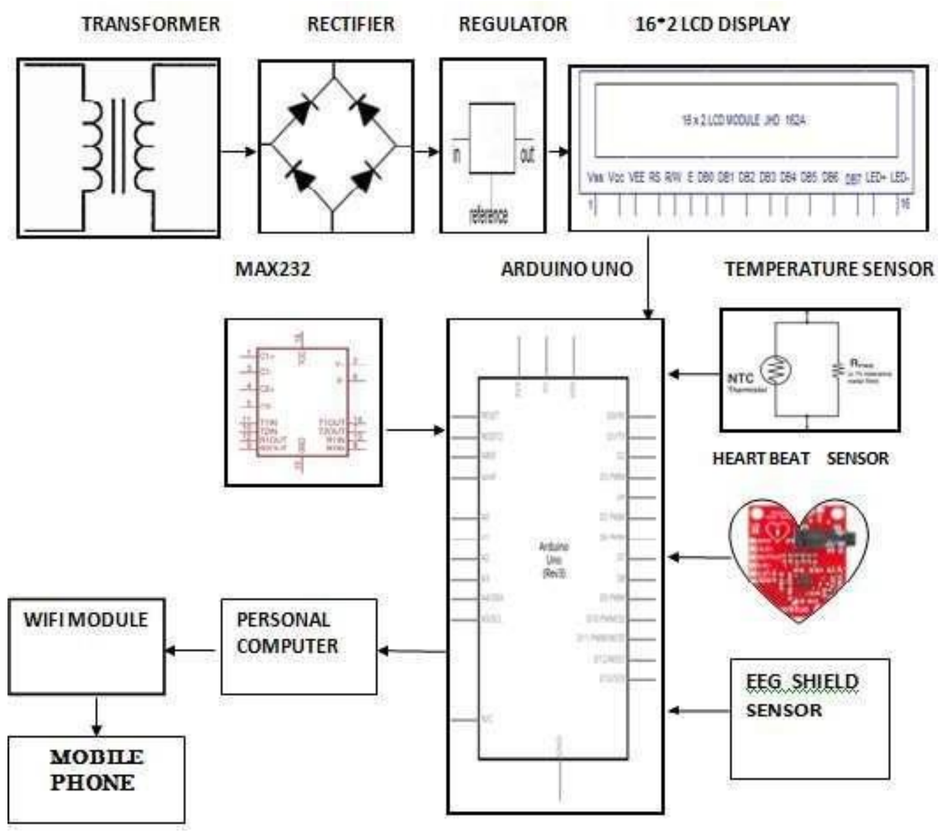


Position of Sensor Pads Input

Block Diagram



Block diagram of sensors connected with the PC



Block diagram of Health monitoring system

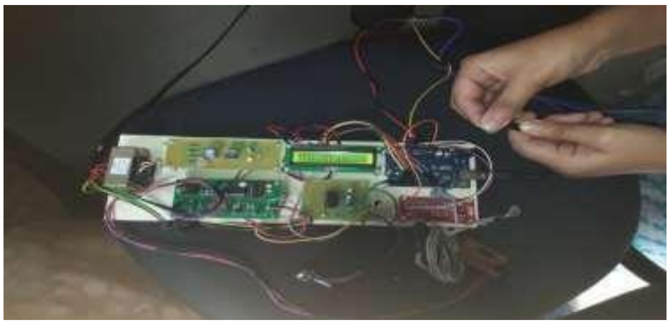
Operating Mechanism

STEP 1: The Heartbeat sensor is fixed to the patient's finger. This contains an IR sensor in it .Every pumping we get pulse from that sensor. This sensor output is given to the arduino viaSignal conditioning unit for amplification

Heart beat sensor

STEP 2

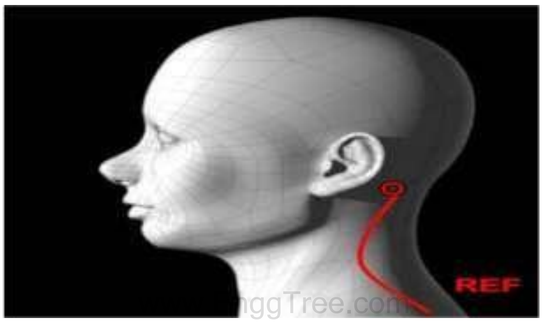
NTC type thermistor is used as a temperature sensor. This temperature sensor output variesbased on the temperature, this output is also given to arduino.



Temperature sensor

STEP 3

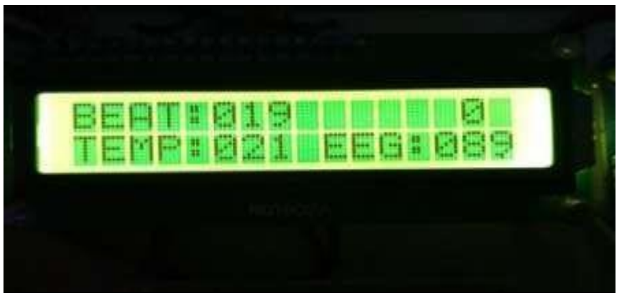
EEG sensor is a cost-effective board used to measure the electrical activity of the heart. This electrical activity can be charted as an ECG or Electrocardiogram output as an analog reading. ECGs can be extremely noisy, the AD8232 Single Lead Heart Rate Monitor acts as an op-amp to help obtain a clear signal from the PR and QT Intervals easily and connected to arduino.



EEG sensor

Step 4

All these values are transferred to PC via RS 232 and by using the URL,it is transferred to the mobile app created.



Output in LCD

Output in the Mobile Application:

The output is displayed in the form of string in a particular interval of time. The application is very simple as it just displays the analog values followed by a statement describing the kind of value displayed.

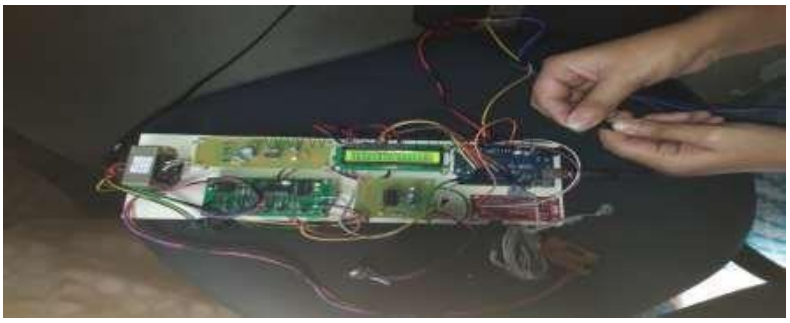
Output displayed in the mobile application device

Testing and findings health care unit

The Patient Health Monitoring System developed is tested using various persons with normal to abnormal health conditions. The various testing and findings producing results with minimal error rate and the observations are listed below.

Temperature Findings

The NC type thermistor used is programmed to display the value at room temperature for demo purposes with minimal error of + or – 5.



5.3.2 Environment monitoring and surveillance

IoT environmental monitoring is a process that uses Internet of Things (IoT) technology to collect data about the environment, such as air quality, temperature, and humidity levels.
This data can then be analysed to better understand the indoor and outdoor environment and make informed decisions about how to reduce the impact of negative aspects of the local environment on the business. Alternatively, it can be used to change business activities to help protect the planet or the local community.
These IoT-based systems can be used to detect issues in the environment that are largely invisible, normalised or taken for granted. Allowing businesses to take action by reducing their negative environmental footprint and protecting employees, visitors and the community at large.
IoT connectivity plays a crucial role in Environmental Monitoring by enabling the collection and transmission of real-time data from various sensors and devices. IoT devices such as air quality monitors, water quality sensors, and weather stations are commonly used to monitor environmental parameters. These devices utilise specific connectivity technologies such as GSM, 4G LTE, LoRa, SigFox, and NB-IoT to ensure that data is efficiently transmitted to the cloud for analysis. This real-time data allows for better decision-making, early detection of environmental issues, and ultimately helps in the conservation and protection of our natural resources. With IoT connectivity, environmental monitoring becomes more efficient, accurate, and sustainable.
IoT environmental monitoring relies on individual devices and IoT ecosystems, IoT network, IoT monitoring tool(s) and applications, IoT device monitoring and IoT device management systems, remote monitoring equipment and analytics. The technical complexity of IoT for intelligent IoT monitoring is the requirement for interfacing of a variety of products, systems and protocols. To utilise the scalability and performance benefits of IoT for centralised control and intelligent end monitoring requires the engagement with IoT experts and IoT services as well as engineering the best form of IoT traffic utilization to avoid system failures.

There are four critical components for IoT-based environmental monitoring to support vital insights and decision-making:

1) Observation (Monitor the Environment and Collect Data):

The first step in the environmental monitoring process is to observe and collect data. This involves using sensors or other IoT devices to measure factors such as air quality, temperature, and humidity levels.
These connected IoT devices gather data about the environment and transmit it to a central hub. From here, the data can be reviewed in real-time or used for further analysis off line. Often these systems produce unexpected results and temporal variances. For example, high CO2 levels when offices are highly populated could explain drowsiness or loss of concentration. This can also apply to public spaces such as bars and restaurants where invisible environmental factors may be making the consumer experience uncomfortable.

2) Analysis (Measure Data):

The next step is to analyse the data collected by IoT devices. This includes looking at trends over time, identifying areas of concern, and any correlations between environmental variables, time of day, behaviours and the relationships between indoor and outdoor metrics.
IoT sensing devices pick out key points of the data that indicate everything from chemical and water leaks to air pollution levels. This data analysis can help businesses measure

their environmental footprint and make informed decisions about how to reduce their environmental impact.
For some businesses, this can be relatively benign or related to levels of comfort for workers, whereas others are related to safety. For example, monitoring systems placed in drains can be on the lookout for external pollutants such as diesel, oil, and paints that can stress the environment or harm livestock, fisheries or members of the public.

3) Storage (Catalogue Data):

Once the data has been analysed, it needs to be stored so that it can be accessed in the future. IoT environmental monitoring systems make this easy by storing the data in a secure cloud-based database, allowing businesses to access the data whenever they need it and analyse how their environmental impact is changing over time.
Global databases, such as the Microsoft Planetary Computer, catalogue enormous quantities of environmental data from around the world – although not every cloud database is that large.

4) Action (Provide Actionable Insights From the Data and Analysis):

Finally, businesses need to be able to take action based on the data that has been gathered and analysed.
IoT-enabled environmental monitoring systems can provide insights into how businesses can best reduce their environmental impact, such as by using renewable energy sources or introducing water conservation measures.
These actionable insights may involve changing operational processes, implementing new technologies, or even making changes to their overall business strategy.

There are a number of benefits associated with using an IoT-based environmental monitoring system, including:

- **Improved understanding of the environment via data**: With real-time data feeds being supplied by remotely deployed IoT sensors, businesses and organisations can better understand and quantify the environment. From here, targeted actions can be taken to reduce environmental impact or to spot problems, such as excessive CO2, noise or airborne chemicals as they occur.

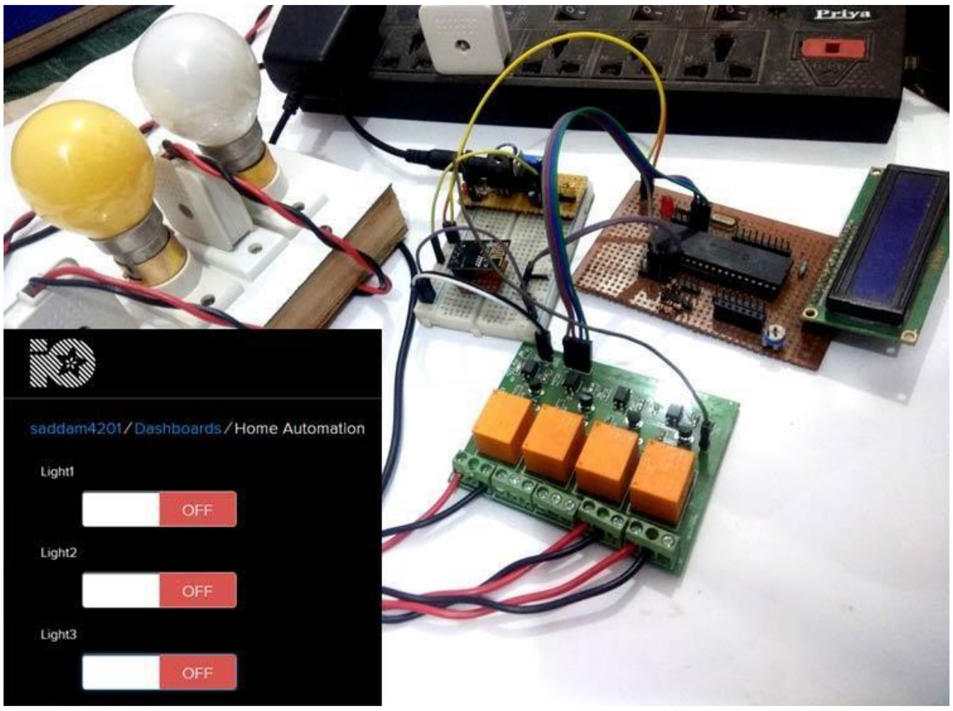
- **Improved efficiency**: With real-time data, organisations can identify and address any problems long before they become more serious. By employing warning alarms, businesses can be more reactive and proactive. This can result in a better working environment, cost savings and less downtime.

- **Increased sustainability:** IoT environmental monitoring systems help organisations identify areas where they can reduce their areas of environmental stress for employees and stakeholders, thus helping them be more sustainable in the long term.

- **Business Growth**: Companies often need to comply with environmental standards in order to assure their customers that they are a progressive organisation whose values chime and adhere to their own policies and direction of travel. Producing evidence-based systems and results can provide greater surety that measures and controls are in place, fitting both the contexts of the business and its (or its customers) environmental concerns.

5.4.1 Home Automation

IoT based Web controlled Home Automation using PIC Microcontroller and Adafruit IO



IoT based Web controlled Home Automation using PIC Microcontroller and Adafruit IO

Home Automation has always been inspiring projects for most of us. Toggling an ACload from the comfort of our chairs or bed of any room without reaching for the switch in another room sounds cool doesn't it!!. And now in the era of IoT, thanks to the ESP8266 module which made it easy to control anything from anywhere in the world.

In this IoT based project, we will use Adafuit IO to control Home appliaces from a webpage using ESP8266 and PIC microcontroller. We have connected three AC light as loads and they can be controlled remotely using either your phone or computer. Here ESP8266 is used with PIC microcontroller.

Component RequiredESP8266
PIC microcontroller(PIC16f877A)
12V 5A Electromagnetic Relay Module -112v Power supply (12V/1A or above) -1 LM7805
Voltage Regulator -1
LM317 Regulator -110k ohm
Resistor -11k
Resistor – 3 10k
Pot – 1
1k Pot -1 16x2 LCD
1000uF capacitor -1
10uF capacitor -2
Wires for connection
18.432 MHz Crystal oscillator -1
LED -2

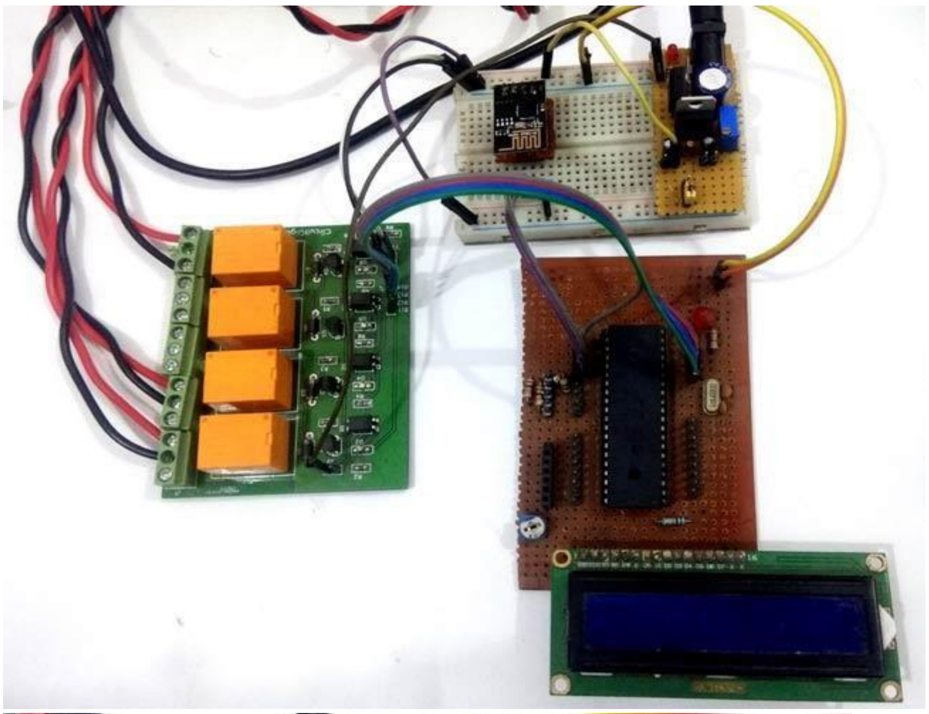
22pF capacitor -2
BreadBoard or PCB (optional)

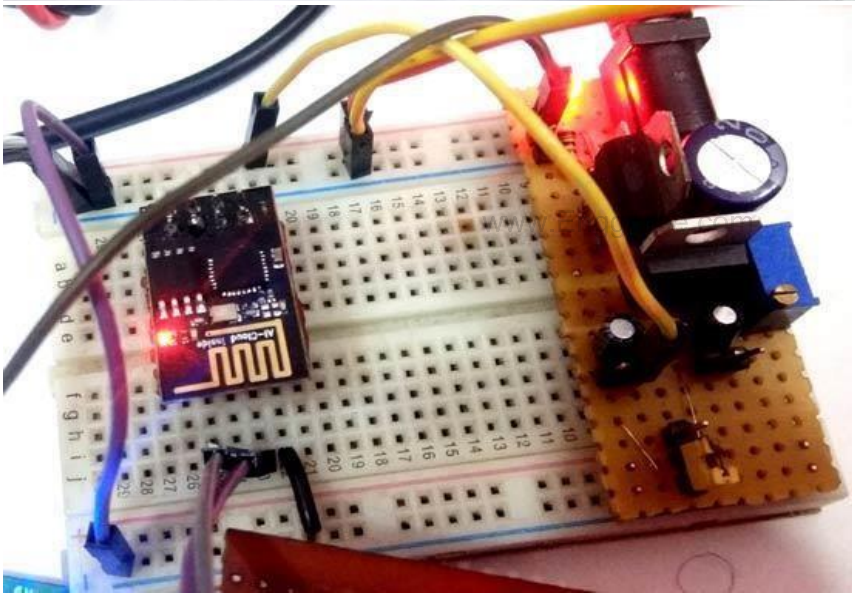
Circuit Diagram
In Web controlled Home Automation project, we have used PIC microcontroller PIC16F877A for performing all the operations. It will communicate with ESP8266 Wi-Fi module to send and receive data from the Adafruit server and take action accordingly to turn ON/OFF relay or load and displaying the status of loads over LCD. We have used 16x2 LCDdisplay for displaying the status of connected AC appliances.
In this project we have three power supplies:
As we have used a 12v relay module we need 12v so we have used a 12v adaptor to power the relay.
We needed 5v for powering the PIC microcontroller, LCD and some of the relay module circuit. So we have used a 7805 voltage regulator connected with a 12v supply. This voltage regulator provides 5v output.
A 3.3v power supply is used for powering the ESP8266 as it works on 3.3v. This supply is made by using LM317 voltage regulator which can be configurable to 3.3v by using some voltage divider circuitry with this. Learn more about creating a LM317 based variable power supply.

Setup Adafruit IO for IoT Home Automation

In this project, we are going to control some home AC appliance via a web page crated using Adafruit IO. Adafruit IO is a simple to use internet service that easily enables IoT devices to GET and POST data. Additionally, it can be used to create GUI interfaces for viewing data, controlling devices, and triggers for alerts/warnings.

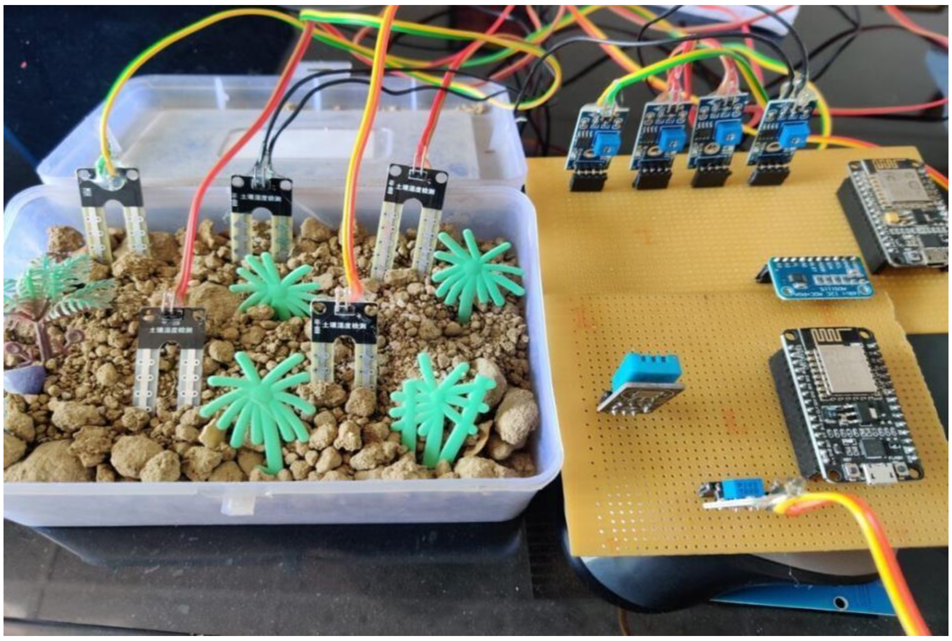
MQTT Protocol
This IoT based Home Automation Project uses MQTT protocol for exchanging data between server and client. This protocol is very fast compared to the TCP/IP protocol. And the working concept is also different from the TCP/IP protocol. This protocol has three maincomponents.
Publish ,Broker ,Subscriber
According to MQTT.org (official website), "MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimize network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging "machine-to-machine" (M2M) or "Internet of Things" world of connected devices, and for mobile applications where bandwidth and battery power are at a premium."





5.4.2 Smart Agriculture

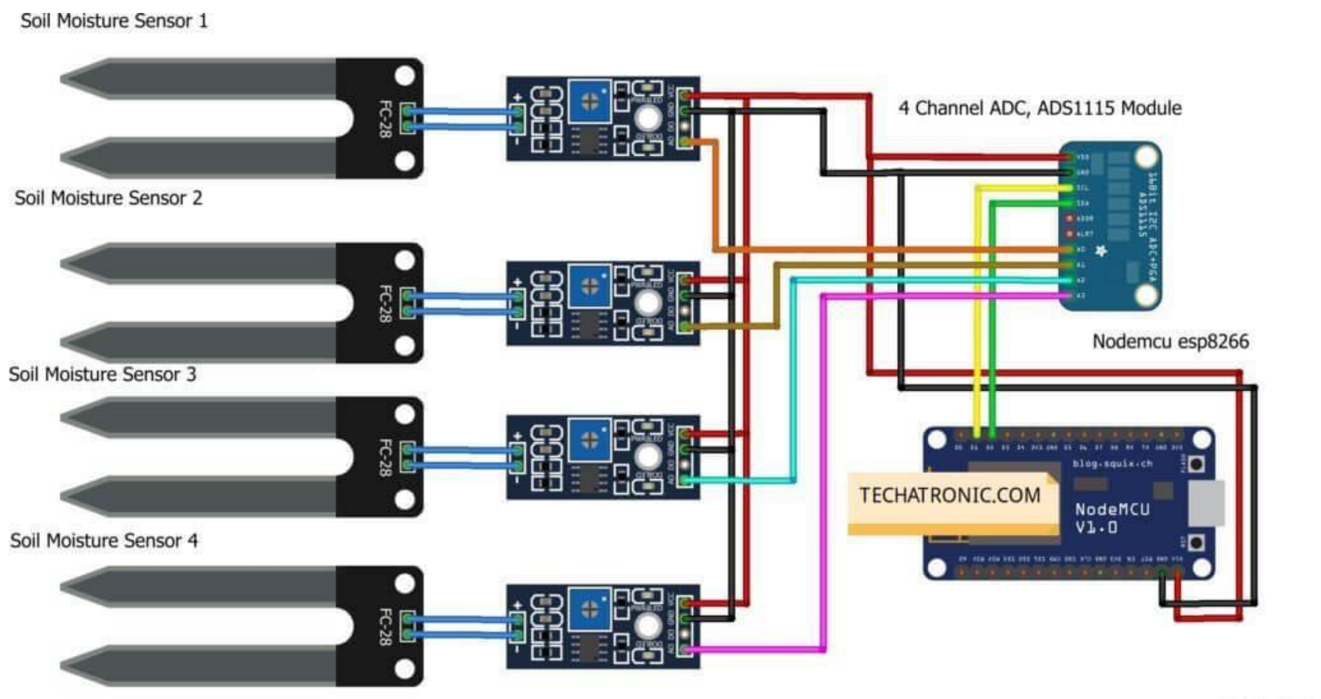
This is smart farming using IoT in this project we will use the server to store the sensor data this is similar to our recent smart agriculture using IoT project.



What is Smart farming using IoT?
Smart farming project is in trend nowadays. everyone wants their farm and land to be smart because it is attractive and techy and also it reduces the manpower they needed to make the system work.

Here, you need a circuit diagram, code, and thingspeak instruction. so we are going to give you all the things below. follow all the steps and make your circuit as we have given.

Here we are going to use two nodemcu.
because we have here 5 analog sensors and 1 DHT.
so we will use 4 soil moisture sensors with the Nodemcu 1 and dht11 and 1 soil moisture sensor with the second node MCU.
and there we need two codes for these two nodemcu.
So, make this project carefully. upload first code with the first nodemcu as we have given below.



Components Required:-

- NodeMcu-2
- General-purpose PCB
- 4 channel ADC multiplexer
- 5 soil moisture sensor
- DHT11
- Breadboard
- Jumper wires

Connection Diagram

Nodemcu esp8266	4 Channel ADC
VV, Vin	VCC
G, GND	GND
D1 Pin	SCL Pin
D2 Pin	SDA Pin

Soil 1 Sensor	Soil 2 Sensor	Soil 3 Sensor	Soil 4 Sensor	4 Channel ADC
VCC	VCC	VCC	VCC	VCC
GND	GND	GND	GND	GND

A0				A0
	A0			A1
		A0		A2
			A0	A3

After making all the connections You need to upload the Code.

Connect all the Soil moisture sensor Vcc to the Nodemcu Vin

Connect Multiplexer Vcc to the Nodemcu Vin

Connect all the Soil moisture sensor Gnd to the Nodemcuo Gnd
Connect Multiplexer Gnd to the Nodemcu Gnd
Soil moisture sensor Output connects to the Multiplexer as given in the above image

Smart agriculture is mostly used to denote the application of IoT solutions in agriculture. So what is smart agriculture using IoT? By using IoT sensors to collect environmental and machine metrics, farmers can make informed decisions, and improve just about every aspect of their work – from livestock to crop farming.
The adoption of IoT solutions for agriculture is constantly growing. COVID-19 has had a positive impact on IoT in the agriculture market share. Disruptions in the supply chain, and the shortage of qualified workers, has propelled its CAGR to 9,9%. In fact, as per recent reports, the smart framing market share is set to reach $6.2 billion by 2021.

There are many types of IoT sensors for agriculture as well as IoT applications in agriculture in general.

Monitoring of climate conditions
Probably the most popular smart agriculture gadgets are weather stations, combining various smart farming sensors. Located across the field, they collect various data from the environment and send it to the cloud.

Greenhouse automation
The use of IoT sensors enables them to get accurate real-time information on greenhouse conditions such as lighting, temperature, soil condition, and humidity.
In addition to sourcing environmental data, weather stations can automatically adjust the conditions to match the given parameters. Specifically, greenhouse automation systems use a similar principle.

Crop management
Just like weather stations, they should be placed in the field to collect data specific to crop farming; from temperature and precipitation to leaf water potential and overall crop health. You can monitor your crop growth and any anomalies to effectively prevent any diseases or infestations that can harm your yield.

Cattle monitoring and management
Just like crop monitoring, there are IoT agriculture sensors that can be attached to the animals on a farm monitoring their health and log performance. Livestock tracking and monitoring help collect data on stock health, well-being, and physical location.
For example, such sensors can identify sick animals so that farmers can separate them from the herd and avoid contamination.

Precision farming
Also known as precision agriculture, precision farming is all about efficiency and making accurate data-driven decisions. It's also one of the most widespread and effective applications of IoT in agriculture.
By using IoT sensors, farmers can collect a vast array of metrics on every facet of the field microclimate and ecosystem: lighting, temperature, soil condition, humidity, CO2 levels, and pest infections. This data enables farmers to estimate optimal amounts of water, fertilizers, and pesticides that their crops need, reduce expenses, and raise better and healthier crops.

Agricultural drones
Perhaps one of the most promising agritech advancements is the use of agricultural drones in smart farming. Also known as UAVs (unmanned aerial vehicles), drones are better equipped than airplanes and satellites to collect agricultural data.

Predictive analytics for smart farming
Precision agriculture and predictive data analytics go hand in hand. While IoT and smart sensor technology are a goldmine for highly relevant real-time data, the use of data analytics helps farmers make sense of it and come up with important predictions: crop harvesting time, the risks of diseases and infestations, yield volume, etc.

Data analytics tools help make farming, which is inherently highly dependent on weather conditions, more manageable, and predictable.

End-to-end farm management systems
A more complex approach to IoT products in agriculture can be represented by the so-called farm productivity management systems. They usually include a number of agriculture IoT devices and sensors, installed on the premises as well as a powerful dashboard with analytical capabilities and in-built accounting/reporting features.
This offers remote farm monitoring capabilities and allows you to streamline most of the business operations.
In addition to the listed IoT agriculture use cases, some prominent opportunities include vehicle tracking (or even automation), storage management, logistics, etc.