## Unit V – IOT Design and System Engineering 6 Hr

**Syllabus:**

Discuss IOT Requirements; Hardware & Software, Study of IOT Sensors; Tagging and Tracking Embedded Products; IOT Design, SIM Card Technology, IOT Connectivity and Management IOT Security & IOT Communication.

### What is IoT and IoT Requirement?

IoT is the extension of Internet connectivity into physical devices and everyday objects. Embedded with electronics, Internet connectivity, and other forms of hardware (such as sensors), these devices can communicate and interact with others over the Internet, and they can be remotely monitored and controlled. Let's break that down:

First, IoT is about **connectivity**. All your things are connected through the internet. Things refer to any physical object that can be uniquely identified (by URI or Unique Resource Identifier) and that can send/receive data by connecting to a network. Examples are buildings, vehicles, smartphones, shampoo bottles, cameras, etc. They can be connected among themselves, with a central server, with a network of servers, with the cloud, or a mix of it all and more.

Second, IoT is about **information and communication**. Everything is sharing information to their designated endpoints either other things or servers. They are constantly sending information about status, actions, sensor data, and more. All of them with their unique ID attached, so that it is possible to know where the data came from.

And finally, IoT is about action and **interaction**. These last two concepts define the core of what IoT is: connection and information sharing. However, all that data isn't generated just to be stored somewhere and forgotten. It has to be used for something. And that use can be automation: computers using the data to automatically (or even autonomously) make decisions and, for example, with the help of Machine learning, act. And that usage can also be monitoring: letting people know the state of something or some process. The people may be the users of a product or the overseers of for example a production line.

There are two types of IoT: CIoT and IIoT. The difference is that CIoT often focuses on convenience for the "Customer", whereas IIoT is strongly focused on the "Industry" and is more system-centric. Its focus is on improving the efficiency, security, and output of operations with a focus on Return on Investment (ROI).

## Requirements

Industrial IoT requirements can be specified as follows:

- **Cloud computing**: Cloud computing enables storage and processing of unstructured and structured data into real-time information.
- **Access**: Another IIoT requirement is its accessibility from anywhere and anytime.
- **Security**: Security is an important factor that forms a part of IIoT requirements since confidential and sensitive information is exchanged across the businesses.
- **User experience**: The more seamless the User Experience (UX), the greater the use of IIoT systems.

- **Smart machines**: Smart machines form the basic components or the starting point from which all connected things can be derived.
- **Asset management**: Managing assets through cloud-based services ease the functioning and maintenance of IIoT systems.
- **Big Data analytics**: Analysis of big data provides intelligent information, an ideal requirement for industrial purposes.

**IoT Hardware and Software**

Hardware and software devices combine to form an Iot ecosystem. Hardware is a set of devices that wire together to serve some functionality. A bread board is one such example. The bread board usually contains components such as sensors, microcontrollers, microprocessors, resistors, transistors and voltage regulators. In this article, we will look into the hardware components that IoT devices use and we will study further about these components.

Assume that you want to build a drone, a flying device. You want to attach sensors to this drone so that it can take photos of your agricultural crops to keep a track of their growth. Or maybe you want to build a smart watch. A watch that keeps a track of your entire schedule, count the number of steps you take daily, measure your heart pulse. This will require you to connect small components, track the battery usage.

You may wonder how we construct these IoT devices? The answer is IoT devices are a combination of hardware and software. The two components integrate together and perform a variety of functions.

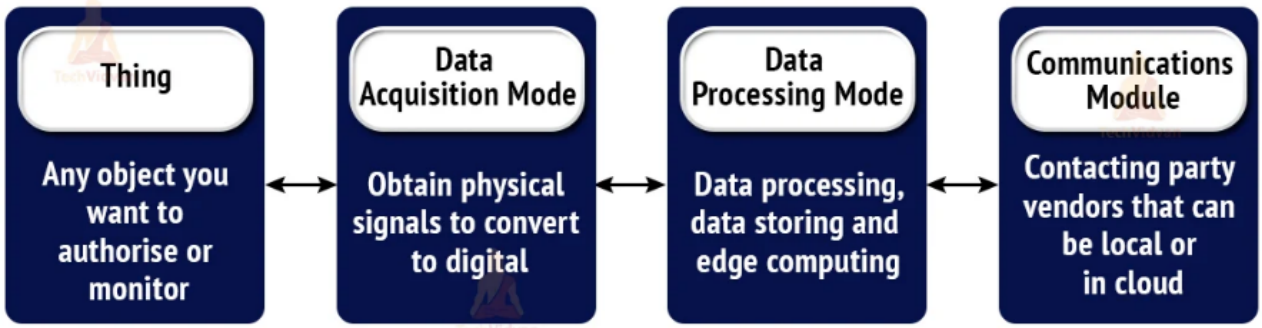
**Building blocks of IoT Hardware**

1. "**Things**": Things in IoT are any devices that are capable of connecting to the internet. They can transmit, retrieve and store huge amounts of data that they collect from the surrounding. They include home appliances such as geysers, microwaves, thermostats and refrigerators

2. **Data Acquisition module**: As the term suggests, this module is responsible for acquiring data from the physical surroundings or environment. These could include changes in the temperature, movement, humidity and pressure. This module also includes the necessary hardware to convert the incoming sensor signal into digital information for the computer to use it. This includes conditioning of incoming signal, removing noise, analog-to-digital conversion, interpretation, and scaling.

3. **Data processing module**: This module includes computers that process the data acquired from the previous module. They analyze the data, store data for future references and other purposes.

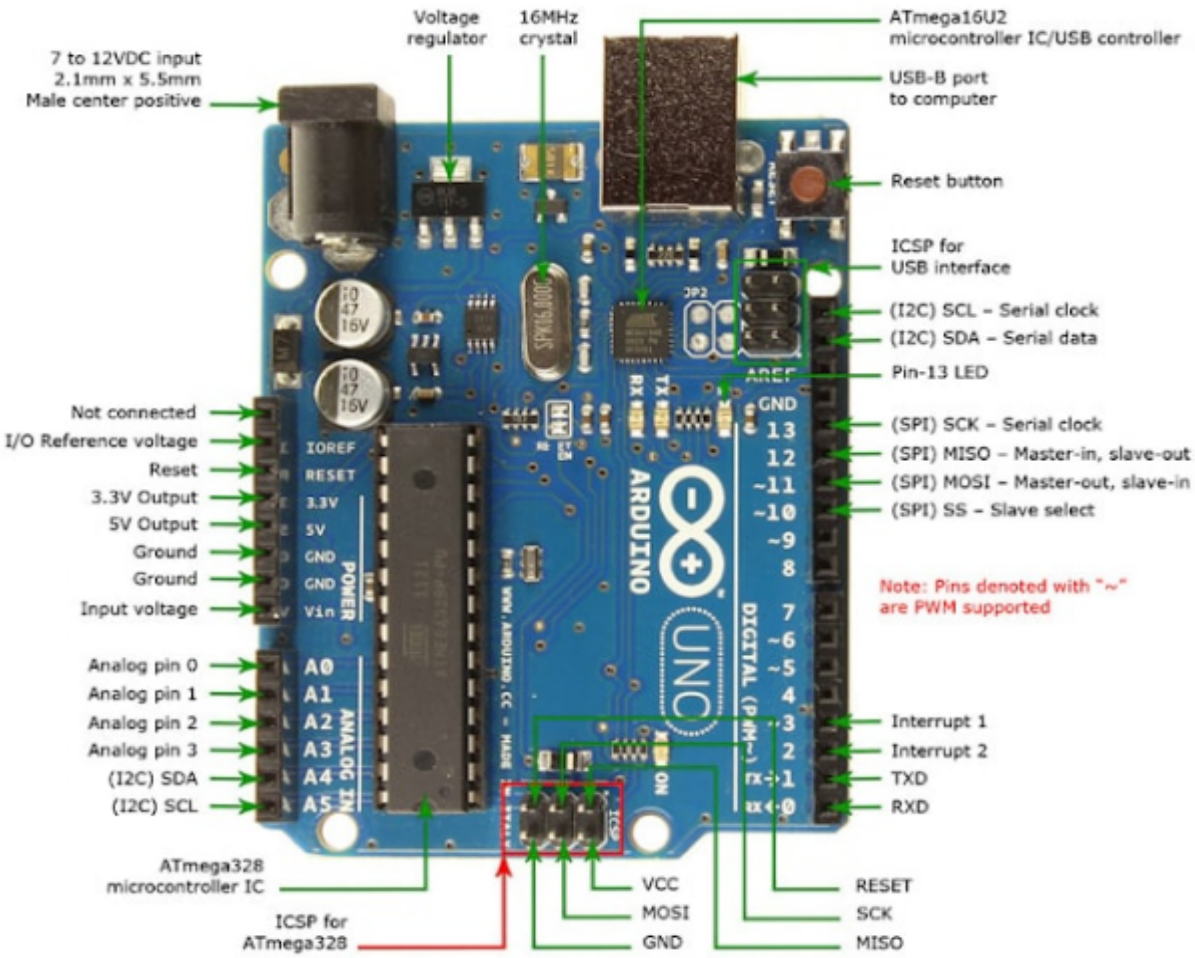
4. **Communication module**: This is the final building block and this module is responsible for communication with third party vendors. This could include device to device, device to server or device to user.




IoT Hardware includes a wide range of devices such as devices for routing, bridges, sensors etc. These IoT devices manage key tasks and functions such as system activation, security, action specifications, communication, and detection of support-specific goals and actions.

**IoT Hardware components**

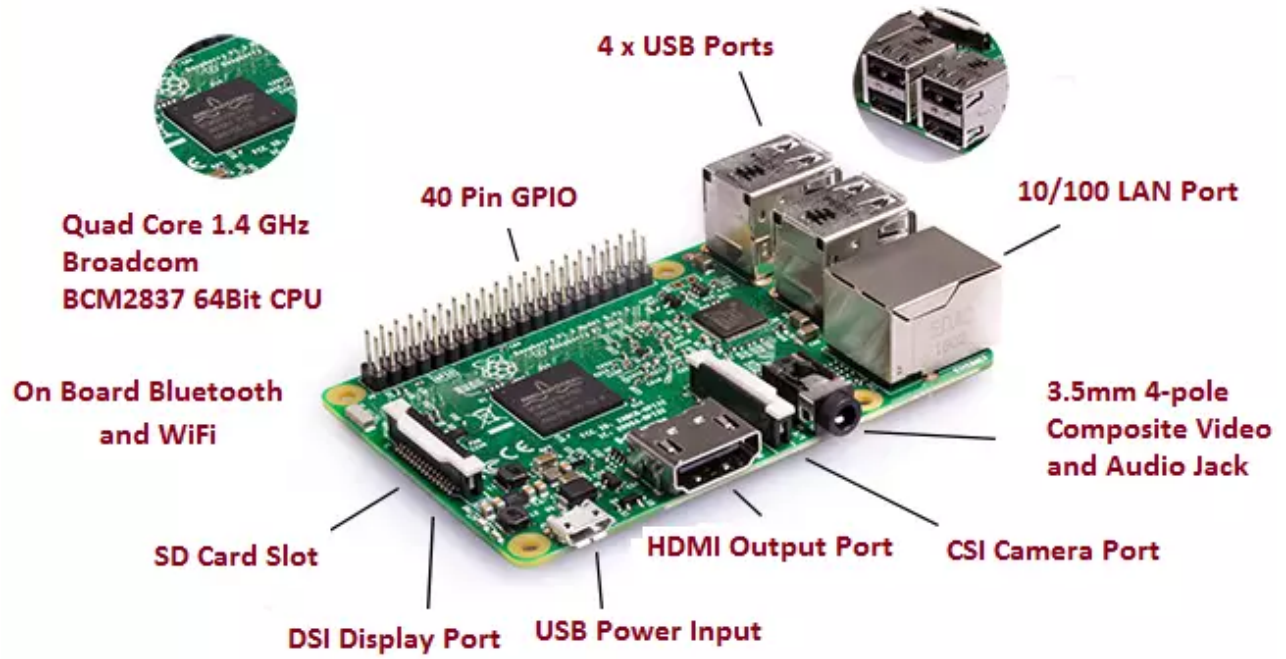
IoT Hardware components can vary from low-power boards; single-board processors like the Arduino Uno which are basically smaller boards that are plugged into mainboards to improve and increase its functionality by bringing out specific functions or features (such as GPS, light and heat sensors, or interactive displays).




**Pins functions:**

1. Serial / UART: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial chip.
2. External interrupts: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
3. PWM (pulse-width modulation): pins 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the analogWrite() function.
4. SPI (Serial Peripheral Interface): pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK). These pins support SPI communication using the SPI library.
5. TWI (two-wire interface) / I²C: pin SDA (A4) and pin SCL (A5). Support TWI communication using the Wire library.
6. AREF (analog reference): Reference voltage for the analog inputs.

Another well-known IoT platform is **Raspberry Pi 2/3/4**, which is a very affordable and tiny computer that can incorporate an entire web server. Often called “RasPi,” it has enough processing power and memory to run Windows 10 on it as well as IoT Core.
RasPi exhibits great processing capabilities, especially when using the Python programming language.




**Raspberry Pi 4 specifications include:**

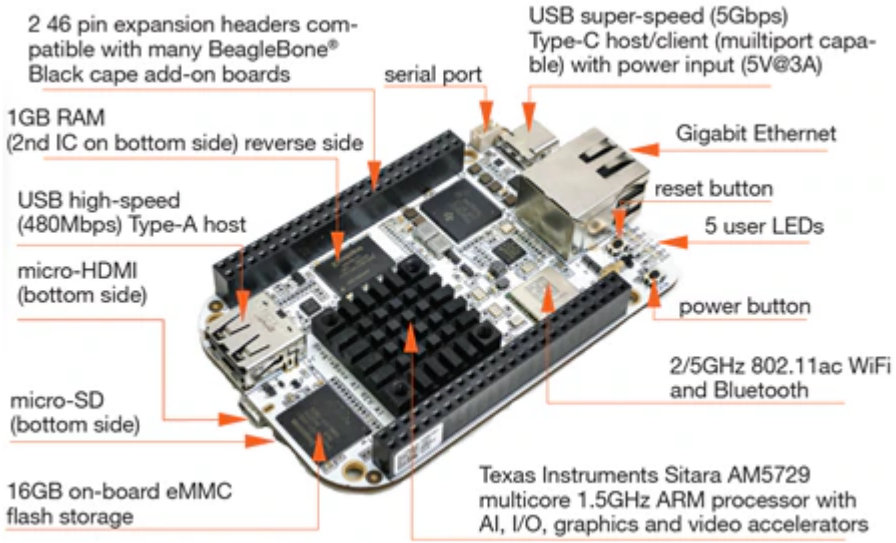
1. It has a 64bit quad-core processor having cortex-A72 (ARM v8) clocked @1.5GHz.
2. The new Pi board includes Broadcom BCM2711 VC6 GPU able to handle two 4kp30 displays also it can handle H.265 decoding at 4kp60. It has two micro HDMI ports.
3. Now the new Pi board comes in 2GB, 4GB, and 8GB LPDDR-4 RAM options.
4. It has a dual-band 2.4/5.0 GHz Wi-Fi, Bluetooth 5.0, Gigabit Ethernet Port, 2 USB 3.0, and 2 USB 2.0 ports.
5. USB type C power input port. Also, it has POE capability via separate POE HAT (add-on).

6. It has a standard 40 pin GPIO header (having backward compatibility)

The new Raspberry Pi is even more fast, powerful and versatile for a huge variety of projects involving robotics, automation, image processing, AI, and many more.

**BeagleBoard** is a single-board computer with a Linux-based OS that uses an ARM processor, capable of more powerful processing than RasPi.

Tech giant Intel's Galileo and Edison boards are other options, both great for larger scale production, and Qualcomm has manufactured an array of enterprise-level IoT technology for cars and cameras to healthcare.




**IoT Software**

The software and the programming languages on which IoT works uses very common programming languages that programmers use and already know. So which language should be chosen?

Firstly, because embedded systems have less storage and processing power, their language needs are different. The most commonly used operating systems for such embedded systems are Linux or UNIX-like OSs like Ubuntu Core or **Android**.

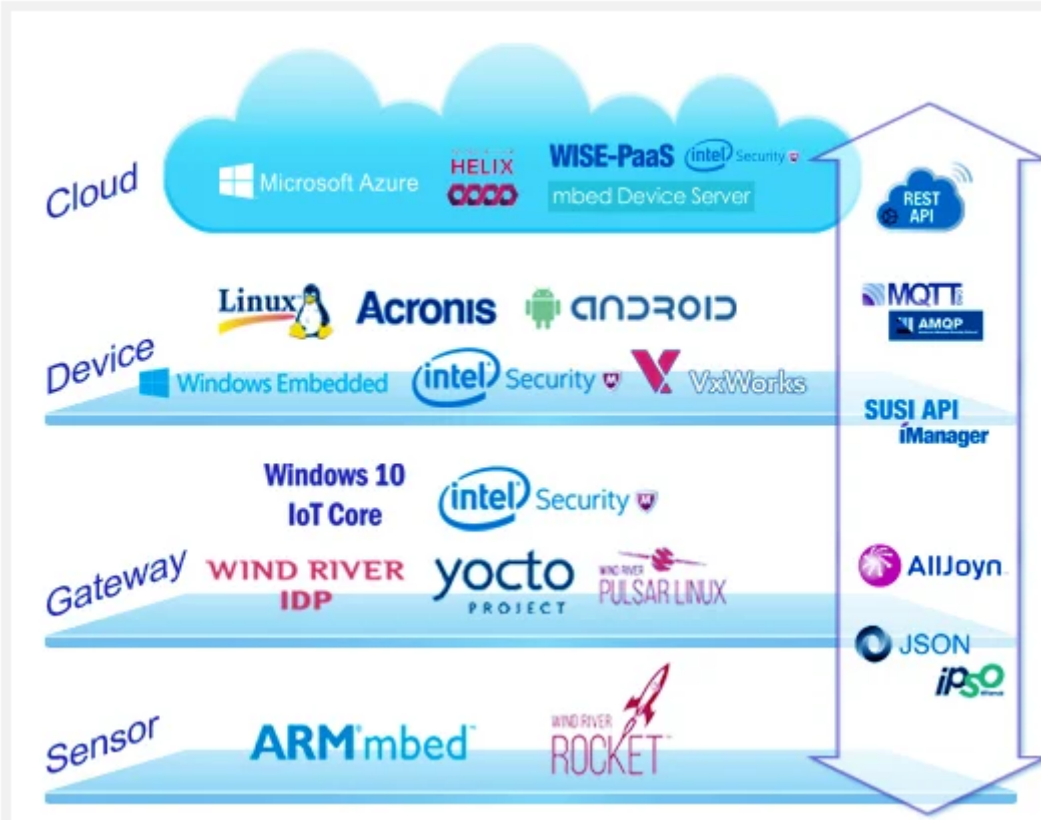
IoT software encompasses a wide range of software and programming languages from general-purpose languages like C++ and Java to embedded-specific choices like Google's Go language or Parasail.

Here's a quick overview of each one of IoT Software-

- **C & C++:** The C programming language has its roots in embedded systems—it even got its start for programming telephone switches. It's pretty ubiquitous, that is, it can be used almost everywhere and many programmers already know it. C++ is the object-oriented version of C, which is a language popular for both the Linux OS and Arduino embedded IoT software systems. These languages were basically written for the hardware systems which makes them so easy to use.
- **Java:** While C and C++ are hardware specific, the code in JAVA is more portable. It is more like a write once and read anywhere language, where you install libraries, invests time in writing codes once and you are good to go.
- **Python:** There has been a recent surge in the number of python users and has now become one of the "go-to" languages in Web development. Its use is slowly spreading to the embedded control and IoT world—specially the Raspberry Pi processor. Python

is an interpreted language, which is, easy to read, quick to learn and quick to write. Also, it's a powerhouse for serving data-heavy applications.

- **B#:** Unlike most of the languages mentioned so far, B# was specifically designed for embedded systems, it's small and compact and has less memory size.
- **Data Collection:** It is used for data filtering, data security, sensing, and measurement. The protocols aid in decision making by sensing form real-time objects. It can work both ways by collecting data from devices or distributing data to devices. All the data transmits to a central server.
- **Device Integration:** This software ensures that devices bind and connect to networks facilitating information sharing. A stable cooperation and communication ensure between multiple devices.
- **Real-Time Analytics:** In this, the input from users serves as potential data for carrying out real-time analysis, making insights, suggesting recommendations to solve an organizations problems and improve its approach. This, as a result, allows automation and increased productivity.
- **Application and Process Extension:** These applications extend the reach of existing systems and software to allow a wider, more effective system. They integrate predefined devices for specific purposes such as allowing certain mobile devices or engineering instruments access. It supports improved productivity and more accurate data collection.




**IoT hardware requirements for deploying your IoT project**

IoT devices operate only within some set environs, and their hardware projects differ widely; hence, they are highly specialized. Nevertheless, it is possible to develop and design your custom PCBs and their components custom-made for the requirements of your IoT solution by prototyping with the generic off-the-shelf hardware.

When deploying your IoT project, it is essential to consider the below IoT hardware requirements:

1. **Security requirements**: Security is an essential component of the Internet of Things. Considering the device's security requirements is imperative at all the development and designing stages. Even when prototyping, make sure the security and integrity of data captured by any device remain intact. All IoT devices, their network, service applications of websites, and mobiles apply the security requirements.
2. **Ease of development**:Ease of development is a requirement of high priority when prototyping. It enables the user to get the IoT device up and running quickly and efficiently when capturing data and interconnecting with other devices and the cloud. When deploying your IoT projects, have in mind the API documentation's quality, accessibility, and availability. Also, consider the tools of development, and support provided by the device's manufacturer or by the development team.
3. **Data acquisition, processing, and storage requirements**:The number of sensors connected to the captured data's resolution and the sampling rate are the main determinants of the data's volume to be processed. They also influence the requirements of storing and processing data.
4. **Connectivity requirements**: Wireless networking has connectivity requirements such as operating range, distance covered by the transmitting signal, and the predicted data and transmitted volume. When checking on the device's connectivity requirements, it is vital to contemplate the fault tolerance, the device's reconnecting capability, and how long a device takes when retrying to send data after it disconnects.
5. **Power requirements**: The power requirements are impacted mainly by the network transmission rate and the number of sensors in the device. Therefore, when deploying your IoT project, it is essential to consider if the device needs a mobile power source such as a super capacitor or a battery or hardwired for power. Also, know the battery's size, capacity requirements, weight, and if the battery is recharged, replaced, or discarded when it dies. In case the battery is rechargeable, check by what means and how often it is charged?
6. **Physical device design requirements**: They include the size and physical appearance of the hardware device. When designing an IoT device, it is essential to consider the ecological situations in which the device will be installed. For instance, consider if the device will require a ruggedized or a waterproof? All appliances installed on a truck's underside as part of a fleet monitoring application should always be safeguarded to ensure that it works well, even when under harsh conditions. The device has to be water-resistant and impervious to shock, dirt, and vibration.
7. **Cost requirements**: The original hardware's outlay and allied components like sensors are the main determinants of the hardware's price. Other components that determine the hardware cost include the ongoing operating cost like the maintenance and power cost. Also, it is essential to think through the reasonable licensing fees for some device's drives and components. Assembling custom boards is more expensive than purchasing commercially accessible off-the-shelf

development boards. It is a wiser alternative to consecrate hardware devices when scaling out in the IoT network with numerous instruments.

8. **Processors**: Data is processed once the sensor data capture it before conveying the outcomes to the cloud. Thus, the quantity of data processing needed to create the subsequent sensor data and the complexity of sensors determines the processing level. For example, the temperature reading is a simple illustration of an average of set values or a single data value over time. Moreover, a security camera unable to record digital video without the scene detection algorithm flagging an event can be more complex.

Based on the complexity and power needed to process data, four IoT hardware processing classes are required. They are;

1. **PC Based Systems**: The PC-based systems are configurable platforms that allow easy creation of custom systems by system integrators from cheap, typical processors, off-the-shelf motherboards, cases, and power supplies. Extensive local data storage capabilities are provided mainly by Solid-state drives (SSDs) or terabyte hard drives.
2. **Mobile Systems**: Mobile systems incorporate embedded systems that have a specialized subset optimized for smartphones and tablets. All mobile systems require frequent charging as the devices are battery-powered. These intrinsically personal devices have advanced power management system capabilities to conserve energy and extend the device's battery life. Also, mobile systems offer processing capabilities of high-performance.
3. **Microprocessor (MPU) Based Embedded Systems**: They offer an inclusive options range of capability and performance elevated to address requirements for specific products. The requirements are primarily for communication systems, consumer electronics, automotive and industrial controls, medical devices, and other vertical market applications.
4. **Microcontroller (MCU) Based Embedded Systems:** These systems require minimal processing necessities, and they offer solutions of low-cost. Nevertheless, microcontrollers are advanced implant-specific hardware modules to speed up the processing of images and security roles like cryptographic acceleration for exchanging public/private keys and True Random Number Generation (TRNG).

# Sensors in Internet of Things(IoT)

A Sensor is an electronic device that is used to measure some sort of physical parameters (e.g. temperature, pressure, light intensity, etc). The output of an electronic sensor is an electrical signal that is either analog or digital. Processing the sensor's output can be done in hardware (using discrete electronic elements) or in software (using some sort of microcontrollers or MPUs).

Each sensor has a different working principle depending on the physical construction and the physical parameter it's actually measuring. The common thing between all sensors is they all convert a physical parameter (such as temperature) to an electric signal. But each one has a specific transfer function (for analog) or a specific communication bus as SPI, UART, etc (for digital). These specific details are demonstrated completely in the datasheet for each sensor with the typical connection diagram and how to interface it.

**Sensors Classification**

There are, in fact, many classifications for sensors. We can classify sensors depending on the type of output signal or the physical parameters they measure and other considerations could be taken resulting in a variety of ways to classify sensors. However, in this article, I'll mention a couple of ways to classify sensors. First of which is the type of output signal and the second one is the physical parameter they measure.

1 **Sensor's Output Signals Classification**

| Analog Output | Digital Output |
|---|---|
| The output of these sensors is an analog voltage that you can measure then determine the desired physical parameter using the sensor's transfer function. It may also be capacitive or resistive or anything analog. | The output of these sensors is digital data that you can read via serial or parallel communication buses (as UART, SPI, I2C, etc). The typical format for the data is demonstrated exactly in the sensor's datasheet. |
| Example: The temperature sensor (specifically LM35) is an analog sensor whose output | Example: The accelerometer sensor (ADXL345) is a digital sensor that sends out its output data via the I2C two-wire bus. |

**2. Sensor's Physical Parameter Classification**

There are sensors to measure everything you can possibly think of and here is a table for the most common ones.

| | | | |
|---|---|---|---|
| Temperature Sensors | Chemical Sensors | Proximity Sensors | Touch Sensors |
| Light Sensors | Tilt Sensor | Metal Detectors | Cameras |

| | | | |
|---|---|---|---|
| Humidity Sensor | Vibration Sensor | Magnetic Sensor | Color Sensor |
| Current Sensor | Pressure Sensor | Fingerprint Sensor | GPS |
| Motor Speed Sensor | Bending Sensor | PIR Sensor | Position Sensor |
| Lidar Sensor | Ultrasonic Sensor | Gyroscope Sensor | Accelerometer Sensor |
| Digital Compass Sensor | Sound Sensor (Mic.) | IR Sensor | Odometer Sensor |

**Applications For Sensors**

Sensors have been around since the early days of electricity and have been in use in a very wide range of applications. We use sensors in electronics projects, robotics, industry, and much more. Down below is a brief list of typical applications of sensors.

1. Robotics
2. Embedded Systems
3. Computers
4. Smart Cars
5. Avionics
6. Satellites
7. Smart Homes
8. Smartphones
9. Smart Watches
10. Energy plants
11. Remote Sensing
12. Communications
13. etc.

Almost in all embedded systems and electronic devices, there will be at least a couple of sensors providing some sort of feedback for a physical property like temperature, pressure, etc. The list goes on, and it's not in the scope of this article to create an exhaustive full list for all possible applications of sensors.

**How To Choose The Right Sensor?**

There are many factors to consider while choosing a sensor for your project. But all starts by selecting the physical parameter you're willing to measure. Then it's the time to consider some other factors to get the best sensors for best results and within the given constraints such as budget, accuracy, etc. Down below are some of the most important factors to consider.

1. Range of Operation: The most important factor to consider in a sensor is the operating range. If you're designing a boiler system that will control some liquid boiling at 500°c, You shouldn't use a small LM35 sensor that can only read up to 150°c. You should make sure that the sensor meets the range requirement of your application in order to get the right sensor for it.
2. Accuracy (Resolution): Decide on the required resolution (accuracy) of the sensor your applications need prior to choosing a sensor. For example, a temperature sensor with an accuracy of 1°c will be sufficient for a boiler embedded system's design. However, the same sensor with the same accuracy may not be sufficient for some critical scientific experiments or devices that require an accuracy of 0.1°c. So, there is a trade-off and you have to make your own decision based on your system's specifications.
3. Total Cost: Electronic sensors range widely in price. You can easily guess that high accuracy sensors are always way more expensive than low accuracy ones. The operating wide dynamic rage also plays a role in determining the price point of the sensor, etc. The point is you have to make sure that you choose the sensor that gives you the best results within the allowed budget for the projects. Yes, you may drop out the resolution and still get a decent project with a small error in the output but at least it's working! instead of burning all the budget for a high-end high-tech sensor with no money left for other parts. That's the point of it, and again you will have to decide on these trade-offs given the exact situation and application specifications for your project.
4. Interfacing Method: As we've stated earlier, some sensors are analog and others are digital. Hence, there are different ways to interface and read these sensors using analog input pins of an MCU. Or connect it on a serial bus like UART, SPI, or I2C. You should also decide on the type of interface that your application can handle much more smoothly without problems or running out of serial ports.
5. Data Rate (For Digital Sensors): Digital sensors can send you readings (data) at a rate we call the sampling rate. Typically sensors' rate is defined by ksp/s (kilo samples per second) which is a thousand sample points (readings) in a second. Some sensors can supply up to a few Msp/s. Most of the time, it's a programmable feature in sensor modules. And you have to check whether this rate of data supplies your MCU with the information it needs to run the algorithm or perform the required calculations flawlessly.
6. Documentation: Good documentation is key whether to choose a sensor or not. Of course, you don't want to get a sensor that has only a couple of Chinese papers describing nothing useful on how to use this crappy sensor even if it's very cheap!

## Classification of Sensors?

We can classify sensors into three categories according to the:

- Type of input required to get desired output from a sensor.
- Working of the sensing mechanism.
- Type of output.

### Active or Passive Sensors

*Active sensors* are a type of sensor that requires continuous input to get the required output from the sensing element. RADAR, Chip-based Humidity and Temperature Sensors, Gas Sensor, GPS, accelerometer are examples of Active Sensors. These Sensors require continuous electric power for Operation.

*Passive sensors* are the type of sensor that does not require any external input such as a power signal to get the required output. Examples of passive sensors are metal detectors, magnetic field detectors, photodiode, thermistors, and strain gauges. They require electric power only to respond or give output results.

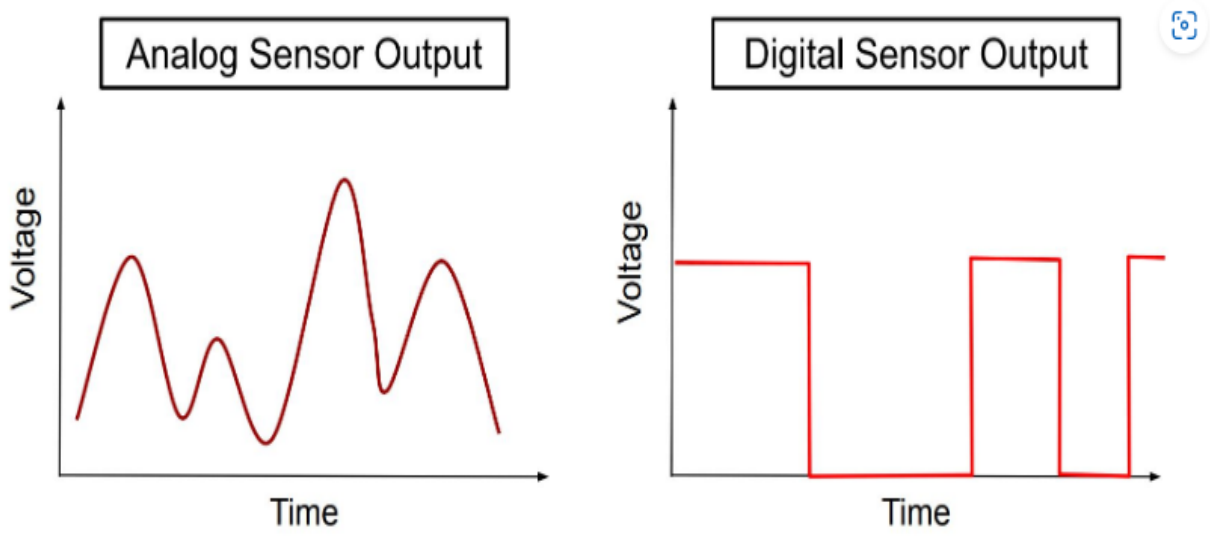
### Electrical, chemical or mechanical Sensors

*Electrical Sensors* are the sensors in which a sensing element detects a physical parameter and converts it into an electrical signal. This electrical signal is used to determine the value of that physical parameter. Most of the sensors used in automation are electrical sensors. NTC, humidity, and capacitive sensors are examples of electrical sensors.

*Chemical sensors* are the type of sensors that respond to the chemical reaction. For example, the water PH chemical sensor shows different colors according to the PH level.

*Mechanical Sensors* utilize mechanisms to detect a physical parameter. For example, a strain gauge is a mechanical sensor that detects mechanical deformation and converts this deformation into an electrical signal. Examples of mechanical sensors are touch sensors, biopic, and stress gauges. Click this link to know the benefits of implementing automation in manufacturing.

Analog or Digital Sensors




Analog vs Digital Sensor Output

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

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

## Different Types of Sensors

- Temperature Sensors: Temperature sensors monitor the temperature of the air or a physical item and convert it to an electrical signal that may be calibrated to represent the observed temperature precisely. These sensors might be used to track the temperature of a crucial piece of equipment to detect when it is overheating or approaching failure.
- Pressure Sensors: Pressure sensors detect air pressure, the pressure of a stored gas or liquid in a sealed system such as a tank or pressure vessel, or the weight of an item by measuring the pressure or force per unit area applied to the sensor.
- Motion SensorsPIR Hardware Motion Detector: Motion sensors or detectors can sense the movement of a physical item by employing any one of numerous technologies, including passive infrared (PIR), microwave detection, or ultrasonic, which utilizes sound to detect things. These sensors may be utilized in security and intrusion detection systems, as well as to automate the control of doors, sinks, air conditioning, and heating systems, and other systems.
- Level Sensors: The level of a liquid relative to a normal value is converted into a signal using level sensors. Gasoline gauges, for example, show the level of fuel in a vehicle's tank and offer a continuous level reading. There are also point-level sensors, which are a digital or go-no-go depiction of the liquid level. When the gasoline level tank gets extremely close to empty, certain vehicles include a light that glows, functioning as an alert to notify the driver that fuel is likely to run out totally.
- Image Sensors: Image sensors collect pictures that are then digitally stored and processed. License plate readers, as well as facial recognition systems, are examples. Image sensors in automated production lines may identify quality concerns such as how effectively a surface is coated after leaving the spray booth.
- Proximity Sensors: Proximity sensors use a range of technological designs to detect the presence or absence of items approaching the sensor. These strategies include: 1. Inductive technology that can be used to detect metal items
  2. Capacitive technologies are those that work with things that have a different dielectric constant than air.
  3. Photoelectric technologies, which use a beam of light to illuminate and reflect back from an item, or photovoltaic technologies, which use a beam of light to illuminate and reflect back from an object.
  4. Ultrasonic technologies detect an item approaching the sensor by sending out a sound signal.
- Water Quality Sensors: The importance of water to humans on the planet, not just for drinking but also as a critical element in many manufacturing processes, necessitates the ability to feel and evaluate characteristics related to water quality. The following are some instances of what is felt and monitored:
  1. Chemical Presence – such as chlorine levels or fluoride levels.
  2. Oxygen Levels – which may impact the growth of algae and bacteria.
  3. Electrical Conductivity – which can indicate the level of ions present in water.
  4. PH Level – a reflection of the relative acidity or alkalinity of the water.
  5. Turbidity Levels – a measurement of the number of suspended solids in water.

- Chemical Sensors: Chemical sensors are designed to detect the presence of certain chemical compounds which may have mistakenly escaped from their containers into places that are occupied by workers and are important in managing industrial process conditions.
- Gas Sensors: Gas sensors, like chemical sensors, are calibrated to detect the presence of combustible, poisonous, or flammable gas in the sensor's surroundings. The following are some examples of particular gases that can be detected: Acetone (e.g. Paints And Glues), Toluene (e.g. Furniture), Ethanol (e.g. Perfume, Cleaning Fluids), Hydrogen Sulfide (e.g. Decaying Food), Benzene (e.g. Cigarette Smoke)
- Smoke Sensors: Smoke sensors or detectors use optical sensors or ionization detection to detect the presence of smoke conditions that might be a sign of a fire.
- Infrared (IR) Sensors: Objects generate infrared radiation, which is detected by infrared sensor technology. These sorts of sensors are used in non-contact thermometers to measure the temperature of an item without having to place a probe or sensor on it directly. They're useful for assessing electronics' heat signatures and monitoring blood flow or blood pressure in patients.
- Acceleration Sensors: While motion sensors detect movement, acceleration sensors, commonly known as accelerometers, measure the rate at which an object's velocity changes. A free-fall state, a quick vibration creating a movement with speed variations, or rotating motion might all cause this shift. Acceleration sensors use a variety of technologies, one of which is:
  1. Hall-Effect Sensors – rely on magnetic field variations to detect changes.
  2. Capacitive Sensors – which depend on monitoring changes in voltage from two surfaces.
  3. Piezoelectric Sensors – create a voltage that varies in response to pressure due to sensor distortion.
- Gyroscopic sensors: Using a 3-axis system, gyroscopes or gyroscopic sensors are used to monitor the rotation of an object and estimate the rate of its movement, known as angular velocity. These sensors allow the orientation of an item to be determined without having to see it.
- Humidity Sensors:Humidity sensors can detect the relative humidity of air or other gases, which is a measure of how much water vapor is present. Controlling environmental conditions is crucial in the manufacturing of materials, and humidity sensors allow for measurements and adjustments to be made to minimize rising or falling levels. To maintain desired comfort levels, HVAC systems are a typical application.
- Optical Sensors: Optical sensors respond to light that is reflected off of an object and generate a corresponding electrical signal for use in detecting or measuring a condition. These sensors work by either sensing the interruption of a beam of light or its reflection caused by the presence of the object. The types of optical sensors include:
  1. Through-Beam Sensors – which detect objects by the interruption of a light beam as the object crosses the path between a transmitter and remote receiver.

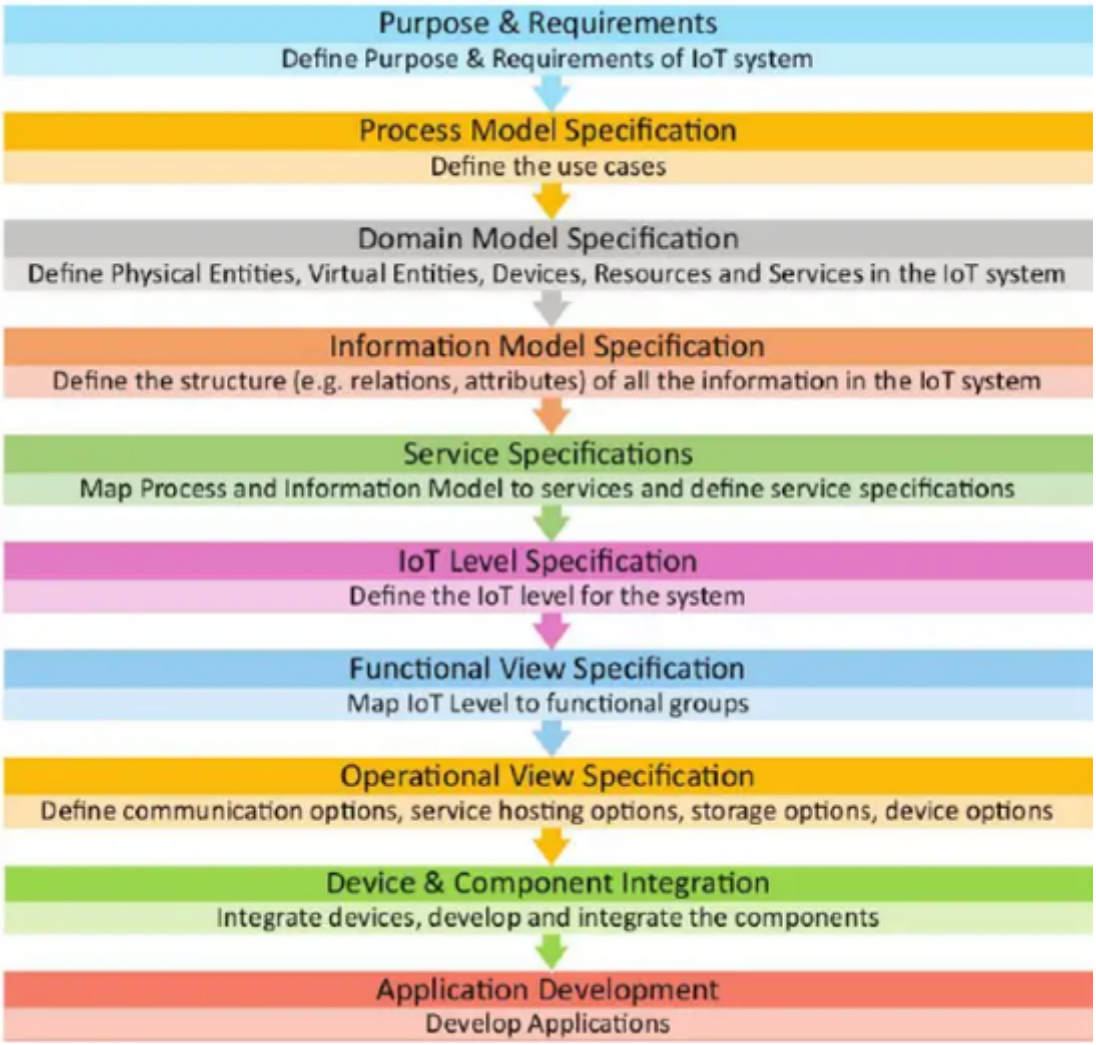
2. Retro-Reflective Sensors – which combine transmitter and receiver into a single unit and use a separate reflective surface to bounce the light back to the device.
3. Diffuse Reflection Sensors – which operate similarly to retro-reflective sensors except that the object being detected serves as the reflective surface.

- AKCP Wired and Wireless Sensors: Wired and Wireless sensors for monitoring a wide range of industries. Data Center Temperature Monitoring, Remote Site Sensors. Power Monitoring and Environmental Monitoring. AKCP has over 30 years of experience and is the world's largest installed base of environmental monitoring sensors.

  AKCP offers powerful yet power-conscious sensors for temperature, humidity, power, contact, and more. They're configurable to transmit infrequently and operate with 10-year battery life. This means minimal maintenance, a network you can deploy and depend on. Our IoT sensors also integrate seamlessly with our Wireless Tunnel Gateways (WTG) pushing data into the AKCPro Server so you can easily connect your data to a variety of devices and applications.

# Introduction to IoT Design

IoT systems designed with this methodology will have reduced design time, testing time, maintenance time, complexity and better interoperability.
The steps involved in the designing of an IoT system or application can be summarized as shown in the below figure:




## Purpose and Requirements Specification

First step is to define the purpose and requirements of the system. In this step, the system purpose, behavior and requirements are captured. Requirements can be:

- Data collection requirements
- Data analysis requirements
- System management requirements
- Security requirements
- User interface requirements

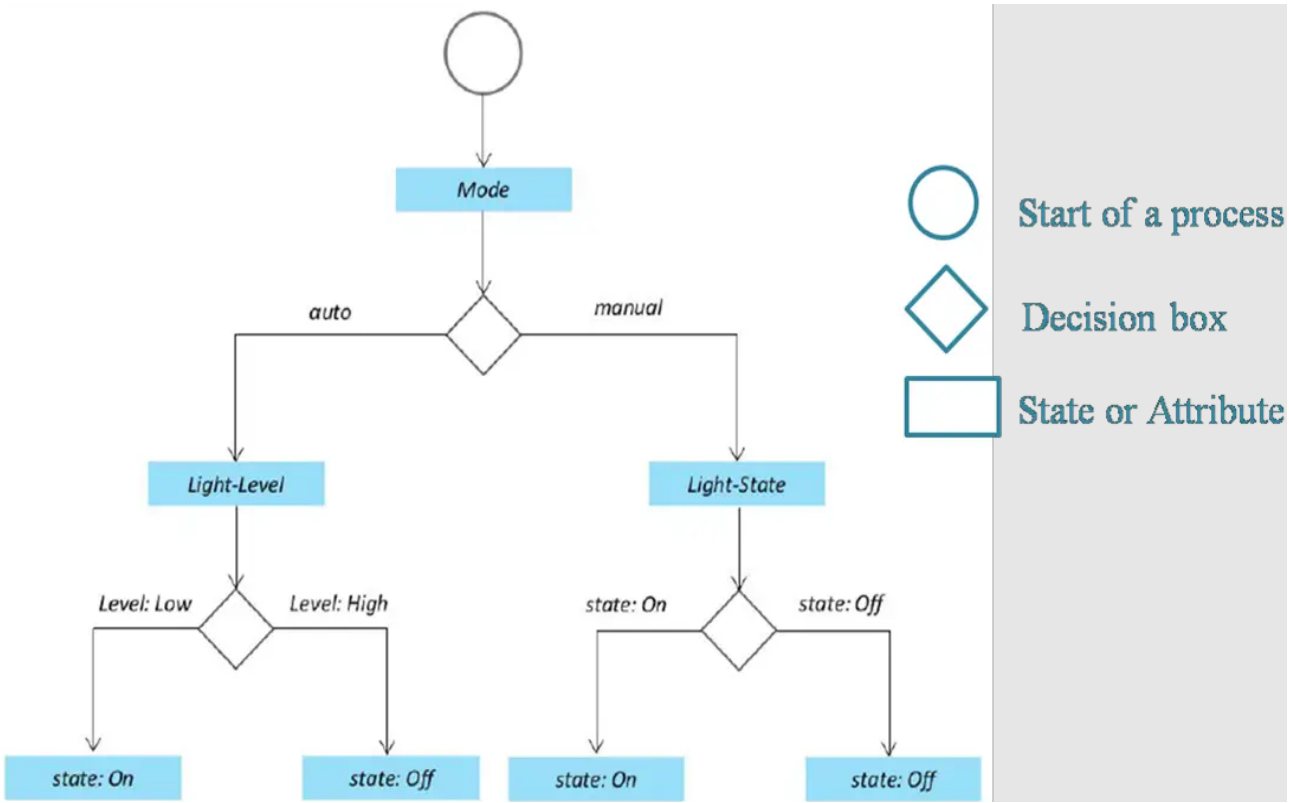
For home automation system the purpose and requirements specification is as follows:

| | |
|---|---|
| **Purpose** | A home automation system that allows controlling the lights remotely |

| | |
|---|---|
| | using a web application |
| **Behavior** | Home automation system should support two modes: auto and manual<br>**Auto:** System measures the light level in the room and switches on the light when it is dark<br><br>**Manual:** Allows remotely switching lights on and off |
| **System Management** | System should provide remote monitoring and control functions |
| **Data Analysis** | System should perform local analysis of the data |
| **Application Deployment** | Application should be deployed locally, but should be accessible remotely |
| **Security** | Should provide basic security like user authentication |

## Process Specification

The use cases of the IoT system are formally described based on or derived from the purpose and requirements specifications. The process specification for home automation system is as shown below.



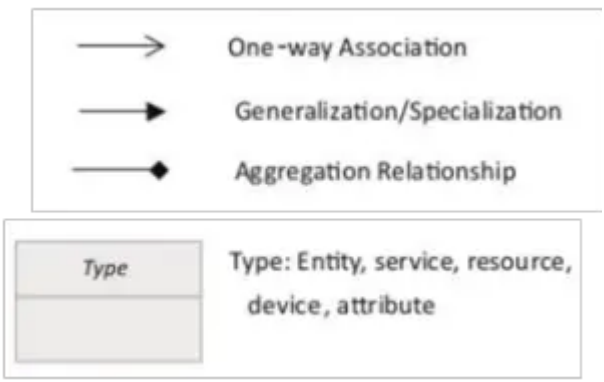

## Domain Model Specification

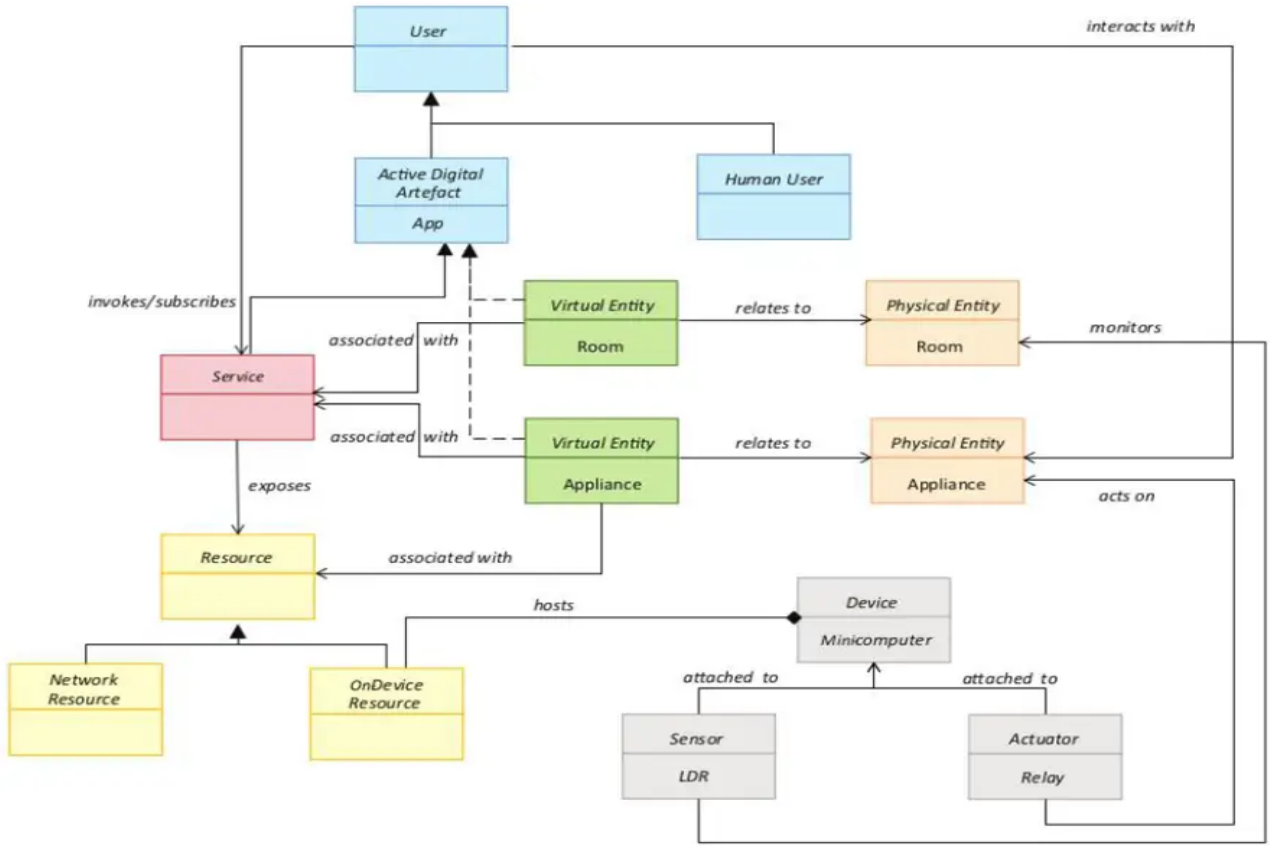
The domain model describes the main concepts, entities and objects in the domain of the IoT system to be designed. Domain model defines the attributes of the objects and relationships between objects. The domain model is independent of any specific technology or platform.

Using domain model, system designers can get an understanding of the IoT domain for which the system is to be designed. The entities, objects and concepts defined in the domain model of home automation system include the following:

| | |
|---|---|
| **Physical Entity** | • The physical identifiable objects in the environment<br>• IoT system provides information about the physical entity (using sensors) or performs actuation upon the physical entity |
| **Virtual Entity** | • Virtual entity is a representation of the physical entity in the digital world<br>• For every physical entity there is a virtual entity |
| **Device** | • Devices provide a medium for interaction between physical and virtual entities<br>• Devices are used to gather information from or perform actuation on physical entities |
| **Resource** | • Resources are software components which can be either on-device or network-resources<br>• On-device resources are hosted on the device and provide sensing or actuation (eg: operating system)<br>• Network-resources include software components that are available on the network (eg: database) |
| **Service** | • Services provide an interface for interacting with the physical entity<br>• Services access resources to perform operations on physical entities |

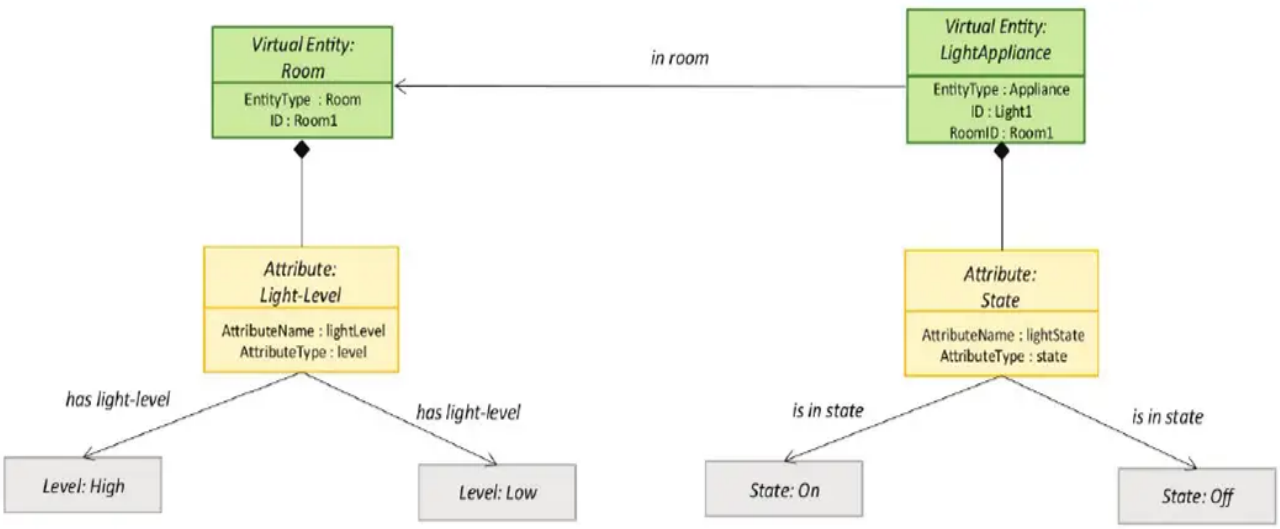
The domain model specification diagram for home automation system is as shown in the below figure.






## Information Model Specification

Information model defines the structure of all the information in the IoT system. Does not describe how the information is stored and represented. To define the information model, we first list the virtual entities. Later more details like attributes and relationships are added. The information model specification for home automation system is as shown below:

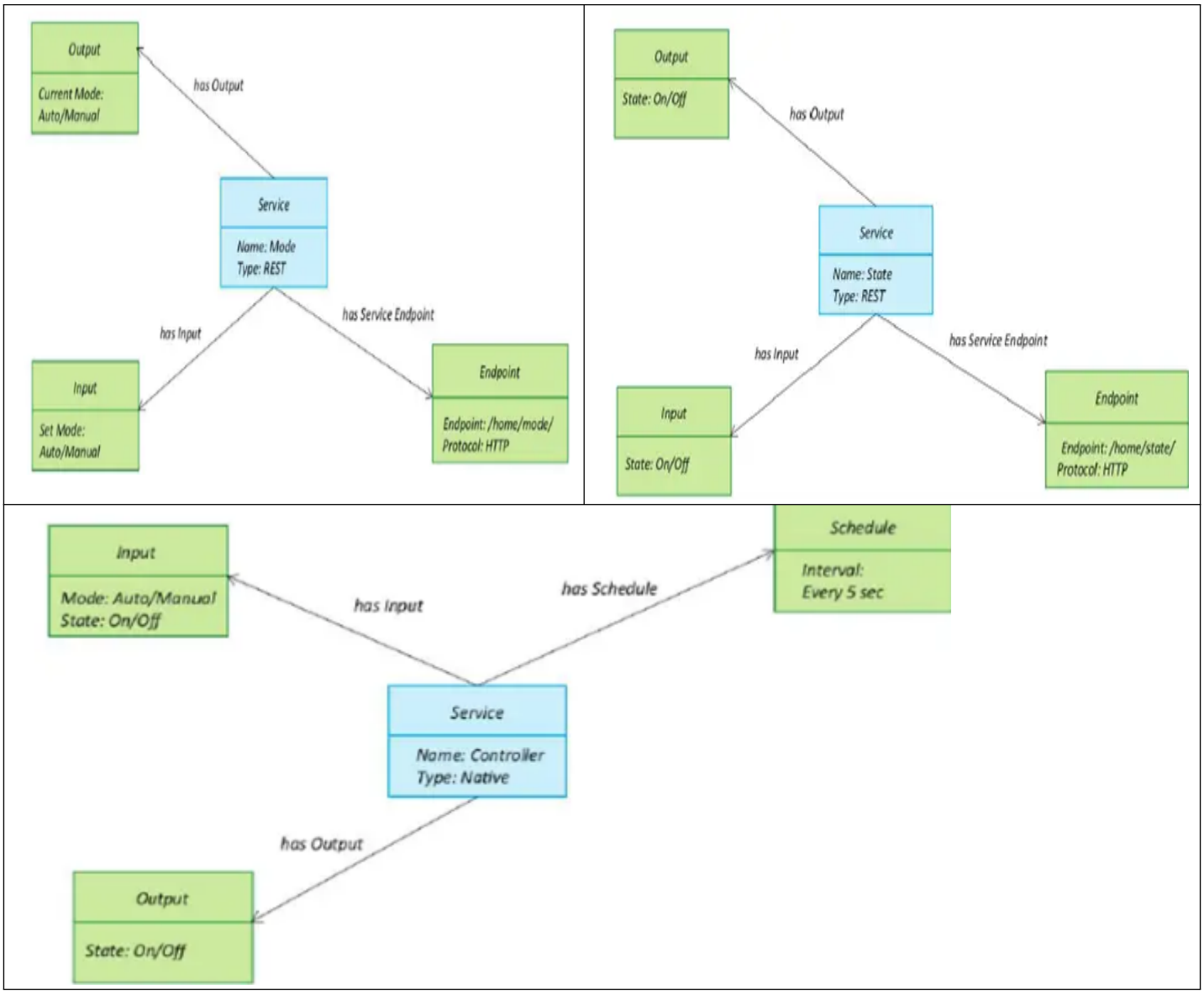



## Service Specifications

The service specification defines the following:

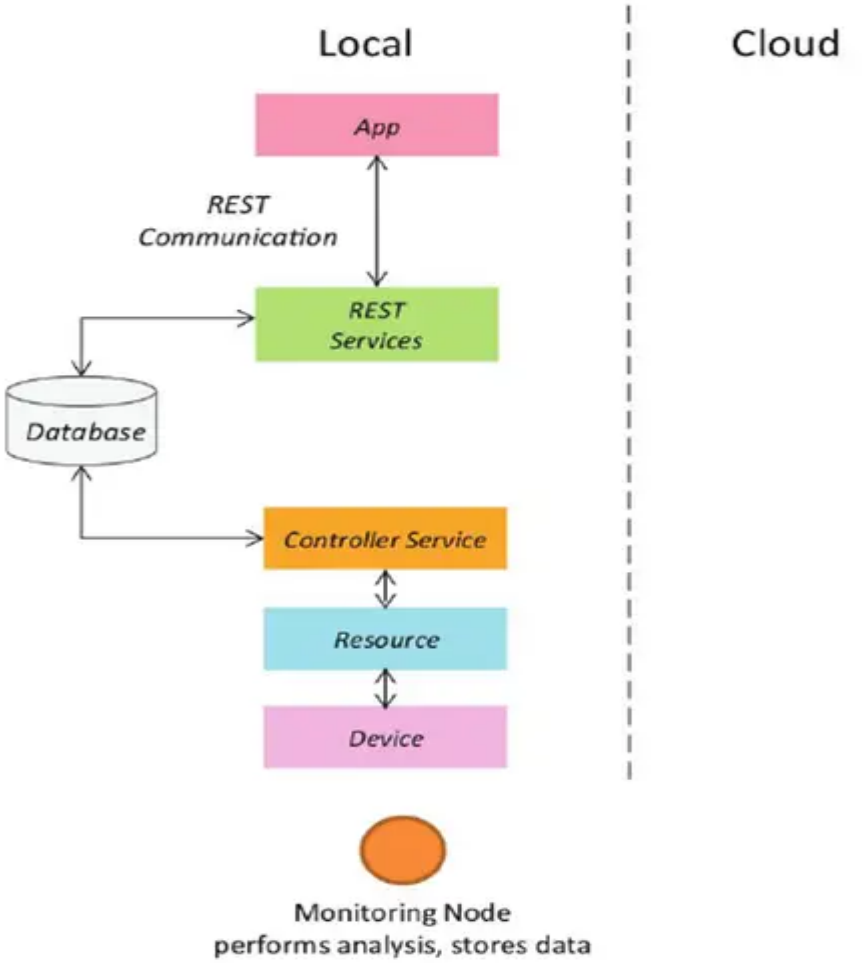
- Services in the system
- Service types
- Service inputs/output
- Service endpoints
- Service schedules
- Service preconditions
- Service effects

For each state and attribute in the process specification and information model, we define a service. Services either change the state of attributes or retrieve their current values. The service specification for each state in home automation systems are as shown below:




## IoT Level Specification

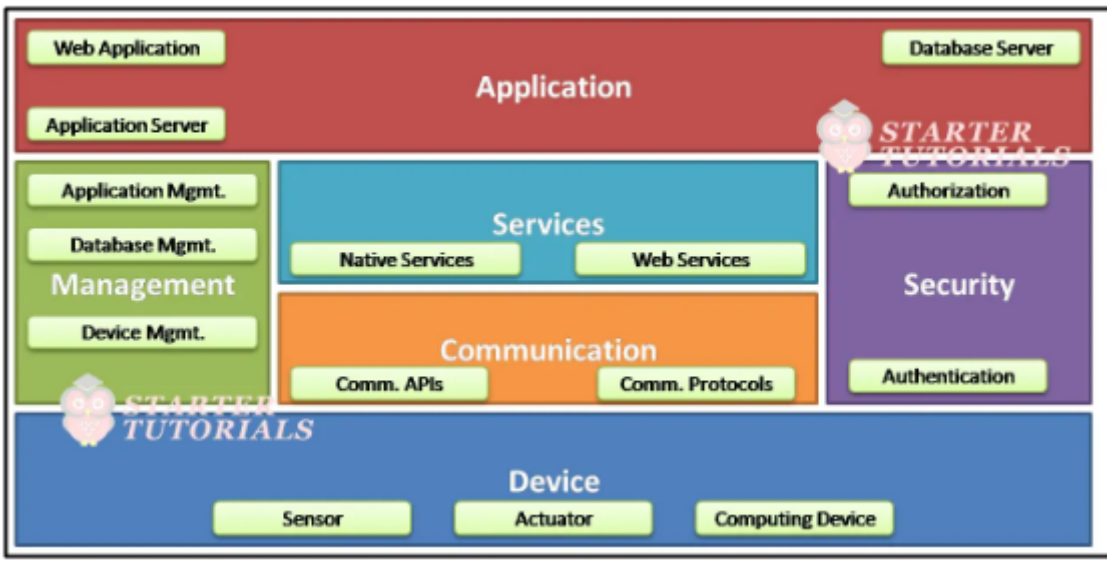
Based on the requirements we will choose the IoT application deployment level. The deployment level for home automation system is shown in the below figure.



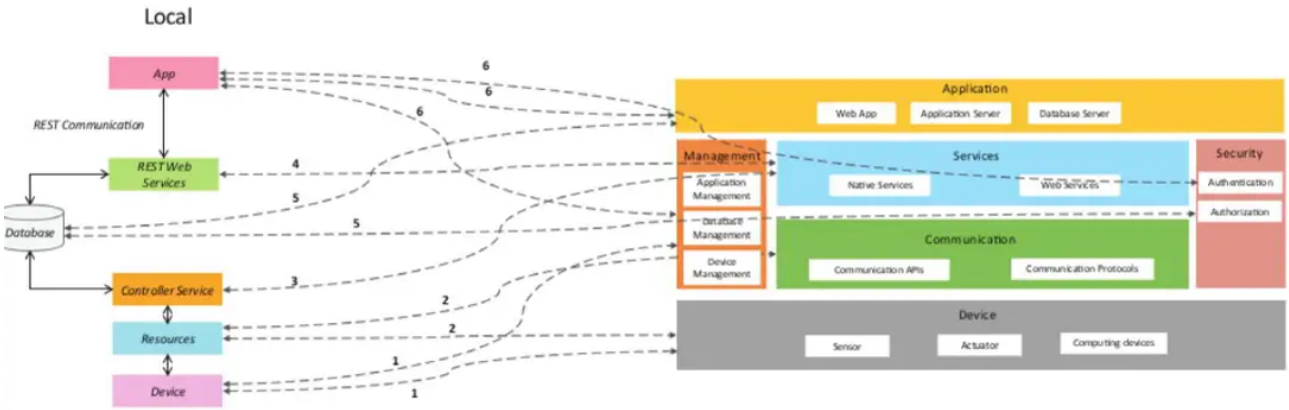

## Functional View Specification

The functional view defines the functions of the IoT systems grouped into various functional groups. Each functional group provides functionalities for interacting with concepts in the domain model and information related to the concepts.

The functional groups in a functional view include: Device, Communication, Services, Management, Security, and Application. The functional view specification for home automation system is shown in the below figure:




The mapping between the IoT level and the functional groups is as shown in the below figure.

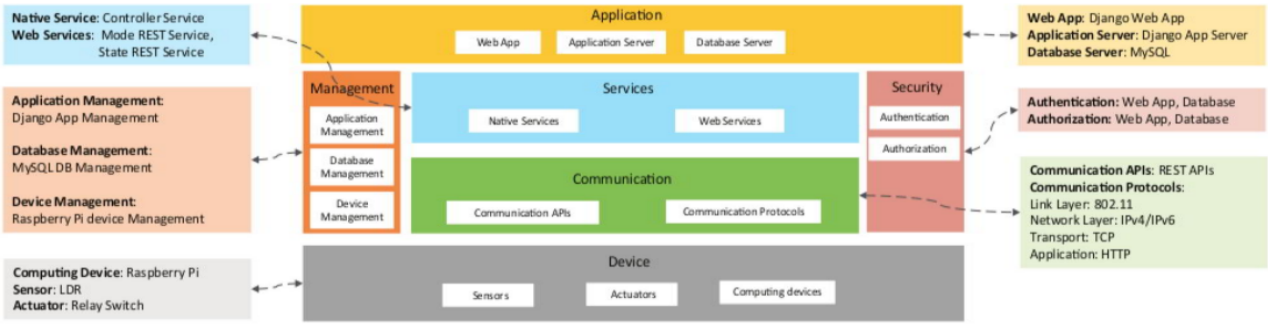



## Operational View Specification

In this step, various options related to the IoT system deployment and operation are defined, such as:

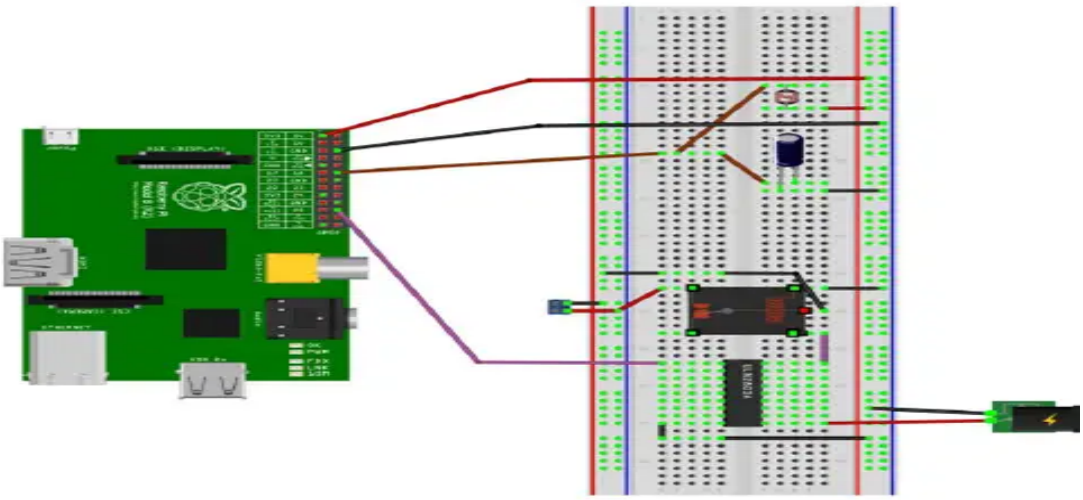
- Service hosting options
- Storage options
- Device options
- Application hosting options

The options chosen for home automation system are as shown in the below figure.




## Device and Component Integration

In this step the devices like sensors, computing devices and other components are integrated together. The interconnection of different components in our home automation system are as shown in the figure given below.



## Application Development

Using all the information from previous steps, we will develop the application (code) for the IoT system. The application interface for home automation system is shown below.


• Auto : Controls the light appliance automatically based on the lighting conditions in the room

• Light : When Auto mode is off, it is used for manually controlling the light appliance. When Auto mode is on, it reflects the current state of the light appliance.

# Tagging and Tracking

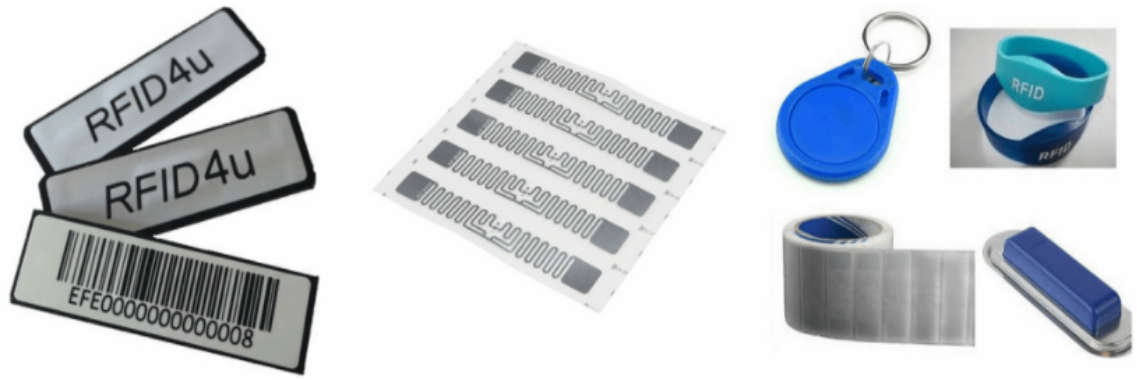
The IoT, or Internet of Things, is a progressive technology that allows for building networks of any scale that allow different types of devices to collect, process, and share data among one another and the Internet.
Tracking and tagging are among the most important features the IoT provides.
Tracking is used when you need to know the location or position of an object, learn how it moves, obtain data on how to improve its route, and so on. Related functions include monitoring the environmental conditions of the objects, building systems for real-time observation, and creating robust communication networks for sharing data obtained from tracking activities.
The majority of these issues are addressed with the help of tagging technologies. This implies putting a tag, a small connected device, on an object for either constant or ad hoc observation.

IoT tracking and tagging technologies:

- RFID tags
- Bluetooth Low Energy (BLE) beacon
- NFC
- Zigbee
- LTE Advanced
- LiFi
- GPS
- LPWAN

**RFID Tags:**

Radio-frequency identification is an IoT object management technology that uses electromagnetic fields. RFID allows for automatic identification and tracking of tags attached to objects.




There are two types of RFID tags:
**1. Passive RFID tags.** These do not have their own power source. Passive tags use radio waves created by an RFID reader to send a signal. These are mostly used in warehousing because of their simplicity: the tag is simply attached to the asset, and when it passes a reader, it is tracked when it passes a particular checkpoint.

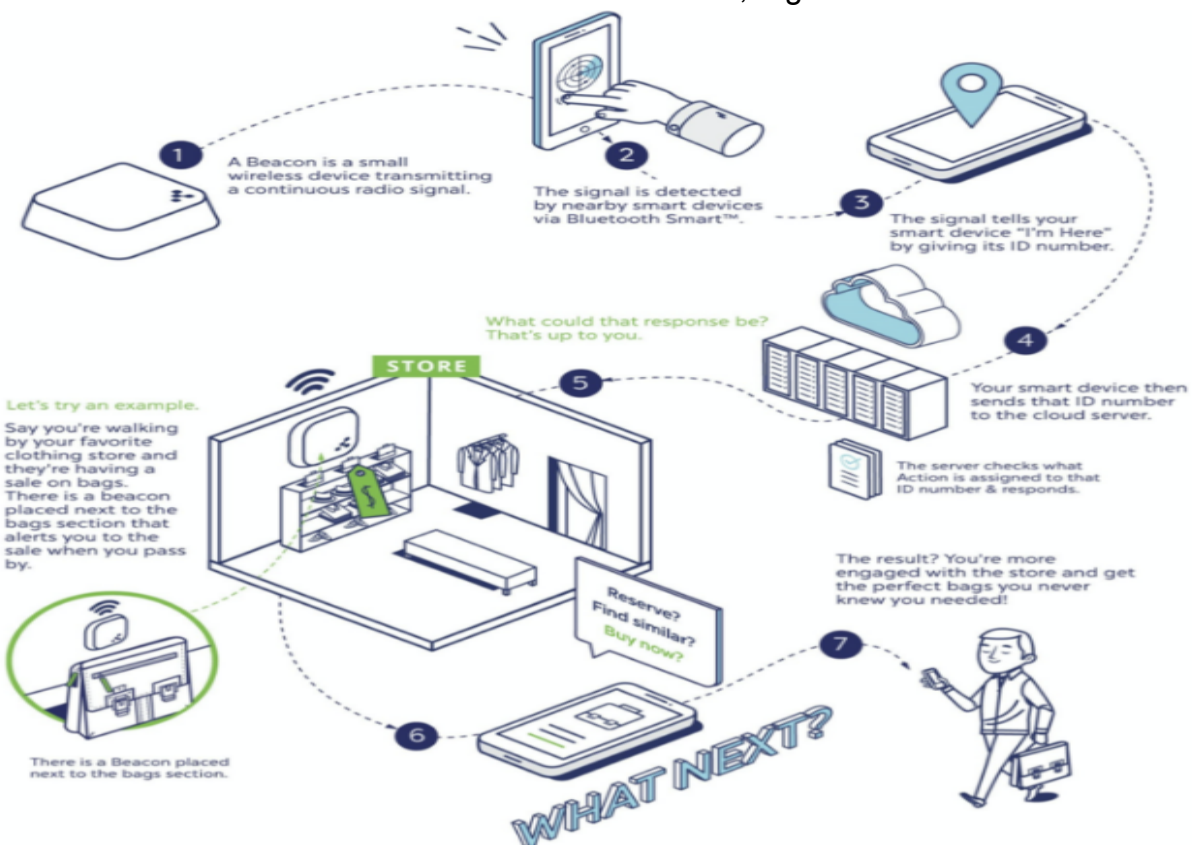
**Pros**: Cheap tags cost only about $0.10 per piece, are small and lightweight, and can work up to 20 years.
**Cons**: Expensive wide-range RFID signal readers cost $10,000 or more, without sensors and memory storage. The tag must be within a few feet of a reader to be recognized.

**2. Active RFID tags.** These tags have a small built-in battery and emit a tag's unique identifier to a particular reader non-stop. The communication range between the tag and the reader is much bigger.

**Pros**: Assets equipped with active RFID tags do not have to pass particular checkpoints to be tracked. Additionally, active tags are sensor-enabled and able to collect data from the equipment.
**Cons**: Collected data still cannot be interpreted easily or sent to the cloud for Big Data analytics. The tags are more expensive — around $20 — and they weigh more and are larger. RFID labels can't be read using a mobile phone, but only by special equipment.

## Bluetooth Low Energy (BLE) beacon

A Bluetooth beacon is a small wireless device that works based on Bluetooth Low Energy. It's kind of like a lighthouse: it repeatedly transmits a constant signal that other devices can see. Instead of emitting visible light, though, it broadcasts a radio signal that is made up of a combination of letters and numbers transmitted on short, regular intervals.




## NFC

NFC, or Near Field Communication, has its origins in RFID — the previously mentioned technology that uses electromagnetic fields to encode and read information. NFC-enabled devices have built-in chips that are activated when coming in proximity to other NFC chips.

The distance can't be more than 20 centimeters but usually does not exceed 5 cm. Therefore NFC is a safe technology for two-way interactions.

Active NFC devices such as smartphones can both send and receive data from other sources.

Passive NFC devices can only send data so that other devices can read it.

**Pros**:

- It is extremely easy to connect to the NFC chip.
- Data transmissions are fast and can be used for contactless transactions.
- NFC chips consume less power than, for example, RFID.
- The chips are well adapted to different uses, from credit cards to human implants.

**Cons**:

- NFC chips work in a very short range, which isn't useful for many industries.
- As NFC-enabled mobile phones develop, they become prone to viruses.

## Zigbee

Zigbee is a specification for a set of high-level communication protocols. These protocols are used to create personal networks for small areas with the help of small and low-power radios.


**Zigbee hardware.**

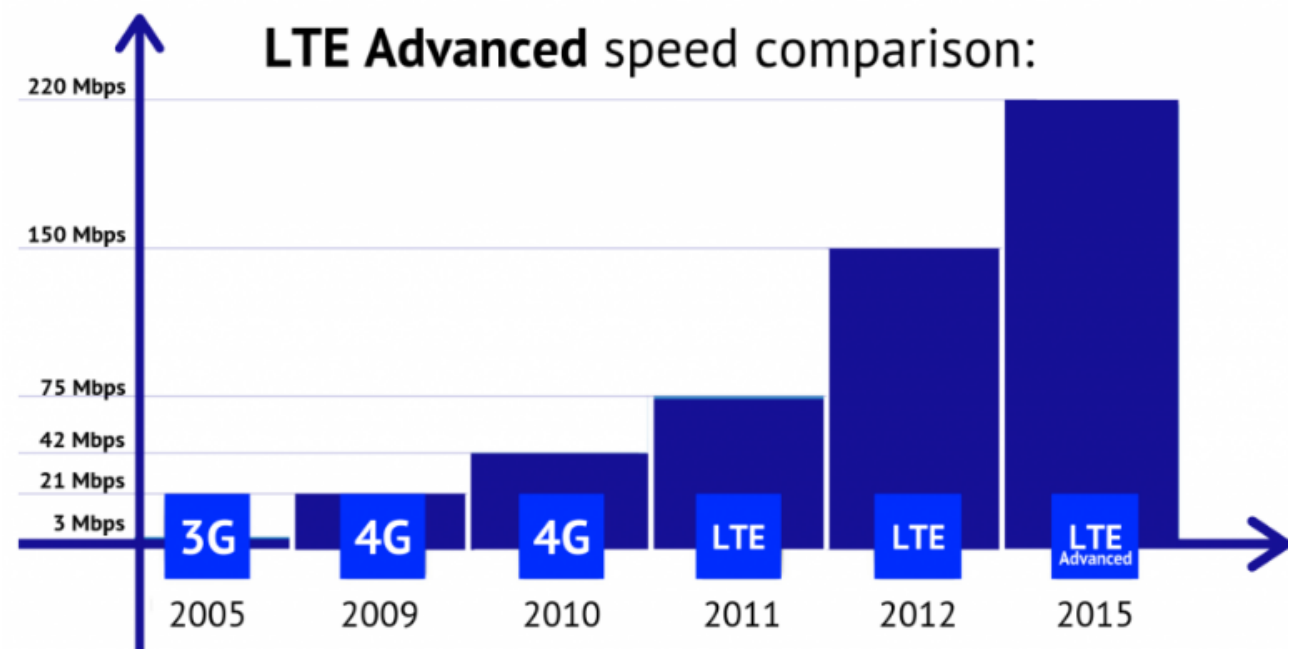
Zigbee was developed to cater to the needs of home automation enthusiasts, medical facilities that need to collect data, and other small-scale projects that require a wireless connection.

The Zigbee tagging solution was created to be a less expensive and simpler alternative to wireless personal area networks (WPAN), Bluetooth and WiFi.

**Pros:**
Can still transmit data over long distances when used with mesh networks of intermediate devices.
Allows for building secure systems.
It has long battery life.
There are open standards that allow you to mix devices from different suppliers.
**Cons:**
Low power consumption limits the data transmission distance to 100 meters maximum, depending on the environment and power output.
Network speed is limited to 200 kb/s for 866 Mhz and to 250 kb/s for 2.4 GHz range.

## LTE Advanced

Long-Term Evolution, or LTE Advanced, is a standard of wireless broadband communication with improved capacity and speed for mobile devices and larger data terminals. Different LTE frequencies and bands operate in different countries so that only multi-band devices can use LTE where it is supported.
The standard is adopted globally and sometimes called a precursor of 5G.




Pros:

- LTE Advanced can be used to create the smallest cellular tracking devices.
- LTE-based trackers consume little energy and can last longer before recharging.
- LTE has been adopted in nearly every country. It offers a longer range and better signal penetration into buildings.

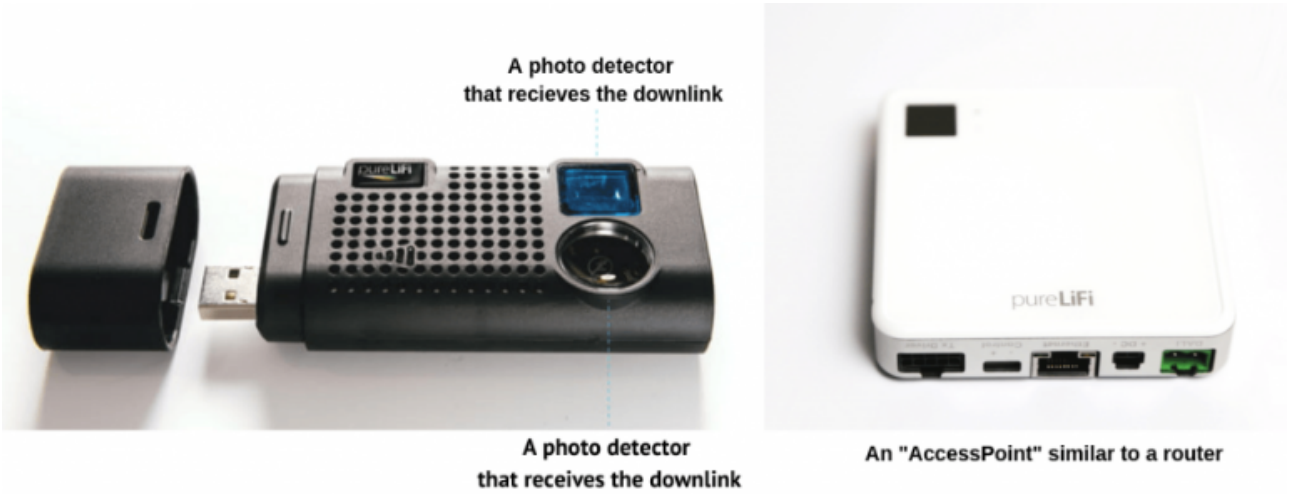
Cons:

- LTE coverage is sensitive to bandwidth and always depends on network loading.
- The longer the distance from the cell tower, the poorer the data performance.
- Applications may fail when network traffic is heavy.

## LiFi

Light Fidelity is a technology that utilizes a wireless form of Visible Light Communication. A set of diodes emit light for high-speed communication. Binary data is transmitted to the photosensitive detector by regulating the intensity of LED light. The intervals between the light signals are nanoseconds so that the human eye can not detect it.

LiFi is considered the future of wireless Internet access and has been dubbed the next technology the whole world will be using after WiFi.




However, LiFi and WiFi are similar. LiFi uses the optical spectrum to send data and WiFi uses the radio spectrum; they work similarly.

Pros:

- In terms of range, you can receive data as long as you are in the range of the emitted light.
- The range then depends on the strength of the light. Installation costs are low since LiFi works with existing LED devices, which are widespread everywhere, work for years and consume little energy.

Cons:

- Any obstacle can interfere with the light signal.
- Compatible with only Infrared Data Association devices.
- Requires an LED system to be integrated with.

## GPS

Global Positioning System is a satellite-based radio navigation technology used globally. GPS provides data on geolocation and time to every GPS receiver anywhere on or near Earth.

GPS was developed for the military but now is used for a variety of civilian applications, such as fleet management. It has high power consumption due to its constant search function, but it is still the most powerful tracking technology.


Garmin GPS computer for distraction-free driving

Small personal GPS tracking device

Matrix GPS module

Pros:

- GPS is already a mature technology, available globally.
- It's easily accessible from mobile phones equipped with a GPS receiver.
- High accuracy when the signal is not interrupted — up to 3.5 meters.
- Pretty secure and not invasive.

Cons:

- Certain obstacles such as buildings block weak GPS signals.
- Possible threats to accuracy are buildings, mountains, atmospheric effects, sky blockage.
- It consumes a lot of energy.
- Requires a specific type of a compatible receiver — different manufacturers have different formatting.
- Possible security threats come from the receivers and server communication methods.

## LPWAN

LPWAN, Low Power Wide Area Networks, is a type of wireless communication created according to the needs of IoT devices. This offers long-distance communication at a low bit rate among connected objects, and the ability to create a private wireless sensor network.


Related popular technologies, some of them competing ones, include:

- NB-IoT — a standardization effort for LPWAN in cellular networks. Evolved from NB-CIoT by Huawei.
- LoRaWAN, which uses a chirp spread spectrum radio modulation with LPWAN.

- DASH7 — low-latency firmware standard, operates over LPWAN.

Cons:

- There are various standards; some of them are not open.
- Some of the standards have no downlink communication.
- WiFi and Bluetooth signals, as well as buildings, can interfere with the LPWAN signal.

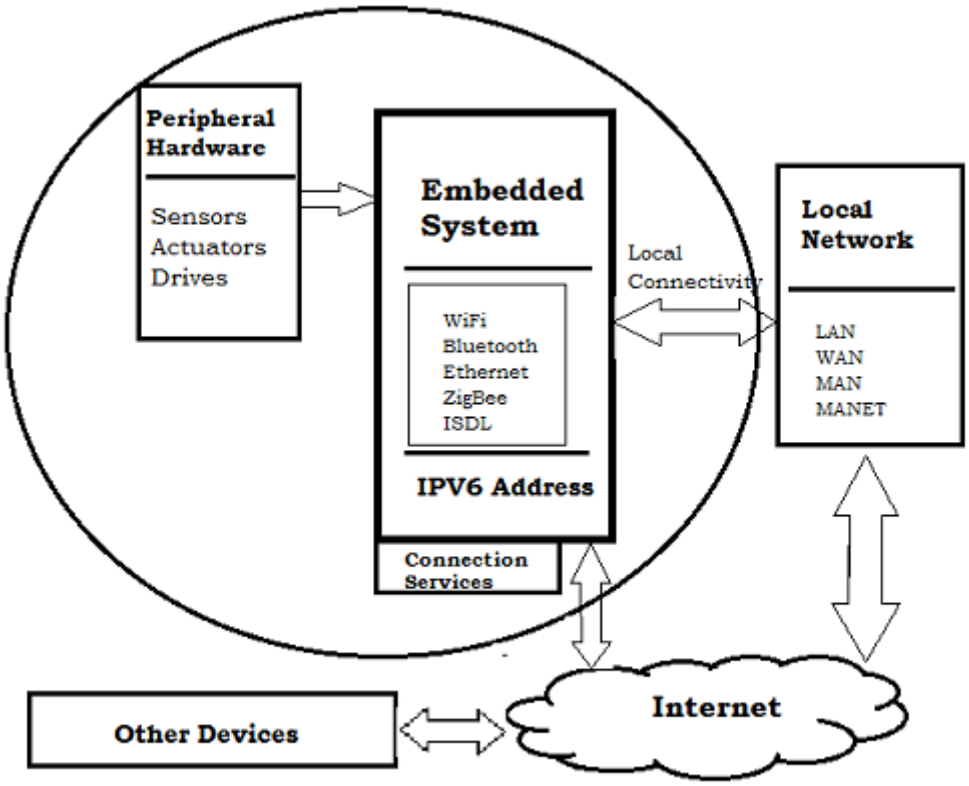
Pros:

- Low power consumption.
- Wide area coverage.
- Low bandwidth – most IoT devices send small packages of data, thus reducing the system cost.

# Embedded Products

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)**

Common embedded systems can be broken into four types based on performance as well as functional requirements:

1. Real-Time
2. Stand-alone
3. Networked
4. Mobile

### Real-Time

Real-time embedded systems are designed and installed to carry out specific tasks within a pre-defined time limit. They are further divided into two different types:

- Soft Real-Time Embedded Systems: For these systems, the completion of the task is of paramount importance, while the deadline is not a priority.
- Hard Real-Time Embedded Systems: These systems prioritize deadlines, so they shouldn't be missed in any case.

Some of the real-time embedded systems examples are:

- Sound System of a computer (Soft real-time system)
- Aircraft control system (Hard real-time system)

**Stand-alone**

These are self-sufficient systems that do not rely on a host system like a processor or a computer to perform tasks. Here are some standalone embedded technology examples:

- Microwave ovens
- Washing machines
- Video game consoles

**Networked**

These systems are connected to a wired or wireless network to perform assigned tasks and provide output to the connected devices. They are comprised of components like controllers and sensors. Here are some network embedded software examples:

- ATMs
- Home security systems
- Card swipe machines

**Mobile**

These systems are smaller in size and easy to use. Though they come with limited memory, people still prefer them due to their portability and handiness. Here are a few mobile embedded control systems examples:

- Digital cameras
- Mobile phones
- Smart watch
- Fitness tracker

## Embedded systems

Embedded systems are at the heart of many different products, machines and intelligent operations, such as machine learning and artificial intelligence applications. As embedded systems applications appear in every industry and sector today, embedded devices and software play a crucial role in the functioning of cars, home appliances, medical devices, interactive kiosks, and other equipment we use in our daily lives. In this article, we have provided embedded system examples with explanations to help you learn how this technology is impacting every facet of modern life.

While real life embedded systems have become a significant part of our lives, they are engineered to operate with minimal human intervention. Characteristics like compact size, simple design, and low cost make them a useful technology in industries like aerospace, automotive, healthcare, and even smart cities. Thus, they are one of the driving forces behind today's digital, connected, and automated world.

**How do embedded systems work?**

Embedded systems comprise hardware and software that work together to perform specific tasks. They rely on microprocessors, microcontrollers, memory, input/output communication interfaces, and a power supply to function.

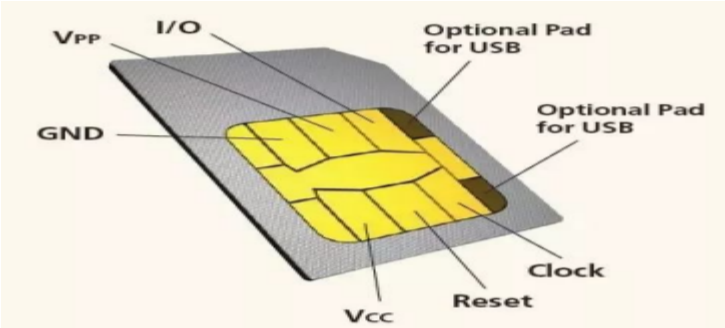
As with virtually all computers, an embedded system employs a printed circuit board (PCB) programmed with software that tells its hardware how to operate and manage data using input/output communication interfaces and memory, which terminally produces outputs valuable to the user.
Hence, embedded systems are not fundamentally different from standard rack-mount servers and workstations.

**Embedded Systems Examples**
There are many things with embedded systems incorporated in the Internet of Things (IoT), as well as in machine to machine (M2M) devices. Exceptionally versatile and adaptable, embedded systems can be found in all smart devices today. It is difficult to find a single portion of modern life that doesn't involve this technology. Here are some of the real-life examples of embedded system applications.

- Central heating systems
- GPS systems
- Fitness trackers
- Medical devices
- Automotive systems
- Transit and fare collection
- ATMs
- Factory robots
- Electric vehicle charging stations
- Interactive kiosks

## What is a SIM Card?

Mobile phones need SIM cards to connect the user to the network of their choice.
A SIM - short for Subscriber Identity Module - is a piece of plastic that slots into your smartphone (or mobile phone) that acts as your unique ID, so that you can connect to, make calls over and be charged for using a particular mobile phone network.
Without a SIM card, you can't connect to a mobile phone network, although for safety reasons, you can still dial 999.




When did SIMs first appear?
SIMs first appeared with the first digital 'GSM' mobile phones back in 1992. The Nokia 1011 was the first GSM mobile phone to go on sale. Unlike the analogue mobiles before it, GSM mobiles used digital technology for (relatively) interference-free communication — among other benefits.

Why do we use SIMs at all?
The SIM was part of a European telecommunications standard that separated mobile phones from the networks they connected to by moving all the necessary security and identification data onto a chip embedded into a removable piece of plastic the size of a credit card.
Since GSM technology also converted the voices at either end of a call into encrypted digital data before it was sent over the airwaves, the SIM also stored the ‘key’ (essentially a password) needed to decrypt this data.

## IoT SIM card

An IoT SIM card, also known as M2M SIM Card (machine-to-machine), is a Subscriber Identity Module that is used in IoT (Internet of Things) devices to identify them as they try to connect to a 2G, 3G, 4G-LTE, Cat-M, NB-IoT, or 5G wireless cellular network. IoT SIM cards are typically billed monthly through IoT Data Plans.

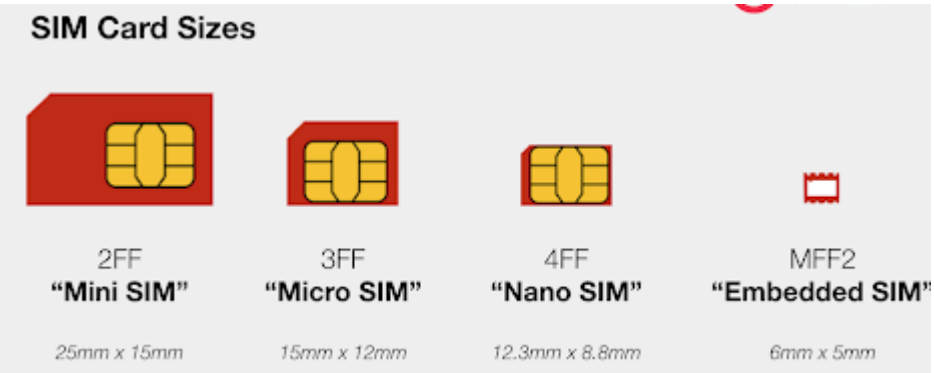
M2M SIM cards, also called Universal Integrated Circuit Cards or UICC, store the credentials and security keys that uniquely identify a cellular subscription. The SIM uses a so-called IMSI number, or International Mobile Subscriber Identity, which is unique for every connected device on or off the network, anywhere in the world. SIMs also run an application that passes that identity information to an onboard cellular modem. The modem in turn, also known as "radio" or "radio module", conducts the actual attachment operation to the network. The application that the SIM card runs, by the way, is also referred to as UICC, i.e. the acronym is often used synonymously with the card or chip itself.

For global IoT depoyments, i.e. connected devices that are running anywhere in the world, IoT SIM card technology needs to provide multi-network access to multiple carriers without requiring different cards for each country or region. These SIM cards are also sometimes called global SIM cards. Global SIM cards come in two different models:

- a traditional SIM model, also known as roaming SIMs, where the home provider uses partnerships around the world to give access to one network per visited country or region. These SIM cards typically do not provide the IoT connectivity and reliability needed to run IoT deployments successfully, as there are no fallback networks in case of outages or changing partner agreements. Furthermore, the business deploying their M2M devices has no control over which networks the connected devices should use.
- the modern SIM model that leverages so-called Multi-IMSI SIMs. These SIM cards store multiple IMSIs on a single SIM profile, where each IMSI gives access to dozens or even hundreds of networks. In sum, this creates an overlapping list of networks to offer IoT and M2M solution companies the level of choice and control needed.

**IoT SIM cards come in different sizes or form factors**

Just like traditional SIMs known from smartphones or tablets, IoT SIM cards come in different form factors, from *mini* to *micro* to *nano* SIMs. These are common in any cellular-connected hardware, as opposed to devices that, e.g., only connect through WiFi.




Common sizes / form factors of IoT SIM cards are:

1. 2FF, also known as Mini SIM: 25mm x 15mm
2. 3FF, also known as Micro SIM: 15mm x 12mm
3. 4FF, also known as Nano SIM: 12.3 x 8.8mm

For many types of IoT devices that are smaller in size, such as wearables, these SIM form factors are still too large to be practical. The predominant form factor for an industrial SIM is embedded SIMs, which are directly mounted and pre-soldered onto the circuit board (PCB), as opposed to sitting as swappable SIM cards in a slot. The most common form factor for embedded SIMs is MFF2, a 6x5mm microchip soldered onto your circuit board.

There are five common SIM card types, which are as follows:

**Nano SIM**
A nano SIM is the smallest removable SIM card size, and it's also the most modern (other than eSIMs, which we'll get to further down) having been introduced in 2012. This is basically just a small integrated circuit with almost nothing around it, and it's the kind used by the vast majority of modern devices.

**Micro SIM**
A micro SIM card is the middle size. These have more plastic around them than a nano SIM, but less than a standard SIM. They also have a slightly larger chip. In terms of smartphones, you might need one of these if your phone is over six years old, but they're rarely used in recent years. Indeed, they were introduced in 2003, so they're getting on a bit now.

**Standard SIM**
A standard SIM is the biggest SIM card size currently in use, and despite being thought of as the 'standard' it's the most rarely used. It has a large amount of plastic around the chip, and tends to only be found in the oldest of phones (and some other devices).
It was introduced back in 1996, and as the oldest of them it was the standard for a while – hence the name.
Note that standard SIM cards are also sometimes known as regular SIMs or mini SIMs – the latter because there was an even bigger credit card-sized SIM card available when these launched, but they're no longer used.

**Combi SIM**
A combi SIM (sometimes called a multi SIM or trio SIM) combines all three sizes of SIM card into one, so you can easily just pop out the one you need.
This is the type you'll typically be sent by a mobile network, so you don't need to worry about requesting the right SIM card size. Note however that while you'll be able to pop out whichever size you want, you won't then be able to make the SIM card bigger again, so if you ever need to move up a size, you'll either need an adapter (explained below) or to request a new SIM card.

**eSIM**
An eSIM is an embedded SIM card (that's what the 'e' is for) meaning that you can't remove it from your phone or other device. This is the newest type of SIM and it's found in quite a lot of modern handsets, such as the iPhone 14 range – along with many other iPhones – and the Samsung Galaxy S22 range.

These are actually even smaller than nano SIMs, therefore freeing up more space for other tech. That small size means eSIMs are also used in wearables such as Apple Watches and Samsung Galaxy Watches.
But the main advantages of eSIMs include making it theoretically much easier to change networks and plans – since you can't change the eSIM in your device, it needs to be possible to change network while keeping the same card, so there's no need to swap to a new one.

Plus, multiple networks and numbers can potentially be stored on an eSIM, so you can theoretically have multiple numbers on a single SIM. And you won't need to worry about navigating all the different SIM card sizes detailed above.
While some phones use these, at the time of writing it's always in addition to a removable SIM card slot (at least for phones sold in the UK), so you never have to use an eSIM on a smartphone. Few networks currently support them either, but it's expected that these will become the standard SIM type eventually.

## IOT CONNECTIVITY AND MANAGEMENT

Syntegra's Internet of Things (IoT) Connectivity Management solution is a multi-vendor, multilayer service that enables you to manage all your IoT and Machine-to-Machine (M2M) deployments.
This means you'll have insight into all user/device and machine/system connections. You'll also be able to manage connections, activate or deactivate connections, assign policy profiles, define connectivity and rating policies, generate usage reporting, etc.

### Why Do You Need IoT Connectivity Management?

IoT is an ascendant industry, with tens of billions of connected devices active all over the world. So, as you conceptualize your future IoT solutions, you should consider that you'll be facilitating hundreds of thousands, if not millions, of connections every day. This means you'll need a good environment from which to facilitate connectivity management and control. Here are some use cases:

**Mobile Operator**

Let's assume you're a Mobile / Cellular Operator and already have many mobile / cellular subscribers. To expand your market, you want to offer a connectivity solution to IoT developers. However, since your current management environment is not ready to handle millions of connections, you do not want to mix mobile / cellular subscribers and IoT connectivity.
The Syntegra IoT Connectivity Management solution, which integrates seamlessly into your existing IT ecosystem, can handle millions of IoT connections and makes it easy for you to manage them.

**IoT Solution Provider**

You are an IoT solution provider and already have agreements with Telecom Operators, Mobile Network Operators, eSIM providers, and other providers of connectivity solutions. You do not have a clear overview of the connections and are unable to manage all these connections.

The Syntegra IoT Connectivity Management solution solves this problem, since it can handle various types of connectivity and facilitates connectivity management from a single central platform.

**IoT Connectivity Management and More!**

Syntegra's Internet of Things (IoT) Connectivity Management Platform is flexible, scalable, and uses state of the art connectivity technologies that enable you to easily connect and manage your IoT services.

Moreover, IoT solution is multi-tenanted, so your partners can connect and manage their own IoT services, in their own secure and controlled environment.

The IoT solution facilitates quick and efficient launches of new IoT services and enables you to manage your connected IoT devices, appliances, and services in real time.
Besides the IoT Connectivity solution, also provides integration for other connectivity environments. This includes:

- Mobile /Cellular IoT Connectivity
- Wi-Fi IoT Connectivity
- Bootstrap IoT Connectivity for OEM

Some of the principal benefits of an IoT connectivity management platform are:
✔ Scalable connection management
✔ Reduced operating costs
✔ Increased control and predictability
✔ Increased security
✔ Increased visibility
✔ Powerful data insights

## What is IoT Security?

IoT Security is the act of securing Internet devices and the networks they're connected to from threats and breaches by protecting, identifying, and monitoring risks all while helping fix vulnerabilities from a range of devices that can pose security risks to your business.

### What Are the Challenges of IoT Security?

Security goals of confidentiality, integrity, and availability (CIA) apply to IoT devices, and achieving these goals poses a challenge, considering restrictions and limitations in terms of computational and power resources.

IoT security can be understood as a cybersecurity strategy and protection mechanism that safeguards against the possibility of cyberattacks which specifically target physical IoT devices that are connected to the network. Without robust security, any connected IoT device is vulnerable to breach, compromise and control by a bad actor to ultimately infiltrate, steal user data and bring down systems.

The overarching challenge for security in IoT is that as large volumes of diverse IoT devices continue to connect to the network, a dramatic expansion of the attack surface is happening in parallel. Ultimately the entire network security posture is diminished to the level of integrity and protection offered to the least secure device.

Security teams are now faced with new and escalating challenges that are unique to IoT security, including:

- Inventory – not having clear visibility and context for what IoT devices are in the network and how to securely manage new devices.
- Threats – lack of well-embedded security into IoT device operating systems that are hard or impossible to patch.
- Data volume – overseeing vast amounts of data generated from both managed and unmanaged IoT devices.
- Ownership – new risks associated with the management of IoT devices by disparate teams within the organization.
- Diversity – the sheer diversity of IoT devices in terms of their limitless forms and functions.
- Operations – the unification crisis wherein IoT devices are critical to core operations yet difficult for IT to integrate into the core security posture.

In addition to these challenges, 98% of all IoT device traffic is unencrypted, putting personal and confidential data at severe risk.

Every IoT device on the network represents an endpoint which provides a potential point of entry for a bad actor to expose the network to outside risks. This includes the IoT devices you know about as well as the IoT devices you don't know about. For example, if infected with malware, IoT devices can be used as botnets to launch distributed denial-of-service (DDoS) attacks on the network the bad actor wants to bring down. However, unlike IT devices, a growing number of IoT devices are virtually invisible in enterprise networks, making it impossible to protect them all in the same way.

### How to secure IoT devices

There are a few general protective measures that you can set to ensure IoT security. These include using authorized software in IoT devices. Also, when an IoT device is switched on, it should authenticate itself into the network before it collects or sends data.

It's necessary to set up **firewalls** to filter packets sent to IoT endpoints, as they have limited computation capability and memory. You should also ensure that updates and patches are installed without consuming the additional bandwidth.
Apart from general security measures, you need to consider some unique security practices while planning the security of IoT devices. You need to ensure device security, **network security**, and make sure that the overall IoT infrastructure and system are secure from **IoT vulnerabilities**.
You can adopt the following security practices to secure IoT devices:

- **Ensure physical security:** Keep IoT devices relatively isolated and protected from physical access.
- **Deploy tamper-resistant devices:** Deploy IoT devices that are tamper-resistant, where the device is disabled when tampered with.
- **Update firmware and install patches:** Be proactive in upgrading, updating firmware, and installing patches as soon as the manufacturer releases them.
- **Perform dynamic testing:** It exposes both code weaknesses and security vulnerabilities presented by the hardware.
- **Protect data on device disposal:** Specify procedures to discard IoT devices when they become obsolete. Improperly discarded devices can pose a threat to privacy and serve various malicious purposes.
- **Use robust authentication:** Avoid using default passwords as they introduce a threat of password hacking. Use sophisticated passwords for authentication and resist educated guessing.
- **Encourage the use of adaptive authentication:** Adaptive authentication or context-aware authentication (CAA) uses contextual information and **machine learning** algorithms to assess the risk of malice. If the risk is high, the user will be asked for a multi-factor token.
- **Use strong encryption and protocols:** Maintain secure data transmission by using strong encryption in various IoT protocols ( Bluetooth, Zigbee, Z-Wave, Thread, Wi-Fi, cellular, 6LoWPAN, NFC, etc.)
- **Minimize device bandwidth:** Restrict network capability and bandwidth to the least that is required for the device to function and avoid being targets of IoT-borne distributed denial of service (DDoS) attacks.
- **Segment the network:** Divide networks into smaller local networks using virtual local area networks (VLANs), IP address ranges, and their combinations. This allows you to create different security zones and represent different segments controlled by firewalls.
- **Protect sensitive information:** Avoid leakages in sensitive personally identifiable information (PII) by restricting the discovery of these devices. You'd need proper service mechanisms and authentication protocols so that authorized clients can discover the IoT device.

# IoT Communication:

IoT is the connection of devices over the internet, where these smart devices communicate with each other , exchange data , perform some tasks without any human involvement. These devices are embedded with electronics, software, network and sensors which help in communication. Communication between smart devices is very important in IOT as it enables these devices to gather, exchange data which contribute in success of that IOT product/project.

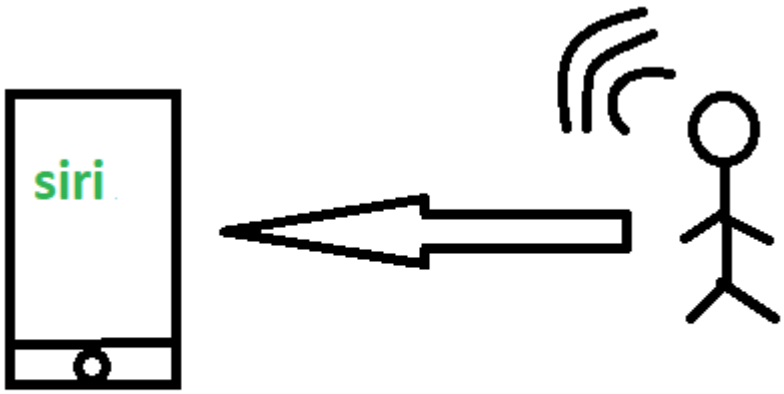
**Types of Communications in IOT:**

The following are some communication types in IoT:-

1. Human to Machine (H2M)
2. M2M
3. Machine to Human (M2H)
4 Human to Human (H2H)

**Human to Machine (H2M):**

In this human gives input to IOT device i.e as speech/text/image etc. IOT device (Machine) like sensors and actuators then understands input, analyses it and responds back to human by means of text or Visual Display. This is very useful as these machines assist humans in every everyday tasks. It is a combo of software and hardware that includes human interaction with a machine to perform a task.




*H2M communication*

Merits: This H2M has a user-friendly interface that can be quickly accessed by following the instructions. It responds more quickly to any fault or failure. Its features and functions can be customized.

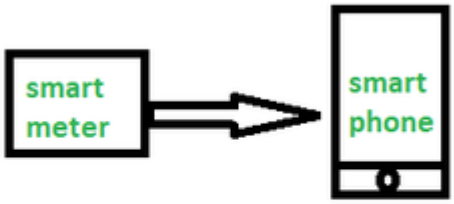
Examples:

- Facial recognition.
- Bio-metric Attendance system.
- Speech or voice recognition.

**Machine to Machine (M2M):**

The process of exchanging information or messages between two or more machines or devices is known as Machine to Machine (M2M) communication.

It is the communication among the physical things which do not need human intervention.




*M2M communication*

M2M communication is also named as Machine Type communication in 3GPP(3rd Generation Partnership Project).In this the interaction or communication takes place between machines by automating data/programs. In this machine level instructions are required for communication. Here communication takes place without human interaction. The machines may be either connected through wires or by wireless connection. An M2M connection is a point-to-point connection between two network devices that helps in transmitting information using public networking technologies like Ethernet and cellular networks. IoT uses the basic concepts of M2M and expands by creating large "cloud" networks of devices that communicate with one another through cloud networking platforms.

**Advantages –**

This M2M can operate over cellular networks and is simple to manage. It can be used both indoors and outdoors and aids in the communication of smart objects without the need for human interaction. The M2M contact facility is used to address security and privacy problems in IoT networks. Large-scale data collection, processing, and security are all feasible.

**Disadvantages –**

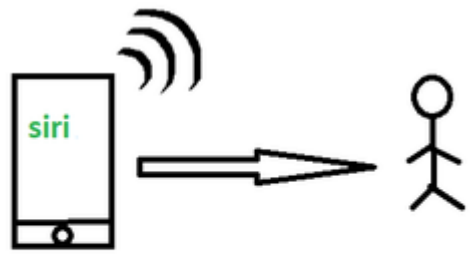
However, in M2M, use of cloud computing restricts versatility and creativity. Data security and ownership are major concerns here. The challenge of achieving interoperability between cloud/M2M IoT systems is daunting. M2M connectivity necessitates the existence of a reliable internet connection.

**Examples:**

1. Smart Washing machine sends alerts to the owners' smart devices after completion of washing or drying of clothes.
2. Smart meters tracks amount of energy used in household or in companies and automatically alert the owner.

**Machine to Human (M2H) :**

In this machine interacts with Humans. Machine triggers information(text messages/images/voice/signals) respective / irrespective of any human presence. This type of communication is most commonly used where machines guide humans in their daily life. It is way of interaction in which humans co-work with smart systems and other machines by using tools or devices to finish a task.




*M2H communication*

Examples:

- Fire Alarms
- Traffic Light
- Fitness bands
- Health monitoring devices

**Human to Human (H2H) :**

This is generally how humans communicate with each other to exchange information by speech, writing, drawing, facial expressions, body language etc. Without H2H, M2M applications cannot produce the expected benefits unless humans can immediately fix issues, solve challenges, and manage scenarios.

The process of exchanging information
or messages between two or more people is known as human to human (H2H) communication. This can be done through various means such as verbal,
non-verbal, or written communication.


*H2H communication*

For, communication of IoT devices many protocols are used. These IoT protocols are modes of communication which give security to the data being exchanged between IoT connected devices. Example bluetooth, wifi, zigbee etc.