SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – II – Internet of Things – SCSA5301

ELEMENTS OF IOT

Application Sensors & Actuators - Edge Networking (WSN) – Gateways - IoT Communication Model – WPAN & LPWA, IoT platform for available applications, Hardware Devices: Arduino, Raspberry pi and Smartwifi, etc, Wearable Development Boards, Softwares, Programs and Stacks available for building IoT applications, Installation of various packages necessary for project and list of tools.

1.SENSORS AND ACTUATORS

A transducer is any physical device that converts one form of energy into another. So, in the case of a sensor, the transducer converts some physical phenomenon into an electrical impulse that can then be interpreted to determine a reading. A microphone is a sensor that takes vibration energy (sound waves), and converts it to electrical energy in a useful way for other components in the system to correlate back to the original sound.

Another type of transducer that we will encounter in many IoT systems is an actuator. In simple terms, an actuator operates in the reverse direction of a sensor. It takes an electrical input and turns it into physical action. For instance, an electric motor, a hydraulic system, and a pneumatic system are all different types of actuators.

Examples of actuators

- Digital micromirror device
- Electric motor
- Electroactive polymer
- Hydraulic cylinder
- Piezoelectric actuator
- Pneumatic actuator
- Screw jack
- Servomechanism
- Solenoid
- Stepper motor

In typical IoT systems, a sensor may collect information and route to a control center where a decision is made and a corresponding command is sent back to an actuator in response to that sensed input. There are many different types of sensors. Flow sensors, temperature sensors, voltage sensors, humidity sensors, and the list goes on. In addition, there are multiple ways to measure the same thing. For instance, airflow might be measured by using a small propeller like the one you would see on a weather station. Alternatively, as in a vehicle measuring the air through the engine, airflow is measured by heating a small element and measuring the rate at which the element is cooling.

We live in a World of Sensors. You can find different types of Sensors in our homes, offices, cars etc. working to make our lives easier by turning on the lights by detecting our presence, adjusting the room temperature, detect smoke or fire, make us delicious coffee, open garage doors as soon as our car is near the door and many other tasks.
The example we are talking about here is the Autopilot System in aircrafts. Almost all civilian and military aircrafts have the feature of Automatic Flight Control system or sometimes called as Autopilot. An Automatic Flight Control System consists of several sensors for various tasks like speed control, height, position, doors, obstacle, fuel and many more. A Computer takes data from all these sensors and processes them by comparing them with pre-designed values. The computer then provides control signal to different parts like engines, flaps, rudders etc. that help in a smooth flight.

All the parameters i.e. the Sensors (which give inputs to the Computers), the Computers (the brains of the system) and the mechanics (the outputs of the system like engines and motors) are equally important in building a successful automated system. Sensor as an input device which provides an output (signal) with respect to a specific physical quantity (input). Sensor means that it is part of a bigger system which provides input to a main control system (like a Processor or a Microcontroller).

S.No	**Sensor**	**Applications**	**Technology**
1.	Inertial sensors	Industrial machinery, automotive, human activity	MEMS and Gyroscope
2.	Speed Measuring Sensor	Industrial machinery, automotive, human activity	Magnetic, light
3.	Proximity sensor	Industrial machinery, automotive, human activity	Capacitive, Inductive, Magnetic, Light, Ultrasound
4.	Occupancy sensor	Home/office monitoring	PassiveIR, Ultrasound most common
5.	Temperature/humidity sensor	Home/office HVAC control, automotive, industrial	Solid state, thermocouple
6.	Light sensor	Home/office/industrial lighting control	Solid state, photocell, Photo resistor, photodiode

7.	Power (current) sensor	Home/office/industrial powermonitoring/control Technology	Coil (Faraday's law), Hall effect
8.	Air/fluid pressure sensor	Industrial monitoring/control, automotive, agriculture	Capacitive, Resistive
9.	Acoustic sensor	Industrial monitoring/control, human interface	Diaphragm condenser
10.	Strain sensor	Industrial monitoring/control, civil infrastructure	Resistive thin films

In the first classification of the sensors, they are divided in to Active and Passive. Active Sensors are those which require an external excitation signal or a power signal. Passive Sensors, on the other hand, do not require any external power signal and directly generates output response. The other type of classification is based on the means of detection used in the sensor. Some of the means of detection are Electric, Biological, Chemical, Radioactive etc.

The next classification is based on conversion phenomenon i.e. the input and the output. Some of the common conversion phenomena are Photoelectric, Thermoelectric, Electrochemical, Electromagnetic, Thermo-optic, etc. The final classification of the sensors are Analog and Digital Sensors. Analog Sensors produce an analog output i.e. a continuous output signal with respect to the quantity being measured.

Digital Sensors, in contrast to Analog Sensors, work with discrete or digital data. The data in digital sensors, which is used for conversion and transmission, is digital in nature.

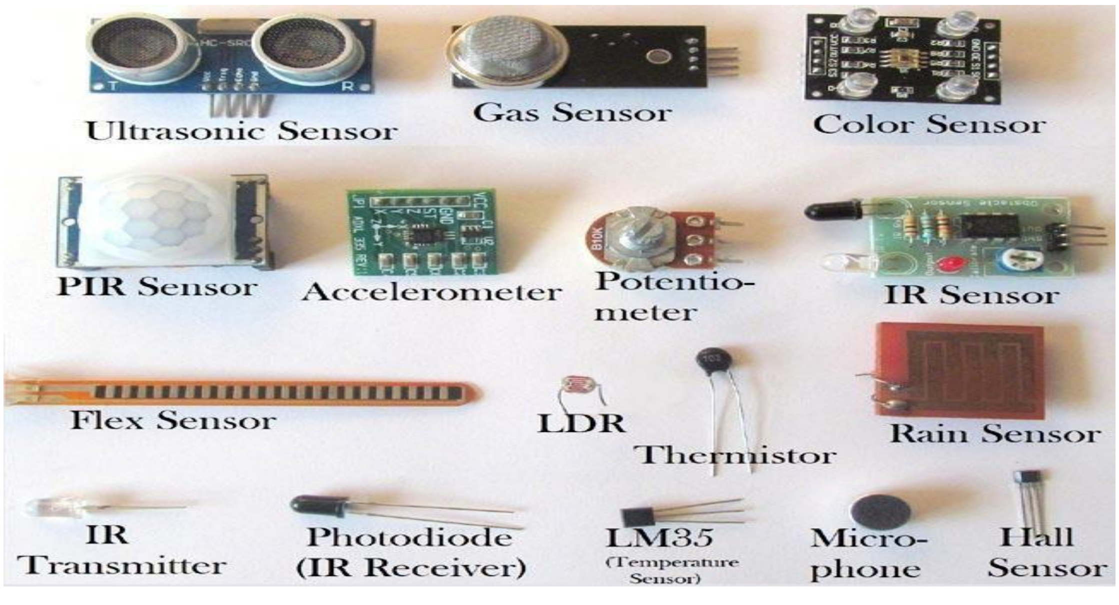


Fig 1.Examples of Sensors

1.IR LED

It is also called as IR Transmitter. It is used to **emit Infrared rays**. The range of these frequencies are greater than the microwave frequencies (i.e. >300GHz to few hundreds of THz). The rays generated by an infrared LED can be sensed by Photodiode explained below. **The pair of** IR LED and photodiode is called IR Sensor**.**

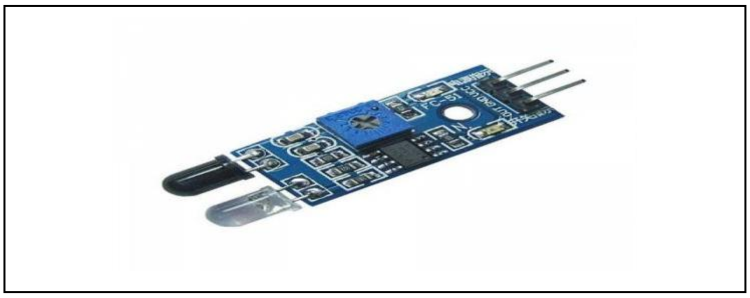
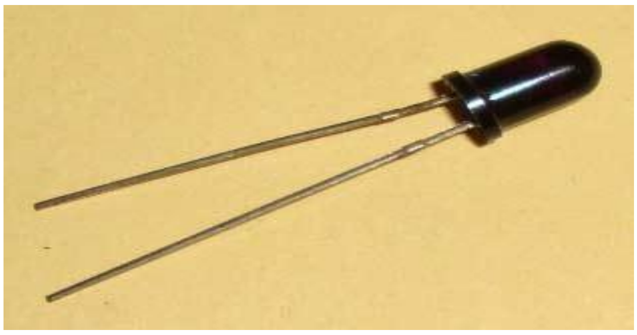


Fig 2. LED sensor

2.Photo Diode (Light Sensor)

It is a semiconductor device which *is* used to detect the light rays and mostly used as IR Receiver. Its construction is similar to the normal PN junction diode but the working principle differs from it. As we know a PN junction allows small leakage currents when it is reverse biased so, this property is used to detect the light rays. A photodiode is constructed such that light rays should fall on the PN junction which makes the leakage current increase based on the intensity of the light that we have applied. So, in this way, a



photodiode can be used to sense the light *rays* and maintain the current through the circuit. Check here the working of Photodiode with IR sensor.

Fig 3.Photo diode

3.Proximity Sensor

A Proximity Sensor is a non-contact type sensor that detects the presence of an object. Proximity Sensors can be implemented using different techniques like Optical (like Infrared or Laser), Ultrasonic, Hall Effect, Capacitive, etc.

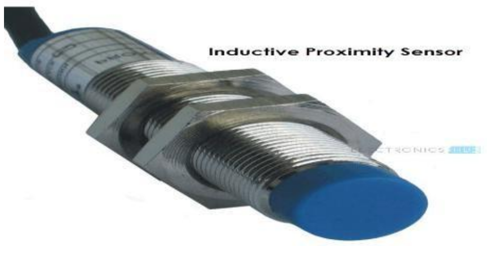


Fig 4.Proximity sensor

Some of the applications of Proximity Sensors are Mobile Phones, Cars (Parking Sensors), industries (object alignment), **Ground Proximity in Aircrafts, etc. Proximity Sensor in Reverse Parking is implemented in this Project:** Reverse Parking Sensor Circuit**.**

4.LDR (Light Dependent Resistor)

As the name itself specifies that the resistor that depends upon the light intensity. It works on the principle of photoconductivity which means the conduction due to the light. It is generally made up of Cadmium sulfide. When light falls on the LDR**,** its resistance decreases and acts similar to a conductor and when no light falls on it, its resistance is almost in the range of MΩ or ideally it acts as an open circuit**.** One note should be considered with LDR is that it won‘t respond if the light is not exactly focused on its surface.

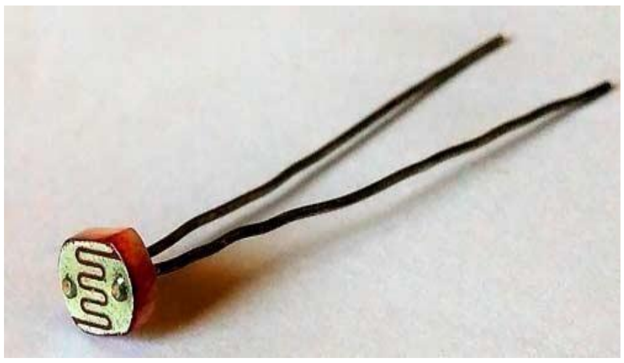


Fig 5.LDR

With a proper circuitry using a transistor it can be used to detect the availability of light. A voltage divider biased transistor with R2 (resistor between base and emitter) replaced with an LDR can work as a light detector.

5.Thermistor (Temperature Sensor)

A thermistor can be used to detect the variation in temperature**.** It has a negative temperature coefficient that means when the temperature increases the resistance decreases. So, the thermistor‘s resistance can be varied with the rise in temperature which causes more current flow through it. This change in current flow can be used to determine the amount of change in temperature. An application for thermistor is, it is used to detect the rise in temperature and control the leakage current in a transistor circuit which helps in maintaining its stability. Here is one simple application for Thermistor to control the DC fan automatically.

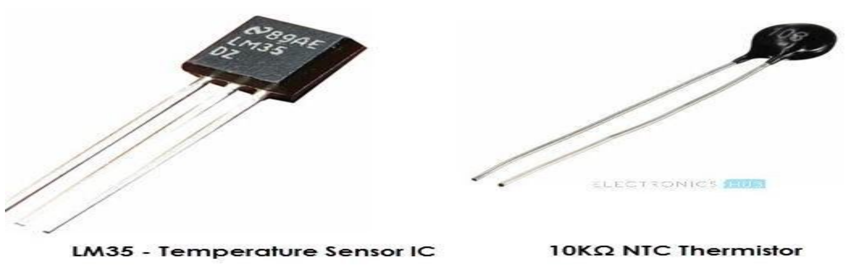


Fig 6.Thermistor

6.Thermocouple (Temperature Sensor)

Another component that can detect the variation in temperature is a thermocouple**.** In its construction, two different metals are joined together to form a junction. Its main principle is when the junction of two different metals are heated or exposed to high temperatures a potential across their terminals varies. So, the varying potential can be further used to measure the amount of change in temperature.

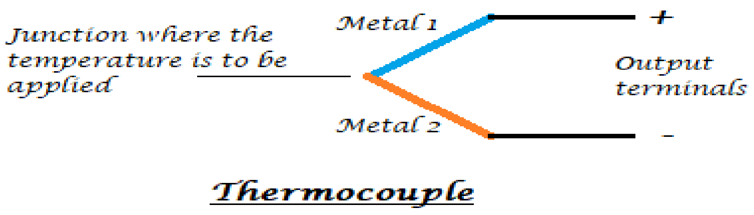


Fig 7.Thermo couple

7.Strain Gauge (Pressure/Force Sensor)

A strain gauge is used to detect pressure when a load is applied**.** It works on the principle of resistance, we know that the resistance is directly proportional to the length of the wire and is inversely proportional to its cross-sectional area (R=ρl/a). The same principle can be used here to measure the load. On a flexible board, a wire is arranged in a zig-zag manner as shown in the figure below. So, when the pressure is applied to that particular board, it bends in a direction causing the change in overall length and cross- sectional area of the wire. This leads to change in resistance of the wire. The resistance thus obtained is very minute (few ohms) which can be determined with the help of the Wheatstone bridge. The strain gauge is placed in one of the four arms in a bridge with the remaining values unchanged. Therefore, when the pressure is applied to it as the resistance changes the current passing through the bridge varies and pressure can be calculated.

Strain gauges are majorly used to calculate the amount of pressure that an airplane wing can withstand and it is also used to measure the number of vehicles allowable on a particular road etc.

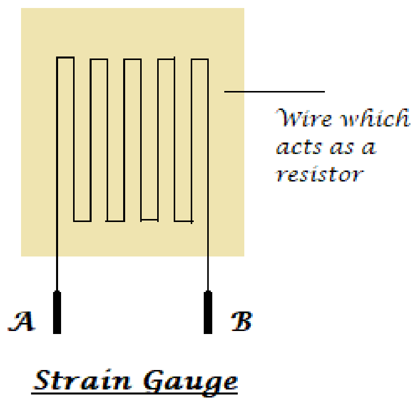
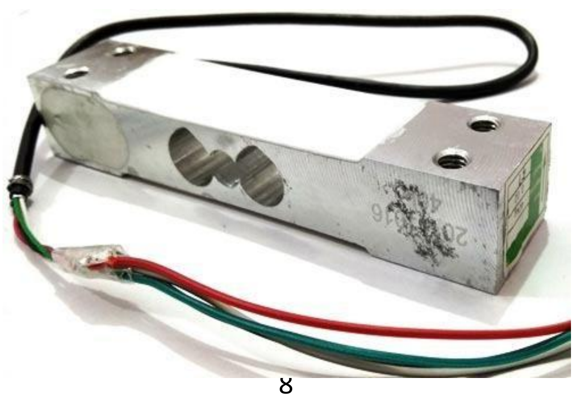


Fig 8.Strain Guage

8.Load Cell (Weight Sensor)

Load cells are similar to strain gauges which measure the physical quantity like force and give the output in form of electrical signals. When some tension is applied on the load cell it structure varies causing the change in resistance and finally, its value can be calibrated



using a Wheatstone bridge. Here is the project on how to measure weight using Load cell.

Fig 9.Load Cell

9.Potentiometer

A potentiometer is used to detect the position. It generally has various ranges of resistors connected to different poles of the switch. A potentiometer can be either rotary or linear type. In rotary type, a wiper is connected to a long shaft which can be rotated. When the shaft has rotated the position of the wiper alters such that the resultant resistance varies causing the change in the output voltage. Thus the output can be calibrated to detect the change its position.

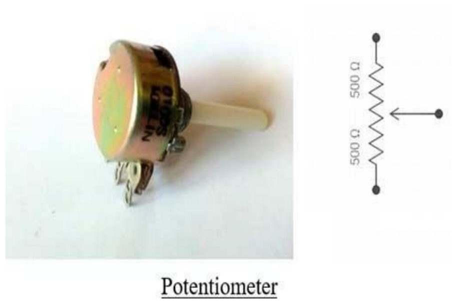


Fig 10.Potentiometer

10.Encoder

To detect the change in the position an encoder can also be used. It has a circular rotatable disk-like structure with specific openings in between such that when the IR rays or light rays pass through it only a few light rays get detected. Further, these rays are encoded into

a digital data (in terms of binary) which represents the specific position.

Fig 11.Encoder

11 Hall Sensor

The name itself states that it is the sensor which works on the Hall Effect. It can be defined as when a magnetic field is brought close to the current carrying conductor (perpendicular to the direction of the electric field) then a potential difference is developed across the given conductor. Using this property a Hall sensor is used to detect the magnetic field and gives output in terms of voltage. Care should be taken that the Hall sensor can detect only one pole of the magnet.

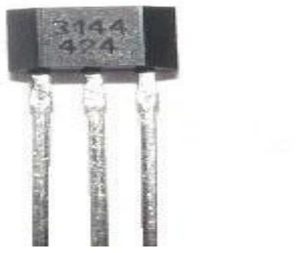


Fig 12.Hall sensor

The hall sensor is used in few smartphones which are helpful in turning off the screen when the flap cover (which has a magnet in it) is closed onto the screen. Here is one practical application of Hall Effect sensor in Door Alarm.

12. Flex Sensor

A FLEX sensor is a transducer which changes its resistance when its shape is changed or when it is bent. A FLEX sensor is 2.2 inches long or of finger length. Simply speaking the sensor terminal resistance increases when it‘s bent. This change in resistance can do no good unless we can read them. The controller at hand can only read the changes in

voltage and nothing less, for this, we are going to use voltage divider circuit, with that we can derive the resistance change as a voltage change.

Fig 13. Flex sensor

13.Microphone (Sound Sensor)

Microphone can be seen on all the smartphones or mobiles. It can detect the audio signal and convert them into small voltage (mV) electrical signals. A microphone can be of many types like condenser microphone, crystal microphone, carbon microphone etc. each type of microphone work on the properties like capacitance, piezoelectric effect, resistance respectively. Let us see the operation of a crystal microphone which works on the piezoelectric effect. A bimorph crystal is used which under pressure or vibrations produces proportional alternating voltage. A diaphragm is connected to the crystal through a drive pin such that when the sound signal hits the diaphragm it moves to and fro, this movement changes the position of the drive pin which causes vibrations in the crystal thus an alternating voltage is generated with respect to the applied sound signal. The obtained voltage is fed to an amplifier in order to increase the overall strength of the signal.

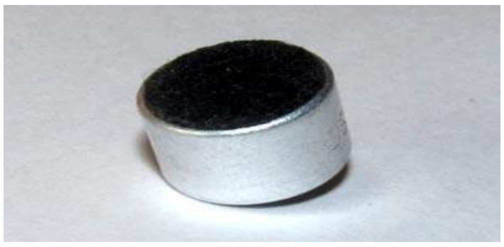


Fig 14.Microphone

14.Ultrasonic sensor

Ultrasonic means nothing but the range of the frequencies. Its range is greater than audible range (>20 kHz) so even it is switched on we can‘t sense these sound signals. Only specific speakers and receivers can sense those ultrasonic waves. This ultrasonic sensor is *used to calculate the distance between the ultrasonic transmitter and the target and also used to measure the velocity of the target***.**

Ultrasonic sensor HC-SR04 can be used to measure distance in the range of 2cm-400cm with an accuracy of 3mm. Let‘s see how this module works. The HCSR04 module generates a sound vibration in ultrasonic range when we make the ‗Trigger‘ pin high for about 10us which will send an 8 cycle sonic burst at the speed of sound and after striking the object, it will be received by the Echo pin. Depending on the time taken by sound vibration to get back, it provides the appropriate pulse output. We can calculate the distance of the object based on the time taken by the ultrasonic wave to return back to the sensor.

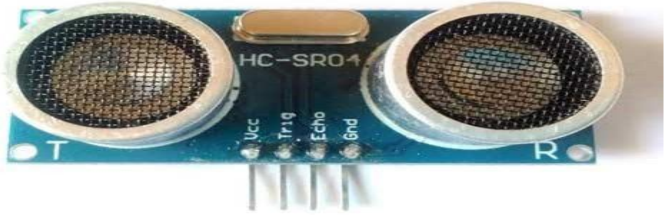


Fig 15.Utrasonic sensor

There are many applications with the ultrasonic sensor. We can make use of it avoid obstacles for the automated cars, moving robots etc. The same principle will be used in the RADAR for detecting the intruder missiles and airplanes. A mosquito can sense the ultrasonic sounds. So, ultrasonic waves can be used as mosquito repellent.

15.Touch Sensor

In this generation, we can say that almost all are using smartphones which have widescreen that too a screen which can sense our touch. So, let‘s see how this touchscreen works. Basically, there are two types of touch sensors *resistive based and a capacitive based touch screens*. Let‘s know about working of these sensors briefly.

The resistive touch screen has a resistive sheet at the base and a conductive sheet under the screen both of these are separated by an air gap with a small voltage applied to the sheets. When we press or touch the screen the conductive sheet touches the resistive sheet at that point causing current flow at that particular point, the software senses the location and relevant action is performed.

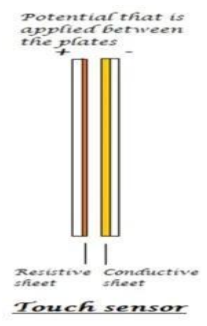


Fig 16.Touch sensor

16.PIR sensor

PIR sensor stands for **Passive Infrared sensor**. These are used to detect the motion of humans, animals or things. We know that infrared rays have a property of reflection. When an infrared ray hits an object, depending upon the temperature of the target the infrared ray properties changes, this received signal determines the motion of the objects or the living beings. Even if the shape of the object alters, the properties of the reflected infrared rays can differentiate the objects precisely. Here is the complete working or PIR sensor.

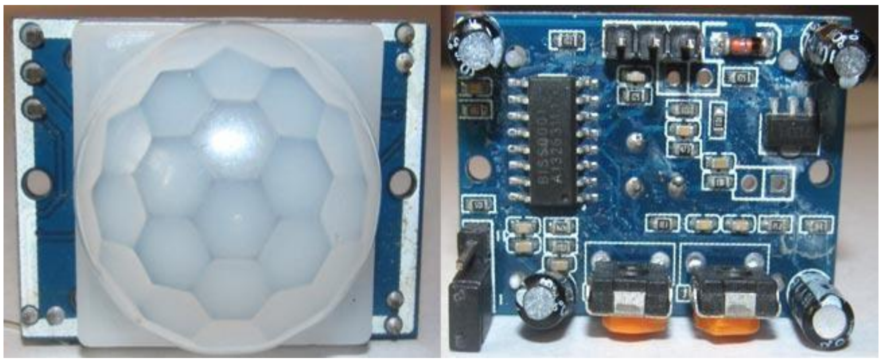
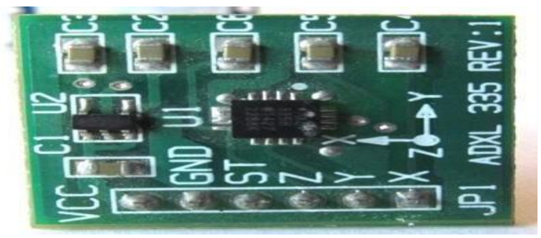


Fig 17.PIR Sensor

17.Accelerometer (Tilt Sensor)

An accelerometer sensor *can* sense the tilt or movement of it in a particular direction**.** It works based on the acceleration force caused due to the earth‘s gravity. The tiny internal parts of it are such sensitive that those will react to a small external change in position. It has a piezoelectric crystal when tilted causes disturbance in the crystal and generates



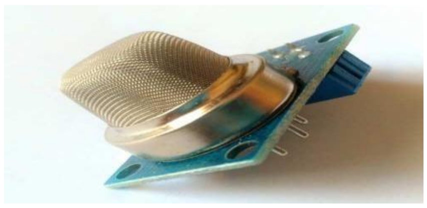
potential which determines the exact position with respect to X, Y and Z axis.

Fig 18.Accelerometer

These are commonly seen in mobiles and laptops in order to avoid breakage of processors leads. When the device falls the accelerometer detects the falling condition and does respective action based on the software.

18.Gas sensor

In industrial applications gas sensors plays a major role in **detecting the gas leakage**. If no such device is installed in such areas it ultimately leads to an unbelievable disaster. These gas sensors are classified into various types based on the type of gas that to be detected. Let‘s see how this sensor works. Underneath a metal sheet there exists a sensing element which is connected to the terminals where a current is applied to it. When the gas particles



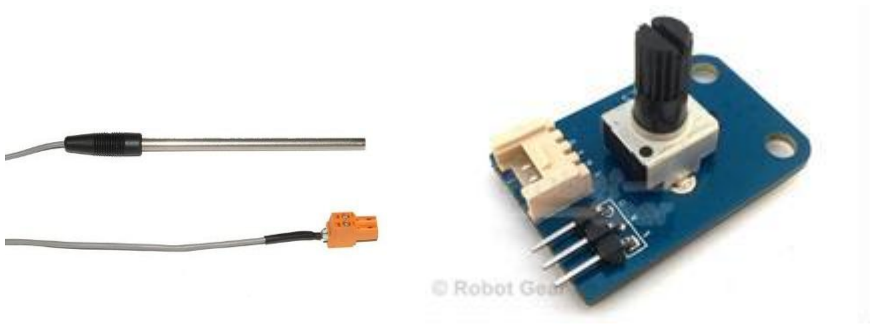
hit the sensing element, it leads to a chemical reaction such that the resistance of the elements varies and current through it also alters which finally can detect the gas.

Fig 19.Gas Sensor

So finally, we can conclude that sensors are not only used to make our work simple to measure the physical quantities, making the devices automated but also used to help living beings with disasters.

19. Resistive Sensors

Resistive sensors, such as the potentiometer, have three terminals: power input, grounding terminal, and variable voltage output. These mechanical devices have varied resistance that can be changed through movable contact with its fixed resistor. Output from the sensor varies depending on whether the movable contact is near the resistor's supple end or

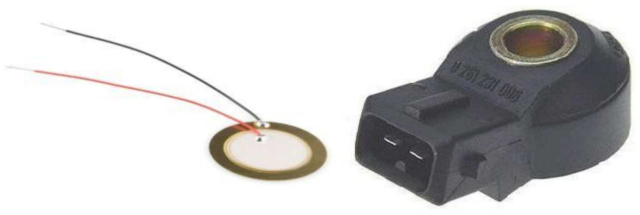


ground end. Thermistors are also variable resistors, although the resistance of the sensor varies with temperature

Fig 20 Resistive Sensors

20.Voltage generating sensors

Voltage-generating sensors, such as piezo electrics, generate electricity by pressure with types of crystals like quartz. As the crystal flexes or vibrates, AC voltage is produced. Knock sensors utilize this technology by sending a signal to an automobile's on-board computer that engine knock is happening. The signal is generated through crystal vibration within the sensor, which is caused by cylinder block vibration. The computer, in turn, reduces the ignition timing to stop the engine knock.



.

Fig 21.Voltage Generating Sensors

21.Switch Sensors

Switch sensors are composed of a set of contacts that open when close to a magnet. A reed switch is a common example of a switch sensor and is most commonly used as a speed or position sensor. As a speed sensor, a magnet is attached to the speedometer cable and spins along with it. Each time one of the magnet's poles passes the reed switch, it opens and then

closes. How fast the magnet passes allows the sensor to read the vehicle's speed.

Fig 22.Switch Sensors

2.Edge Networking

Embedded systems are already playing a crucial role in the development of the IoT. In broad strokes, there are four main components of an IoT system:

1. The Thing itself (the device)
2. The Local Network; this can include a gateway, which translates proprietary communication protocols to Internet Protocol
3. The Internet
4. Back-End Services; enterprise data systems, or PCs and mobile devices

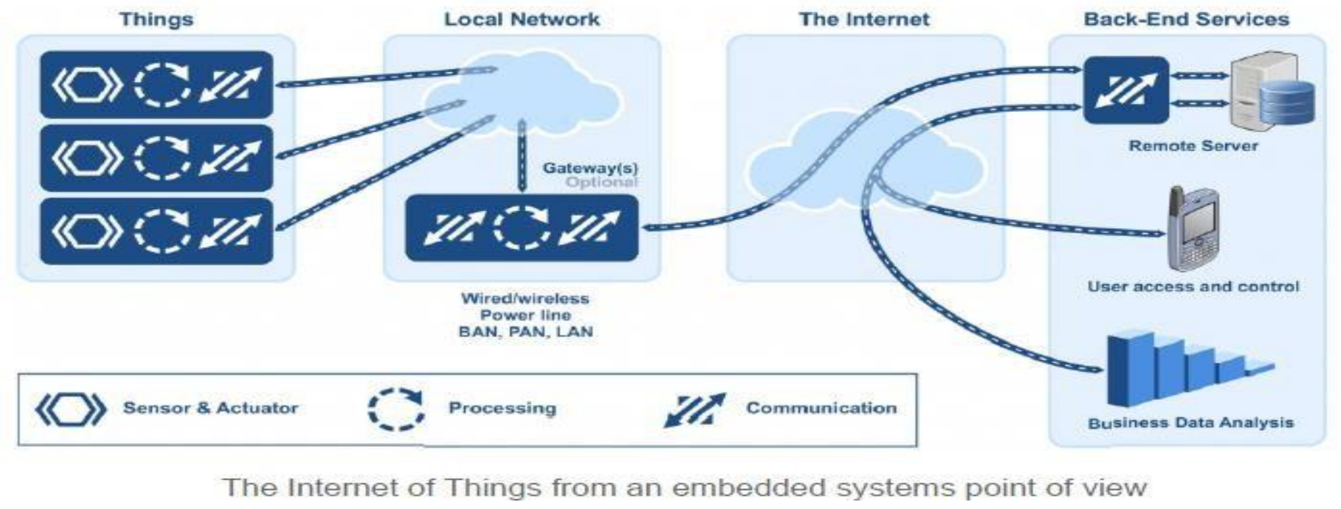


Fig 23.Embedded Point of View

We can also separate the Internet of Things in two broad categories:

.

1. Industrial IoT, where the local network is based on any one of many different technologies. The IoT device will typically be connected to an IP network to the global Internet.

2. Commercial IoT, where local communication is typically either Bluetooth or Ethernet (wired or wireless). The IoT device will typically communicate only with local devices.

So to better understand how to build IoT devices, you first need to figure out how they will communicate with the rest of the world.

Local Network
Your choice of communication technology directly affects your device's hardware requirements and costs. Which networking technology is the best choice?
IoT devices are deployed in so many different ways — in clothing, houses, buildings, campuses, factories, and even in your body — that no single networking technology can fit all bills.

Let's take a factory as a typical case for an IoT system. A factory would need a large number of connected sensors and actuators scattered over a wide area, and a wireless technology would be the best fit.

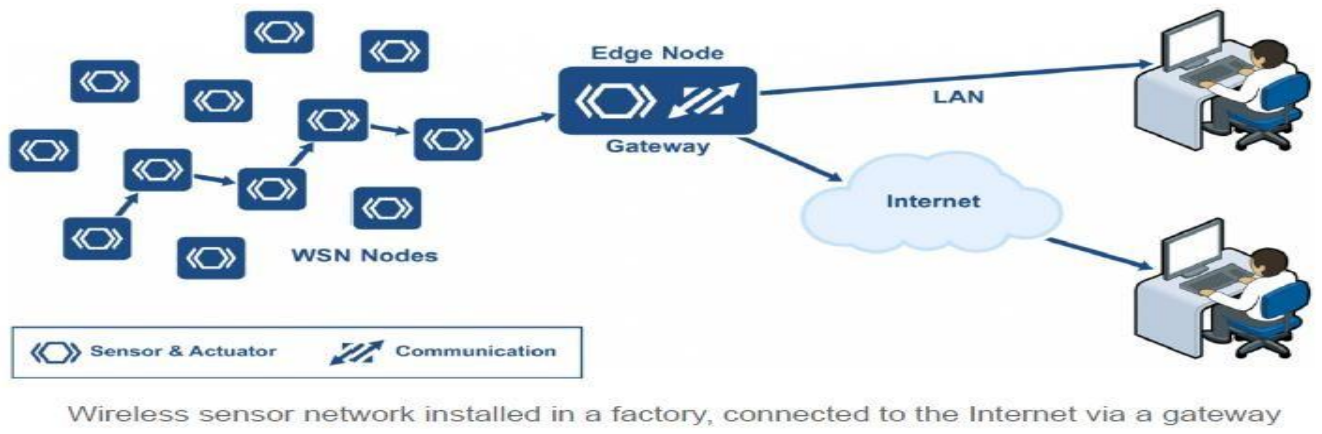

Edge Node
LAN
Gateway
Internet
WSN Nodes
Sensor & Actuator
Communication
Wireless sensor network installed in a factory, connected to the Internet via a gateway

Fig 24.Wireless Sensor Network Architecture

A wireless sensor network (WSN) is a collection of distributed sensors that monitor physical or environmental conditions, such as temperature, sound, and pressure. Data from each sensor passes through the network node-to-node.
WSN Nodes
WSN nodes are low cost devices, so they can be deployed in high volume. They also operate at low power so that they can run on battery, or even use energy harvesting. A WSN node is an embedded system that typically performs a single function (such as measuring temperature or pressure, or turning on a light or a motor).
Energy harvesting is a new technology that derives energy from external sources (for example, solar power, thermal energy, wind energy, electromagnetic radiation, kinetic energy, and more). The energy is captured and stored for use by small, low-power wireless autonomous devices, like the nodes on a WSN.

WSN Edge Nodes

A WSN edge node is a WSN node that includes Internet Protocol connectivity. It acts as a gateway between the WSN and the IP network. It can also perform local processing, provide local storage, and can have a user interface.

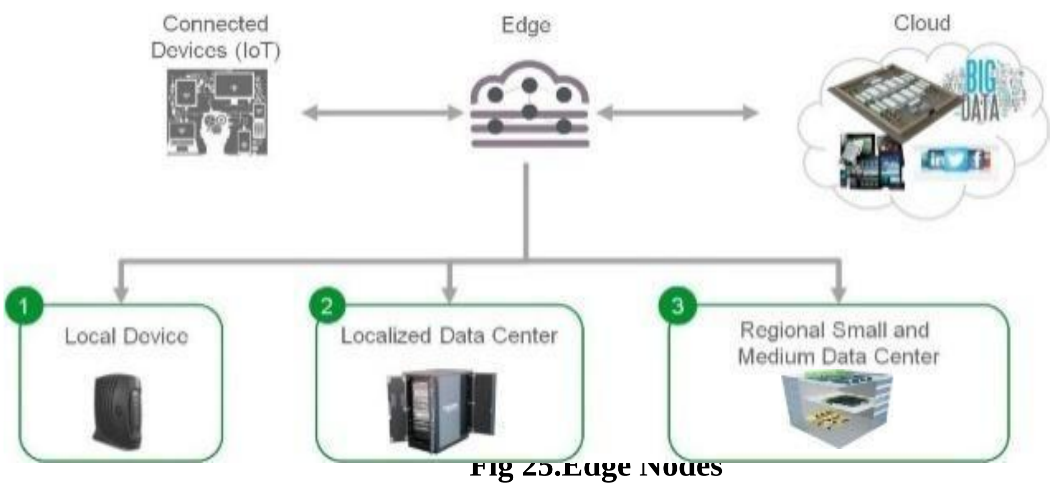


Fig 25.Edge Nodes

Fig 25.WSN Edge

WSN Technologies

The battle over the preferred networking protocol is far from over. There are multiple candidates.

Wi-Fi

The first obvious networking technology candidate for an IoT device is Wi-Fi, because it is so ubiquitous. Certainly, Wi-Fi can be a good solution for many applications. Almost every house that has an Internet connection has a Wi-Fi router. However, Wi-Fi needs a fair amount of power. There are myriad devices that can‘t afford that level of power: battery operated devices, for example, or sensors positioned in locations that are difficult to power from the grid.

New application protocols and data formats that enable autonomous operation For example, EnOceanhas patented an energy-harvesting wireless technology to meet the power consumption challenge. EnOcean‘s wireless transmitters work in the frequencies of 868 MHz for Europe and 315 MHz for North America. The transmission range is up to 30 meters in buildings and up to 300 meters outdoors.

EnOcean wireless technology uses a combination of energy harvesting and very low power wireless communications to enable virtually indefinite communications to be maintained without the need for recharging.

The EnOcean technology is used for wireless sensors, controllers and gateways.

One of the key issues with small machines is the need for ensuring that batteries are maintained charged. In traditional systems, either mains power was required, or batteries needed to be replaced, even if only infrequently. The use of EnOcean removes the need for power to be directly applied thereby reducing the cost of the system operation**.**

IEEE 802.15.4 Low-Rate Wireless Personal Area Networks (LR-WPANs)

One of the major IoT enablers is the IEEE 802.15.4 radio standard, released in 2003.Commercial radios meeting this standard provide the basis for low-power systems. This IEEE standard was extended and improved in 2006 and 2011 with the 15.4e and 15.4g amendments. Power consumption of commercial RF devices is now cut in half compared to only a few years ago, and we are expecting another 50% reduction with the next generation of devices.

6LoWPAN

Devices that take advantage of energy-harvesting must perform their tasks in the shortest time possible, which means that their transmitted messages must be as small as possible. This requirement has implications for protocol design.

3. Internet of Things Communications Models

From an operational perspective, it is useful to think about how IoT devices connect and communicate in terms of their technical communication models. In March 2015, the Internet Architecture Board (IAB) released a guiding architectural document for networking of smart objects which outlines a framework of four common communication models used by IoT devices. The discussion below presents this framework and explains key characteristics of each model in the framework.

Device-to-Device Communications

The device-to-device communication model represents two or more devices that directly connect and communicate between one another, rather than through an intermediary application server. These devices communicate over many types of networks, including IP networks or the Internet. Often, however these devices use protocols like Bluetooth, Z-Wave, or ZigBee to establish direct device-to-device communications, as shown in Figure 26.

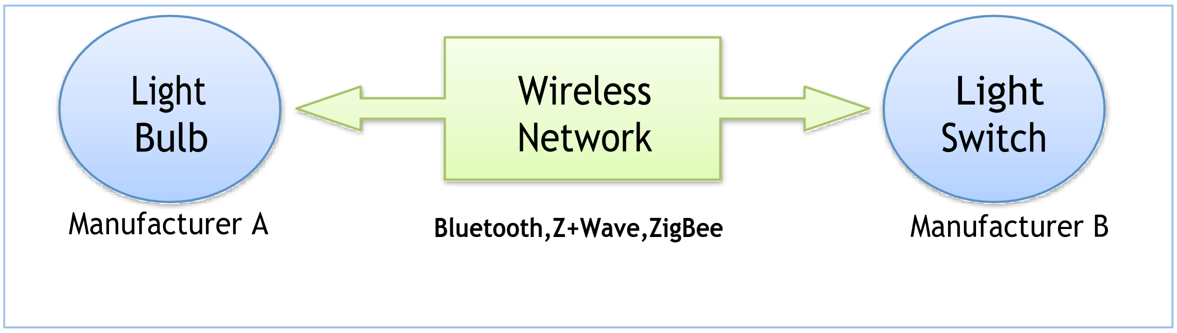


Fig 26.Example of device-to-device communication model

These device-to-device networks allow devices that adhere to a particular communication protocol to communicate and exchange messages to achieve their function. This communication model is commonly used in applications like home automation systems, which typically use small data packets of information to communicate between devices with relatively low data rate requirements. Residential IoT devices like light bulbs, light switches, thermostats, and door locks normally send small amounts of information to each other (e.g. a door lock status message or turn on light command) in a home automation scenario.

From the user‘s point of view, this often means that underlying device-to-device communication2 protocols are not compatible, forcing the user to select a family of devices that employ a common protocol. For example, the family of devices using the Z-Wave protocol is not natively compatible with the ZigBee family of devices. While these incompatibilities limit user choice to devices within a particular protocol family, the user benefits from knowing that products within a particular family tend to communicate well.

Device-to-Cloud Communications

In a device-to-cloud communication model, the IoT device connects directly to an Internet cloud service like an application service provider to exchange data and control message traffic. This approach frequently takes advantage of existing communications mechanisms like traditional wired Ethernet or Wi-Fi connections to establish a connection between the device and the IP network, which ultimately connects to the cloud service. This is shown in Figure 27.

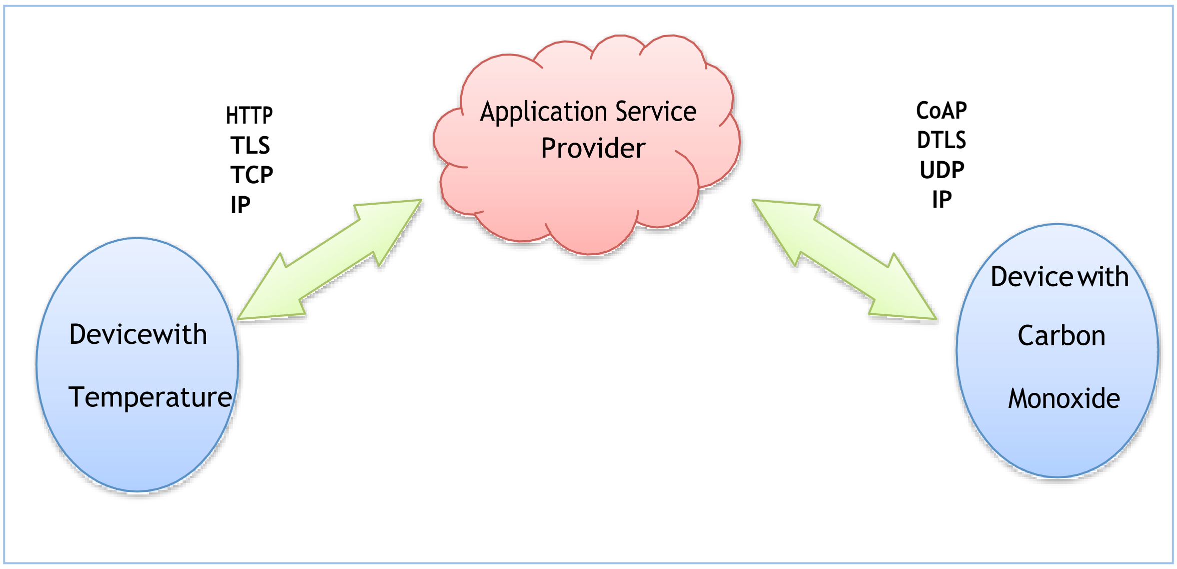


Fig 27.Device-to-cloud communication model diagram.

This communication model is employed by some popular consumer IoT devices like the Nest Labs Learning Thermostat and the Samsung SmartTV. In the case of the Nest Learning Thermostat, the device transmits data to a cloud database where the data can be used to analyze home energy consumption.
Further, this cloud connection enables the user to obtain remote access to their thermostat via a smartphone or Web interface, and it also supports software updates to the thermostat. Similarly, with the Samsung SmartTVtechnology, the television uses an Internet connection to transmit user viewing information to Samsung for analysis and to enable the interactive voice recognition features of the TV. In these cases, the device-to-cloud model adds value to the end user by extending the capabilities of the device beyond its native features.

However, interoperability challenges can arise when attempting to integrate devices made by different manufacturers. Frequently, the device and cloud service are from the same vendor. If proprietary data protocols are used between the device and the cloud service, the device owner or user may be tied to a specific cloud service, limiting or preventing the use of alternative service providers. This is commonly referred to as ―vendor lock-in‘‘, a term that encompasses other facets of the relationship with the provider such as ownership of and access to the data. At the same time, users can generally have confidence that devices designed for the specific platform can be integrated.

Device-to-Gateway Model

In the device-to-gateway model, or more typically, the device-to-application-layer gateway (ALG) model, the IoT device connects through an ALG service as a conduit to reach a cloud service. In simpler terms, this means that there is application software operating on a local gateway device, which acts as an intermediary between the device and the cloud service and provides security and other functionality such as data or protocol translation.

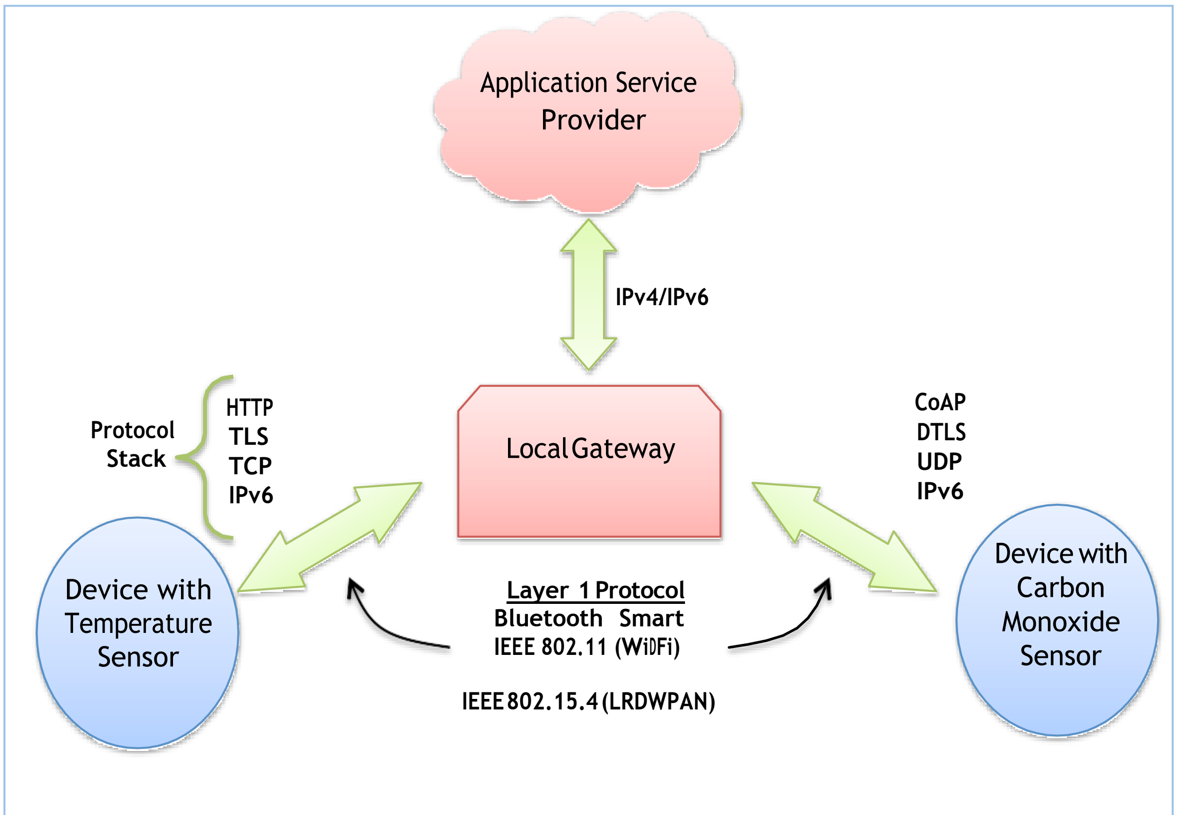


Fig 28.Device-to-gateway communication model diagram.

Several forms of this model are found in consumer devices. In many cases, the local gateway device is a Smartphone running an app to communicate with a device and relay data to a cloud service. model employed with popular consumer items like personal fitness trackers. These devices do not have the native ability to connect directly to a cloud service, so they frequently rely on Smartphone app software to serve as an intermediary gateway to connect the fitness device to the cloud.

The other form of this device-to-gateway model is the emergence of –hub‖ devices in home automation applications. These are devices that serve as a local gateway between individual IoT devices and a cloud service, but they can also bridge the interoperability gap between devices themselves. For example, the Smart Things hub is a stand-alone gateway device that has Z-Wave and Zigbee transceivers installed to communicate with both families of devices. It then connects to the Smart Things cloud service, allowing the user to gain access to the devices using a Smartphone app and an Internet connection. The evolution of systems using the device-to-gateway communication model and its larger role in addressing interoperability challenges among IoT devices is still unfolding.

Back-End Data-Sharing Model

The back-end data-sharing model refers to a communication architecture that enables users to export and analyze smart object data from a cloud service in combination with data from other sources. This architecture supports –the [user‘s] desire for granting access to the uploaded sensor data to third parties. This approach is an extension of the single device-to-cloud communication model, which can lead to data silos where –IoT devices upload data only to a single application

service provider‘‘. A back-end sharing architecture allows the data collected from single IoT device data streams to be aggregated and analyzed.

For example, a corporate user in charge of an office complex would be interested in consolidating and analyzing the energy consumption and utilities data produced by all the IoT sensors and Internet-enabled utility systems on the premises. Often in the single device-to-cloud model, the data each IoT sensor or system produces sits in a stand-alone data silo. An effective back-end data sharing architecture would allow the company to easily access and analyze the data in the cloud produced by the whole spectrum of devices in the building. Also, this kind of architecture facilitates data portability needs. Effective back-end data- sharing architectures allow users to move their data when they switch between IoT services, breaking down traditional data silo barriers.

The back-end data-sharing model suggests a federated cloud services approach or cloud applications programmer interfaces (APIs) are needed to achieve interoperability of smart device data hosted in the cloud. A graphical representation of this design is shown in Fig 29.

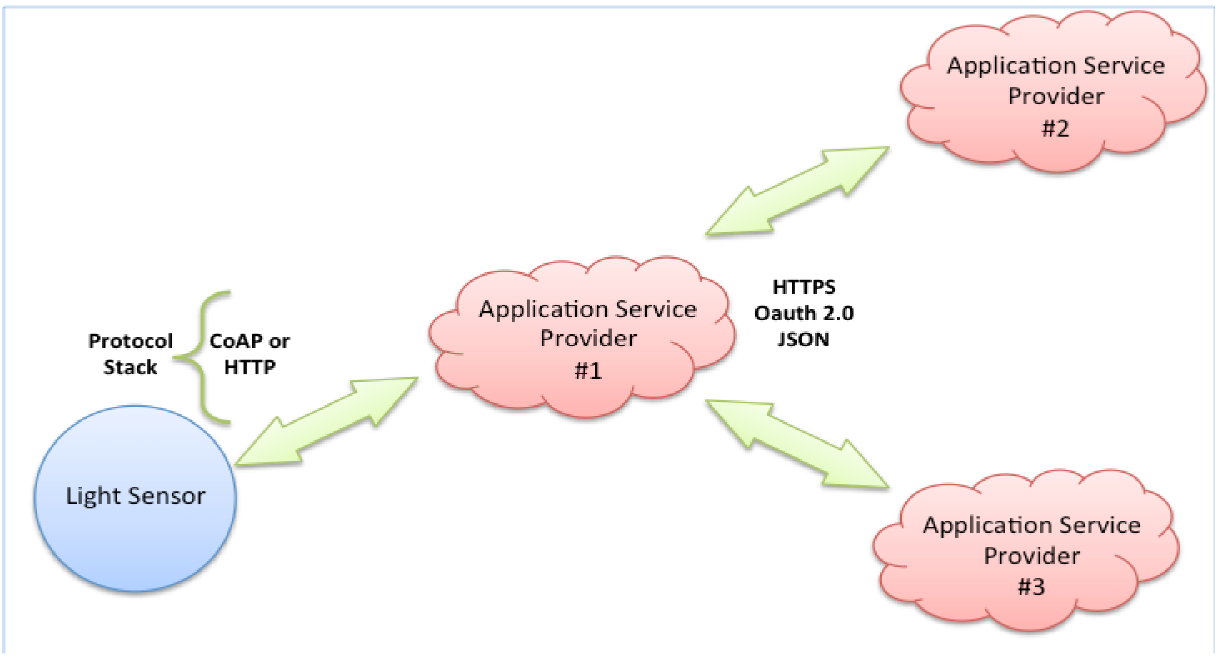

Application Service Provider #2
Protocol Stack
CoAP or HTTP
Application Service Provider #1
HTTPS
Oauth 2.0
JSON
Light Sensor
Application Service Provider #3

Fig 29.Back-end data sharing model

diagram Internet of Things Communications Models Summary

The four basic communication models demonstrate the underlying design strategies used to allow IoT devices to communicate. Aside from some technical considerations, the use of these models is largely influenced by the open versus proprietary nature of the IoT devices being networked. And in the case of the device-to-gateway model, its primary feature is its ability to overcome proprietary device restrictions in connecting IoT devices. This means that device interoperability and open standards are key considerations in the design and development of internetworked IoT systems.

From a general user perspective, these communication models help illustrate the ability of networked devices to add value to the end user. By enabling the user to achieve better access to an IoT device and its data, the overall value of the device is amplified. For example, in three of the four communication models, the devices ultimately connect to data analytic services in a cloud computing setting. By creating data communication conduits to the cloud, users, and service providers can more readily employ data aggregation, big data analytics, data visualization, and predictive analytics technologies to get more value out of IoT data than can be achieved in traditional data-silo applications. In other words, effective communication architectures are an important driver of value to the end user by opening possibilities of using information in new ways. It should be noted, however, these networked benefits come with trade-offs. Careful consideration needs to be paid to the incurred cost burdens placed on users to connect to cloud resources when

4. Low Power Wide Area Networks: An Overview

Low Power Wide Area (LPWA) networks represent a novel communication paradigm, which will complement traditional cellular and short range wireless technologies in addressing diverse requirements of IoT applications. LPWA technologies offer unique sets of features including wide- area connectivity for low power and low data rate devices, not provided by legacy wireless technologies.

LPWA networks are unique because they make different tradeoffs than the traditional technologies prevalent in IoT landscape such as short-range wireless networks e.g., Zig- Bee, Bluetooth, Z-Wave, legacy wireless local area networks (WLANs) e.g., Wi-Fi, and cellular networks e.g. Global Sys- tem for Mobile Communications (GSM), Long-Term Evolution (LTE) etc. The legacy non-cellular wireless technologies are not ideal to connect low power devices distributed over large geographical areas. The range of these technologies is limited to a few hundred meters at best. The devices, therefore, cannot be arbitrarily deployed or moved *anywhere*, a requirement for many applications for smart city, logistics and personal health The range of these technologies is extended using a dense deployment of devices and gateways connected using multihop mesh networking. Large deployments are thus prohibitively expensive. Legacy WLANs, on the other hand, are characterized by shorter coverage areas and higher power consumption for machine-type communication (MTC).

A wide area coverage is provided by cellular networks, a reason of a wide adoption of second generation (2G) and third generation (3G) technologies for M2M communication. How- ever, an impending decommissioning of these technologies[5], as announced by some mobile network operators (MNOs),will broaden the technology gap in connecting low-power devices. In general, traditional cellular technologies do not achieve energy efficiency high enough to offer ten years of battery lifetime. The complexity and cost of cellular devices is high due to their ability to deal with complex waveforms, optimized for voice, high speed data services, and text. For low-power MTC, there is a clear need to strip complexity to reduce cost. Efforts in this direction are underway for cellular networks by the Third Generation Partnership Project and are covered as

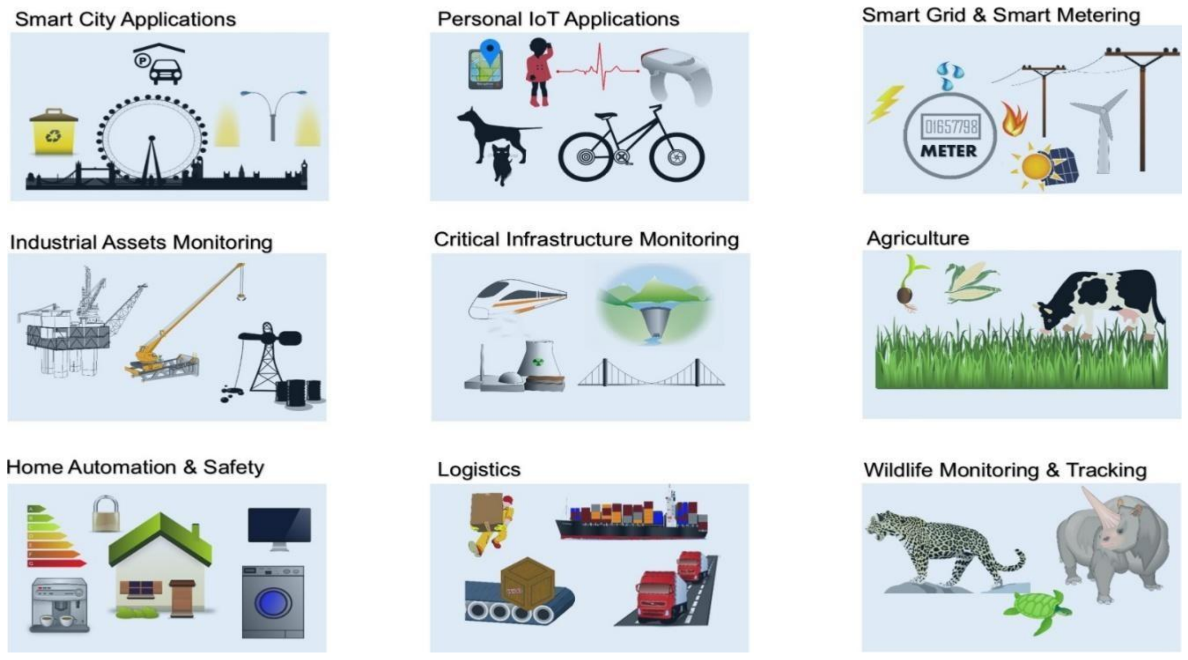

Smart City Applications
Personal IoT Applications
Smart Grid & Smart Metering
01657798
METER
Industrial Assets Monitoring
Critical Infrastructure Monitoring
Agriculture
Home Automation & Safety
Logistics
Wildlife Monitoring & Tracking

Fig 30.Applications of LPWA technologies across different sectors

Key Objective Of LPWA Technologies

A. Long range

LPWA technologies are designed for a wide area coverage and an excellent signal propagation to hard-to-reach indoor places such as basements. The physical layer compromises on high data rate and slows downs the modulation rate to put more energy in each transmitted bit (or symbol). Due to this reason, the receivers can decode severely attenuated signals correctly. Typical sensitivity of state of the art LPWA receivers reaches as low as -130 dBm.

B. Ultra low power operation

Ulra-low power operation is a key requirement to tap into the huge business opportunity provided by battery-powered IoT/M2M devices. A battery lifetime of 10 years or more with AA or coin cell batteries is desirable to bring the maintenance cost down.

C. Topology

While mesh topology has been extensively used to extend the coverage of short range wireless networks, their high deployment cost is a major disadvantage in con- necting large number of geographically distributed devices. Further, as the traffic is forwarded over multiple hops towards a gateway, some nodes get more congested than others depend- ing on their location or network traffic patterns. Therefore, they deplete their batteries quickly, limiting overall network lifetime to only a few months to years .On the other hand, a very long range of LPWA technologies overcomes these limitations by connecting end devices directly to base stations, obviating the need for the dense and expensive deployments of relays and gateways altogether. The resulting topology is a star that is used extensively in cellular networks and brings huge energy saving advantages. As opposed to the mesh topology, the devices need not to waste precious energy in busy-listening to other devices that want to relay their traffic through them. An always-on base station provides convenient and quick access when required by the end-devices.

D. **Duty Cycling**: Low power operation is achieved by opportunistically turning off power hungry components of M2M/IoT devices e.g., data transceiver. Radio duty cycling allows LPWA end devices to turn off their transceivers, when not required. Only when the data is to be transmitted or received, the transceiver is turned on.

E. **Lightweight Medium Access Control**: Most-widely used Medium Access Control (MAC) rotocols for cellular net- works or short range wireless networks are too complex for LPWA technologies. For example, cellular networks synchro- nize the base stations and the user equipment (UE) accurately to benefit from complex MAC schemes that exploit frequency.

CHALLENGES AND OPEN RESEARCH DIRECTIONS LPWA

On the business side, the proprietary solution providers are in a rush to bring their services to the market and capture their share across multiple verticals. In this race, it is easy but counter- productive to overlook important challenges faced by LPWA technologies. In this section, we highlight these challenges and some research directions to overcome them and improve performance in long-term.

1. Scaling networks to massive number of devices

LPWA technologies will connect tens of millions of devices transmitting data at an unprecedented scale over limited and often shared radio resources. This complex resource allocation problem is further complicated by several other factors. First, the device density may vary significantly across different geographical areas, creating the so called *hot-spot* problem. These hot-spots will put the LPWA base stations to a stress test. Second, cross-technology interference can severely de- grade the performance of LPWA technologies.

2. Interoperability between different LPWA technologies

   Given that market is heading towards an intense competition between different LPWA technologies, it is safe to assume that several may coexist in future. Interoperability between these heterogeneous technologies is thus crucial to their long- term profitability. With little to no support for interoperability between different technologies, a need for standards that glue them together is strong. Interoperability is a still an open challenge. Test beds and open-source tool chains for LPWA technologies are not yet widely available to evaluate interoperability mechanisms.

3. Localization

   LPWA networks expect to generate significant revenue from logistics, supply chain management, and personal IoT applications, where location of mobile objects, vehicles, humans, and animals may be of utmost interest. An accurate localization support is thus an important feature for keeping track of valuables, kids, elderly, pets, shipments, vehicle fleets, etc. In fact, it is regarded as an important feature to enable new applications.

4. Link optimizations and adaptability

   If a LPWA technology permits, each individual link should be optimized for high link quality and low energy consumption to maximize overall network capacity. Every LPWA technology allows multiple link level configurations that introduce tradeoffs between different performance metrics such as data rate, time-on-air, area coverage, etc. This motivates a need for adaptive techniques that can monitor link quality and then read just its parameters for better performance. However for such techniques to work, a feedback from gateway to end devices is usually required over down link.

5. LPWA test beds and tools

   LPWA technologies enable several smart city applications. A few smart city test beds e.g. Smart Santander have emerged in recent years. Such test beds incorporate sensors equipped with different wireless technologies such as Wi-Fi, IEEE 802.15.4 based networks and cellular networks. How- ever, there are so far no open test beds for LPWA networks. Therefore, it is not cost-effective to widely design LPWA systems and compare their performance at a metropolitan scale. At the time of writing, only a handful of empirical studies compare two our more LPWA technologies under same conditions. In our opinion, it is a significant barrier to entry for potential customers. Providing LPWA technologies as a scientific instrumentation for general public through city governments can act as a confidence building measure.

6. Authentication, Security, and Privacy

Authentication, security, and privacy are some of the most important features of any communication system. Cellular networks provide proven authentication, security, and privacy mechanisms. Use of Subscriber Identity Modules (SIM) simplifies identification and authentication of the cellular devices. LPWA technologies, due to their cost and energy considerations, not only settle for simpler communication protocols but also depart from SIM based authentication. Techniques and protocols are thus required to provide equivalent or better authentication support for LPWA technologies. Further to assure that end devices are not exposed to any security risks over prolonged duration, a support for over-the-air (OTA) updates is a crucial feature. A lack of adequate support for OTA updates poses a great security risk to most LPWA technologies.

7. Mobility and Roaming

Roaming of devices between different network operators is a vital feature responsible for the commercial success of cellular networks. Whilst some LPWA technologies do not have the notion of roaming (work on a global scale such as SIGFOX), there are others that do not have support for roaming as of the time of this writing. The major challenge is to provide roaming without compromising the lifetime of the devices. To this effect, the roaming support should put minimal burden on the battery powered end-devices. Because the end-devices duty cycle aggressively, it is reasonable to assume that the low power devices cannot receive downlink traffic at all times. Data exchanges over the uplink should be exploited more aggressively. Network assignment is to be resolved in backend systems as opposed to the access network. All the issues related to agility of roaming process and efficient resource management have to bead dressed.

5. Wireless Personal Area Network (WPAN)

WPANs are used to convey information over short distances among a private, intimate group of participant devices. Unlike a WLAN, a connection made through a WPAN involves little or no infrastructure or direct connectivity to the world outside the link. This allows small, power-efficient, inexpensive solutions to be implemented for a wide range of device.

Applications

- Short-range (< 10 m) connectivity for multimedia applications
- PDAs, cameras, voice (hands free devices)
- High QoS, high data rate (IEEE 802.15.3)
- Industrial sensor applications
- Low speed, low battery, low cost sensor networks (IEEE 802.15.4)
- Common goals
- Getting rid of cable connections
- Little or no infrastructure
- Device interoperability

WPAN Topologies:

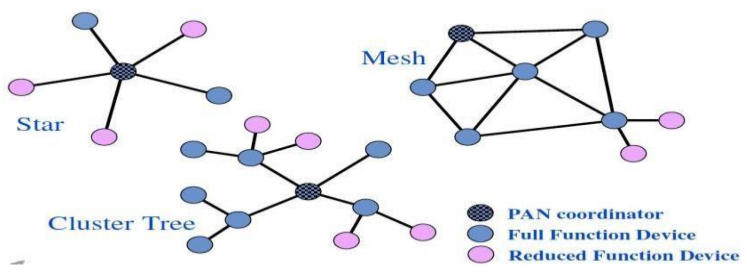


Fig 31 WPAN Topologies

IEEE 802.15 WPAN Standards:

1. IEEE 802.15.2- Co existence of Bluetooth and 802.11b
2. IEEE 802.15.3- High Rate WPAN
   Low power and low cost applications for digital imaging and multimedia applications.
3. IEEE 802.15.4- Low Rate WPAN
   Industrial ,Medical and agriculture applications.

Bluetooth ≈ IEEE 802.15.1

A widely used WPAN technology is known as Bluetooth (version 1.2 or version 2.0). The IEEE
standard specifies the architecture and operation of Bluetooth devices, but only as

far as physical layer and medium access control (MAC) layer operation is concerned (the core system architecture)

.Higher protocol layers and applications defined in usage profiles are standardized by the Bluetooth SIG. Bluetooth is the base for IEEE Std 802.15.1-2002 (rev. 2005).Data rate of 1 Mbps (2 or 3 Mbps with enhanced data rate).

Piconets

- Bluetooth enabled electronic devices connect and communicate wirelessly through short-range, ad hoc networks known as piconets. Piconets are established dynamically and automatically as Bluetooth enabled devices enter and leave radio proximity. Up to 8 devices in one piconet (1 master and up to 7 slave devices)
- Max range is 10 m. The **piconet master** is a device in a piconet whose clock and device address are used to define the piconet physical channel characteristics.All other devices in the piconet are called **piconet slaves.**All devices have the same timing and frequency hopping sequence. At any given time, data can be transferred between the master and one slave.
- The master switches rapidly from slave to slave in a round-robin fashion. Any Bluetooth device can be either a master or a slave. Any device may switch the master/slave role at any time.

Scatternet

Any Bluetooth device can be a master of one piconet and a slave of another piconet at the same time (scatternet). Scatternet is formed by two ormore Piconets. Master of one piconet can participate as a slave in another connected piconet. No time or frequency synchronization between piconets

Bluetooth Protocol
Stack Radio Layer

The radio layer specifies details of the air interface, including the usage of the frequency hopping sequence, modulation scheme, and transmit power. The radio layer FHSS operation and radio parameters

Baseband Layer

The baseband layer specifies the lower level operations at the bit and packet levels. It supports Forward Error Correction (FEC) operations and Encryption, Cyclic Redundancy Check (CRC) calculations. Retransmissions using the Automatic Repeat Request (ARQ) Protocol.

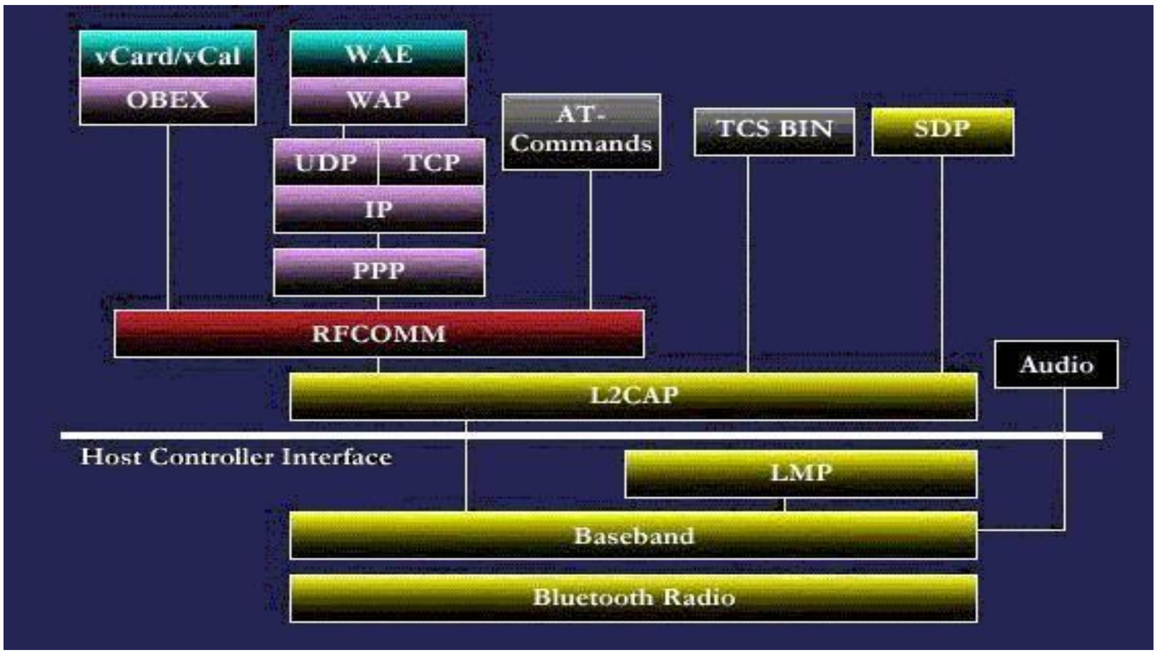


Fig 32 Bluetooth Protocol Stack

Link Manager layer

The link manager layer specifies the establishment and release links, authentication, traffic scheduling, link supervision, and power management tasks. Responsible for all the physical link resources in the system. Handles the control and negotiation of packet sizes used when transmitting data.Sets up, terminates, and manages baseband connections between devices.

L2CAP layer

The Logical Link Control and Adaptation Protocol (L2CAP) layer handles the multiplexing of higher layer protocols and the segmentation and reassembly (SAR) of large packets The L2CAP layer provides both connectionless and connection-oriented services

L2CAP performs 4 major functions

Managing the creation and termination of logical links for each connection through **channel** structures. Adapting Data, for each connection, between application (APIs) and Bluetooth Baseband formats through Segmentation and Reassembly (SAR). Performing Multiplexing to support multiple concurrent connections over a single common radio interface (multiple apps. using link between two devices simultaneously). L2CAP segments large packets into smaller baseband manageable packets. Smaller received baseband packets are reassembled coming back up the protocol stack.

RFCOMM

Applications may access L2CAP through different support protocols Service Discovery Protocol (SDP) RFCOMM Telephony Control Protocol Specification (TCS) TCP/IP based applications, for instance information transfer using the Wireless Application Protocol (WAP), can be extended to Bluetooth devices by using the Point-to-Point Protocol (PPP) on top of RFCOMM.

OBEX Protocol

The Object Exchange Protocol (OBEX) is a sessionlevel protocol for the exchange of objects This protocol can be used for example for phonebook, calendar or messaging synchronization, or for file transfer between connected devices.

TCSBIN Protocol

The telephony control specification - binary (TCS BIN) protocol defines the call-control signaling for the establishment of speech and data calls between Bluetooth devices In addition, it defines mobility management procedures for handling groups of Bluetooth devices.

Service Discovery Protocol

The Service Discovery Protocol (SDP) can be used to access a specific device (such as a digital camera) and retrieve its capabilities, or to access a specific application (such as a print job) and find devices that support this application.

6. Smart Wi-Fi module

Smart Wi-Fi is an IoT-enabler tool. The applications it can cater to are only limited by the imagination of makers. The very basic applications could be for smart homes or smart offices. This module can be used for data logging, data monitoring and more, and provides very good support for product development. It also has all features to act as a full-fledged product. Platforms such as ThingSpeak add to the benefits and provide support for testing and development of an IoT product. Smart Wi-Fi enables making a product quickly and reliably. With open software resources and hardware data, moving to the final product after the proof of concept is also easy.

7. IoT platform

The purpose of any **IoT device** is to connect with other IoT devices and applications (cloud-based mostly) to relay information using internet transfer protocols.

The gap between the device sensors and data networks is filled by an **IoT Platform**. Such a platform connects the data network to the sensor arrangement and provides insights using backend applications to make sense of plethora of data generated by hundreds of sensors.

While there are hundreds of companies and a few startups venturing into IoT platform development, players like Amazon and Microsoft are way ahead of others in the competition. Read on to know about top 10 IoT platforms you can use for your applications.

IoT platform: Amazon Web Services (AWS) IoT

Last year Amazon announced the AWS IoT platform at it s Re:Invent conference. Main features of AWS IoT platform are:

- Registry for recognizing devices
- Software Development Kit for devices
- Device Shadows
- Secure Device Gateway
- Rules engine for inbound message evaluation

According to Amazon, their **IoT platform** will make it a lot easier for developers to connect sensors for multiple applications ranging from automobiles to turbines to smart home light bulbs.
Taking the scope of AWS IoT to the next level, the vendor has partnered with hardware manufacturers like Intel, Texas Instruments, Broadcom and Qualcomm to create starter kits compatible with their platform.

IoT Platform: Microsoft Azure IoT

Microsoft is very much interested in bringing up products for internet of things. For the initiative the **Microsoft Azure cloud services** compatible IoT platform, **the Azure IoT suite** is on the offer. Features included in this platform are:

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

For processing the massive amount of information generated by sensors Azure IoT suite comes with Azure Stream Analytics to process massive amounts of information in real-time.

Top IoT Platforms-Google Cloud Platform (New)

Google can make things happen. With its end-to-end platform, Google cloud is among the **best IoT platforms** we have currently. With the ability to handle the vast amount of data using Cloud IoT Core,

Google stands out from the rest. You get advanced analytics owing to Google‘s Big Query and Cloud Data Studio.
Some of the features of the Google Cloud platform are:-

- Accelerate your Business
- Speed up your devices
- Cut Cost with Cloud Service.
- Partner Ecosystem

IoT Platform: ThingWorx IoT Platform

In vendor‘s own words, –ThingWorx is the industry‘s leading Internet of Things (IoT) technology platform. It enables innovators to rapidly create and deploy game-changing applications, solutions and experiences for today‘s smart, connected world.*"*
Thingsworx is an IoT platform which is designed for enterprise application development. It offers features like:

- Easy connectivity of devices to the platform
- Remove complexity from IoT application development
- Sharing platform among developers for rapid development
- Integrated machine learning for automating complex big data analytics
- Deploy cloud, embedded or on-premise IoT solutions on the go

IoT platform: IBM Watson

We can never expect the *Big Blue* to miss on the opportunity to making a mark in the internet of things segment. IBM Watson is an IoT platform which is pretty much taken among developers already. Backed by IBM‘s hybrid cloud PaaS (platform as a service) development platform, the Bluemix, Watson IoT enables developers to easily deploy IoT applications.
Users of IBM Watson get:

- Device Management
- Secure Communications
- Real Time Data Exchange
- Data Storage
- Recently added data sensor and weather data service

Top IoT Platforms-Artik
(New):samsung IoT Platform:
Cisco IoT Cloud Connect
Top IoT Platforms-Universal of Things (IoT) Platform
(New) :HP Top IoT Platforms-Datav by Bsquare (New)
IoT platform: Salesforce IoT Cloud
Top IoT Platforms-Mindsphere by

Siemens (New) Top IoT Platforms-Ayla
Network by Ayla (New) Top IoT
Platforms-Bosch IoT Suite(New)
IoT Platform: Carriots
IoT Platform: Oracle Integrated
Cloud IoT Platform: General
Electric‘s Predix
Top IoT Platforms-MBED IoT Device platform(New)

Top IoT Platforms-Mosaic (LTI)
(New) IoT Platform: Kaa
AWS IoT Core is a managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices. AWS IoT Core can support billions of devices and trillions of messages, and can process and route those messages to AWS endpoints and to other devices reliably and securely. With AWS IoT Core, your applications can keep track of and communicate with all your devices, all the time, even when they aren‘t connected.
AWS IoT Core makes it easy to use AWS services like AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch, AWS CloudTrail, and Amazon Elasticsearch Service with built-in Kibana integration, to build IoT applications that gather, process, analyze and act on data generated by connected devices, without having to manage any infrastructure.

AWS IoT provides secure, bi-directional communication between Internet-connected devices such as sensors, actuators, embedded micro-controllers, or smart appliances and the AWS Cloud. This enables you to collect telemetry data from multiple devices, and store and analyze the data. You can also create applications that enable your users to control these devices from their phones or tablets.

AWS IoT Components

AWS IoT consists of the following components:

A. Device gateway

Enables devices to securely and efficiently communicate with AWS IoT.

B. Message broker

Provides a secure mechanism for devices and AWS IoT applications to publish and receive messages from each other. You can use either the MQTT protocol directly or MQTT over WebSocket to publish and subscribe. You can use the HTTP REST interface to publish.

C. Rules engine

Provides message processing and integration with other AWS services. You can use an SQL-based language to select data from message payloads, and then process and send the data to other services, such as Amazon S3, Amazon DynamoDB, and AWS Lambda. You can also use the message broker to republish messages to other subscribers.

D. Security and Identity service:

Provides shared responsibility for security in the AWS Cloud. Your devices must keep their credentials safe in order to securely send data to the message broker. The message broker and rules engine use AWS security features to send data securely to devices or other AWS services.

E. Registry

Organizes the resources associated with each device in the AWS Cloud. You register your devices and associate up to three custom attributes with each one. You can also associate certificates and MQTT client IDs with each device to improve your ability to manage and troubleshoot them.

F. Group registry

Groups allow you to manage several devices at once by categorizing them into groups. Groups can also contain groups—you can build a hierarchy of groups. Any action you perform on a parent group will apply to its child groups, and to all the devices in it and in all of its child groups as well. Permissions given to a group will apply to all devices in the group and in all of its child groups.

G. Device shadow

A JSON document used to store and retrieve current state information for a device.

H. Device Shadow service

Provides persistent representations of your devices in the AWS Cloud. You can publish updated state information to a device's shadow, and your device can synchronize its state when it connects. Your devices can also publish their current state to a shadow for use by applications or other devices.

I. Device Provisioning service

Allows you to provision devices using a template that describes the resources required for your device: a *thing*, a certificate, and one or more policies. A thing is an entry in the registry that contains attributes that describe a device. Devices use certificates to authenticate with AWS IoT. Policies determine which operations a device can perform in AWS IoT.

The templates contain variables that are replaced by values in a dictionary (map). You can use the same template to provision multiple devices just by passing in different values for the template variables in the dictionary.

J. Custom Authentication service

You can define custom authorizers that allow you to manage your own authentication and authorization strategy using a custom authentication service and a Lambda function. Custom authorizers allow AWS IoT to authenticate your devices and authorize operations using bearer token authentication and authorization strategies. Custom authorizers can implement various authentication strategies and must return policy documents which are used by the device gateway to authorize MQTT operations.

K. Jobs Service

Allows you to define a set of remote operations that are sent to and executed on one or more devices connected to AWS IoT. For example, you can define a job that instructs a set of devices to download and install application or firmware updates, reboot, rotate certificates, or perform remote troubleshooting operations.To create a job, you specify a description of the remote operations to be performed and a list of targets that should perform them. The targets can be individual devices, groups or both.

Accessing AWS IoT

AWS IoT provides the following interfaces to create and interact with your devices:

- **AWS Command Line Interface (AWS CLI)**—Run commands for AWS IoT on Windows, macOS, and Linux. These commands allow you to create and manage things, certificates, rules, and policies. To get started, see the AWS Command Line Interface User Guide. For more information about the commands for AWS IoT, see iot in the AWS CLI Command Reference.

- **AWS IoT API**—Build your IoT applications using HTTP or HTTPS requests. These API actions allow you to programmatically create and manage things, certificates, rules, and policies. For more information about the API actions for AWS IoT, see Actions in the AWS IoT API Reference.
- **AWS SDKs**—Build your IoT applications using language-specific APIs. These SDKs wrap the HTTP/HTTPS API and allow you to program in any of the supported languages.
- **AWS IoT Device SDKs**—Build applications that run on devices that send messages to and receive messages from AWS IoT.

Related Services

AWS IoT integrates directly with the following AWS services:

- **Amazon Simple Storage Service**—Provides scalable storage in the AWS Cloud. For more information, see Amazon S3.
- **Amazon DynamoDB**—Provides managed NoSQL databases. For more information, see Amazon DynamoDB.
- **Amazon Kinesis**—Enables real-time processing of streaming data at a massive scale. For more information, see Amazon Kinesis.
- **AWS Lambda**—Runs your code on virtual servers from Amazon EC2 in response to events. For more information, see AWS Lambda.
- **Amazon Simple Notification Service**—Sends or receives notifications. For more information, see Amazon SNS.
- **Amazon Simple Queue Service**—Stores data in a queue to be retrieved by applications. For more information, see Amazon SQS.

Benefits

Connect and Manage Your Devices

AWS IoT Core allows you to easily connect devices to the cloud and to other devices. AWS IoT Core supports HTTP, WebSockets, and MQTT, a lightweight communication protocol specifically designed to tolerate intermittent connections, minimize the code footprint on devices, and reduce network bandwidth requirements. AWS IoT Core also supports other industry-standard and custom protocols, and devices can communicate with each other even if they are using different protocols

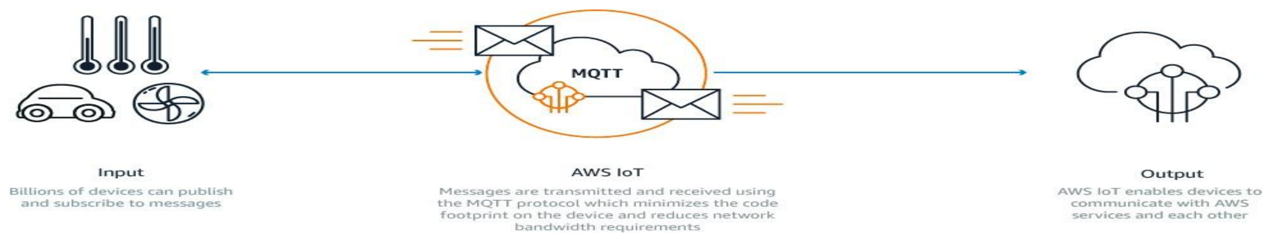
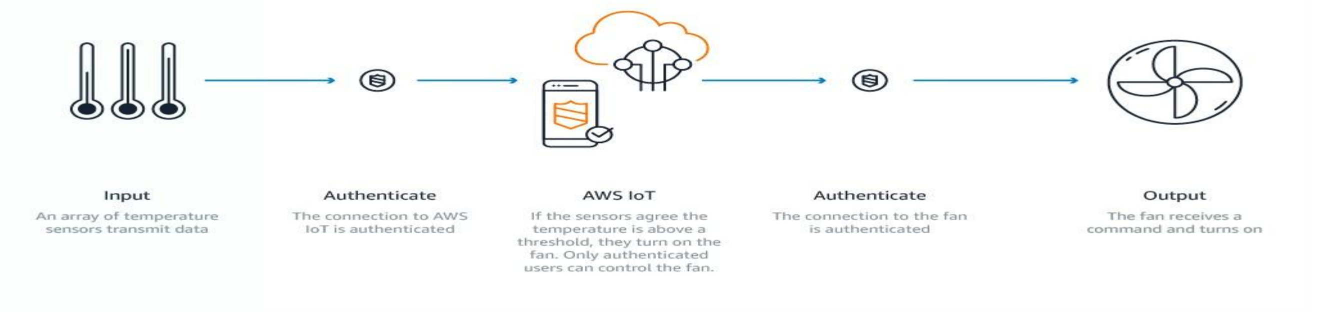

MQTT
Input
Billions of devices can publish and subscribe to messages
AWS IoT
Messages are transmitted and received using the MQTT protocol which minimizes the code footprint on the device and reduces network bandwidth requirements
Output
AWS IoT enables devices to communicate with AWS services and each other

Fig 33. IOT Device Management

Secure Device Connections and Data

AWS IoT Core provides authentication and end-to-end encryption throughout all points of connection, so that data is never exchanged between devices and AWS IoT Core without proven identity. In addition, you can secure access to your devices and applications by applying policies with granular permissions



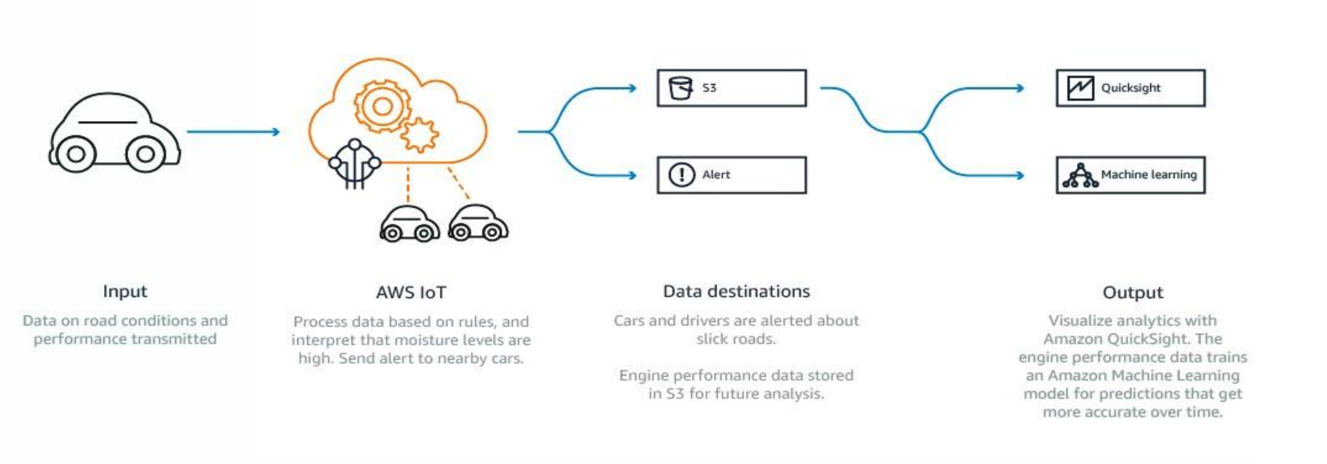
Input
An array of temperature sensors transmit data
Authenticate
The connection to AWS IoT is authenticated
AWS IoT
If the sensors agree the temperature is above a threshold, they turn on the fan. Only authenticated users can control the fan.
Authenticate
The connection to the fan is authenticated
Output
The fan receives a command and turns on

Fig 34.Secure

Connection PROCESS AND ACT UPON DEVICE

DATA

With AWS IoT Core, you can filter, transform, and act upon device data on the fly, based on business rules you define. You can update your rules to implement new device and application features at any time. AWS IoT Core makes it easy to use AWS services like AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch, and Amazon Elasticsearch Service for even more powerful IoT applications



S3
Alert
Quicksight
Machine learning
Input
Data on road conditions and performance transmitted
AWS IoT
Process data based on rules, and interpret that moisture levels are high. Send alert to nearby cars.
Data destinations
Cars and drivers are alerted about slick roads.
Engine performance data stored in S3 for future analysis.
Output
Visualize analytics with Amazon QuickSight. The engine performance data trains an Amazon Machine Learning model for predictions that get more accurate over time.

Fig 35. Process And Act Upon

Device Data Read and set device state at any time

AWS IoT Core stores the latest state of a device so that it can be read or set at anytime, making the device appear to your applications as if it were online all the time. This means that your application can read a device‘s state even when it is disconnected, and also allows you to set a device state and have it implemented when the device reconnects

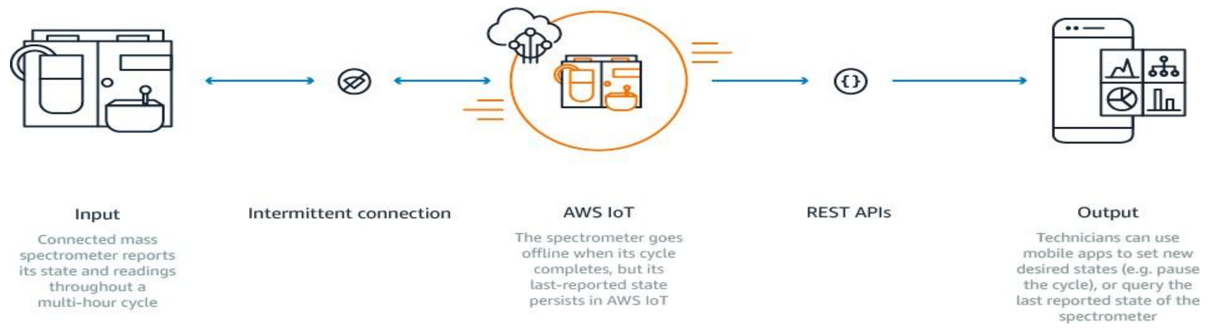

Input
Connected mass spectrometer reports its state and readings throughout a multi-hour cycle
Intermittent connection
AWS IoT
The spectrometer goes offline when its cycle completes, but its last-reported state persists in AWS IoT
REST APIs
Output
Technicians can use mobile apps to set new desired states (e.g. pause the cycle), or query the last reported state of the spectrometer

Fig 36. Read and set device state at any time

7. PROGRAMS for Arduino,

Raspberry pi PIR motion sensor

The passive infra red sensor or simply PIR sensor is integrated with an arduino uno board and we can get serial data through it. The PIR sensor basically works on the thermal radiation which are being emitted by the body of humans as well as animals.

The parts needed for this build are given as
1.Arduino uno or clone
2.PIR Sensor
3.breadboard
4.led(any
colour)
5.buzzer
The following fig 26 shows the pin description of PIR sensor.

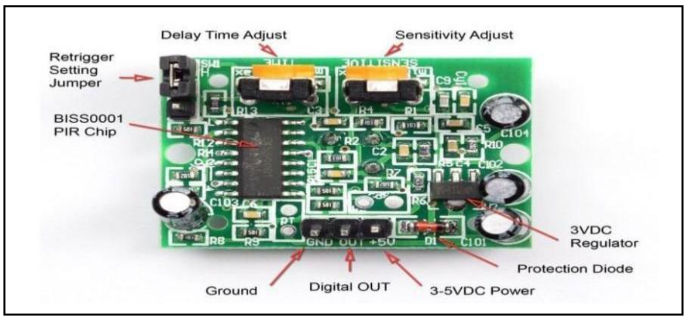


Fig 37.Arduino Board

Features and Electrical Specification

Compact size (28 x 38 mm)
Supply current: DC5V-20V(can design DC3V-24V)
Current drain :< 50uA (Other choice: DC0.8V-4.5V; Current drain: 1.5mA-0.1mA)
Voltage Output: High/Low level

signal ：3.3V (Other choice: Open-Collector Output) TTL output
High sensitivity Delay time ：5s-18 minute

Blockade time ：0.5s-50s (acquiescently 0 seconds) the connections follows as

ARDUINO UNO PIR SENSOR::
+5V------------------------------ +5V(VCC)
GND ---------------------------- GND
5 PIN----------------------------- OUT
The coding related to the following circuit is very simple the code follows in this way At the starting we need to declare the pins which we are going to use

int PIR_output=5; // output of pir
sensor int led=13; // led pin
int buzzer=12;// buzzer pin
After that is done we can move on to the void setup fuction
pinMode(PIR_output, INPUT);// setting pir output as
arduino input pinMode(led, OUTPUT);//setting led as
output
pinMode(buzzer, OUTPUT);//setting buzzer as output
Serial.begin(9600);//serial communication between
arduino and pc
you can download the full program from above PIR_sensor_by_kj_electronics file and upload it to the arduino uno by using the arduino ide

Ultrasonic sensor
It works by sending sound waves from the transmitter, which then bounce off of an object and then return to the receiver. You can determine how far away something is by the time it takes for the sound waves to get back to the sensor. Let's get right to it!. Connections

The connections are very simple:

- VCC to 5V
- GND to GND
- Trig to pin 9
- Echo to pin 10

You can actually connect Trig and Echo to whichever pins you want, 9 and 10 are just the ones

I'm using.

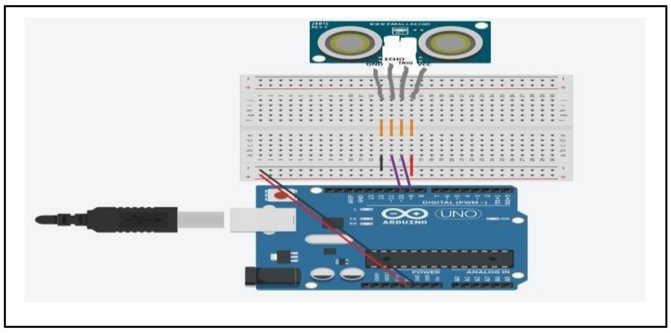


Fig 38.Ultra Sonic Sensor

Code:

we define the pins that Trig and Echo are connected to.

```
const int trigPin = 9;
const int echoPin = 10;
```

Then we declare 2 floats, duration and distance, which will hold the length of the sound wave and how
far away the object is.

```
float  duration, distance;
```

Next, in the setup, we declare the Trig pin as an output, the Echo pin as an input, and start Serial communications.

```
void setup() {
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 Serial.begin(9600);
}
```

Now, in the loop, what we do is first set the trigPin low for 2 microseconds just to make sure that the pin in low first. Then, we set it high for 10 microseconds, which sends out an 8 cycle sonic burst from the transmitter, which then bounces of an object and hits the receiver(Which is connected to t he Echo Pin).

```
void loop() {
 digitalWrite(trigPin,
 LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin,
 HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin,
 LOW);
```

When the sound waves hit the receiver, it turns the Echo pin high for however long the waves were traveling for. To get that, we can use a handy Arduino function called pulseIn(). It takes 2 arguments, the pin you are listening to(In our case, the Echo pin), and a state(HIGH or LOW). What the function does is waits for the pin to go whichever sate you put in, starts timing, and then stops timing when it switches to the other state. In our case we would put HIGH since we want to start timing when the Echo pin goes high. We will store the time in the duration variable. (It returns the time in microseconds)

```
duration = pulseIn(echoPin, HIGH);
```

Now that we have the time, we can use the equation speed = distance/time, but we will make it time x
speed = distance because we have the speed. What speed do we have? The speed of sound, of course! The speed of sound is approximately 340 meters per second, but since the pulseIn() function returns the time in microseconds, we will need to have a speed in microseconds also, which is easy to get. A quick Google search for "speed of sound in centimeters per microsecond" will say that it is .0343 c/μS. You could do the math, but searching it is easier. Anyway, with that information, we can calculate the
distance! Just multiply the duration by .0343 and the divide it by 2(Because the sound waves travel to the object AND back). We will store that in the distance variable.

```
distance = (duration*.0343)/2;
```

The rest is just printing out the results to the Serial
Monitor. Serial.print("Distance: ");
 Serial.println(distan
 ce); delay(100);
}

Temperature Sensor

Connecting to a Temperature Sensor

These sensors have little chips in them and while they're not that delicate, they do need to be handled properly. Be careful of static electricity when handling them and make sure the power supply is connected up correctly and is between 2.7 and 5.5V DC - so don't try to use a 9V battery!

They come in a "TO-92" package which means the chip is housed in a plastic hemi-cylinder with three legs. The legs can be bent easily to allow the sensor to be plugged into a breadboard. You can also solder to the pins to connect long wires.

Reading the Analog Temperature Data

Unlike the FSR or photocell sensors we have looked at, the TMP36 and friends doesn't act like a resistor. Because of that, there is really only one way to read the temperature value from the sensor, and that is plugging the output pin directly into an Analog (ADC) input.

Remember that you can use anywhere between 2.7V and 5.5V as the power supply. For this example I'm showing it with a 5V supply but note that you can use this with a 3.3v supply just as easily. No matter what supply you use, the analog voltage reading will range from about 0V (ground) to about 1.75V.
If you're using a 5V Arduino, and connecting the sensor directly into an Analog pin, you can use these formulas to turn the 10-bit analog reading into a temperature:

Voltage at pin in milliVolts = (*reading from ADC*) * (5000/1024)

This formula converts the number 0-1023 from the ADC into 0-5000mV (= 5V) If you're using a 3.3V Arduino, you'll want to use this:

Voltage at pin in milliVolts = (*reading from ADC*) * (3300/1024)

This formula converts the number 0-1023 from the ADC into 0-3300mV (= 3.3V) Then, to convert millivolts into temperature, use this formula:

Centigrade temperature = [(analog voltage in mV) - 500] / 10

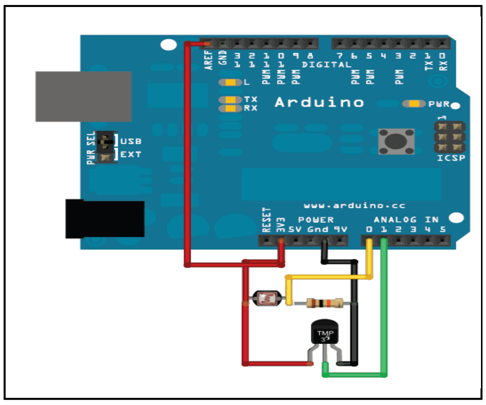


Fig 39.Temperature Sensor

Arduino Sketch - Simple Thermometer

This example code for Arduino shows a quick way to create a temperature sensor, it simply prints to the serial port what the current temperature is in both Celsius and Fahrenheit.

For better results, using the 3.3v reference voltage as ARef instead of the 5V will be more precise and less noisy

```
int sensorPin = 0;
 * setup() - this function runs once when you turn your
Arduino on void setup()
{
  Serial.begin(9600); //Start the serial connection with the computer
                      //to view the result open the serial  monitor
}
void loop()                // run over and over  again
{
 //getting the voltage reading from the temperature
 sensor int reading = analogRead(sensorPin);
 // converting that reading to voltage, for 3.3v arduino
 use 3.3 float voltage = reading *  5.0;
 voltage /= 1024.0;
 // print out the voltage
 Serial.print(voltage); Serial.println(" volts");
 // now print out the temperature
 float temperatureC = (voltage - 0.5) * 100 ; //converting from 10 mv per degree wit 500 mV offset
                                        //to degrees ((voltage - 500mV) times 100)
 Serial.print(temperatureC); Serial.println(" degrees C");
 // now convert to Fahrenheit
 float temperatureF = (temperatureC * 9.0 / 5.0)
 +          32.0;          Serial.print(temperatureF);
 Serial.println("    degrees    F");    delay(1000);
                                     //waiting         a
 second
}
```

PYTHON CODE :

PIR sensor code

```
import RPi.GPIO as GPIO              #Import GPIO
library import time                  #Import time library
GPIO.setmode(GPIO.BOARD)                       #Set GPIO pin
numbering pir = 26              #Associate pin 26 to pir
GPIO.setup(pir, GPIO.IN)               #Set pin as GPIO in
print "Waiting for sensor to settle"
time.sleep(2)                          #Waiting 2 seconds for the sensor to
initiate print "Detecting motion"
while True:
  if GPIO.input(pir):                  #Check whether pir is
    HIGH print "Motion Detected!"
    time.sleep(2)                      #D1- Delay to avoid multiple detection
  time.sleep(0.1)                      #While loop delay should be less than detection(hardware)
  delay
```

Ultrasonic

```
import RPi.GPIO as
GPIO import time
GPIO.setmode(GPIO.B
CM)

TRIG = 23
ECHO = 24

print "Distance Measurement In Progress"

GPIO.setup(TRIG,GPIO.
OUT)
GPIO.setup(ECHO,GPIO.
IN)

GPIO.output(TRIG, False)
```

```
print "Waiting For Sensor To
Settle" time.sleep(2)

GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)

while
  GPIO.input(ECHO)==0:
  pulse_start = time.time()


while GPIO.input(ECHO)==1:
  pulse_end = time.time()

pulse_duration = pulse_end -

pulse_start distance =

pulse_duration * 17150 distance =

round(distance, 2)

print

"Distance:",distance,"cm"

GPIO.cleanup()
```

8. WERABLE DEVELOPMENT BOARDS

Wearable devices are creating great buzz in the market and has become the trend of the day. Innovations are more focused on body-borne computers (also referred as wearables) which are miniature electronic devices that are worn on face, wrist, clothes or even shoes!

Eco-system of wearable devices involves extensive product development knowledge and expertise in technology areas from hardware design to cloud applications to mobile application development. Specifically, for developing wearable devices one requires following technology expertise:

- Hardware / electronic design and development expertise for developing products with minimum form factor size. Moreover, power management expertise is a must-have to ensure that devices can last several days / weeks / months on a single battery charge.

- BSP / firmware development expertise ensuring memory optimized driver development with minimum footprint, enhanced power management and highest performance in spite of low-end microcontroller.
- Smartphone application development on Android and iPhone platforms to allow connectivity with the wearable device.
- Cloud / web application for enabling M2M (machine-to-machine) / IoT (Internet of things) for remote monitoring and control.

Challenges

Having end-to-end expertise for product development is tough ask and there are very few organizations which can ensure quality end-to-end product development. In order to achieve seamless communication, all the components of wearable devices from hardware to cloud to smartphones need to be closely knit. Integration is key and experience of native operating system primitives is a must to extract maximum performance with minimum code! Here are some basic skills required:

- Complex hardware designing abilities smaller sized form factor development.
- Knowledge of choosing the right components for size and efficient power management.
- Certifications to ensure radiations do not affect human body.
- Expertise in driver development for developing optimized drivers for new types of sensors.
- Ability to develop code that fits on lower capacity RAM / flash.

- Domain knowledge for efficient application development.
- Ability to develop iPhone / Android / Windows applications.
- Ability to develop Cloud / web / smartphone connectivity applications.

Input wearables are devices which get input from something and transmit it to the server, for example health monitoring devices. Maven has experience in medical electronics coupled with remote communication capabilities that can be used for developing such type of applications.

Output wearables are device those provide you some information like a smart watch or smart glasses. Maven has already worked on smart watch technology based on Pebble platform.

BSP / firmware / operating system development services

- Developing firmware for microcontrollers from Microchip, TI, Atmel, Cypress, Freescale, NXP etc.

Firmware is a software program permanently etched into a hardware device such as a keyboards, hard drive, BIOS, or video cards. It is programmed to give permanent instructions to communicate with other devices and perform functions like basic input/output tasks. Firmware is typically stored in the flash ROM (read only memory) of a hardware device. Firmware was originally designed for high level software and could be changed without having to exchange the hardware for a newer device. Firmware also retains the basic instructions for hardware devices that make them operative. Without firmware, a hardware device would be non-functional.

Development Boards for IoT

- Driver development for various sensors such as accelerometers, gyroscopes, motion sensors, proximity sensors, magnetometers etc.
- Driver development for communication interfaces like BLE 4.0, NFC, RF and even using the 3.5 mm audio headphone jack.
- Development of power management algorithms.
- Power management by using the right components and critical designs.

Embedded / real-time application development services

Key aspects of application development include:

- Developing algorithms based on various type of sensors such as using combination of accelerometer, gyrometer and magnetometer for determining relative positions, sudden falls, acceleration, angle and so on.
- Development of fast and intelligent data transfer protocols with minimum overheads considering both wired and wireless data transfer mechanisms
- Development of over the air firmware upgrade / remote diagnostics and health monitoring protocols.
- Integrating with smartphones.

Smart phone application development services

Usability, rich user interface and performance are the key parameters for application development. Some of the aspects of smartphone application development include:

- Developing communication protocols over BLE, NFC and 3.5 mm jack for getting data from the wearable devices.
- Intelligence to take decisions based on the data, such as send event / alarm notifications over SMS, transferring data to the server.
- Integrating navigation applications to give direction on an output type of wearable device such as a smart watch.

Cloud / web application development services

With applications hosted on platforms like Amazon, Microsoft and similar, availability and uptimes are assured. Development expertise include:

- Developing applications in HTML5, Dot Net and Java with database servers such as SQL, Oracle, PostgreSQL, MySQL and application servers such as IIS, Tomcat and JBOSS.
- Building applications for displaying real-time parameters, historical trends.

1. C.H.I.P

CHIP is the new kid on the block. It comes with a powerful 1GHz processor powered by Allwinner R8. The best thing about CHIP is that it comes with embedded Bluetooth 4.0 and WiFi radios, providing out- of-the-box connectivity. The board has 4GB of high-speed storage to run a special Linux distribution based on Debian. You don‘t need a separate SD Card to install and run the OS. With 8 GPIO pins, CHIP can be connected to a variety of sensors. The board also supports PWM, UART, I2C for connecting motors and other actuators. One of the key advantages of CHIP is the cost and the form-factor. Developers can SSH into the Linux OS and install required packages.

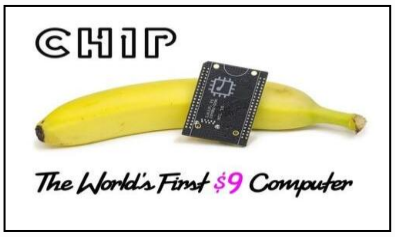


Fig 40. CHIP

2. Mediatek Linkit One

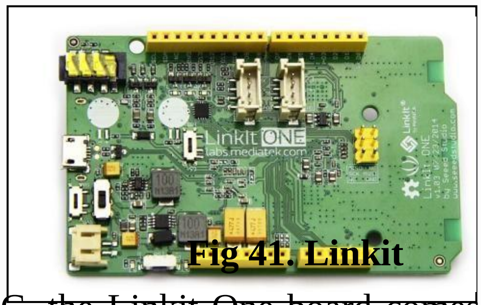


Fig 41. Linkit

Based on the smallest SOC, the Linkit One board comes with compatible Arduino pinout features. The chipset is based on with 260MHz speed. Regarding connectivity, Linkit One has the most comprehensive collection of radios – GPS, GSM, GPRS, WiFi, and Bluetooth. One of the unique features of Linkit One is the rich API that can be used from Arduino IDE. The SDK comes with libraries to connect the board to AWS and PubNub. Because it supports the Arduino pinout, multiple shields from the Arduino ecosystem can be used with the board.

3.Particle Photon

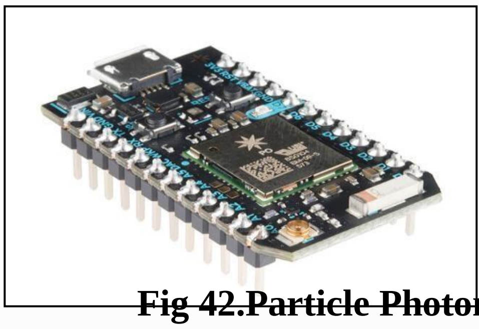
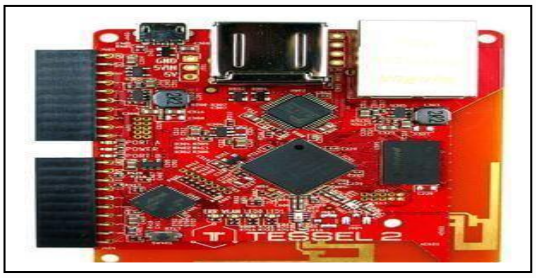


Fig 42.Particle Photon

Photon is one of the smallest prototyping boards available in the market. It comes with the same Broadcom BCM43362 Wi-Fi chip that powers Next, LiFX, and Amazon Dash buttons. Powered by the STM32F205 120Mhz ARM Cortex M3 processor, Photon has 1MB flash and 128KB RAM. Once configured, the board is accessible from the Internet, which makes it an ideal prototyping platform to build connected applications. The board comes with five analog pins and eight digital pins for connecting various sensors and actuators. The official iOS and Android Apps that Particle has published come handy in controlling these pins directly. The powerful web-based IDE lets you write sketches that are compatible with Arduino

3.Tessel



Tessel 2 is a solid development board for serious developers. It comes with a choice of sensors and actuators that can be directly connected to the module ports. The board is powered by a 580MHz MediaTek MT7620n processor for faster execution. It has 64 MB DDR2 RAM & 32 MB Flash, which is more than sufficient for running complex code. The Micro-USB port is used for both powering the board as well as connecting to the PC. The embedded Wi-Fi and Ethernet ports bring connectivity to Tessel. It has a wide collection of sensors and actuators that come along with the required libraries. Based on JavaScript and Node.js, it is easy to get started with the platform. Developers looking for a rapid prototyping platform can go for Tessel 2.

3. Adafruit Flora

It‘s a wearable electronic platform based on the most popular Arduino microcontroller. Flora‘s size makes it an ideal choice for embedded it in clothes and apparel. It comes with a thin, sewable, conductor thread which acts as the wire that connects the power and other accessories. The latest version of Flora ships with a micro-USB and Neopixel LEDs for easy programmability and testing.
Adafruit Flora is based on Atmega 32u4 microcontroller, which powers Arduino Mega and Leonardo. There is an onboard polarized 2 JST battery connector with protection Schottky diode for use with external battery packs from 3.5v to 9v DC. Given its compatibility with Arduino, most of the sketches would run without modifications. You can use the same Arduino IDE with that you may already be familiar.

Fig 44.Adafruit Flora

4. LightBlue Bean

LightBlue Bean is an Arduino-compatible microcontroller board that ships with embedded Bluetooth Low Energy (BLE), RGB LED, temperature sensor, and an accelerometer. Bean+ is the successor to the already popular, which includes a rechargeable LiPo battery along with a couple of Grove connectors. The board comes with a coin-cell battery, which further helps it to maintain the small form factor. It can be paired with Android or iOS devices for remote connectivity and control. It also comes with a software called BeanLoader for programming from Windows or Mac equipped with BLE. BeanLoader installs an Arduino IDE add-on for programming the Bean platform. LightBlue Bean / Bean+ is powered by an ATmega328p microcontroller with 32KB Flash memory and 2KB SRAM. With 8 GPIO pins, two analog pins, four PWM pins, and an I2C port, Bean is perfect for quickly prototyping BLE-based IoT projects.

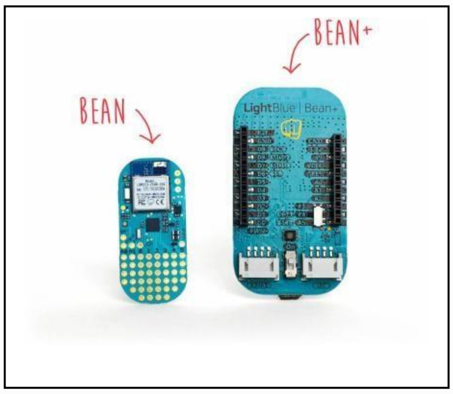


Fig 45.Light blue Bean

5. Udoo Neo

Udoo Neo is a full-blown computer that also has an Arduino-compatible microcontroller. It‘s positioned as the combination of Raspberry Pi and Arduino. The board has the same pinout as Arduino Uno. Neo embeds two cores on the same processor – a powerful 1GHz ARM Cortex-A9, and an ARM Cortex-M4 I/O real-time co-processor. It packs a punch with an embedded 9-axis motion sensors and a Wi-Fi + Bluetooth 4.0 module. You can install Android Lollipop or a customized flavor of Debian Linux called UDOObuntu, which is compatible with Ubuntu 14.04 LTS.
When it comes to the power-packed features and specifications, Udoo NEO is nothing short of a desktop computer. With a Freescale i.MX 6SoloX applications processor with an embedded ARM Cortex-A9 core and a Cortex-M4 Core, Neo comes with 1GB RAM. The Micro HDMI port can be connected to an external display and audio sources. The standard Arduino pin layout is compatible with Arduino shields. You can install Node.js, Python, and even Java on Udoo Neo.

Fig 46.Udoo neo

6. Intel Edison

Trust Intel to deliver the most powerful single-board computer for advanced IoT projects. Intel Edison is a high-performance, dual-core CPU with a single core micro-controller that can support complex data collection. It has an integrated Wi-Fi certified in 68 countries, Bluetooth® 4.0 support, 1GB DDR and 4GB flash memory. Edison comes with two breakout boards – one that‘s compatible with Arduino and the other board designed to be a smaller in size for easy prototyping. The Arduino breakout board has 20 digital input/output pins, including four pins as PWM outputs, Six analog inputs, one UART (Rx/Tx), and one I2C pin. Edison runs on a distribution of embedded Linux called Yocto. It‘s one of the few boards to get certified by Microsoft, AWS, and IBM for cloud connectivity.

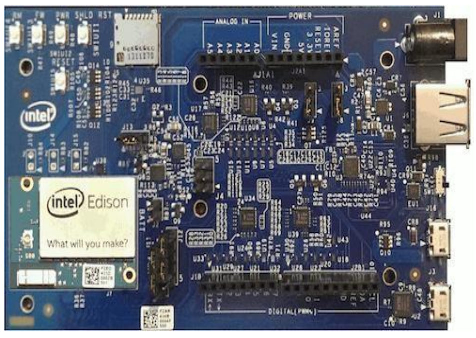


Fig 47.Intel Edison

7. Raspberry Pi

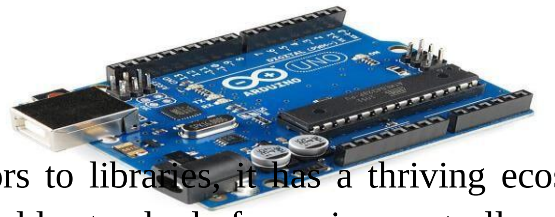
Raspberry Pi is undoubtedly the most popular platform used by many hobbyists and hackers. Even non- technical users depend on it for configuring their digital media systems and surveillance cameras. The recently launched Raspberry Pi 3 included built-in WiFi and Bluetooth making it the most compact and standalone computer. Based on a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor and 1GB RAM, the Pi is a powerful platform. The Raspberry Pi 3 is equipped with
2.4 GHz WiFi 802.11n and Bluetooth 4.1 in addition to the 10/100 Ethernet port. The HDMI port makes it further easy to hook up A/V sources.
Raspberry Pi runs on a customized Debian Linux called Raspbian, which provides an excellent user experience. For developers and hackers, it offers a powerful environment to install a variety of packages including Node.js, the LAMP stack, Java, Python and much more. With four USB ports and 40 GPIO pins, you can connect many peripherals and accessories to the Pi.

Fig 48.RaspBerry pi

There are third party breakout boards to connect various Arduino shields to the Pi. At a throwaway price of $35, Raspberry Pi 3 is certainly the most affordable and powerful computing platform.

8. Arduino Uno

Arduino Uno remains to be the top favorite of absolute beginners and experts. Considered to be one of the first microcontroller-based development boards, the Arduino Uno R3 is simplest yet the most powerful prototyping environment. It is based on the ATmega328P which has 14 digital input/output pins and six analog inputs. Though it comes with just 32 KB of Flash memory, it can accommodate code that deals with complex logic and operations.Arduino enjoys the best community participation and support.



From sensors to actuators to libraries, it has a thriving ecosystem. The board layout has become almost the gold standard for microcontrollers. Almost every prototyping environment tries to be compatible with the Arduino pin breakout. The open source IDE to develop sketches is another reason for its popularity. With a simple syntax based on ‗C‘ language, the code is easy to learn. If you are eager to learn basics of electronics and IoT, look no further. Do yourself a favor and get an Arduino Uno R3.

9. Programs and Stack for Constrained Devices Sensors and Actuators

The ―Thing‖ in the IoT is the starting point for an IoT solution It is typically the originator of the data, and it interacts with the physical world Things are often very constrained in terms of size or power supply; therefore, they are often programmed using microcontrollers (MCU) that have very limited capabilities The microcontrollers powering IoT devices are specialized for a specifc task and are designed for mass production and low cost

The software running on MCU-based devices aims at supporting specifc tasks. The key features of the software stack running on a device may include

1. IoT Operating System – many devices will run with ‗bare metal‘, but some will have embedded or real- time operating systems that are particularly suited for small constrained devices, and that can provide Io T- specifc capabilities.

2. Hardware Abstraction – a software layer that enables access to the hardware features of the MCU, such as fash memory, GPIOs, serial interfaces, etc

3. Communication Support – drivers and protocols allowing to connect the device to a wired or wireless protocol like Bluetooth, Z-Wave, Thread, CAn bus, MQTT, CoAP, etc , and enabling device communication 4 Remote Management – the ability to remotely control the device to upgrade its frmware or to monitor its battery level.

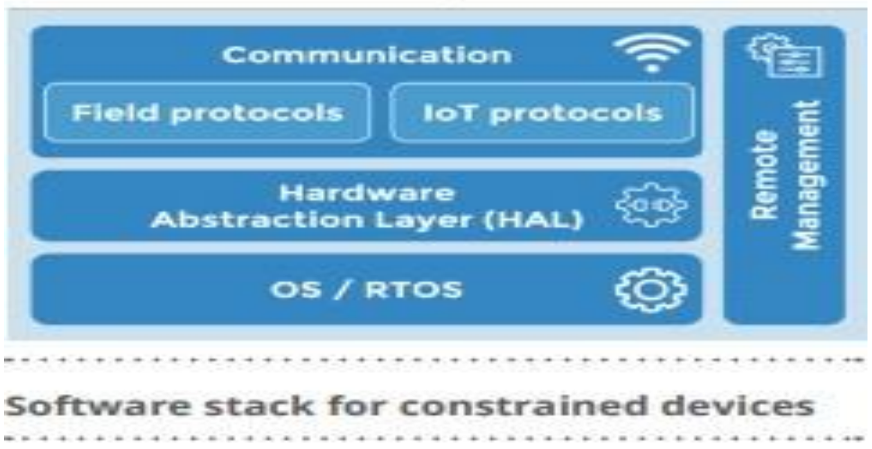


Fig 50.Software Stack for devices

Stack for Gateways

Connected and Smart

Things

The IoT gateway acts as the aggregation point for a group of sensors and actuators to coordinate the connectivity of these devices to each other and to an external network An IoT gateway can be a physical piece of hardware or functionality that is incorporated into a larger ―Thing‖ that is connected to the network For example, an industrial machine might act like a gateway, and so might a connected automobile or a home automation appliance

An IoT gateway will often processing of the data ―at the edge‖ and storage capabilities to deal with network latency and reliability For device to device connectivity, an IoT gateway deals with the interoperability issues between incompatible devices A typical IoT architecture would have many IoT gateways supporting masses of devices

IoT gateways are becoming increasingly dependant on software to implement the core functionality The key features of a gateway software stack include

1 Operating System – typically a general purpose operating system such as Linux

2 Application Container or Runtime Environment – IoT gateways will often have the ability to run application code, and to allow the applications to be dynamically updated For example, a gateway may have support for Java, Python, or node js

3 Communication and Connectivity – IoT gateways need to support different connectivity protocols to connect with different devices (e.g. Bluetooth, Wi-Fi, Z-Wave, ZigBee). IoT gateways also need to connect to different types of networks (e g Ethernet, cellular, Wi-Fi, satellite, etc …) and ensure the reliability, security, and confidentiality of the communications.

4 Data Management & Messaging – local persistence to support network latency, ofine mode, and real- time analytics at the edge, as well as the ability to forward device data in a consistent manner to an IoT Platform

5 Remote Management – the ability to remotely provision, configure, startup/shutdown gateways as well as the applications running on the gateways

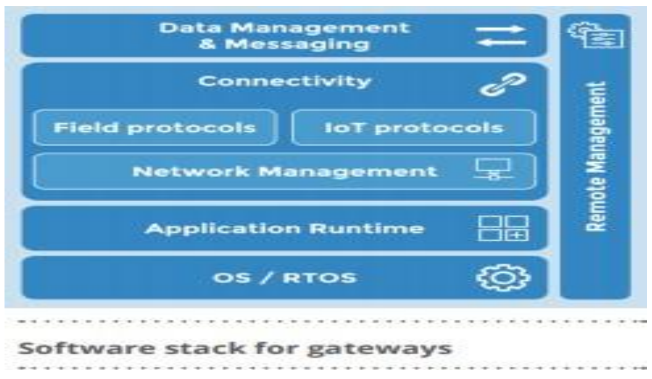


Fig 51. Software Stack for Gateway

Stack for IoT Cloud Platforms

The IoT Cloud Platform represents the software infrastructure and services required to enable an IoT solution An IoT Cloud Platform typically operates on a cloud infrastructure (e g OpenShift, AWS, Microsoft Azure, Cloud Foundry) or inside an enterprise data center and is expected to scale both horizontally, to support the large number of devices connected, as well as vertically to address the variety of IoT solutions The IoT Cloud Platform will facilitate the interoperability of the IoT solution with existing enterprise applications and other IoT solutions The core features of an IoT Cloud Platform include

1. Connectivity and Message Routing – IoT platforms need to be able to interact with very large numbers of devices and gateways using diferent protocols and data formats, but then normalize it to allow for easy integration into the rest of the enterprise

2. Device Management and Device Registry – a central registry to identify the devices/gateways running in an IoT solution and the ability to provision new software updates and manage the devices

3 . Data Management and Storage – a scalable data store that supports the volume and variety of IoT data Software stack for gateways Software stack for IoT Cloud Platforms 7 The Three Software Stacks Required for IoT Architectures IOT ARCHITECTURES

4 Event Management, Analytics & UI – scalable event processing capabilities, ability to consolidate and analyze data, and to create reports, graphs, and dashboards

5 Application Enablement – ability to create reports, graphs, dashboards, … and to use API for application integration.

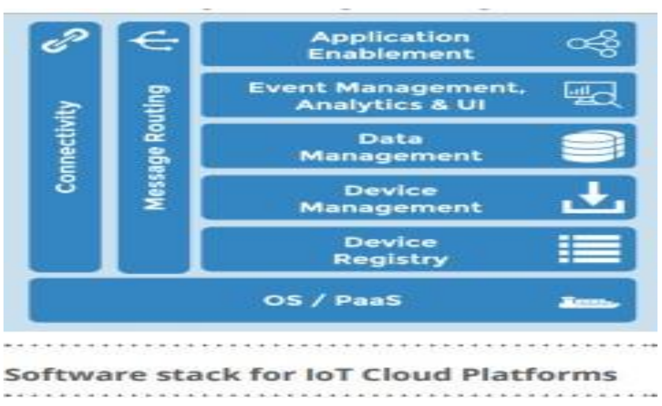


Fig 52.Software Stack for Cloud

Cross-Stack Functionality

Across the different stacks of an IoT solution are a number of features that need to be considered for any IoT architecture, including

1 Security – Security needs to be implemented from the devices to the cloud Features such as authentication, encryption, and authorization need be part of each stack

2 Ontologies – The format and description of device data is an important feature to enable data analytics and data interoperability. The ability to define ontologies and metadata across heterogeneous domains is a key area for IoT

3 Development Tools and SDKs – IoT Developers will require development tools that support the diferent hardware and software platforms involved

10. Packages for Project

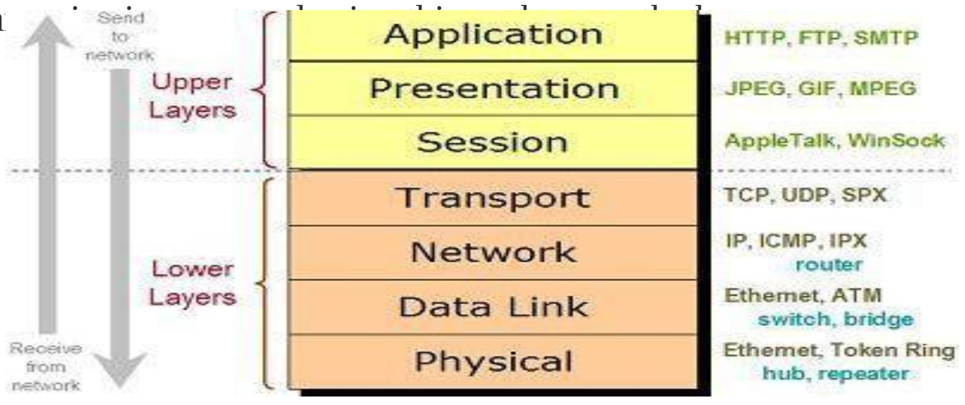
Python packages for developing IoT applications. I mostly develop applications on the Raspberry Pi Model 3 or the Intel Edison. Both are extremely capable Single Board Computers, with a lot of peripherals to play around with. Both support Linux derived OS distributions and support full Python 2.7 or 3.4. With the right Python packages installed, both of these devices become very versatile components in the IoT domain.

Mraa

mraa is a skeleton GPIO library for most SBCs which support Python. The good thing about it is that there's just one library for all devices, so I don't have to use different ones for an Edison and a Pi. Being a high level library, reading from and writing to pins is a one line affair, and the library also provides support for communication protocols such as I2C, UART and SPI

Sockets

sockets is a package which facilitates networking over TCP/IP and UDP using Python. It provides access to Berkeley socket APIs to access the Internet. Both TCP/IP and UDP being Transport layer protocols, are ideal for communication with devices on the same WiFi network. One of the more interesting uses of sockets, in my experience is that one can build their own com



.

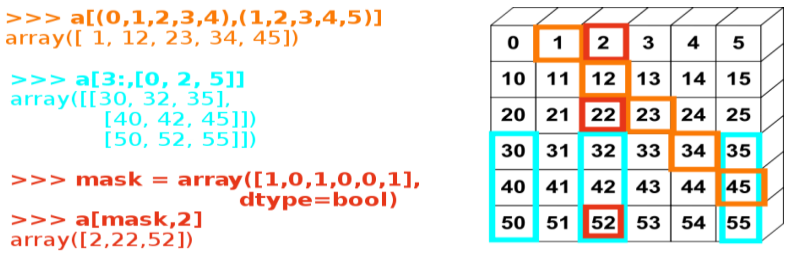
Fig 53.Socket

Mysqldb

A database is a no-brainer when it comes to most IoT applications. For something whose sole purpose is to send data to the internet, there should be a database, at least a remote one which stores all this generated data. MySQL is the go-to relational database for most developers. In this regard, mysqldb is a very convenient little tool which circumvents the need to execute shell commands within a Python script to read and write to a database.

Numpy

Having used MatLab extensively during my undergraduate studies, I've grown accustomed to dealing with arrays. Python, on the other hand, deals with lists as a substitute for the array which is the same as having a birch tree replace your Rottweiler as the guardian of the house. It just doesn't work. Thankfully, numpy is there to help you out. It is, in essence, a package for scientific computing using Python, very similar to MatLab, but *much* lighter. The feature I use most is to read sensor data in bulk from my databases and work on them using the inbuilt functions.



```
>>> a[(0,1,2,3,4),(1,2,3,4,5)]
array([ 1, 12, 23, 34, 45])

>>> a[3:,[0, 2, 5]]
array([[30, 32, 35],
          [40, 42, 45]])
          [50, 52, 55]])

>>> mask = array([1,0,1,0,0,1],
                    dtype=bool)
>>> a[mask,2]
array([2,22,52])
```

Fig 54.numpy

Matplotlib

Data visualization is one of the most fundamental operations that can be performed. It looks pretty impressive when you convert a huge list of numbers to a concise graph which can be understood intuitively. It's also very useful if you happen to be an academician. You know how important those graphs can be in a publication. matplotlib provides a number of different styles of graphs that can be plotted using local data. If you're a data scientist, this is althemore useful to you. Personally, matplotlib is a very usefultool to quickly give me insight to the data I have at my disposal.

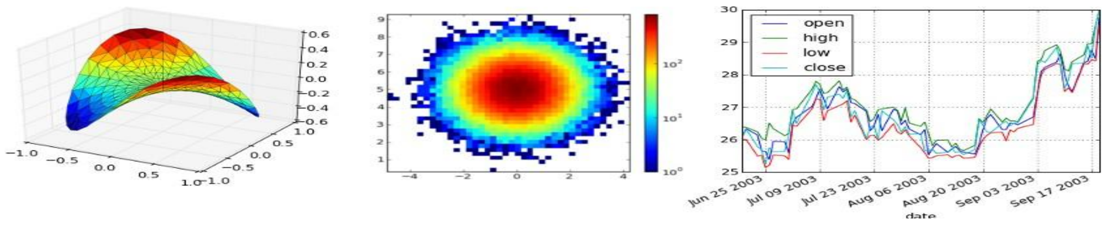
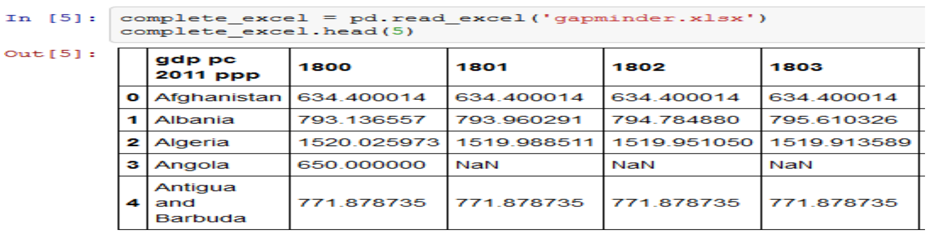


Fig 55. Matplot

Pandas

Another library for data scientists, pandas is a package dedicated towards data analysis. It is in essence, a local alternative to using SQL databases which is more suited to dealing with data as it is built on numpy. It has many advantages over the former, such as a more streamlined approach to data handling and analysis, direct operations on local datasets and the ability to handle heterogeneous and unordered data



```
In [5]:  complete_excel = pd.read_excel('gapminder.xlsx')
         complete_excel.head(5)
Out[5]:
```

	gdp pc 2011 ppp	1800	1801	1802	1803
0	Afghanistan	634.400014	634.400014	634.400014	634.400014
1	Albania	793.136557	793.960291	794.784880	795.610326
2	Algeria	1520.025973	1519.988511	1519.951050	1519.913589
3	Angola	650.000000	NaN	NaN	NaN
4	Antigua and Barbuda	771.878735	771.878735	771.878735	771.878735

Openv

The big brother of signal processing, image processing, was traditionally the domain of high performance, custom built hardware. Although such devices still carry out the job much faster than their single board counterparts, it is at the very least, a possibility. And, in situations where mobility and connectivity are prioritized over speed, this may just be the solution for those rare times. Opencv is a Python port of the very successful C library for image processing. It contains high-level variants of familiar image processing functions which make photo analysis much easier.

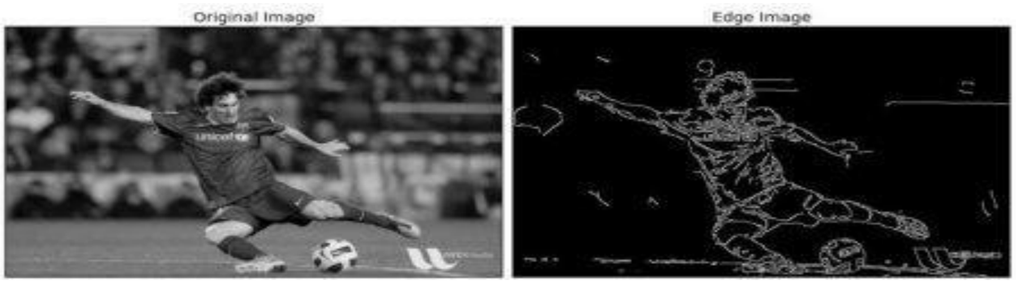
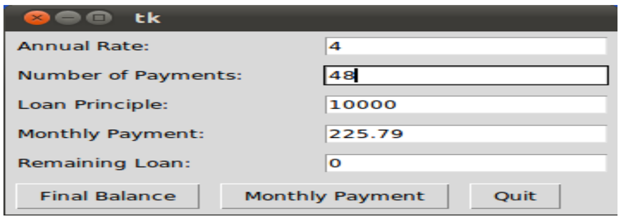


Fig 57. Opencv

tkinter

Although this library does come preinstalled with all installations of Python, its still worthy of a mention. Tkinter is a GUI development library which comes bundled in with all distributions of Python. For people who are more comfortable with a stab wound rather than object oriented programming, learning how to use this package may be a bit daunting at first, but the rewards more than make up for the effort. Every aspect of your Python script can be controlled via a completely ad hoc GUI. This is extremely useful in situations such as functionality testing or repeated executions of the same code.



tk

Annual Rate:	4
Number of Payments:	48
Loan Principle:	10000
Monthly Payment:	225.79
Remaining Loan:	0

Fig58. Tkinter

Tensorflow

Tensorflow is a package for numerical computations for machine learning. It utilizes a different mathematical representation called data flow graphs which use nodes as mathematical operations and edges as data arrays. This is a very useful library to have if you deal with a lot of non linear datasets or work extensively with decision trees and neural networks.

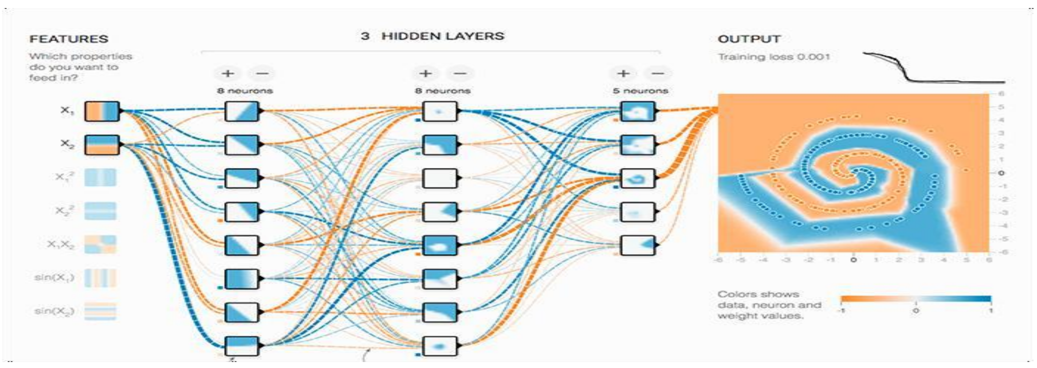

FEATURES
Which properties do you want to feed in?
3 HIDDEN LAYERS
8 neurons
8 neurons
5 neurons
OUTPUT
Training loss 0.001
Colors shows data, neuron and weight values.

Fig 59. Tensorflow

Requests

HTTP is one of the major protocols used in traditional internet based resource exchange, being more suited towards large data exchanges. The requests package is used in Python to make HTTP calls and parse responses. This package is useful when dealing with HTTP based third party cloud services.

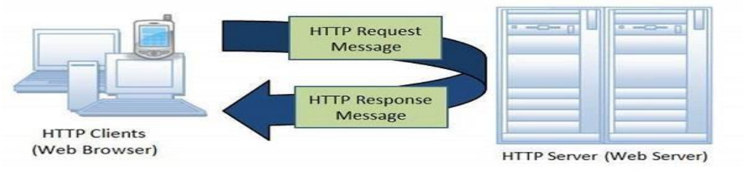


Fig 60. Requests

11. Open source tools and resources for IoT

1. AllSeen Alliance
2. Open Interconnect Consortium (OIC)
3. COMPOSE
4. Eclipse
5. Open Source Hardware Association (OSHWA)

Protocols

6. Advanced Message Queuing Protocol (AMQP)
7. Constrained Application Protocol (CoAP)
8. Extensible Messaging and Presence Protocol (XMPP)
9. OASIS Message Queuing Telemetry Transport (MQTT)
10. Very Simple Control Protocol (VSCP)

Operating systems (OS)

11. ARM mbed
12. Canonical Ubuntu and Snappy Ubuntu Core
13. Contiki
14. Raspbian

15. RIOT
16. Spark
17. webinosAPIs
18. BipIO
19. Qeo Tinq
20. Zetta
21.
1248.io

Horizontal platforms

22. Canopy
23. Chimera IoT
24. DeviceHive
25. Distributed Services Architecture (DSA)
26. Pico Labs (Kynetx open source assigned to Pico Labs)
27. M2MLabs Mainspring
28. Nimbits
29. Open Source Internet of Things (OSIOT)
30. prpl Foundation
31. RabbitMQ
32. SiteWhere
33. webinos
34. Yaler

Middleware

35. IoTSyS
36. OpenIoT
37. OpenRemote
38. Kaa

Node flow editors

39. Node-RED
40. ThingBox

Toolkits

41. KinomaJS
42. IoT Toolkit

Data visualization

43. Freeboard
44. ThingSpeak

Search

45. Thingful

Hardware

46. Arduino Ethernet Shield
47. BeagleBone

48. Intel Galileo
49. openPicus FlyportPro
50. Pinoccio
51. WeIO
52. WIZnet

In-memory data grids

53. Ehcache
54. Hazelcast

Home automation

55. Home Gateway Initiative (HGI)
56. Ninja Blocks
57. openHAB
58. Eclipse SmartHome
59. PrivateEyePi
60. RaZberry
61. The Thing System

Robotics

62. Open Source Robotics Foundation

Mesh networks

63. Open Garden
64. OpenWSN

Health

65. e-Health Sensor Platform

Air pollution

66. HabitatMap Airbeam

Water

67. Oxford Flood Network