SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – I – Introduction to IoT – SCSA1308

SCSA1308 INTRODUCTION TO IoT

UNIT 1

Introduction to IoT, Current technological trends and future prospects- Evolution of IoT - IoTDevices - IoT Devices vs. Computers - Trends in the Adoption of IoT - Societal Benefits of IoT-, Business Scope, Relation with embedded system - IoT Reference Architecture – physical-logical design of IOT-From M2M to IoT, Software define Network

1.1 INTRODUCTION TO IOT

It is a network of physical objects or things sending, receiving, or communicating using the internet or other communication technologies. The Internet of Things (IoT), as this intelligentinterconnectivity between the real and the digital world is called, will rapidly transform every aspect of how we work and do business. By connecting apps with their integrated systems, businesses are able to transform their industry significantly. Today almost 90% of all data generated by tablets, smartphones or connected appliances is never acted upon. Imagine you could change that. It seems safe to say that we have never encountered a single technological platform that combines this much complexity, global reach and novelty. Since IoT allowsdevices to be controlled remotely across the internet, thus it created opportunities to directly connect & integrate the physical world to the computer-based systems using sensors and internet. The interconnection of these multiple embedded devices will be resulting in automation in nearlyall fields and enabling advanced applications. This is resulting in improved accuracy, efficiency and economic benefit with reduced human intervention. It encompasses technologies such as smart grids, smart homes, intelligent transportation and smart cities. The major benefits of IoT are:

- **Improved Customer Engagement** – IoT improves customer experience by automating the action. For e.g. any issue in the car will be automatically detected by the sensors. The driver, as well as the manufacturer, will be notified about it. Until the time driver reaches the service station, the manufacturer will make sure that the faulty part is available at the service station.
- **Technical Optimization** – IoT has helped a lot in improving technologies and making them better. The manufacturer can collect data from different car sensors and analyze them to improve their design and make them much more efficient.
- **Reduced Waste** – Our current insights are superficial, but IoT provides real-time information leading to effective decision-making & management of resources. For example, if a manufacturer finds fault in multiple engines, he can track the manufacturingplant of those engines and can rectify the issue with manufacturing belt.

1.2 CURRENT TECHNOLOGICAL TRENDS AND FUTURE PROSPECTS:

Many firms see big opportunity in IoT uses and enterprises start to believe that IoT holds the promise to enhance customer relationships and drive business growth by improving quality, productivity, and reliability on one side, and on the other side reducing costs, risk, and theft.

IoT and Security Attacks

Forrester thinks that the recent DDoS attack that hit a whopping 1600 websites in the United States was just the tip of the iceberg when it comes to the threat that the connected deviceposes to the world. That attack confirmed the fear of vulnerability of IoT devices with a massive distributed denial of service attack that crippled the servers of services like Twitter, NetFlix, NYTimes, and PayPal across the U.S. It_s the result of an immense assault that involved millionsof Internet addresses and malicious software. All indications suggest that countless Internet of Things (IoT) devices that power everyday technology like closed-circuit cameras and smart- home devices were hijacked by the malware, and used against the servers.

IoT and Mobile Elements

IoT is creating new opportunities and providing a competitive advantage for businesses incurrent and new markets. It touches everything—not just the data, but how, when, where and why you collect it. The technologies that have created the Internet of Things aren't changing the internet only, but rather change the things connected to the internet. More mobile moments (the moments in which a person pulls out a mobile device to get what he or she wants, immediately and in context) will appear on the connected device, right from home appliances to cars to smartwatches and virtual assistants. All these connected devices will have the potential ofoffering a rich stream of data that will then be used by product and service owners to interactwith their consumers.

IoT and artificial Intelligence

In an IoT situation, AI can help companies take the billions of data points they have and boil them down to what_s really meaningful. The general premise is the same as in the retail applications – review and analyzes the data we_ve collected to find patterns or similarities that can be learned from so that better decisions can be made.

IoT and Connectivity

Connecting the different parts of IoT to the sensors can be done by different technologies including Wi-Fi, Bluetooth, Low Power Wi-Fi, Wi-Max, regular Ethernet, Long Term Evolution (LTE) and the recent promising technology of Li-Fi (using light as a medium of communication between the different parts of a typical network including sensors).

IoT and New Business Models

The bottom line is a big motivation for starting, investing in, and operating any business, withouta sound and solid business model for IoT we will have another bubble, this model must satisfy all the requirements for all kinds of e-commerce; vertical markets, horizontal markets, and consumer markets. One key element is to bundle service with the product, for example, devices like Amazon_s Alexa will be considered just

another wireless speaker without the services provided like voice recognition, music streaming, and booking Uber service to mention few.

The IoT can find its applications in almost every aspect of our daily life. Below are some of the examples.

1) Prediction of natural disasters: The combination of sensors and their autonomous coordination and simulation will help to predict the occurrence of land-slides or other natural disasters and to take appropriate actions in advance.

2) Industry applications: The IoT can find applications in industry e.g., managing a fleet of cars for an organization. The IoT helps to monitor their environmental performance and process the data to determine and pick the one that need maintenance.

3) Water Scarcity monitoring: The IoT can help to detect the water scarcity at different places. The networks of sensors, tied together with the relevant simulation activities might notonly monitor long term water interventions such as catchment area management, but mayeven be used to alert users of a stream, for instance, if an upstream event, such as the accidental release of sewage into the stream, might have dangerous implications.

4) Design of smart homes: The IoT can help in the design of smart homes e.g., energy consumption management, interaction with appliances, detecting emergencies, home safety and finding things easily, home security etc.

5) Medical applications: The IoT can also find applications in medical sector for saving lives or improving the quality of life e.g., monitoring health parameters, monitoring activities, support for independent living, monitoring medicines intake etc.

6) Agriculture application: A network of different sensors can sense data, perform data processing and inform the farmer through communication infrastructure e.g., mobile phone text message about the portion of land that need particular attention. This may include smart packaging of seeds, fertilizer and pest control mechanisms that respond to specific local conditions and indicate actions. Intelligent farming system will help agronomists to have better understanding of the plant growth models and to have efficient farming practices by having the knowledge of land conditions and climate variability. This will significantlyincrease the agricultural productivity by avoiding the inappropriate farming conditions.

7) Intelligent transport system design: The Intelligent transportation system will provide efficient transportation control and management using advanced technology of sensors, information and network. The intelligent transportation can have many interesting features such as non-stop electronic highway toll, mobile emergency command and scheduling, transportation law enforcement, vehicle rules violation monitoring, reducing environmental pollution, anti-theft system, avoiding traffic jams, reporting traffic incidents, smart beaconing,minimizing arrival delays etc.

8) Design of smart cities: The IoT can help to design smart cities e.g., monitoring air quality,

discovering emergency routes, efficient lighting up of the city, watering gardens etc.

9) Smart metering and monitoring: The IoT design for smart metering and monitoring will help to get accurate automated meter reading and issuance of invoice to the customers. The IoT can be used to design such scheme for wind turbine maintenance and remote monitoring, gas, water as well as environmental metering and monitoring.

10) Smart Security: The IoT can also find applications in the field of security and surveillancee.g., surveillance of spaces, tracking of people and assets, infrastructure and equipmentmaintenance, alarming etc.

1.3 EVOLUTION OF IOT AND BUSINESS SCOPE:

The term ―Internet of Things‖ (IoT) was first used in 1999 by British technology pioneer Kevin Ashton to describe a system in which objects in the physical world could be connected to the Internet by sensors.12 Ashton coined the term to illustrate the power of connecting Radio- Frequency Identification (RFID) tags13 used in corporate supply chains to the Internet in order to count and track goods without the need for human intervention.

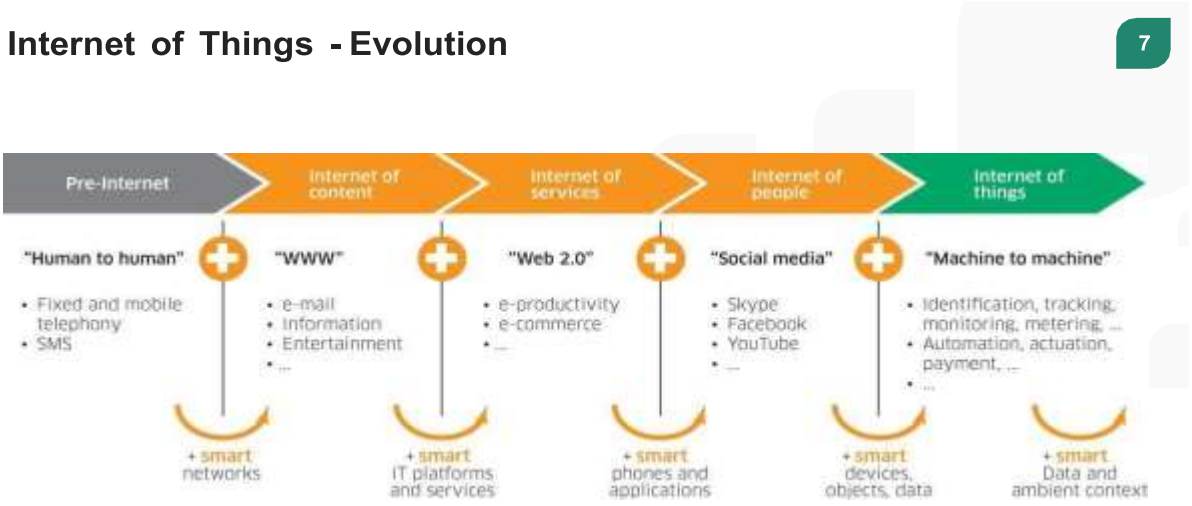


Fig 1.1. IOT Evolution Model

By the late 1970s, for example, systems for remotely monitoring meters on the electrical grid via telephone lines were already in commercial use.14 In the 1990s, advances in wireless technologyallowed ―machine–to–machine‖ (M2M) enterprise and industrial solutions for equipment monitoring and operation to become widespread. Many of these early M2M solutions, however, were based on closed purpose–built networks and proprietary or industry.

From a broad perspective, the confluence of several technology and market trends20 is making itpossible

to interconnect more and smaller devices cheaply and easily:

• Ubiquitous Connectivity—Low–cost, high–speed, pervasive network connectivity, especially through licensed and unlicensed wireless services and technology, makes almost everything –connectable‗.

• Widespread adoption of IP–based networking— IP has become the dominant global standard for networking, providing a well–defined and widely implemented platform of software and tools that can be incorporated into a broad range of devices easily and inexpensively.

• Miniaturization— Manufacturing advances allow cutting-edge computing and communications technology to be incorporated into very small objects. Coupled with greater computing economics, this has fueled the advancement of small and inexpensive sensor devices, whichdrive many IoT applications.

• Advances in Data Analytics— New algorithms and rapid increases in computing power, data storage, and cloud services enable the aggregation, correlation, and analysis of vast quantities of data; these large and dynamic datasets provide new opportunities for extracting information and knowledge.

• Rise of Cloud Computing– Cloud computing, which leverages remote, networked computing resources to process, manage, and store data, allows small and distributed devices to interactwith powerful back-end analytic and control capabilities.

From this perspective, the IoT represents the convergence of a variety of computing and connectivity trends that have been evolving for many decades. At present, a wide range of industry sectors – including automotive, healthcare, manufacturing, home and consumer electronics, and well beyond -- are considering the potential for incorporating IoT technology into their products, services, and operations.

BUSINESS SCOPE

Increase Business Opportunities

IoT opens the door for new business opportunities and helps companies benefit from newrevenue streams developed by advanced business models and services. IoT-driven innovations build strong business cases, reduce time to market and increase return on investments. IoT has the potential to transform the way consumers and businesses approach the world by leveraging the scope of the IoT beyond connectivity.

Enhanced Asset Utilization

IoT will improve tracking of assets (equipment, machinery, tools, etc.) using sensors and connectivity, which helps organizations benefit from real-time insights. Organizations couldmore easily locate issues in the assets and run preventive maintenance to improve asset utilization.

Efficient Processes

Being connected with a maximum number of devices to the internet, IoT allow businesses to be smarter with real-time operational insights while reducing operating costs. The data collected from logistics network, factory floor, and supply chain will help reduce inventory, time to marketand downtime due to maintenance.

Improved Safety and Security

IoT services integrated with sensors and video cameras help monitor workplace to ensureequipment safety and protect against physical threats. The IoT connectivity coordinates multiple teams to resolve issues promptly.

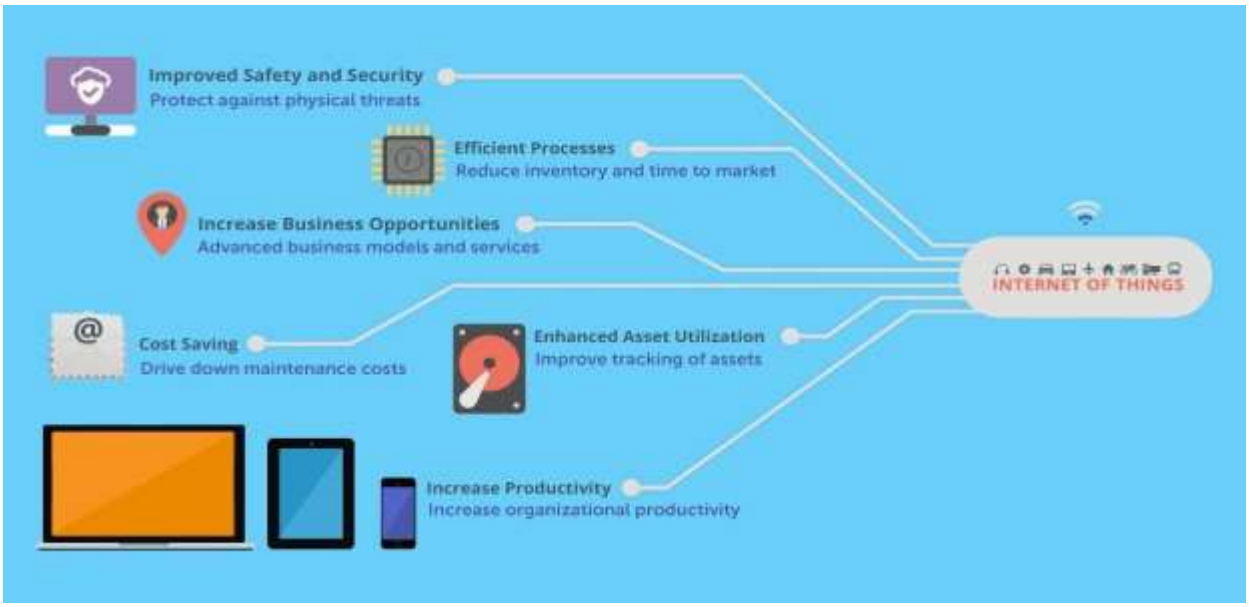


Fig 1.2. Business Scope

Increase Productivity

Productivity plays a key role in the profitability of any business. IoT offers just-in-time training for employees, improve labor efficiency, and reduce mismatch of skills while increasing organizational productivity.

Cost Saving

The improved asset utilization, productivity, and process efficiencies can save your expenditures. For example, predictive analytics and real-time diagnostics drive down the maintenance costs.IoT has reached the pinnacle of inflated expectations of emerging technologies. Even though IoT offers great potential value, organizations must overcome some significant challenges like data and information management issues, lack of interoperable technologies, security and privacy concerns, and the skills to manage IoT_s

growing complexity. However, a professional IoT service provider can overcome these challenges and increase your return on investment.

Logistics

With IoT sensors, supply chain management and order fulfillment processes improve markedlyto meet customer demand. For example, sensors on delivery containers and trucks in transit give managers real-time status updates, allowing them to track their items and ensure they reach the right location at the right time.

Streamlined Industry

IoT also presents automation opportunities for businesses that need to manage and replenish theirstock. When data recorded from IoT devices are tied to your enterprise resource planning (ERP) system, you can accurately monitor your inventory, analyze purchase and consumption rates of a particular product, and automatically reorder items when IoT sensors detect that supply isrunning low. This minimizes out-of-stock incidents and prevents excess stock build-up.

Fast Payment

Given how most payments are done electronically via point-of-sale systems or the internet, IoT has the potential to revolutionize the way businesses process transactions. We re already seeing afew examples of this today as ApplePay not only allows users to purchase goods and services using smartphone applications, but through wearable technology as well.

Soon enough, IoT devices might even allow restaurants and retailers to register or charge their customers the moment they walk through the door.

Market Insight

Businesses that can somehow make sense of IoT-collected data will gain a competitive edge. Marketers, for example, can gather valuable insight into how their products are used and which demographic is utilizing them the most. This information can then inform future marketing efforts and give businesses more direction on how to improve their products and services for their customers. Although businesses will certainly face many challenges in implementing the Internet of Things, those who manage to overcome them will reap all the benefits of thisburgeoning technology.

1.4 IoT Devices

IoT devices are the nonstandard computing devices that connect wirelessly to a networkand have the ability to transmit data, such as the many devices on the internet of things (IoT).IoT involves extending internet connectivity beyond standard devices, such as desktops, laptops, smartphones and tablets, to any range of

Fig 1.3. IoT Devices

traditionally "dumb" or non-internet- enabled physical devices and everyday objects. Embedded with technology, these devicescan communicate and interact over the internet. They can also be remotely monitored and controlled.

Example of an IoT device:

Connected devices are part of an ecosystem in which every device talks to other related devices in an environment to automate home and industry tasks. They can communicate usable sensor data to users, businesses and other intended parties. The devices can becategorized into three main groups: consumer, enterprise and industrial.

Consumer connected devices include smart TVs, smart speakers, toys, wearables andsmart appliances.

In a smart home, for example, devices are designed to sense and respond to a person's presence. When a person arrives home, their car communicates with the garage to open the door. Once inside, the thermostat is already adjusted to their preferred temperature, andthe lighting is set to a lower intensity and color, as their smart watch data indicates it has been a stressful day. Other smart home devices include sprinklers that adjust the amount ofwater given to the lawn based on the weather forecast and robotic vacuum cleaners that learn which areas of the home must be cleaned most often.

Enterprise IoT devices are edge devices designed to be used by a business. There are a huge variety of enterprise IoT devices available. These devices vary in capability but tend tobe geared toward maintaining a facility or improving operational efficiency. Some options include smart locks, smart thermostats, smart lighting and smart security. Consumer versions of these technologies exist as well.

In the enterprise, smart devices can help with meetings. Smart sensors located in a conference room can help an employee locate and schedule an available room for a meeting, ensuring the proper room type, size and features are available. When meeting attendees enter the room, the temperature will adjust according to the occupancy, the lightswill dim as the appropriate PowerPoint loads on the screen and the speaker begins his or her presentation.

1.5 IoT Devices vs. Computers:

Computers versus the Internet of Things (IoT). In the Internet of Computers, the main data producers and consumers are human beings. However, in the IoT, the main actors becomethings, where things are the majority of data producers and consumers

IoT devices include wireless sensors, software, actuators, and computer devices. They areattached to a particular object that operates through the internet, enabling the transfer of data among objects or people automatically without human intervention.

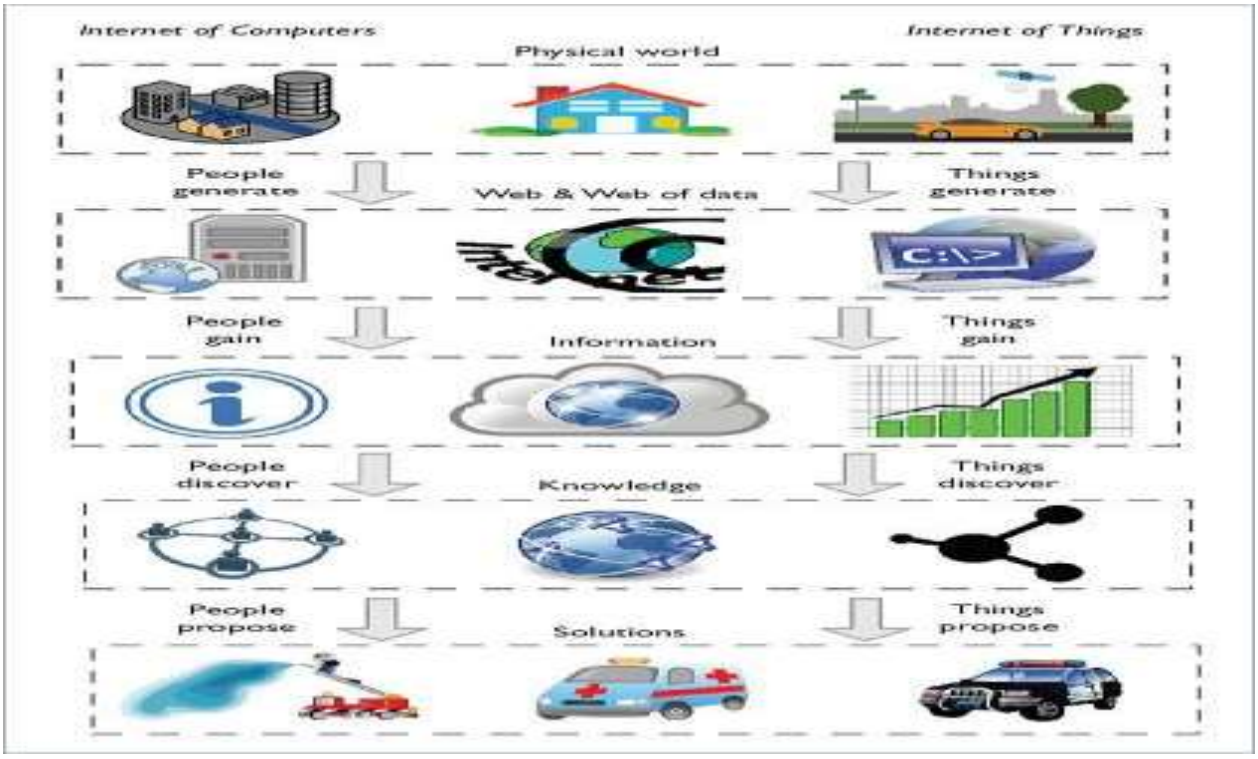


Fig 1.4. IoT Devices vs Computer

1.6 Trends in the Adoption of IoT

- **REDUCTION IN IOT COMPONENT COSTS**

A key trend improving enterprise opportunities for IoT is falling system costs at boththe device and network levels. Because of architectural similarities, IoT stands to significantly benefit from ongoing advancements in the smartphone market whose large volumes and increasing competition are dramatically reducing prices

throughout the value chain. This includes key component parts such as sensors, embedded processors, memory and cellular modems whose steady price declines will continue to drive cost reductions for developing new IoT endpoint applications and thus ease capital requirements for enterprise deployment. Similarly, price reductions continue to occur in cellular communications that support IoT device connectivity. As demand for IoT devices increases, prices for both 3G and 4G networks will continue to decline, which will drive reductions in recurring operationalexpenses for enterprise solutions. Additionally, expanded use of cloud services in IoT and the benefits of their elastic pricing models will enable cost-effective computational resources that will further reduce cost of ownership and improve the value proposition for enterprise clients.

- **END-TO-END IOT PLATFORMS**

While component price reductions will play an important role in the growth of user adoption, reducing the complexity of IoT application development and deployment will also be a major driver in the market. Unlike IT, the IoT market is a highly fragmented ecosystem where requirements for enterprise solutions can vary significantly from clientto client even in the same market vertical. This includes variability in requirements related to device functionality, communications and applications logic, which require integrating a host of different hardware and software elements. Moreover, added complexities are introduced when integration is required with client systems such as local area networks or core enterprise software, which inevitably will require upgrades that will further complicate the IoT offering. As such, these integration, installation and maintenance complexities establish significant barriers to entry for most enterprise IoT solutions, especially when large numbers of device deployments are required over large geographic regions. As such, the expense associated with development and deployment quickly erodes the solution‘s value proposition and return on investment.

To address these issues, new forms of IoT platforms will emerge that will drive out the complexity of application and enterprise solution development and radically decrease cost-of-ownership. These will include what I would describe as configurable, end-to-endIoT platforms that will integrate functionality from the edge of the network into to the cloud. Inherent in the architectural design will be built-in features and functions for sensing, communications and business logic, which can be easily configured for new applications, making implementation very simple and cost effective. Moreover, reducingbarriers for the delivery of sensor information to the cloud will enable 3rd party application developers an entirely new framework from which to build advanced solutions based on cloud-to-cloud data exchange.

As an example, we have released the initial version of our Quantus Sensor Fusion Engine product, an end-to-end IoT platform, which has demonstrated significant benefits related to ease-of-installation, network scalability and application configurability. Configurable IoT system approaches such as Quantus will continue totightly integrate new features throughout the network stack with the aim of reducing time to

market and cost of ownership for new enterprise solutions. These features will also include advancements in analytics both at the device and cloud levels, which is thebasis for the 3rd trend as described below.

- **SENSOR ANALYTICS AND FUSION**

The business value of IoT at its basic level takes the form of what I would describe as improved situational awareness. This includes a better understanding of the status of remote physical systems in terms of their condition, which drives improved operational decisions, actions and efficiencies. For example, working with our partner FCC Environmental, we have deployed an IoT solution built on Quantus that includes cellularbased tank level monitoring devices installed at automotive retail locations. Data from differential pressure sensors mounted on used oil tanks are processed using analytics at the device level to generate volume status and alert messages, which are communicated wirelessly to the cloud for storage, additional processing and reporting to desktop and mobile devices. In this case, Quantus provides operations personnel with actionable information regarding a store‘s past, current and forecasted volumes of used oil, which enables informed decisions regarding the routing of service vehicles dispatched over large geographic locations. Specifically, this includes the ability to now service automotive clients based on need as opposed to fixed routing schedules, an operational change enabled by IoT that drives significant improvements in oil collection efficiency.

This form of sensor analytics is the most common in the industry today and has its rootsin the machine-to-machine (M2M) market where similar sensor solutions for wirelessly monitoring remote assets have been in service for many years. While these types of solutions deliver clear value in specific market verticals, opportunities are emerging to create completely new levels of actionable information through a process known as sensor fusion, which is the generation of new, richer forms of actionable intelligence through the application of analytical processing techniques to diverse, multi-source datasets.

For example, research efforts at IBM recently demonstrated an IoT solution for the mining industry that accurately predicted machine failures using real-time analytics supplied by a mix of equipment sensors and operational data. These multi-source datasets included information regarding truckload bearing capacity, environmental conditions, repair history and other sensors, which were processed using real-time analytics and modeling techniques. The new operational insights provided by the system stands to drive significant productivity gains, which IBM predicts could be worthbillions of dollars annually. Implementation of these sensor analytics and fusion techniques and their ability to capture higher levels of actionable information represents a new stage in the technical evolution of IoT and will significantly increase the value of

information products delivered by these solutions.

1.7 Societal Benefits of IoT:

IoT devices impact society in a meaningful way. Smart homes and offices can save energy costs by controlling the electricity or temperature when one is away from home or work, andthey

Fig 1.5. New IoT use cases across technologies

can offer better security by constant surveillance and taking proactive action in case of a security breach. Smart health devices can improve health care by monitoring patients andremotely administering medication to them, and smart automobiles can request assistance ifrequired or assist in monitoring vehicle speed based on traffic.

IoT is delivering positive economic and social impacts, transforming our societies, theenvironment and our food supply chains for the better. Here are just a few examples:

- Monitoring And Reducing Air Pollution

Cities account for approximately 70% of the world‘s harmful greenhouse gas emissions, despite comprising 2% of the global land area. Each year, more than 3million people die from air pollution. To change this, cities around the world are incorporating IoT-enabled sensors and devices throughout their infrastructure to monitor air quality and are using the data to implement new urban services that canreduce traffic congestion and associated pollution.

- Improving Water Conservation

Cities are also using IoT technologies to conserve water. Barcelona implemented an IoT-enabled, smart

irrigation system using underground probes placed in parks throughout the city to monitor soil moisture. The remotely monitored devices upload data to the cloud and can automatically open electronic valves, watering the landscaping only when needed and when weather conditions are right. As a result, the municipal water bill has been reduced by 25% and, more importantly, water usage hasbeen reduced.

- Saving Critical Species

The IoT is not just about connecting devices, sensors and machines to each other andthe internet. Even people and other living creatures can be connected. Case in point: bees. Bees are the world‘s main pollinator of food crops. However, the global bee population is declining quickly. To help reverse the trend, an international alliance of researchers and scientists is collecting data on bees using micro sensing technology.

By equipping bees with tiny backpacks that use radio-frequency identification (RFID) technology, researchers use electronic readers to record the behavior of individual bees. The technology has revealed an unprecedented level of detail about not only the movements of bees but also the factors impacting their ability to pollinate, such as disease, pesticides, air pollution, water contamination, diet and extreme weather. Ciscoworked together with the researchers to use IoT technologies to gather data from the hives and make it available for a variety of applications and scientific analysis, in the hopes that we can make the changes necessary to save our most precious pollinators.

1.8 RELATIONSHIP WITH EMBEDDED SYSTEMS

Embedded systems are part and parcel of every modern electronic component. These are low power consumption units that are used to run specific tasks for example remote controls, washing machines, microwave ovens, RFID tags, sensors, actuators and thermostats used in various applications, networking hardware such as switches, routers, modems, mobile phones, PDAs, etc. Usually embedded devices are a part of a larger device where they perform specific task of the device. For example, embedded systems are used as networked thermostats inHeating, Ventilation and Air Conditioning (HVAC) systems, in Home Automation embedded systems are used as wired or wireless networking to automate and control lights, security, audio/visual systems, sense climate change, monitoring, etc. Embedded microcontrollers can be found in practically all machines, ranging from DVD players and power tools to automobiles and computed tomography scanners. They differ from PCs temperature when one is away from home or work, andthey in their size and processing power. Embedded systems typically have a microprocessor, a memory, and interfaces with the external world, but they are considerably smaller than their PC counterparts.

Frequently, the bulk of the electronic circuitry can be found in a single chip.

Fig 1.6. Embedded Processing

A sensor detects (senses) changes in the ambient conditions or in the state of another device or asystem, and forwards or processes this information in a certain manner.

1.9 **Analog Sensors** produce a continuous output signal or voltage which is generallyproportional to the quantity being measured. Physical quantities such as Temperature, Speed, Pressure, Displacement, Strain etc. areall analog quantities as they tend to be continuous in nature.

1.10 **Digital Sensors** produce discrete digital output signals or voltages that are a digitalrepresentation of the quantity being measured. Digital sensors produce a binary output signal in the form of a logic –1‖ or a logic –0‖ (–ON‖ or –OFF‖).

An actuator is a component of a machine or system that moves or controls the mechanism or thesystem. An actuator is the mechanism by which a control system acts upon an environment

An actuator requires a control signal and a source of energy.

Power Conservation

Until recently, a common strategy to save power in an embedded system was to execute as quickly as possible, and then go into sleep mode immediately. But there are now processor core

architectures that consume almost no power, although with reduced performance. This is anattractive option for a WSN edge node design.

The programming languages used in deeply embedded systems include C, C++ and sometimes Java. It is important to note that Java always runs on top of an operating system. So, your choice is not between C/C++ or Java; it is whether you will use C/C++ and Java. Java is attractive for IoT devices because the number of Java developers worldwide brings tremendous growth potential to the industry. Oracle_s Java ME Embedded is designed for small devices

When cost is not an issue, we can select a single powerful processor to run all the tasks required of your device. However, a common engineering compromise is to use two processors in the sensor/actuator device.

One low-cost processor (8 or 16 bit) is used for the physical-world interface, and a second 32-bit processor runs the network interface. This second processor isoften placed in a separate module, one that has already been certified for the protocol and FCC compliance.

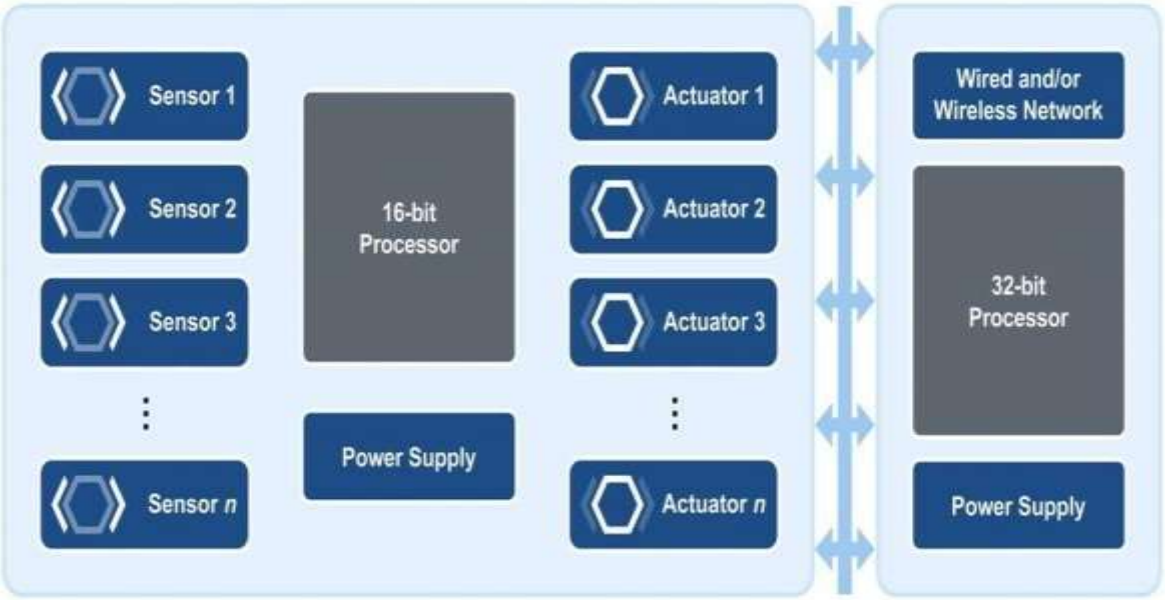


Fig 1.7 IoT Devices with Two Processors

Gateway Design

A gateway connects two dissimilar networks so that data can flow between them. Usually this isa connection between a proprietary network and the Internet.

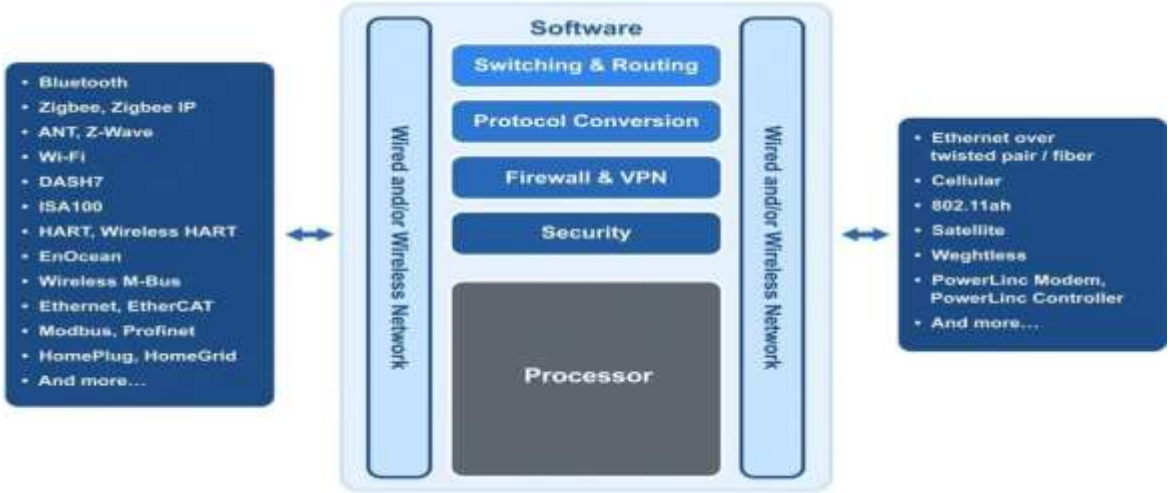


Fig 1.8 Embedded Devices with Gateway

Bluetooth is a wireless technology standard for exchanging data over short distances from fixed and mobile

devices, and building personal area networks.

Zigbee wireless technology is specially designed for sensors and control devices that employ low cost connectivity and widely used for several applications.

Z-Wave is a wireless communications protocol used primarily for home automation. It is a meshnetwork using low-energy radio waves to communicate from appliance to appliance, allowingfor wireless control of residential appliances and other devices, such as lighting control, security systems, thermostats, windows.

Wi-Fi is the name of a popular wireless networking technology that uses radio waves to provide wireless high-speed Internet and network connections. A common misconception is that theterm **Wi-Fi** is short for "wireless fidelity.

ISA100.11a is a wireless networking technology standard developed by the International Society of Automation (ISA). The official description is "Wireless Systems for Industrial Automation: Process Control and Related Applications.

The **EnOcean** technology is an energy harvesting wireless technology used primarily in building automation systems, and is also applied to other applications in industry, transportation, logistics and smart homes. Modules based on EnOcean technology combine micro energy converters withultra low power electronics, and enable wireless communications between batteryless wireless sensors, switches, controllers and gateways.

In home automation, different utilities companies may install a wide variety of IoT devices in your house, each with their own gateway. These can include electricity or gas, water, phone, Internet, cable/satellite, alarm system, medical devices, and so on. Some of these gateways may require additional functions, such as local storage, or a user interface.

1.12 KEY ELEMENTS OF IOT

1. Sensing

The first step in IoT workflow is gathering information at a ‑point of activity.‖ This can beinformation captured by an appliance, a wearable device, a wall mounted control or any number of commonly found devices. The sensing can be biometric, biological, environmental, visual or audible (or all the above). The unique thing in the context of IoT is that the device doing the sensing is not one that typically gathered information in this way. Sensing technology specific to this purpose is required.

2. Communication

This is where things start to get interesting. Many of the new IoT devices we are seeing today arenot

designed for optimal communication with cloud services. IoT devices require a means for transmitting the information sensed at the device level to a Cloud-based service for subsequent processing. This is where the great value inherent in IoT is created. This requires either WiFi
(wireless LAN based communications) or WAN (wide area network… i.e. cellular) communications. In addition, depending on the need short range communication, other capabilities may also be needed. These could include Bluetooth, ZigBee, Near-field or a range ofother short range communication methods. For positioning, GPS is often required as well.

3. **Cloud Based Capture & Consolidation**

Gathered data is transmitted to a cloud based service where the information coming in from the IoT device is aggregated with other cloud based data to provide useful information for the end user. The data being consolidated can be information from other internet sources as well as from others subscribing with similar IoT devices. Most often, there will be some data processingrequired to provide useful information that is not necessarily obvious in the raw data.

4. **Delivery of Information**

The last step is delivery of useful information to the end user. That may be a consumer, a commercial or an industrial user. It may also be another device in the M2M workflow. The goalin a consumer use case is to provide the information in as simple and transparent a method as possible. It requires execution of a well thought out, designed and executed user interface that provides an optimized experience across multiple device platforms – tablets, smartphones, desktop – across multiple operating systems – iOS, Android, Windows, etc.

REFERENCE ARCHITECTURE OF IOT:

The reference architecture consists of a set of components. Layers can be realized by means of specific technologies, and we will discuss options for realizing each component. There are also some cross-cutting/vertical layers such as access/identity management.

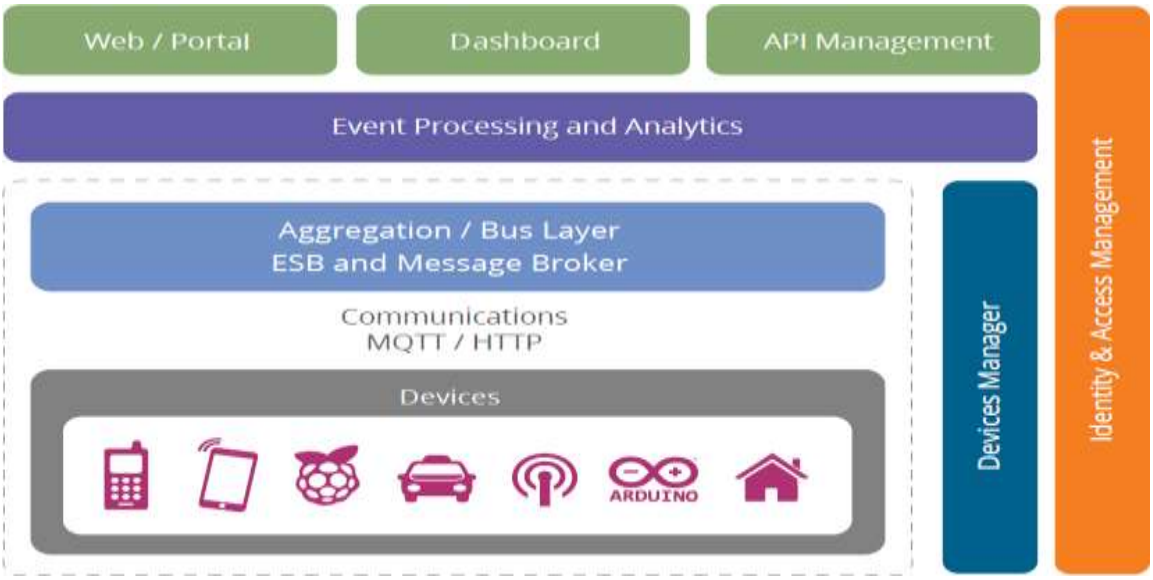


Fig 1.9 Reference architecture for IoT

The layers are

- Client/external communications - Web/Portal, Dashboard, APIs
- Event processing and analytics (including data storage)
- Aggregation/bus layer – ESB and message broker
- Relevant transports - MQTT/HTTP/XMPP/CoAP/AMQP, etc.
- Devices

The cross-cutting layers are

- Device manager
- Identity and access managements

THE DEVICE LAYER

The bottom layer of the architecture is the device layer. Devices can be of various types, but in order to be considered as IoT devices, they must have some communications that either indirectly or directly attaches to the Internet. Examples of direct connections are

- Arduino with Arduino Ethernet connection
- Arduino Yun with a Wi-Fi connection
- Raspberry Pi connected via Ethernet or Wi-Fi
- Intel Galileo connected via Ethernet or Wi-Fi Examples of indirectly connected device include
- ZigBee devices connected via a ZigBee gateway
- Bluetooth or Bluetooth Low Energy devices connecting via a mobile phone
- Devices communicating via low power radios to a Raspberry PiThere are many more such examples of each type.

Each device typically needs an identity. The identity may be one of the following:

•A unique identifier (UUID) burnt into the device (typically part of the System-on-Chip, orprovided by a secondary chip)

- A UUID provided by the radio subsystem (e.g. Bluetooth identifier, Wi-Fi MAC address)
- An OAuth2 Refresh/Bearer Token (this may be in addition to one of the above)
- An identifier stored in nonvolatile memory such as EEPROM

For the reference architecture we recommend that every device has a UUID (preferably anunchangeable ID provided by the core hardware) as well as an OAuth2 Refresh and Bearer token stored in EEPROM.

The specification is based on HTTP; however, (as we will discuss in the communications section)the reference architecture also supports these flows over MQTT.

COMMUNICATIONS LAYER

The communication layer supports the connectivity of the devices. There are multiple potential protocols for communication between the devices and the cloud. The most wellknown three potential protocols are

- HTTP/HTTPS (and RESTful approaches on those)
- MQTT 3.1/3.1.1
- Constrained application protocol (CoAP)

Let_s take a quick look at each of these protocols in turn.

HTTP is well known, and there are many libraries that support it. Because it is a simple text based protocol, many small devices such as 8-bit controllers can only partially support the protocol – for example enough code to POST or GET a resource. The larger 32-bit based devices can utilize full HTTP client libraries that properly implement the whole protocol. There are several protocols optimized for IoT use. The two best known are MQTT**6** and CoAP**7**. MQTT was invented in 1999 to solve issues in embedded systems and SCADA. It has been through some iterations and the current version (3.1.1) is undergoing standardization in the OASIS MQTT Technical Committee**8**. MQTT is a publish-subscribe messaging system based on a broker model. The protocol has a very small overhead (as little as 2 bytes per message), and was designed to support lossy and intermittently connected networks. MQTT was designed to flow over TCP. In addition there is an associated specification designed for ZigBee-style networks called MQTT-SN (Sensor Networks). CoAP is a protocol from the IETF that is designed to provide a RESTful application protocol modeled on HTTP semantics, but with a much smaller footprint and a binary rather than a text- based approach. CoAP is a more traditional client-server approach rather than a brokered approach. CoAP is designed to be used over UDP.

For the reference architecture we have opted to select MQTT as the preferred device communication protocol, with HTTP as an alternative option.

The reasons to select MQTT and not CoAP at this stage are

- Better adoption and wider library support for MQTT;
- Simplified bridging into existing event collection and event processing systems; and
- Simpler connectivity over firewalls and NAT networks

However, both protocols have specific strengths (and weaknesses) and so there will be some situations where CoAP may be preferable and could be swapped in. In order to support MQTT we need to have an MQTT broker in the architecture as well as device libraries. We will discuss this with regard to security and scalability later.

One important aspect with IoT devices is not just for the device to send data to the cloud/ server, but also the reverse. This is one of the benefits of the MQTT specification: because it is a brokeredmodel, clients connect an outbound connection to the broker, whether or not the device is acting asa publisher or subscriber. This usually avoids firewall problems because this approach works even behind firewalls or via NAT. In the case where the main communication is based on HTTP, the traditional approach for sending data to the device would be to use HTTP Polling. This is very inefficient and costly, both in terms of network traffic as well as power requirements. The modern replacement for this is the WebSocket protocol**9** that allows an HTTP connection to be upgraded into a full two-way connection. This then acts as a socket channel (similar to a pure TCP channel) between the server and client. Once that has been established, it is up to the system to choose an ongoing protocol to tunnel over the connection. For the reference architecture we once again recommend using MQTT as a protocol with WebSockets. In some cases, MQTT over WebSocketswill be the only protocol. This is because it is even more firewall-friendly than the base MQTT specification as well as supporting pure browser/JavaScript clients using the same protocol. Note that while there is some support for WebSockets on small controllers, such as Arduino, the combination of network code, HTTP and WebSockets would utilize most of the available code space on a typical Arduino 8-bit device. Therefore, we only recommend the use of WebSockets onthe larger 32-bit devices.

AGGREGATION/BUS LAYER

An important layer of the architecture is the layer that aggregates and brokers communications.This is an important layer for three reasons:

1. The ability to support an HTTP server and/or an MQTT broker to talk to the devices

2. The ability to aggregate and combine communications from different devices and to route communications to a specific device (possibly via a gateway)

3. The ability to bridge and transform between different protocols, e.g. to offer HTTP based APIsthat are mediated into an MQTT message going to the device.

The aggregation/bus layer provides these capabilities as well as adapting into legacy protocols.The bus layer may also provide some simple correlation and mapping from different correlation models (e.g. mapping a device ID into an owner_s ID or vice-versa). Finally the aggregation/bus layer needs to perform two key security roles. It must be able to act as an OAuth2 Resource Server(validating Bearer Tokens and

associated resource access scopes). It must also be able to act as a policy enforcement point (PEP) for policy-based access. In this model, the bus makes requests to the identity and access management layer to validate access requests. The identity and access management layer acts as a policy decision point (PDP) in this process. The bus layer then implements the results of these calls to the PDP to either allow or disallow resource access.

EVENT PROCESSING AND ANALYTICS LAYER

This layer takes the events from the bus and provides the ability to process and act upon these events. A core capability here is the requirement to store the data into a database. This may happenin three forms. The traditional model here would be to write a server side application, e.g. this could be a JAX-RS application backed by a database. However, there are many approaches where we can support more agile approaches. The first of these is to use a big data analytics platform. This is a cloud-scalable platform that supports technologies such as Apache Hadoop to provide highly scalable map reduce analytics on the data coming from the devices. The second approach isto support complex event processing to initiate near real-time activities and actions based on data from the devices and from the rest of the system.

Our recommended approach in this space is to use the following approaches:

- Highly scalable, column-based data storage for storing events
- Map-reduce for long-running batch-oriented processing of data
- Complex event processing for fast in-memory processing and near real-time reaction and autonomic actions based on the data and activity of devices and other systems
- In addition, this layer may support traditional application processing platforms, such as Java Beans, JAX-RS logic, message-driven beans, or alternatives, such as node.js, PHP, Ruby or Python.

CLIENT/EXTERNAL COMMUNICATIONS LAYER

The reference architecture needs to provide a way for these devices to communicate outside of the device-oriented system. This includes three main approaches. Firstly, we need the ability to create web-based front-ends and portals that interact with devices and with the event-processing layer. Secondly, we need the ability to create dashboards that offer views into analytics and event processing. Finally, we need to be able to interact with systems outside this network usingmachine-to-machine communications (APIs). These APIs need to be managed and controlled and this happens in an API management system. The recommended approach to building the web frontend is to utilize a modular front-end architecture, such as a portal, which allows simple fast composition of useful UIs. Of course the architecture also supports existing Web server-side technology, such as Java Servlets/ JSP, PHP, Python, Ruby, etc. Our recommended approach is based on the Java framework and the most popular Java-based web server, Apache Tomcat. The dashboard is a re-usable system focused on creating graphs and other

visualizations of datacoming from the devices and the event processing layer.

The API management layer provides three main functions:

- The first is that it provides a developer-focused portal (as opposed to the user focused portal previously mentioned), where developers can find, explore, and subscribe to APIs from thesystem. There is also support for publishers to create, version, and manage the available and published APIs;
- The second is a gateway that manages access to the APIs, performing access control checks (for external requests) as well as throttling usage based on policies. It also performs routing and load- balancing;
- The final aspect is that the gateway publishes data into the analytics layer where it is stored as well as processed to provide insights into how the APIs are used.

DEVICE MANAGEMENT

Device management (DM) is handled by two components. A server-side system (the devicemanager) communicates with devices via various protocols and provides both individualand bulk control of devices. It also remotely manages software and applications deployed on the device. It can lock and/or wipe the device if necessary. The device manager works in conjunction with the device management agents. There are multiple different agents for different platforms and device types. The device manager also needs to maintain the list of device identities and map these into owners. It must also work with the identity and access management layer to manage access controls over devices (e.g. who else can manage the device apart from the owner, how much control does the owner have vs. the administrator, etc.)

There are three levels of device: non-managed, semi-managed and fully managed (NM, SM, FM).

Fully managed devices are those that run a full DM agent. A full DM agent supports:

- Managing the software on the device
- Enabling/disabling features of the device (e.g. camera, hardware, etc.)
- Management of security controls and identifiers
- Monitoring the availability of the device
- Maintaining a record of the device_s location if available
- Locking or wiping the device remotely if the device is compromised, etc.

Non-managed devices can communicate with the rest of the network, but have no agent involved. These may include 8-bit devices where the constraints are too small to support the agent. The device manager may still maintain information on the availability and location of the device if this is available.

Semi-managed devices are those that implement some parts of the DM (e.g. feature control, but not software management).

IDENTITY AND ACCESS MANAGEMENT

The final layer is the identity and access management layer. This layer needs to provide thefollowing services:

- OAuth2 token issuing and validation
- Other identity services including SAML2 SSO and OpenID Connect support for identifyinginbound requests from the Web layer
- XACML PDP
- Directory of users (e.g. LDAP)
- Policy management for access control (policy control point)

The identity layer may of course have other requirements specific to the other identity and access management for a given instantiation of the reference architecture. In this section we have outlined the major components of the reference architecture as well as specific decisions we have taken around technologies. These decisions are motivated by the specific requirements of IoT architectures as well as best practices for building agile, evolvable, scalable Internet architectures. Of course there are other options, but this reference architecture utilizes proven approaches that are known to be successful in real-life IoT projects we have worked on.

1.13 Physical and Logical Design of IoT -From M2M to IoT

- Physical Design of IoT system refers to IoT Devices and IoT Protocols. Things are Nodedevice which have unique identities and can perform remote sensing, actuating and monitoring capabilities. Communication established between things and cloud based server over the Internet by various IoT protocols.
- Logical design of IoT system refers to an abstract representation of the entities &processes without going into the low-level specifies of the implementation.

- Physical Design of IoT

Physical Design of IoT refers to IoT Devices and IoT Protocols. Things are Node device which have unique identities and can perform remote sensing, actuating and monitoring capabilities. IoTProtocols helps Communication established between things and cloud based server over the Internet.

Things:

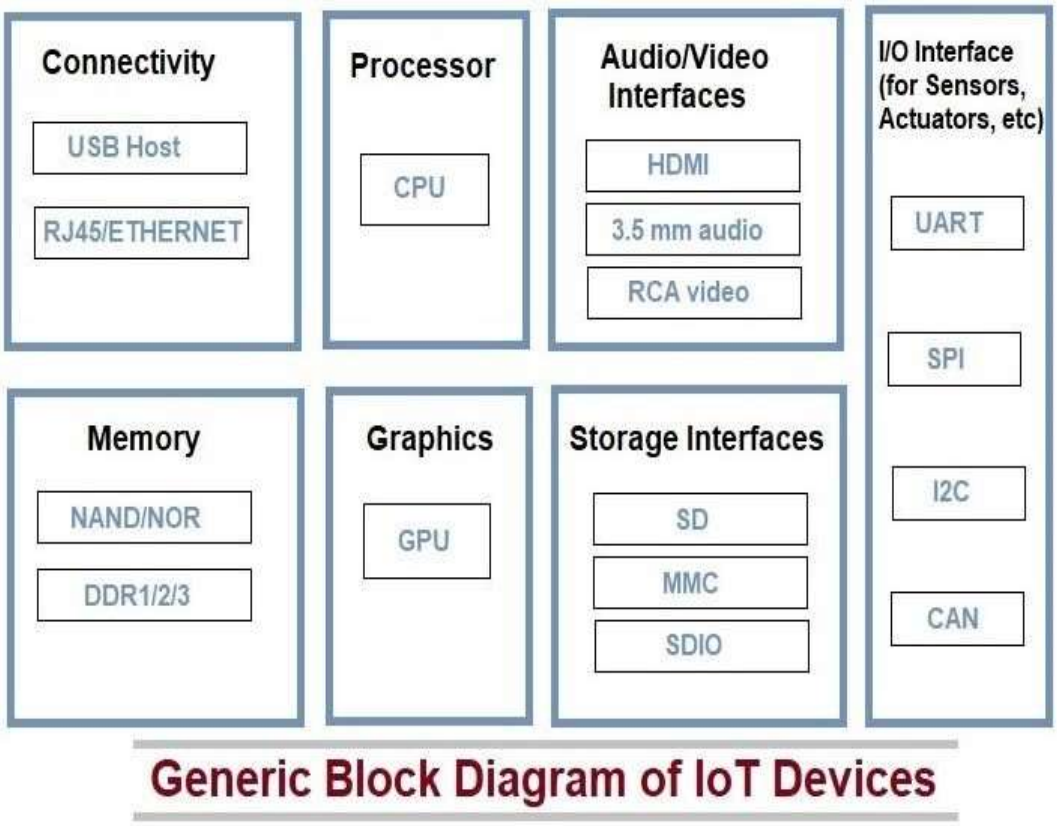


Fig 1.10. Generic block diagram

For example, Temperature data generated by a Temperature Sensor in Home or other place, when processed canhelp in determining temperature and take action according to users. Above picture, shows a generic block diagram of IoT device. It may consist of several interfaces for connectionsto other devices. IoT Device has I/O interface for Sensors, Similarly for Internet connectivity, Storage and Audio/Video.

IoT Protocols: IoT protcols help to establish Communication between IoT Device (Node Device) and Cloud based Server over the Internet. It help to sent commands to IoT Device and received data from an IoT device over the Internet. An image is given below.

By this image you canunderstand which protocols used.

Link layer protocols determine how data is physically sent over the network‘s physical layer or medium (Coxial calbe or other or radio wave). Link Layer determines how the packets are coded and signaled by the hardware device over the medium to which the hostis attached (eg. coxial cable).

Network Layer Responsible for sending of IP datagrams from the source network to the destination network. Network layer performs the host addressing and packet routing. Weused IPv4 and IPv6 for Host identification. IPv4 and IPv6 are hierarchical IP addrssing schemes.

Transport Layer This layer provides functions such as error control, segmentation, flowcontrol and congestion control. So this layer protocols provide end-to-end message transfer capability independent of the underlying network.

Application Layer Application layer protocols define how the applications interface with the lower layer protocols to send over ther network. (Hypertext Transfer Protocol (HTTP), CoAP-Constrained Application

Protocol, MQTT is a machine-to-machine (M2M)/‖Internet of Things‖ XMPP : Extensible Messaging and Presence Protocol (XMPP), DDS : The Data Distribution Service (DDS™), AMQP : The AMQP – IoT protocols)

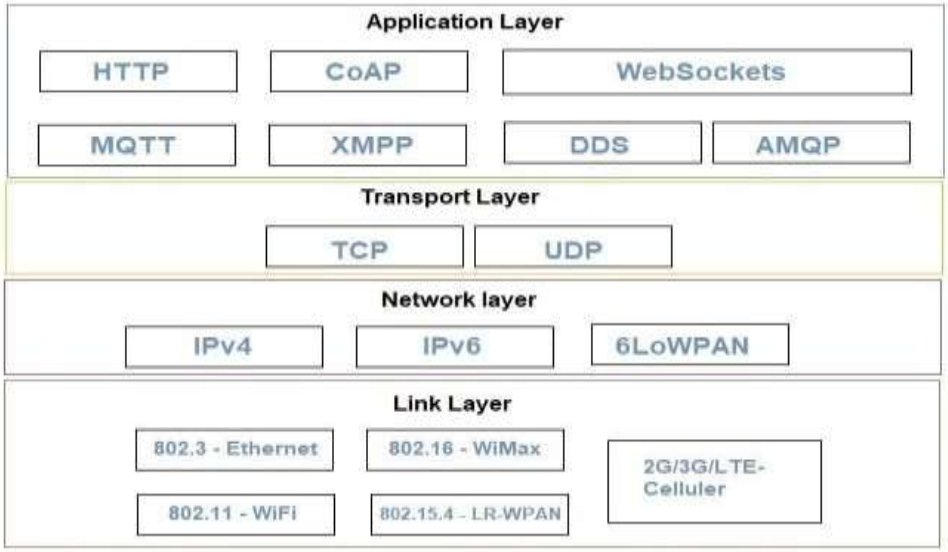


Fig 1.11: IoT Protocols

Logical Design of IoT:

In this we discuss Logical design of Internet of things. Logical design of IoT system refers to anabstract representation of the entities & processes without going into the low-level specifies of the implementation. Forunderstanding Logical Design of IoT, we describes given below terms.

IoT Functional Blocks
IoT Communication Models
IoT Communication APIs

IoT Functional Blocks:

An IoT system comprises of a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication and management.

functional blocks are:

Device: An IoT system comprises of devices that provide sensing, actuation, monitoring and control functions.

Communication: Handles the communication for the IoT system.

Services: services for device monitoring, device control service, data publishing services and services for device discovery.

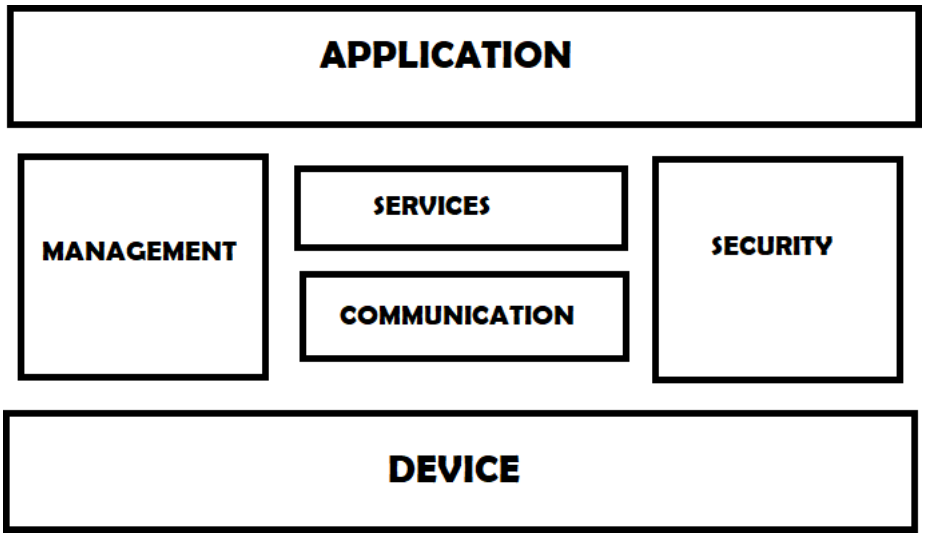


Fig 1.12: IoT Services

Management: this blocks provides various functions to govern the IoT system.

Security: this block secures the IoT system and by providing functions such as authentication , authorization, message and content integrity, and data security.

Application: This is an interface that the users can use to control and monitor various aspects of the IoT system. Application also allow users to view the system status and view or analyze the processed data.

IoT Communication Models:

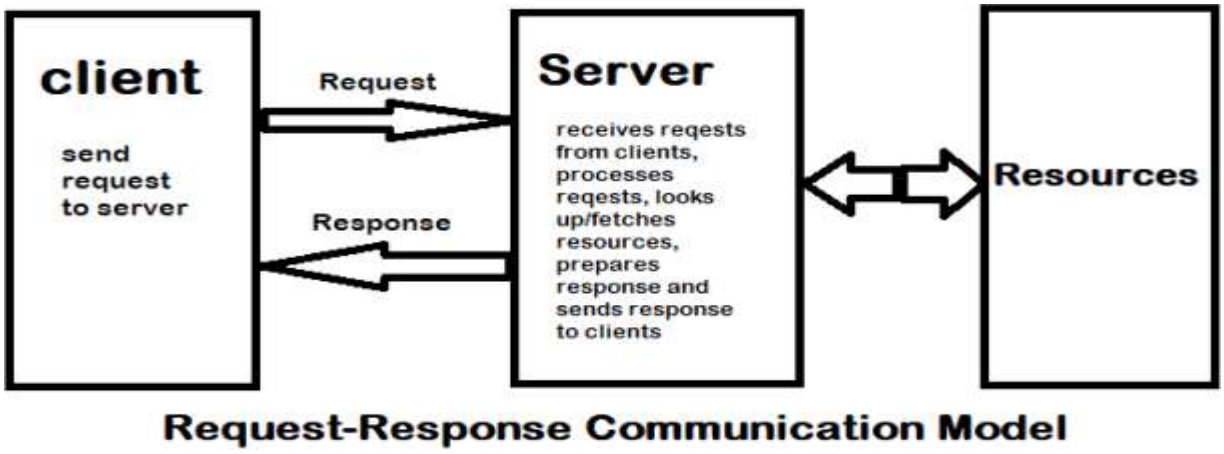


Fig 1.13: Request Response Communication model

Request-Response Model

Request-response model is communication model in which the client sends requests to the server and the server responds to the requests. When the server receives a request, it decides how to respond, fetches the data, retrieves resource representation, prepares the response, and then sends the response to the client. Request-response is a stateless communication model and each request-response pair is independent of others.

HTTP works as a request-response protocol between a client and server. A web browser may be theclient, and an application on a computer that hosts a web site may be the server.

Example: A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

Publish-Subscribe Model:

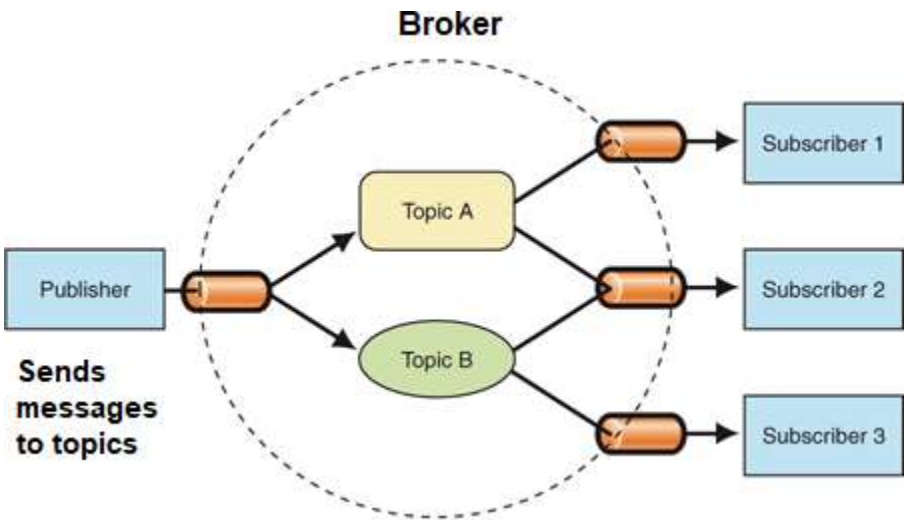


Fig 1.14: Publish-Subscribe model

Publish-Subscribe is a communication model that involves publishers, brokers and consumers. Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics which are managed by the broker. When the broker receive data for a topic from the publisher, it sends the data to all the subscribedconsumers.

Push-Pull Model:

Push-Pull is a communication model in which the data producers push the data to queues and the consumers

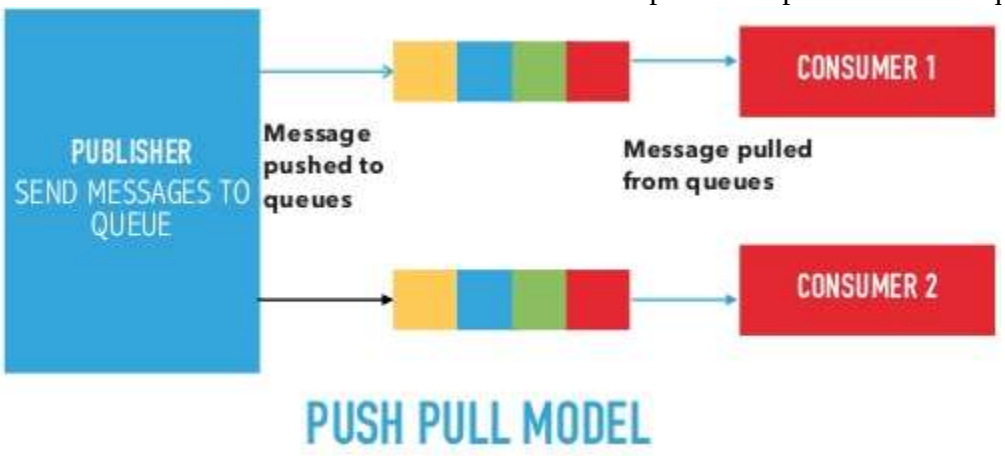


Fig 1.15: Push Pull model

Pull the data from the Queues. Producers do not need to be aware of the consumers. Queues help in decoupling the messaging between the Producers and Consumers. Queues also act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data andthe rate rate at which the consumer pull data.

Exclusive Pair Model:

Fig 1.16: Exclusive pair model

Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and server. Connection is setup it remains open until the client sends a request to close the connection. Client and server can send messages to each other after connection setup. Exclusive pair is stateful communication model and the server is aware of all the open connections.

Generally we used Two APIs For IoT Communication. These IoT Communication APIs are:

- REST-based Communication APIs
- WebSocket-based Communication APIs

REST-based Communication APIs:

Representational state transfer (REST) is a set of architectural principles by which you can design Web services the Web APIs that focus on systems‘s resources and how resource states are addressed and transferred. REST APIs that follow the request response communication model, the rest architectural constraint apply to the components, connector and data elements, within a distributed hypermediasystem.

WebSocket based communication API:

Websocket APIs allow bi-directional, full duplex communication between clients and servers. Websocket APIs follow the exclusive pair communication model. Unlike request-response model such asREST, the WebSocket APIs allow full duplex communication and do not require new coonection to be setup for each message to be sent. Websocket communication begins with a connection setup request sentby the client to the server. The request (called websocket handshake) is sent over HTTP and the server interprets it is an upgrade request. If the server supports websocket protocol, the server responds to the websocket handshake response. After the connection setup client and server can send data/mesages to each other in full duplex mode. Websocket API reduce the network traffic and latency as there is no overhead for connection setup and termination requests for each message. Websocket suitable for IoT applications that have low latency or high throughput requirements. So Web socket is most suitable IoT Communication APIs for IoT System.

1.14 Software defined Network:

Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today‘s applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The OpenFlow® protocol is a foundational element for building SDN solutions. For an in-depth understanding of SDN-based networking and use cases, check out the open source micro-book, ―Software-Defined Networks: A Systems Approach‖.

SDN Architecture :

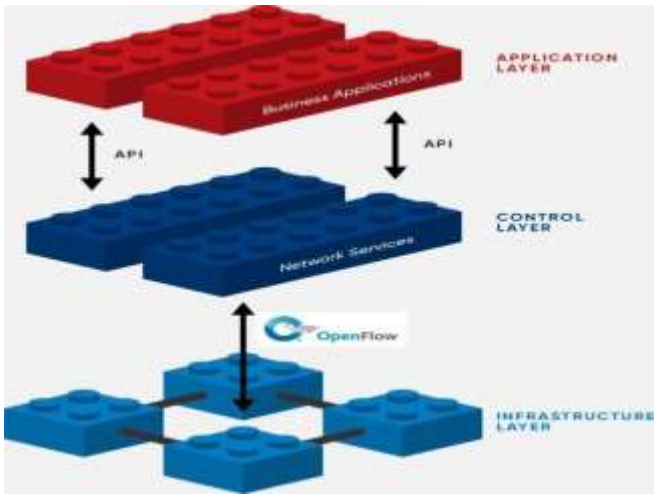


Fig 1.17: SDN Architecture

DIRECTLY PROGRAMMABLE:

Network control is directly programmable because it is decoupled from forwarding functions.

AGILE: Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flowto meet changing needs.

CENTRALLY MANAGED:

Network intelligence is (logically) centralized in software-based SDN controllers that maintain aglobal view of the network, which appears to applications and policy engines as a single, logicalswitch.

PROGRAMMATICALLY CONFIGURED:

SDN lets network managers configure, manage, secure, and optimize network resources very quicklyvia dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.

OPEN STANDARDS-BASED AND VENDOR-NEUTRAL:

When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

TEXT / REFERENCE BOOKS

1. Boswarthick,Omar Elloumi., The Internet of Things: Applications and Protocols, Wiley publications., 2012
2. Dieter Uckelmann, Mark Harrison, Florian Michahelles., Architecting the Internet of Things, Springer publications.2011
3. Marco Schwatrz Internet of Things with Arduino Cookbook, Packt Publications.2016 .
4. Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, “From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence”, 1st Edition, Academic Press, 2014.
5. Vijay Madisetti, Arshdeep Bahga, “Internet of Things: A Hands-On Approach” published by Vijay Madisetti 2014

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – 2 – Introduction to IoT – SCSA1308

UNIT 2-ELEMENTS OF IOT

Application Sensors & Actuators - Edge Networking (WSN) – Gateways - IoT Communication Model – WPAN & LPWA, Overview of IoT supported Hardware platforms such as: Raspberry pi, ARM Cortex Processors, Arduino and Intel Galileo boards, Wearable Development Boards

2.1.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 is 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 that would be used to 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. There are 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 discussed is 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 (thebrains 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). Sensormeans that it is part of a bigger system which provides input to a main control system (like aProcessor 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 datain digital sensors, which is used for conversion and transmission, is digital in nature.

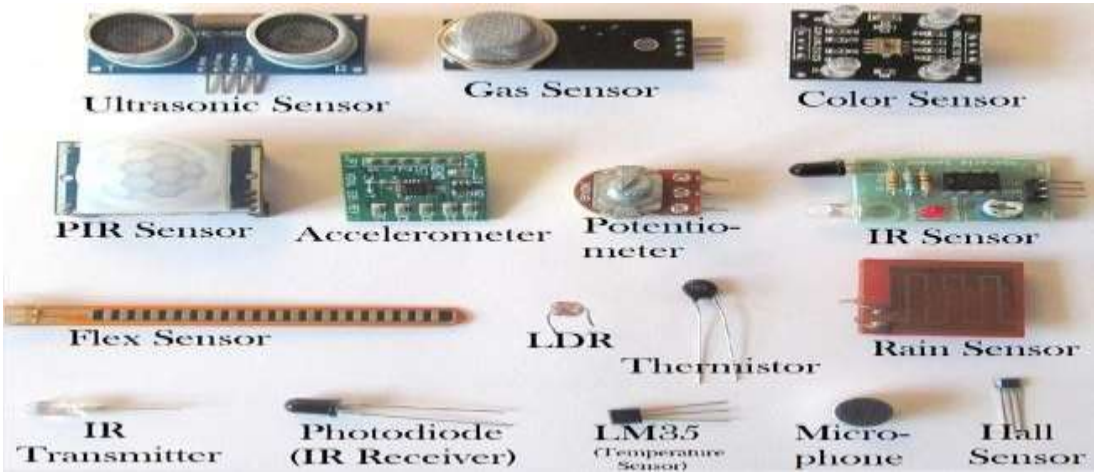


Fig 2.1. Examples of Sensors

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**.**

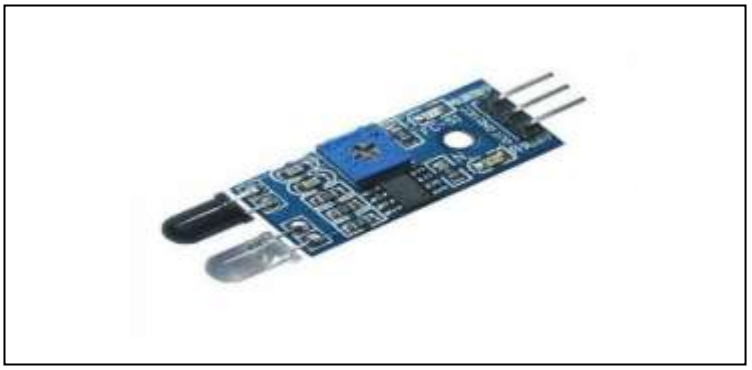
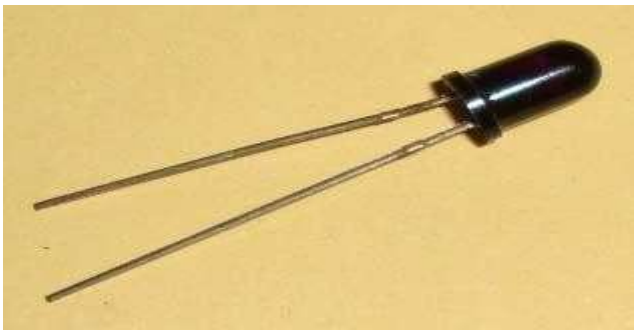


Fig 2.2. LED sensor

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 it is known, a PN junction allows small leakage currents when itis 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 have applied. So, in this way, a

Fig 2.3.Photo diode



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.

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 Infraredor Laser), Ultrasonic, Hall Effect, Capacitive, etc.

Fig 2.4.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 resistancedecreases 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.

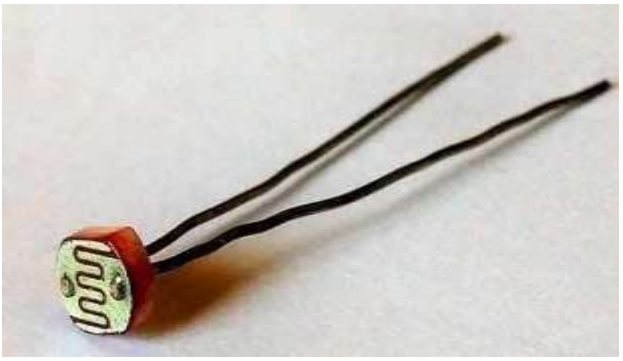


Fig 2.5.LDR

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 2.6.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.

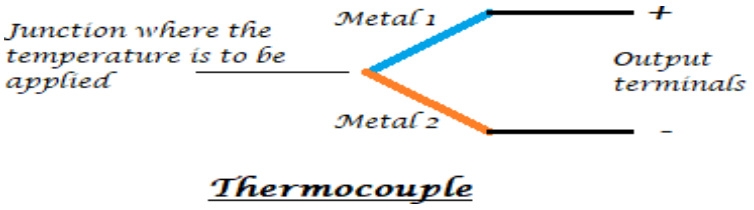


Fig 2.7.Thermo couple

7. Strain Gauge (Pressure/Force Sensor)

A strain gauge is used to detect pressure when a load is applied**.** It works on the principleof resistance, 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 beused 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.

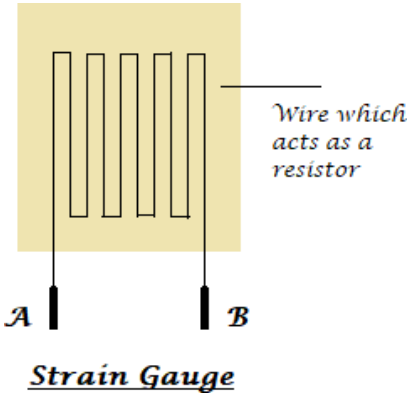


Fig 2.8.Strain Guage

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 2.9.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 2.10.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 2.11.Encoder

11 Hall Sensor

The name itself states that it is the sensor which works on the Hall Effect. It can be definedas 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 2.12.Hall sensor

The hall sensor is used in few smartphones which are helpful in turning off the screenwhen 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 it is read them. The controller at hand can only read the changes in voltage and nothing less, for this, voltage divider circuit is used, with that it can be derived the resistance change as a voltage change.

Fig 2.13. 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 manytypes 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 producesproportional 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 2.14.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 human can‘t sense these sound signals. Only specificspeakers 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 Trigger‘ pin was made 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. Calculate thedistance of the object based on the time taken by the ultrasonic wave to return back to the sensor.

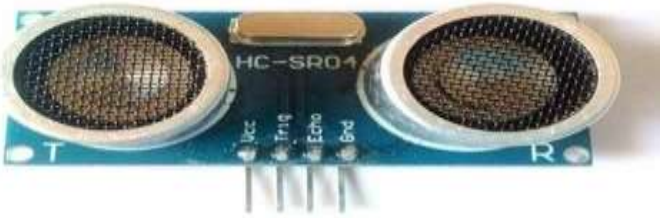


Fig 2.15.Utrasonic sensor

There are many applications with the ultrasonic sensor. It is used to 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 theultrasonic sounds. So, ultrasonic waves can be used as mosquito repellent.

15.Touch Sensor

In this generation, almost all are using smartphones which have widescreenthat too a screen which can sense user 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 thescreen both of these are separated by an air gap with a small voltage applied to the sheets. When human 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.

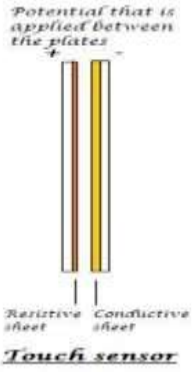


Fig 2.16.Touch sensor

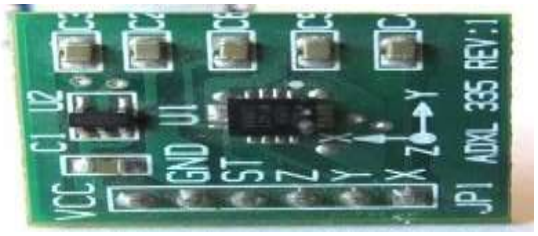
16.PIR sensor

PIR sensor stands for **Passive Infrared sensor**. These are used to detect the motion of humans, animals or things. Infrared rays have a property of reflection. When an infrared ray hits an object, depending upon the temperature of the target the infrared rayproperties 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 2.17.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 2.18.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 nosuch 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 2.19.Gas Sensor

So finally, it is concluded that sensors are not only used to make the 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 thatcan 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 2.20 Resistive Sensors

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 2.21.Voltage Generating Sensors

21.Switch Sensors

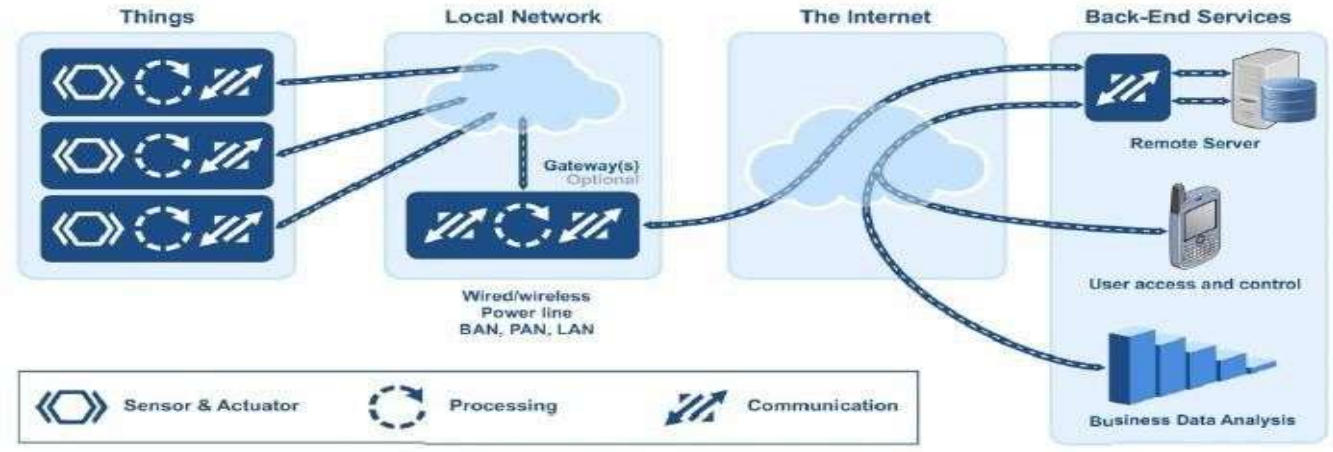
Switch sensors are composed of a set of contacts that open when close to a magnet. A reed switch is a common example of a switch sensor and is most commonly used as a speed or position sensor. As a speed sensor, a magnet is attached to the speedometer cable and spinsalong with it. Each time one of the 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 2.22.Switch Sensors

2.Edge Networking

Embedded systems are already playing a crucial role in the development of theIoT. 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 proprietarycommunication protocols to Internet Protocol
3. The Internet
4. Back-End Services; enterprise data systems, or PCs and mobile devices



The Internet of Things from an embedded systems point of view

Fig 2.23.Embedded Point of View

Internet of Things are separated in two broad categories:

.

1. Industrial IoT, where the local network is based on any one of many differenttechnologies. 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 orEthernet (wired or wireless). The IoT device will typically communicate only with local devices.

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

Local Network

choice of communication technology directly affects device‘s hardware requirements and costs. IoT devices are deployed in so many different ways — in clothing, houses, buildings, campuses, factories, and even in human body — that no single networking technology can fitall bills.

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

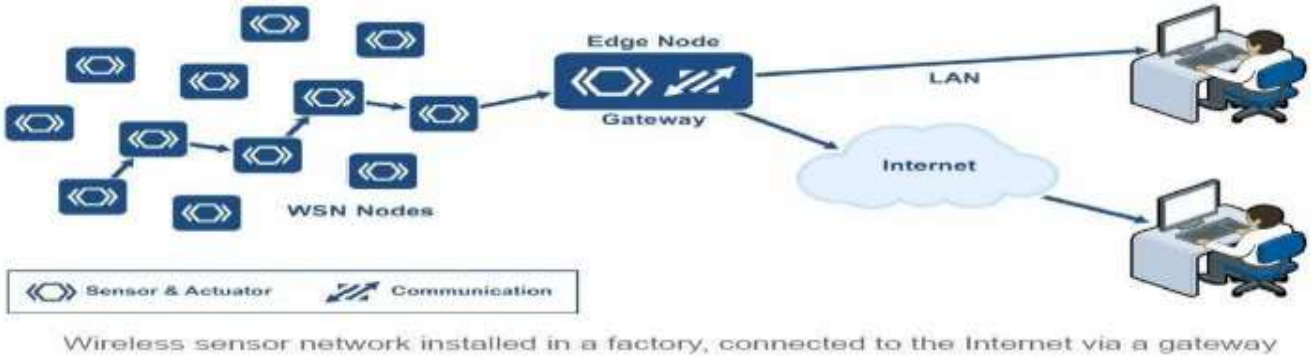


Fig 2.24. Wireless Sensor Network Architecture

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 alsooperate at low power so that they can run on battery, or even use energy harvesting. A WSNnode 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.

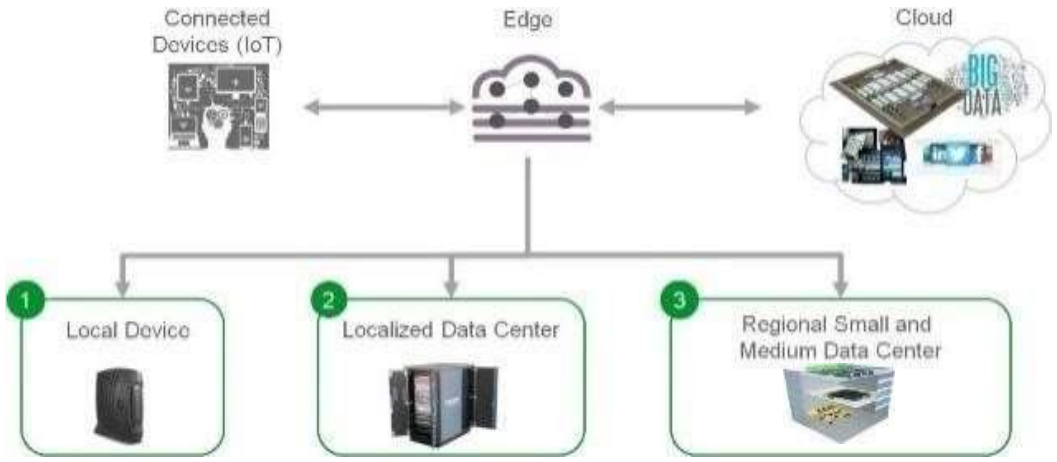


Fig 2.25.WSN Edge

WSN Technologies

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

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 30meters in buildings and up to 300 meters outdoors.

EnOcean wireless technology uses a combination of energy harvesting and very low powerwireless 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 it is expected 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 devicesconnect and communicate in terms of their technical communication models. In March 2015, the Internet Architecture Board (IAB) released a guiding architecturaldocument 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.

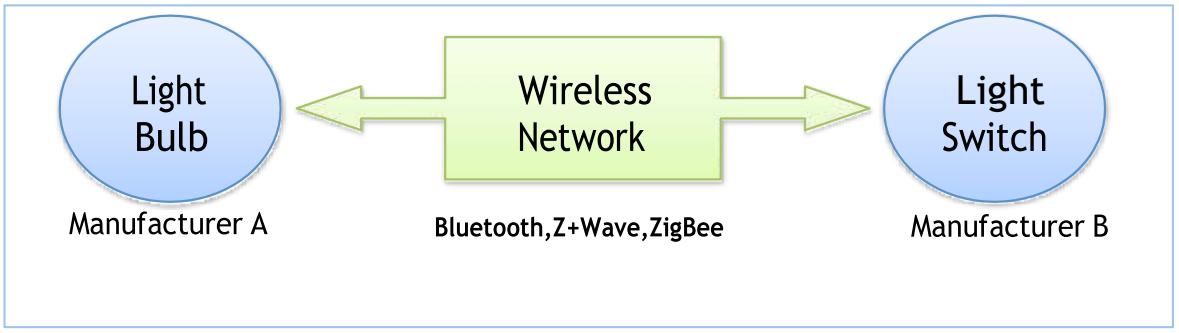


Fig 2.26.Example of device-to-device communication model

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 connectionsto establish a connection between the device and the IP network, which ultimately connects to the cloud service. This is shown in Figure 27.

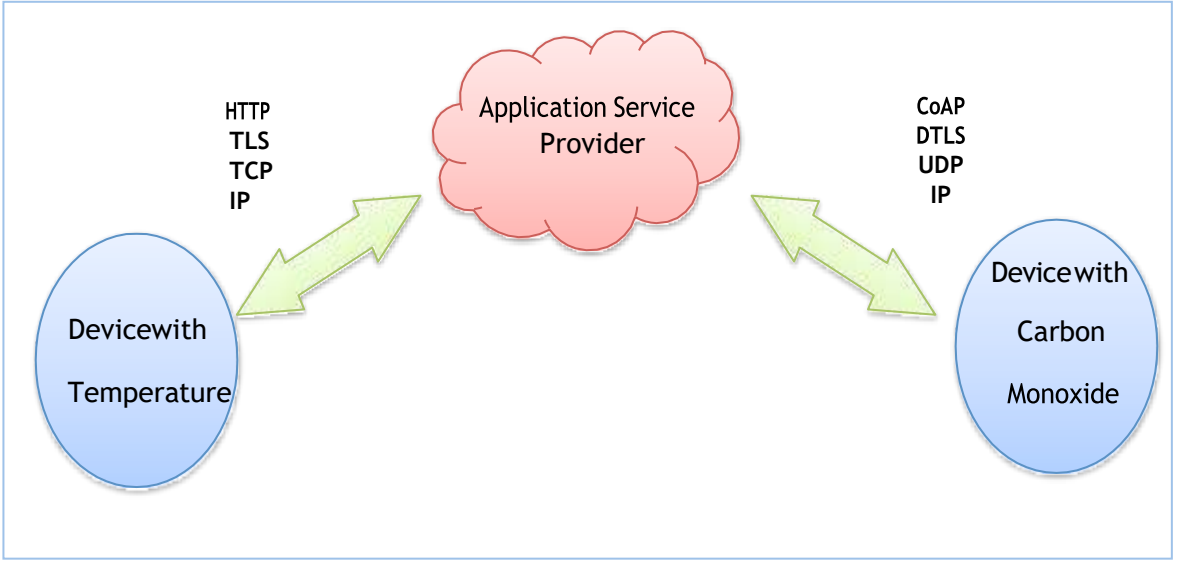


Fig 2.27.Device-to-cloud communication model diagram.

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

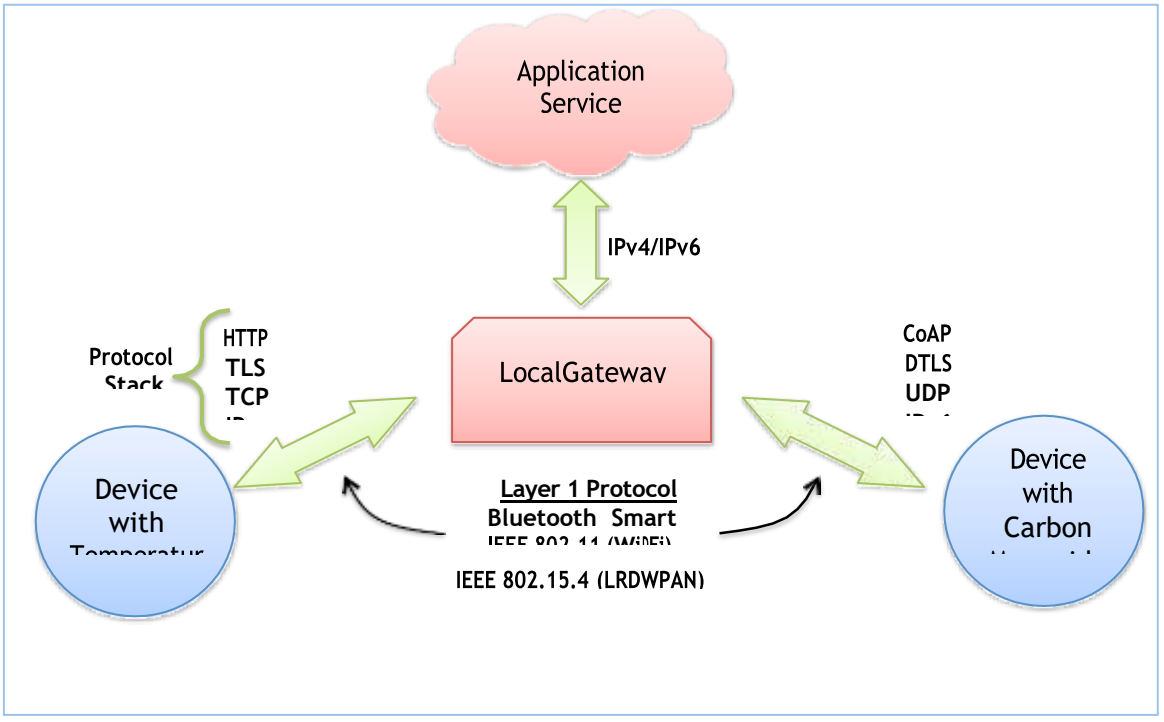


Fig 2.28.Device-to-gateway communication model

smartphone or Web interface, and it also supports software updatesto 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 thecapabilities of the device beyond its native features.

However, interoperability challenges can arise when attempting to integrate devicesmade 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 aconduit 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.
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 enablesusers 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 inconsolidating and analyzing the energy consumption and utilities data produced by all the IoT sensors and Internet-enabled utility systems on the premises. Often inthe 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 architecturefacilitates data portability needs. Effective back-end data- sharing architecturesallow 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.

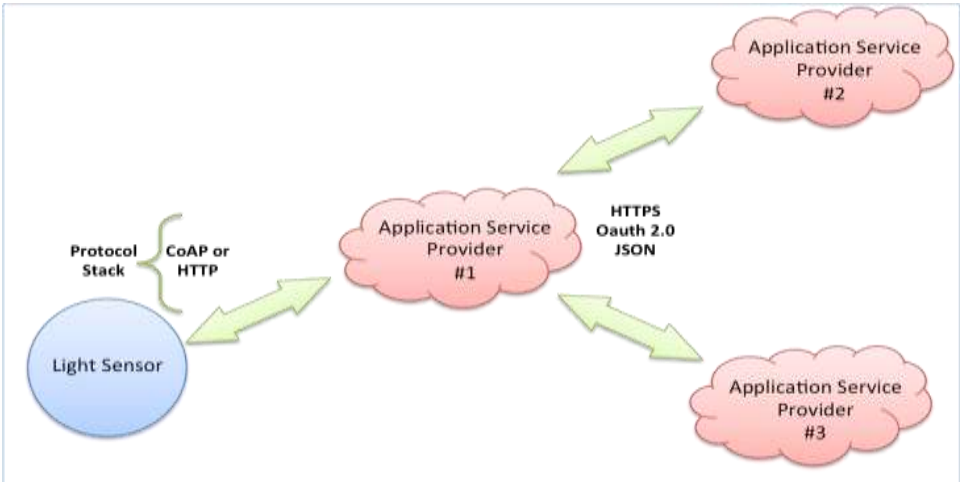


Fig 2.29.Back-end data sharing model

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 theability 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. Bycreating data communication conduits to the cloud, users, and service providerscan 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, arecharacterized 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 ofsecond 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

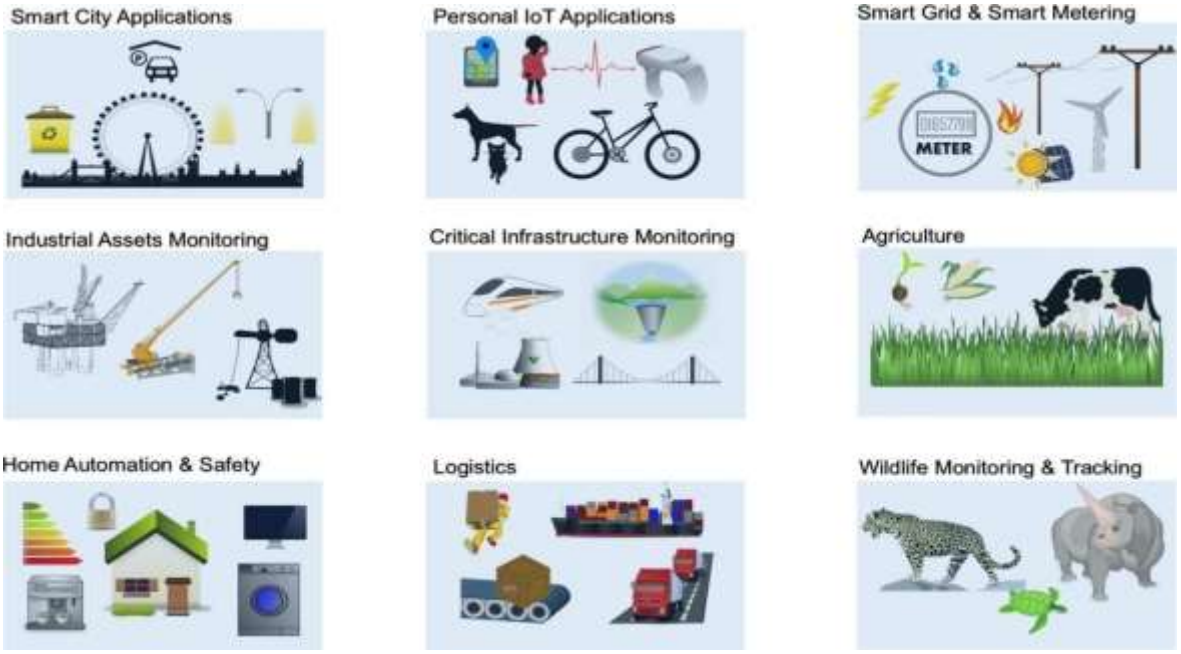


Fig 2.30.Applications of LPWA technologies acrossdifferent sectors

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 fewmonths 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. Onlywhen 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 complexfor 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. The challenges are highlighted and analyzed 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, thedevice density may vary significantly across different geographical areas, creatingthe so called *hot-spot* problem. These hot-spots will put the LPWA base stations to astress 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 monitorlink 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 farno 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 or more LPWA technologies under same conditions. Opinion is that, it is a significant barrier to entry for potential customers. Providing LPWA technologies as a scientific instrumentation for general public through citygovernments 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:

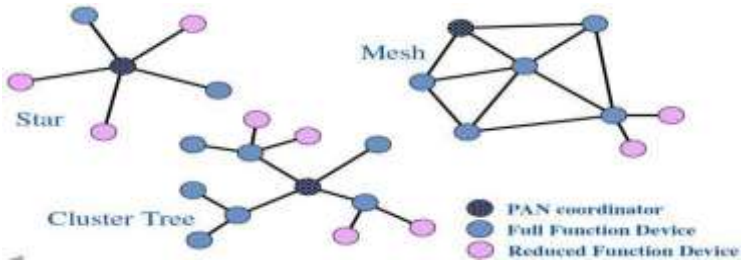


Fig 2.31 WPAN TopologiesIEEE 802.15 WPAN Standards:

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 1Mbps (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. AnyBluetooth 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 ProtocolStack 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.

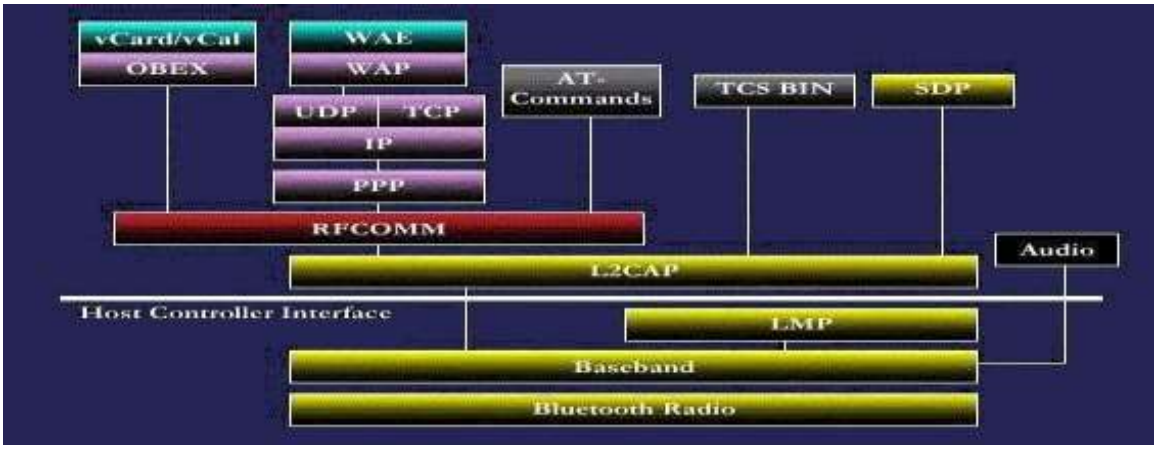


Fig 2.32 Bluetooth Protocol Stack

Link Manager layer

The link manager layer specifies the establishment and release links, authentication, trafficscheduling, 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 DiscoveryProtocol (SDP) RFCOMM Telephony Control Protocol Specification (TCS) TCP/IP basedapplications, for instance information transfer using the Wireless Application Protocol(WAP), can be extended to Bluetooth devices by using the Point-to-Point Protocol (PPP) ontop 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-controlsignaling 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 printjob) 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 that can be used for the 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 connectsensors 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** that are used currently. With the ability to handle the vast amount of data using Cloud IoT Core,Google stands out from the rest. Can get advanced analytics owing to Google‘s Big Queryand Cloud Data Studio.

Some of the features of the Google Cloud platform are:-

- Accelerate the Business
- Speed up the 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

It can never be expected 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-AylaNetwork by Ayla (New)
Top IoT Platforms-Bosch IoT Suite(New)
IoT Platform: Carriots
IoT Platform: Oracle IntegratedCloud 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, applications can keep track of and communicate with all 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 to collect telemetry data from multiple devices, and store and analyze the data. User can also create applications that enable 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. User can use either the MQTT protocol directly or MQTT over WebSocket to publish and subscribe. User can use the HTTP REST interface to publish. Rules engine Provides message processing and integration with other AWS services. User 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. User can also use the message broker to republish messages to other subscribers.

C. Security and Identity service:
Provides shared responsibility for security in the AWS Cloud. 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.

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

E. Group registry

Groups allow user to manage several devices at once by categorizing them into groups. Groups can also

contain groups—user can build a hierarchy of groups. Any action user perform on a parent group will apply to its child groups, and to all the devices in it and inall 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.

F. Device shadow

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

G. Device Shadow service

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

H. Device Provisioning service

Allows user to provision devices using a template that describes the resources required for user 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). User can use the same template to provision multiple devices just by passing in different values for the template variables in the dictionary.

I. Custom Authentication service

User can define custom authorizers that allow to manage own authentication and authorization strategy using a custom authentication service and a Lambda function. Custom authorizers allow AWS IoT to authenticate 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.

J. Jobs Service

Allows user to define a set of remote operations that are sent to and executed on one or moredevices connected to AWS IoT. For example, user 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, user 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 user devices:

- **AWS Command Line Interface (AWS CLI)**—Run commands for AWS IoT on Windows, macOS, and Linux. These commands allow user 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 IoT applications using HTTP or HTTPS requests. These API actions allow user 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 IoT applications using language-specific APIs. These SDKs wrap the HTTP/HTTPS

API and allow user 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. Formore 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 user code on virtual servers from Amazon EC2 in response toevents. 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 user Devices

AWS IoT Core allows user to easily connect devices to the cloud and to other devices. AWSIoT 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 evenif they are using different protocols

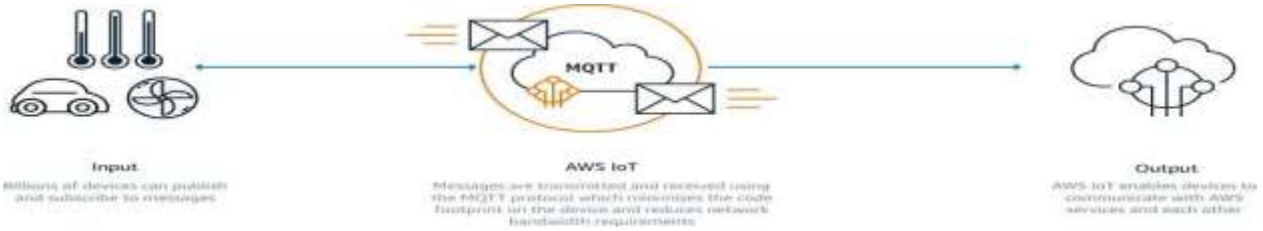


Fig 2.33. IOT Device Management

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, user can secure access to user devices and applications by applying policies with granular permissions

Fig 34.Secure Connection PROCESS AND ACT UPON DEVICE

DATA

With AWS IoT Core, user can filter, transform, and act upon device data on the fly, basedon business rules user defines. Can update 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

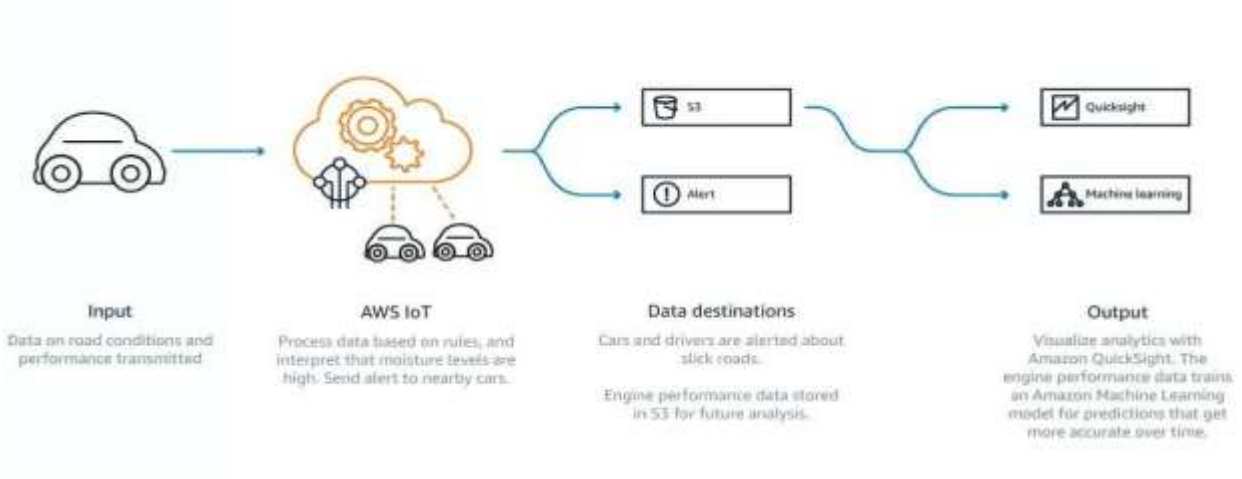


Fig 2.35. Process And Act Upon Device Data Read and set device state at any time

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 user applications as if it were online all the time. This means that application can read a device's state even when it is disconnected, and also allows user to set a device state and have it implemented when the device reconnects

Fig 2.36. 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 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 as1.Arduino uno or clone
2. PIR Sensor3.breadboard4.led(any colour) 5.buzzer
The following fig 26 shows the pin description of PIR sensor.

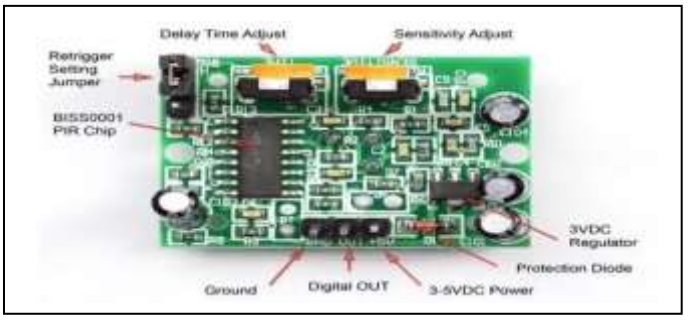


Fig 2.37.Arduino BoardFeatures and Electrical Specification

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 outputHigh sensitivity Delay time : 5s-18 minute
Blockade time : 0.5s-50s (acquiescently 0seconds) 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 followsin this way At the starting declare the pins which are going to be used

int PIR_output=5; // output of pirsensor 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 outputSerial.begin(9600);//serial communication between arduino and pc**
Download the full program from above PIR_sensor_by_kj_electronics file andupload 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. user can determine how far away something is by the time ittakes 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

Connect Trig and Echo to whichever pins want, 9 and 10 are just the ones

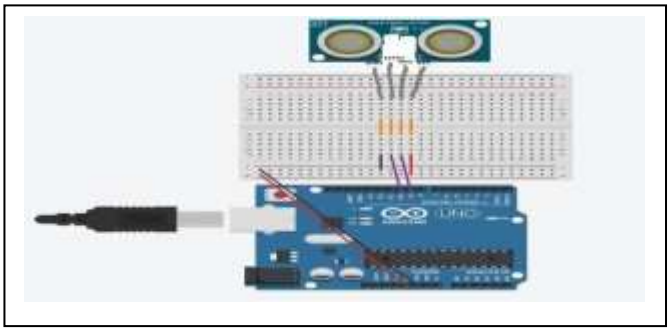


Fig 2.38.Ultra Sonic Sensor

Code:

define the pins that Trig and Echo are connected to.

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

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

```
float duration, distance;
```

Next, in the setup, 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 to do is to first set the trigPin low for 2 microseconds just to make sure that the pin in low first. Then, set it high for 10 microseconds, which sends out an8 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, use a handy Arduino function called pulseIn(). It takes 2 arguments, the pin user are listening to(In this case, the Echo pin), anda state(HIGH or LOW). What the function does is waits for the pin to go whichever sate user put in, starts timing, and then stops timing when it switches to the other state. In this case put HIGH since need to start timing when the Echo pin goes high. Store the time in the duration variable. (It returns the time in microseconds)

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

Now got the time, use the equation speed = distance/time, but make time x speed = distance because speed is there. 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, 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. So do the math, but searching it is easier. Anyway, with that information, 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). store that in the distance variable.

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

The rest is just printing out the results to the SerialMonitor. Serial.print("Distance: ");

```
Serial.println(distance);
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 tobe handled properly. Be careful of static electricity when handling them and make sure thepower supply is connected up correctly and is between 2.7 and 5.5V DC - so don't try to usea 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. user can also solder to the pins to connect long wires.

Reading the Analog Temperature Data

Unlike the FSR or photocell sensors analyzed, 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 user 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 user can use this with a 3.3v supply just as easily. No matter what supply user use, the analog voltage reading will range from about 0V (ground) to about 1.75V.
If 5V Arduino is used, and connecting the sensor directly into an Analog pin, 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 using a 3.3V Arduino, use the below formula:

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

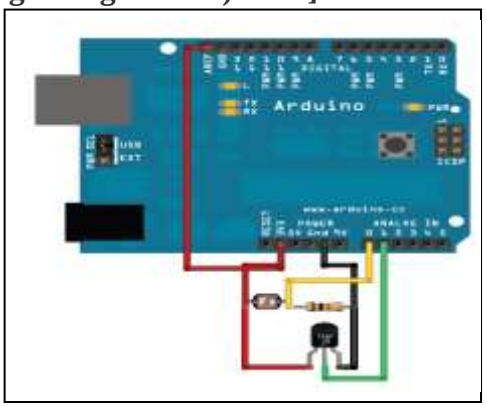


Fig 2.39.Temperature Sensor

Arduino Sketch - Simple Thermometer

This example code for Arduino shows a quick way to create a temperature sensor, itsimply 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 moreprecise and less noisy

```
int sensorPin = 0;
 * setup() - this function runs once when Arduino is 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 temperaturesensor int reading = analogRead(sensorPin);
// converting that reading to voltage, for 3.3v arduinouse 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 mVoffset//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 GPIOlibrary import time    #Import time library
```

```
GPIO.setmode(GPIO.BOARD)          #Set GPIO pinnumbering 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 isHIGH 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.BCM)

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 ToSettle" 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 theday. 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 withminimum 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 highestperformance in spite of low-end microcontroller.
- Smartphone application development on Android and iPhone platforms to allowconnectivity 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 tocloud 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 powermanagement.
- Certifications to ensure radiations do not affect human body.
- Expertise in driver development for developing optimized drivers for new types ofsensors.
- 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 user 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 functionslike 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 fora 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, motionsensors, proximity sensors, magnetometers etc.
- Driver development for communication interfaces like BLE 4.0, NFC, RF and evenusing 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 ofaccelerometer,

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 applicationdevelopment. Some of the aspects of smartphone application development include:

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

Cloud / web application development services

With applications hosted on platforms like Amazon, Microsoft and similar, availability anduptimes 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 andJBOSS.
- 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. user 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 2.40. CHIP

2. Mediatek Linkit One

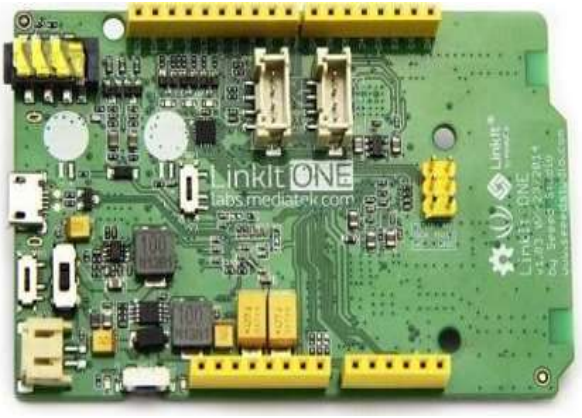


Fig 2.41. Linkit

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

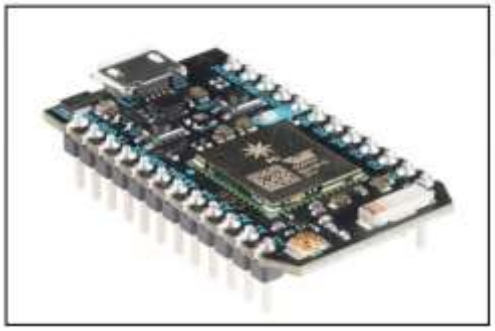


Fig 2.42.Particle Photon

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 user write sketches that arecompatible with Arduino

3.Tessel

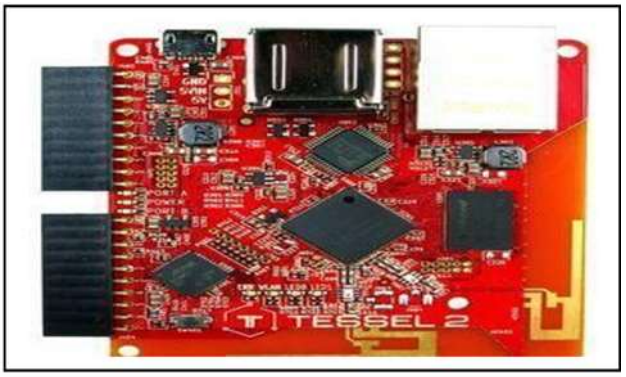


Fig 2.43.Tessel

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 andapparel. It comes with a thin, sewable, conductor thread which acts as the wire that connectsthe 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. user can use the same Arduino IDE with that user may already be familiar.

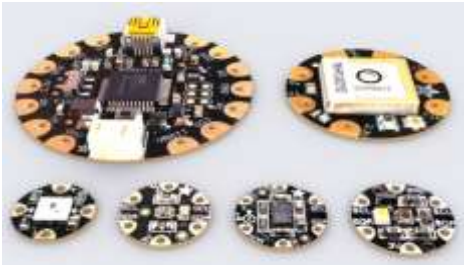


Fig 2.44.Adafruit Flora

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 2.45.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 punchwith an embedded 9-axis motion sensors and a Wi-Fi + Bluetooth 4.0 module. user 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. user can install Node.js, Python, and even Java on Udoo Neo.

Fig 2.46.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/outputpins, 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 2.47. 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, user can connect many peripherals and accessories to the Pi.

Fig 2.48: 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.

Fig 2.49: Arduino uno

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. Eager to learn basics of electronics and IoT, look no further, get an Arduino Uno R3.

ARM CORTEX

The ARM Cortex-M3 processor was the first product to be developed in the Cortex-M processor family. Microcontrollers based on the Cortex-M3 were first released in 2006, and now there are five processors in the Cortex-M processor family, with 12 microcontroller vendors supplying microcontrollers based on the Cortex-M processors and thousands of devices available. Different Cortex-M processor products support different ranges of instruction set. The Cortex-M0, CortexM0+ and Cortex-M1 processors are all based on the ARMv6-M architecture. The Cortex-M3 and CortexM4 are based on ARMv7-M architecture, which has a larger instruction set. For general data processing and normal I/O control tasks, the ARMv6-M architecture is sufficient and provides the best energy efficiency. For more complex data handling, the processors from the ARMv7-M architecture such as the Cortex-M3 provide additional instructions which accelerate data processing, as well as providing additional instructions for hardware divide, bit field processing and Multiply-Accumulate (MAC). In more demanding data processing applications like Digital Signal Processing (DSP), the Cortex-M4 processor provides further instructions such as SIMD to enhance DSP performance. Some of the Cortex-M4 devices also include a floating point unit (i.e. Cortex-M4F) which provides optimized single precision floating calculation in the floating point hardware.

Meeting the requirements of the IoT applications

The ARM Cortex-M processors are well suited to a wide range of applications, including IoT applications. In most typical IoT applications we can find the following processor system requirements:

- Low power
- High code density
- High performance
- Memory space and addressing modes
- OS friendly
- Security

Low power

From the users point of view many IoT systems are "always on". In practice the processors or microcontrollers could be in sleep mode most of the time, and wake up regularly to decode received Ethernet data, process the required data and for OS task scheduling. Although many of them might be connected to the main electricity supply all the time, the energy consumption of the system can be considerable over a long period of time. The power wasted can be significant if the processor does no have good energy efficiency or low standby power. And obviously, the requirement of low power is much more critical for battery powered products. The Cortex-M processors are designed to be very energy efficient

and provide various low power features including architectural sleep modes, clock gating support, multi-power domains, etc. This enables the microcontrollers designed with Cortex-M processors to maintain low active power as well as ultra low idle power during sleep modes, whilst not compromising on performance and features. A processor with lower power consumption also has potential additional benefits by reducing the costs of the required power supply, and lower Electro Magnetic Interference (EMI). In many wireless and safety critical applications, low EMI is critical. Not only can it help the quality of the wireless communications, it also allows a reduction of the power required for data transmission, and reduced risk of causing interference problems to other devices which is especially important in medical and aviation applications.

High code density All the Cortex-M processors are based on Thumb-2 Technology. The Thumb instruction set provides a mixture of 16-bit and 32-bit instructions under one operation state, and in most cases, the 16-bit instructions are used for better code density. The high code density allows the designer to implement a system with a microcontroller device using a smaller flash memory, which can reduce the cost of the IoT product. It can also reduce the power consumed by the microcontroller device as the flash memory size required is reduced. Since most of the time 16-bit instructions are used, the processor often only needs to fetch instructions every other clock cycle. This reduces the power consumption of the system, as well as allowing more bus bandwidth for other data transfer operations. This can be important in devices with a fast Ethernet interface, as a DMA controller might be present in the system and need bus bandwidth to transfer data between the memory and the Ethernet interface.

High performance All the Cortex-M processors are 32-bit processors, and offer much higher performance than many 8-bit and 16-bit microcontrollers. Although it might be possible to implement an IoT product using legacy 16-bit or even 8-bit microcontrollers, using 32-bit processors has a number of advantages: With high processing power, it allows the IoT device to operate more reliably even if the amount of network activity generated by the end product might be unpredictable. If the network activity is higher than expected, an IoT device developed using a microcontroller with low performance might not be able to cope and crash. Using a 32-bit microcontroller provides sufficient performance headroom to handle unexpected network traffic. In some case, it might also provide a better chance for the IoT device to survive a DoS attack. A higher performance microcontroller can also provide the opportunity to include extra security measures such as encryption and authorization. These features provide better security to the end users, and should be deployed if the IoT device handles any sensitive or personal data. Obviously, better performance also enables better user experiences. For example, user can drive a graphic user interface easily and enable better feature set in user products.

Memory space and addressing modes: When the amount of internet activity increases in an IoT device, often it requires more memory to cope with the additional data for network packet processing. In addition, the increasing feature requirements of IoT products means, might need more memory space to accommodate the program image. As a result, having a larger addressable memory space can be beneficial for system reliability as well as the performance of an IoT product. However, in most 8-bit and 16-bit microcontrollers, the architecture only supports up to 64KB or 128KB of total memory space. This can become a major limitation for the potential features available in IoT products. In the Cortex-M microcontrollers, the architecture uses a 32- bit linear address and allows up to 4GB of total memory space. There is no need to use any memory paging technique, thereby allowing better performance and easier software development in Cortex-M microcontrollers when dealing with a larger memory space. The Cortex-M processors support various address modes of memory access. These make the architecture C-friendly and hence provide good performance, as well as making it easy to program and debug.

OS friendly: In many IoT applications, it is essential to use an embedded OS to handle different tasks. The architecture of the Cortex-M processors is designed with OS support in mind. For example, the dual stack pointer mechanism allows an OS to be implemented easily, and at the same time allows efficient memory usage and reliable operation. The processors also include a 24-bit timer which is dedicated for OS System Tick interrupt generation. These features make the porting of an OS across different CortexM

devices a simple task. For safety critical embedded applications, the embedded OS can also utilize the optional Memory Protection Unit (MPU) to provide memory access isolation. This can prevent any application from corrupting data in other application task, hence greatly improve the reliability of a system. Currently there are already more than 30 embedded OS available for the Cortex-M processors, and the number is still rising. **Security:** Besides better reliability, the optional MPU also provides better security to the embedded system. When used together with the privileged and unprivileged execution levels, many of the tasks can be run in unprivileged mode to prevent security issues. For example, the TCP/IP stack can be run at unprivileged level. In this way, even if the TCP/IP stack has a vulnerability and gets attacked by a hacker, it is likely that the attacker can only gain unprivileged access and cannot access other memory areas protected by the MPU. The high performance nature of the Cortex-M processors also makes it possible to implement sophisticated security measures in IoT. For example, an application can implement additional checking to validate the external inputs from the Internet, and handle encryption/decryption of the data transfers even without hardware AES accelerators. In addition, the higher performance also provides a better chance for an embedded device to survive a Denial-of-Service (DoS) attack. The Cortex-M processors also have a fault exception handling capability, which triggers fault exception handlers when certain types of errors are detected. For example, when a system is under attack and an error condition has occurred, such as a branch to an invalid address space, the fault exception will be triggered and the fault handler can then carry out remedial actions such as restarting the system or killing a crashed application task

INTEL GALILEO

Intel Galileo combines Intel technology with support for Arduino ready-made hardware expansion cards (called "shields") and the Arduino software development environment and libraries. The development board runs an open-source Linux operating system with the Arduino software libraries, enabling re-use of existing software, called "sketches". The sketch runs every time the board is powered. Intel Galileo can be programmed through OS X, Microsoft Windows and Linux host operating software. The board is also designed to be hardware and software compatible with the Arduino shield ecosystem.

Intel Galileo features the Intel Quark SoC X1000, the first product from the Intel Quark technology family of low-power, small-core products. Intel Quark represents Intel's attempt to compete within markets such as the Internet of Things and wearable computing. Designed in Ireland, the Quark SoC X1000 is a 32-bit, single core, single-thread, Pentium (P54C/i586) instruction set architecture (ISA)-compatible CPU, operating at speeds up to 400 MHz. The Quark is seen by some as Intel's answer to ARM, the processor design featured in smartphones and other single-board computers.

At a clock speed of 400 MHz, together with 256 Mb of DDR3 RAM and 8 Mb flash memory, the Galileo is much more powerful than competing Arduino boards. The Mega 2560, for example, has a clock speed of 16 MHz, 8 Kb RAM and 256 Kb flash memory.It would be more appropriate to compare the Galileo to another single-board computer, such as the Raspberry Pi. The latest iteration, the Pi 3 Model B, replaced the Pi 2 Model B in February 2016.It is more powerful than the older Galileo Gen 2, featuring a 1.2 GHz CPU and 1 Gb RAM.The Pi, however, does not have any flash memory.

Both Galileo boards support the Arduino shield ecosystem. Unlike most Arduino boards, the Intel boards support both 3.3V and 5V shields. The Intel development board comes with several computing industry standard I/O interfaces. The support for PCI Express means that Wifi, Bluetooth or GSM cards can be plugged in to the board. It also enables usage of solid state drives with the Galileo. The 10/100 Mbit Ethernet support enables the board to be connected to a LAN. It also enables accessing the Linux shell. The boards further support Micro SD, which means the available storage can be extended by up to 32 Gb. Other I/O interfaces include ACPI, USB 2.0 device and EHCI/OHCI USB host ports, high-speed UART, RS-232 serial port, programmable 8 MB NOR flash, and a JTAG port for easy debug.

Although the Galileo shipped with Linux, it was possible to have a custom version of Windows on both the Gen 1 and the Gen 2. This support was, however, suspended by Microsoft on 30 November 2015.

Microsoft cited hardware concerns, with some specifically attributing it to the low clock speed of the Galileo.

The Galileo supports the Arduino IDE running atop an unmodified Linux software stack, supported by a common open-source tool chain. The board comes pre-loaded with an SPI image of Linux. Although this version (Yocto 1.4 Poky Linux) has very limited features (e.g. it does not include a Wi-Fi module), it does not require any storage devices to be added. Intel also provides more functional versions of Linux for the boards. The "SD-Card" image can be downloaded and loaded onto the board via a Micro SD card. It includes, among a multitude of modules, a Wi-Fi module, support for OpenCV to enable computer vision, ALSA for sound processing and Node.js for JavaScript capabilities. A more advanced IoT DevKit version is also available to enable complex IoT projects, adding for example support for OpenCV-Python.

The Raspberry Pi, as well as most boards from Arduino, does not have an onboard real time clock. The Galileo boards have a real time clock, requiring only a 3V coin cell battery.The boards can therefore keep accurate time without being connected to either a power source or internet.

The Galileo can be seen as truly open source, as both the schematics and the source code are freely available for download without a software license agreement. However, some argued that the hardware shouldn't be designated open source if the processor core isn't also made open-source.

The Arduino ecosystem has three "levels"

1. "Arduino" is manufactured and distributed by Arduino.
2. "AtHeart" identifies any board which is manufactured using an Arduino-supported processor.
3. "Certified" means that the board is supported by the Arduino platform, but does not use an Arduino-supported processor.

The Galileo falls into the third category. Although it is the lowest level in the Arduino ecosystem, it still means that Galileo boards can be programmed using the official Arduino IDE, bought on the Arduino online shop and is compatible with Arduino peripherals such as shields.

Difference between Gen 1 and Gen 2

Intel Galileo Gen 2 Is similar to Gen 1 with the following changes:

- Replaces the RS-232 console port (audio jack) with a 1x6-pin 3.3V USB TTL UART header
- Adds 12-bit pulse-width modulation (PWM)
- Console UART1 redirection to Arduino* headers
- Power over Ethernet (PoE) capability (Requires installation of Silvertel Ag9712-2BR/FL power module)
- A power regulation system that accepts power supplies from 7V to 15V.
- Improved PWM control line means finer resolution for movement control.

Feature	GEN 1	GEN 2
SoC	Intel Quark X1000 32-bit 400 MHz	Intel Quark X1000 32-bit 400 MHz
Power (Barrel)	5V	7V-15V
Power (PoE)	No	Yes (Requires installation of Silvertel Ag9712-BR2/FL power module)

Fig 2.50: Intel Galileo Gen.1

Fig 2.51: Intel Galileo Gen.2

TEXT / REFERENCE BOOKS

1. Boswarthick,Omar Elloumi., The Internet of Things: Applications and Protocols, Wiley publications., 2012
2. Dieter Uckelmann, Mark Harrison, Florian Michahelles., Architecting the Internet of Things, Springer publications.2011
3. Marco Schwatrz Internet of Things with Arduino Cookbook, Packt Publications.2016 .
4. Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, “From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence”, 1st Edition, Academic Press, 2014.
5. Vijay Madisetti, Arshdeep Bahga, “Internet of Things: A Hands-On Approach” published by Vijay Madisetti 2014

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – 3 – Introduction to IoT– SCSA1308

Unit 3

COMMUNICATION AND CONNECTIVE TECHNOLOGIES

IoT Communication Model - Wireless medium access issues - MAC protocol survey -Survey routing protocols, Sensor deployment & Node discovery, Data aggregation & dissemination. Communication technologies, Long-range Wireless Protocols : LoRa WAN, Ingenu, Satellite Communications, Short-range Wireless Protocols : ANT+, WiFi, ZigBee, WHART, EnOcean, Z-Wave, NFC communication technologies : Zigbee – Wifi - Zwave.

3.1 IoT Communication Model:

The term of internet of things (IoT) communication offered by Internet protocols . Many of the devices often called as smart objects operated by humans as components in buildings or vehicles, or are spread out in the environment. IoT devices are found everywhere and will enable circulatory intelligence in the future. For operational perception, it is important and useful to understand how various IoT devices communicate with each other. Communication models used in IoT have great value. The IoTs allow people and things to be connected any time, any space, with anything and anyone, using any network and any service.

Communication types

1. Device-to-Device Communications
2. Device-to-Cloud Communications
3. Device to Gateway Model
4. Back End Data Sharing Model

Communications and Design Patterns

1. Request & Response Model
2. Publisher-Subscriber Model
3. Push-Pull Model
5. Exclusive Pair

3.1.1 Device-to-Device Communications:

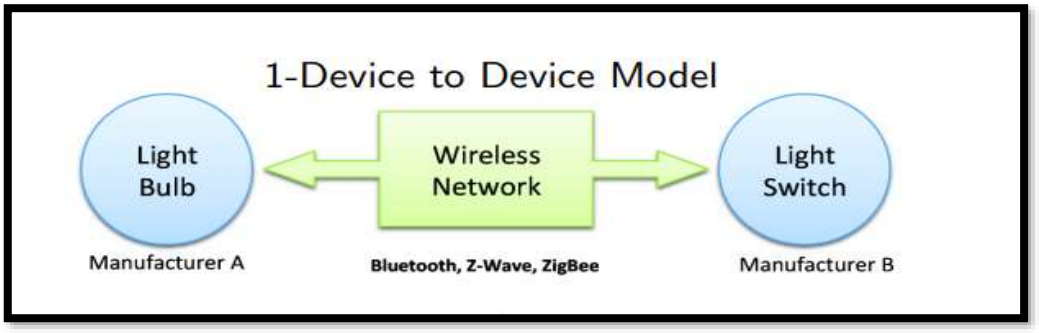


Figure 3.1: Device to Device Communication

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-Wave, or ZigBee to establish direct device-to-device communications.

Attack Surfaces on Device to Device Communication:

- Credentials stealing from the firmware
- Sensitive information disclosure
- No proper updating mechanism of firmware
- DoS Attacks

Buffer-overflow attacks

A buffer is a temporary area for data storage. When more data gets placed by a program or system process, the extra data overflows. It causes some of that data to leak out into other buffers, which can corrupt or overwrite whatever data they were holding. In a buffer- overflow attack, the extra data sometimes holds specific instructions for actions intended by

a hacker or malicious user; for example, the data could trigger a response that damages files, changes data or unveils private information.

Best Practices for securing Device to Device Communication:

• Evaluate hardware components, firmware, software, communications protocols

• Try to Make the signed Firmware, software and hash your binaries.

• Implement the machine to machine authentication securely.

• Get the feedback from the clients to improve the device security levels

3.1.2 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.

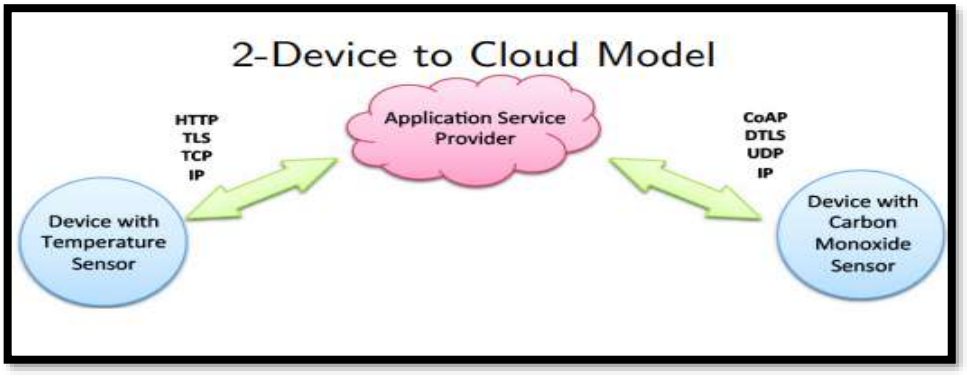


Figure 3.2: Device to Cloud Communication

Device to Cloud protocols . Below table 1 explains the details about the protocols :

Protocols	AMQP	MQTT	XMPP	CoAP
Transport	TCP/IP	TCP/IP	TCP/IP	UDP/IP
Message pattern	Publish — Subscribe	Publish — Subscribe	Point — Point Publish – Subscribe	Request – Response

The Advanced Message Queuing Protocol (AMQP) and the MQTT Protocol are often seen as mutually exclusive choices, especially in the Internet of Things (IoT). AMQP is a general-purpose message transfer protocol suitable for a broad range of messaging- middleware infrastructures, and also for peer-to-peer data transfer. It's a symmetric and bi- directional protocol that allows either party on an existing connection to initiate links and transfers, and has rich extensibility and annotation features at practically all layers. Both protocols share that they can be tunneled over Web Sockets and therefore function well in environments that restrict traffic to communication over TCP port 443 (HTTPS).

APPLICATION LAYER PROTOCOL:

‣ HTTP‣ CoAP ‣ Web Socket ‣ MQTT ‣ XMPP ‣ DDS ‣ AMQP

TRANSPORT LAYER PROTOCOL

‣TCP ‣ UDP

NETWORK LAYER PROTOCOLS:

IPv4

‣ Exhausted in 2011

‣ 32bit address

IPv6

‣ 128 bit addresses

6LoWPAN

- Limited processing capability

- Shows compression mechanism with IPv6 over 802.15.4

LINK LAYER PROTOCOLS:

- 802.3 - Ethernet

- 802.11 - wifi

- 802.16 – WiMax

- 802.15.4 - Low Rate WPAN

- 2G/3G/4G - Mobile Communication

3.1.3 MQTT Concepts

In MQTT, all messages are published into a shared topic space at the broker level. A –topic‖ in MQTT is a filter condition on the consolidated message stream that runs through the MQTT broker from all publishers. Publishing topics have a hierarchical structure (a path through topic space) and filters can be expressed as direct matching conditions (topic name and filter expression must match), or the filter can use wild-cards for single or multiple path segments.

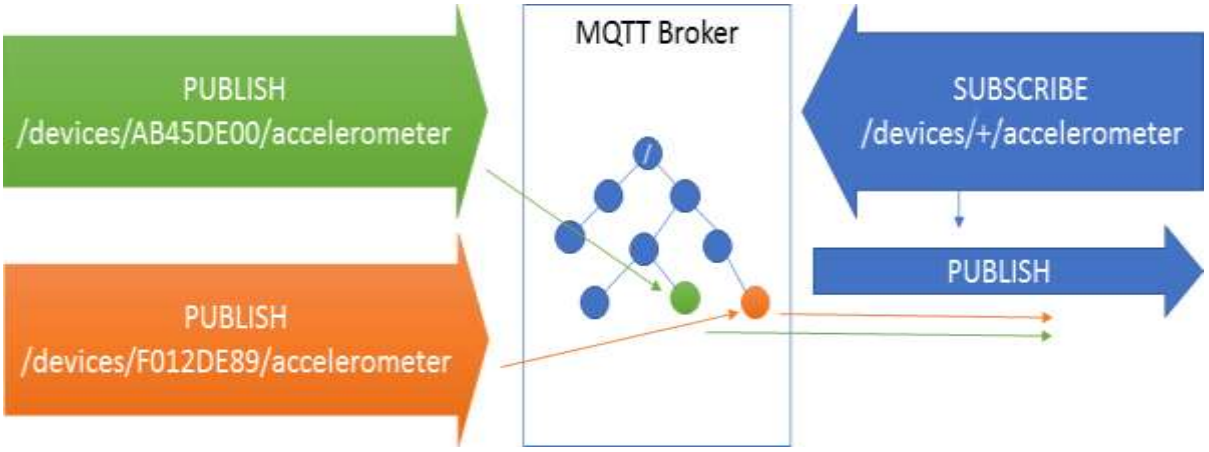


Figure 3.3: MQTT Protocol

Every published message from any publisher is eligible for delivery into any client session where a subscription exists with a matching topic filter. MQTT is very suitable for fast and –online‖ dispatch and distribution of messages to many subscribers in scenarios

where it's feasible for the entirety of the consolidated published message stream to be inspected on behalf of all

concurrent subscribers.

MQTT's –subscribe| gesture is much lighter weight. It establishes a filter context and simultaneously initiates and unbounded receive operation on that context. If session recovery is used, to scope of undelivered messages is that individual filter context. Subscribing is receiving. In some brokers, such an MQTT subscription context may indeed be backed by a volatile queue to allow leveling between fast and slow subscribers and to allow for caching of messages while a subscriber is temporarily disconnected and if session recovery is supported; but that's an implementation detail, not an explicit construct. The trouble with MQTT is that it uses TCP connections to a MQTT broker. Having an always-on connection will limits the time the devices can be put to sleep. This can be somewhat mitigated by using MQTT-S, which works with UDP instead of TCP. But MQTT also lacks encryption since the protocol was intended to be lightweight and encryption would add significant overhead.

Advanced Message Queuing Protocol (AMQP) is an open source published standard for asynchronous messaging by wire. AMQP enables encrypted and interoperable messaging between organizations and applications. The protocol is used in client/server messaging and in IoT device management. AMPQ is efficient, portable, multichannel and secure. The messaging protocol is fast and features guaranteed delivery with acknowledgement of received messages. AMPQ works well in multi-client environments and provides a means for delegating tasks and making servers handle immediate requests faster. Because AMPQ is a streamed binary messaging system with tightly mandated messaging behavior, the interoperability of clients from different vendors is assured.

AMQP allows for various guaranteed messaging modes specifying a message be sent:

At-most-once(sent one time with the possibility of being missed).

At-least-once (guaranteeing delivery with the possibility of duplicated messages).

Exactly-once (guaranteeing a one-time only delivery).

eXtensible Messaging and Presence Protocol (XMPP) is a TCP protocol based on XML. It enables the exchange of structured data between two or more connected entities, and out of the box it supports presence and contact list maintenance (since it started as a chat protocol). Because of the open nature of XML, XMPP can be easily extended to include publish- subscribe systems, making it a good choice for information that is handed to a central server and then distributed to numerous IoT devices at once. It is decentralized, and authentication can be built in by using a centralized XMPP server. The downsides of XMPP for IoT is that it lacks end-to-end encryption. It also doesn't have quality-of-service functionality, which can be a real deal-breaker depending on the application.

Constrained Application Protocol (CoAP) is a protocol specifically developed for resource- constrained devices. It uses UDP instead of TCP, and developers can work with CoAP the same way they work with REST-based APIs. Since it uses minimal resources, it is a good option or low-power sensors. Since it uses UDP, it also can run on top of packet-based technologies such as SMS, and messages can be marked confirmable or non-confirmable to work with QoS. Datagram Transport Layer Security (DTLS) can be used for encryption. The downside of CoAP is that it is a one-to-one protocol, so broadcast capabilities are not native to the protocol.

3.2 Attack Surfaces on Device to Cloud Communication

3.2.1 SQL injection , Cross-site scripting , Cross-site Request Forgery possible attacks on cloud application interfaces.

SQL Injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements (also commonly referred to as a malicious *payload*) that control a web application's database server (also commonly referred to as a Relational Database Management System – RDBMS). Since an SQL Injection vulnerability could possibly affect any website or web application that makes use of an SQL-based database, the vulnerability is one of the oldest, most prevalent and most dangerous of web application vulnerabilities.

3.3.2 Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts (also commonly referred to as a malicious payload) into a legitimate website or web application. XSS is amongst the most rampant of web application vulnerabilities and occurs when a web application makes use of invalidated or uuencoded user input within the output it generates. By leveraging XSS, an attacker does not target a victim directly. Instead, an attacker would exploit vulnerability within a website or web application that the victim would visit, essentially using the vulnerable website as a vehicle to deliver a malicious script to the victim's browser.

3.2.3 Username and password enumeration attacks

3.2.4 MITM attacks

Man-in-the-middle attack (MITM) is an attack where the attacker secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other. One example of man-in-the-middle attacks is active eavesdropping, in which the attacker makes independent connections with the victims and relays messages between them to make them believe they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker. The attacker must be able to intercept all relevant messages passing between the two victims and inject new ones.

3.3.5 Man in the Cloud (MiTC) attacks

Man in the cloud (MitC) attacks are interesting, and worrying, as they do not require any exploits or the running malicious code in order to get a grip during the initial infection stage. Instead they rely upon the type of common file synchronization service that we have all become used to, the likes of DropBox or Google Drive for example, to be the infrastructure providing command and control, data exfiltration and remote access options. Man in the cloud attacks are interesting, and worrying, as they do not require any exploits or the running malicious code .Simply by reconfiguring these cloud services, without end user knowledge and without the need for plaintext credential compromise to have occurred. It is hard for common security measures to detect as the synchronization protocol being used makes it all but impossible to distinguish between malicious and normal traffic.

Best Practices for securing Device to Cloud Security:

- Check all cloud interfaces are reviewed for security vulnerabilities (e.g. API interfaces and cloud-based web interfaces)
- Make sure cloud-based web interface not having weak passwords
- Ensure that any cloud-based web interface has an account lockout mechanism
- Implement two-factor authentication for cloud-based web interfaces
- Maintain transport encryption
- Ensure that any cloud-based web interface has been tested for XSS, SQLi and CSRF vulnerabilities.

3.3 Communications and Design Patterns

3.3.1 Request & Response Model

This model follows a client-server architecture.

The client, when required, requests the information from the server. This request is usually in the encoded format. This model is stateless since the data between the requests is not retained and each request is independently handled.

The server Categories the request, and fetches the data from the database and its resource representation. This data is converted to response and is transferred in an encoded format to the client. The client, in turn, receives the response.

On the other hand — In Request-Response communication model client sends a request to the server and the server responds to the request. When the server receives the request it decides how to respond, fetches the data retrieves resources, and prepares the response, and sends it to the client.

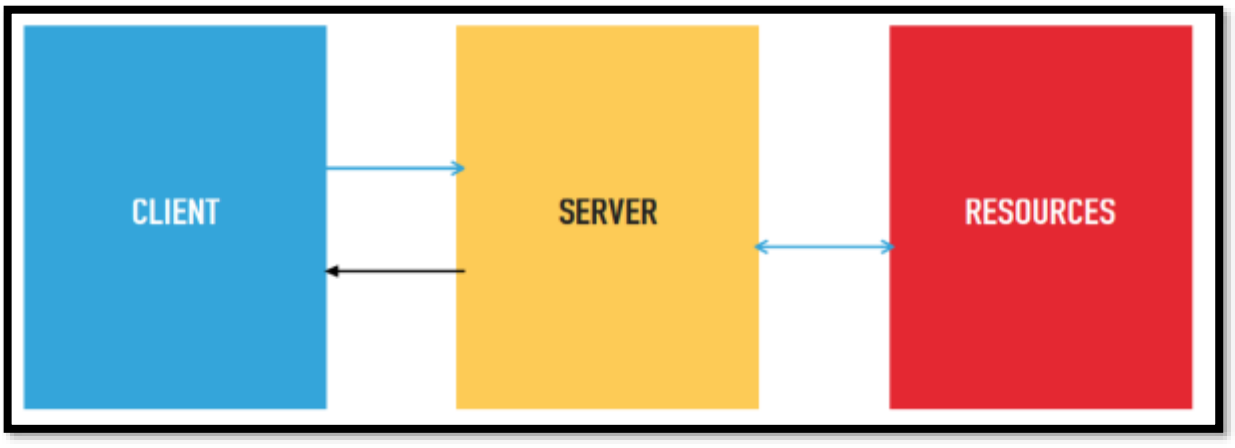


Figure 3.4: Client Server

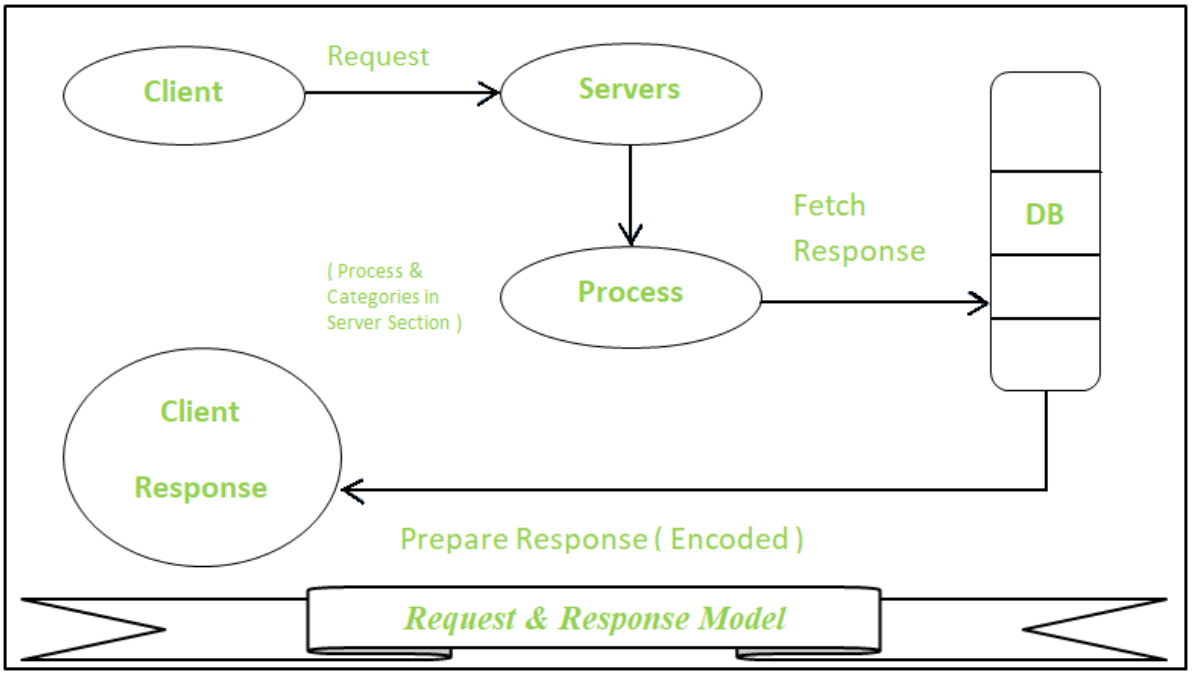


Figure 3.5: Request & Response model

3.3.2 Publisher-Subscriber Model

This model comprises three entities: Publishers, Brokers, and Consumers.

- Publishers are the source of data. It sends the data to the topic which are managed by the broker. They are not aware of consumers.
- Consumers subscribe to the topics which are managed by the broker.
- Hence, Brokers responsibility is to accept data from publishers and send it to the appropriate consumers. The broker only has the information regarding the consumer to which a particular topic belongs to which the publisher is unaware of.

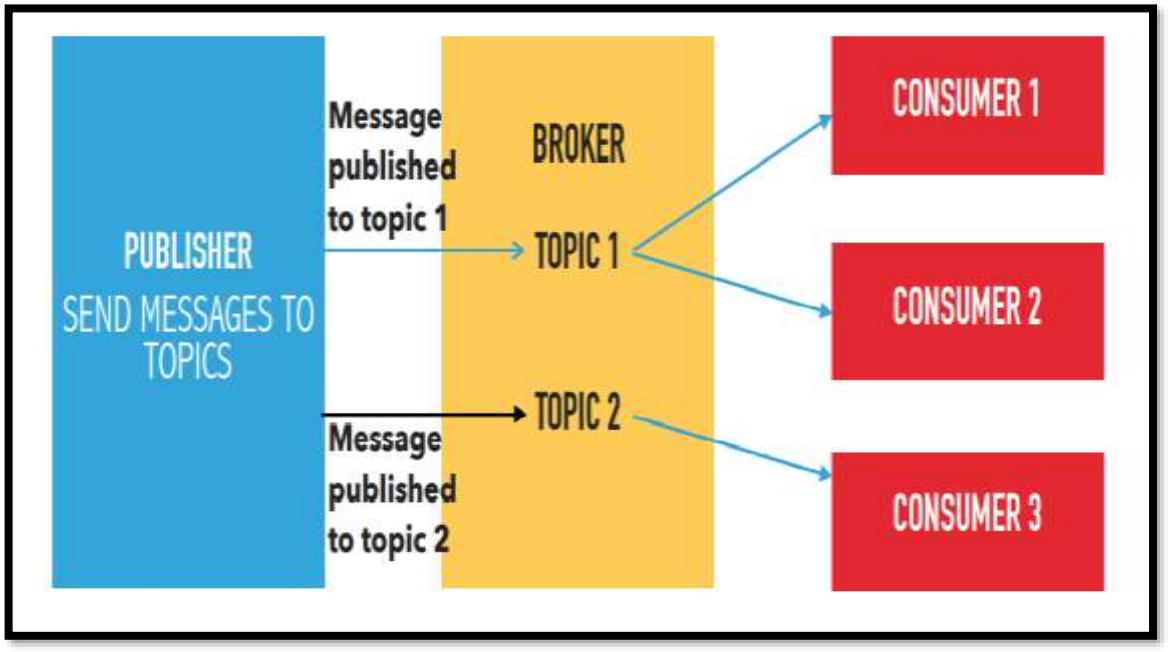


Figure 3.6: Publish-Subscribe architecture

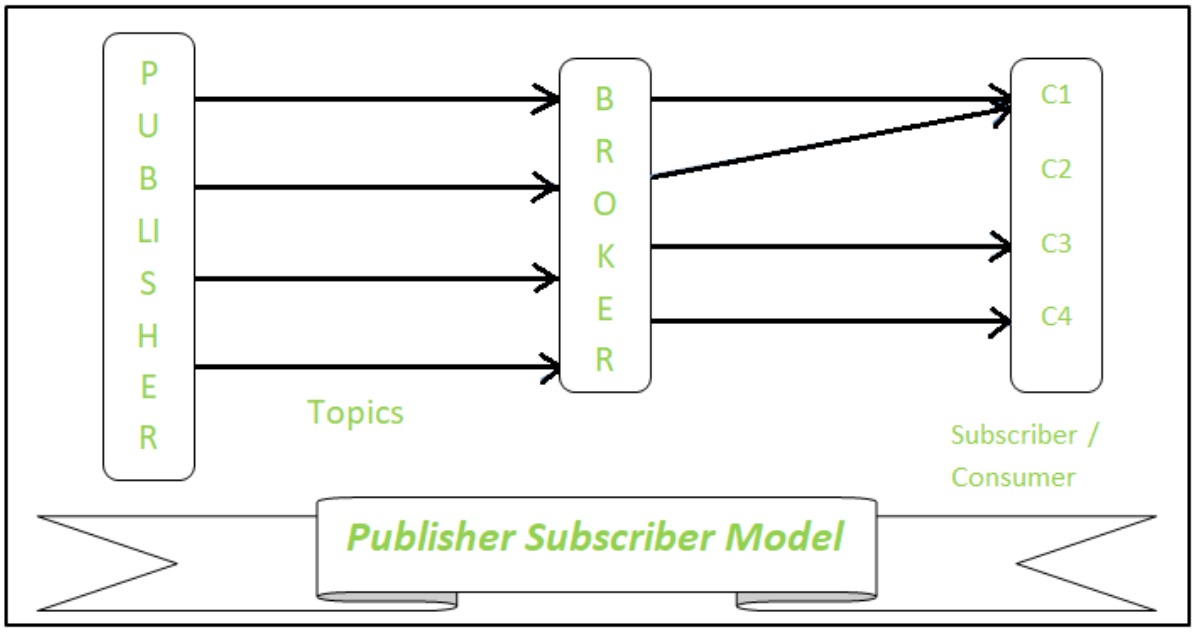


Figure 3.7: Publish-Subscribe model

3.3.3 Push-Pull Model

The push-pull model constitutes data publishers, data consumers, and data queues.

- Publishers and Consumers are not aware of each other.
- Publishers publish the message/data and push it into the queue. The consumers, present on the other side, pull the data out of the queue. Thus, the queue acts as the buffer for the message when the difference occurs in the rate of push or pull of data on the side of a publisher and consumer.
- Queues help in decoupling the messaging between the producer and consumer. Queues also act as a buffer which helps in situations where there is a mismatch between the rate at which the producers push the data and consumers pull the data.

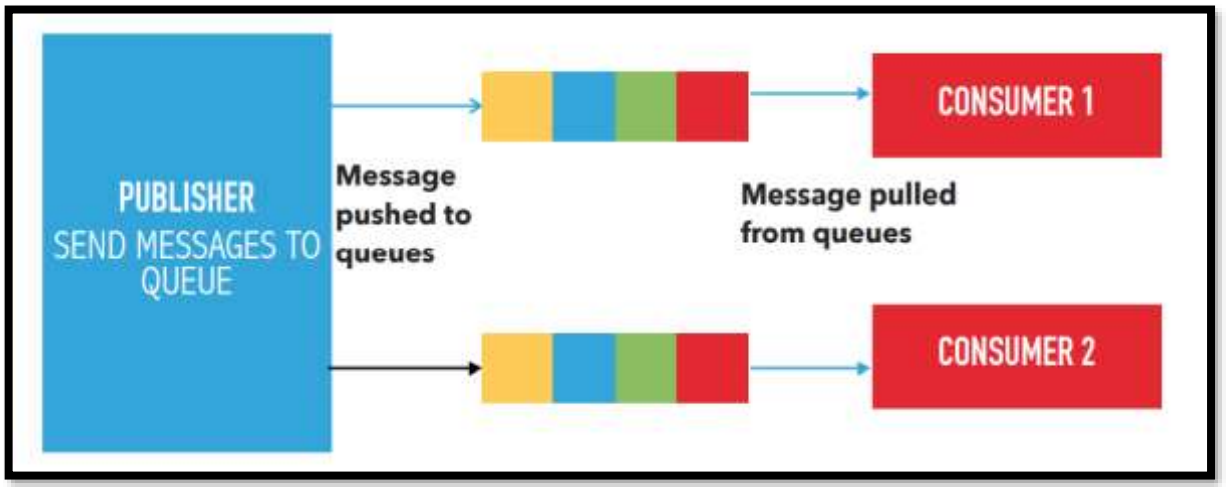


Figure 3.8: Push-Pull model

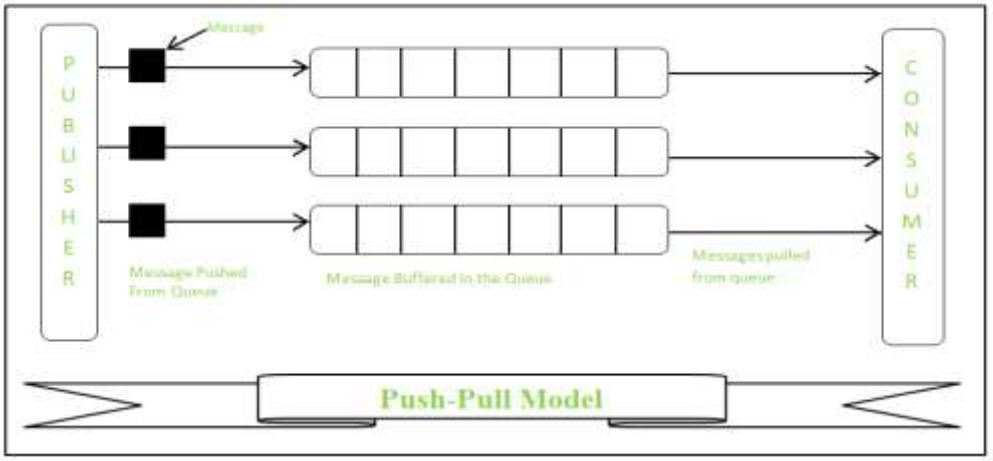


Figure 3.9: Push-Pull model

3.3.4 Exclusive Pair

- Exclusive Pair is the bi-directional model, including full-duplex communication among client and server. The connection is constant and remains open till the client sends a request to close the connection.
- The Server has the record of all the connections which has been opened.
- This is a state-full connection model and the server is aware of all open connections.
- WebSocket based communication API is fully based on this model.

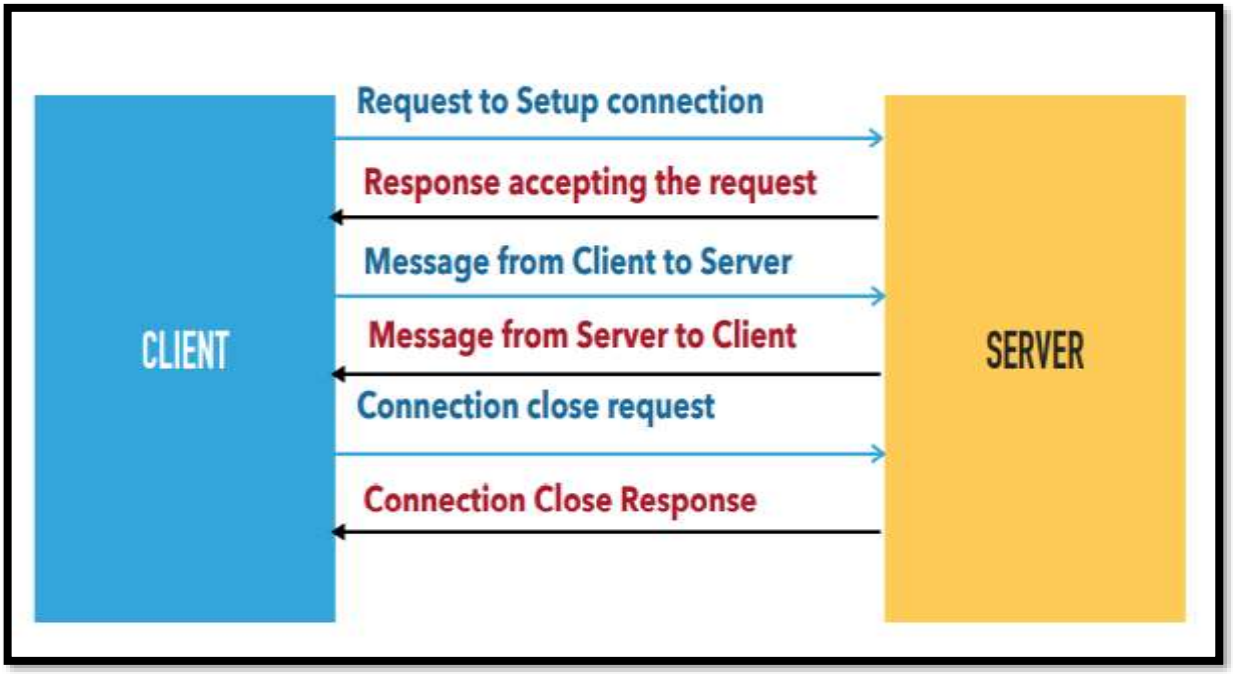


Figure 3.10: Exclusive pair model

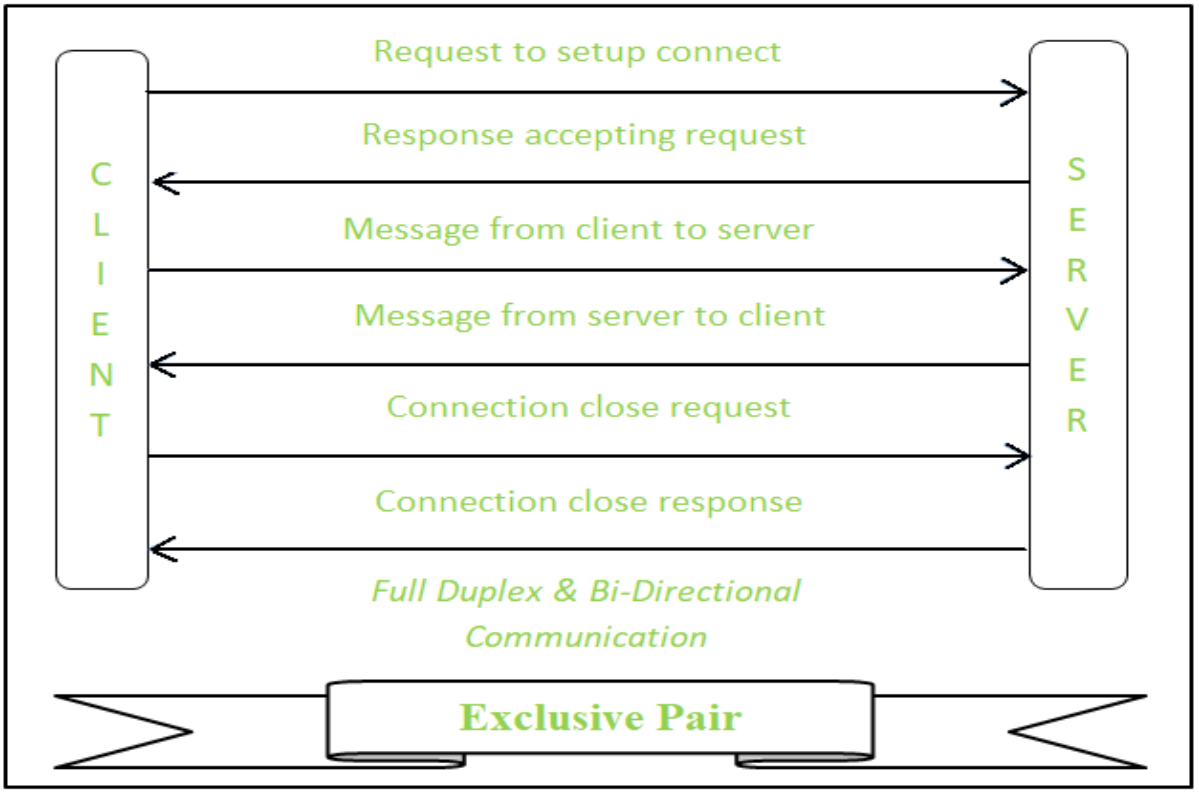


Figure 3.11: Exclusive pair model

3.4 Wireless Media Access Issues in Internet of Things

When it comes to communication using a wireless medium there is always a concern about the interference due to other present wireless communication technologies. Wireless means communication and message transfer without the use of physical medium i.e., wires.

Let us understand how communication is done between them. Different Mobile stations(MS) are attached to a transmitter/receiver which communicates via a shared channel by other nodes. In this type of communication, it makes it difficult for the MAC design rather than the wireline networks.

The very important issues which are observed are:

- Half Duplex operation
- Time-varying channel
- Burst channel errors.

These are explained as following below.

3.4.1 Half Duplex operation:

Half-duplex transmission means when the sender and receiver both are capable of sharing data but one at a time. In wireless transmission, it is difficult to receive data when the transmitter is sending the data because during transmission a large amount or a large fraction of signal energy is leaked while broadcasting. The magnitude of the transferred signal and received signal differs a lot. Due to which collision detection is even not possible by the sender as the intensity of the transferred signal is large than the received one. Hence this causes the problem of collision and the prime focus should be to minimize the collision

3.4.2 Time-varying channel :

Time-varying channels include the three mechanisms for radio signal propagations they are Reflection, Diffraction, and Scattering.

- Reflection –

This occurs when a propagating wave carrying information intrudes on an object that has very large dimensions than the wavelength of the wave.

- Diffraction –

This occurs when the radio path between the transmitter and the receiver is collided by the surface with sharp edges. This is a phenomenon which causes the diffraction of the wave from the targeted position.

- Scattering –

This occurs when the medium through from the wave is traveling consists of some objects which have dimensions smaller than the wavelength of the wave.

While transmitting the signal by the node these are time shifted and this is called multipath propagation. While when this node signals intensity is dropped below a threshold value, then this is termed as fade. As a result Handshaking strategy is widely used so as a healthy communication can be set up.

3.4.3 Burst channel errors :

Burst channel errors are called as a contiguous sequence of symbols, which are received in a communication channel, in which the first and last symbols has an error and there is no evidence of contiguous sub-sequence of corrected received symbols. When time-varying channels are used then signals strengths are introduced due to which errors are observed in transmission. For these channels in wireline networks, the Bit rate is high as 10 -3.

3.5. MEDIUM ACCESS CONTROL:

In the seven-layer OSI model of computer networking, media access control (MAC) data communication protocol is a sub layer of the data link layer (layer 2). The MAC sub layer provides addressing and channel access control mechanisms that make it possible for several terminals or network nodes to communicate within a multiple access network that incorporates a shared medium, e.g. an Ethernet network. The hardware that implements the MAC is referred to as a media access controller.

The MAC sub layer acts as an interface between the logical link control (LLC) sub layer and the network's physical layer. The MAC layer emulates a full-duplex logical communication channel in a multi-point network. This channel may provide unicast, multicast or broadcast communication service.

3.6 OSI Reference Model

What is OSI Model ???

"OSI Model (Open Systems Interconnection) is a standardised Reference Framework for conceptualising data communications between networks"

Official Standard: ISO7498 (1984)

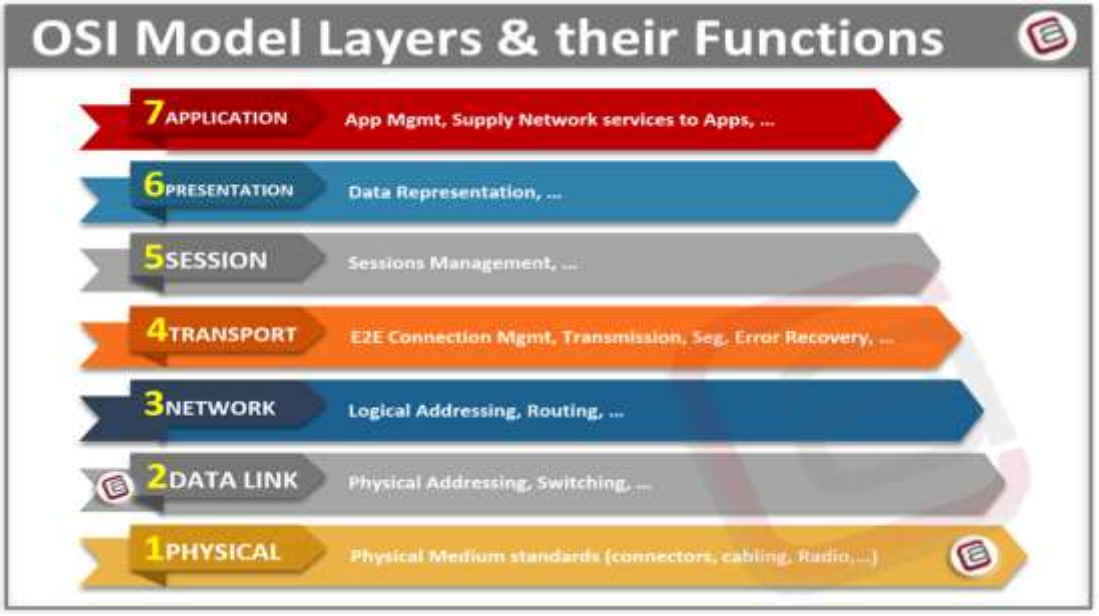


Figure 3.12: ISO-OSI Reference model

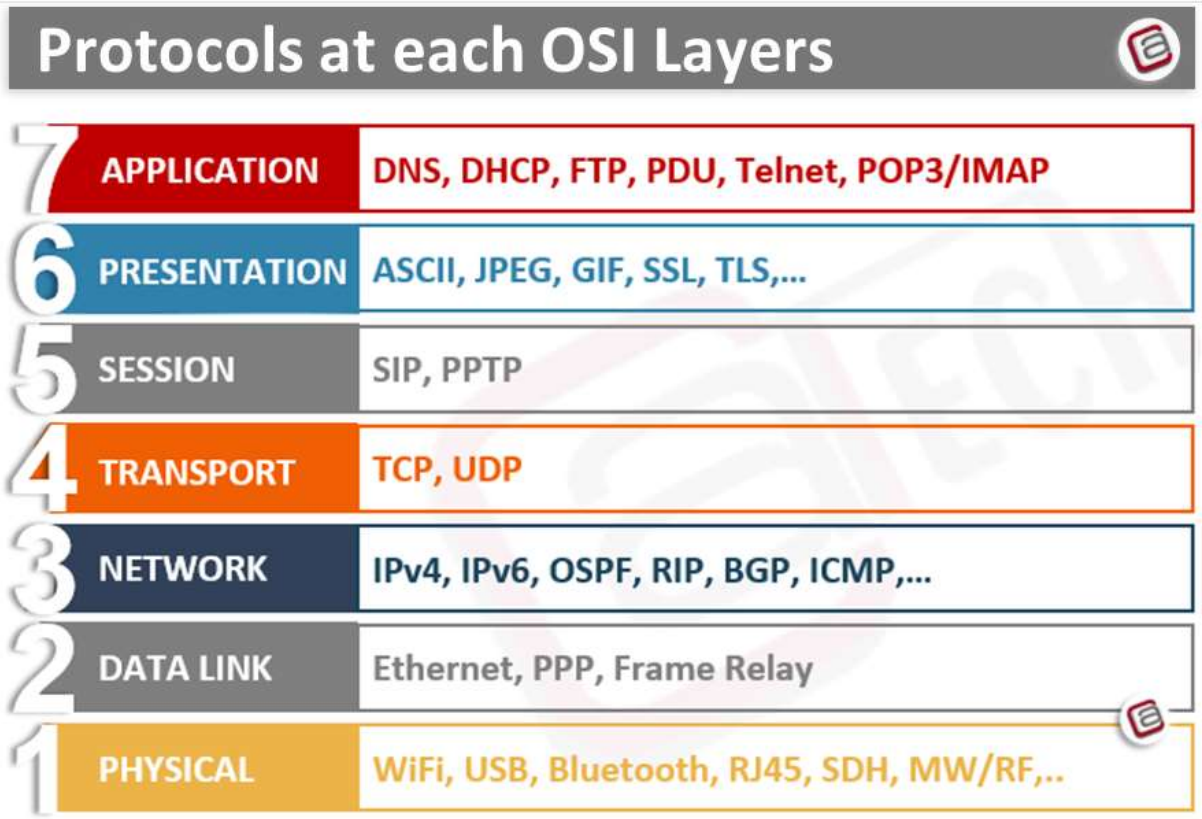


Figure 3.13: ISO-OSI protocols

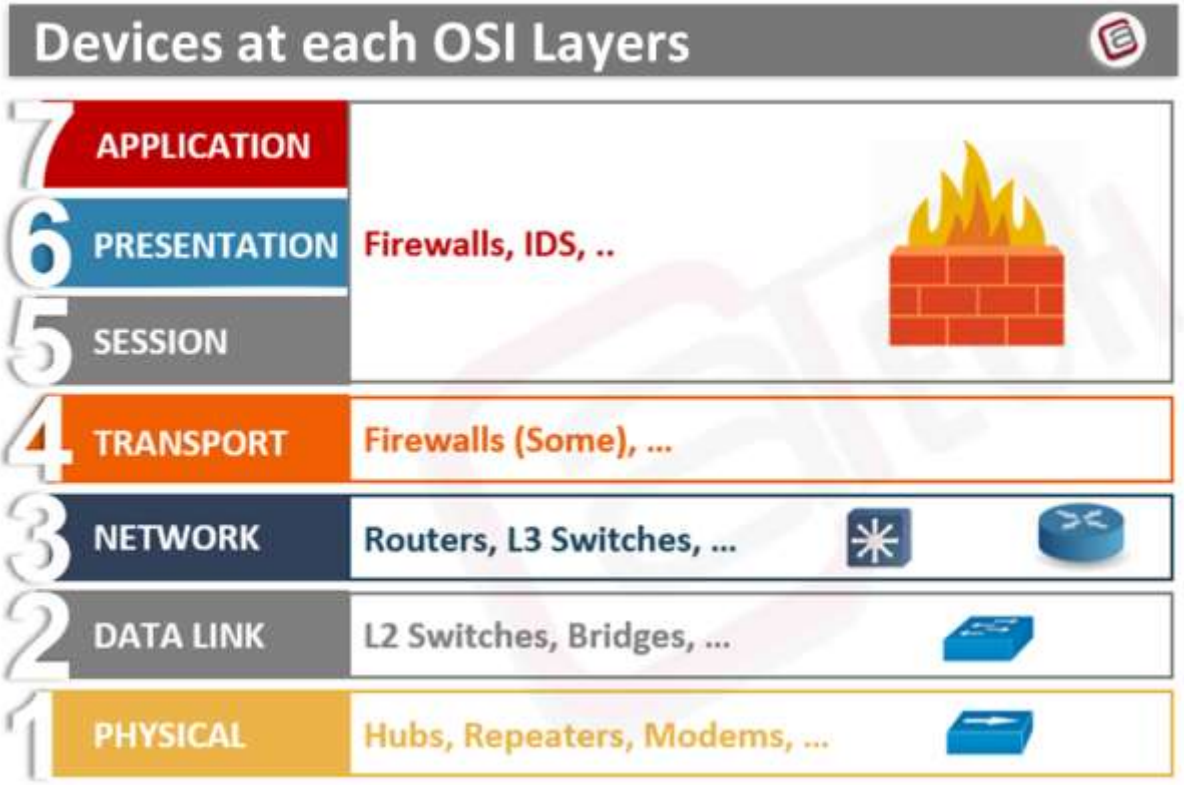


Figure 3.14. ISO-OSI Devices

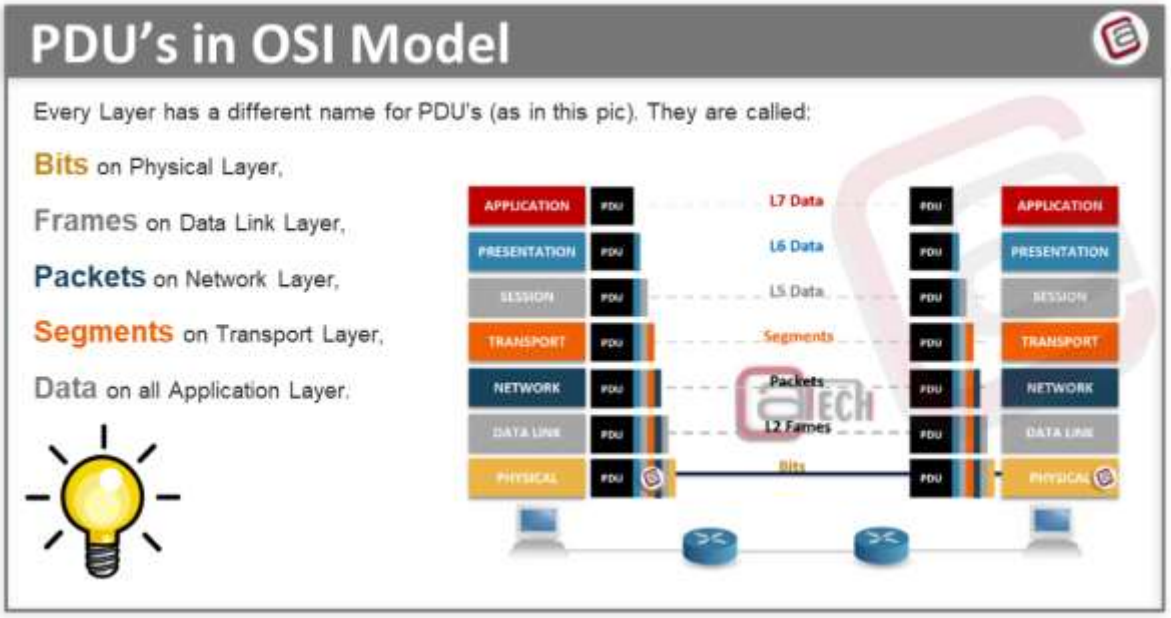


Figure 3.15. ISO-OSI Protocol Data Unit

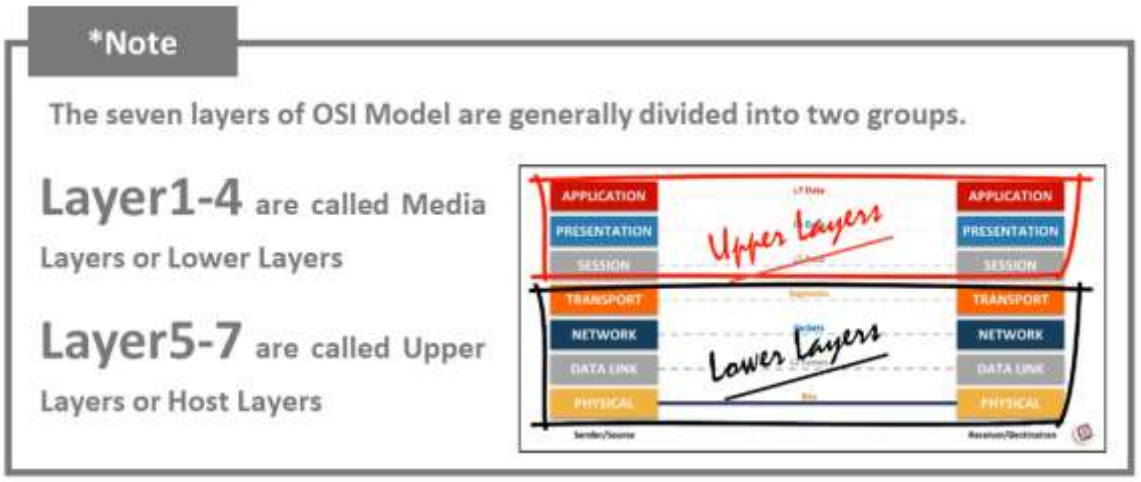


Figure 3.16. ISO-OSI upper-lower layers

Transport Layer Ports

Category	Range	Comments
Well Known Ports	0 - 1023	Used by system processes e.g. FTP(21)
Registered Ports	1024 - 49151	For specific services e.g. Port 8080
Private Ports	49152 – 65535	For Private purposes

Important Ports on Transport Layer

Port Number	Protocol	Application
20	TCP	FTP data
21	TCP	FTP control
22	TCP	SSH
23	TCP	Telnet
25	TCP	SMTP
53	UDP, TCP	DNS
67, 68	UDP	DHCP
69	UDP	TFTP
80	TCP	HTTP (WWW)
110	TCP	POP3
161	UDP	SNMP
443	TCP	SSL
16,384–32,767	UDP	RTP-based Voice (VoIP) and Video

Figure 3.17. Transport layer ports

3.7 MAC protocol survey:

MAC protocol is the first protocol layer above the Physical Layer in ad hoc .The primary task of any MAC protocol is to control the access of the nodes to shared medium.

Classification of MAC protocols:

These are as following below.

1. Contention-based protocols without reservation/scheduling –
- Bandwidths are not reserved.
- No guarantees.

2. Contention-based protocols with reservation mechanisms –
- Bandwidth is reserved for transmission.
- Guarantees can be given.

3. Contention-based protocols with scheduling mechanisms –
- Distributed scheduling is done between nodes.
- Guarantees can be given.

4. Other protocols –
- Combine multiple features of other protocols.

- It can also use a completely new approach.

Most Internet of Things (IoT) technology features are defined by the protocols used to design the technology for specific applications. Features such as network topology, power consumption, transmission power efficiency, and delays are important issues in the definition or choice for using a certain technology for a particular solution. Medium access techniques, data rates, communication mode between devices, transmission range, power consumption, and others are all examples of characteristics derived from the development and deployment of each protocol. Therefore, the study of MAC layer protocols can show how to design a suitable technological solution for an application.

Based on its own needs, IoT applications may require the adaptation of the existing network protocols so that they can meet the requirements of IoT applications. Protocols may need to be adjusted, evolved or developed to meet the IoT applications that demand different performance characteristics such as far-reaching, reliable and robust low power transmission techniques. According to requirements, it is possible to classify and point out the main MAC layer protocols suitable to attend a service characteristic.

3.8 Short Range MAC Layer Protocols

Short range coverage medium access control (MAC) protocols are defined by the Institute of Electrical and Electronics Engineers (IEEE) as Wireless Personal Area Networks (WPAN), which is the network established between elements that surround the human body. WPAN communication technologies differ from other conventional wireless network technologies. These networks call for easy connectivity in order to reach personal wearable or hand-held devices. Moreover, WPAN requires power efficiency, small size, low cost and maybe most importantly easy to use devices.

Short-distance technologies such as near field communication (NFC) and radio frequency
Identification (RFID) are technologies that fit into this study context due to their usage with
Differentiated mechanisms for the physical and linking layers.

- **Radio Frequency Identification (RFID):**

It refers to a set of technologies that are aimed at identifying and recognizing elements (tags). An RFID system is basically composed of two types of devices: the identified devices (tags) and the device identifiers or readers. Tagged devices are triggered by RF (Radio Frequency) waves emitted by the reader devices and reply its identification (ID) tags. Readers handle data exchange between them. When necessary, readers send RF pulses interrogating the tags in the area. Tags reply to this question by submitting their tag IDs.

- **Near Field Communication (NFC)**

Near Field Communication is a short range transmission technology that uses low-power
transmission links that, differently from Bluetooth, do not require pairing for transmission. Just bringing one device close enough to the other allows communication. This feature forces the user of the device to be handling it during use. As the facility of the device works only with is owner, it is a manner to ensure the safe security of the technology usage. Its operation is comparable to RFID technology because NFC devices can act as both a reader

and a tag. The communication is performed in active or passive mode, operating in the 13.56 MHz band.

- **Bluetooth IEEE 802.15.1**

WPAN IEEE 802.15.1 also called the Bluetooth Basic Rate (BR) is a global 2.4 GHz specification working with short-range wireless networking. It covers the versions v1.0, 1.0B with voice dialing, call mute, last number redial and a 10 m range as the main facilities and a v1.2 with adaptive frequency hopping added. Versions v2.0+Enhanced Data Rates (EDR) and v2.1+EDR added more capabilities such as improved resistance to radio frequency interference as well as improving indoor coverage and LOS range to 100 m. Fast transmission speeds and low power consumption mechanisms were also increased on the v2.0+EDR and v2.1+EDR versions. Version v2.1 counts on a sniff subtracting function that results in less transmissions do access the medium and so reducing interference.

- **Bluetooth Low Energy**

Being part of the Bluetooth v4.0 standard adopted in 2010-06-30, Bluetooth Low Energy (BLE) is also known as Smart Bluetooth. BLE is an IEEE 802.15.1 variation with better and more suitable capacities for low power applications than the classic Bluetooth Basic Rate. Devices that demand communication with both standards of Bluetooth are required to implement and support both protocol stacks due the incompatibilities among them. Star is the only topology accepted by BLE due the standard definition that does not permit physical link connections among slave devices. Any data exchanged between two slave devices shall pass through the unique master and a slave device may not be connected to two master units at the same time.

- **IEEE 802.15.4**

It is a subgroup of features that refers to physical and medium access control layers that can support ZigBee and 6LoWPAN upper. IEEE 802.15.4 focuses on physical and data link layer specifications while ZigBee Alliance aims to provide the upper characteristics.It is a standard that defines PHY and MAC layers for personal area networks that demand low rate and low cost applications.

3.9 Long Range MAC Layer Protocols :

Based on their own requirements such as rate, distance coverage, robustness, etc., the existing

network protocols need some adaptation to meet the necessary requirements to attend IoT services. In some cases, some protocols were developed to meet IoT applications that demand far-reaching, reliable and robust transmission. Some of the protocols classified as protocols for LP-WANs are able to satisfy the demand for protocols with a large coverage area. LP-WANS protocols can overcome some mobile cellular network failures increasing strong adaptations to meet the IoT requirements.

LP-WAN are presented as good candidates to support several of the previously mentioned

requirements of the IoT structure, and are able to surpass the short-range restriction of the

LANs Among the possible solutions are the proprietary and unlicensed ISM band technologies

Sigfox, LoRa/LoRaWAN, against mobile cellular network solutions such as LTE-A (Long Term

Evolution—Advanced) and Narrow Band IoT (NB-IoT).

- **NB-IoT**

According to the LTE eMTC regional specifications, it can operate only within the bandwidth of an LTE carrier. NB-IoT systems can be implemented as autonomous systems in the Global System for Mobile Communications (GSM) band , employed in the LTE bandwidth carrier or in the LTE bandwidth guard band. Due to the reduction of the NB-IoT bandwidth to 180 kHz, low data rate devices can have extended coverage, complexity reduction, and low power consumption. For scenarios with coverage problems of cellular network operators, NB-IoT is seen as the future of IoT devices using mobile network infrastructure.

- **LTE—Long Term Evolution**

Long Term Evolution enhanced Machine-Type Communication (LTE eMTC) standards-based technologies support CAT-0 and CAT-M modes. While LPWAN LTE CAT-0 is commonly used to implement M2M/IoT, CAT-M reduces complexity keeping the coverage aspect using existing mobile cellular network infrastructure . LTE eMTC counts on the same mobile technology benefits as security, privacy, data reliability and device identification.

- LoRa—Long Range Protocol

LoRa defines a physical layer technology developed by Cycleo in 2010, a company that was acquired by Semtech from Camarillo, United States of America.The LoRa module manufacturer offers the user a programmed library that allows communication between LoRa nodes, providing a simple link protocol. Libelium, a company based in Zaragoza, Spain provides the tools and libraries needed to operate with LoRa The LoRa protocol is an open standard defining physical layer to use direct sequence spread spectrum (DSSS) with multiple spreading factors that range from 7 to 12. This combination allows the establishment of a relationship between distance coverage and the desired data rate.

- **Long Range Protocols**

In the case of the LPWANs presented, the main parameters that differ are the distance range of the protocols discussed. Improvements, adaptations, and innovations have occurred mostly at the PHY layer while the MAC layer was adjusted as needed.

With no power control and baud rate adjustment mechanisms, the LoRa modulation itself becomes inefficient for applications that demand frequent transmissions and high transmission rates (tens of kilobits or more). Due to these characteristics, LoRa is automatically surpassed by its evolution, the LoRaWAN. LoRaWAN improves the MAC layer with the power control mechanism by dividing the operational mode of the devices into classes and using adaptive data rates (ADR).

3.10 Survey routing protocols:

A routing protocol specifies how routers communicate with each other to distribute information that enables them to select routes between nodes on a computer network. There exist lots of routing protocols each having a unique operating standard with significant performance for Wireless Sensor Networks that can be deployed for IoT with few modifications for bandwidth and power consumption. We discuss few of the broad categories of routing protocols in this section.

- **Naive Routing**

The idea deployed in naïve routing is flooding. Each node can overhear its neighbors within its range. The source node floods the network with route request packets called as beacons. Destination nodes respond with a route reply message to the beacon and communication link is established between these nodes. Beaconing is typically utilized for location tracking, discovering routes to destinations and tracking neighbors through keep-alive requests. One of the most important factors that affect performance is the beacon interval in the route discovery process. If the beacon interval is too small, the number of beacons generated becomes huge. On the other hand, a higher beacon interval incurs a lesser number of generated beacons. Popular routing protocols such as DSR, DSDV and AODV fall under this category.

- **Hierarchical Routing:**

Nodes form clusters based on polling. The cluster head is responsible for all communications on behalf of the members of the cluster. Group mobility can be achieved by the cluster head following some metric to devise the mobility pattern of the nodes in the cluster. LEACH is a common example, where the cluster head is rotated among the members to facilitate load balancing that can be deployed for IoT environments.

- **Query based Routing**

  The underlying principle of query based routing is data dissemination within the network. A querying node can retrieve data from any node in the network. Common examples are SPIN and Direct Diffusion.

- **Multipath routing**

Protocols employing multipath routing seek to and use alternate paths towards every destination. This distributes the cost of forwarding packets among more nodes, saving the energy of individual, highly-frequented nodes.

- **Probabilistic routing**

Routing decision is based on the calculated probabilistic value. A primitive method to compute these values is by gossiping. Data packets are flooded into the network like a rumor with a probability p. Unlike other flooding mechanisms, these packets are forwarded only once and thereby the traffic overhead is reduced. A highly structured approach is to refer the prior history of packet delivery and mobility pattern, based on this we can decide which nodes can form a route to the destination.

- **Ad-hoc On-Demand Distance Vector (AODV)**

AODV computes a loop free single path on demand. A mobile node discovers and maintains a route to another node only when it needs to communicate. One observation of AODV is that, though the source actually discovers multiple paths during the route discovery process, it chooses only the best route and discards the rest. Also, frequent route breaks cause the intermediate nodes to drop packets because no alternate path to the destination is available. This reduces the overall throughput and the packet delivery ratio. Moreover, in high mobility scenarios, the average end-toend delay can be significantly high due to frequent route discoveries. When route failures occur, the process of route discovery has to begin from scratch consuming more network resources and overhead.

3.11 Comparison of IoT Protocols - Low Power Wide Area Network (LPWAN)

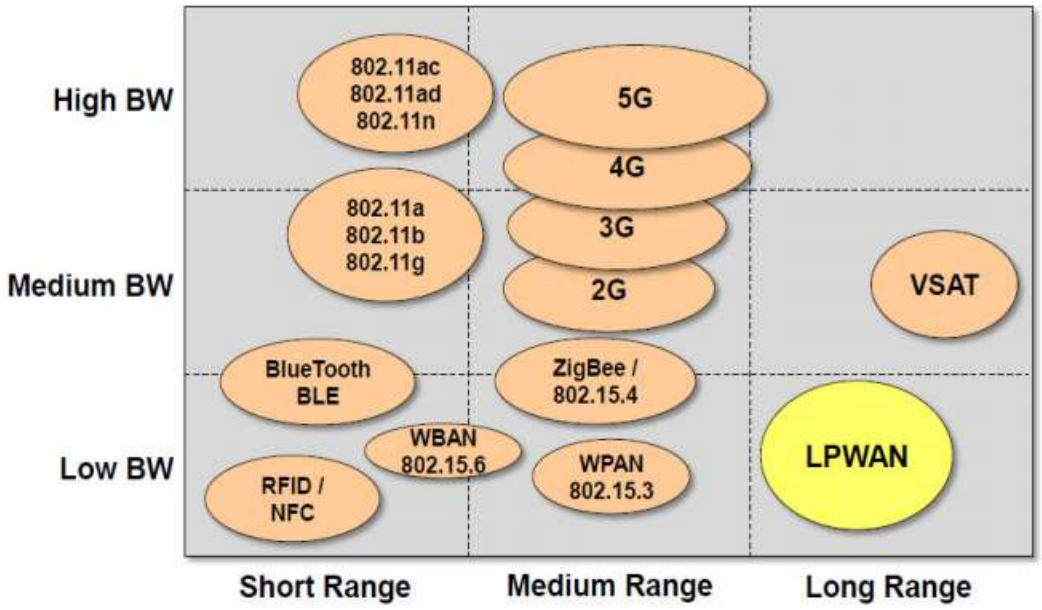


Figure 3.18.Comparison

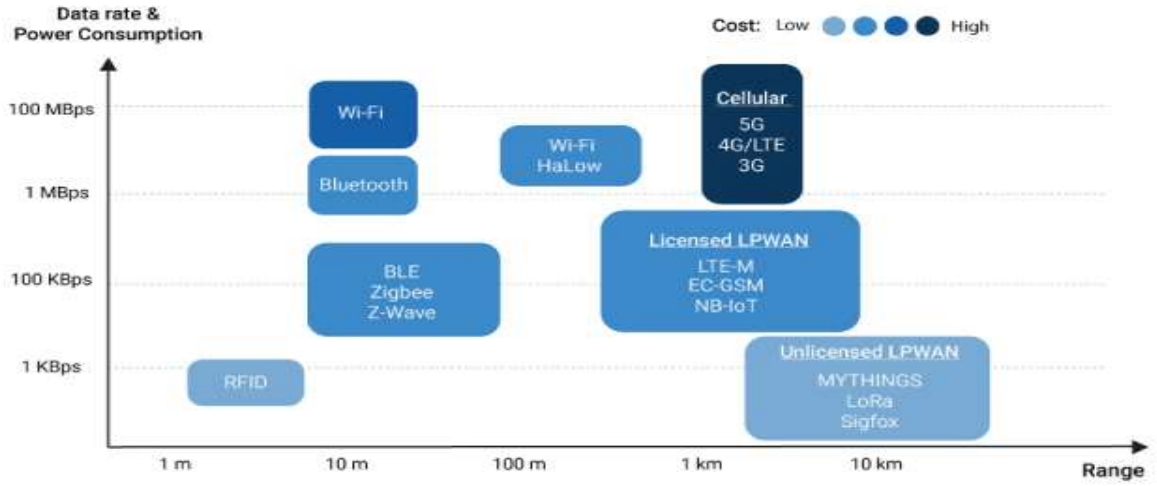


Figure 3.19. coverage distance

Key IoT Verticals	LPWAN (Star)	Cellular (Star)	Zigbee (Mostly Mesh)	BLE (Star & Mesh)	Wi-Fi (Star & Mesh)	RFID (Point-to-point)
Industrial IoT	●	○	○			
Smart Meter	●					
Smart City	●					
Smart Building	●		○	○		
Smart Home			●	●	●	
Wearables	○			●		
Connected Car					○	
Connected Health		●		●		
Smart Retail		○		●	○	●
Logistics & Asset Tracking	○	●				●
Smart Agriculture	●					

● Highly applicable ○ Moderately applicable

Figure 3.20. comparison

3.12 Sensor deployment & Node discovery

Five static sensor deployment strategies are introduced. The deployment strategies are one uniform random deployment and four regular deployments: square grid, triangle grid, hexagon grid, and THT. These strategies are shown in Fig. 1 . A square field is considered in this work.

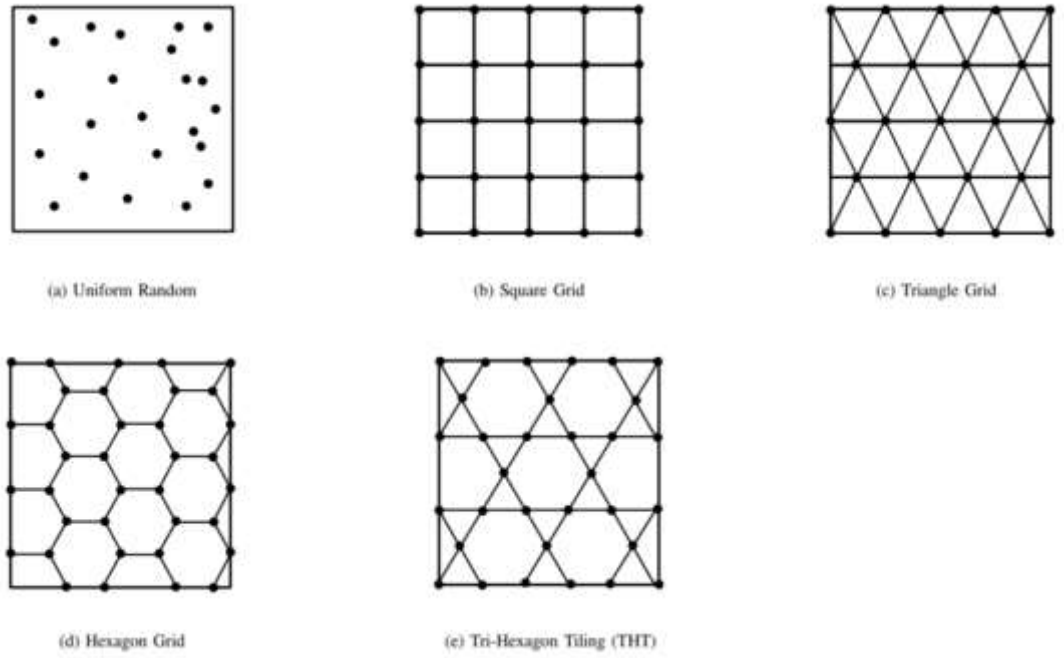


Figure 3.21. Sensor Deployment Strategies

- Uniform Random Deployment
- Regular Deployment

1. Square Grid
2. Triangle Grid

3. Hexagon Grid
4. Tri-Hexagon Tiling (THT)

Uniform Random Deployment:

In the random deployment scheme, sensors are deployed in a random way. Therefore, the exact locations of nodes are not known. Random deployment is usually suitable in thefollowing cases:

- When the environment is harsh and therefore, deployment of nodes is exceedingly difficult.
- When wireless sensor network is large scale.
- When the cost of sensors is not an issue

In this method, sensors are dropped form a helicopter, an airplane or an unmanned vehicle. Uniform random deployment is random deployment where nodes are placed in a sensing area randomly with uniform distribution. Fig. 1a illustrates the uniform random deployment method.

Regular Deployment:

Regular deployment refers to placing sensors in a regular form in a sensing area. This type of deployment can usually be applied in non-harsh environment and non-large-scale regions. In this paper, four regular deployment strategies are introduced.

Square Grid: In the square grid deployment, a square sensing area is divided into small squares and the nodes are placed at the points of grid intersection as shown in Fig. 1b.

Triangle Grid: A square sensing area in this model is divided into small triangles and the sensors are located in the points of triangle heads as shown in Fig. 1c

Hexagon Grid: In hexagon grid deployment scheme, a sensing area is divided into hexagons, as illustrated in Fig. 1d, and the sensors are placed at the vertices of hexagons.

Tri-Hexagon Tiling (THT): THT is a deterministic deployment method that was proposed in [12]. TThis model is a pattern of hexagonal stars that consist of triangles and hexagons. Tiling refers to covering the area without leaving any gaps and without any overlapping. Fig. 1e displays the THT deployment strategy.

IoT Sensor Deployment Challenges:

The business benefits of IoT coupled with market trends are driving rapid IoT adoption in every industry vertical like smart cities, building automation, industrial, healthcare, etc. This growing demand for IoT connectivity is paving the way to a plethora of sensor types for various use cases such as traffic sensors, parking meters, pressure sensors, electricity sensors, and so on. Efficient sensor deployment is one of the key success factors in every IoT investment and that’s where most enterprises struggle a lot today.

Challenge #1: Variety of sensors and chipsets

There is an increasing number of commercial launches of cellular technologies like NB-IoT, Cat-M1/M2, LTE-M, LoRa, etc. Each of these technologies has specific electronics for sensing endpoints. Although the cost of mobile chipsets has been declining over time, currently there’s no cost-effective solution that can work with the widespread in electronics of the cellular-connected IoT sensors to measure connectivity parameters.

Challenge #2:Identify an optimal location to deploy sensors

Whether it is a factory floor or a smart building, it is never easy to identify the perfect spot to deploy the IoT sensors. To successfully capture and transmit the ambient inputs over-the-air, the sensor must be located near the input-source and also where the network signal strength is reliable. To determine signal quality spread, today operators mostly rely on statistical modeling using terrain and clutter models. This results in statistical variabilities. Currently, there is no way to capture empirical network data, the lack of which often leads to installing sensors in sub optimal locations where the signal quality is poor and unreliable. Unreliable connectivity results in poor sensor performance which in turn affects the performance of the overall IoT solution and impacts the customer experience.

Challenge # 3: Not easy to remediate sensor performance issues

IoT sensors are typically installed in hard-to-access locations. When a sensor exhibits sub-optimal performance due to network connectivity, it is hard to root-cause the problem. Currently, there is no way to obtain real-time visibility into connectivity data to assess network health. The technicians may need to try a different location hoping for better wireless connectivity or replace the sensor itself. In such trial-and-error methodology, multiple truck rolls could be needed before the problem is identified and rectified. This impacts both OpEx and TCO and also skews up inventory management.

Challenge # 4 Network validations for connectivity SLAs

It is difficult to guarantee a satisfactory level of service if the IoT devices fail to deliver due to poor connectivity. Currently, RF and RAN design are done based on statistical models. Once the IoT network is deployed, due to the lack of network health data, it is not possible to validate your network design assumptions and performance in the context of the initial SLAs.

In a highly competitive digital marketplace, businesses can’t live with these challenges for too long. The next blog discusses ways to overcome these challenges.

NEIGHBOUR DISCOVERY FOR lot SCENARIOS :

1. Time Synchronised Protocols
2. Deterministic Approach
3. Colocation based Approaches
4. A Fully Distributed Opportunistic Approach
5. Learning Based Approach

Time Synchronised Protocols:

These protocols make use of synchronization methods to schedule communication with neighboring nodes. With these methods it is possible for the nodes to arrive at a common time reference.

Deterministic Approach:

Deterministic approaches make use of mathematical methods to guarantee overlap. Though they require higher discovery latency, they do not need any synchronization. The searchlight protocol by Bakht et al can ensure

discovery within a definite latency by sequential search. This protocol
has two slots - Active slots and probe slots. The nodes decide to remain awake during active slots, and sleep during the remaining slots. During an active slot, the node may sendlreceive or do both. Successful discovery takes place between two neighboring nodes whenever their active slots overlap. When a discovery scheme uses few active slots to discover neighbors within a reasonable time limit, then it is said to be efficient. In Searchlight, each node has two active slots in t. The first
active slot, called the anchor slot, is the first slot in the period. In the symmetric case, since the position of this anchor slot is fixed in t but the start times for the t vary for different nodes, the anchor slots for two nodes overlap only if the difference between the start times of the two periods is less than a timeslot. The two anchor slots would never meet since the offset remains constant. The relative position of the anchor slot of one node remains the same with respect to that of the other node always and is in the range [1, t - 1]. Searchlight introduces a second active slot in t called the probe slot, which searches for the anchor slot of the other node. Probe-anchor, probe-probe and anchor-anchor overlaps all result in discovery. One disadvantage of this scheme is that it is a mobile agnostic method and is not well suited for mobile scenarios.

Colocation based Approaches:

These approaches make use of knowledge coming from neighbouring nodes. This knowledge is used to coordinate schedule between the nodes.

A Fully Distributed Opportunistic Approach:

In order to optimize the discovery of constrained nodes in loT network, Djamaa et al have proposed EADP (Efficient Application-layer Discovery Protocol). The protocol targets low power applications. EADP architecture has the following components - User agent, Service agent,Reply Agent. The user agent delivers a limited-flooding algorithm for discovering nodes in a pull-mode. The service agent is responsible for maintenance of remote service information. The service agent also has a state-maintenance mechanism that allows the protocol to react to topology changes. The reply agent is responsible for control and delivery of unicast replies. In addition, the Service Agent uses Trickle-inspired Algorithm which controls the frequency and size of messages that are used for advertising. Thus, the algorithm allows minimizing the number of advertising messages and also ensures a bound on the number of descriptions of the service.

Learning Based Approach:

Context Aware Resource Discovery framework leverages Q-Learning techniques to do node discovery. The system is able to reduce energy wastage when contacts are not expected while reducing discovery latency by employing an approach in which both mobile and static devices can learn their contact opportunities. Thus the time of contact useful for service is increased provisioning data collection.

Q-Learning is becoming widely used in communication scenarios for solving problems that require learning of knowledge to be exploited at a later time. This algorithm belongs to the Temporal Difference (TD) methods of Reinforcement Learning (RL) methods. The other two methods of RL are Dynamic Programming and Monte Carlo methods. TD methods combine the pros of the other two types of RL methods; that is, they don't require an accurate model of the environment.

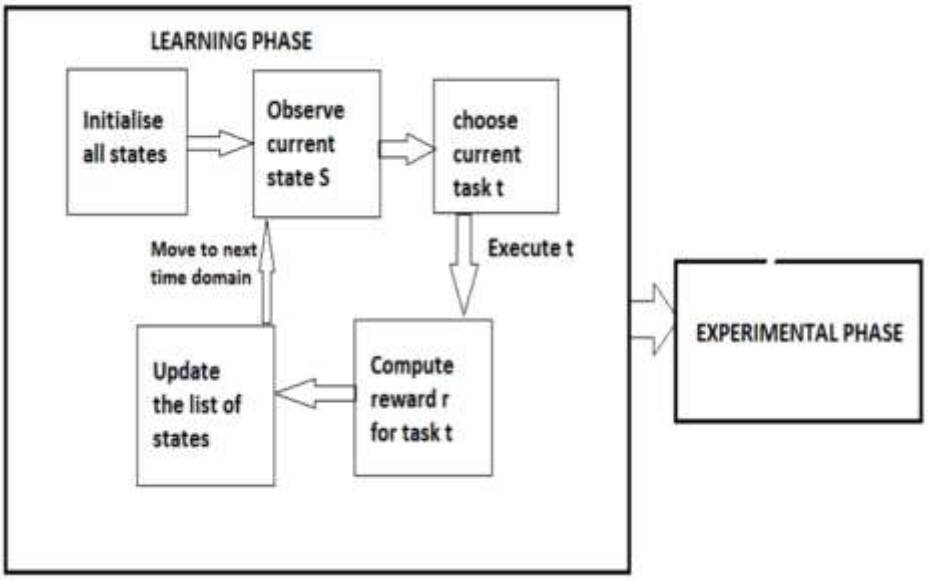


Figure 3.22. Learning based approach

After the completion of the learning phase, the framework enters the experimental phase. In this phase, the framework is tested with the real world data and optimal decisions are taken.

3.13 Data Aggregation :

Data Aggregation is the process of one or more sensor nodes and detects the information result from the other sensor nodes. The aim of the data aggregation is removes data redundancy and improves the energy lifetime in wireless sensor network. Therefore reducing the number of data packets transmitted over the network because aggregation need less power as compare to multiple packets sending having same data

Data Aggregation Approaches:

Data aggregation is further divided into four basic approaches.

a) Cluster based approach

b) Tree based approach

c) Multipath approach

d) Hybrid approach.

Cluster Based Approach: Cluster Based Approach is defined as the hierarchical approach in which whole network is separated into various cluster. Each cluster has cluster heads and which is cluster choose from members. The main role of cluster head aggregate data received from cluster members locally and then transmits the result to base station. The cluster head can share information with the sink directly via long range transmissions or multi hopping using other cluster heads.

Tree Based Approach: The Tree Based Approach is actually defining aggregation concept which is used to make aggregation tree. This tree define as minimum spanning tree in which sink node act as root and source node act as leaves. Data start flowing from leave nodes up to root nodes. The main disadvantage of this approach is data

packet loss at any level of tree which may cause failure whole network.

Multipath Approach: This approach is used to overcome the drawbacks of tree based approach. Accordingly to this approach each and every node could send data packets over multiple paths using multiple neighbors in aggregation tree. So a data packet sends from source to destination using multiple paths with the help of intermediate nodes. The example of this approach like ring topology. Overhead is the disadvantage of this approach

Hybrid Approach: The Hybrid Approach is the mixture of cluster based approach, multipath approach and tree based approach. This approach is mainly used for adaptively for optimal performance of their data aggregation

3.14 Data Dissemination:

Data Dissemination is the process in which sensor nodes is collecting the data and communicate to the base station or any other interested node. The source node is generating the data and the information to be reported is known as event. Those nodes which are interested in event and seek information are known as sink. So in this whole process data are routed in sensor network. It is two steps process; in first step interested nodes are broadcast to their neighbor nodes in the network and in second step nodes after receiving the request nodes sends requesting data.

Data Dissemination Approach:

There are many data dissemination methods or approaches which are following as:

Flooding: If the destination node is not receive the data packet or specified number of hops is not reached. Then each node broadcast the gathered data until the packet is reached to their destination node. The main advantage of flooding is not requires costly topology maintain or route discovery but it face several problems like implosion, overlap and resource blindness.

Gossiping: The gossiping is the version of flooding approach .In this approach the packet is sent to a single neighbor chosen from neighbour table randomly instead of broadcasting each packet to the entire neighbor. This process can take long time from completion. Gossiping avoids the problem faced in flooding approach like implosion.

SPIN: (Sensor Protocol for Information via Negotiation) this is the enhancement of flooding protocols based on data centric routing. Flooding has mainly three problems like: implosion, overlap and resource blindness. To overcome these problems the spin family protocols used three ways: ADV, REQ, DATA are used. The nodes which are interested in the event to transmit REQ message for DATA. After receiving REQ message source node sends DATA message to interested node. In this way data can reach to all interested node in entire network. This technique prevents the problems implosion, overlap and resource blindness which is faced by flooding. Wireless Sensor Network is the important in the field networking. A WSN is basically made up of number of several sensor nodes which work together. In this paper, we focus on existing various Data Dissemination and Data Aggregation approaches which is play very important role to increase the lifetime of network.

TEXT / REFERENCE BOOKS

1. Boswarthick,Omar Elloumi., The Internet of Things: Applications and Protocols, Wiley publications., 2012
2. Dieter Uckelmann, Mark Harrison, Florian Michahelles., Architecting the Internet of Things, Springer publications.2011
3. Marco Schwatrz Internet of Things with Arduino Cookbook, Packt Publications.2016 .
4. Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, “From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence”, 1st Edition, Academic Press, 2014.
5. Vijay Madisetti, Arshdeep Bahga, “Internet of Things: A Hands-On Approach” published by Vijay Madisetti 2014

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – 4 – Introduction to IoT– SCSA1308

Unit 4

IoT AND CLOUD

Interoperability in IoT - Introduction to Arduino Programming - Integration of Sensors and Actuators with Arduino - Cloud computing in IoT, IoT in cloud architecture, Logging on to cloud, - cloud based IoT platforms - IBM Watson, Google cloud

4.1 Interoperability in IoT

Internet of Things (IoT) is an ever-growing network of physical devices embedded with sensors, actuators and wireless connectivity to communicate and share their information among themselves. The application of IoT is in diverse areas such as agriculture, poultry and farming, smart city, and health care, where a sensor node must support heterogeneous sensors/actuators, and varying types of wireless connectivity. Interoperability is the ability of two or more devices, systems, platforms or networks to work in conjunction. Interoperability enables communication between heterogeneous devices or system in order to achieve a common goal. However, the current devices and systems are fragmented with respect to the communication technologies, protocols, and data formats. This diversity makes it difficult for devices and systems in the IoT network to communicate and share their data with one another. The utility of IoT network is limited by the lack of interoperability.

In IoT, one vital issue is interoperability among smart objects that is the ability to interconnect and communicate different vendors' systems to form a cost effective and easy to implement network.

Interoperability enables communication between heterogeneous devices or system in order to achieve a common goal. This diversity makes it difficult for devices and systems in the IoT network to communicate and share their data with one another. The utility of IoT network is limited by the lack of interoperability.

Interoperability is so hard because, hundreds of government-certified EHR products are in use across the country, each with different clinical terminologies, technical specifications, and functional capabilities. These differences make it difficult to create one standard interoperability format for sharing data.

Device Interoperability Testing

- Reliability of the application or device.
- Data protection and cybersecurity concerns.

- Performance during use.
- Compatibility with other devices or networks.
- Out of box testing for first time users.
- Identify defects throughout common use cycles.

4.2 Introduction to Arduino Programming

The Arduino software is open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL.

Sketch − The first new terminology is the Arduino program called “sketch”.

Structure

Arduino programs can be divided in three main parts: Structure, Values (variables and constants), and Functions.

Software structure consist of two main functions −

Setup( ) function

Loop( ) function

```
Void setup ( ) {


}
```

The setup() function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

```
Void Loop ( ) {


}
```

After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing the program to change and respond. Use it to actively control the Arduino board.

Sample sketch

When you create a file in Arduino* software, it opens up a sketch with the basic layout of an Arduino program. Here is the user interface:

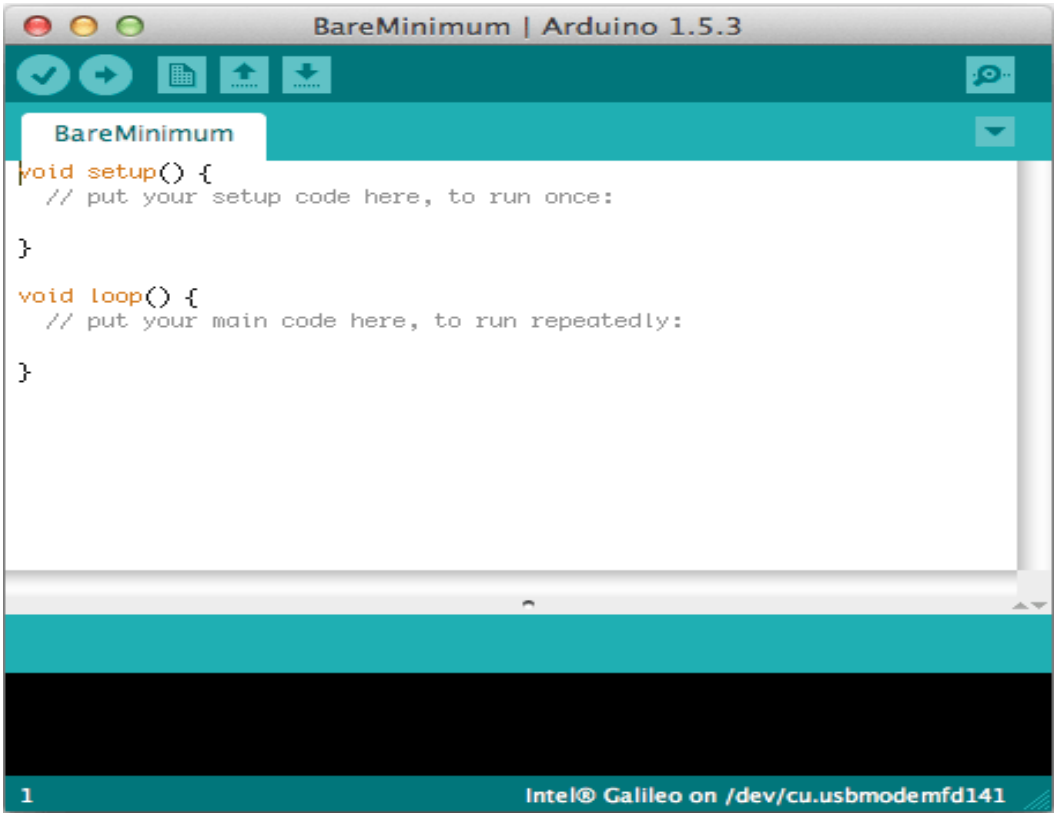


Fig 4.1 arduino program structure

From left to right, the icons at the top of the Arduino user interface represent the following:

Verify compiles code. Use to check your code for errors before uploading the sketch.

Upload a sketch.

New Editor Window opens a new code editing window in place of the current one.

Opens a file.

Saves a sketch.

Serial Monitor opens the serial monitor, useful for debugging

Down arrow gives you options like adding a sketch to the current project. It opens as a new tab in the current code editor that is useful for organizing your code into logical files.

Comments

The two forward slashes (between the {and}) represent the beginning of an inline code comment. When your code is uploaded to the board, the compiler ignores the text after the two slashes. Using the inline code comment allows you to leave notes for yourself, and for people reading your code. You can also write multiline comments by starting your comment with /* and ending with */.

/* You are reading an example of a comment that has many lines. */

Variables

Passing around data throughout a program can get messy quick. Variables are like storage containers that hold different types of values. Using variables to pass values around is a great way to keep your code organized and readable.

When declaring a variable (introducing it into the program), choosing the right data type is important. If you are trying to measure light intensity using a photometer, you might want a precise reading. Declaring a variable type of double reserves space in memory for a number with a decimal point.

Example: double light_sensitivity;

Where double is the type of variable you are declaring and light_sensitivity is the name of the variable. To reference a variable in your code, simply use the name you gave it.

Hello World Example for Intel® Galileo Boards

Pin 13

This content might look familiar if you've completed Intel® Galileo Board Getting Started.

Pin 13 has an LED connected to it. You can run your first sketch by accessing this onboard LED. This is the first step in connecting external LEDs and modules.

Most LEDs are sensitive when it comes to electricity. Just like the board, LEDs have a maximum amount of voltage they can take before they heat up causing damage. As an engineer, you learn to calculate the amount of resistance against the voltage to allow the right amount of current to flow throughout your circuit. Doing so keeps your equipment at a good temperature and under control. A resistor is used to limit the amount of current within a circuit.

Load sketch

The Arduino* software comes with code examples showing how to use some of the available modules for the Intel® Galileo Board. To run your first sketch, go to File > Examples > 01.Basics > Blink.

You should already have a connection with the serial port. To confirm, check out the bottom right side of the Arduino Integrated Development Environment.

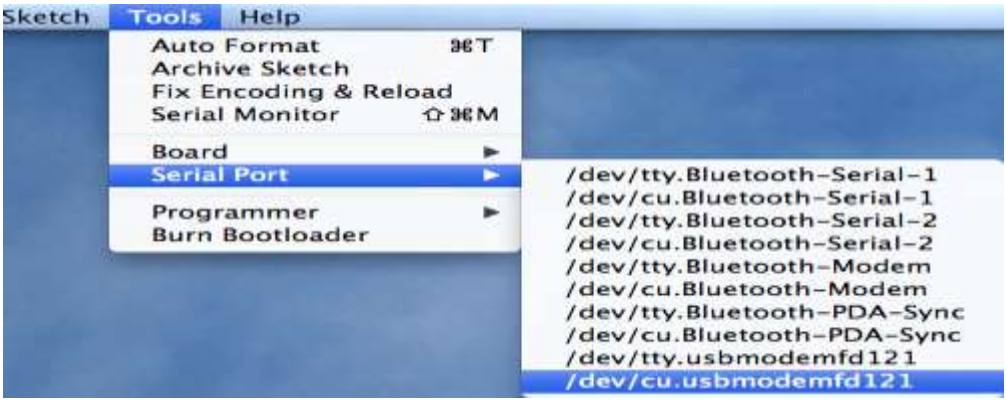


Fig 4.2 Execution of Arduino

Your serial port ID might be different. Simply select the one starting with cu.usbmodem. If you are using Windows*, the port begins with COM.

When you're ready to run your sketch, go to: File > Upload.

Once the sketch uploads, you should see an LED on the board blinking. This blinking LED is programmatically associated with pin 13.

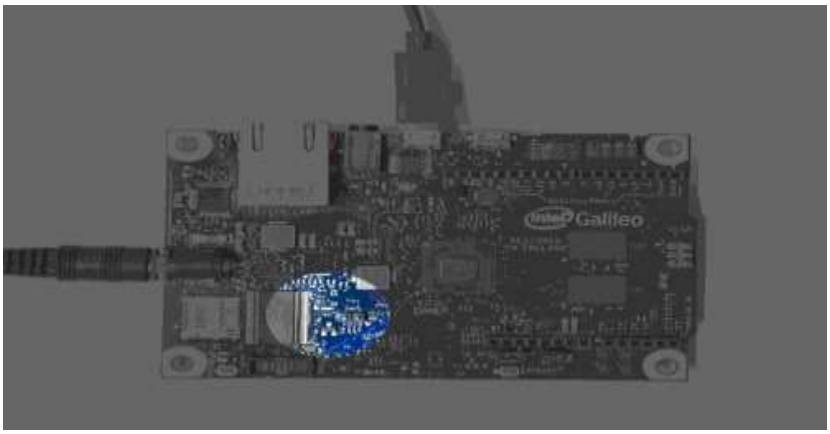


Fig 4.3 Blink Light

The pins on the Arduino board can be configured as either inputs or outputs. We will explain the functioning of the pins in those modes. It is important to note that a majority of Arduino analog pins, may be configured, and used, in exactly the same manner as digital pins.

Pins Configured as INPUT

Arduino pins are by default configured as inputs, so they do not need to be explicitly declared as inputs with pinMode() when you are using them as inputs. Pins configured this way are said to be in a high-impedance state. Input pins make extremely small demands on the circuit that they are sampling, equivalent to a series resistor of 100 megaohm in front of the pin.

This means that it takes very little current to switch the input pin from one state to another. This makes the pins useful for such tasks as implementing a capacitive touch sensor or reading an LED as a photodiode.

Pins configured as pinMode(pin, INPUT) with nothing connected to them, or with wires connected to them that are not connected to other circuits, report seemingly random changes in pin state, picking up electrical noise from the environment, or capacitively coupling the state of a nearby pin.

Pull-up Resistors

Pull-up resistors are often useful to steer an input pin to a known state if no input is present. This can be done by adding a pull-up resistor (to +5V), or a pull-down resistor (resistor to ground) on the input. A 10K resistor is a good value for a pull-up or pull-down resistor.

Using Built-in Pull-up Resistor with Pins Configured as Input

There are 20,000 pull-up resistors built into the Atmega chip that can be accessed from software. These built-in pull-up resistors are accessed by setting the pinMode() as INPUT_PULLUP. This effectively inverts the behavior of the INPUT mode, where HIGH means the sensor is OFF and LOW means the sensor is ON. The value of this pull-up depends on the microcontroller used. On most AVR-based boards, the value is guaranteed to be between 20kΩ and 50kΩ. On the Arduino Due, it is between 50kΩ and 150kΩ. For the exact value, consult the datasheet of the microcontroller on your board.

When connecting a sensor to a pin configured with INPUT_PULLUP, the other end should be connected to the ground. In case of a simple switch, this causes the pin to read HIGH when the switch is open and LOW when the switch is pressed. The pull-up resistors provide enough current to light an LED dimly connected to a pin configured as an input. If LEDs in a project seem to be working, but very dimly, this is likely what is going on.

Same registers (internal chip memory locations) that control whether a pin is HIGH or LOW control the pull-up resistors. Consequently, a pin that is configured to have pull-up resistors turned on when the pin is in INPUTmode, will have the pin configured as HIGH if the pin is then switched to an OUTPUT mode with pinMode(). This works in the other direction as well, and an output pin that is left in a HIGH state will have the pull-up resistor set if switched to an input with pinMode().

Example

```
pinMode(3,INPUT) ; // set pin to input without using built in pull up resistor
```

pinMode(5,INPUT_PULLUP) ; // set pin to input using built in pull up resistor

Pins Configured as OUTPUT

Pins configured as OUTPUT with pinMode() are said to be in a low-impedance state. This means that they can provide a substantial amount of current to other circuits. Atmega pins can source (provide positive current) or sink (provide negative current) up to 40 mA (milliamps) of current to other devices/circuits. This is enough current to brightly light up an LED (do not forget the series resistor), or run many sensors but not enough current to run relays, solenoids, or motors.

Attempting to run high current devices from the output pins, can damage or destroy the output transistors in the pin, or damage the entire Atmega chip. Often, this results in a "dead" pin in the microcontroller but the remaining chips still function adequately. For this reason, it is a good idea to connect the OUTPUT pins to other devices through 470Ω or 1k resistors, unless maximum current drawn from the pins is required for a particular application.

pinMode() Function

The pinMode() function is used to configure a specific pin to behave either as an input or an output. It is possible to enable the internal pull-up resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pull-ups.

pinMode() Function Syntax

```
Void setup () {
  pinMode (pin , mode);
}
```

pin − the number of the pin whose mode you wish to set

mode − INPUT, OUTPUT, or INPUT_PULLUP.

Example

```
int button = 5 ; // button connected to pin 5
int LED = 6; // LED connected to pin 6
```

```
void setup () {
  pinMode(button , INPUT_PULLUP);
  // set the digital pin as input with pull-up resistor
  pinMode(button , OUTPUT); // set the digital pin as output
}


void setup () {
  If (digitalRead(button ) == LOW) // if button pressed {
    digitalWrite(LED,HIGH); // turn on led
    delay(500); // delay for 500 ms
    digitalWrite(LED,LOW); // turn off led
    delay(500); // delay for 500 ms
  }
}
```

digitalWrite() Function

The digitalWrite() function is used to write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW. If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin. It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor.

If you do not set the pinMode() to OUTPUT, and connect an LED to a pin, when calling digitalWrite(HIGH), the LED may appear dim. Without explicitly setting pinMode(), digitalWrite() will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

digitalWrite() Function Syntax

```
Void loop() {
  digitalWrite (pin ,value);
}
```

pin − the number of the pin whose mode you wish to set

value − HIGH, or LOW.

Example

```
int LED = 6; // LED connected to pin 6

void setup () {
  pinMode(LED, OUTPUT); // set the digital pin as output
}

void setup () {
  digitalWrite(LED,HIGH); // turn on led
  delay(500); // delay for 500 ms
  digitalWrite(LED,LOW); // turn off led
  delay(500); // delay for 500 ms
}
```

analogRead( ) function

Arduino is able to detect whether there is a voltage applied to one of its pins and report it through the digitalRead() function. There is a difference between an on/off sensor (which detects the presence of an object) and an analog sensor, whose value continuously changes. In order to read this type of sensor, we need a different type of pin.

In the lower-right part of the Arduino board, you will see six pins marked “Analog In”. These special pins not only tell whether there is a voltage applied to them, but also its value. By using the analogRead() function, we can read the voltage applied to one of the pins.

This function returns a number between 0 and 1023, which represents voltages between 0 and 5 volts. For example, if there is a voltage of 2.5 V applied to pin number 0, analogRead(0) returns 512.

analogRead() function Syntax

```
analogRead(pin);
```

pin − the number of the analog input pin to read from (0 to 5 on most boards, 0 to 7 on the Mini and Nano, 0 to 15 on the Mega)

Example

```
int analogPin = 3;//potentiometer wiper (middle terminal)
  // connected to analog pin 3
int val = 0; // variable to store the value read
void setup() {
  Serial.begin(9600); // setup serial
}
void loop() {
  val = analogRead(analogPin); // read the input pin
  Serial.println(val); // debug value
}
```

4.3 Integration of Sensors and Actuators with Arduino

Humidity Sensor (DHT22)

The DHT-22 (also named as AM2302) is a digital-output, relative humidity, and temperature sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and sends a digital signal on the data pin.

In this example, you will learn how to use this sensor with Arduino UNO. The room temperature and humidity will be printed to the serial monitor.

The DHT-22 Sensor

Fig 4.4 DHT-22 Sensor

The connections are simple. The first pin on the left to 3-5V power, the second pin to the

data input pin and the right-most pin to the ground.

Technical Details

Power − 3-5V

Max Current − 2.5mA

Humidity − 0-100%, 2-5% accuracy

Temperature − 40 to 80°C, ±0.5°C accuracy

Components Required

You will need the following components −

1 × Breadboard

1 × Arduino Uno R3

1 × DHT22

1 × 10K ohm resistor

Procedure

Follow the circuit diagram and hook up the components on the breadboard as shown in the image below.

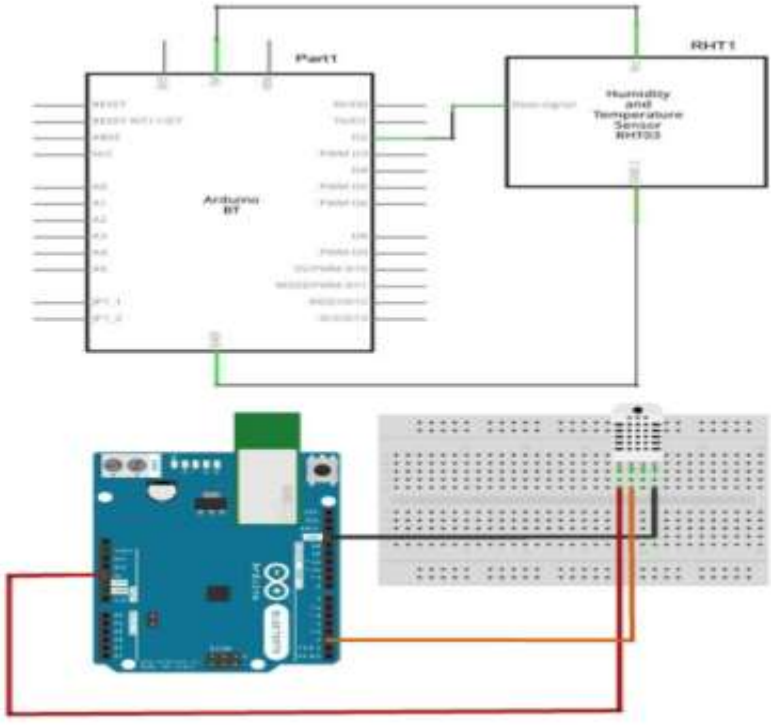


Fig 4.5: circuit diagram

Sketch

Open the Arduino IDE software on your computer. Coding in the Arduino language will control your circuit. Open a new sketch File by clicking New.

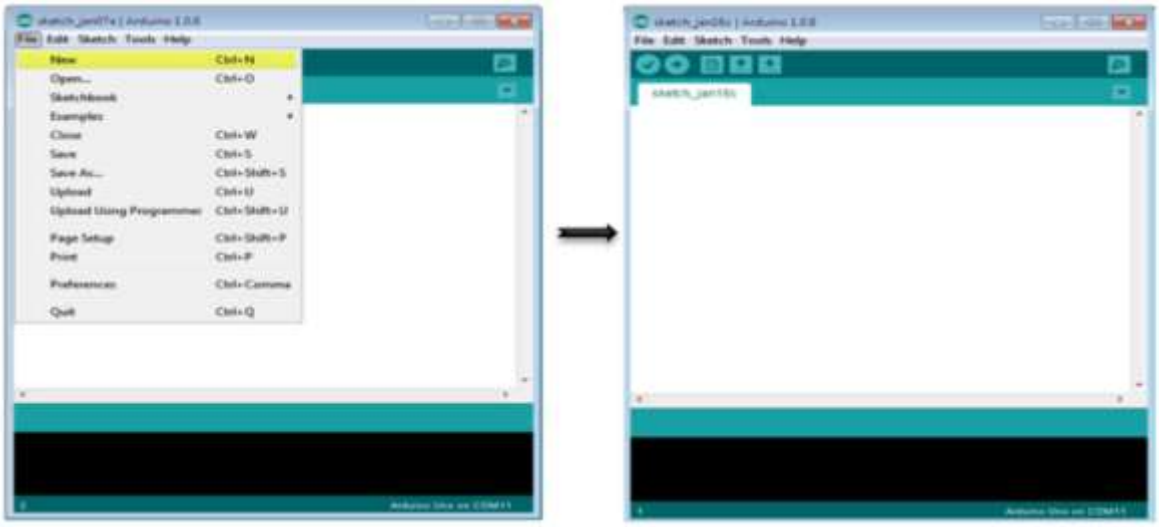


Fig 4.6: Procedure to open new sketch

```
// Example testing sketch for various DHT humidity/temperature sensors

#include "DHT.h"

#define DHTPIN 2 // what digital pin we're connected to

// Uncomment whatever type you're using!

//#define DHTTYPE DHT11 // DHT 11

#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

//#define DHTTYPE DHT21 // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V

// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1

// to 3.3V instead of 5V!

// Connect pin 2 of the sensor to whatever your DHTPIN is

// Connect pin 4 (on the right) of the sensor to GROUND

// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.

// Note that older versions of this library took an optional third parameter to

// tweak the timings for faster processors. This parameter is no longer needed
```

```
// as the current DHT reading algorithm adjusts itself to work on faster procs.
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");
  dht.begin();
}
void loop() {
  delay(2000); // Wait a few seconds between measurements
  float h = dht.readHumidity();
  // Reading temperature or humidity takes about 250 milliseconds!
  float t = dht.readTemperature();
  // Read temperature as Celsius (the default)
  float f = dht.readTemperature(true);
  // Read temperature as Fahrenheit (isFahrenheit = true)
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print ("Humidity: ");
  Serial.print (h);
  Serial.print (" %\t");
```

```
  Serial.print ("Temperature: ");
  Serial.print (t);
  Serial.print (" *C ");
  Serial.print (f);
  Serial.print (" *F\t");
  Serial.print ("Heat index: ");
  Serial.print (hic);
  Serial.print (" *C ");
  Serial.print (hif);
  Serial.println (" *F");
}
```

DHT22 sensor has four terminals (Vcc, DATA, NC, GND), which are connected to the board as follows −

DATA pin to Arduino pin number 2

Vcc pin to 5 volt of Arduino board

GND pin to the ground of Arduino board

We need to connect 10k ohm resistor (pull up resistor) between the DATA and the Vcc pin

4.4 SERVO MOTOR

A Servo Motor is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. If the coded signal changes, the angular position of the shaft changes. In practice, servos are used in radio-controlled airplanes to position control surfaces like the elevators and rudders. They are also used in radio-controlled cars, puppets, and of course, robots. Servos are extremely useful in robotics. The motors are small, have built-in control circuitry, and are extremely powerful for their size. A standard servo such as the Futaba S-148 has 42 oz/inches of torque, which is strong for its size. It also draws power proportional to the mechanical load. A lightly loaded servo, therefore, does not consume much energy.

The guts of a servo motor is shown in the following picture. You can see the control circuitry, the motor, a set of gears, and the case. You can also see the 3 wires that connect to the outside world. One is for power (+5volts), ground, and the white wire is the control wire.

Fig 4.7: Servo motor

Working of a Servo Motor

The servo motor has some control circuits and a potentiometer (a variable resistor, aka pot) connected to the output shaft. In the picture above, the pot can be seen on the right side of the circuit board. This pot allows the control circuitry to monitor the current angle of the servo motor.If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will turn the motor until it is at a desired angle. The output shaft of the servo is capable of traveling somewhere around 180 degrees. Usually, it is somewhere in the 210-degree range, however, it varies depending on the manufacturer. A normal servo is used to control an angular motion of 0 to 180 degrees. It is mechanically not capable of turning any farther due to a mechanical stop built on to the main output gear. The power applied to the motor is proportional to the distance it needs to travel. So, if the shaft needs to turn a large distance, the motor will run at full speed. If it needs to turn only a small amount, the motor will run at a slower speed. This is called proportional control.

To Communicate the Angle at Which the Servo Should Turn

The control wire is used to communicate the angle. The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse Coded Modulation. The servo expects to see a pulse every 20 milliseconds (.02 seconds). The length of the pulse will determine how far the motor turns. A 1.5 millisecond pulse, for example, will make the motor turn to the 90-degree position (often called as the neutral position). If the pulse is shorter than 1.5 milliseconds, then the motor will turn the shaft closer to 0 degrees. If the pulse is longer than 1.5 milliseconds, the shaft turns closer to 180 degrees.

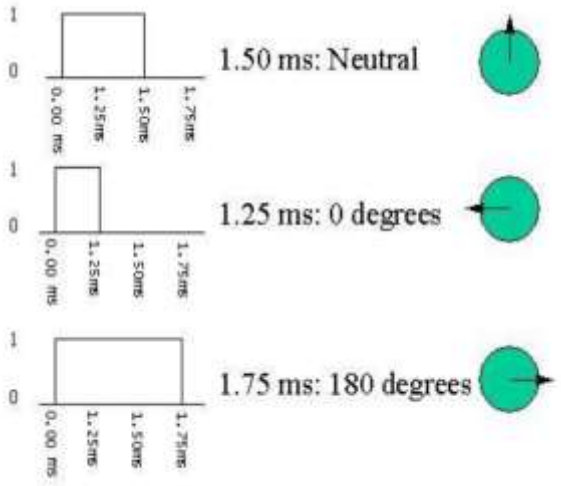


Fig 4.8: Communicate angle

Components Required

Need the following components −

1 × Arduino UNO board

1 × Servo Motor

1 × ULN2003 driving IC

1 × 10 KΩ Resistor

Procedure

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

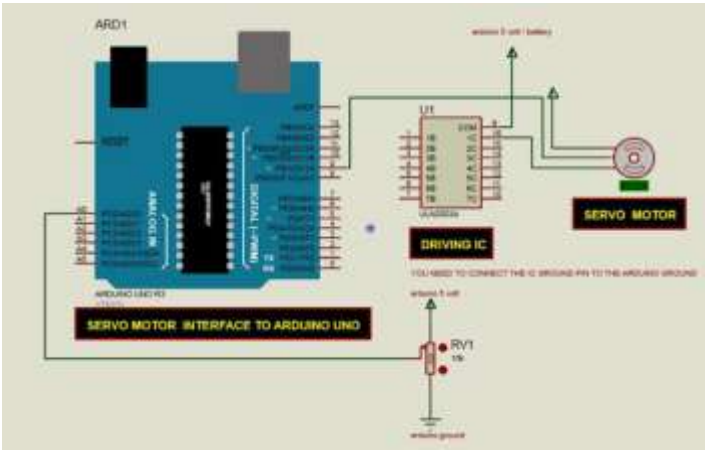


Fig 4.9: Servo motor connection

Sketch

Open the Arduino IDE software on your computer. Coding in the Arduino language will control your circuit. Open a new sketch File by clicking on New.

```
/* Controlling a servo position using a potentiometer (variable resistor) */
  #include <Servo.h>
  Servo myservo; // create servo object to control a servo
  int potpin = 0; // analog pin used to connect the potentiometer
  int val; // variable to read the value from the analog pin
void setup() {
  myservo.attach(9); // attaches the servo on pin 9 to the servo object
}
void loop() {
  val = analogRead(potpin);
  // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);
  // scale it to use it with the servo (value between 0 and 180)
  myservo.write(val); // sets the servo position according to the scaled value
  delay(15);
}
```

Servo motors have three terminals - power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino. The ground wire is typically black or brown and should be connected to one terminal of ULN2003 IC (10 -16). To protect your Arduino board from damage, you will need some driver IC to do that. Here we have used ULN2003 IC to drive the servo motor. The signal pin is typically yellow or orange and should be connected to Arduino pin number 9.

Connecting the Potentiometer

A voltage divider/potential divider are resistors in a series circuit that scale the output voltage to a particular ratio of the input voltage applied. Following is the circuit diagram −

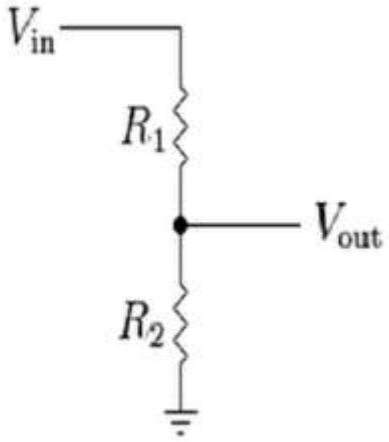


Fig 4.10: Pot connection

$$V_{out} = (V_{in} \times R_{2})/ (R_{1} + R_{2})$$

Vout is the output potential, which depends on the applied input voltage (Vin) and resistors (R1 and R2) in the series. It means that the current flowing through R1 will also flow through R2 without being divided. In the above equation, as the value of R2 changes, the Vout scales accordingly with respect to the input voltage, Vin.

Typically, a potentiometer is a potential divider, which can scale the output voltage of the circuit based on the value of the variable resistor, which is scaled using the knob. It has three pins: GND, Signal, and +5V as shown in the diagram below −

Fig 4.11: Potentiometer

4.5 Cloud computing in IoT

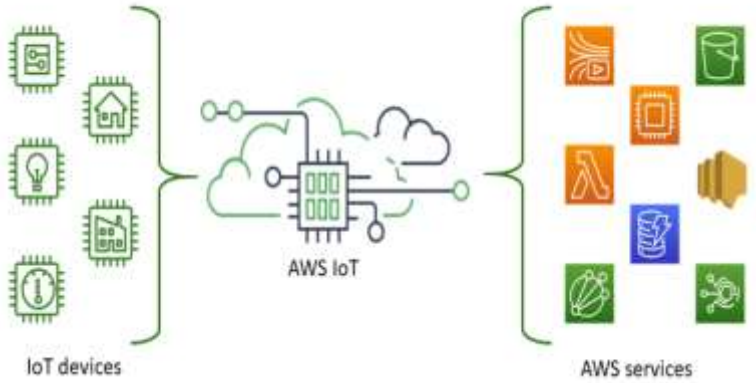


Fig 4.12: cloud computing

AWS IoT lets to select the most appropriate and up-to-date technologies for your solution. To help you manage and support your IoT devices in the field, AWS IoT Core supports these protocols:

- MQTT (Message Queuing and Telemetry Transport)
- MQTT over WSS (Websockets Secure)
- HTTPS (Hypertext Transfer Protocol - Secure)
- LoRaWAN (Long Range Wide Area Network)

The AWS IoT Core message broker supports devices and clients that use MQTT and MQTT over WSS protocols to publish and subscribe to messages. It also supports devices and clients that use the HTTPS protocol to publish messages. AWS IoT Core for LoRaWAN helps you connect and manage wireless LoRaWAN (low-power long-range Wide Area Network) devices. AWS IoT Core for LoRaWAN replaces the need for you to develop and operate a LoRaWAN Network Server (LNS). If you don't require AWS IoT features such as device communications, rules, or jobs, see AWS Messaging for information about other AWS IoT messaging services that might better fit your requirements.

The advent of cloud computing has acted as a catalyst for the development and deployment of scalable Internet-of-Things business models and applications. Therefore, IoT and cloud are nowadays two very closely affiliated future internet technologies, which go hand-in-hand in non-trivial IoT deployments. Cloud computing is the next evolutionary step in Internet-based computing, which provides the means for delivering ICT resources as a service. Cloud computing is a technology that uses the internet for storing and managing data on remote servers, and then access data via the internet. The ICT resources that can be delivered through cloud computing model include computing power, computing infrastructure (e.g., servers

and/or storage resources), applications, business processes and more. Cloud computing infrastructures and services have the following characteristics, which typically differentiate them from similar (distributed computing), technologies:

- **Elasticity and the ability to scale up and down**: Cloud computing services can scale upwards during high periods of demand and downward during periods of lighter demand. This elastic nature of cloud computing facilitates the implementation of flexibly scalable business models, e.g., through enabling enterprises to use more or less resources as their business grows or shrinks.

- **Self-service provisioning and automatic deprovisioning**: Contrary to conventional web-based Application Service Providers (ASP) models (e.g., web hosting), cloud computing enables easy access to cloud services without a lengthy provisioning process. In cloud computing, both provisioning and de-provisioning of resources can take place automatically.

- **Application programming interfaces (APIs)**: Cloud services are accessible via APIs, which enable applications and data sources to communicate with each other.

- **Billing and metering of service usage in a pay-as-you-go model**: Cloud services are associated with a utility-based pay-as-you-go model. To this end, they provide the means for metering resource usage and subsequently issuing bills.

- **Performance monitoring and measuring**: Cloud computing infrastructures provide a service management environment along with an integrated approach for managing physical environments and IT systems.

- **Security**: Cloud computing infrastructures offer security functionalities towards safeguarding critical data and fulfilling customers' compliance requirements.

The two main business drivers behind the adoption of a cloud computing model and associated services including:

- **Business Agility:** Cloud computing alleviates tedious IT procurement processes, since it facilitates flexible, timely and on-demand access to computing resources (i.e. compute cycles, storage) as needed to meet business targets.

Depending on the types of resources that are accessed as a service, cloud computing is associated with different service delivery models.

- **Infrastructure as a Service (IaaS)**: IaaS deals with the delivery of storage and computing resources towards supporting custom business solutions. Enterprises opt for an IaaS cloud computing model in order to benefit from lower prices, the ability to aggregate resources, accelerated deployment, as well as increased and customized security. The most prominent example of IaaS service Amazon's Elastic Compute Cloud (EC2), which uses the Xen open-source hypervisor to create and manage virtual machines.

- **Platform as a Service (PaaS)**: PaaS provides development environments for creating cloud-ready business applications. It provides a deeper set of capabilities comparing to IaaS, including development, middleware, and deployment capabilities. PaaS services create and encourage deep ecosystem of partners who commit to this environment. Typical examples of PaaS services are Google's App Engine and Microsoft's Azure cloud environment, which both provide a workflow engine, development tools, a testing environment, database integration functionalities, as well as third-party tools and services.

- **Software as a Service (SaaS)**: SaaS services enable access to purpose-built business applications in the cloud. Such services provide the pay-go-go, reduced CAPEX and elastic properties of cloud computing infrastructures.

  Cloud services can be offered through infrastructures (clouds) that are publicly accessible (i.e. public cloud services), but also by privately owned infrastructures (i.e. private cloud services). Furthermore, it is possible to offer services supporting by both public and private clouds, which are characterized as hybrid cloud services.

4.6 IoT/Cloud Convergence

Internet-of-Things can benefit from the scalability, performance and pay-as-you-go nature of cloud computing infrastructures. Indeed, as IoT applications produce large volumes of data and comprise multiple computational components (e.g., data processing and analytics algorithms), their integration with cloud computing infrastructures could provide them with opportunities for cost-effective on-demand scaling. As prominent examples consider the following settings:

- A Small Medium Enterprise (SME) developing an energy management IoT product, targeting smart homes and smart buildings. By streaming the data of the product (e.g., sensors and WSN data) into the cloud it can accommodate its growth needs in a scalable and cost effective fashion.

- A smart city can benefit from the cloud-based deployment of its IoT systems and applications. A city is likely to deploy many IoT applications, such as applications for smart energy management, smart water management, smart transport management, urban mobility of the citizens and more. These applications comprise multiple sensors and devices, along with computational components. Furthermore, they are likely to produce very large data volumes. Cloud integration enables the city to host these data and applications in a cost-effective way. Furthermore, the elasticity of the cloud can directly support expansions to these applications, but also the rapid deployment of new ones without major concerns about the provisioning of the required cloud computing resources.

- A cloud computing provider offering pubic cloud services can extend them to the IoT area, through enabling third-parties to access its infrastructure in order to integrate IoT data and/or computational components operating over IoT devices. The provider can offer IoT data access and services in a pay-as-you-fashion, through enabling third-parties to access resources of its infrastructure and accordingly to charge them in a utility-based fashion.

  One of the earliest efforts has been the famous Pachube.com infrastructure (used extensively for radiation detection and production of radiation maps during earthquakes in Japan). Pachube.com has evolved (following several evolutions and acquisitions of this infrastructure) to Xively.com, which is nowadays one of the most prominent public IoT clouds. Nevertheless, there are tens of other public IoT clouds as well, such as ThingsWorx, ThingsSpeak, Sensor-Cloud, Realtime.io and more. The list is certainly non-exhaustive. These public IoT clouds offer commercial pay-as-you-go access to end-users wishing to deploying IoT applications on the cloud. Most of them come with developer friendly tools, which enable the development of cloud applications, thus acting like a PaaS for IoT in the cloud.

  Similarly to cloud computing infrastructures, IoT/cloud infrastructures and related services can be classified to the following models:

1. **Infrastructure-as-a-Service (IaaS) IoT/Clouds**: These services provide the means for accessing sensors and actuator in the cloud. The associated business model involves the IoT/Cloud provide to act either as data or sensor provider. IaaS services for IoT provide access control to resources as a prerequisite for the offering of related pay-as-you-go services.

2. **Platform-as-a-Service (PaaS) IoT/Clouds**: This is the most widespread model for IoT/cloud services, given that it is the model provided by all public IoT/cloud infrastructures outlined above. As already illustrate most public IoT clouds come with a range of tools and related

environments for applications development and deployment in a cloud environment. A main characteristic of PaaS IoT services is that they provide access to data, not to hardware. This is a clear differentiator comparing to IaaS.

3. **Software-as-a-Service (SaaS) IoT/Clouds**: SaaS IoT services are the ones enabling their uses to access complete IoT-based software applications through the cloud, on-demand and in a pay-as-you-go fashion. As soon as sensors and IoT devices are not visible, SaaS IoT applications resemble very much conventional cloud-based SaaS applications.

The benefits of integrating IoT into Cloud are discussed in this section as follows.

a. Communication

The Cloud is an effective and economical solution which can be used to connect, manage, and track anything by using built-in apps and customized portals . The availability of fast systems facilitates dynamic monitoring and remote objects control, as well as data real-time access. It is worth declaring that, although the Cloud can greatly develop and facilitate the IoT interconnection, it still has weaknesses in certain areas. Thus, practical restrictions can appear when an enormous amount of data needs to be transferred from the Internet to the Cloud.

b. Storage

As the IoT can be used on billions of devices, it comprises a huge number of information sources, which generate an enormous amount of semi-structured or non-structured data . This is known as Big Data, and has three characteristics : variety (e.g. data types), velocity (e.g. data generation frequency), and volume (e.g. data size). The Cloud is considered to be one of the most cost-effective and suitable solutions when it comes to dealing with the enormous amount of data created by the IoT. Moreover, it produces new chances for data integration, aggregation, and sharing with third parties .

c. Processing capabilities

IoT devices are characterized by limited processing capabilities which prevent on-site and complex data processing. Instead, gathered data is transferred to nodes that have high capabilities; indeed, it is here that aggregation and processing are accomplished. However, achieving scalability remains a challenge without an appropriate underlying infrastructure. Offering a solution, the Cloud provides unlimited virtual processing capabilities and an on-

demand usage model . Predictive algorithms and data-driven decisions making can be integrated into the IoT in order to increase revenue and reduce risks at a lower cost .

d. Scope

With billions of users communicating with one another together and a variety of information being collected, the world is quickly moving towards the Internet of Everything (IoE) realm - a network of networks with billions of things that generate new chances and risks . The Cloud-based IoT approach provides new applications and services based on the expansion of the Cloud through the IoT objects, which in turn allows the Cloud to work with a number of new real world scenarios, and leads to the emergence of new services .

e. New abilities

The IoT is characterised by the heterogeneity of its devices, protocols, and technologies. Hence, reliability, scalability, interoperability, security, availability and efficiency can be very hard to achieve. Integrating IoT into the Cloud resolves most of these issues. It provides other features such as easeof-use and ease-of-access, with low deployment costs .

f. New Models

Cloud-based IoT integration empowers new scenarios for smart objects, applications, and services. Some of the new models are listed as follows:

- SaaS (Sensing as a Service) , which allows access to sensor data;
- EaaS (Ethernet as a Service), the main role of which is to provide ubiquitous connectivity to control remote devices;
- SAaaS (Sensing and Actuation as a Service), which provides control logics automatically.
- IPMaaS (Identity and Policy Management as a Service) , which provides access to policy and identity management.
- DBaaS (Database as a Service), which provides ubiquitous database management;
- SEaaS (Sensor Event as a Service) , which dispatches messaging services that are generated by sensor events;
- SenaaS (Sensor as a Service) , which provides management for remote sensors;
- DaaS (Data as a Service), which provides ubiquitous access to any type of data.

4.7 IoT in Cloud Architecture

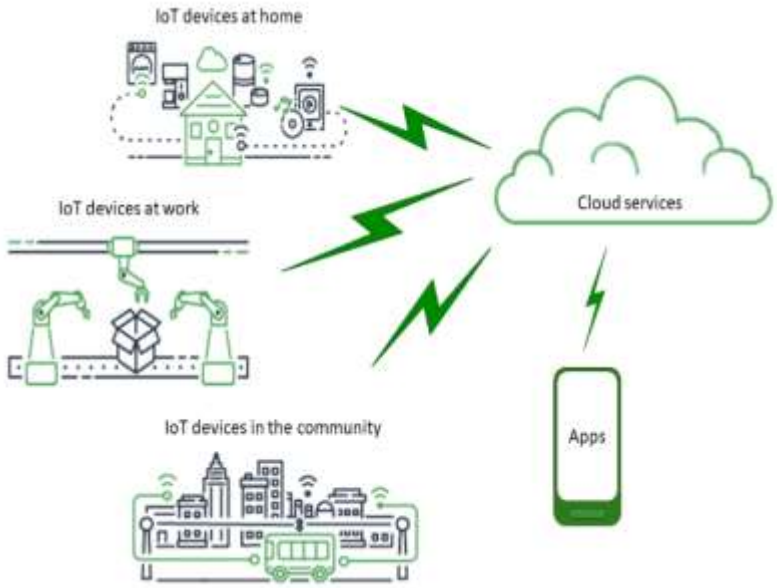
The cloud components of IoT architecture are positioned within a three-tier architecture pattern comprising edge, platform and enterprise tiers, as described in the Industrial Internet

Consortium Reference Architecture.

- The edge-tier includes Proximity Networks and Public Networks where data is collected from devices and transmitted to devices. Data flows through the IoT gateway or optionally directly from/to the device then through edge services into the cloud provider via IoT transformation and connectivity.

- The Platform tier is the provider cloud, which receives processes and analyzes data flows from the edge tier and provides API Management and Visualization. It provides the capability to initiate control commands from the enterprise network to the public network as well.

- The Enterprise tier is represented by the Enterprise Network comprised of Enterprise Data, Enterprise User Directory, and Enterprise Applications. The data flow to and from the enterprise network takes place via a Transformation and Connectivity component. The data collected from structured and non-structured data sources, including real-time data from stream computing, can be stored in the enterprise data.



4.13 : Fig IoT in Cloud Architecture

Apps

Apps give end users access to IoT devices and the features provided by the cloud services to which those devices are connected.

Cloud services

Cloud services are distributed, large-scale data storage and processing services that are connected to the internet. Examples include:

- IoT connection and management services.
- AWS IoT is an example of an IoT connection and management service.
- Compute services, such as Amazon Elastic Compute Cloud and AWS Lambda.
- Database services, such as Amazon DynamoDB

Communications

Devices communicate with cloud services by using various technologies and protocols. Examples include:

- Wi-Fi/Broadband internet
- Broadband cellular data
- Narrow-band cellular data
- Long-range Wide Area Network (LoRaWAN)
- Proprietary RF communications

Devices

A device is a type of hardware that manages interfaces and communications. Devices are usually located in close proximity to the real-world interfaces they monitor and control. Devices can include computing and storage resources, such as microcontrollers, CPU, memory. Examples include:

- Raspberry Pi
- Arduino
- Voice-interface assistants
- LoRaWAN and devices
- Amazon Sidewalk devices

- Custom IoT devices

Interfaces

An interface is a component that connects a device to the physical world.

- **User interfaces:** Components that allow devices and users to communicate with each other.
- **Input interfaces:** Enable a user to communicate with a device. Examples: keypad, button

- **Output interfaces:** Enable a device to communicate with a user. Examples: Alpha-numeric display, graphical display, indicator light, alarm bell

Sensors

Input components that measure or sense something in the outside world in a way that a device understands. Examples include:

- Temperature sensor (converts temperature to an analog voltage)
- Humidity sensor (converts relative humidity to an analog voltage)
- Analog to digital convertor (converts an analog voltage to a numeric value)
- Ultrasonic distance measuring unit (converts a distance to a numeric value)
- Optical sensor (converts a light level to a numeric value)
- Camera (converts image data to digital data)

Actuators

Output components that the device can use to control something in the outside world. Examples include:

- Stepper motors (convert electric signals to movement)
- Relays (control high electric voltages and currents)

One of the features of IoT systems is the need for application logic and control logic in a hierarchy of locations, depending on the timescales involved and the datasets that need to be brought to bear on the decisions that need to be made.

Some code may execute directly in the devices at the very edge of the network, or alternatively in the IoT Gateways close to the devices. Other code executes centrally in the provider cloud services or in the enterprise network. This is sometimes alternatively called "fog computing" to contrast with centralised "cloud computing", although fog computing can also contain one or more layers below the cloud that each could potentially provide capabilities for a variety of services like analytics.

Aspects of the architecture include:

- The user layer is independent of any specific network domain. It may be in or outside any specific domain.

- The proximity network domain has networking capabilities that typically extend the public network domain. The devices (including sensor/actuator, firmware and management agent) and the physical entity are part of the proximity network domain. The devices communicate for both data flow and control flow either via an IoT Gateway and edge services or directly over the public network via edge services.

- The public network and enterprise network domains contain data sources that feed the entire architecture. Data sources include traditional systems of record from the enterprise as well as new sources from Internet of Things (IoT). The public network includes communication with peer clouds.

- The provider cloud captures data from devices, peer cloud services and other data sources (for example Weather services). It can use integration technologies or stream processing to transform, filter and analyse this data in real time and it can store the data into repositories where further analytics can be performed. This processing, which can be augmented with the use of Cognitive and Predictive analytics, is used to generate Actionable Insights. These insights are used by users and enterprise applications and can also be used to trigger actions to be performed by IoT Actuators. All of this needs to be done in a secure and governed environment.

The following figure shows the capabilities and relationships for supporting IoT using cloud computing.

User Layer - contains IoT users and their end user applications.

- IoT User (people/system) - a person or alternatively an automated system that makes use of one or more end user applications to achieve some goal. The IoT User is one of the main beneficiaries of the IoT solution.

- End User Application - domain specific or device specific application. The IoT user may use end user applications that run on smart phones, tablets, PCs or alternatively on specialised IoT devices including control panels.

- Proximity Network - contains the physical entities that are at the heart of the IoT system, along with the devices that interact with the physical entities and connect them to the IoT system.

- **Physical Entity** - the physical entity is the real-world object that is of interest – it is subject

to sensor measurements or to actuator behavior. It is the “thing” in the Internet of Things. This architecture distinguishes between the physical entities and the IT devices that sense them or act on them. For example, the thing can be the ocean and the device observing is it a water temperature thermometer.

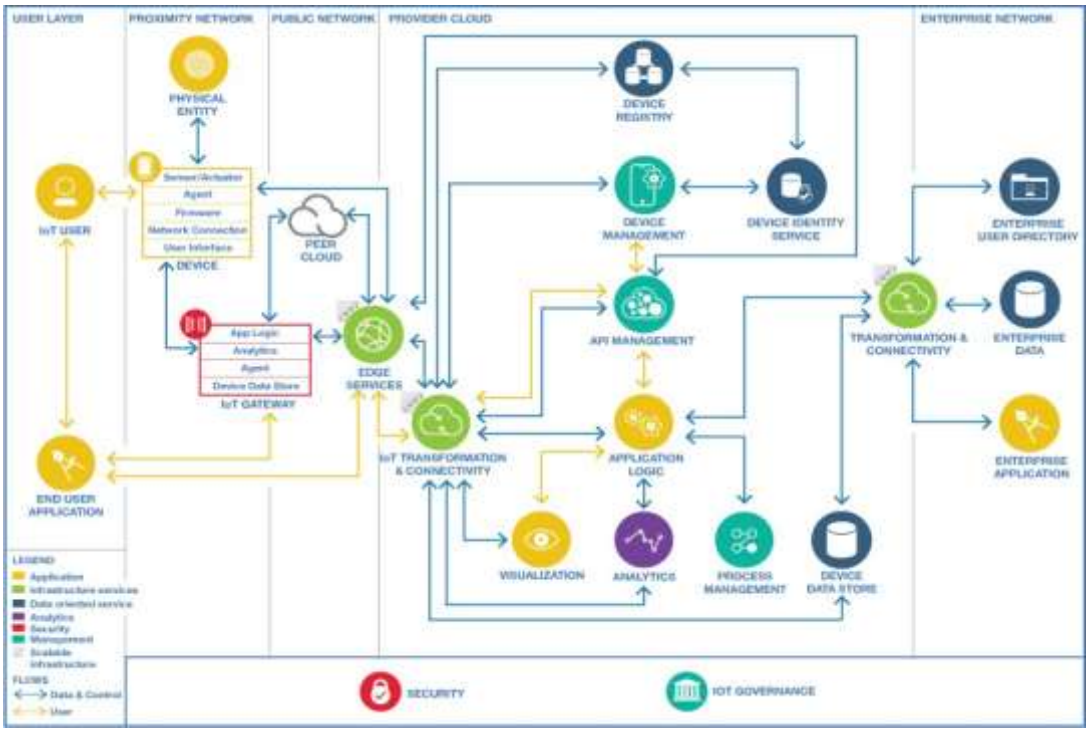


Fig 4.14 Cloud Components for IoT

Device - contains sensor(s) and/or actuator(s) plus a network connection that enables interaction with the wider IoT system. There are cases where the device is also the physical entity being monitored by the sensors – such as an accelerometer inside a smart phone.

- Sensor/Actuator - senses and acts on physical entities. A sensor is a component that senses or measures certain characteristics of the real world and converts them into a digital representation. An actuator is a component that accepts a digital command to act on a physical entity in some way.

- Agent - provides remote management capabilities for the device, supporting a device management protocol that can be used by the Device Management service or IoT management system.

- Firmware - software that provides control, monitoring and data manipulation of engineered products and systems. The firmware contained in devices such as consumer electronics provides the low-level control program for the devices.

- Network Connection - provides the connection from the device to the IoT system. This is often a local network that connects the device with an IoT gateway – low power and low range in many cases to reduce the power demands on the device.

- User Interface - allows users to interact with applications, agents, sensors and actuators (optional – some devices have no user interface and all interaction takes place from remote applications over the network).

IoT Gateway - acts as a means for connecting one or more devices to the public network (typically the Internet). It is commonly the case that devices have limited network connectivity – they may not be able to connect directly to the Internet. This can be for a number of reasons, including the limitation of power on the device, which can restrict the device to using a low-power local network. The local network enables the devices to communicate with a local IoT Gateway, which is then able to communicate with the public network. The IoT Gateway contains the following components:

- App Logic - provides domain specific or IoT solution specific logic that runs on the IoT Gateway. For IoT systems that have Actuators which act on physical entities, a significant capability of the app logic is the provision of control logic which makes decisions on how the actuators should operate, given input from sensors and data of other kinds, either held locally or held centrally.

- Analytics - provides Analytics capability locally rather than in the provider cloud.

- Agent - allows management of the IoT Gateway itself and can also enable management of the attached devices by providing a connection to the provider cloud layer's Device Management.service via the device management protocol.

- Device Data Store - stores data locally. Devices may generate a large amount of data in real time it may need to be stored locally rather than being transmitted to a central location. Data in the device data store can be used by the application logic and analytics capability in the IoT Gateway.

Public Network - contains the wide area networks (typically the internet), peer cloud

systems, the edge services.

Peer Cloud - a 3rd party cloud system that provides services to bring data and capabilities to the IoT platform. Peer clouds for IoT may contribute to the data in the IoT system and may also provide some of the capabilities defined in this IoT architecture. For example an IoT for Insurance solution may use services from partners, such as weather data.

Edge Services - services needed to allow data to flow safely from the internet into the provider cloud and into the enterprise. Edge services also support end user applications. Edge services include:

Domain Name System Server - resolves the URL for a particular web resource to the TCP-IP address of the system or service that can deliver that resource.
Content Delivery Networks (CDN) - support end user applications by providing geographically distributed systems of servers deployed to minimize the response time for serving resources to geographically distributed users, ensuring that content is highly available and provided to users with minimum latency. Which servers are engaged will depend on server proximity to the user, and where the content is stored or cached.
Firewall - controls communication access to or from a system permitting only traffic meeting a set of policies to proceed and blocking any traffic that does not meet the policies. Firewalls can be implemented as separate dedicated hardware, or as a component in other networking hardware such as a load-balancer or router or as integral software to an operating system.
Load Balancers - provides distribution of network or application traffic across many resources (such as computers, processors, storage, or network links) to maximize throughput, minimize response time, increase capacity and increase reliability of applications. Load balancers can balance loads locally and globally. Load balancers should be highly available without a single point of failure. Load balancers are sometimes integrated as part of the provider cloud analytical system components like stream processing, data integration, and repositories.
Provider Cloud - provides core IoT applications and associated services including storage of device data; analytics; process management for the IoT system; create visualizations of data. Also hosts components for device management including a device registry.

A cloud computing environment provides scalability and elasticity to cope with varying data volume, velocity and related processing requirements. Experimentation and iteration using

different cloud service configurations is a good way to evolve the IoT system, without upfront capital investment.

IoT Transformation and Connectivity - enables secure connectivity to and from IoT devices. This component must be able to handle and perhaps transform high volumes of messages and quickly route them to the right components in the IoT solution. The Transformation and Connectivity component includes the following capabilities:

- Secure Connectivity - provides the secured connectivity which authenticates and authorizes access to the provider cloud.

- Scalable Messaging - provides messaging from and to IoT devices. Scalability of the messaging component is essential to support high data volume applications and applications with highly variable data rates, like weather.

- Scalable Transformation - provides transformation of device IoT data before it gets to provider cloud layer, to provide a form more suitable for processing and analysis. This may include decoding messages that are encrypted, translating a compressed formatted message, and/or normalizing messages from varying devices.

- **Application Logic** - The core application components, typically coordinating the handling of IoT device data, the execution of other services and supporting end user applications. An Event based programming model with trigger, action and rules is often a good way to write IoT application logic. Application logic can include workflow. Application logic may also include control logic, which determines how to use actuators to affect physical entities, for those IoT systems that have actuators.

Visualization - enables users to explore and interact with data from the data repositories, actionable insight applications, or enterprise applications. Visualization capabilities include End user UI, Admin UI & dashboard as sub components.

- End User UI - allows users to communicate and interact with Enterprise applications, analytics results, etc. This also includes internal or customer facing mobile user interfaces.

- Admin UI - enables administrators to access metrics, operation data, and various logs.

- Dashboard - allows users to view various reports. Admin UI and Dashboard are internal facing user interfaces.

Analytics - Analytics is the discovery and communication of meaningful patterns of

information found in IoT data, to describe, to predict, and to improve business performance.

Process Management - activities of planning, developing, deploying and monitoring the performance of a business process. For IoT systems, real-time process management may provide significant benefits.

Device Data Store - stores data from the IoT devices so that the data can be integrated with processes and applications that are part of the IoT System. Devices may generate a large amount of data in real time calling for the Device Data Store to be elastic and scalable.

API Management - publishes catalogues and updates APIs in a wide variety of deployment environments. This enables developers and end users to rapidly assemble solutions through discovery and reuse of existing data, analytics and services.

Device Management - provides an efficient way to manage and connect devices securely and reliably to the cloud platform. Device management contains device provisioning, remote administration, software updating, remote control of devices, monitoring devices. Device management may communicate with management agents on devices using management protocols as well as communicate with management systems for the IoT solutions.

Device Registry - stores information about devices that the IoT system may read, communicate with, control, provision or manage. Devices may need to be registered before they can connect to and or be managed by the IoT system. IoT deployments may have a large number of devices therefore scalability of the registry is important.

Device Identity Service - ensures that devices are securely identified before being granted access to the IoT systems and applications. In the IoT systems, device identification can help address threats that arise from fake servers or fake devices.

Transformation and Connectivity - enables secure connections to enterprise systems and the ability to filter, aggregate, or modify data or its format as it moves between cloud and IoT systems components and enterprise systems (typically systems of record). Within the IoT reference architecture the transformation and connectivity component sits between the cloud provider and enterprise network. However, in a hybrid cloud model these lines might become blurred. The Transformation and Connectivity component includes the following capabilities:

- Enterprise Secure Connectivity - integrates with enterprise data security systems to authenticate and authorize access to enterprise systems.

- Transformation - transforms data going to and from enterprise systems.
- Enterprise Data Connectivity - enables provider cloud components to connect securely to enterprise data. Examples include VPN and gateway tunnels.

Enterprise Network - host a number of business specific enterprise applications that deliver critical business solutions along with supporting elements including enterprise data. Typically, enterprise applications have sources of data that are extracted and integrated with services provided by the cloud provider. Analysis is performed in the cloud computing environment, with output consumed by the enterprise applications.

Enterprise Data - includes metadata about the data as well as systems of record for enterprise applications. Enterprise data may flow directly to data integration or the data repositories providing a feedback loop in the analytical system for IoT. IoT systems may store raw, analyzed, or processed data in appropriate Enterprise Data elements. Enterprise **Data includes:** Enterprise User Directory - stores user information to support authentication, authorization, or profile data. The security services and edge services use this to control access to the enterprise network, enterprise services, or enterprise specific cloud provider services.

Enterprise Applications - Enterprise applications consume cloud provider data and analytics to produce results that address business goals and objectives. Enterprise applications can be updated from enterprise data or from IoT applications or they can provide input and content for enterprise data

Security and Privacy - Security and Privacy in IoT deployments must address both information technology (IT) security as well as operations technology (OT) security elements. Furthermore, the level of attention to security and the topic areas to address varies depending upon the application environment, business pattern, and risk assessment. A risk assessment will take into account multiple threats and attacks along with an estimate of the potential costs associated with such attacks. In addition to security considerations, the connecting of IT systems with physical systems also brings with it the need to consider the impact to safety that the IoT system may have. IoT systems must be designed, deployed, and managed such that they can always bring the system to a safe operating state, even when disconnected from communications with other systems that are part of the deployment.

Identity and Access Management- As with any computing system, there must be strong identification of all participating entities – users, systems, applications, and, in the case of IoT, devices and the IoT gateways through which those devices communicate with the rest of the system. Device identity and management necessarily involves multiple entities, starting with chip and device manufacturers, including IoT platform providers, and also including enterprise users and operators of the devices. In IoT solutions it is often the case that multiple of these entities will continue to communicate and address the IoT devices throughout their operational lifetime.

Data Protection -Data in the device, in flight throughout the public network, provider cloud, and enterprise network, as well as at rest in a variety of locations and formats must be protected from inappropriate access and use. Multiple methods can be utilized, and indeed, in many cases, multiple methods are applied simultaneously to provide different levels of protection of data against different types of threats or isolation from different entities supporting the system.

4.8 Logging on to Cloud, Selecting and Creating Cloud Service

AWS IoT

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 consists of the following components:

Device gateway -Enables devices to securely and efficiently communicate with AWS IoT.
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.
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.

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.

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.

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.

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

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.

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.

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 (for example: JWT verification, OAuth provider call out, and so on) and must return policy documents which are used by the device gateway to authorize MQTT operations.

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.

4.9 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.
- **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.
- **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. **Amazon DynamoDB**—Provides managed NoSQL databases.
- **Amazon Kinesis**—Enables real-time processing of streaming data at a massive scale. **AWS Lambda**—Runs your code on virtual servers from Amazon EC2 in response to events.
- **Amazon Simple Notification Service**—Sends or receives notifications.
- **Amazon Simple Queue Service**—Stores data in a queue to be retrieved by applications.

Working of AWS IoT

- AWS IoT enables Internet-connected devices to connect to the AWS Cloud and lets applications in the cloud interact with Internet-connected devices. Common IoT applications either collect and process telemetry from devices or enable users to control a device remotely.
- Devices report their state by publishing messages, in JSON format, on MQTT topics. Each MQTT topic has a hierarchical name that identifies the device whose state is being updated. When a message is published on an MQTT topic, the message is sent to the AWS IoT MQTT message broker, which is responsible for sending all messages published on an MQTT topic to all clients subscribed to that topic.
- Communication between a device and AWS IoT is protected through the use of X.509 certificates. AWS IoT can generate a certificate for you or you can use your own. In either case, the certificate must be registered and activated with AWS IoT, and then copied onto your device. When your device communicates with AWS IoT, it presents the certificate to AWS IoT as a credential.
- recommend that all devices that connect to AWS IoT have an entry in the registry. The registry stores information about a device and the certificates that are used by the device to secure communication with AWS IoT.
- User can create rules that define one or more actions to perform based on the data in a message. For example, you can insert, update, or query a DynamoDB table or invoke a Lambda function. Rules use expressions to filter messages. When a rule matches a message, the rules engine invokes the action using the selected properties. Rules also contain an IAM role that grants AWS IoT permission to the AWS resources used to perform the action.

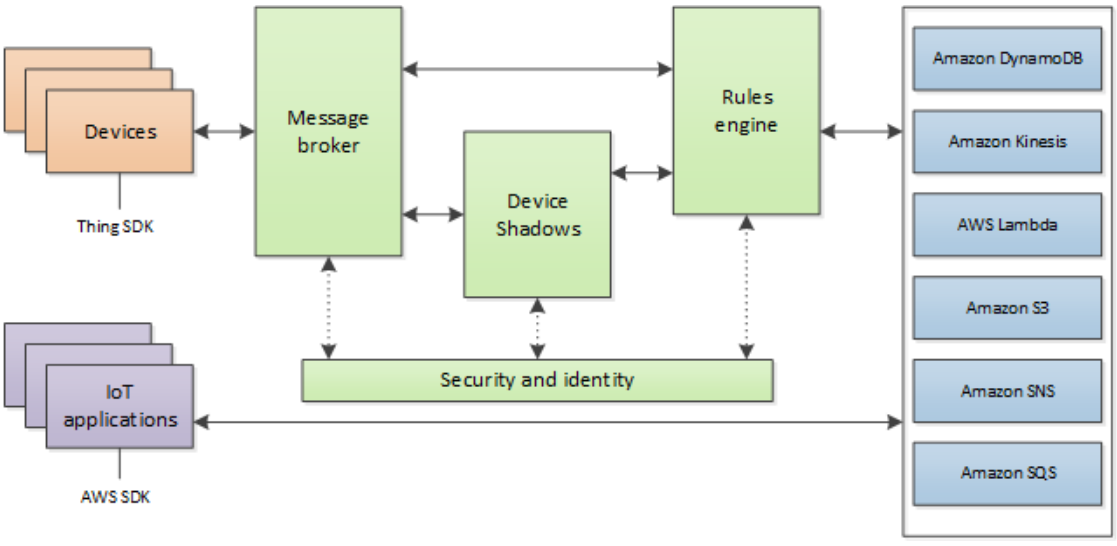


Fig 4.15 AWS IOT Architecture

- Each device has a shadow that stores and retrieves state information. Each item in the state information has two entries: the state last reported by the device and the desired state requested by an application. An application can request the current state information for a device. The shadow responds to the request by providing a JSON document with the state information (both reported and desired), metadata, and a version number. An application can control a device by requesting a change in its state. The shadow accepts the state change request, updates its state information, and sends a message to indicate the state information has been updated. The device receives the message, changes its state, and then reports its new state.

Managing Cloud Account Credentials

If you do not have an AWS account, create one.

To create an AWS account:

1.Open the AWS home page and choose **Create an AWS Account**.

2.Follow the online instructions. Part of the sign-up procedure involves receiving a phone call and entering a PIN using your phone's keypad.

3.Sign in to the AWS Management Console and open the AWS IoT console.

4.On the **Welcome** page, choose **Get started**.

Register a Device in the Registry

Devices connected to AWS IoT are represented by things in the registry. The registry allows you to keep a record of all of the devices that are connected to your AWS IoT account. The fastest way to start using your AWS IoT Button is to download the mobile app for iOS or Android. The mobile app creates the required AWS IoT resources for you, and adds an event source to your button that uses a Lambda blueprint to invoke a new AWS Lambda function of your choice. If you are unable to use the mobile apps, follow these instructions.

1.On the **Welcome to the AWS IoT Console** page, in the left navigation pane, choose **Manage** to expand the choices, and then choose **Things**.

Fig 4.16 Registration

2.On the page that says You don't have any things yet, choose Register a thing.

3.On the **Creating AWS IoT things** page, choose **Create a single thing**.

4.On the **Create a thing** page, in the **Name** field, type a name for your device, such as **MyIoTButton**. Choose **Next** to add your device to the registry.

Create and Activate a Device Certificate

Communication between your device and AWS IoT is protected through the use of X.509 certificates. AWS IoT can generate a certificate for you or you can use your own X.509 certificate. AWS IoT generates the X.509 certificate for you. Certificates must be activated prior to use.

1.Choose **Create certificate**.

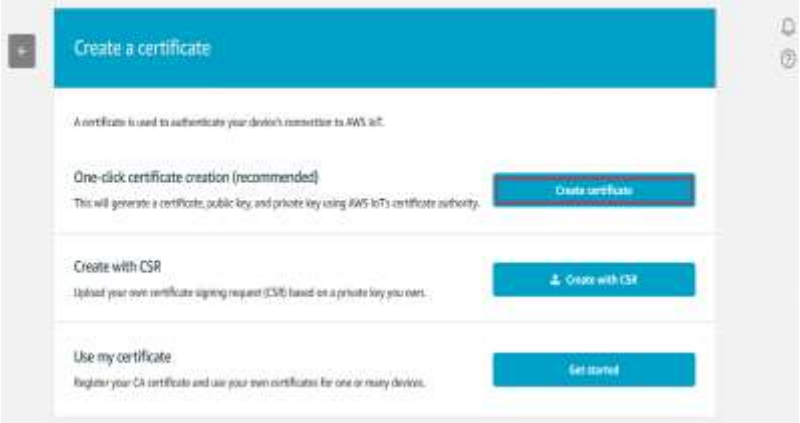


Fig 4.17 Certificate Creation

2.On the **Certificate created!** page, choose **Download** for the certificate, private key, and the root CA for AWS IoT (the public key need not be downloaded). Save each of them to your computer, and then choose **Activate** to continue.Be aware that the downloaded filenames may be different than those listed on the **Certificate created!** page. For example:

- 2a540e2346-certificate.pem.crt.txt
- 2a540e2346-private.pem.key
- 2a540e2346-public.pem.key

Note

Although it is unlikely, root CA certificates are subject to expiration and/or revocation. If this should occur, you must copy new a root CA certificate onto your device.

3.Choose the back arrow until you have returned to the main **AWS IoT** console screen.

Create an AWS IoT Policy

X.509 certificates are used to authenticate your device with AWS IoT. AWS IoT policies are used to authorize your device to perform AWS IoT operations, such as subscribing or publishing to MQTT topics. Your device will presents its certificate when sending messages to AWS IoT. To allow your device to perform AWS IoT operations, you must create an AWS IoT policy and attach it to your device certificate.

1.In the left navigation pane, choose **Secure**, and then **Policies**. On the **You don't have a policy yet** page, choose **Create a policy**.

2.On the **Create a policy** page, in the **Name** field, type a name for the policy (for example,**MyIoTButtonPolicy**). In the **Action** field, type **iot:Connect**. In the **Resource ARN** field, type *. Select the Allow checkbox. This allows all clients to connect to AWS IoT.

You can restrict which clients (devices) are able to connect by specifying a client ARN as the resource. The client ARNs follow this format:

arn:aws:iot:*your-region*:*your-aws-account*:client/<*my-client-id*>

Finally, select the **Allow** check box. This allows your device to publish messages to the specified topic.After you have entered the information for your policy, choose **Create**.

Attach an AWS IoT Policy to a Device Certificate

Now that you have created a policy, you must attach it to your device certificate. Attaching an AWS IoT policy to a certificate gives the device the permissions specified in the policy.

1.In the left navigation pane, choose **Secure**, and then **Certificates**.
2.In the box for the certificate you created, choose **...** to open a drop-down menu, and then choose **Attach policy**.
3.In the **Attach policies to certificate(s)** dialog box, select the check box next to the policy you created in the previous step, and then choose **Attach**.

Attach a Certificate to a Thing

A device must have a certificate, private key and root CA certificate to authenticate with AWS IoT. We recommend that you also attach the device certificate to the thing that represents your device in AWS IoT. This allows you to create AWS IoT policies that grant permissions based on certificates attached to your things. For more information. see Thing Policy Variables

1.In the box for the certificate you created, choose **...** to open a drop-down menu, and then choose **Attach thing**.
2.In the **Attach things to certificate(s)** dialog box, select the check box next to the thing you registered, and then choose **Attach**.
3.To verify the thing is attached, select the box representing the certificate.
4.On the **Details** page for the certificate, in the left navigation pane, choose **Things**.
5.To verify the policy is attached, on the **Details** page for the certificate, in the left navigation pane, choose **Policies**.

Configure Your Device and Button

Configuring your device allows it to connect to your Wi-Fi network. Your device must be connected to your Wi-Fi network to install required files and send messages to AWS IoT. All devices must install a device certificate, private key, and root CA certificate in order to communicate with AWS IoT. The easiest way to configure your AWS IoT button is to use the AWS IoT button smart phone app. You can download it from the Apple App Store or the Google Play Store. If you are unable to use the smart phone app, follow these directions to configure your button.

Turn on your device

1.Remove the AWS IoT button from its packaging, and then press and hold the button until a blue blinking light appears. (This should take no longer than 15 seconds.)

2.The button acts as a Wi-Fi access point, so when your computer searches for Wi-Fi networks, it will find one called **Button ConfigureMe - XXX** where *XXX* is a three-character string generated by the button. Use your computer to connect to the button's Wi-Fi access point.

Configure a Different Device

Consult your device's documentation to connect to it and copy your device certificate, private key, and root CA certificate onto your device. You can use the AWS IoT MQTT client to better understand the MQTT messages sent by a device. Devices publish MQTT messages on topics. You can use the AWS IoT MQTT client to subscribe to these topics to see these messages.

To view MQTT messages:

1.In the AWS IoT console, in the left navigation pane, choose **Test**.

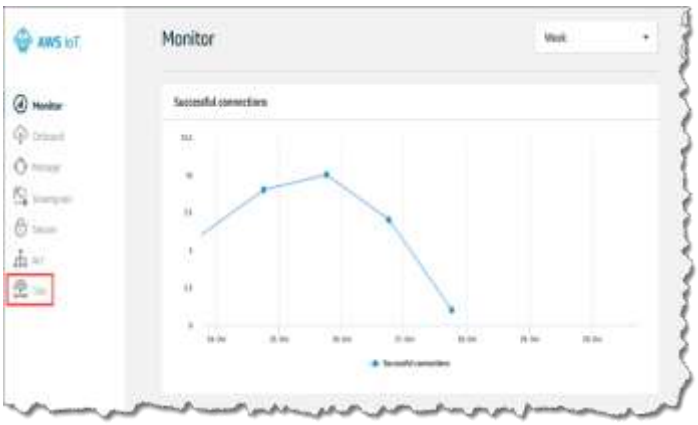


Fig 4.18 MQTT Messages

2.Subscribe to the topic on which your thing publishes. In the case of the AWS IoT button, you can subscribe to **iotbutton/+** (note that + is the wildcard character). In **Subscribe to a topic**, in the **Subscription topic** field, type **iotbutton/+**, and then choose **Subscribe to topic**. Choosing **Subscribe to topic** above, results in the topic **iotbutton/+** appearing in the**Subscriptions** column.

3.Press your AWS IoT button, and then view the resulting message in the AWS IoT MQTT client. If you do not have a button, you will simulate a button press in the next step.

4.To use the AWS IoT console to publish a message:

On the MQTT client page, in the **Publish** section, in the **Specify a topic and a message to publish**… field, type **iotbutton/ABCDEFG12345**. In the message payload section, type the following JSON:

```
{
   "serialNumber": "ABCDEFG12345",
   "clickType": "SINGLE",
   "batteryVoltage": "2000 mV"
}
```

Choose **Publish to topic**. You should see the message in the AWS IoT MQTT client (choose **iotbutton/+** in the **Subscription** column to see the message).

Configure and Test Rules

The AWS IoT rules engine listens for incoming MQTT messages that match a rule. When a matching message is received, the rule takes some action with the data in the MQTT message (for example, writing data to an Amazon S3 bucket, invoking a Lambda function, or sending a message to an Amazon SNS topic). In this step, you will create and configure a rule to send the data received from a device to an Amazon SNS topic. Specifically, you will:

- Create an Amazon SNS topic.
- Subscribe to the Amazon SNS topic using a cell phone number.
- Create a rule that will send a message to the Amazon SNS topic when a message is received from your device.
- Test the rule using your AWS IoT button or an MQTT client.

4.10 Cloud based IoT platforms

1. Thingworx 8 IoT Platform

Thingworx is one of the leading IoT platforms for industrial companies, which provides easy connectivity for devices. It enables the experience from today's connected world. Thingworx 8 is a better, faster, easier platform, providing the functionality to build, deploy, and extend industrial projects and apps.

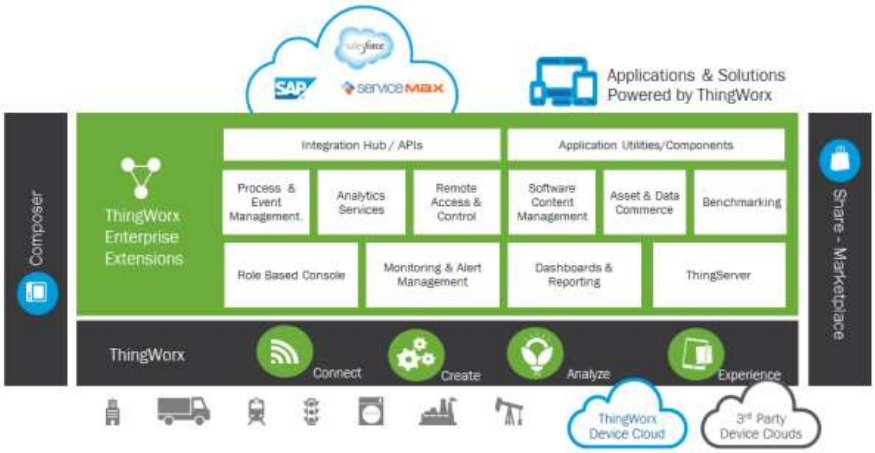


Fig 4.19 Thingworx 8 IoT Platform

Thingworx is an IoT platform designed by PTC for development of enterprise app development. It offers basic features, such as:

- Easy connectivity with electronic devices, like sensors and RFIDs
- You can work remotely once you are done with the setup
- Pre-built widgets for the dashboard
- Remove Complexity of the project
- Integrated machine learning

Pros

- Easy web page designs for customers
- Easy to manage devices
- Simple connectivity solutions

Cons

- Difficult to use with custom programs in C#

- Hard to manage complex systems.
- The limitation to install edge program on a custom platform.

2. Microsoft Azure IoT Suite

Microsoft Azure provides multiple services to create IoT solutions. It enhances your profitability and productivity with pre-built connected solutions. It analyzes untapped data to transform business. This provides the solutions for a small PoC to Rolling out your ideas. Azure Suite can easily analyze and act on new data.

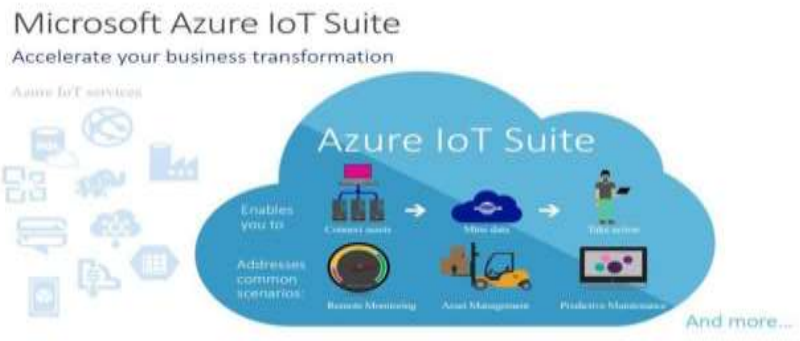


Fig 4.20 Microsoft Azure IoT Suite

Azure IoT Suite provides features like:

- Easy Device Registry.
- Rich Integration with SAP, Salesforce, Oracle, WebSphere, etc
- Dashboards and visualization
- Real-time streaming

Pros

- Offers third-party services
- Secure and scalable
- High availability

Cons

- Requires management
- Expensive
- No support for bugs

3 . Google Cloud’s IoT Platform

Google's platform is among the best platforms we currently have. Google has an end-to-end platform for Internet-of-Things solutions. It allows you to easily connect, store, and manage IoT data. This platform helps you to scale your business.

Their main focus is on making things easy and fast. Pricing on Google Cloud is done on a per-minute basis, which is cheaper than other platforms.

Google Cloud's IoT platform provides features, including:

- Provides huge storage
- Cuts cost for server maintenance
- Business through a fully protected, intelligent, and responsive IoT data
- Efficient and scalable
- Analyze big data

Pros

- Fastest input/output
- Lesser access time
- Provides integration with other Google services

Cons

- Most of the components are Google technologies
- Limited programming language choices

4 IBM Watson IoT Platform

IBM Watson is a powerful platform backed by IBM ’s the Bluemix and hybrid cloud PaaS (platform as a service) development platform. By providing easy sample apps and interfaces for IoT services, they make it accessible to beginners. You can easily try out their sample to see how it works, which makes it stand out from other platforms.

Fig 4.21 IBM Watson IoT Platform

Users can get the following features:

- Real-time data exchange
- Secure Communication
- Cognitive systems
- Recently added data sensor and weather data service

Pros

- Process untapped data
- Handle huge quantities of data
- Improve customer services

Cons

- Need a lot of maintenance.
- Take time for Watson integration
- High switching cost.

5. AWS IoT Platform

Amazon made it much easier for developers to collect data from sensors and Internet-connected devices. They help you collect and send data to the cloud and analyze that information to provide the ability to manage devices.

You can easily interact with your application with the devices even they are offline.

Fig 4.22 AWS IoT Platform

Main features of the AWS IoT platform are:

- Device management
- Secure gateway for devices
- Authentication and encryption
- Device shadow

Pros

- Good integration with laas offering.
- Price dropped over the last six years
- Open and flexible

Cons

- A big learning curve for AWS
- Three outages in the last 2 years
- Not secure for hosting critical enterprise applications

6. Cisco IoT Cloud Connect

Cisco Internet of Things accelerates digital transformation and actions from your data. Cisco IoT Cloud Connect is a mobile, cloud-based suite. It offers solutions for mobile operators to provide phenomenal IoT experience. It provides flexible deployment options for your devices.

Fig 4.23 Cisco IoT Cloud Connect

The main feature of the Cisco Cloud Connect:

- Data and voice connectivity
- Device and IP session report
- Billing is customizable
- Flexible deployment options

7. Salesforce IoT Cloud

Salesforce IoT Cloud is powered by Salesforce Thunder. It gathers data from devices, websites, applications, and partners to trigger actions for real-time responses. Salesforce combined with IoT delivers improved customer service.

Fig 4.24 Salesforce IoT Cloud

Key features of Salesforce IoT Cloud:

- Enhanced data collection

- Improved customer engagement
- Real-time event processing
- Technology optimization

Pros

- Scale to billions of devices and messages.
- Easy UI designs to connect with customers.

Cons

- Security liability
- Flexibility limitations

8. Kaa IoT Platform

Kaa is an open-source, multipurpose, middleware platform for complete end-to-end IoT development and smart devices. It reduces cost, risk, and market time. Also, Kaa offers a range of IoT tools that can be easily plugged in and implemented in IoT use cases.

Fig 4.25 Kaa IoT Platform

It provides many features that make it unique, such as:

- Reduce development time
- Open source and free
- Easy and direct device implementation
- Reduce marketing time
- Handle millions of devices

Pros

- Ease of use
- Third-party integration
- Data security

Cons

- Not able to deploy applications based on the PaaS model

9. Oracle IoT Platform

Oracle offers real-time Internet of Things data analysis, endpoint management, and high speed messaging where the user can get real-time notification directly on their devices. Oracle IoT cloud service is a Platform as a Service (PaaS), cloud-based offering that helps you to make critical business decisions.

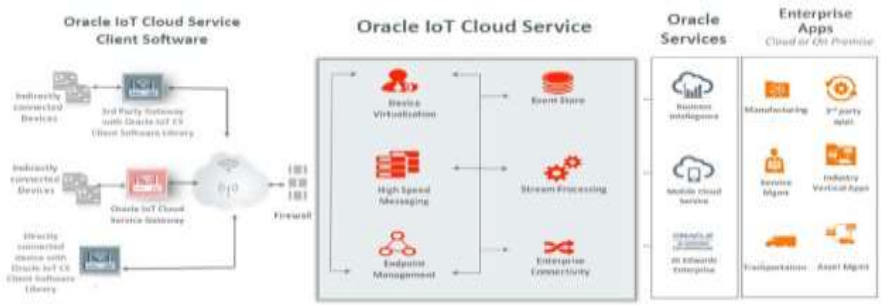


Fig 4.26 Oracle IoT Platform

Features offering to users

- Secure and scalable
- Real-time insight
- Integrated
- Faster to market

Pros

- Device visualization
- High speed messaging
- Event store

10. Thingspeak IoT Platform

Thingspeak is an open-source platform that allows you to collect and store sensor data to the cloud. It provides you the app to analyze and visualize your data in Matlab. Can use Arduino, Raspberry Pi, and Beaglebone to send sensor data. You can create a separate channel to store data.

Fig 4.27 Thingspeak IoT Platform

Features of Thingspeak:

- Collect data in private channels
- App integration
- Event scheduling
- MATLAB analytics and visualization

Pros

- Free hosting for channels
- Easy visualization
- Provides additional features for Ruby, Node.js, and Python

Cons

- Limited data uploading for API
- ThingSpeak API can be a hurdle for beginners

11. GE Predix IoT Platform

Predix is the world's first industrial platform. Predix was designed to target factories and provides simple ecosystem. It can directly analyze data from the machine and store. GE

wants to provide the growing industrial Internet of Things for its cloud platform. This platform is secure and scalable.

Fig 4.28 GE Predix IoT Platform

According to the customers their IoT platform can:

- Optimize assets and operations
- Provides key performance data
- Reduces unplanned downtime
- Real-time operational data

4.11 IBM Watson IoT Platform

IBM Watson is a powerful platform backed by IBM's the Bluemix and hybrid cloud PaaS (platform as a service) development platform. By providing easy sample apps and interfaces for IoT services, they make it accessible to beginners. You can easily try out their sample to see how it works, which makes it stand out from other platforms.

It is a fully managed, cloud-hosted service designed to make it simple to derive value from Internet of Things devices. It provides capabilities such as device registration, connectivity, control, rapid visualization and storage of Internet of Things data.

Connect and Manage devices: Connect to the Watson IoT Platform on IBM Cloud with ease, then set up and manage IoT devices and data to start creating your own applications, visualization dashboards and mobile IoT apps.

Build quickly and securely: Provides the tools and services need to create IoT applications including cognitive APIs, Weather Company data, blockchain and more.

Extend with Watson Cognitive APIs: Create a better experience with a natural voice interface or image recognition.

Figure below is a deployment diagram that is typical of IBM Watson IoT Platform - Message Gateway applications. In the diagram, IBM Watson IoT Platform - Message Gateway connects many users and devices on the internet to services that are deployed on an intranet. The users, devices, and services interact with each other by exchanging messages through IBM Watson IoT Platform - Message Gateway.

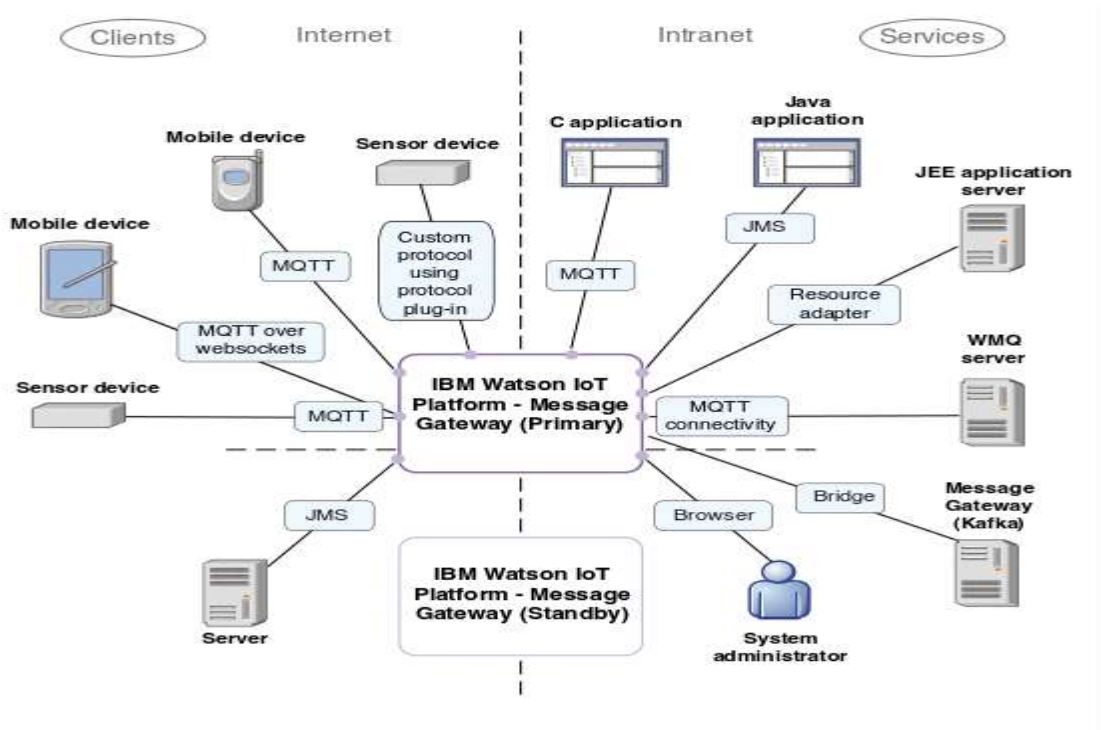


Fig 4.29 IBM Watson Platform

IBM Watson IoT Platform - Message Gateway connects large numbers of clients and routes large volumes of messages.

IBM Watson IoT Platform - Message Gateway supports both publish/subscribe and point-to-point messaging.

Like IBM MQ, it is a subject-based publish subscribe broker, not a content-based broker.

Unlike a content-based broker, message-content is not queried or altered by IBM Watson IoT Platform - Message Gateway.

No user-based applications can run on IBM Watson IoT Platform - Message Gateway.

IBM Watson IoT Platform - Message Gateway typically sits alongside other edge-of-network devices with the same objectives of forwarding large volumes of internet traffic from many different clients.

Connectivity

Connect clients to IBM Watson IoT Platform - Message Gateway by using MQTT, or JMS protocols, which are natively supported

Connect clients to IBM Watson IoT Platform - Message Gateway over a custom protocol by using the protocol plug-in.

Connect IBM Watson IoT Platform - Message Gateway to a IBM MQ network by using MQ Connectivity.

Connect IBM Watson IoT Platform - Message Gateway to a Java™ Platform, Enterprise Edition application server by using IBM Watson IoT Platform - Message Gateway resource adapter. Connecting in this way allows application server-based JMS messaging, such as asynchronous message driven beans, to be used with IBM Watson IoT Platform - Message Gateway.

Connect standby IBM Watson IoT Platform - Message Gateway server to your primary IBM Watson IoT Platform - Message Gateway server to set up a high availability (HA) pair.

Users can get the following features:

- Real-time data exchange
- Secure Communication
- Cognitive systems
- Recently added data sensor and weather data service

Pros

- Process untapped data
- Handle huge quantities of data
- Improve customer services

Cons

- Need a lot of maintenance.
- Take time for Watson integration
- High switching cost.

4.12 Google Cloud’s IoT Platform

Google's platform is among the best platforms we currently have. Google has an end-to-end platform for Internet-of-Things solutions. It allows you to easily connect, store, and manage IoT data. This platform helps to scale business.

Google Cloud Platform is a set of Computing, Networking, Storage, Big Data, and Machine Learning and Management services provided by Google that runs on the same Cloud infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, Google Photos and YouTube.

Their main focus is on making things easy and fast. Pricing on Google Cloud is done on a per-minute basis, which is cheaper than other platforms.

Google Cloud's IoT platform provides features, including:

- Provides huge storage
- Cuts cost for server maintenance
- Business through a fully protected, intelligent, and responsive IoT data
- Efficient and scalable
- Analyze big data
- IoT is helping businesses gain new revenue and new intelligence
- ***Tools available for all IoT applications***

- From ingestion to intelligence, take advantage of Google Cloud’s wide range of IoT building blocks to derive value from our device data.

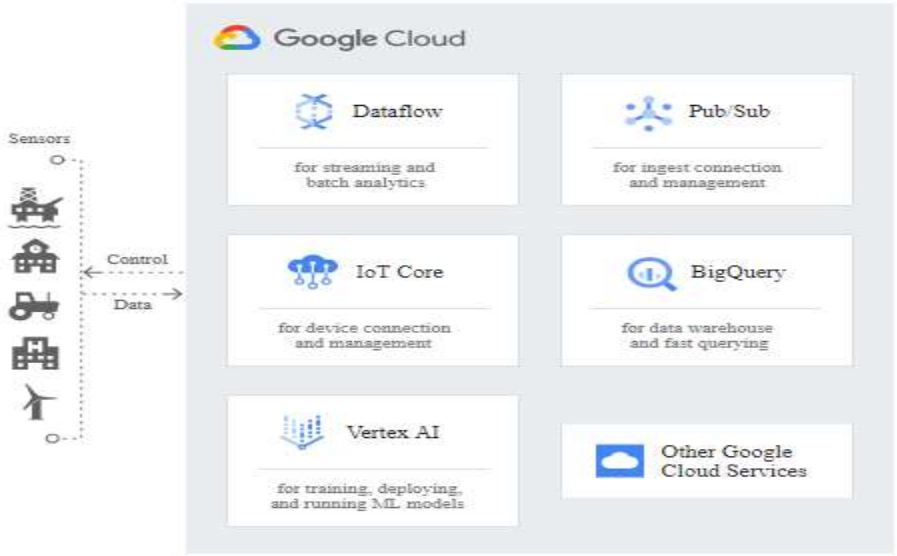


Fig 4.30 Google Cloud

Predictive maintenance

Google Cloud's IoT platform lets you automatically predict when equipment needs maintenance and optimize its performance in real time while predicting downtime, detecting anomalies, and tracking device status, state, and location.

Real-time asset tracking

Track valuable assets in real time and perform complex analytics and machine learning on the data you collect in order to deliver actionable business insights.

Logistics and supply chain management

Perform fleet management, inventory tracking, cargo integrity monitoring, and other business-critical functions with Google Cloud IoT's logistics solution.

Smart cities and buildings

Bring new levels of intelligence and automation to entire homes, buildings, or cities by building a comprehensive solution that spans billions of sensors and edge devices.

Google offers a wide range of Services. Following are the major Google Cloud Services:

- **Compute**
- **Networking**

- **Storage and Databases**
- **Big Data**
- **Machine Learning**
- **Identity & Security**
- **Management and Developer Tools**

Fig 4.31 Google Cloud Services

- Google Compute Engine
- Google App Engine
- Google Kubernetes Engine
- Google Cloud Container Registry
- Cloud Functions

Networking: The Storage domain includes services related to **networking**, it includes the following services

- Google Virtual Private Cloud (VPC)
- Google Cloud Load Balancing
- Content Delivery Network
- Google Cloud Interconnect
- Google Cloud DNS

Storage and Databases: The Storage domain includes services related to data **storage**, it includes the following services

- Google Cloud Storage

- Cloud SQL
- Cloud Bigtable
- Google Cloud Datastore
- Persistent Disk

Big Data: The Storage domain includes services related to **big data**, it includes the following services

- Google BigQuery
- Google Cloud Dataproc
- Google Cloud Datalab
- Google Cloud Pub/Sub

Cloud AI: The Storage domain includes services related to **machine learning,** it includes the following services

- Cloud Machine Learning
- Vision API
- Speech API
- Natural Language API
- Translation API
- Jobs API

Identity & Security: The Storage domain includes services related to **security,** it includes the following services

- Cloud Resource Manager
- Cloud IAM
- Cloud Security Scanner
- Cloud Platform Security

Management Tools: The Storage domain includes services related to **monitoring and management**, it includes the following services

- Stackdriver
- Monitoring
- Logging

- Error Reporting
- Trace
- Cloud Console

Developer Tools: The Storage domain includes services related to **development**, it includes the following services

-
  - Cloud SDK
  - Deployment Manager
  - Cloud Source Repositories
  - Cloud Test Lab

Pros

- Fastest input/output
- Lesser access time
- Provides integration with other Google services

Cons

- Most of the components are Google technologies
- Limited programming language choices

TEXT / REFERENCE BOOKS

1. Boswarthick,Omar Elloumi., The Internet of Things: Applications and Protocols, Wiley publications., 2012
2. Dieter Uckelmann, Mark Harrison, Florian Michahelles., Architecting the Internet of Things, Springer publications.2011
3. Marco Schwatrz Internet of Things with Arduino Cookbook, Packt Publications.2016 .
4. Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, “From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence”, 1st Edition, Academic Press, 2014.
5. Vijay Madisetti, Arshdeep Bahga, “Internet of Things: A Hands-On Approach” published by Vijay Madisetti 2014

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – 5 – Introduction to IoT– SCSA1308

Unit 5

DOMAIN SPECIFIC APPLICATIONS OF IoT

Home automation, Industry applications, Surveillance applications, Other IoT applications - Introduction to different IoT tools, Developing applications through IoT tools, Developing sensor based application through embedded system platform – case studies: Soil moisture monitoring, Weather monitoring, Air quality Monitoring, Movement Detection.

5.1 Home Automation an IoT software-based approach on the field of Home Automation. Common use-cases include measuring home conditions, controlling home appliances and controlling home access through RFID cards as an example and windows through servo locks. However, the main focus of this paper is to maximize the security of homes through IoT. More specifically, monitoring and controlling servo door locks, door sensors, surveillance cameras, surveillance car and smoke detectors, which help ensuring and maximizing safety and security of homes. A user has the following features through a mobile application in which he/she: 1. can turn on or o_ LED lights and monitor the state of the LED. 2. can lock and unlock doors through servo motors and monitor if the doors are locked or unlocked. 3. can monitor if the doors are closed or opened through IR sensors. 4. is notified through email if the door is left open for too long. 5. is notified of who entered through the door as the camera captures the face image and send it to him/her via email. 6. is notified through email if the _re detector detect ssmoke. 7. is able to control the surveillance car from anywhere to monitor his/herhome. As the field of Home Automation through IoT is a wide application in a very wide and challenging field due to the reasons mentioned in the previous paragraphs, I chose to work on that field as part of this thesis, specifically in maintaining and ensuring security and safety inside home.

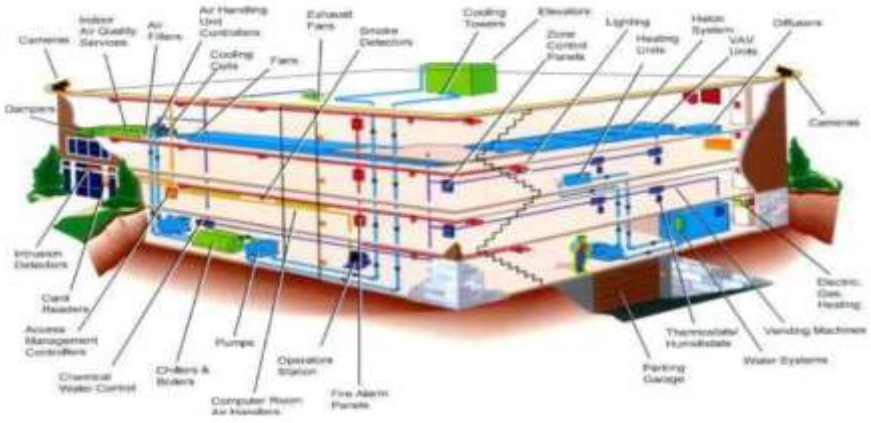


Fig 5.1. Home automation

IoT aims in creating a network between objects embedded with sensors, that can store, analyze, communicate and exchange data together over the internet. This leads to efficient industry,

manufacturing, efficient energy management, resource management, accurate health care, smarter business decisions based on analyzed data, safer driving through smart cars that are able to communicate together, smart home automation and countless more applications. The system designed for the home automation project presented in this paper needs a control unit, a computer, to be able to control the different electrical devices connected to it. Raspberry Pi, is a credit-card tiny computer, that can be plugged to a monitor, uses standard keyboard and mouse, that enables people of different ages learn how to program.

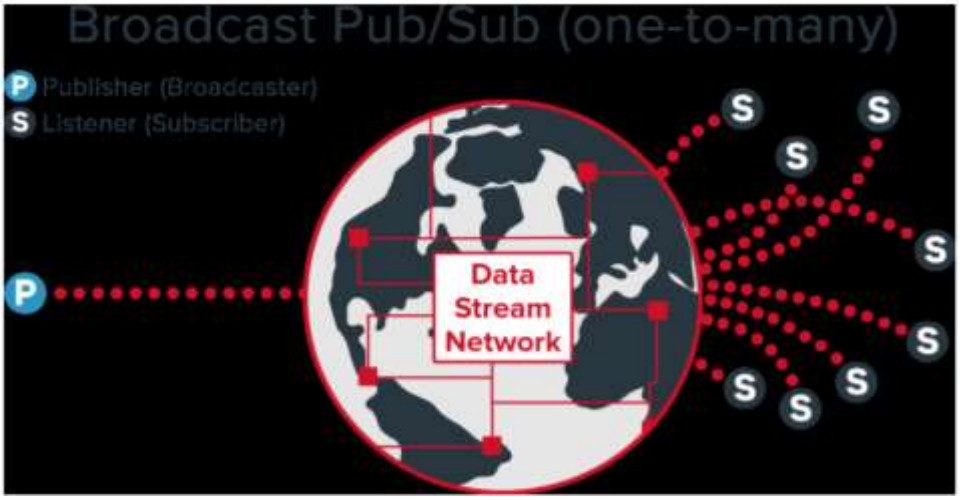


Fig 5.2. publish/subscribe model

Illustrates the publish/subscribe model provided by PubNub

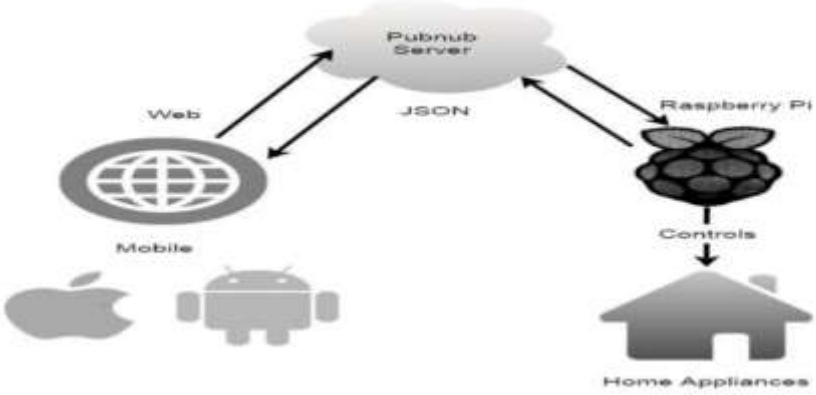


Fig 5.3 Illustrates the system architecture used in this home automation project

To simplify the publish/subscribe model along with the system architecture used in this Home Automation project, here is the explanation of the steps of constructing it: Different sensors, cameras and servo motors were connected to the Raspberry Pi. It was programmed to collect and publish the data, in the form of JSON string, acquired from these devices to PubNub. Data is published from the Raspberry Pi by providing it with the "publish key" and the "channel name". The data is sent to the channel provided by PubNub servers, and forwarded by PubNub to the subscribers of this channel. The subscriber in this

scenario, of a user acquiring data and readings by the sensors and monitoring devices, is the web/mobile application. The "subscription key" and "channel name" is embedded in the web/mobile application's code. Allowing it to receive messages forwarded by PubNub. On the other hand, in a scenario where the user wants to send a command to home appliances, controlling the LED lights for example, the web/mobile application is the publisher provided by the "publish key" and the "channel name". The commend is sent in the form of JSON string to PubNub servers, while the "subscription key" and "channel name" is embedded in the Raspberry Pi code. This allows the Raspberry Pi to receive any published strings on the channel it is subscribed to. Upon receiving the JSON string, the Raspberry Pi take the action specified by that string. This allows full control and monitoring of all devices connected to the Raspberry Pi by the user.

5.2 Applications of Industrial Internet of Things

1. **Industrial Automation**

Fig 5.4 Industrial automation

Industrial automation is one of most significant and common application of Internet of Things. Automation of machines and tools enables companies to operate in an efficient way with sophisticated software tools to monitor and make improvements for next process iterations.

Accuracy of process stages can be improved to a greater level using machine automation. Automation tools like PLC (Programmable Logic Control) and PAC (Programmable Automation Control) are used with smart sensor networks connected to a central cloud system which collect huge amount of data. Specially designed software and applications are used to analyze the data and its behavior for improvements.

Industrial automation improves accuracy, efficiency; reduces errors, easy to control and remotely accessible via applications. Machines can operate at harsh environments than humans; automation of machines and tools reduces man power requirements for specific tasks.

Connected Factories

Connected Factory concept is an effective solution for improvements in all areas of operation. Major components such as machines, tools and sensors will be connected to a network for easier management and access. Overview of process flow, monitor down time, status checking of inventory, shipment, schedule maintenance and stop/pause a particular process for further analysis etc… can be done remotely using industrial IoT solutions.

2. Smart Robotics

Many companies are developing intelligent robotics system for IoT-enabled factories. Smart robotics ensures smooth handling of tools and materials in the manufacturing line with precise accuracy and efficiency. Predefined specifications can be set for maximum precision (up to few nanometers scale for some applications) using intelligent robotic arms. Man machine interface design concept will reduce the complexity of operation and it will reflect in future IoT enabled manufacturing as improved productivity.

3. Predictive Maintenance

Modern industrial machines equipped with smart sensors continuously monitoring the status of each major components and it can detect any critical issues before the system is completely down. Smart sensors will trigger maintenance warning to the centralized system and the alert messages will be delivered to responsible persons/groups. Maintenance engineers can analyze the data and plan for schedules maintenance effectively without affecting routine task. Predictive maintenance is an effective solution to avoid unnecessary downtime in the production line. Unexpected failure of machines could cause damage to products, delay in delivery and business loss for manufacturers.

Status of each machines are stored to a cloud system in a real-time basis. History of each machines, performance, and next scheduled maintenance are easily accessible remotely (on PCs, via web interface or via smartphone applications). Performance improvements can be calculated and implemented for each machines and process stages of products using collected data analysis.

4. Integration of Smart Tools / Wearables

Integration of smart sensors to tools and machines enables the workforce to perform the task with improved accuracy and efficiency. Specially designed wearables and smart glass helps employees to reduce error and improve safety at the working environments.

Smart wearables can trigger instant warning messages to employees during emergency situations like gas leak or fire. Wearables can monitor health condition of individuals continuously and feedback if not fit for particular task.

5. Smart Logistics Management

Logistics is one of the important areas in many industries, which needs continuous improvements to support increasing demands. Smart sensor technology is a perfect fit to solve many of the complex logistics operations and manage goods efficiently.

Retail giants like Amazon using drones to deliver goods to their customers. Advanced technologies like drones offer better efficiency; accessibility, speed and it require less manpower. However, initials investments are huge compared to conventional methods and implementation has limitations.

Airline is another major industry, which uses IoT for its daily operations at the production and predictive maintenance of airplanes in service. At the manufacturing plant, airline companies use IoT solutions to track thousands of components required for every single day at work. Centralised management of inventories helps to manage its supplies effortlessly.

Suppliers will be automatically informed if any items are required to top up. Without much human action, inventory management can be effectively implemented using IoT.

Smart sensors continuously monitor airplane's machineries, the data is collected real-time and send to the airplane manufacturer. Maintenance of any part of an airplane will be triggered, concerned team will be informed and maintenance will be carried out once the plane is landed without any delay. Manufacturers can plan and deliver spare parts efficiently based on the data shared by the system.

6. Software integration for product optimization

Smart analytics solution is one of most important component of any IoT system which further enhances the possibilities of the system for improvement and optimization.

Major companies are implementing customized software for deep analysis of huge amount of data collected from large sensor networks and machines. Detailed analysis of data and understanding the behavior over time gives much better overview of process improvement strategies for product optimization.

Improvement ideas could be directly related to product recipe or optimization of particular machinery for better performance and output. Cost effective solutions can be achieved using analysis of data and its behavior patterns over a period of time. Analysis of huge amount of data was a hard, inaccurate and time consuming task before introduction of these software tools.

7. Smart Package Management

Package management using IoT technology gives lot of convenience and efficiency for manufacturing units. Smart sensors can monitor each stages of packing and update status in real-time manner. Embedded sensors can detect vibrations, atmospheric conditions like temperature and humidity etc…

and feedback if something goes wrong during transit or storage.

8. Enhanced Quality and Security

Introduction of IoT technology in to manufacturing offers enhanced product quality. Continuous monitoring and analysis of each stages ensure better quality by improving process steps for optimum quality.

Integration of smart tools and software assisted procedures offer higher level of security. Software controlled automation and data collection from huge sensor network is connected to a highly secure gateway and cloud server platform.

Complex encryption techniques are used in IIoT platform for enhanced security.

9. Autonomous vehicles

Automotive industries are using IoT enables self driving vehicles to supply goods and logistics management within their company premises. Smart vehicles can detect traffic congestions along its path and make deviation to reach its destination is shortest time. These vehicles are equipped with many smart sensors continuously detect location data using GPS and wireless technologies for communication with the control station.

10. Power Management

IoT can offer better solutions for power management in industries. Specific sensors can detect environment and trigger to turn on/off control of lights, air conditioners, humidity controls, liquid flow etc… for efficient power management.

Advantages of Industrial Internet of Things

Improved accuracy

Product and process optimization

Predictive maintenance and analysis

Higher efficiency

Remote accessibility and monitoring

Enhanced security

Scalability of network

Reduced down time for machines and process

Power savings

Cost effectiveness

5.3 Surveillance Applications

In the present world, situation security assumes a vital part. Numerous individuals utilize distinctive sorts of security system to keep their property from unapproved person's entry. Security system helps individuals to feel somewhat safe while they have to travel or avoid their home for work. A large number of the security system works just inside a specific territory limit or instance, CCTV, as a person need to see camera footage from control room. The current security systems against robbery are entirely costly as a certain measure of cash must be paid to administration supplier to store the recorded video despite the fact that there is no human movement is recognized. The solution for this problem is an intelligent surveillance system that can start recording video only after a human motion is detected. This eventually minimizes the required storage space and makes system cost-effective. The framework gives more security with the assistance of Web at less expensive cost and requires less storage space. The framework gives a smart security system which gives home security with SMS and e-mail notice about the unapproved people nearness, programmed human checking and switching off all the appliances which consumes more power by customizing coding with particular appliances. System performs various tasks such as motion detection, human detection and counting, alarm activation, SMS notification through GSM and Internet Twilio account, and e-mail notification. To improve the system performance, two boards are used—Raspberry Pi and Arduino. Raspberry Pi works in surveillance mode and Arduino works in normal mode. Arduino verifies the password and allows Raspberry Pi to start the surveillance mode. Once the password is verified, Arduino turns off all the electrical appliances by customizing coding with specific appliances. Raspberry Pi performs various tasks in surveillance mode such as motion and human detection, human counting, sending SMS, and e-mail notification to user after human detection. After human detection, Raspberry Pi sends command to Arduino for sending SMS to user by communicating with GSM module. By default, system remains in normal mode. As the user enters correct password, system starts working in surveillance mode. In surveillance mode, Raspberry Pi detects human motion and counts number of people in a room. The location of a camera is at the entrance of a room. The human count is implemented by background subtraction method in OpenCV. If any human is detected in surveillance mode, then using the GSM module and Twilio account message is sent to the owner of the house. The highlights of system are as follows:

(1) The framework includes people counting, and two notices are sent to client by SMS: One SMS is sent through GSM and one SMS is sent through Twilio trial account with the assistance of Web. The recorded video is sent as a e-mail to client. At the point when there is no individual in the premises, the framework works in ordinary mode.

(2) Raspberry Pi detects motion and human presence and it counts number of humans in a room. As the system detects human presence, immediately a SMS notification is sent to the user. The system also

sends the recorded video to users mail id. As a human is detected, GSM module gets instruction from Arduino regarding SMS notification. Another SMS notification is sent through Internet Twilio trial account. The alarm is turned on as human presence is detected.

(3) The proposed system also provides a facility to control electrical appliances by turning them off. The proposed system offers few advantages such as- (i) Less memory storage space is used for recording video as system start recording the video only after motion is detected. (ii) Recorded video is e-mail to user so that the user can inspect it later. (iii) User gets noticed (SMS and Email) just after human detection, so that he can take necessary actions immediately.

Working Principle The proposed framework is initiated by entering right password. The movement recognition algorithm is actualized to distinguish the moving items and human count in room is done by utilizing OpenCV. After the password verification system starts working in surveillance mode and all the electrical appliances are turned off appliances by customizing coding with specific appliances. If motion is detected, the system checks for human detection. As system detects human presence, a notification through SMS is sent and alarm is turned on. The activity of that human is recorded and e-mailed to user. If the video consists of less than or equal to 100 moving frames, the video is immediately sent to user, and if the video exceeds 100 moving frames, then the video of those moving frames will be sent in the next e-mail. Motion is detected by background subtraction method MOG2 algorithm. In the event if the movement is identified, then human discovery is executed by HAAR cascade classifier. The proposed framework is sufficiently keen to identify human movement, checks number of individuals, and informs client by sending SMS, and e-mails the recorded video to client. Figure 1 demonstrates the block diagram of proposed framework. Step by step working process is as follows: (1) Start process by entering correct password. System goes in surveillance mode as Arduino allows Raspberry pi to turn on the camera and all electrical appliances are turned off. (2) Wait for motion detection—Confirm the human detection, people counting mode and send SMS to the owner, Alarm is turned ON.

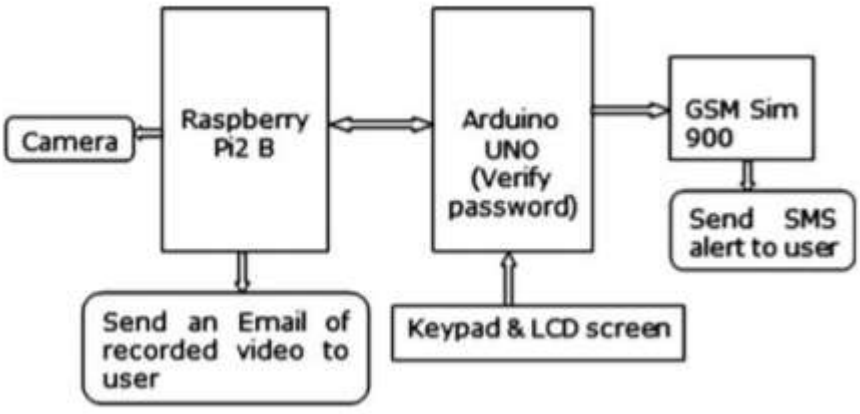


Fig 5.5 Surveillance applications

(3) Security mode is ON—Record video as security system is broken and e-mail that recorded video to user. (4) Enter password again to make system work in normal mode.

System Architecture

Elements of the System

Raspberry Pi2 is the primary handling unit. OpenCV is build and introduced on it for image processing. Arduino is subprocessing unit which is in charge for initialization of fundamental handling unit after getting password from client. Python is utilized for interfacing between Raspberry Pi2 and Arduino, and Python is additionally utilized for sending SMS through Web; furthermore, it is utilized to send e-mail to the client/user.

Hardware and Software Design

In addition to Raspberry Pi, Arduino and GSM module are the principle equipments utilized for the framework. The GSM module (needs a SIM card to work) is associated with Arduino by USB to serial converter. A LCD screen is utilized to show the entered secret word. Two relays of 12 volts are utilized which are associated between raspberry pi and Arduino. One relay is in charge of activation and deactivation of fundamental procedure, i.e., Raspberry Pi. Other relay gives a control to turn off/on electrical appliances. Figure 2 demonstrates the algorithm utilized for the framework.

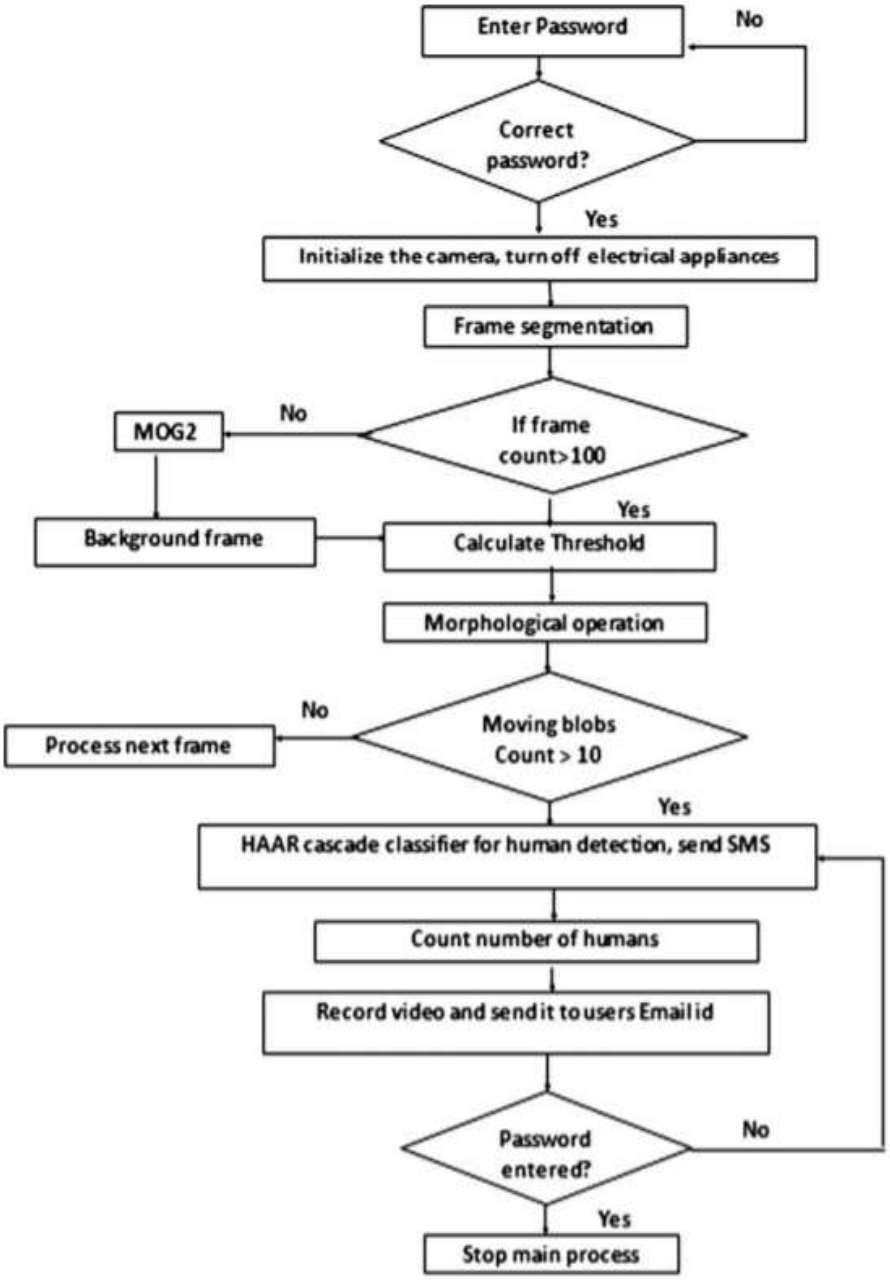


Fig 5.6 Flow chart-Surveillance applications

5.4 IoT DEVELOPMENT TOOLS

IoT development tools are **created for tailing IoT applications across various networks** and managing diverse updates to test how app changes can affect hardware responses.

Microsoft Azure IoT kits

Microsoft Azure created a team by coordinating with Adafruit for building six IoT kits that come Azure Certified for the need of IoT developers having single-board PCs, actuators, and sensors. Generally, developers can make use of the WiFi boards, SD cards, sensors, and colored LEDs inside the kits. Some of the IoT kits from Azure are intended for the need of top IoT product development by experts. Those who are beginners or have intermediate knowledge can try the Adafruit Raspberry Pi Kit, Adafruit

Feather M0 Kit, and SparkFun Thing Dev Kit.

Arduino (IDE)

Adruino is a top IT company based in Italy famous for building microcontroller boards, and interactive kits and objects that are reputed as the most preferred IDEs among other IoT development tools. Arduino crafted a full-blown, optimized, and mature platform for interconnecting diverse hardware systems. Arduino provides a full IoT package that is enriched with many top examples and libraries that supports the industry-grade IoT app development projects.

Arduino offers IoT packages enriched with library support for top industry-grade IoT app development projects. Arduino is easy-to-use to implement strategies that any beginner can adopt and start with it.

Raspbian

Raspbian IoT IDE was built for the Raspberry Pi board offered by IoT tech specialists. With more than 35,000 packages and various examples of rapid installation that come with the use of pre-compiled software make it an important IoT development tool. Maybe Raspbian's greatest quality is that it's under constant development and has widened reach for computing so users receive maximum benefits.

DeviceHive

DeviceHive is an open-source machine to machine communication framework that was launched in 2012. DeviceHive is considered as one of the most preferred IoT application development platforms because it has a cloud-based API that anybody can control remotely and independently of network configuration.

The same applies to its management portal, protocols, and libraries. DeviceHive works best with applications that address security, sensors, automation, and smart home technology. As a bonus, DeviceHive's website includes support and references from its community and online blog resources.

OpenSCADA

OpenSCADA is a tool that's part of the SCADA project represented by Eclipse IoT industry groups. It is well-known for its security and flexibility having a modern design. OpenSCADA supports editing and debugging and comes with front-end apps, back-end applications, libraries, interface apps, and configuration tools. The set of diverse tools can be combined with the development of advanced IoT apps. Just like the other IDEs, OpenSCADA supports diverse programming languages and consists of sub-projects including Utgard, Atlantis, Orilla, and others.

Home Assistant

Home Assistant is aimed at home automation and functions on the Python-based coding system. It's an open-source tool whose IoT system is controlled with desktop browsers and mobile. Home Assistant is

known for its frictionless operations, privacy standards, and security. The software can support any systems that are running on Python 3. However, it lacks cloud computing and its ability to secure data is a significant disadvantage.

DeviceHub

DeviceHub is an integrated solution that offers a combination of business intelligence and cloud integration for delivering hardware and web technologies. Usually, the kit is offered as a Platform as a Service (PaaS) that allows software developers to use its power for the cause of IoT app development. It's especially beneficial for enterprise bodies who want to rebrand and install software for the need of deploying enterprise apps using Virtual Private Cloud. DeviceHub has achieved success in the fast going building of fleet management systems, smart vending machines, and wearable software

Tessel 2

Tessel 2 is a hardware provider used for creating basic IoT solutions and prototypes. It always gives a helping hand using its several modules and sensors. Tessel has plug and play module ecosystems including 10-pin modules, USB modules, community-created modules, and high-level hardware APIs. This is a kind of board that can hold up a large number of modules that cover the camera, RFID, accelerator, and GPS. Java developers who are well aware of the use of Node.JS can make the use of Tessel and can program it with the use of Node.JS. In this manner, Tessel can be used for churning out a host of servers and also the hardware firmware IoT solutions.

Flutter

Undoubtedly, Flutter is the best choice to make if you're looking for IoT product development. Flutter understands your needs and decreases unnecessary and repetitive electronic tasks. It refers to a programmable processor center that is dependent on Arduino. It is a remote transmitter with the proper inclination for achieving over a half-mile. Shudder sheets have the right to provide permission for coordinating with the correspondence with one another and apply where there is no need for any switch.

Kinoma

Kinoma allows two types of IoT projects that are both serious and fun. To fulfill its purpose, it required only two products: Create and Element boards. Kinoma Create is a scriptable hardware kit that utilizes JavaScript for connecting with the sensors and building the structure of IoT-enabled devices. The kit has complete supporting essential like a BLE (Bluetooth Low Energy), Integrated WiFi, Speaker, touchscreen and a microphone.

5 main principles that must be taken into account by IoT developers before creating an application

Ensure the safe collection of data. The collection of information through special equipment (sensors,

etc.) is carried out outside of the usual data transmission networks. Therefore, when developing a custom Internet of Things application, it is very important to think of ways to protect the received information (in particular, the integrity and security of the initial settings in such devices, and mechanisms for the sensitive data encryption).

Organize high-performance data streaming. As a rule, data collection systems consist of hundreds, even thousands of electronic devices. Therefore, for the efficient streaming of such large volumes of information, it is necessary to think over independent mechanisms that are different from traditional packet transfers.

Create an Internet of Things platform. The IoT platform is a set of software tools that collectively help to systematize, store and process data received from electronic devices.

Develop an Internet of Things solution in the cloud. In order to guarantee the fast delivery of processed data to a user device, and also to organize centralized storage, cloud solutions are usually used. Such systems can ensure the efficient operation of the Internet of Things application with minimal operating costs and requirements for carrier networks.

Provide for effective data management. In-memory analysis and data processing systems are most often used. Such solutions ensure the rapid delivery of the processed results to the end user, even in the event of the data collection devices' failure.

Developing an Internet of Things-based application: 4 consecutive stages

Choose the hardware. Devices designed to collect information are characterized by:

Extremely low levels of power consumption;

Wireless transmitters to support communication;

Primitive embedded OS.

In most cases, the purchase of such equipment is not included in the list of tasks for the developer, since all the popular platforms for development of Internet of Things applications are integrated with a number of physical and virtual tools for the collection of necessary data.

Select centralized data storage. A cloud or similar centralized repository is ideal for accumulation, systematization and further analysis of collected data. In particular, cloud solutions are ideal from the point of view of providing efficient data transmission over low-speed networks.

Develop the server-side of data-handling algorithms. To sort and analyze the collected information, IoT application development platforms offer extensive functionality that allow for the creation of a high-performance, intelligent and scalable backend. Our article about IoT Architecture might help you.

Create a front-end. Almost all modern platforms for the development of Internet of Things applications

allow for the creation of full-featured, user-friendly front-end application interfaces with advanced search functions and intuitive structures.

Development from scratch: choosing the best options to develop IoT apps

To help develop IoT apps from scratch, many special platforms have been created over the last few years. Below, we have listed the most popular, which will allow you to create your own Internet of Things application as quickly as possible.

The top 5 tools to build an Internet of Things application

Azure IoT Suite. Azure IoT Suite is an incredibly popular software package from Microsoft, specifically created for the simple integration of information collection devices into a consolidated system for transferring, storing, analyzing and processing data. Thanks to cloud architecture, the Azure IoT Suite provides access to a reliable and scalable storage bank of large volumes of information. Microsoft Azure Cloud is also supplemented with an extensive list of advanced services, including Azure IoT Hub (for device-to-cloud and cloud-to-device messaging), Azure Stream Analytics (for data sorting), Azure Storage, Azure Cosmos DB (for secure metadata storage and management of assembler devices), and Azure Web Apps Microsoft Power BI (for the creation of front-ends).

Amazon Web Services. It's no news that one of the world's largest trading platforms has introduced special tracking chips that allow for monitoring the location of goods at any stage of their delivery. To provide efficient data exchange between chips and hardware such as PCs, Amazon Web Service was developed in 2006, which is a full-fledged infrastructure of agnostic platforms, including file hosting, cloud computing, virtual servers, and much more. The main advantages of this solution for the Internet of Things apps development include increased security (in compliance with DSS, FISMA, HIPAA and many other specifications), flexibility (thanks to the support of agnostic protocols) and adaptability (due to its cloud architecture).

IBM Watson. The artificial intelligence based IBM Watson software implements support for a reliable relationship between information collection devices, servers and user parts of the developed applications. Creating such an application with the help of IBM Watson services is really simple. Thanks to the advanced functionality that allows for quick connection of gateways and data collection equipment, thoughtful storing and processing algorithms, real-time analysis instruments, and advanced security measures, this set of intuitive tools allows for the quick creation of enterprise level Internet of Things applications. IBM Watson services are available on the popular IBM Bluemix platform - cloud-based PaaS based on SoftLayer infrastructure, with support for Python, Java, Swift, Ruby, Node.js, PHP and other equally popular development languages and frameworks.

Oracle IoT. Oracle IoT is one of the leading software solutions for the development of Internet of Things

applications, built over one of the most flexible programming environments - Oracle. Based on cloud computing technologies, applications created with Oracle IoT have a whole host of advanced capabilities, including device virtualization, high-speed messaging, endpoint management, stream processing, data enrichment, event storage, REST API support, and enterprise connectivity. This is just a few of the Oracle IoT features and is by no means an extensive list. In 2017, this platform was complemented with new products: IoT Asset Monitoring Cloud, IoT Fleet Monitoring Cloud, IoT Production Monitoring Cloud, and IoT Connected Worker Cloud, designed to solve a number of business tasks and improve the process of converting data into the user-friendly form.

KAA IoT. The open-source KAA IoT platform offers an incredibly rich toolkit for developing IoT applications, embodying best practices for software creation. KAA IoT has a lot of advanced features; among them, a well-thought-out functionality for the adjustment of mobile device compatibility, flexible management of an unlimited number of sensors for the collection of information through an SDK server, real-time sensor monitoring, cloud services, automation of software updates, automated user personal device settings distribution, etc. All these features, when combined, make KAA IoT one of the most advanced products for the development of this kind of software.

The 5 fastest-growing areas for Internet of Things applications

Smart homes. Automated house management systems are being actively promoted in highly developed countries. Software for water, electrical, gas and heating resource planning, security and remote control systems are all based on the Internet of Things concept, and will soon minimize the need for human oversight by becoming completely common attributes of everyday life.

Retail. The retail sphere opens up ample opportunities to develop apps for the Internet of Things. Such trade process branches as supply chain control (monitoring of the storage conditions of goods at each stage of their delivery) and intelligent shopping (creating a collection of goods based on location, acceptable price range and individual characteristics of the consumer) are the most fertile for the creation of high-end Internet of Things software.

Electronic health care. Innovative technologies are being introduced to healthcare every year. The IoT concept is actively used in the development of applications for patient status monitoring, diagnostics, monitoring of temperature, moisture and UV radiation in medical products storage, as well as analysis of environmental conditions, etc.

Logistics. Logistics are one of the most viable areas for the implementation of Internet of Things. Using active and passive RFID tags, equipped with integrated chips, antennas, and GPS trackers, such applications provide an integrated approach to tracking the location of transported goods. In order to transform data collected from sensors, logistics companies hire professionals to develop highly

specialized software based on the Internet of Things. Such solutions are compatible with even low-performance mobile devices.

Manufacturing. The "machine-to-machine" principle was the forerunner of the Internet of Things concept, with a simpler structure (unlike M2M, IoT implies the use of middleware for data processing). This has been an indispensable attribute of industrial software in recent years. Nevertheless, existing M2M solutions can be integrated into the Internet of Things. An example of this is the software for the implementation of production tasks. Such solutions include digital product quality control systems, asset management systems, production equipment management systems, etc.

Raspberry Pi

IoT Device

A "Thing" in Internet of Things (IoT) can be any object that has a unique identifier and which can send/receive data (including user data) over a network (e.g., smart phone, smart TV, computer, refrigerator, car, etc.).

• IoT devices are connected to the Internet and send information about themselves or about their surroundings (e.g. information sensed by the connected sensors) over a network (to other devices or servers/storage) or allow actuation upon the physical entities/environment around the remotely.

IoT Device Examples

A home automation device that allows remotely monitoring the status of appliances and controlling the appliances.

• An industrial machine which sends information about its operation and health monitoring data to a server.

• A car which sends information about its location to a cloud-based service.

• A wireless-enabled wearable device that measures data about a person such as the number of steps walked and sends the data to a cloud-based service.

Basic building blocks of an IoT Device

1. Sensing: Sensors can be either on-board the IoT device or attached to the device.

2. Actuation: IoT devices can have various types of actuators attached that allow taking actions upon the physical entities in the vicinity of the device.

3. Communication: Communication modules are responsible for sending collected data to other devices or cloud-based servers/storage and receiving data from other devices and commands from remote applications.

4. **Analysis & Processing:** Analysis and processing modules are responsible for making sense of the collected data.

Raspberry Pi

Raspberry Pi is a low-cost mini-computer with the physical size of a credit card. Raspberry Pi runs various flavors of Linux and can perform almost all tasks that a normal desktop computer can do. Raspberry Pi also allows interfacing sensors and actuators through the general purpose I/O pins. Since Raspberry Pi runs Linux operating system, it supports Python "out of the box".

Raspberry Pi is a credit card sized micro processor available in different models with different processing speed starting from 700 MHz. Whether you have a model B or model B+, or the very old version, the installation process remains the same. People who have checked out the official Raspberry Pi website, but using the Pi is very easy and from being a beginner, one will turn pro in no time. So, it's better to go with the more powerful and more efficient OS, the Raspbian. The main reason why Raspbian is extremely popular is that it has thousands of pre built libraries to perform many tasks and optimize the OS. This forms a huge advantage while building applications.

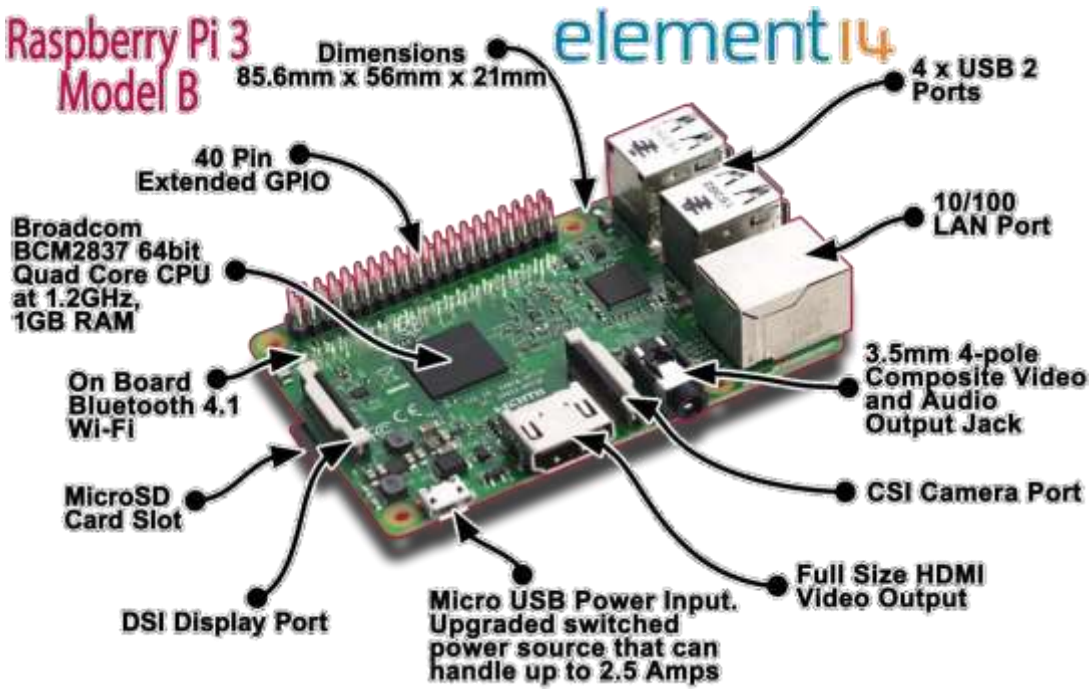


Fig5.7. Raspberry Pi Element

Specifications and performance

As for the specifications, the Raspberry Pi is a credit card-sized computer powered by the Broadcom BCM2835 system-on-a-chip (SoC). This SoC includes a 32-bit ARM1176JZFS processor, clocked at 700MHz, and a Videocore IV GPU.

It also has 256MB of RAM in a POP package above the SoC. The Raspberry Pi is powered by a 5V micro USB AC charger or at least 4 AA batteries (with a bit of hacking).

While the ARM CPU delivers real-world performance similar to that of a 300MHz Pentium 2, the

Broadcom GPU is a very capable graphics core capable of hardware decoding several high definition video formats. The Raspberry Pi model available for purchase at the time of writing — the Model B — features HDMI and composite video outputs, two USB 2.0 ports, a 10/100 Ethernet port, SD card slot, GPIO (General Purpose I/O Expansion Board) connector and analog audio output (3.5mm headphone jack). The less expensive Model A strips out the Ethernet port and one of the USB ports but otherwise has the same hardware.

	Raspberry pi 3 model B	**Raspberry pi 2 model B**	**Raspberry Pi zero**
RAM	1GB SDRAM	1GB SDRAM	512 MB SDRAM
CPU	Quad cortex A53@1.2GHz	Quad cortex A53@900MHz	ARM 11@ 1GHz
GPU	400 MHz video core IV	250 MHz video core IV	250 MHz video core IV
Ethernet	10/100	10/100	None
Wireless	802.11/Blueto oth 4.0	None	None
Video output	HDMI/ Composite	HDMI/ Composite	HDMI/Composite
GPIO	40	40	40

Fig 5.8 Configuration

Raspberry Pi Basics: installing Raspbian and getting it up and running

1. Downloading Raspbian and Image writer.

You will need an image writer to write the downloaded OS into the SD card (micro SD card in case of Raspberry Pi B+ model). So download the "win32 disk imager" from the website.

2. Writing the image

Insert the SD card into the laptop/pc and run the image writer. Once open, browse and select the downloaded Raspbian image file. Select the correct device that is the drive representing the SD card. If the drive (or device) selected is different from the SD card then the other selected drive will become corrupted. SO be careful.

After that, click on the "Write" button in the bottom. As an example, see the image below, where the SD card (or micro SD) drives is represented by the letter "G:\"

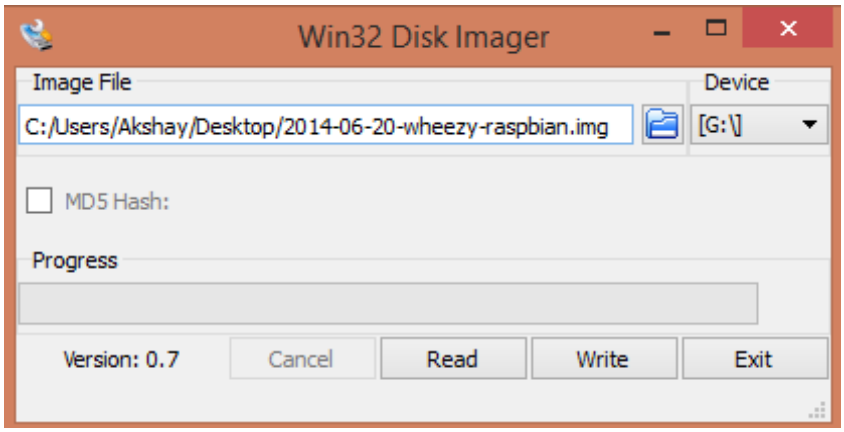


Fig 5.9 OS Installation

Once the write is complete, eject the SD card and insert it into the Raspberry Pi and turn it on. It should start booting up.

3. Setting up the Pi

Please remember that after booting the Pi, there might be situations when the user credentials like the "username" and password will be asked. Raspberry Pi comes with a default user name and password and so always use it whenever it is being asked. The credentials are:

Login: pi

Password: raspberry

When the Pi has been booted for the first time, a configuration screen called the "Setup Options" should appear and it will look like the image below.

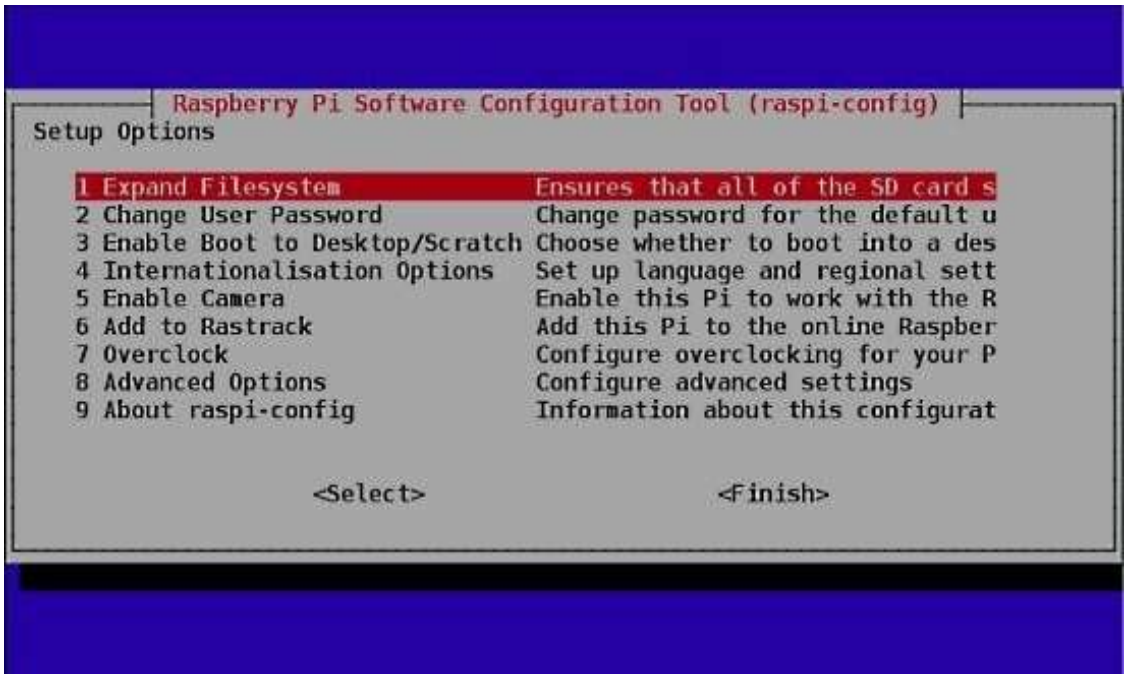


Fig 5.10 Raspberry Configuration

If you have missed the "Setup Options" screen, its not a problem, you can always get it by typing the following command in the terminal.

sudoraspi-config

Once you execute this command the "Setup Options" screen will come up as shown in the image above. Now that the Setup Options window is up, we will have to set a few things. After completing each of the steps below, if it asks to reboot the Pi, please do so. After the reboot, if you don't get the "Setup Options" screen, then follow the command given above to get the screen/window.

The first thing to do:

Select the first option in the list of the setup options window, that is select the "Expand Filesystem" option and hit the enter key. We do this to make use of all the space present on the SD card as a full partition. All this does is, expand the OS to fit the whole space on the SD card which can then be used as the storage memory for the Pi

The second thing to do:

Select the third option in the list of the setup options window, that is select the "Enable BootTo Desktop/Scratch" option and hit the enter key. It will take you to another window called the "choose boot option" window that looks like the image below.

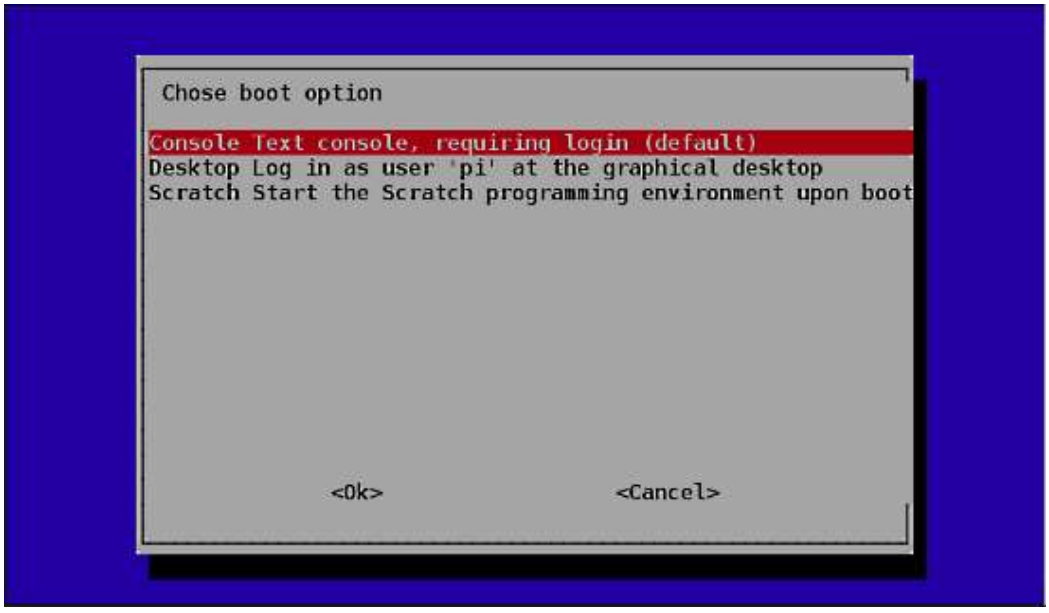


Fig 5.11 Boot Options

In the "choose boot option window", select the second option, that is, "Desktop Log in as user 'pi' at the graphical desktop" and hit the enter button. Once done you will be taken back to the "Setup Options" page, if not select the "OK" button at the bottom of this window and you will be taken back to the previous window. We do this because we want to boot into the desktop environment which we are familiar with. If we don't do this step then the Raspberry Pi boots into a terminal each time with no GUI options. Once, both the steps are done, select the "finish" button at the bottom of the page and it should

reboot automatically. If it doesn't, then use the following command in the terminal to reboot.

sudo reboot

Updating the firmware

After the reboot from the previous step, if everything went right, then you will end up on the desktop which looks like the image below.

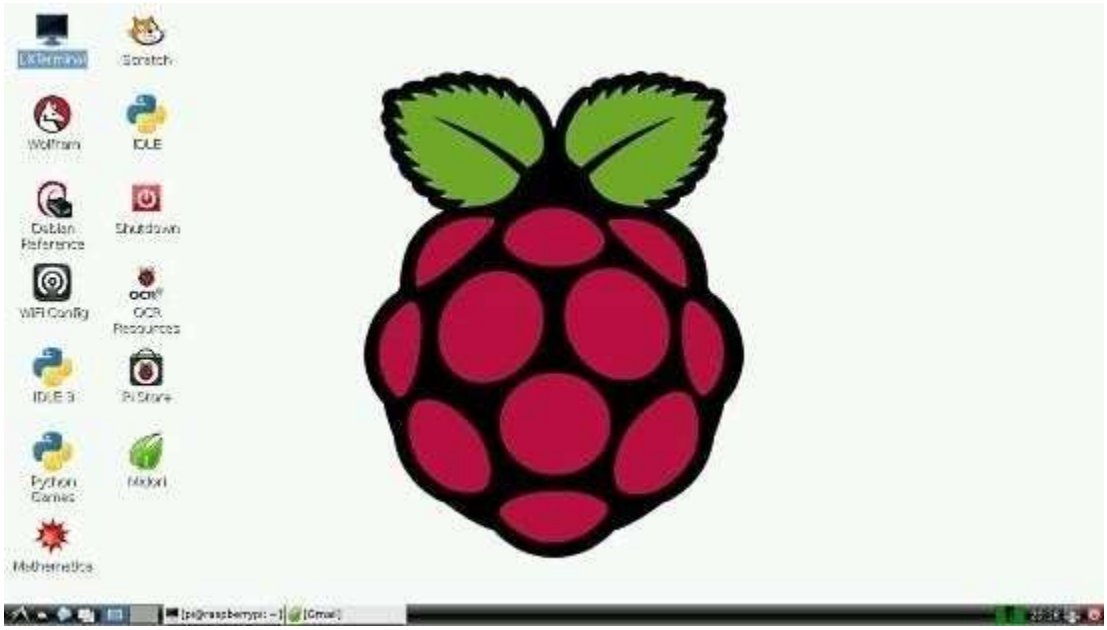


Fig 5.12 Raspberry Desktop

Once you are on the desktop, open a terminal and enter the following command to update the firmware of the Pi.

sudorpi-update

Updating the firmware is necessary because certain models of the Pi might not have all the required dependencies to run smoothly or it may have some bug. The latest firmware might have the fix to those bugs, thus its very important to update it in the beginning itself.

5. Conclusions

So, we have covered the steps to get the Pi up and running. This method works on all the different models of Raspberry Pi (model A, B, B+ and also RPi 2) as Raspbain was made to be supported on all models. However, while installing other software or libraries, the procedure might change a bit while installing depending on the model of the Pi or the version of Raspbian itself. The concept of Raspberry is to keep trying till you get the result or build that you want. This might involve a lot of trial and error but spending the time will be worth it. The actual usage doesn't end here. This is just the beginning. It is up to you to go ahead to build something amazing out of it.

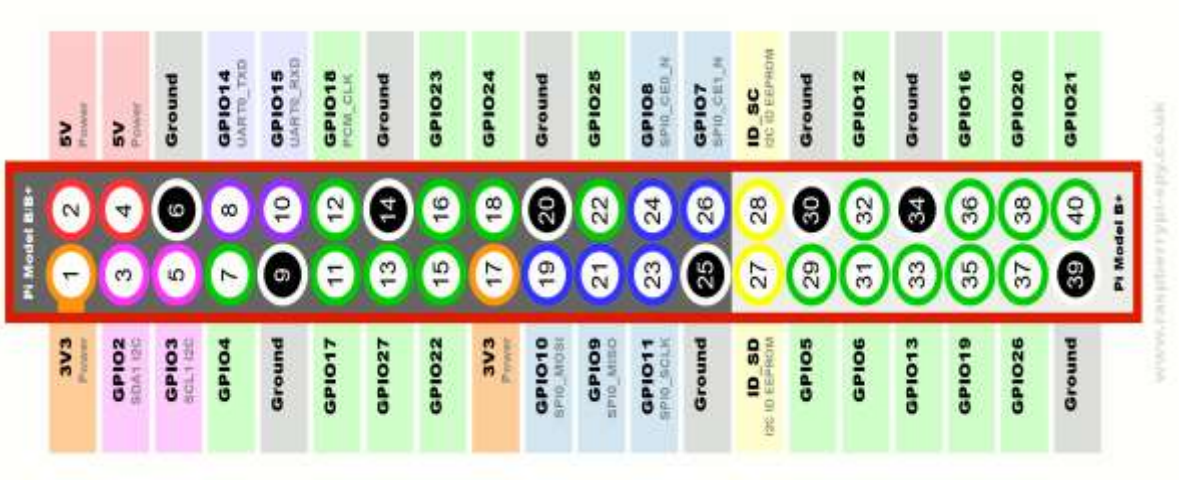


Fig 5.13 GPIO Pins

GPIO:

Act as both digital output and digital input.

Output: turn a GPIO pin high or low.

Input: detect a GPIO pin high or low

Installing GPIO library:

Open terminal

Enter the command ―sudoapt-get install python-dev" to install python development Enter the command "sudoapt-get install python- rpi.gpio" to install GPIO library. Basic python coding: Open terminal enter the command sudonanofilename.py

This will open the nano editor where you can write your code Ctrl+O : Writes the code to the file Ctrl+X : Exits the editor

Blinking LED Code:

import RPi.GPIO as GPIO #GPIO library import time

GPIO.setmode(GPIO.BOARD) # Set the type of board for pin

numbering GPIO.setup(11, GPIO.OUT) # Set GPIO pin 11as output

pin

for i in range (0,5):

GPIO.output(11,True) # Turn on GPIO pin 11 time.sleep(1) GPIO.output(11,False)

Time.sleep(2)

GPIO.output(11,Tr ue)

GPIO.cleanup()

Power Pins

The header provides 5V on Pin 2 and 3.3V on Pin 1. The 3.3V supply is limited to 50mA. The 5V supply draws current directly from your microUSB supply so can use whatever is left over after the board has

taken its share. A 1A power supply could supply up to 300mA once the Board has drawn 700mA.

Basic GPIO

The header provides 17 Pins that can be configured as inputs and outputs. By default they are all configured as inputs except GPIO 14 & 15.

In order to use these pins you must tell the system whether they are inputs or outputs. This can be achieved a number of ways and it depends on how you intend to control them. I intend on using Python.

SDA & SCL: The 'DA' in SDA stands for data, the 'CL' in SCL stands for clock; the S stands for serial. We can do more reading about the significance of the clock line for various typesof computer bus, We will probably find I2C devices that come with their own userspace drivers and the linux kernel includes some as well. Most computers have an I2C bus, presumably for some of the purposes listed by wikipedia, such as interfacing with the RTC (real time clock) and configuring memory. However, it is not exposed, meaning you can't attach anything else to it, and there are a lot of interesting things that could be attached -- pretty much any kind of common sensor (barometers, accelerometers, gyroscopes, luminometers, etc.) as well as output devices and displays. You can buy a USB to I2C adapter for a normal computer, but they cost a few hundred dollars. You can attach multiple devices to the exposed bus on the pi.

UART, TXD & RXD: This is a traditional serial line; for decades most computers have had a port for this and a port for parallel.1 Some pi oriented OS distros such as Raspbian by default boot with this serial line active as a console, and you can plug the other end into another computer and use some appropriate software to communicate with it. Note this interface does not have a clock line; the two pins may be used for full duplex communication (simultaneous transmit and receive).

PCM, CLK/DIN/DOUT/FS: PCM is is how uncompressed digital audio is encoded. The data stream is serial, but interpreting this correctly is best done with a separate clock line (lowest level stuff).

SPI, MOSI/MISO/CE0/CE1: SPI is a serial bus protocol serving many of the same purposes as I2C, but because there are more wires, it can operate in full duplex which makes it faster and more flexible.

Raspberry Pi Terminal Commands

[sudo apt-get update] - Update Package Lists

[sudo apt-get upgrade] - Download and Install Updated Packages

[sudoraspi-config] - The Raspberry Pi

Configuration Tool

[sudo apt-get clean] - Clean Old Package Files

[sudo reboot] - Restart your Raspberry Pi

[sudo halt] - Shut down your Raspberry Pi

5.5 DIY Kits – Soil moisture monitoring

1. Soil Moisture Monitoring

Soil moisture sensors measure the volumetric water content in soil. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighting of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content. The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, temperature, or electric conductivity. Reflected microwave radiation is affected by the soil moisture and is used for remote sensing in hydrology and agriculture.

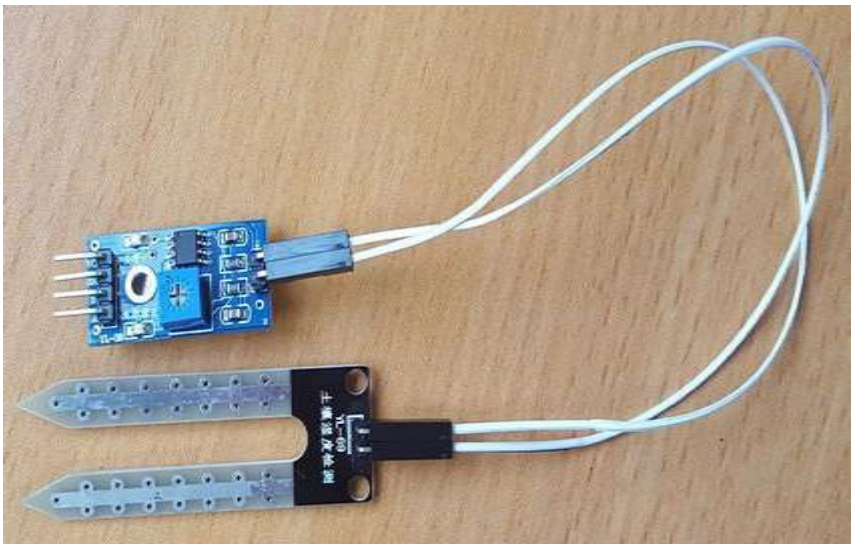


Fig 5.14 Soil Moisture Sensor

Now let's wire the sensor to the Raspberry Pi.

VCC-->3v3

GND --> GND

D0 --> GPIO 17 (Pin 11)

With everything now wired up, we can turn on the Raspberry Pi. Without writing any code we can test to see our moisture sensor working. When the sensor detects moisture, a second led will illuminate .So as a quick test, grab a glass of water (be very careful not to spill water!!) then place the probes into the water and see the detection led shine. If the detection light doesn't illuminate, you can adjust the potentiometer on the sensor which allows you to change the detection threshold (this only applies to the digital output signal). In this example we want to monitor the moisture levels of our plant pot. So we want to set the detection point at a level so that if it drops below we get notified that our plant pot is too dry and needs watering. Our plant here, is a little on the dry side, but ok for now, if it gets any drier it'll need watering.

Fig 5.15 Experimental Setup

To run the script simply runs the following command from the same directory as the script: sudo python moisture.py

1.1 Code

```
import RPi.GPIO as GPIO import smtplib
import time
smtp_username = "enter_username_here"
# This is the username used to login to your SMTP provider
smtp_password = "enter_password_here"
# This is the password used to login to your SMTP provider
smtp_host = "enter_host_here"
 # This is the host of the SMTP provider smtp_port = 25
# This is the port that your SMTP provider uses smtp_sender = "sender@email.com"
```

This is the FROM email address smtp_receivers = ['receiver@email.com']

This is the TO email address message_dead = """From: Sender Name <sender@email.com>

To: Receiver Name <receiver@email.com>

Subject: Moisture Sensor Notification

This is the message that will be sent when moisture IS detected again message_alive = """From:

Sender Name <sender@email.com>

To: Receiver Name <receiver@email.com>

Subject: Moisture Sensor Notification

This is our sendEmail function

def sendEmail(smtp_message): try:

smtpObj = smtplib.SMTP(smtp_host, smtp_port)

smtpObj.login(smtp_username, smtp_password)

If you don't need to login to your smtp provider, simply remove this line

smtpObj.sendmail(smtp_sender, smtp_receivers, smtp_message)

print "Successfully sent email" except smtplib.SMTPException:

print "Error: unable to send email"

This is our callback function, this function will be called every time there is a change on the specified GPIO channel, in this example we are using 17

def callback(channel):

if GPIO.input(channel):

print "LED off" sendEmail(message_dead)

else:

print "LED on" sendEmail(message_alive)

Set our GPIO numbering to BCM G

PIO.setmode(GPIO.BCM)

Define the GPIO pin that we have our digital output from our sensor connected to channel = 17

This line tells our script to keep an eye on our gpio pin and let us know when the pin goes HIGH or LOW

GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300)

This line assigns a function to the GPIO pin so that when the above line tells us there is a change on the pin, run this function

GPIO.add_event_callback(channel, callback)

This is an infinte loop to keep our script running while True:

# This line simply tells our script to wait 0.1 of a second, this is so the script doesnt hog all of the CPU
time.sleep(0.1)

Soil Moisture Monitoring using arduino

The soil moisture sensor consists of two probes that measure the volume of water in the soil. The two probes allow the electric current to pass through the soil and, according to its resistance, measures the moisture level of the soil.

When there is more water, the soil conducts more electricity, which means that the resistance will be less. So the moisture level will be higher. Dry soil reduces conductivity. So, when there is less water, the soil conducts less electricity, which means it has more resistance. So the moisture level will be lower.

Soil Moisture Sensor – Pin Out

There are different types of soil moisture sensor on the market, but their working principal are all similar; All of these sensors have at least three pins: VCC, GND, and AO. The AO pin changes according to the amount of moisture in the soil and increases as there is more water in the soil. Some models have an additional base called DO. If the moisture amount is less than the permissible amount (which can be changed by the potentiometer on the sensor) the DO pin will be “1”, otherwise will remain”0″.

Interfacing Soil Moisture Sensor and Arduino

Soil moisture Sensor has a detection length of 38mm and a working voltage of 2V-5V. It has a Fork-like design, which makes it easy to insert into the soil. The analog output voltage boosts along with the soil moisture level increases.

Circuit

Using this sensor is quite easy. You connect the AO pin to any analog pin. If your sensor has a DO pin, you can connect it to any digital pin.

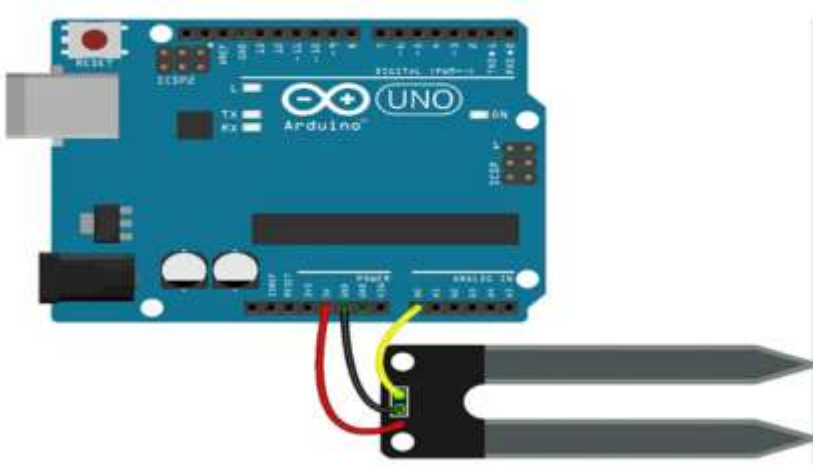


Fig 5.16: Soil moisture sensor connection

```
/*Soil Moisture Sensor
```

```
 modified on 21 Feb 2019
 by Saeed Hosseini @ Electropeak
 https://electropeak.com/learn/
*/
#define SensorPin A0
float sensorValue = 0;
void setup() {
 Serial.begin(9600);
}
void loop() {
 for (int i = 0; i <= 100; i++)
 {
   sensorValue = sensorValue + analogRead(SensorPin);
   delay(1);
 }
 sensorValue = sensorValue/100.0;
 Serial.println(sensorValue);
 delay(30);
}
```

For each soil moisture measurement, we took an average of 100 sensor data to make the data more stable and accurate.

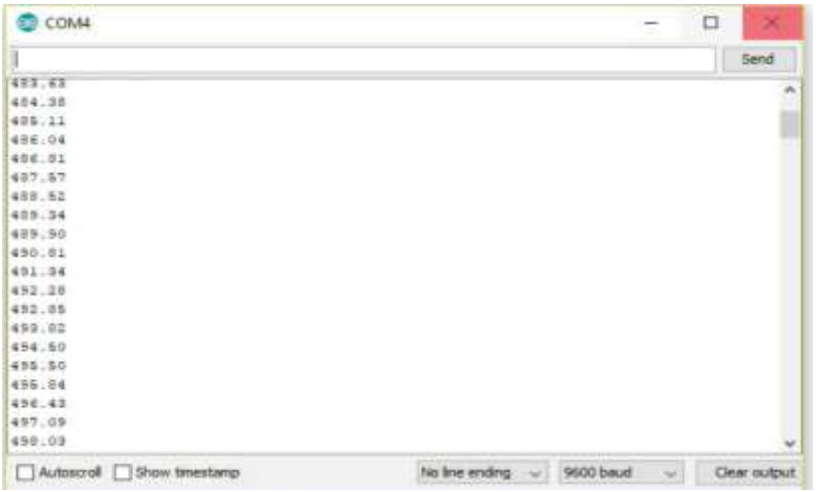


Fig 5.17: Output

note that after 10-20 months, the sensor may get oxidized in the soil and lose its accuracy. Therefore it should be replaced every year. Since it has a low price and easy setup, it worths the annual replacement.

5.6 Weather monitoring using arduino

The system deals with monitoring and controlling the environmental conditions like temperature, relative humidity and CO level with sensors and sends the information to the web page and then plot the sensor data as graphical statistics.

Many high-end systems are now available for 24-hour weather observation. But these systems are being plotted on a Large scale to monitor real-time Weather in a state or an entire state. The implementation

of such a system for a small area is not possible because they are not set for it and the effort for the maintenance of such systems for a small area is very high. The new system uses 3 sensors to measure atmospheric and environmental factors such as temperature, humidity, light intensity, dew point and thermal index. The values read by the sensors are processed by the Arduino microcontroller and stored in a text file that can be processed for analysis. The readings are also displayed on an integrated LCD screen for quick viewing. All these measurements can be analyzed to determine the weather characteristics of a particular region and record the weather profile. These saved settings are essential and vary from place to place. All these requirements are entered into the database and these values are essential and are recorded over time. With these input values, we can draw a weather map for a particular region in time.

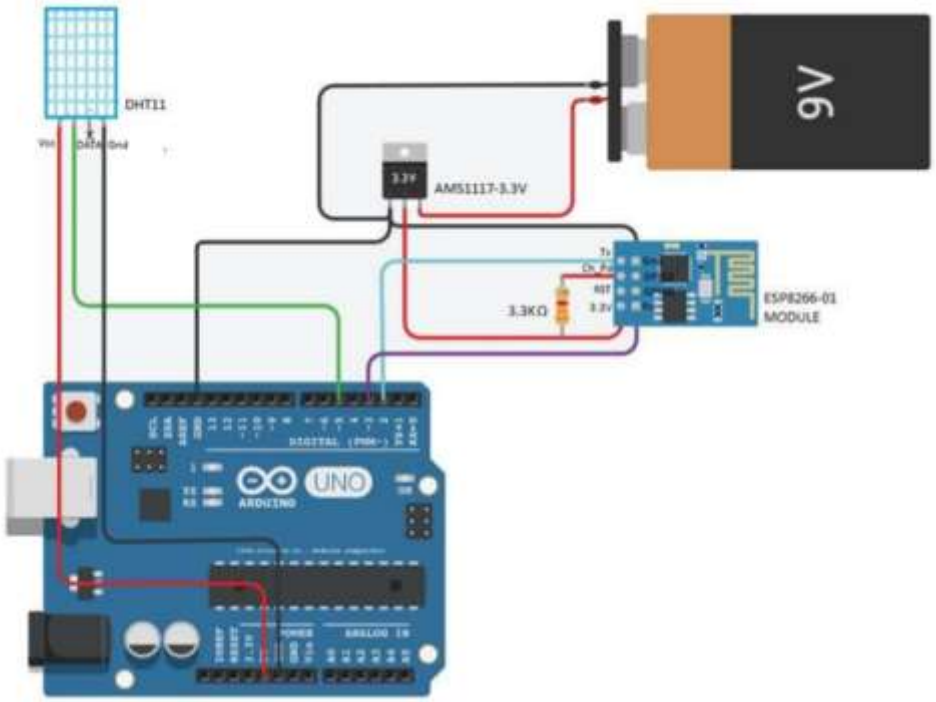


Fig 5.18: Weather monitoring

The Block diagram is followed by the construction of the circuit diagram in which all the components i.e., different sensors, Wi-Fi module etc. are mounted over the microcontroller and the breadboard.

Temperature & Humidity Sensor DHT11 temperature and humidity sensor will be used which is best model for measurement. In this Temperature will display in Celsius. Humidity(H) will display in % The sensor tells 20 % - 80% readings which is 5% accurate and -40 - 80-degree Celsius temperature that is +-0.5 accurate, it can be decrease or increase.

This code is written in Arduino IDE and then uploaded over to the respective port over which the

UNO microcontroller is mounted.

```
#include <DHT.h>
#include<software_Serial.h>
#Define DHTPIN 6
#define DHTTYPE DHT11
String APIKey= "RIZHUWGD2V9L8";
String host_name= "Pixels";
String Password = "abh523";
Software _serial ser(2, 4);
int I =1;
DHT(DHT_pin,DHT_type);
void _setup()
{
Serial.Begin (8600);
Ser_begin (8600);
Ser _println("AT+RST");
DHT.bgin();
char inv = “ “';
string_cmd = "AT+cajwp";
cmd+= "=";
cmd+= inv;
cmd+=H_name;
cmd == inv;
cmd+= pass;
cmd+= inv;
ser.println (CMD);
 }
void loop()
{
```

```
 int HUMIDITY= dht.readHUMIDITY ();
 int TEMPERATURE= dht.readTMPERATURE ();
 String_State1=String (HUMIDITY);
String state2=String(TEMPERATURE);
String _cmd = "AT+STARTCIP=\"TCP\",\"";
Cmd += "184.106.153.149";
cmd += "\" ,85";
seri_println (cmd);
Serial_println(cmd);
if(ser_find("ERROR")){
Serial_Println("AT+STARTCIP error");
Return;
}
String get_Str = "GET /update?api_key=";
Get_Str += api KEY;
Get_Str +="&FIELD1=";
Get_Str += S tring (STATE1);
Get_Str +="&FIELD2=";
Get_Str += String(STATE2);
Get_Str += "\r\n\r\n";
Cmd = "AT+CIP_SEND=";
Cmd += string(getStr.length());
ser_println(cmd);
Serial_println(cmd);
if(SER_FIND(">")){
ser_print(getStr);
Serial_ print(getStr);
 }
```

```
else{
ser.println("AT+CLOSECIP");
Serial.println("AT+CLOSECIP");
 }
delay(1200);
}
```

Weather monitoring offers different kind of measurements to provide various important information about temperature, soil, some of them are:

1.Air and soil temperature

2. Humidity

3.Moisture

4.Rainfall level

5.Wind speed

6.Evapotranspiration

7.Air pollution

Features and advantages

1.Our proposed ‘Smart weather monitoring system’ unlike conventional weather monitoring instruments is very small and compact allowing it to be installed easily on rooftops.

2. It is light and portable; this advantage allows us to easily carry it to remote location for installation. Due to its design it can be easily be carried by a weather balloon to measure atmospheric changes at high altitudes

3. The power requirements for our system (sensors and boards) is much less compared to the existing instruments in the market hence enabling us to use solar cells as power supply. This not only cuts down on cost but allows us to leave the monitoring system in remote, areas where power is not easily available, for long periods of time. Addition of solar panels also helps our design be eco-friendly.

4. The sensors used in our product are much cheaper compared to the ones that are used in the existing weather monitoring systems making our design more cost effective.

5. These sensors send the data to a web page and the sensor data is plotted as graphical statistics. The data uploaded to the web page can easily be accessible from anywhere in the world. The data gathered in these web pages can also be used for future references. Unlike the existing system where data has to be physically transferred.6. Due to the presence of fewer moving parts less amount of maintenance will be needed cutting down on maintenance charges.

The system deals with monitoring weather and climate changes like

1.Temperature, humidity by using the DHT11 sensor

2.Wind speed using an Anemometer

3. Light intensity using an LDR

4. UV radiation using a GY8511 solar sensor

5. Carbon monoxide levels in the air using MQ7

6. Soil moisture using Hygrometer

7. Ultrasonic sensor for rain water level

8. Raindrop sensor for detecting rainfall or snow fall.

Here Arduino Unos a microcontroller for the simple brain of the system. When using Arduino as a microcontroller, there need a Wi-Fi module to establish your Internet connection. And the DHT sensor, which (digital humidity sensor) can detect differences in temperature, humidity and humidity at a certain location, must be integrated into the system. The sensor continuously monitors temperature changes and sends data to the microcontroller. The microcontroller transfer the data for its storage and visualization to cloud.

Weather monitoring

The DHT11 is a low-cost temperature and humidity sensor. It isn't the fastest sensor around but its cheap price makes it useful for experimenting or projects where you don't require new readings multiple times a second. The device only requires three connections to the Pi.

+3.3v, ground and one GPIO pin.

DHT11 Specifications

The device itself has four pins but one of these is not used. You can buy the 4-pin device on its own or as part of a 3-pin module. The modules have three pins and are easy to connect directly to the Pi's GPIO header.

Humidity: 20-80% (5% accuracy)

Temperature: 0-50°C (±2°C accuracy)

Hardware Setup

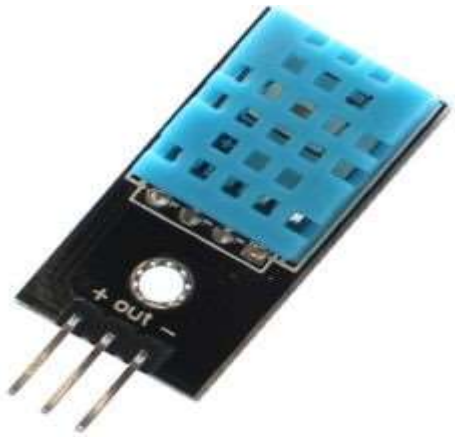


Fig 5.18 Humidity and Temperature Sensor

The 4-pin device will require a resistor (4.7K-10K) to be placed between Pin 1 (3.3V) and Pin 2 (Data). The 3-pin modules will usually have this resistor included which makes the wiring a bit easier. The 3 pins should be connected to the Pi as shown in the table below:

DHT Pin	Signal	Pi Pin
1	3.3V	1
2	Data/Out	11 (GPIO17)
3	not used	–
4	Ground	6 or 9

Your data pin can be attached to any GPIO pin you prefer. In my example I am using physical pin 11 which is GPIO 17. Here is a 4-pin sensor connected to the Pi's GPIO header. It has a 10K resistor between pin 1 (3.3V) and 2 (Data/Out).

Python Library

The DHT11 requires a specific protocol to be applied to the data pin. In order to save time trying to implement this yourself it's far easier to use the Adafruit DHT library. The library deals with the data that needs to be exchanged with the sensor but it is sensitive to timing issues. The Pi's operating system may get in the way while performing other tasks so to compensate for this the library requests a number of readings from the device until it get one that is valid. To start with update your package lists and install a few Python libraries:

sudo apt-get update

sudo apt-get install build-essential python-dev

Then clone the Adafruit library from their repository :

Git clone https://github.com/adafruit Cd Adafruit_Python_DHT

Then install the library for Python 2 and Python 3 sudo python setup.py install

sudo python3 setup.py install python AdafruitDHT.py 11 17

The example script takes two parameters. The first is the sensor type so is set to “11” to represent the DHT11. The second is the GPIO number so for my example I am using “17” for GPIO17. You can change this if you are using a different GPIO pin for your data/out wire. You should see an output similar to this :

Temp=22.0*Humidity=68.0% import Adafruit_DHT

Set sensor type : Options are DHT11,DHT22 or AM2302 sensor=Adafruit_DHT.DHT11

Set GPIO sensor is connected to gpio=17

Use read_retry method. This will retry up to 15 times to

get a sensor reading (waiting 2 seconds between each retry). humidity, temperature = Adafruit_DHT.read_retry(sensor, gpio)

Reading the DHT11 is very sensitive to timings and occasionally

the Pi might fail to get a valid reading. So check if readings are valid.

if humidity is not None and temperature is not None:

print('Temp={0:0.1f}*C Humidity={1:0.1f}%'.format(temperature, humidity)) else:

print('Failed to get reading. Try again!')

Air Quality Monitoring

Air pollution is a major problem in urban centers as well as rural set-up. The major pollutants of concern are primarily carbon monoxide, nitrogen oxides, hydrocarbons and particulate matter (PM10, PM2.5). Ozone, PAN and PBN are other secondary pollutants generated as a result of the photochemical reactions of the primary pollutants. These pollutants affect human health as well as environment. Therefore, air pollution monitoring is necessary to keep a check on the concentration of these pollutants in ambient air. The grove sensors, grove DHT (for temperature and humidity), grove gas sensor modules like dust, MQ-5 (for smoke), MQ-7 (for CO) and MQ-135 (for CO2) are interfaced to this shield for monitoring in our proposed

.

Adafruit CCS811 is a gas sensor that can detect a wide range of Volatile Organic Compounds (VOCs) and is intended for indoor air quality monitoring. When connected to your microcontroller (running our library code) it will return a Total Volatile Organic Compound (TVOC) reading and an equivalent carbon dioxide reading (eCO2) over I2C. There is also an on- board thermistor that can be used to calculate the approximate local ambient temperature.

Power Pins

• Vin - this is the power pin. Since the sensor uses 3.3V, we have included an onboard voltage regulator that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V

• 3Vo - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like

• GND - common ground for power and logic

Logic pins

• SCL - this is the I2C clock pin, connect to your microcontrollers I2C clock line. There is a 10K pull-up on this pin and it is level shifted so you can use 3 - 5VDC.

• SDA - this is the I2C data pin, connect to your microcontrollers I2C data line. There is a 10K pull-up on this pin and it is level shifted so you can use 3 - 5VDC.

• INT - this is the interrupt-output pin. It is 3V logic and you can use it to detect when a new reading is ready or when a reading gets too high or too low.

• WAKE - this is the wakeup pin for the sensor. It needs to be pulled to ground in order to communicate with the sensor. This pin is level shifted so you can use 3- 5VDC logic.

• RST - this is the reset pin. When it is pulled to ground the sensor resets itself. This pin is level shifted so you can use 3-5VDC logic.

Raspberry Pi Wiring & Test

The Raspberry Pi also has an I2C interface that can be used to communicate with this sensor. Once your Pi is all set up, and you have internet access set up, let’s install the software we will need. First make sure your Pi package manager is up to date

from time import sleep

from Adafruit_CCS811 import Adafruit_CCS811 ccs = Adafruit_CCS811()

while not ccs.available(): pass

temp = ccs.calculateTemperature() ccs.tempOffset = temp - 25.0

while(1):

if ccs.available():

temp = ccs.calculateTemperature() if not ccs.readData():

print "CO2: ", ccs.geteCO2(), "ppm, TVOC: ", ccs.getTVOC(), " temp: ", temp else:

print "ERROR!" while(1):

pass sleep(2)

5.7 Movement Detection

PIR stands for passive infrared. This motion sensor consists of a fresnel lens, an infrared detector, and supporting detection circuitry. The lens on the sensor focuses any infrared radiation present around it toward the infrared detector. Our bodies generate infrared heat, and as a result, this heat is picked up by the motion sensor. The sensor outputs a 5V signal for a period of one minute as soon as it detects the presence of a person. It offers a tentative range of detection of about 6–7 meters and is highly sensitive. When the PIR motion sensor detects a person, it outputs a 5V signal to the Raspberry Pi through its GPIO and we define what the Raspberry Pi should do as it detects an intruder through the Python coding. Here we are just printing "Intruder detected".

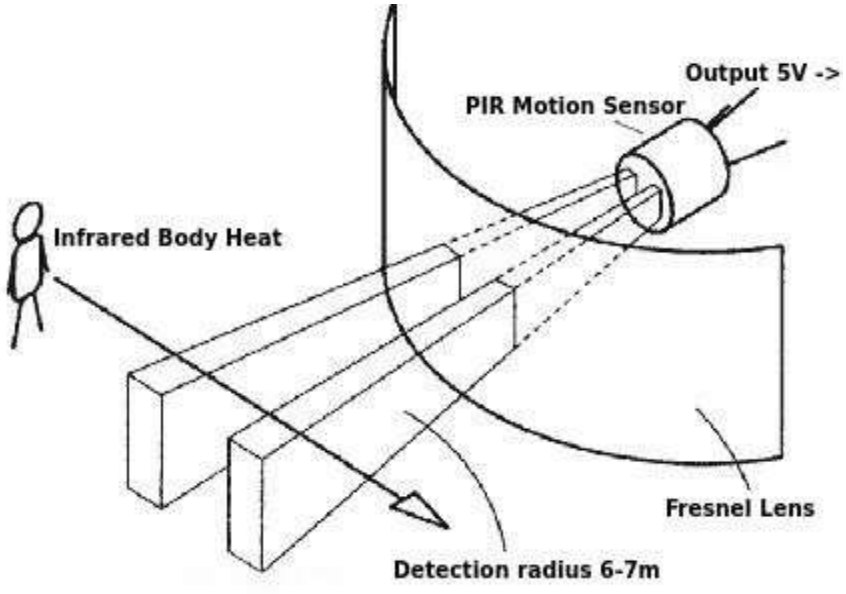


Fig 5.19 Working of PIR Sensor

Working Mechanism

All living beings radiate energy to the surroundings in the form of infrared radiations which are invisible to human eyes. A PIR (Passive infrared) sensor can be used to detect these passive radiations. When an object (human or animal) emitting infrared radiations passes through the field of view of the sensor, it detects the change in temperature and therefore can be used to detect motion.HC-SR501 uses differential detection with two pyroelectric infrared sensors. By taking a difference of the values, the average temperature from the field of view of a sensor is removed and thereby reducing false positives.

```
import RPi.GPIO as GPIO
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
```

TEXT / REFERENCE BOOKS

1. Boswarthick,Omar Elloumi., The Internet of Things: Applications and Protocols, Wiley publications., 2012
2. Dieter Uckelmann, Mark Harrison, Florian Michahelles., Architecting the Internet of Things, Springer publications.2011
3. Marco Schwatrz Internet of Things with Arduino Cookbook, Packt Publications.2016 .
4. Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, “From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence”, 1st Edition, Academic Press, 2014.
5. Vijay Madisetti, Arshdeep Bahga, “Internet of Things: A Hands-On Approach” published by Vijay Madisetti 2014