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 INFORMATION TECHNOLOGY

UNIT - I

SITA3008 - Internet of Things

UNIT 1

Introduction to IoT, Current technological trends and future prospects, - Evolution of IoT , Business Scope, Relation with embedded system, - Basic Architecture of an IoT, From M2M to IoT, M2M towards IoT, IoT Value Chains, An emerging industrial structure for IoT.

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 intelligent interconnectivity 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 allows devices 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 nearly all 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 manufacturing plant 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.

1.2.1 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 device poses 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 millions of 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.

1.2.2 IoT and Mobile Elements

IoT is creating new opportunities and providing a competitive advantage for businesses in current 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 of offering a rich stream of data that will then be used by product and service owners to interact with their consumers.

1.2.3 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.

1.2.4 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).

1.2.5 IoT and New Business Models

The bottom line is a big motivation for starting, investing in, and operating any business, without a sound and solid business models 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 not only monitor long term water interventions such as catchment area management, but may

even 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 significantly increase 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 surveillance e.g., surveillance of spaces, tracking of people and assets, infrastructure and equipment maintenance, 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.

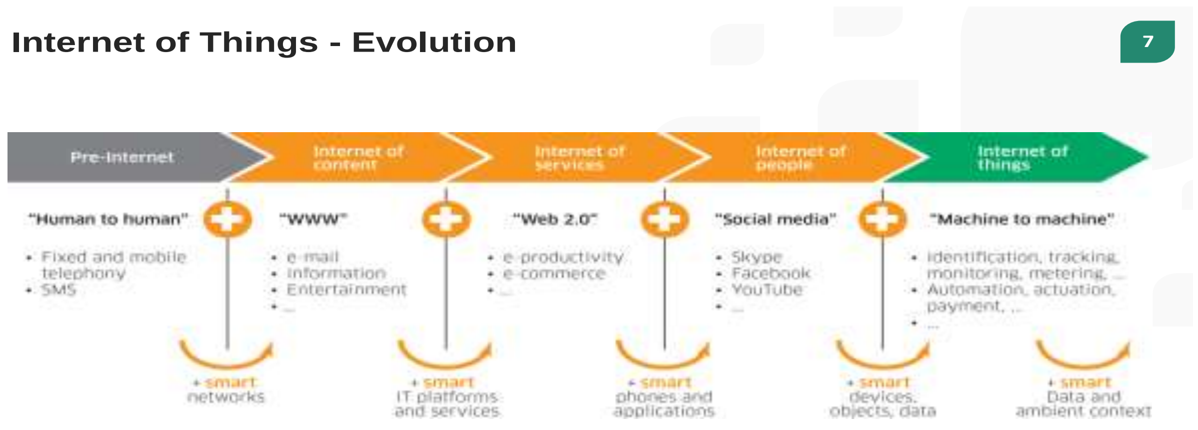


Fig 1. IOT Evolution Model

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 technology allowed “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 it possible 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, which drive 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 interact with 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.

1.4 BUSINESS SCOPE

1.4.1 Increase Business Opportunities

IoT opens the door for new business opportunities and helps companies benefit from new revenue 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.

1.4.2 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 could more easily locate issues in the assets and run preventive maintenance to improve asset utilization.

1.4.3 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 market and downtime due to maintenance.

1.4.4 Improved Safety and Security

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

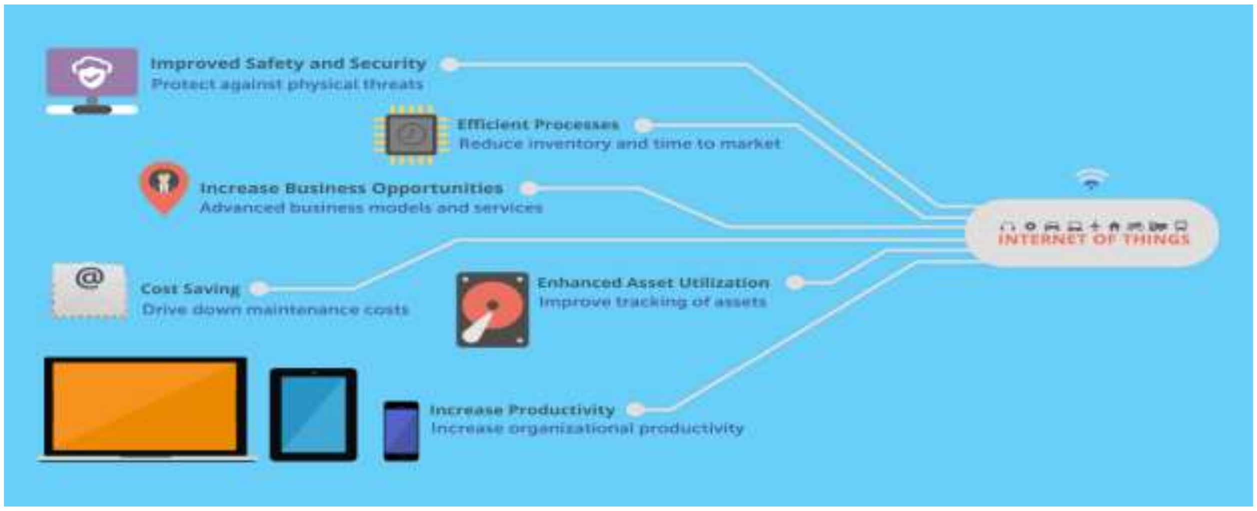

Improved Safety and Security
Protect against physical threats
Efficient Processes
Reduce inventory and time to market
Increase Business Opportunities
Advanced business models and services
INTERNET OF THINGS
Cost Saving
Drive down maintenance costs
Enhanced Asset Utilization
Improve tracking of assets
Increase Productivity
Increase organizational productivity

Fig 2. Business Scope

1.4.5 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.

1.4.6 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.

1.4.7 Logistics

With IoT sensors, supply chain management and order fulfillment processes improve markedly to 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.

1.4.8 Streamlined Industry

IoT also presents automation opportunities for businesses that need to manage and replenish their stock. 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 is running low. This minimizes out-of-stock incidents and prevents excess stock build-up.

1.4.9 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 a few 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.

1.4.10 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 this burgeoning technology.

1.5 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 in Heating, 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 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.

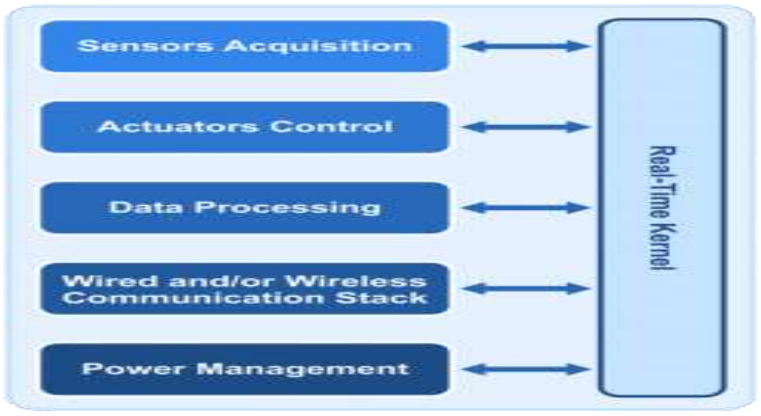


Fig 3. Embedded Processing

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

- **Analog Sensors** produce a continuous output signal or voltage which is generally proportional to the quantity being measured.
- Physical quantities such as Temperature, Speed, Pressure, Displacement, Strain etc. are all analog quantities as they tend to be continuous in nature.
- **Digital Sensors** produce discrete digital output signals or voltages that are a digital representation 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 the system. 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.

1.5.1 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 an attractive 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 is often placed in a separate module, one that has already been certified for the protocol and FCC compliance.

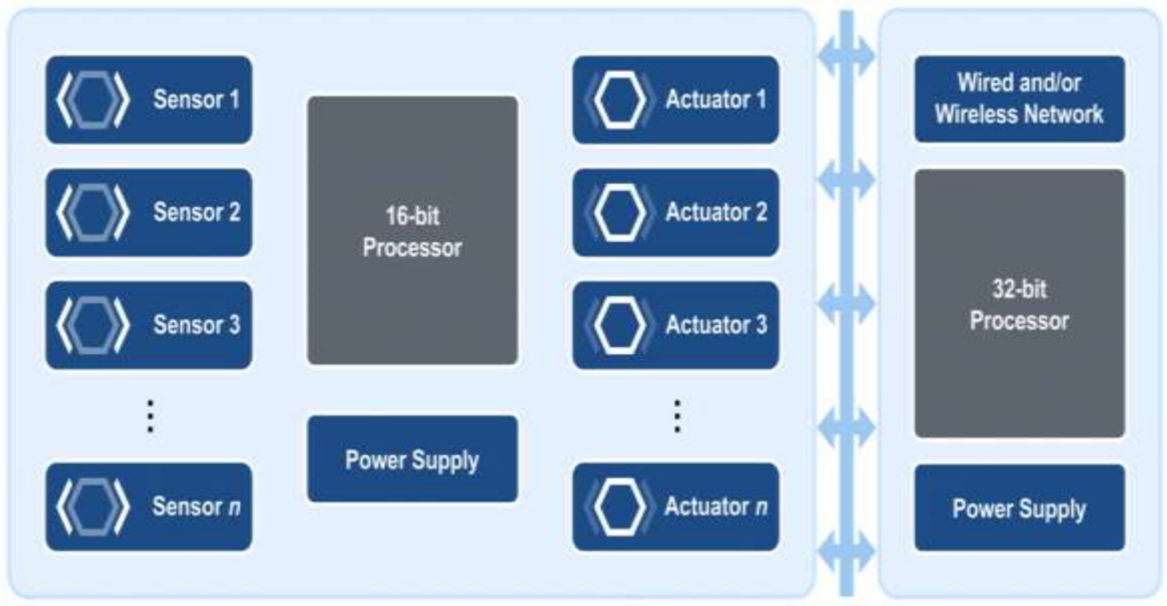


Fig 4. IoT Devices with Two Processors

1.5.2 Gateway Design

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

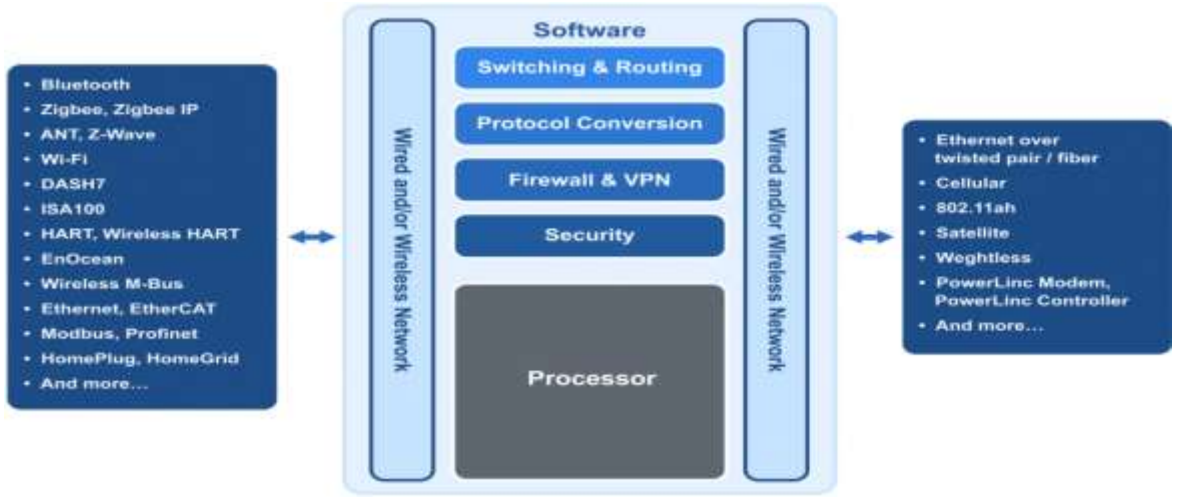


Fig 5 Embedded Devices with Gateway

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 mesh network using low-energy radio waves to communicate from appliance to appliance, allowing for 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 the term **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 with ultra 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.

1.6 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.

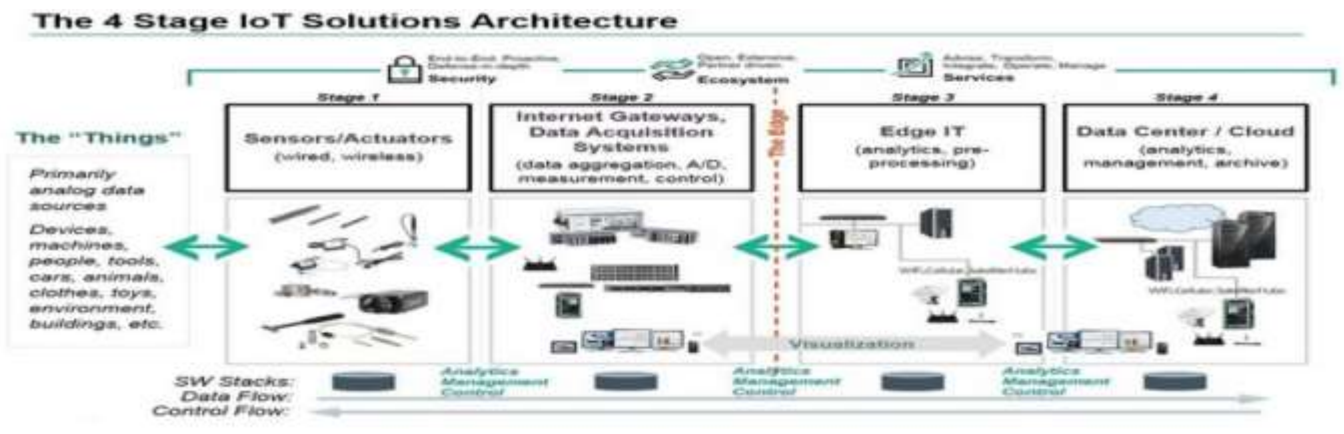

The 4 Stage IoT Solutions Architecture
Security
Ecosystem
Services
Stage 1
Stage 2
Stage 3
Stage 4
The "Things"
Primarily analog data sources
Devices, machines, people, tools, cars, animals, clothes, toys, environment, buildings, etc.
Sensors/Actuators (wired, wireless)
Internet Gateways, Data Acquisition Systems (data aggregation, A/D, measurement, control)
The Edge
Edge IT (analytics, pre-processing)
Data Center / Cloud (analytics, management, archive)
Visualization
SW Stacks:
Data Flow:
Control Flow:
Analytics Management Control

Fig 6. Reference architecture for IoT

Networked things (wireless sensors and actuators)

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 Pi

There 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, or provided 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 an unchangeable 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.

Sensor data aggregation systems and analog-to-digital data conversion

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 brokered model, clients connect an outbound connection to the broker, whether or not the device is acting as a 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 WebSockets will 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 on the larger 32-bit devices.

1.6.1 The appearance of edge IT systems

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 APIs that 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.

1.6.2 Analysis, management, and storage of data

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 happen in 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 is to 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.

1.6.3 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 using machine-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 front end 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 data coming 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 the system. 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.

1.6.4 DEVICE MANAGEMENT

Device management (DM) is handled by two components. A server-side system (the device manager) 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).

1.6.5 IDENTITY AND ACCESS MANAGEMENT

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

• OAuth2 token issuing and validation

• Other identity services including SAML2 SSO and OpenID Connect support for identifying inbound 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.

1.6.7 Delivering Business Value with IOT enabled Infrastructure Management

IoT enabled infrastructure Management is the art of knowing the condition of your assets before you act on them. Knowing in such detail, that failure prediction and preventive action becomes a reliable and efficient infrastructure management strategy. This opposed to doing maintenance and replacement investments based on static decision rules like ‘we always rebuild after 30 years’. Or, waiting for a fatal breakdown to happen.

Knowledge-based Infrastructure Management requires frequent collection of data. It includes data collection of the infrastructure itself but also of the corresponding throughput it facilitates. And, of the external circumstances in which the infrastructure operates. In fact, the condition of the infrastructure and the way it changes or deteriorates over time must be compared with load, volume, and variability data over time. Also, external conditions like temperature, moisture, terrain conditions, etc. may affect failure risks and the rate of degradation.

Analysing breakdown events or loss of performance by looking at the corresponding data prior to the event, statistical decision support models can be built. It tells you with a high degree of accuracy when to do maintenance. And, it indicates when to replace infrastructure components in order to prevent critical events. In this way, you manage your operational risks at the lowest possible cost.

IT technologies, like the development of industrial Internet-of-Things (IoT) devices, cloud computing, artificial intelligence, and machine learning, monitoring infrastructure with IoT enabled devices has become a highly cost-effective approach to collect and deliver data. It also builds decision support models for managing infrastructure businesses.

1.7 From M2M to IoT

Machine-to-machine, or M2M, is a broad label that can be used to describe any technology that enables networked devices to exchange information and perform actions without the manual assistance of humans. Artificial intelligence (AI) and machine learning (ML) facilitate the communication between systems, allowing them to make their own autonomous choices.M2M technology was first adopted in manufacturing and industrial settings, where other technologies, such as SCADA and remote monitoring, helped remotely manage and control data from equipment. M2M has since found applications in other sectors, such as healthcare, business and insurance. M2M is also the foundation for the internet of things (IoT).

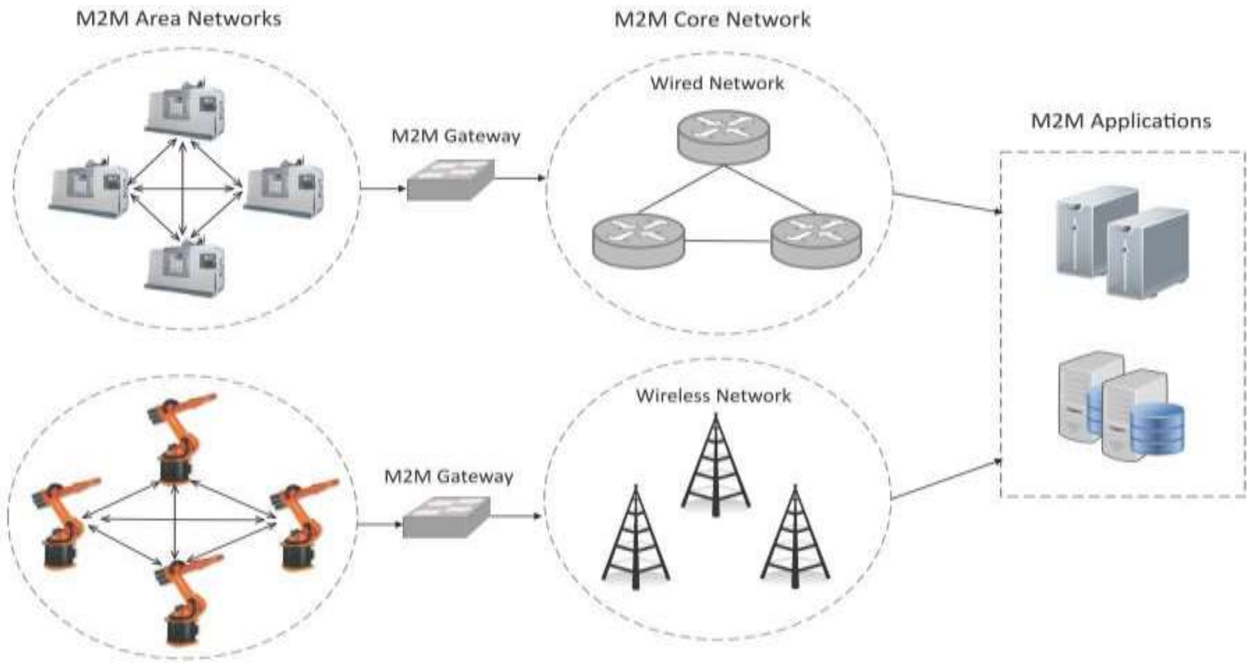


Fig 8 Machine to Machine communication

1.7.1 How M2M works?

The main purpose of machine-to-machine technology is to tap into sensor data and transmit it to a network. Unlike SCADA or other remote monitoring tools, M2M systems often use public networks and access methods -- for example, cellular or Ethernet -- to make it more cost-effective.

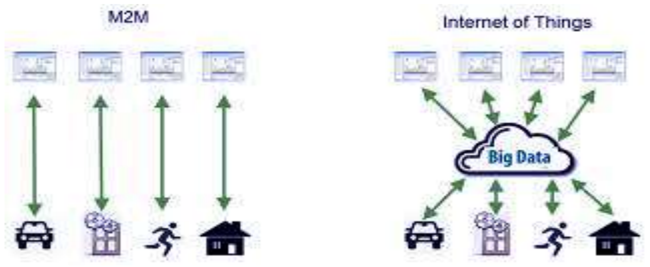


Fig 9 Machine to Machine communication

The main components of an M2M system include sensors, RFID, a Wi-Fi or cellular communications link, and autonomic computing software programmed to help a network device interpret data and make decisions. These M2M applications translate the data, which can trigger preprogrammed, automated actions.

One of the most well-known types of machine-to-machine communication is telemetry, which has been used since the early part of the last century to transmit operational data. Pioneers in telemetrics first used telephone lines, and later, radio waves, to transmit performance measurements gathered from monitoring instruments in remote locations.

The Internet and improved standards for wireless technology have expanded the role of telemetry from pure science, engineering and manufacturing to everyday use in products such as heating units, electric meters and internet-connected devices, such as appliances.

Beyond being able to remotely monitor equipment and systems, the top benefits of M2M include:

- Reduced costs by minimizing equipment maintenance and downtime;
- Boosted revenue by revealing new business opportunities for servicing products in the field; and
- Improved customer service by proactively monitoring and servicing equipment before it fails or only when it is needed.

1.7.2 M2M applications and examples

Machine-to-machine communication is often used for remote monitoring. In product restocking, for example, a vending machine can message the distributor's network, or *machine,* when a particular item is running low to send a refill. An enabler of asset tracking and monitoring, M2M is vital in warehouse management systems (WMS) and supply chain management (SCM).Utilities companies often rely on M2M devices and applications to not only harvest energy, such as oil and gas, but also to bill customers -- through the use of smart meters -- and to detect worksite factors, such as pressure, temperature and equipment status.

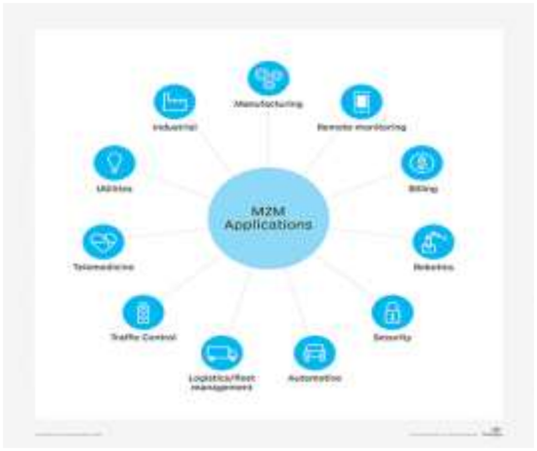


Fig 8 M2M Applications

In telemedicine, M2M devices can enable the real time monitoring of patients' vital statistics, dispensing medicine when required or tracking healthcare assets.

The combination of the IoT, AI and ML is transforming and improving mobile payment processes and creating new opportunities for different purchasing behaviors. Digital wallets, such as Google Wallet and Apple Pay, will most likely contribute to the widespread adoption of M2M financial activities.

Smart home systems have also incorporated M2M technology. The use of M2M in this embedded system enables home appliances and other technologies to have real time control of operations as well as the ability to remotely communicate.

M2M is also an important aspect of remote-control software, robotics, traffic control, security, logistics and fleet management and automotive.

Key features of M2M

Key features of M2M technology include:

- Low power consumption, in an effort to improve the system's ability to effectively service M2M applications.
- A Network operator that provides packet-switched service
- Monitoring abilities that provide functionality to detect events.
- Time tolerance, meaning data transfers can be delayed.
- Time control, meaning data can only be sent or received at specific predetermined periods.
- Location specific triggers that alert or wake up devices when they enter particular areas.
- The ability to continually send and receive small amounts of data.

M2M requirements

According to the European Telecommunications Standards Institute (ETSI), requirements of an M2M system include:

- Scalability - The M2M system should be able to continue to function efficiently as more connected objects are added.

- Anonymity - The M2M system must be able to hide the identity of an M2M device when requested, subject to regulatory requirements.
- Logging - M2M systems must support the recording of important events, such as failed installation attempts, service not operating or the occurrence of faulty information. The logs should be available by request.
- M2M application communication principles - M2M systems should enable communication between M2M applications in the network and the M2M device or gateway using communication techniques, such as short message service (SMS) and IP Connected devices should also be able to communicate with each other in a peer-to-peer (P2P) manner.
- Delivery methods - The M2M system should support Unicast, anycast, multicast and broadcast communication modes, with broadcast being replaced by multicast or anycast whenever possible to minimize the load on the communication network.
- Message transmission scheduling - M2M systems must be able to control network access and messaging schedules and should be conscious of M2M applications' scheduling delay tolerance.
- Message communication path selection - Optimization of the message communication paths within an M2M system must be possible and based on policies like transmission failures, delays when other paths exist and network costs.

1.7.3 M2M Gateway

- Since non-IP based protocols are used within M2M area networks, the M2M nodes within one network cannot communicate with nodes in an external network.
- To enable the communication between remote M2M area networks, M2M gateways are used.

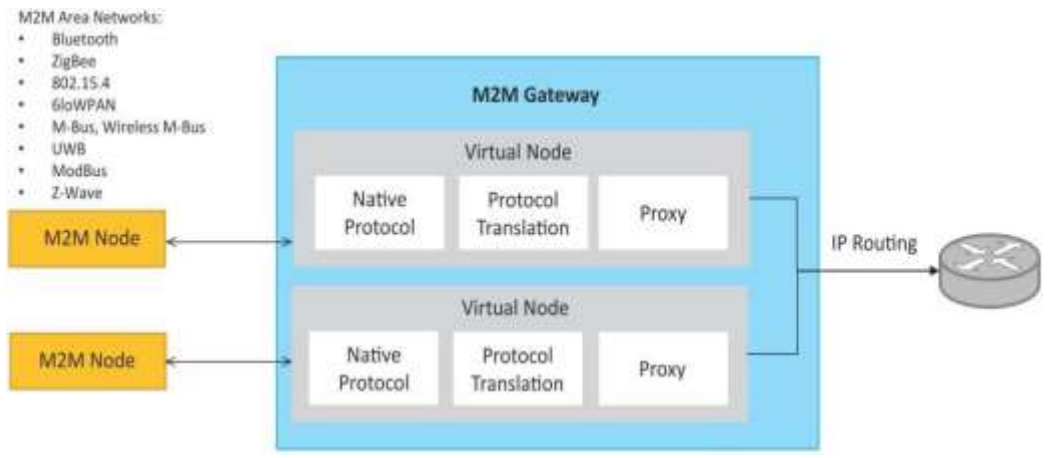


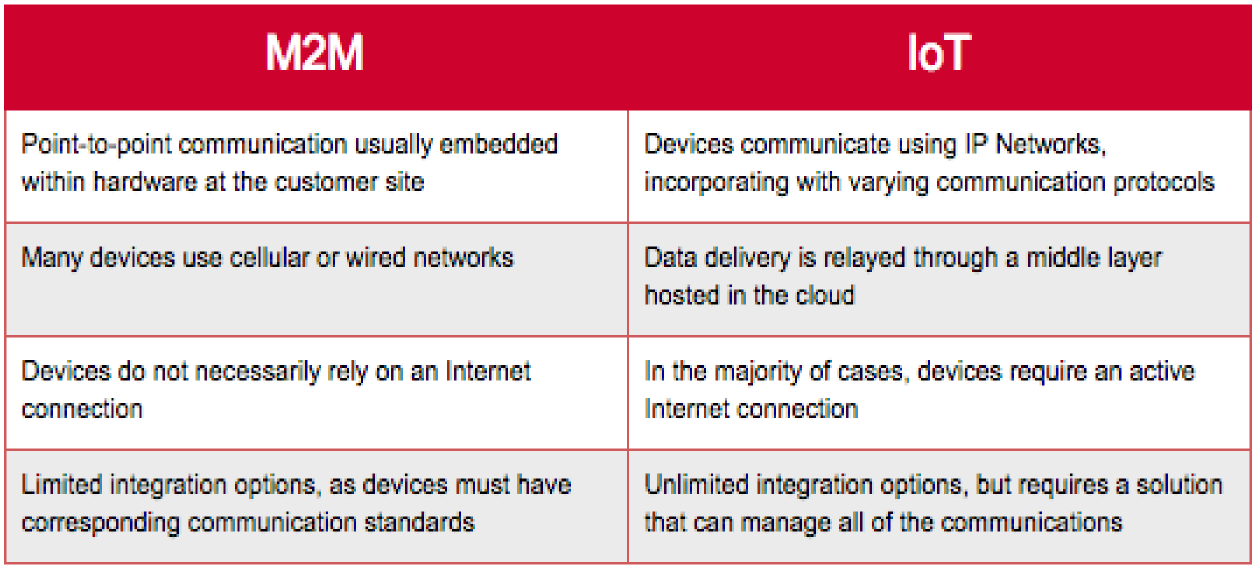
Fig 9 M2M Gateway

1.7.4 M2M vs. IoT

While many use the terms interchangeably, M2M and IoT are not the same. IoT needs M2M, but M2M does not need IoT.

Both terms relate to the communication of connected devices, but M2M systems are often isolated, stand-alone networked equipment. IoT systems take M2M to the next level, bringing together disparate systems into one large, connected ecosystem.

M2M systems use point-to-point communications between machines, sensors and hardware over cellular or wired networks, while IoT systems rely on IP-based networks to send data collected from IoT-connected devices to gateways, the cloud or middleware platforms.



M2M	IoT
Point-to-point communication usually embedded within hardware at the customer site	Devices communicate using IP Networks, incorporating with varying communication protocols
Many devices use cellular or wired networks	Data delivery is relayed through a middle layer hosted in the cloud
Devices do not necessarily rely on an Internet connection	In the majority of cases, devices require an active Internet connection
Limited integration options, as devices must have corresponding communication standards	Unlimited integration options, but requires a solution that can manage all of the communications

Fig 9 M2M VS IOT

Data collected from M2M devices is used by service management applications, whereas IoT data is often integrated with enterprise systems to improve business performance across multiple groups. Another way to look at it is that M2M affects how businesses operate, while IoT does this and affects end users.

For example, in the product restocking example above, M2M involves the vending machine communicating to the distributor's machines that a refill is needed. Incorporate IoT and an additional layer of analytics is performed; the vending machine can predict when particular products will need refilling based on purchase behaviors, offering users a more personalized experience.

1.7.5 M2M security

Machine-to-machine systems face a number of security issues, from unauthorized access to wireless intrusion to device hacking. Physical security, privacy, fraud and the exposure of mission-critical applications must also be considered.

Typical M2M security measures include making devices and machines tamper-resistant, embedding security into the machines, ensuring communication security through encryption and securing back-end servers, among others. Segmenting M2M devices onto their own network and managing device identity, data confidentiality and device availability can also help combat M2M security risks.

1.7.6 M2M standards

Machine-to-machine technology does not have a standardized device platform, and many M2M systems are built to be task- or device-specific. Several key M2M standards, many of which are also used in IoT settings, have emerged over the years, including:

- OMA DM (Open Mobile Alliance Device Management), a device management protocol
- OMA LightweightM2M, a device management protocol
- MQTT, a messaging protocol
- TR-069 (Technical Report 069), an application layer protocol
- HyperCat, a data discovery protocol
- OneM2M, a communications protocol
- Google Thread, a wireless mesh protocol
- AllJoyn, an open source software framework

1.7.7 Concerns about M2M

The major concerns surrounding M2M are all related to security. M2M devices are expected to operate without human direction. This increases the potential of security threats, such as hacking, data breaches and unauthorized monitoring. In order to repair itself after malicious attacks or faults, an M2M system must allow remote management, like firmware updates.

The necessity of remote management also becomes a concern when considering the length of time M2M technology spends deployed. The ability to service mobile M2M equipment becomes unrealistic since it is impossible to send personnel to work on them.

The inability to properly service the M2M equipment creates various unique security vulnerabilities for the M2M systems and the wireless networks they use to communicate.

1.7.8 History of M2M

While the origins of the acronym are unverified, the first use of machine-to-machine communication is often credited to Theodore Paraskevakos, who invented and patented technology related to the transmission of data over telephone lines, the basis for modern-day caller ID.Nokia was one of the first companies to use the acronym in the late 1990s. In 2002, it partnered with Opto 22 to offer M2M wireless communication services to its customers.

In 2003, M2M Magazine launched. The publication has since defined the six pillars of M2M as remote monitoring, RFID, sensor networking, smart services, telematics and telemetry.

1.8. M2M towards IOT

M2M solutions have been around for decades and are quite common in many different scenarios. While the need to remotely monitor and control assets personal, enterprise or other is not new, a number of concurrent things are now converging to create drivers for change not just with in the technology industry, but within the wider global economy and society. Our planet is facing massive challenges environmental, social, and economic.The changes that humanity needs to deal with in the coming decades are unprecedented, not because similar things have not happened before during our common history on this planet, but because many of them are happening at the same time. From constraints on natural resources to a reconfiguration of the world’s economy, many people are looking to technology to assist with these issues. Essentially, therefore, a set of megatrends are combining to create needs and capabilities, which in turn produce a set of IoT Technology and Business Drivers. A megatrend is a pattern or trend that will have a fundamental and global impact on society at a macro level over several generations. It is something that will have a significant impact on the world in the foreseeable future. We here imply both game changers as challenges, as well as technology and science to meet these challenges.

The game changers come from a set of social, economic, and environmental shifts that create pressure for solutions to address issues and problems, but also opportunities to reformulate the manner in which our world faces them. There is an extremely strong emerging demand for monitoring, controlling, and understanding the physical world, and the game changers are working in conjunction with technological and scientific advances. The transition from M2M towards IoT is one of the key facets of the technology evolution required to face these challenges. We outline some of these more globally significant game changers below, and their relationship to IoT:

• Natural Resource Constraints. The world needs to increasingly do more with less, from raw materials to energy, water or food, the growing global population and associated economic growth demands put increasing constraints on the use of resources. The use of IoT to increase yields, improve productivity, and decrease loss across global supply chains is therefore escalating.

• Economic Shifts. The overall economy is in a state of flux as it moves from the post-industrial era to a digital economy. One example of this is found in the move from product-oriented to service-oriented

economies. This implies a lifetime responsibility of the product used in the service offering, and will in many cases require the products to be connected and contain embedded technologies for gathering data and information. At the same time, there are fluctuations in global economic leadership. Economies across the globe must handle the evolving nature of these forces. As technology becomes increasingly

embedded and more tasks automated, countries need to manage this shift and ensure that M2M and IoT also create new jobs and industries.

• Changing Demographics. With increased prosperity, there will be a shift in the demographic structures around the world. Many countries will need to deal with an aging population without increasing economic expenditure. As a result, IoT will need to be used, for example, to help provide assisted living and reduce costs in healthcare and emerging “wellcare” systems.

• Socioeconomic Expectations. The global emerging middle-class results in increasing expectations on well-being and Corporate Social Responsibility. Lifestyle and convenience will be increasingly enabled by technology as the same disruption and efficiency practices evident in industries will be applied within people's lives and homes as well.

• Climate Change and Environmental Impacts. The impact of human activities on the environment and climate has been long debated, but is now in essence scientifically proven. Technology, including IoT, will need to be applied to aggressively reduce the impact of human activity on the earth's systems.

• Safety and Security. Public safety and national security become more urgent as society becomes more advanced, but also more vulnerable. This has to do both with reducing fatalities and health as well as crime prevention, and different technologies can address a number of the

issues at hand. Urbanization. We see the dramatic increase in urban populations and discussions about megacities. Urbanization creates an entirely new level of demands on city infrastructures in order to support increasing urban populations. IoT technologies will play a central role in the optimization for citizens and enterprises within the urban realm, as well as providing increased support for decision-makers in cities.

1.9. IOT Value chain

1.9.1 Global value chains

A value chain describes the full range of activities that firms and workers perform to bring a product from its conception to end use and beyond, including design, production, marketing, distribution, and support to the final consumer
Analyzing an industry from a global value chain (GVC) perspective permits understanding of the context of globalization on the activities contained within them by "focusing on the sequences of tangible and intangible value-adding activities, from conception and production to end use. GVC analysis therefore provides a holistic view of global industries both from the top down and from the bottom up".

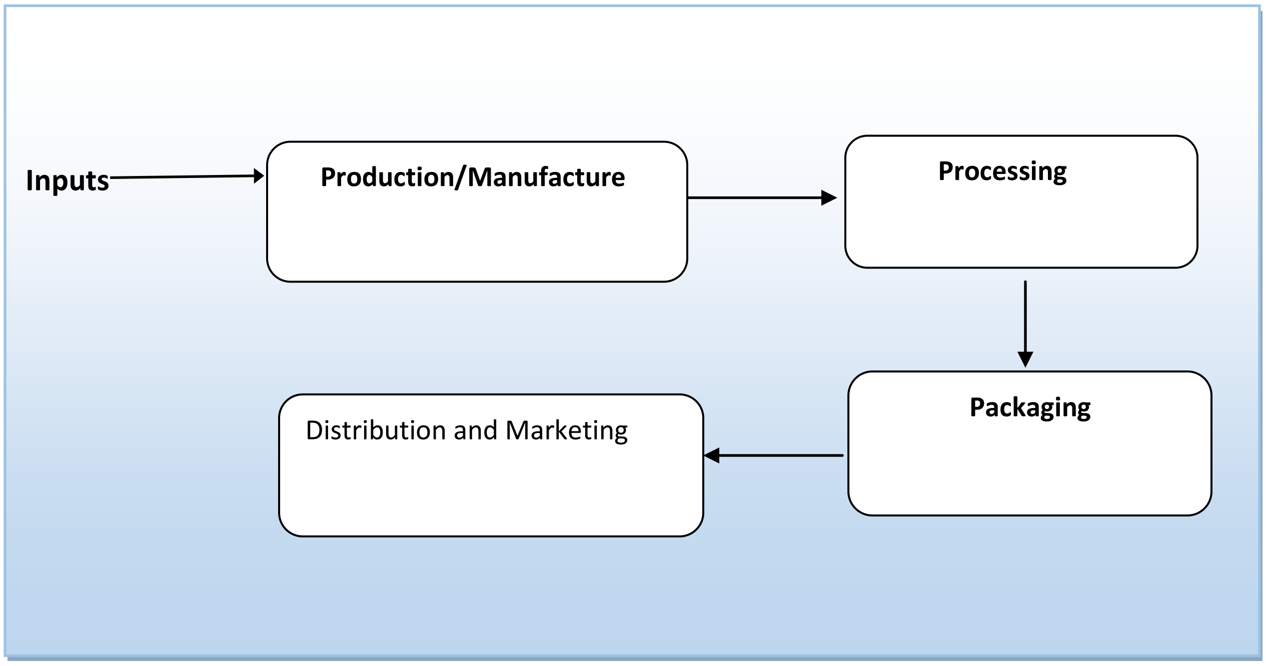


Fig 10 M2M value chain

1.9.2 Global value Example

Input	Input are the base raw ingredients that turned into a product
Example	Cocoa beans for the manufacture of chocolate
M2M Example	Data from a M2M device that will be turned into a piece of information
Production/Manufacture	Raw ingredients put through to become part of value
Example	Cocoa beans may be dried and separated before being transported to overseas market
M2M Example	Data from an M2M device that will be turned into a piece of information
Processing	Processing refers to the process that the raw inputs are put through become a part of value chain
Example	Cocoa beans now be made into cocoa powder ready for use in chocolate bars
M2M Example	M2M refers to aggregation of multiple data sources to create a

	information component something that is ready to combined with other data sets to make it useful for corporate decision making
Packaging	Packaging refers to the process where by a product can be branded as would be recognizable to end user consumers
Example	A chocolate bar would now be ready to eat and have a red wrapper with the words KitKat on it
M2M Example	M2M data will have to be combined with other information from internal corporate databased for example to see whether the data received requires any action. This data would be recognizable to the end users that need to use the information, either in the form of visualizations or an Excel spreadsheet
Distribution/Marketing	This process refers to the channels to market for products
Example	A chocolate bar may be sold at a supermarket a kiosk or even online
M2M Example	Will have produced an Information product that can be used to create new knowledge within a corporate environment examples include more detailed scheduling of maintenance based on real world information or improved product design due to feedback from the M2M solution.

1.9.3 IoT value chains example

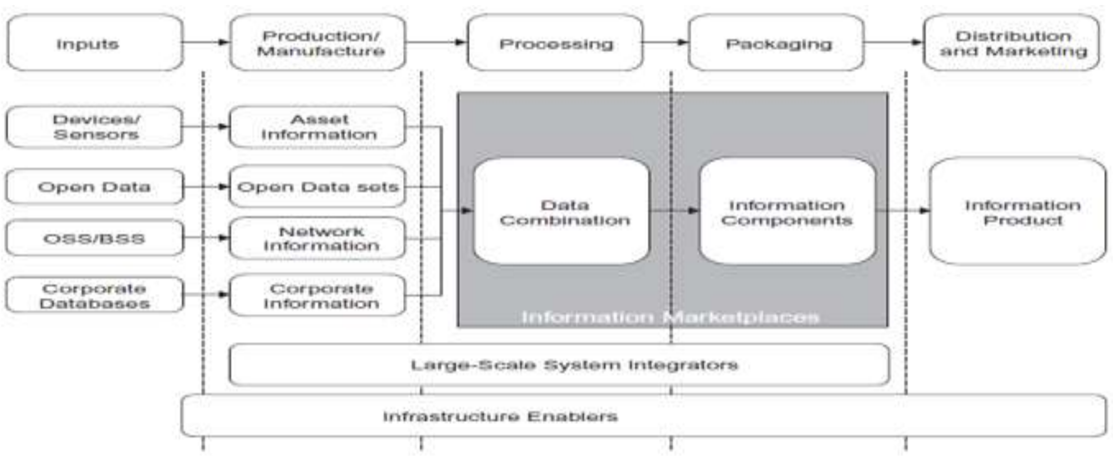


Fig.11 IoT value chain

IoT value chains example

Input	Significantly more inputs than for an M2M solution
Data sensors	Data from devices and sensors is used to provide a different and much broader marketplace then M2M does
Open data	A piece of data is open if anyone is free to use,resuse and redistribute it subject only at most the requirements of attribute and/or share alike Example: city maps, provided by organizations such as ordinance survey in the united kingdom
OSS/BSS	The operational support system (OSS) and Business support system closed information marketplace that allow operators to delivers services to enterprises. Example: where phone usage data is already owned by the company.

Corporate database	Companies of a certain size generally have multiple corporate database covering various functions, including supply chain management, payroll ,accounting .All the use of devices and sensors increases these database will be connected to this data to create new information sources and new knowledge.
Production/Manufacturer	Process will need to include tagging and linking of relevant data items in order to provide provenance and traceability across the information value chain
Asset Information	Asset information may include data such as temperature over time of container during transit or air quality during a particular month
Open data set	Maps, rail timetables or demographics about a certain area in a country or city
Network information	GPS data,services accessed via the mobile network
Corporate information	The current state of demand for a particular product in the supply chain at a particular moment in time
Processing	The data from the various inputs from the production and manufacture stage are combined together to create information
Packaging	The final stage of the information value chain is the creation of an information product
Distributed Marketing	The packaging section of the information value chain creates information components. These components could be produced as charts or other traditional methods of communicating information to end users

Information products for improving decision making	These information products are the results of ether detailed information analysis that allows better decision to be made during various internal corporate process, or they enable the creation of previously unavailable knowledge about company’s product, strategy or internal processes
Information product for resale to other economic actors	These information products have high value for other economic actors and can be sold to them

1.9.4 The Information-Driven Global Value Chain(I-GVC)

- Fundamental roles within the I-GVC
- Inputs

  Sensors, RFID, and other devices,End-Users.

- Data Factories.
- Service Providers/Data Wholesalers.
- Intermediaries.
- Resellers.

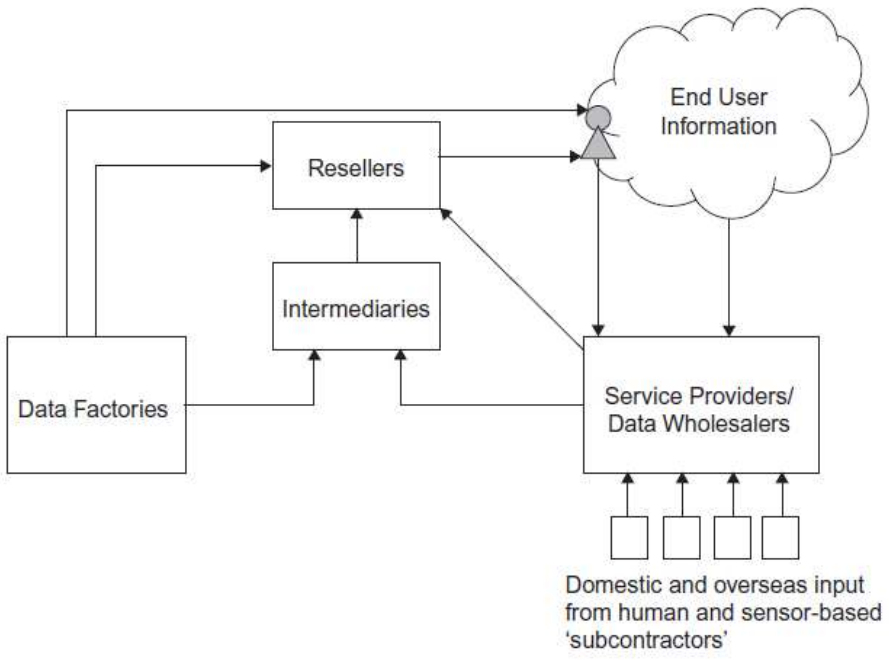

End User Information
Resellers
Intermediaries
Data Factories
Service Providers/ Data Wholesalers
Domestic and overseas input from human and sensor-based 'subcontractors'

Fig 12 The Information-Driven Global Value Chain.

1.9.5 Inputs to the information-driven global commodity chain

- Sensors and other devices (e.g. RFID and NFC).
- End-users

  Both of these information sources input tiny amounts of data into the I-GVC chain, which are then aggregated, analyzed, repackaged, and exchanged from value chain. As a result, sensor devices and networks, RFIDs, mobile and consumer devices, Wi-Fi hotspots, and end-users all form part of a network of "subcontractors" in the value chain, all contributing to the increased value of the information products

1.9.6 Sensors and radio frequency identification

- Sensors and RFID devices are working as inputs to the I-GVC through the capture and transmission of data necessary for the development of information products.
- Smart phones have also been developed that allow mobile devices to interact with sensors and RFID.

- This allows for a two-way interaction between a mobile terminal and the sensor technology.
- The data is used as one part of the input to the commodity chain, which uses it to create the information products that are eventually exchanged.
- In this sense, the sensor networks, and NFC and RFID technologies may be viewed as subcontractors Resellers Intermediaries Data Factories Service Providers/ Data Wholesalers End User Information Domestic and overseas input from human and sensor-based ‘subcontractors’ to the I-GVC, workers that constantly gather data for further processing and sale.

1.9.7 End-users

- End-users that choose to use and participate within the digital world are now deeply embedded into the very process of production.
- Every human that enters a search query into a search engine,
- Every human that agrees to allow the mobile broadband platform to inform a service of their location,
- Every human that uses NFC to allow a bank to establish and confirm their identity are also functioning as subcontractors to the global information systems that form the basis of the I-GVC.
- Production processes of the information – driven global value chain
- Data factories
- Data factories are those entities that produce data in digital forms for use in other parts of the I-GVC.
- Service providers/data wholesaler:
- Service Providers and Data wholesalers are those entities that collect data from various sources worldwide, and through the creation of massive databases, use it to either improve their own information products or sell information products in various forms.
- Example: Twitter, Facebook, Google

1.9.8 Intermediaries

- In the emerging industrial structure of the I-GVC, there is a need for intermediaries that handle several aspects of the production of information products.
- The development of databases such as the ones created by Google, Facebook, and Twitter may therefore require the creation of entities that are able to protect individuals' privacy rights in relevant regional settings.
- Intermediary is to reduce transaction costs associated with the establishment of a market for many different companies to participate in.

1.9.9 Resellers

- Resellers are those entities that combine inputs from several different intermediaries, combine it together, analyze, and sell it to either end-users or to corporate entities.
- These resellers are currently rather limited in terms of the data that they are able to easily access via the converged communications platform, but they are indicative of the types of corporate entities that are forming within this space.

1.10 An emerging industrial structure for IoT.

1. Meet key societal needs for the Internet of Things including open governance, security, privacy and trustworthiness.

The Internet of Things should not be owned by single interest groups, as it should be an open global infrastructure as the Internet and WWW are today. One of the key issues in Europe and Asia in the past years has been the predominance of VeriSign, an American company operating the Object Name Service (ONS) under contract for the EPCglobal Network (Clendenin 2006, Heise online 2008). Federated structures are needed to provide a power balance. Security, privacy and trustworthiness need to be considered, but are in most aspects not specific to the Internet of Things. The same technologies that have been successfully used in the Internet can be utilised in the Internet of Things as well, although there are some specific challenges due to characteristics of the Internet of Things application scenarios, which often include mobile or portable objects that change custody or ownership during their lifetimes. However, there is a difference in the Auto-ID, sensor and actuator part, where different attacks on the network are possible. Nevertheless, it has to be remembered that the highest achievable security level is not always required. There are for example different levels of security required for passports or logistic applications.

2. Bridge the gap between B2B, business-to-consumer (B2C) and machineto-machine (M2M) requirements through a generic and open Internet of Things infrastructure.

While there has been a clear focus on B2B requirements in the last years, B2C and M2M will gain importance in the future Internet of Things. While in B2C ease of use as well as human readable data are important, in M2M communications, the data should be machine-readable structured and semantically well-defined.

3. Design an open, scalable, flexible and sustainable infrastructure for the Internet of Things.

The Internet of Things has to be open by definition. Open standards are required to use and extend its functionality. It will be a huge network, considering that every object has its virtual representation. Therefore, scalability is required. The Internet of Things will need to be flexible enough to adapt to changing requirements and technological developments. Its development can be accelerated through the availability of open source software, such as Fosstrak17

4. Develop migration paths for disruptive technological developments to the Internet of Things.

Rather than requiring disruptive new and parallel approaches, there have to be means of integrating new developments into the fundamental infrastructure, otherwise there can be no guarantee of sustainability or enduring value. Examples include autonomous objects that do not essentially require a networked infrastructure. Nevertheless, providing a migration path for autonomous control in the Internet of Things would broaden its usage and provide a solid networked infrastructure for autonomous objects (Uckelmann et al. 2010). to allow anyone to implement and test new functionalities. Another opportunity to experiment and test new functionalities are living lab initiatives, where service providers and users participate in a collaborative environment. Finally, it needs a sustainable infrastructure to provide a basis for the necessary investments.

5. Excite and enable businesses and people to contribute to the Internet of Things.

If stakeholders cannot benefit from the Internet of Things, they will not participate. In contrast, any user benefiting from the Internet of Things will attract and excite more participants. Research on how to benefit from the Internet of Things is needed. Business needs to see a clear business case. End-users need to find a personal benefit. Funded research, such as that described in section 1.3, can trigger new ideas and stakeholders, but in a longer view benefits have to be generated from within the network and not through external funds.

6. Enable businesses across different industries to develop high added value products and services.

New business models (both industry-specific and cross-sector) are required based on retrieving and contributing information to/from the Internet of Things. Researchers can help to identify new potentials but business entrepreneurs are needed to actually raise the potential of the Internet of Things.

7. Encourage new market entrants, such as third party service and information providers, to enter the Internet of Things. Information in the Internet of Things can be accumulated, processed and sold independently of owning the physical product. Service providers should be encouraged for example to provide access to multiple sources of information about things and adding technical billing capabilities for information access.

8**. Provide an open solution for sharing costs, benefits and revenue generation in the Internet of Things.** Information should be freely tradable, irrespective of the physical product. Today, wider usage of the Internet of Things is most often hindered by missing concepts on human, organisational and technical shortcomings to share cost and benefits, or even generate revenue from the Internet of Things.

9. Public initiatives to support the usage of the Internet of Things for social relevant topics. Legislation has always been a push mechanism for adoption of new technologies.

While it is obvious that the Internet of Things can be used to provide society with relevant data, some legislative requirements on topics such as carbon footprint, green logistics, and animal welfare would help to show the utility of the Internet of Things for society.

10. Enable people to seamlessly identify things to access as well as contribute related information.

How many people carry an Auto-ID reader all day to identify objects and access corresponding information? Mobile phones today already include a camera that can scan barcodes and 2D matrix symbologies. Near Field Communication (NFC) is expected to be the next logical step for user interaction with the Internet of Things. However, it is questionable how many mobile phone owners will use these technologies. Besides mobile phones, there may be cheap dedicated devices. Nabaztag18 These ten key requirements are not intended to provide a complete set of requirements. They are meant to focus on certain aspects of the Internet of Things to start a rethinking process for future developments. provides a set including reader, tags and internet-based applications for about 40 Euro. Mobile barcode scanners and RFID readers that can be attached to a key chain and that are as easy to operate as a USB-stick are yet another opportunity to enable mass participation in the Internet of Things

1.11 A Possible Architecture for the Future Internet of Things

While it is quite obvious that there are and will be numerous approaches towards the Internet of Things, thus leading to a creative variety of applications in the Internet of

Things, we favour an architectural approach that is based on extensions to a successful standardised open architecture – the EPCglobal Network. The EPCglobal Network is widely accepted and has gained the biggest support from IT companies that have adopted the standardised interfaces into their own applications. Numerous products have been developed and certified (EPCglobal 2010). Therefore, the EPCglobal Network provides a solid foundation, despite the fact that it is still under development. However, the Internet of Things requires a more holistic architecture as described before. This can build on the same design principles as the EPCglobal Architecture Framework (EPCglobal 2007). These include layering of standards, separation of data models and interfaces, provision of extension mechanisms, specification of data models and interfaces, initially in a neutral abstract manner (e.g., using UML), then with provision of specific transport bindings (e.g., web services) and schema bindings (e.g., XML). A future Internet of Things has to integrate stakeholders who will be affected by the Internet of Things, such as citizens, small and medium enterprises, governmental institutions and policy makers, to meet and match key societal and economic needs. Applications that recognise and improve the fundamental qualities of life for users, businesses, society and the environment are needed. The foundation will need to provide open architectures, protocols and technologies for new classes of smart Internet-/Web-based public and business applications. Social platforms to share experience and personalised insights will be integrated with business-centric applications. Discovery and retrieval of useful and relevant information beyond personal expectations will be achieved though engineering for serendipity. Users shall be empowered to access more information about things (e.g., Where has an item been produced? – Who owned it previously? - What was it used for?) instantly at their fingertips, subject to compliance with privacy regulations. Mash-ups and end-user programming will enable people to contribute to the Internet of Things with data, presentation and functionality. Things-generated ‘physical world’ content from Auto-ID, sensors, actuators or meshed networks shall be aggregated and combined with information and events from ‘virtual worlds’, such as business databases and social platforms, and processed based on new business intelligence concepts. Results will be displayed in a user-centred design, including intuitive interfaces and Web 2.0 functionalities. Direct action on the physical world will be supported through Internet of

Things machine-interfaces and introduction of agile strategies. Buying decisions will be supported through the access to relevant information as needed. Agile strategies in this context refer to real-time management and execution capability under consideration of conflicting optimisation values (e.g., shipment size). Information sharing will be rewarded through incentives, including transparent, open billing interfaces between numerous stakeholders, thus transforming the Internet of Things from a cost-focused infrastructure to a benefit-focused infrastructure to accelerate business innovation. Distributed data ownership across the object life cycle will be addressed by integrated billing. Information will be as easily tradable as products and services. The gap between distributed intelligence concepts (e.g., autonomous logistics) and the Internet of Things will be overcome through integration of open interfaces, protocols and lookup services as well as information services on mobile devices, acting as a mediator among decentralised information systems. Openness, scalability and security will be addressed as an integral part of the core architecture. Openness includes social (e.g., governance, privacy), organisational (e.g., industries) and technical (e.g., infrastructures, identifiers) dimensions. The integration and interoperability with mainstream business software platforms will be enhanced and its functionality will be extended through real-time analytics and business intelligence.

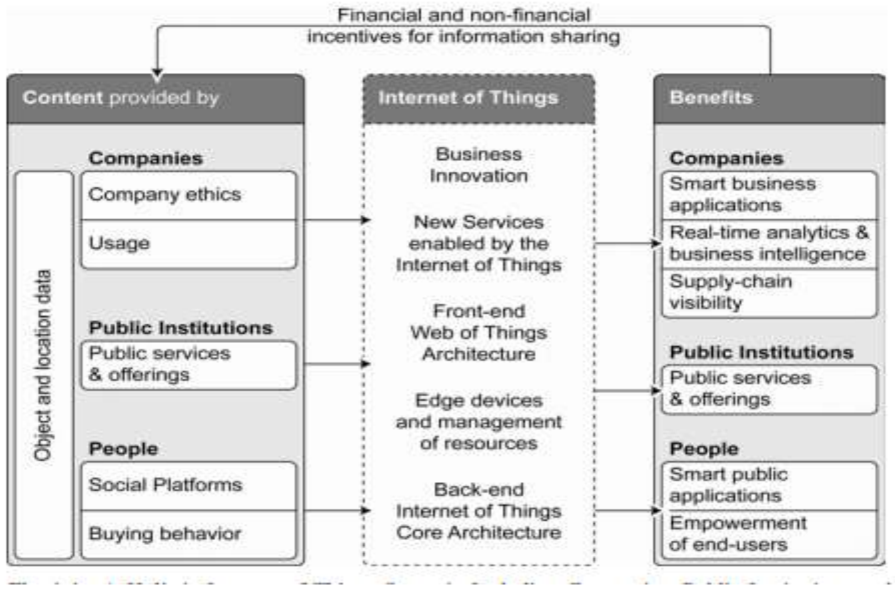

Financial and non-financial incentives for information sharing
Content provided by
Object and location data
Companies
Company ethics
Usage
Public Institutions
Public services & offerings
People
Social Platforms
Buying behavior
Internet of Things
Business Innovation
New Services enabled by the Internet of Things
Front-end Web of Things Architecture
Edge devices and management of resources
Back-end Internet of Things Core Architecture
Benefits
Companies
Smart business applications
Real-time analytics & business intelligence
Supply-chain visibility
Public Institutions
Public services & offerings
People
Smart public applications
Empowerment of end-users

Fig 13 Scenario for Future Internet of Things

Figure 13 shows one possible scenario that includes content providers (producers) and content users (consumers) that utilise the Internet of Things and share benefits. Company data includes for example product and usage data as well as company ethics that may influence buying behaviour. Public institutions as well **as** people will be able to contribute content. New services and business innovation will be enabled by an enhanced Internet of Things infrastructure including edge devices and back-end services as well as front-end user-interfaces. Companies, public institutions and people will be able to access data for their own benefits and financial as well as non-financial benefit compensation will further add to a fast adoption process of the Internet of Things. Key goals for a future Internet of Things architecture to achieve are: • An open, scalable, flexible and secure infrastructure for the Internet of Things and People • A user-centric, customisable 'Web of Things' including interaction possibilities for the benefit of society • New dynamic business concepts for the Internet of Things including flexible billing and incentive capabilities to promote information sharing

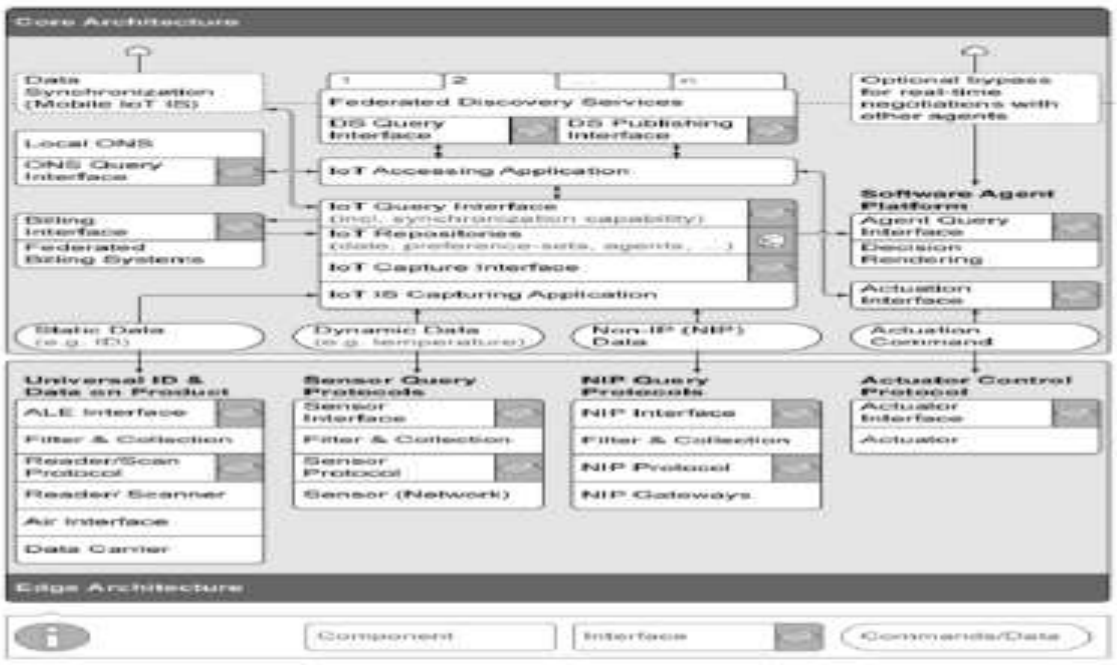


Fig 14 EPC Global Network architecture of IOT

The EPCglobal Network architecture is currently only one aspect of the broader Internet of Things. However, if openness, scalability and security can be assured, the EPCglobal

Network could be the most promising and comprehensive architecture in the Internet of Things. The availability of free, open standards and free open source implementations for the EPCglobal Network architecture may play a significant enabling role in its development, alongside complementary technologies and standards, such as Open Geospatial Consortium (OGC) Sensor Web Enablement. Other extensions, such as support for multiple identifier schemes, federated discovery services, actuator integration and software agents for decentralised data processing and decision rendering, could further extend the functionality of the EPCglobal Network. The vision of the future Internet of Things includes extended Internet of Things Information Services based on the EPC Information Services. The extensions are necessary to provide a broader support for other identifiers than the EPC, additional static and dynamic data, actuator support, software agent integration, integration of non-IP devices and offline-capabilities. In detail, the vision includes the following components:

• Extended static data support

The EPCglobal Network today is based on the EPC. The EPC is not a single identifier scheme but a framework supporting multiple identifier schemes including GS1 identifiers such as Serialised Global Trade Identification Number (SGTIN), Serial Shipping Container Code (SSCC), and Global Returnable Asset Identifier (GRAI).

This framework is not limited to GS1 identifiers; EPC formats are also defined for unique identifier constructs specified by the US Department of Defense. In principle, other approaches such as the Uniform Resource Names (URNs) could be used to support identifiers based on ISO 15962 and even identifiers based on Uniform Resource Locators (URLs) could be included, since they are a subset of Uniform Resource Identifiers (URIs). There is a need to support all things that carry a unique ID, because changing an established identifier scheme in an industry can cost millions of Euro and should be compared to the efforts involved for changing databases in the last millennium to make them year 2000 compliant. There have been and continue to be approaches to transform existing established identification schemes into a format that is compatible with the EPCglobal Network, as well as EPCglobal standards such as Tag Data Standard (TDS) and Tag Data Translation (TDT) that enable two-way translation between an EPC representation and an existing legacy representation. Additional structured data in barcodes (e.g., for best-before-date) may need to be supported to fully integrate existing optical identification techniques and to exploit the user memory capabilities of RFID tags, as well as facilitating stock rotation, product recalls, etc. An open, universal identifier translation framework would enable all things that carry a unique ID to be part of the Internet of Things. However, until everything carries a unique ID, the Internet of Things may also need to support objects identified by a classID (productID) and attributes.

• Integration of dynamic data

In order to bring the real and the virtual world closer together there is a need to sense environmental conditions as well as the status of devices. A standardized sensor interface to the Internet of Things would help to minimise costs and foster implementation. Sensors are key components of the next generation of internet services because they empower bottom-up interaction with things by enabling the gathering of information about their state or condition within the real world. The state of the things can be used to feed services at the infrastructure layer, transforming everyday things into true enablers of the Internet of Things. • Support for non-IP devices – Non-IP devices offer only limited capability. They can be integrated in the Internet of Things through gateways that take care of the computational overhead required to share physical devices over the Internet, while also providing advanced functionality that are not available on the devices themselves.

• **Integration of an actuator interface** – Actuator integration into the Internet of Things will allow standardised communication with machines executing decisions either rendered by humans or software-agents on their behalf. Actuators complement bidirectional interaction processes by providing the means for services and users to influence the state of things. The combination of sensors and actuators and their integration in the core Internet of Things infrastructure is an indispensable feature and needs to be considered at all layers of the architecture.

• **Optional integration of software agents** – The complexity of global supply networks will require more decentralised and automated decision making. Software-agents have been researched broadly but have not yet gained considerable acceptance in industries. The reason for this may be the lack of standardisation. A standardised interface in the Internet of Things would help to boost the usage of software agents. Smart objects in the Internet of Things need to execute intelligent algorithms to be able to discard irrelevant data, interact with other things in an efficient way, raise warnings about their state or the state of their environment, and take informed decisions and actions on behalf of human end-users to eliminate or assist control / management activities by humans. Additionally, software agents may help to increase scalability and robustness in the Internet of Things (Uckelmann et al. 2010). In a holistic scenario we imagine things to host a certain infrastructure subset of the Internet of Things. These things may not always be connected to the Internet. Therefore, we envision a certain degree of smart characteristics and autonomy.

• Extended, federated discovery services

The EPCglobal Network today does not yet provide ratified standards for federated discovery services, although a technical standard for discovery services is currently under development. At the time of writing, the only lookup service currently provided by EPCglobal is the ONS, which only holds class-level records pointing to authoritative information. This is currently operated under contract by VeriSign Corp. under the onsepc.com domain. The existing ONS implementation is distributed across multiple servers globally. Nevertheless, there are political concerns that the ONS is defined under the .com Top-Level-Domain, which is under the authority of the US Department of Commerce and that the ONS service is operated only by one American company. This has led to political discussions on governance in the Internet of Things, resulting in national focused approaches in China and Europe (Muguet 2009). Federated discovery services are needed to enable open governance, scalability and choice of lookup service in the Internet of Things.

• Data-synchronisation for offline support

The EPCglobal Network requires online connection to access data related to the identified product. In certain cases online-connectivity cannot be assured. Data-synchronisation is needed to support mobile scenarios and decentralised decision making.

• Interface to federated billing services

In order to enable competition between billing service providers, a standardised interface to these services is needed. This billing interface will enable balancing of costs and benefits as well as new business models and revenue generation opportunities for business and citizens based on micro-trading of information in the Internet of Things.

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 INFORMATION TECHNOLOGY

UNIT - II

SITA3008 - Internet of Things

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

2.1.1 Examples of actuators

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

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

We live in a World of Sensors. You can find different types of Sensors in our homes, offices, cars etc. working to make our lives easier by turning on the lights by detecting our presence, adjusting the room temperature, detect smoke or fire, make us delicious coffee, open garage doors as soon as our car is near the door and many other tasks.

The example we are talking about here is the Autopilot System in aircrafts. Almost all civilian and military aircrafts have the feature of Automatic Flight Control system or sometimes called as Autopilot. An Automatic Flight Control System consists of several sensors for various tasks like speed control, height, position, doors, obstacle, fuel and many more. A Computer takes data from all these sensors and processes them by comparing them with pre-designed values. The computer then provides control signal to different parts like engines, flaps, rudders etc. that help in a smooth flight.

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

S.No	Sensor	Applications	Technology
1.	Inertial sensors	Industrial machinery, automotive, human activity	MEMS and Gyroscope
2.	Speed Measuring Sensor	Industrial machinery, automotive, human activity	Magnetic, light
3.	Proximity sensor	Industrial machinery, automotive, human activity	Capacitive, Inductive, Magnetic, Light, Ultrasound
4.	Occupancy sensor	Home/office monitoring	PassiveIR, Ultrasound most common
5.	Temperature/humidity sensor	Home/office HVAC control, automotive, industrial	Solid state, thermocouple
6.	Light sensor	Home/office/industrial lighting control	Solid state, photocell, Photo resistor, photodiode
7.	Power (current) sensor	Home/office/industrial powermonitoring/control Technology	Coil (Faraday's law), Hall effect
8.	Air/fluid pressure sensor	Industrial monitoring/control, automotive, agriculture	Capacitive, Resistive
9.	Acoustic sensor	Industrial monitoring/control, human interface	Diaphragm condenser
10.	Strain sensor	Industrial monitoring/control, civil infrastructure	Resistive thin films

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

The next classification is based on conversion phenomenon i.e. the input and the output. Some of the common conversion phenomena are Photoelectric, Thermoelectric, Electrochemical, Electromagnetic, Thermo-optic, etc. The final classification of the sensors are Analog and Digital Sensors. Analog Sensors produce an analog output i.e. a continuous output signal with respect to the quantity being measured. Digital Sensors, in contrast to Analog Sensors, work with discrete or digital data. The data in digital sensors, which is used for conversion and transmission, is digital in nature.

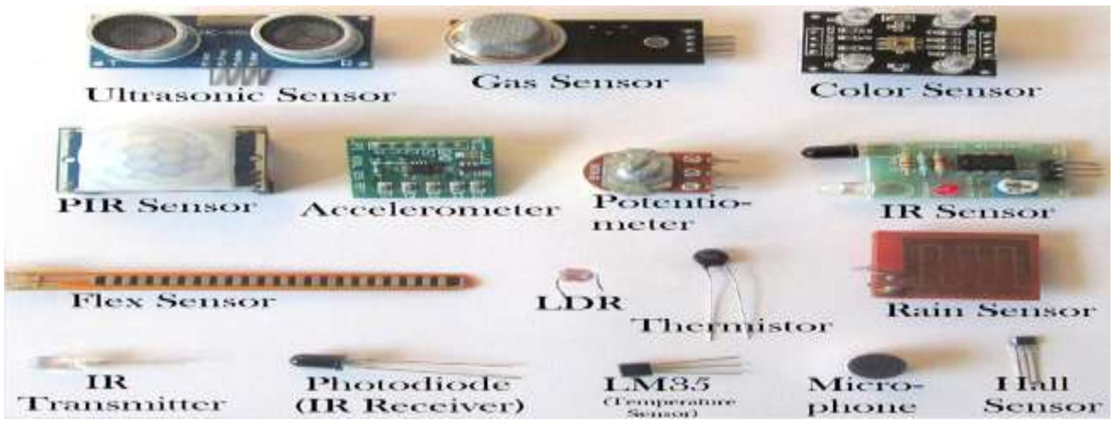


Fig 1. Examples of Sensors

2.1.2 IR LED

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

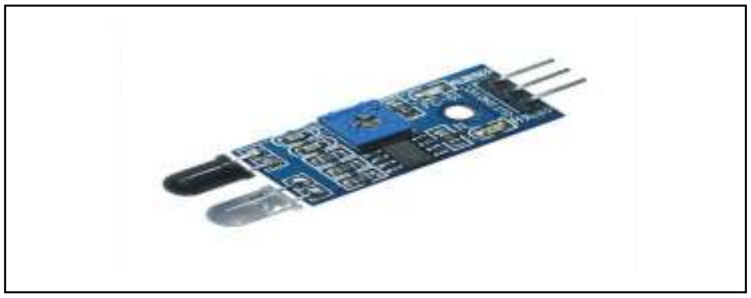


Fig 2. LED sensor

2.1.3 Photo Diode (Light Sensor)

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

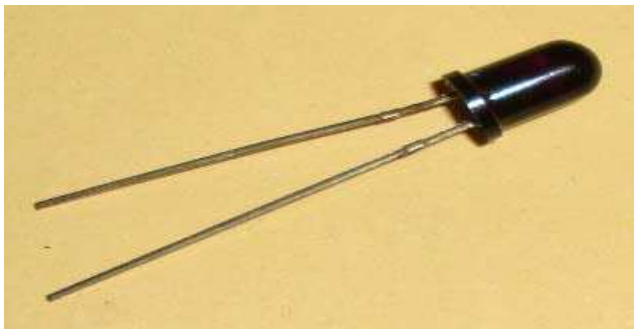


Fig 3.Photo diode

2.1.4 Proximity Sensor

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

Fig 4.Proximity sensor

Some of the applications of Proximity Sensors are Mobile Phones, Cars (Parking Sensors), industries (object alignment), Ground Proximity in Aircrafts, etc. Proximity Sensor in Reverse .

2.1.5 LDR (Light Dependent Resistor)

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

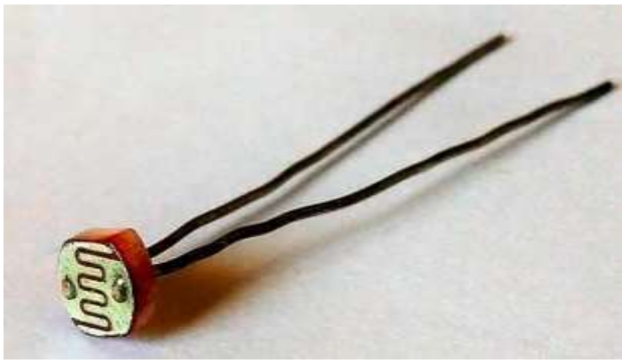


Fig 5.LDR

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

2.1.6 Thermistor (Temperature Sensor)

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

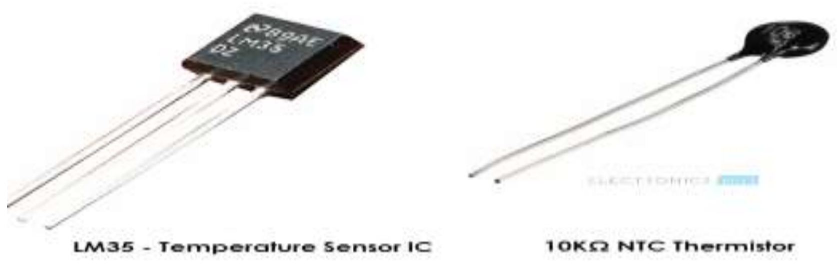


Fig 6.Thermistor

2.1.7 Thermocouple (Temperature Sensor)

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

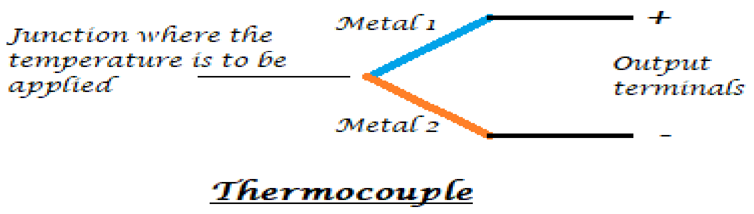


Fig 7. Thermo couple

2.1.8 Strain Gauge (Pressure/Force Sensor)

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

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

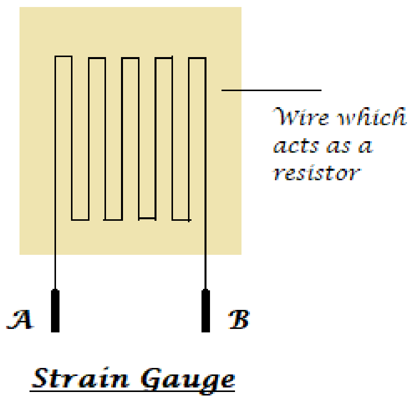


Fig 8.Strain Guage

2.1.9 Load Cell (Weight Sensor)

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

Fig 9.Load Cell

2.1.10 Potentiometer

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

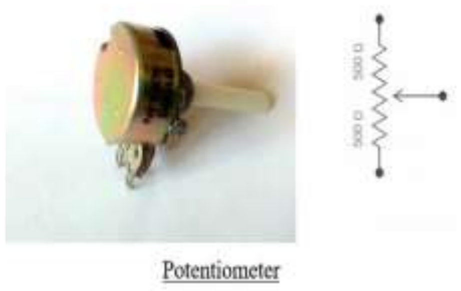


Fig 10.Potentiometer

2.1.11 Encoder

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

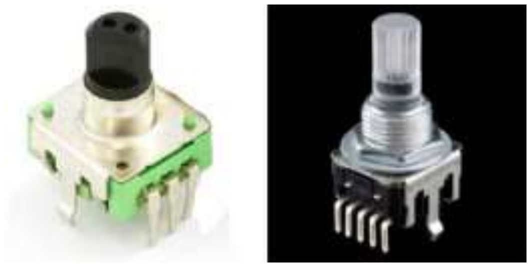


Fig 11.Encoder

2.1.12 Hall Sensor

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

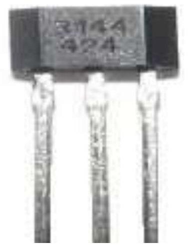


Fig 12.Hall sensor

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

2.1.13 Flex Sensor

A FLEX sensor is a transducer which changes its resistance when its shape is changed or when it is bent. A FLEX sensor is 2.2 inches long or of finger length. Simply speaking the sensor terminal resistance increases when it’s bent. This change in resistance can do no good unless we can read them. The controller at hand can only read the changes in voltage and nothing less, for this, we are going to use voltage divider circuit, with that we can derive the resistance change as a voltage change.

Fig 13. Flex sensor

2.1.14 Microphone (Sound Sensor)

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

Fig 14.Microphone

2.1.15 Ultrasonic sensor

Ultrasonic means nothing but the range of the frequencies. Its range is greater than audible range (>20 kHz) so even it is switched on we can't sense these sound signals. Only specific speakers and receivers can sense those ultrasonic waves. This ultrasonic sensor *is* used to calculate the distance between the ultrasonic transmitter and the target and also used to measure the velocity of the target**.**

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

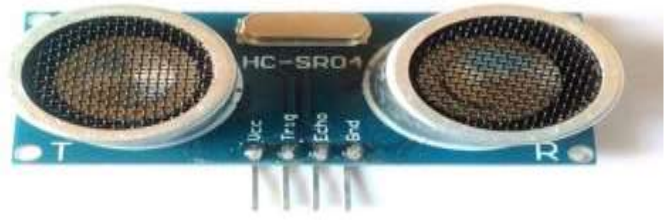


Fig 15.Utrasonic sensor

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

2.1.16 Touch Sensor

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

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

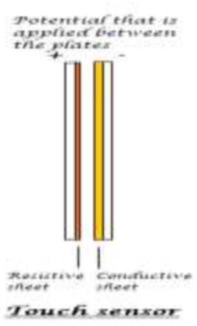


Fig 16.Touch sensor

2.1.17 PIR sensor

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

Fig 17.PIR Sensor

2.1.18 Accelerometer (Tilt Sensor)

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

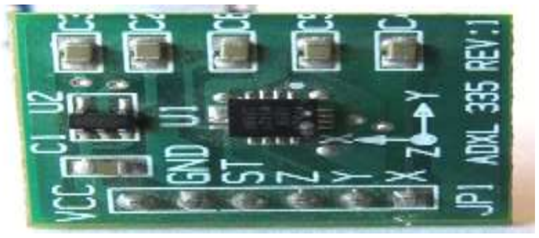


Fig 18.Accelerometer

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

2.1.19 Gas sensor

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

Fig 19.Gas Sensor

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

2.1.20 Resistive Sensors

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

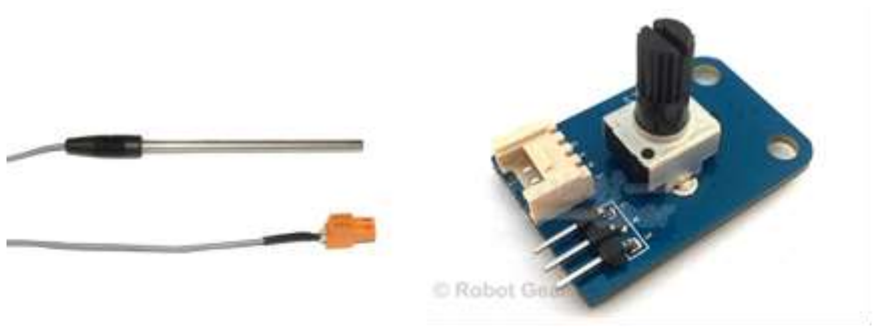


Fig 20 Resistive Sensors

2.1.21 Voltage generating sensors

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

.

Fig 21.Voltage Generating Sensors

2.1.22 Switch Sensors

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

Fig 22.Switch Sensors

2.2 Edge Networking

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

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

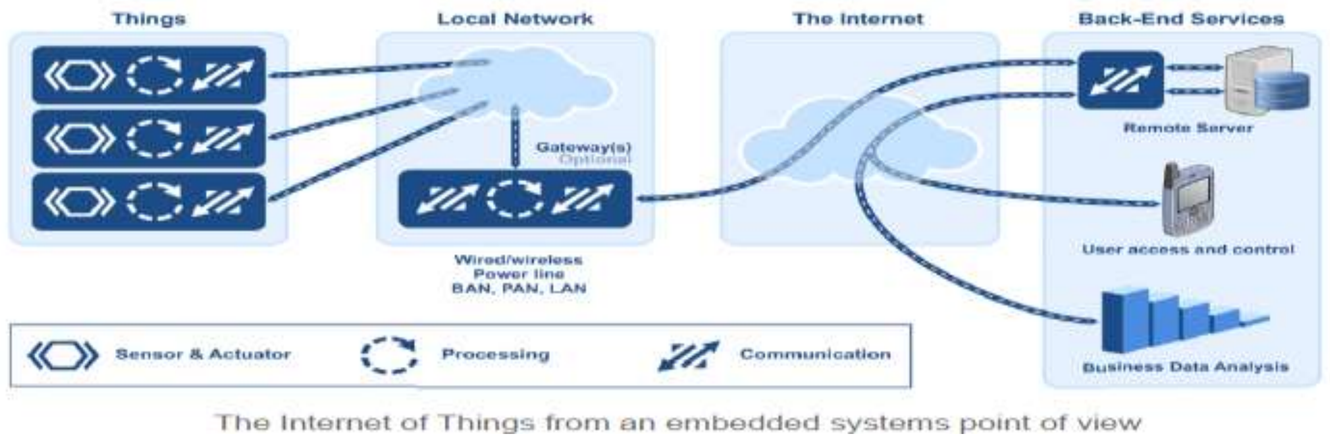


Fig 23.Embedded Point of View

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

.

1. Industrial IoT, where the local network is based on any one of many different technologies.

   The IoT device will typically be connected to an IP network to the global Internet.

2. Commercial IoT, where local communication is typically either Bluetooth or Ethernet (wired

   or wireless). The IoT device will typically communicate only with local devices.

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

2.2.1 Local Network

Your choice of communication technology directly affects your device's hardware requirements and costs. Which networking technology is the best choice?

IoT devices are deployed in so many different ways — in clothing, houses, buildings, campuses, factories, and even in your body — that no single networking technology can fit all bills.

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

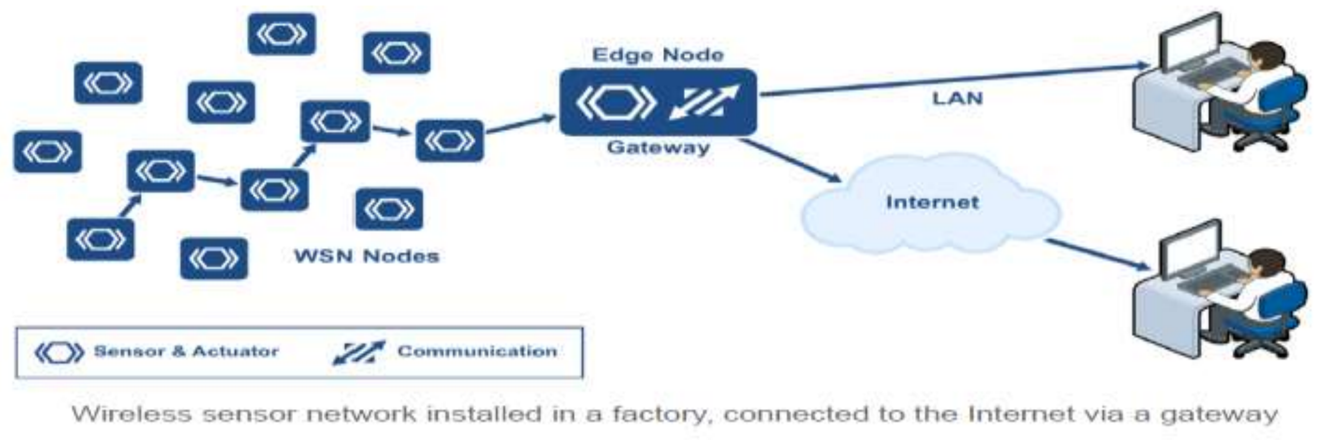

Edge Node
Gateway
LAN
Internet
WSN Nodes
Sensor & Actuator
Communication
Wireless sensor network installed in a factory, connected to the Internet via a gateway

Fig 24. Wireless Sensor Network Architecture

A wireless sensor network (WSN) is a collection of distributed sensors that monitor physical or environmental conditions, such as temperature, sound, and pressure. Data from each sensor passes through the network node-to-node.

2.2.2 WSN Nodes

WSN nodes are low cost devices, so they can be deployed in high volume. They also operate at low power so that they can run on battery, or even use energy harvesting. A WSN node is an embedded system that typically performs a single function (such as measuring temperature or pressure, or turning on a light or a motor).

Energy harvesting is a new technology that derives energy from external sources (for example, solar power, thermal energy, wind energy, electromagnetic radiation, kinetic energy, and more). The energy is captured and stored for use by small, low-power wireless autonomous devices, like the nodes on a WSN.

2.2.3 WSN Edge Nodes

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

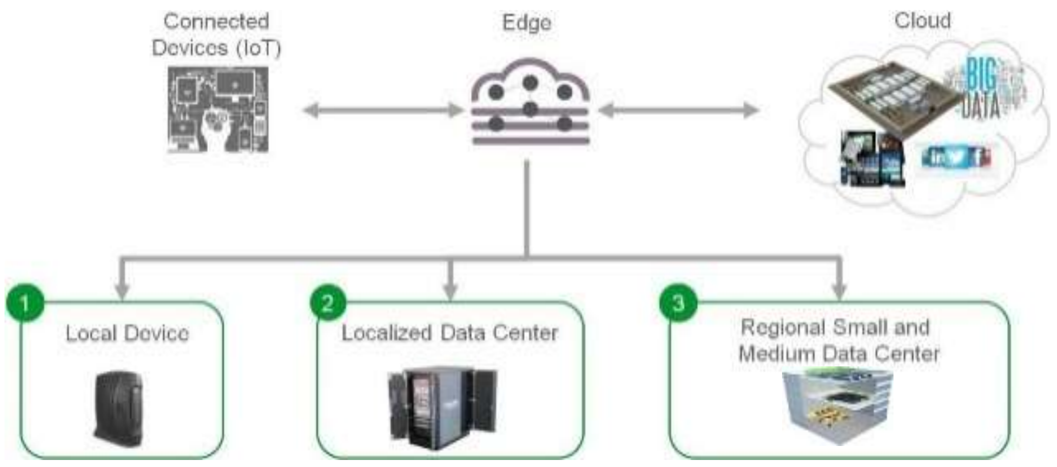


Fig 25.Edge Nodes

Fig 25.WSN Edge

2.2.4 Wi-Fi

The first obvious networking technology candidate for an IoT device is Wi-Fi, because it is so ubiquitous. Certainly, Wi-Fi can be a good solution for many applications. Almost every house that has an Internet connection has a Wi-Fi router. However, Wi-Fi needs a fair amount of power. There are myriad devices that can't afford that level of power: battery operated devices, for example, or sensors positioned in locations that are difficult to power from the grid.

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

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

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

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

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

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

2.3 Internet of Things Communications Models

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

2.3.1 Device-to-Device Communications

The device-to-device communication model represents two or more devices that

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

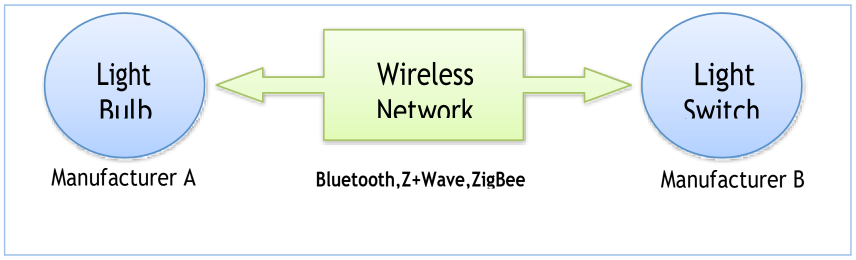


Fig 26. Example of device-to-device communication model

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

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

2.3.2 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. This is shown in Figure 27.

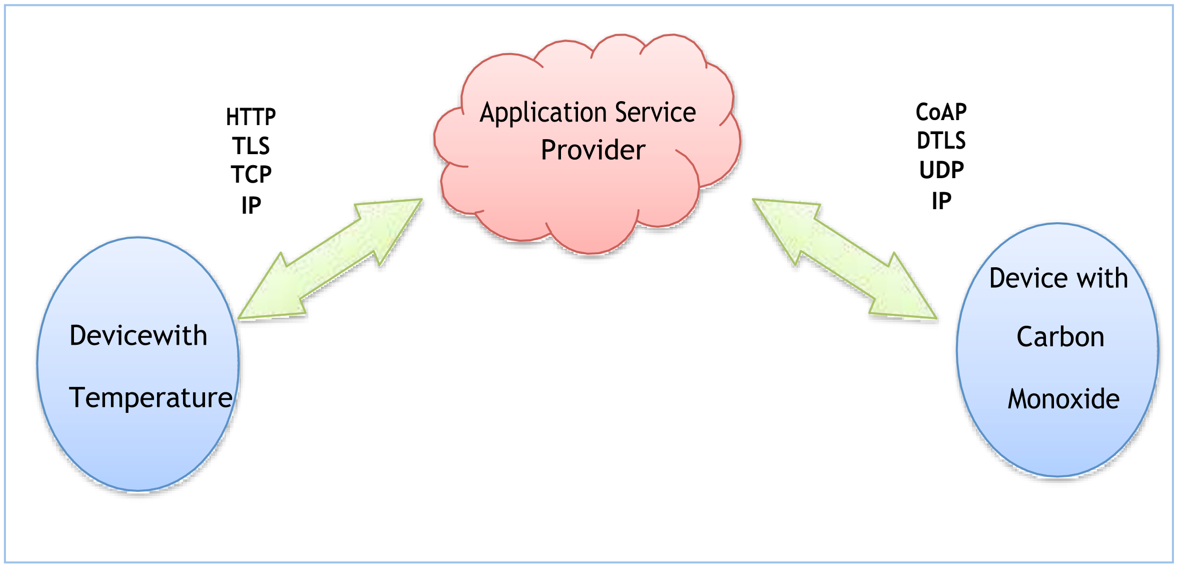


Fig 27.Device-to-cloud communication model diagram.

This communication model is employed by some popular consumer IoT devices like the Nest Labs Learning Thermostat and the Samsung SmartTV. In the case of the Nest Learning Thermostat, the device transmits data to a cloud database where the data can be used to analyze home energy consumption.

Further, this cloud connection enables the user to obtain remote access to their thermostat via a smartphone or Web interface, and it also supports software updates to the thermostat. Similarly, with the Samsung SmartTVtechnology, the television uses an Internet connection to transmit user viewing information to Samsung for analysis and to enable the interactive voice recognition features of the TV. In these cases, the device-to-cloud model adds value to the end user by extending the capabilities of the device beyond its native features.

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

2.3.3 Device-to-Gateway Model

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

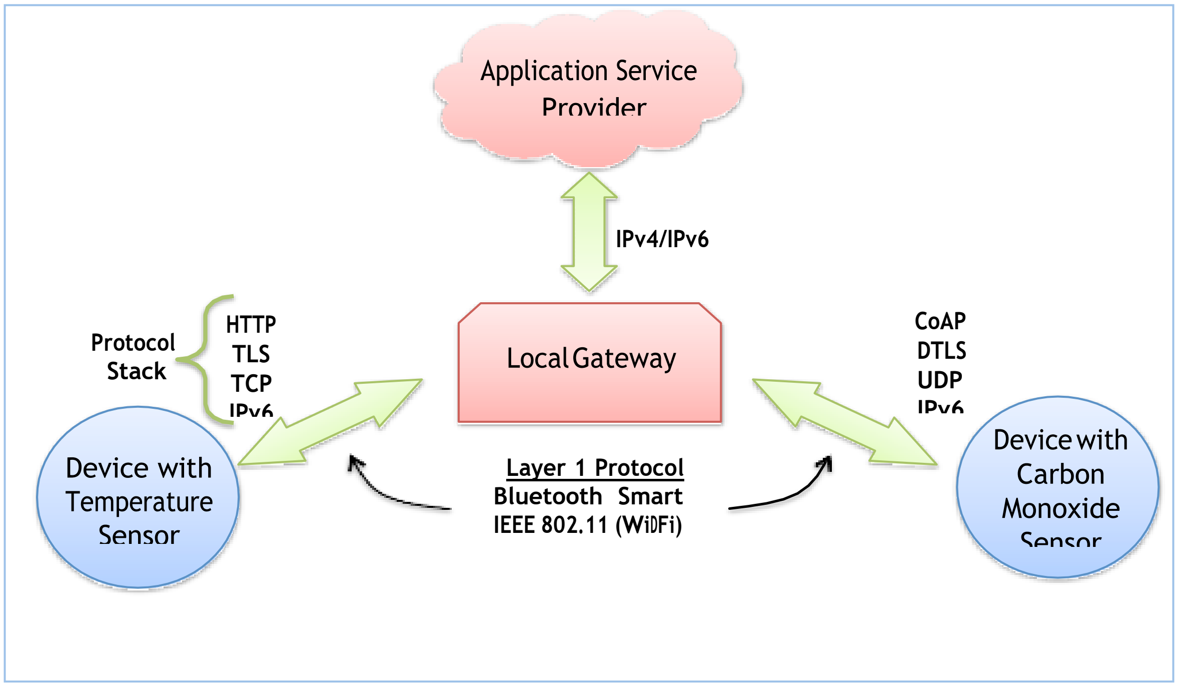


Fig 28.Device-to-gateway communication model diagram.

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

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

2.3.4 Back-End Data-Sharing Model

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

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

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

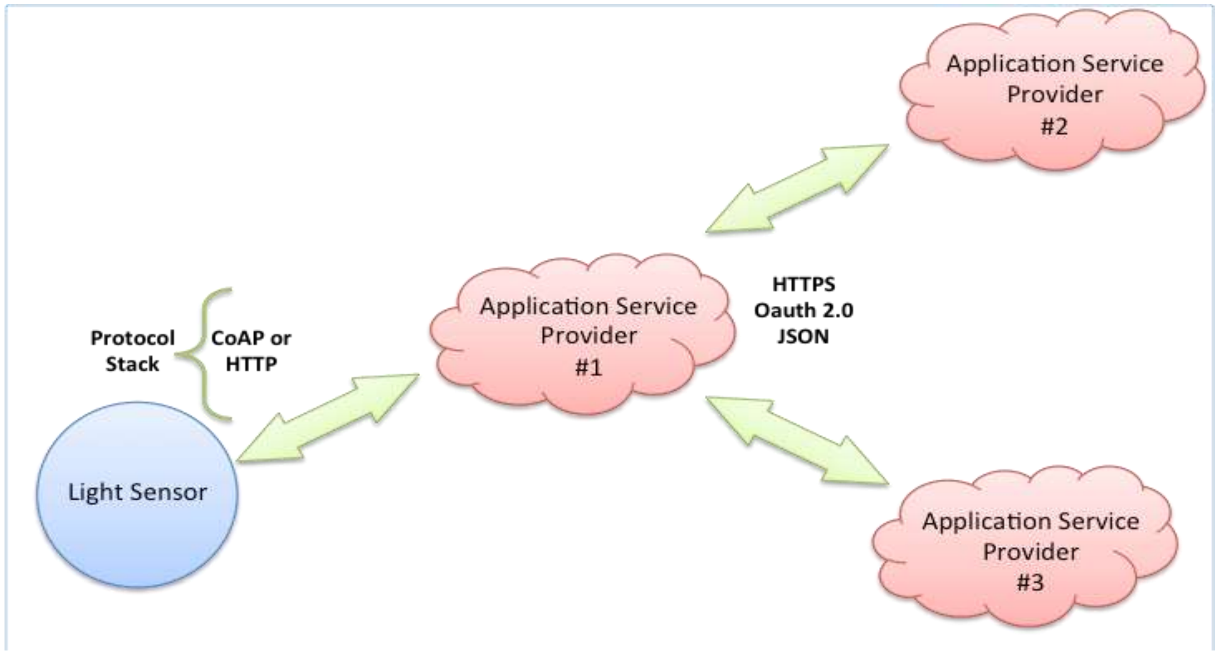

Application Service Provider #2
HTTPS
Oauth 2.0
JSON
Application Service Provider #1
Protocol Stack
CoAP or HTTP
Light Sensor
Application Service Provider #3

Fig 29. Back-end data sharing model diagram

Internet of Things Communications Models Summary

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

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

2.4 Low Power Wide Area Networks: An Overview

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

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

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

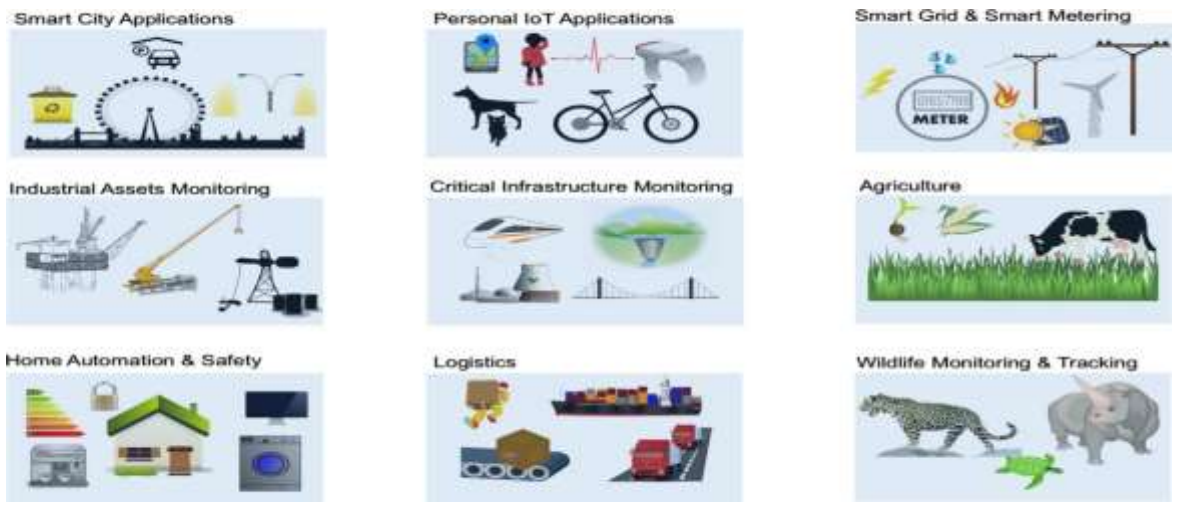

Smart City Applications
Personal IoT Applications
Smart Grid & Smart Metering
METER
Industrial Assets Monitoring
Critical Infrastructure Monitoring
Agriculture
Home Automation & Safety
Logistics
Wildlife Monitoring & Tracking

Fig 30.Applications of LPWA technologies across different sectors

2.4.1 Key Objective Of LPWA Technologies

A. **Long range**

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

B. **Ultra low power operation**

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

C. Topology

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

D. **Duty Cycling**: Low power operation is achieved by opportunistically turning off power hungry components of M2M/IoT devices e.g., data transceiver. Radio duty cycling allows

LPWA end devices to turn off their transceivers, when not required. Only when the data is to be transmitted or received, the transceiver is turned on.

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

2.4.2 CHALLENGES AND OPEN RESEARCH DIRECTIONS LPWA

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

1. Scaling networks to massive number of devices

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

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

3. Localization

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

4. Link optimizations and adaptability

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

5. LPWA test beds and tools

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

6. Authentication, Security, and Privacy

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

7. Mobility and Roaming

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

2.5 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.

2.5.1 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

2.5.2 WPAN Topologies:

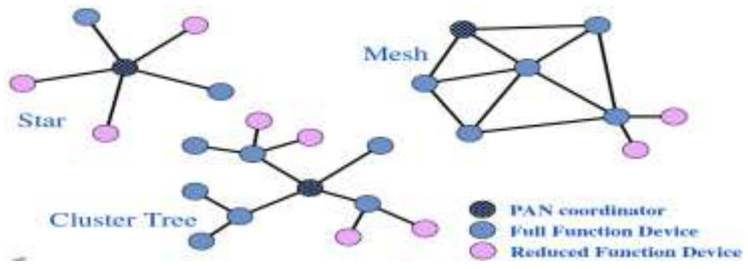


Fig 31 WPAN Topologies

2.5.3 IEEE 802.15 WPAN Standards:

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

2.5.4 Bluetooth ≈ IEEE 802.15.1

A widely used WPAN technology is known as Bluetooth (version 1.2 or version 2.0). The IEEE 802.15.1 standard specifies the architecture and operation of Bluetooth devices, but only as far as physical layer and medium access control (MAC) layer operation is concerned (the core system architecture) .Higher protocol layers and applications defined in usage profiles are standardized by the Bluetooth SIG. Bluetooth is the base for IEEE Std 802.15.1-2002 (rev. 2005).Data rate of 1 Mbps (2 or 3 Mbps with enhanced data rate).

2.5.4.1 Piconets

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

2.5.4.2 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

2.6 Bluetooth Protocol Stack

2.6.1 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

2.6.2 Baseband Layer

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

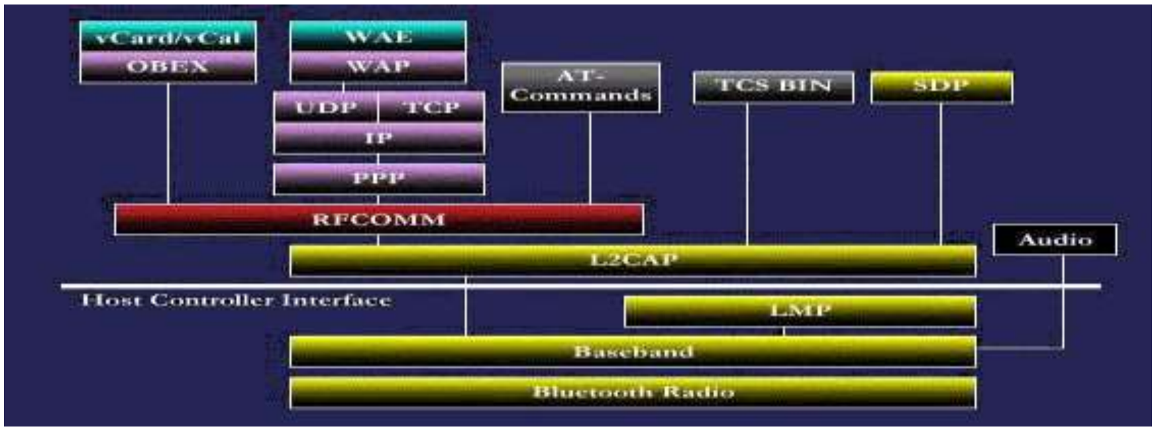


Fig 32 Bluetooth Protocol Stack

2.6.3 Link Manager layer

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

2.6.4 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

2.6.5 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.

2.6.7RFCOMM

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

2.6.8 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.

2.6.9 TCSBIN Protocol

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

2.6.10 Service Discovery Protocol

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

2.7 IoT platform

The purpose of any **IoT device** is to connect with other IoT devices and applications (cloud-based mostly) to relay information using internet transfer protocols.
The gap between the device sensors and data networks is filled by an **IoT Platform**. Such a platform connects the data network to the sensor arrangement and provides insights using backend applications to make sense of plethora of data generated by hundreds of sensors.

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

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

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

According to Amazon, their **IoT platform** will make it a lot easier for developers to connect sensors for multiple applications ranging from automobiles to turbines to smart home light bulbs.

Taking the scope of AWS IoT to the next level, the vendor has partnered with hardware manufacturers like Intel, Texas Instruments, Broadcom and Qualcomm to create starter kits compatible with their platform.

IoT Platform: Microsoft Azure IoT

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

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

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

Top IoT Platforms-Google Cloud Platform (New)

Google can make things happen. With its end-to-end platform, Google cloud is among the **best IoT platforms** we have currently. With the ability to handle the vast amount of data using Cloud IoT Core, Google stands out from the rest. You get advanced analytics owing to Google's Big Query and Cloud Data Studio.

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

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

IoT Platform: ThingWorx IoT Platform

Thingsworx is an IoT platform which is designed for enterprise application development. It offers features like:

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

IoT platform: IBM Watson

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

Users of IBM Watson get:

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

Top IoT Platforms-Artik (New):samsung

IoT Platform: Cisco IoT Cloud Connect

Top IoT Platforms-Universal of Things (IoT) Platform (New) :HP

Top IoT Platforms-Datav by Bsquare (New)

IoT platform: Salesforce IoT Cloud

Top IoT Platforms-Mindsphere by Siemens (New)

Top IoT Platforms-Ayla Network by Ayla (New)

Top IoT Platforms-Bosch IoT Suite(New)

IoT Platform: Carriots

IoT Platform: Oracle Integrated Cloud

IoT Platform: General Electric's Predix

Top IoT Platforms-MBED IoT Device platform(New)

Top IoT Platforms-Mosaic (LTI) (New)

IoT Platform: Kaa

AWS IoT Core is a managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices. AWS IoT Core can support billions of devices and trillions of messages, and can process and route those messages to AWS endpoints and to other devices reliably and securely. With AWS IoT Core, your applications can keep track of and communicate with all your devices, all the time, even when they aren't connected.

AWS IoT Core makes it easy to use AWS services like AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch, AWS CloudTrail, and Amazon Elasticsearch Service with built-in Kibana integration, to build IoT applications that gather, process, analyze and act on data generated by connected devices, without having to manage any infrastructure.

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

2.7.1 AWS IoT Components

AWS IoT consists of the following components:

A. Device gateway

Enables devices to securely and efficiently communicate with AWS IoT.

B. Message broker

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

C. Rules engine

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

D. Security and Identity service:

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

E. Registry

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

F. Group registry

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

G. Device shadow

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

H. Device Shadow service

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

I. Device Provisioning service

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

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

J. **Custom Authentication service**

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

K. **Jobs Service**

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

2.7.2 Accessing AWS IoT

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

- **AWS Command Line Interface (AWS CLI)**—Run commands for AWS IoT on Windows, macOS, and Linux. These commands allow you to create and manage things, certificates, rules, and policies. To get started, see the AWS Command Line Interface User Guide. For more information about the commands for AWS IoT, see iot in the *AWS CLI Command Reference.*
- **AWS IoT API**—Build your IoT applications using HTTP or HTTPS requests. These API actions allow you to programmatically create and manage things, certificates, rules, and policies. For more information about the API actions for AWS IoT, see Actions in the *AWS IoT API Reference.*
- **AWS SDKs**—Build your IoT applications using language-specific APIs. These SDKs wrap the HTTP/HTTPS API and allow you to program in any of the supported languages.

- **AWS IoT Device SDKs**—Build applications that run on devices that send messages to and receive messages from AWS IoT.

Related Services

AWS IoT integrates directly with the following AWS services:

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

2.7.3 Benefits

a. Connect and Manage Your Devices

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

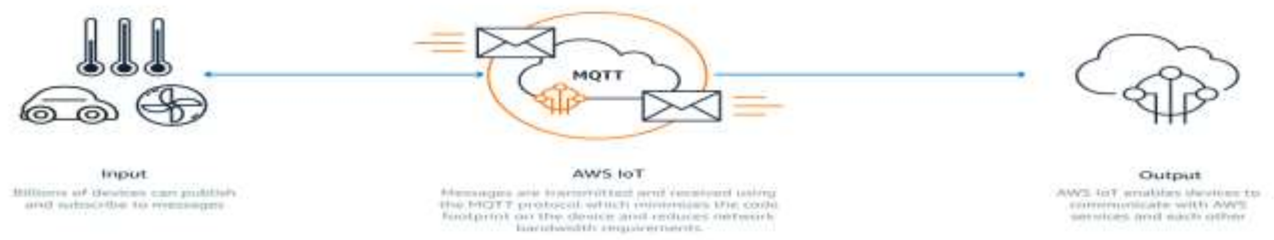

MQTT
Input
Billions of devices can publish and subscribe to messages
AWS IoT
Messages are transmitted and received using the MQTT protocol which minimizes the code footprint on the device and reduces network bandwidth requirements.
Output
AWS IoT enables devices to communicate with AWS services and each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

Fig 33. IOT Device Management

B. Secure Device Connections and Data

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

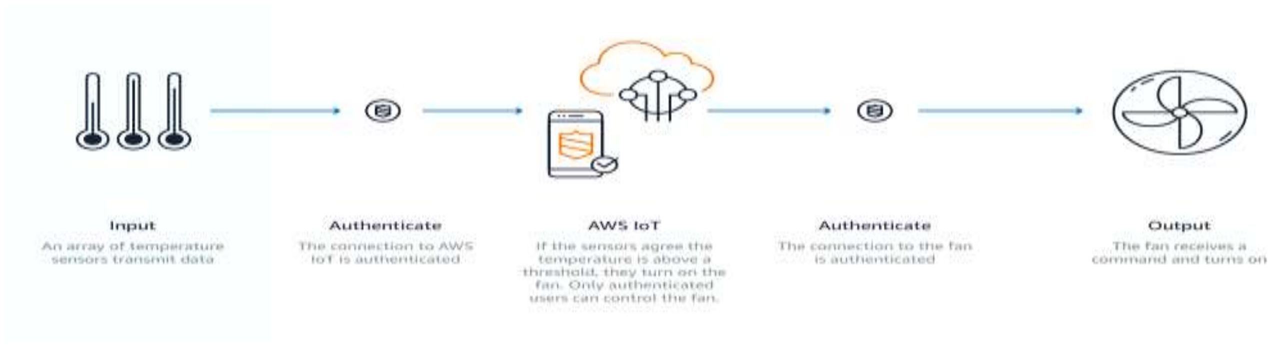

Input
An array of temperature sensors transmit data
Authenticate
The connection to AWS IoT is authenticated
AWS IoT
If the sensors agree the temperature is above a threshold, they turn on the fan. Only authenticated users can control the fan.
Authenticate
The connection to the fan is authenticated
Output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

Fig 34. Secure Connection

2.7.4 PROCESS AND ACT UPON DEVICE DATA

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

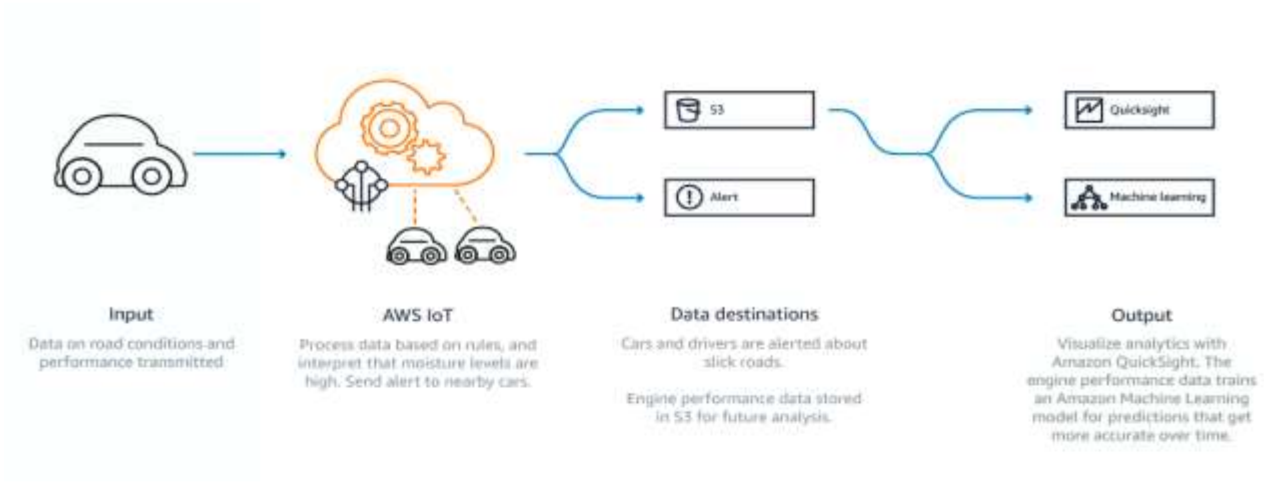

S3
Alert
Quicksight
Machine learning
Input
Data on road conditions and performance transmitted
AWS IoT
Process data based on rules, and interpret that moisture levels are high. Send alert to nearby cars.
Data destinations
Cars and drivers are alerted about slick roads.
Engine performance data stored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

Fig 35. Process And Act Upon Device Data

2.7.5 Read and set device state at any time

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

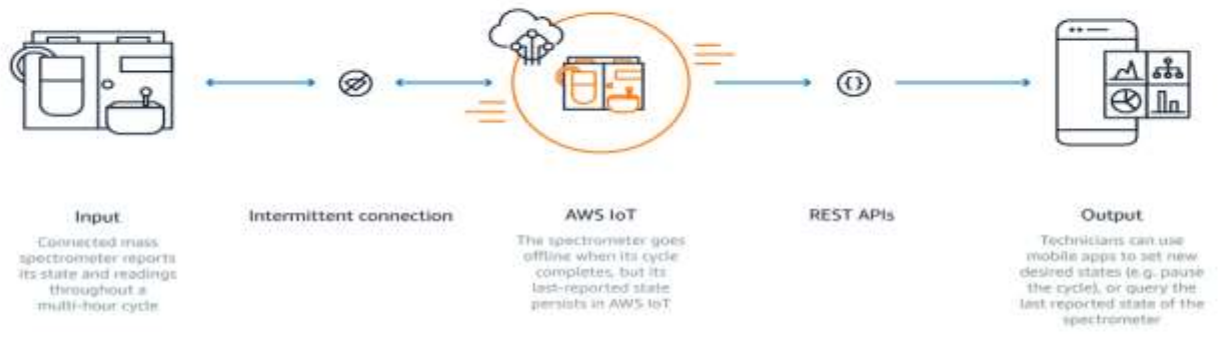

Input
Connected mass spectrometer reports its state and readings throughout a multi-hour cycle
Intermittent connection
AWS IoT
The spectrometer goes offline when its cycle completes, but its last-reported state persists in AWS IoT
REST APIs
Output
Technicians can use mobile apps to set new desired states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

Fig 36. Read and set device state at any time

2.8 PROGRAMS for Arduino, Raspberry pi

PIR motion sensor

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

The parts needed for this build are given as

1.Arduino uno or clone

2.PIR Sensor

3.breadboard

4.led(any colour)

5.buzzer

The following fig 26 shows the pin description of PIR sensor.

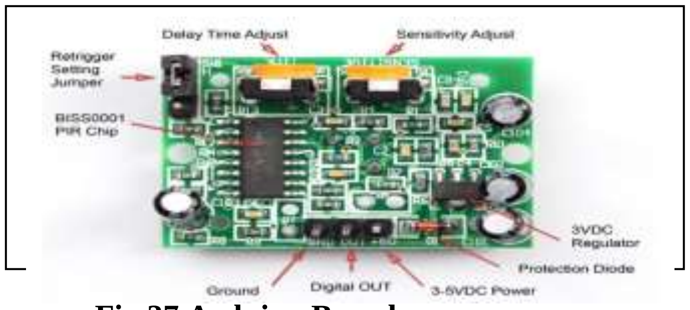


Fig 37.Arduino Board

Features and Electrical Specification

Compact size (28 x 38 mm)

Supply current: DC5V-20V(can design DC3V-24V)

Current drain :< 50uA (Other choice: DC0.8V-4.5V; Current drain: 1.5mA-0.1mA)

Voltage Output: High/Low level

signal ：3.3V (Other choice: Open-Collector Output) TTL output High sensitivity

Delay time：5s-18 minute

Blockade time：0.5s-50s (acquiescently 0 seconds)

the connections follows as

ARDUINO UNO PIR SENSOR::

+5V--------------------------------------+5V(VCC)

GND-------------------------------------GND

5 PIN-------------------------------------OUT

The coding related to the following circuit is very simple the code follows in this way

int PIR_output=5; // output of pir sensor

int led=13; // led pin

int buzzer=12;// buzzer pin

After that is done we can move on to the void setup fuction

pinMode(PIR_output, INPUT);// setting pir output as arduino input

pinMode(led, OUTPUT);//setting led as output

pinMode(buzzer, OUTPUT);//setting buzzer as output

Serial.begin(9600);//serial communication between arduino and pc

you can download the full program from above PIR_sensor_by_kj_electronics file and upload it to the arduino uno by using the arduino ide

Ultrasonic sensor

It works by sending sound waves from the transmitter, which then bounce off of an object and then return to the receiver. You can determine how far away something is by the time it takes for the sound waves to get back to the sensor. **The connections are very simple:**

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

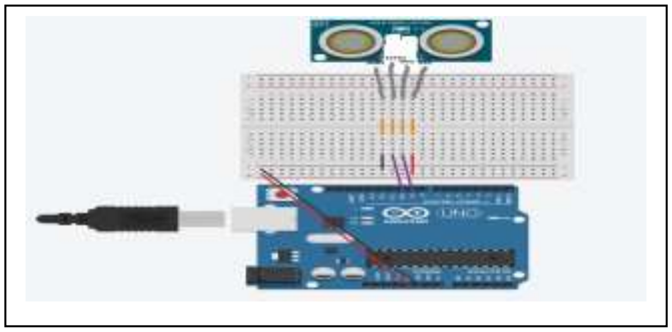


Fig 38.Ultra Sonic Sensor

Code:

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

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

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

```
float duration, distance;
```

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

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

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

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

When the sound waves hit the receiver, it turns the Echo pin high for however long the waves were traveling for. To get that, we can use a handy Arduino function called pulseIn().

It takes 2 arguments, the pin you are listening to(In our case, the Echo pin), and a state(HIGH or LOW). What the function does is waits for the pin to go whichever sate you put in, starts timing, and then stops timing when it switches to the other state. In our case we would put HIGH since we want to start timing when the Echo pin goes high. We will store the time in the duration variable. (It returns the time in microseconds)

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

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

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

The rest is just printing out the results to the Serial Monitor.

```
Serial.print("Distance: ");
 Serial.println(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 to be handled properly. Be careful of static electricity when handling them and make sure the power supply is connected up correctly and is between 2.7 and 5.5V DC - so don't try to use a 9V battery.

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

Reading the Analog Temperature Data

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

Remember that you can use anywhere between 2.7V and 5.5V as the power supply. For this example I'm showing it with a 5V supply but note that you can use this with a 3.3v supply just as easily. No matter what supply you use, the analog voltage reading will range from about 0V (ground) to about 1.75V.

If you're using a 5V Arduino, and connecting the sensor directly into an Analog pin, you can use these formulas to turn the 10-bit analog reading into a temperature:

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

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

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

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

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

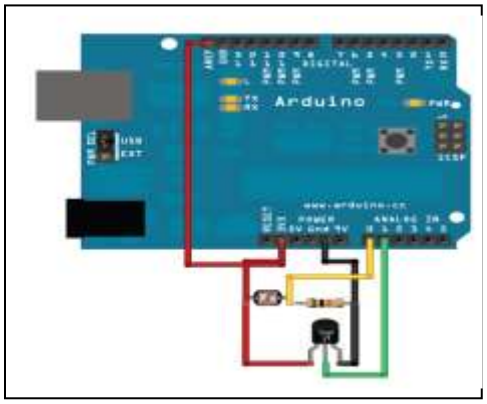


Fig 39 Temperature Sensor

Arduino Sketch - Simple Thermometer

This example code for Arduino shows a quick way to create a temperature sensor, it simply prints to the serial port what the current temperature is in both Celsius and Fahrenheit.
For better results, using the 3.3v reference voltage as ARef instead of the 5V will be more precise and less
noisy

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

PYTHON CODE :

PIR sensor code

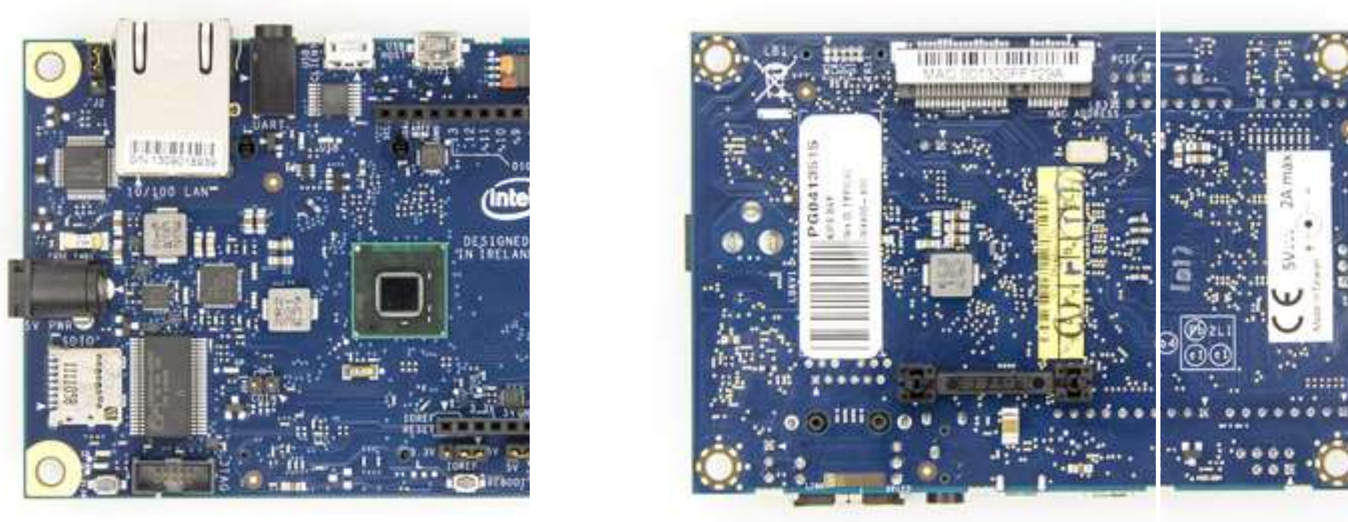
```
import RPi.GPIO as GPIO
  #Import GPIO library
import time
 #Import time library
GPIO.setmode(GPIO.BOARD)
  #Set GPIO pin numbering
pir = 26
 #Associate pin 26 to pir
GPIO.setup(pir, GPIO.IN)
   #Set pin as GPIO in
print "Waiting for sensor to settle"
time.sleep(2)
 #Waiting 2 seconds for the sensor to initiate
print "Detecting motion"
```

```
while True:
   if GPIO.input(pir):
 #Check whether pir is HIGH
      print "Motion Detected!"
      time.sleep(2)
 #D1- Delay to avoid multiple detection
   time.sleep(0.1)
  #While loop delay should be less than detection(hardware) delay
```

Ultrasonic

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.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 To Settle"
time.sleep(2)

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

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

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

pulse_duration = pulse_end - pulse_start

distance = pulse_duration * 17150

distance = round(distance, 2)

print "Distance:",distance,"cm"

GPIO.cleanup()
```

2.9 Intel Galileo



Intel Galileo Front   Intel Galileo Back

Fig 40.Intel Galileo

2.9.1 Overview

Galileo is a microcontroller board based on the Intel® Quark SoC X1000 Application Processor, a 32-bit Intel Pentium-class system on a chip. It’s the first board based on Intel® architecture designed to be hardware and software pin-compatible with Arduino shields designed for the Uno R3. Digital pins 0 to 13 (and the adjacent AREF and GND pins), Analog inputs 0 to 5, the power header, ICSP header, and the UART port pins (0 and 1), are all in the same locations as on the Arduino Uno R3. This is also known as the Arduino 1.0 pinout.

Galileo is designed to support shields that operate at either 3.3V or 5V. The core operating voltage of Galileo is 3.3V. However, a jumper on the board enables voltage translation to 5V at the I/O pins. This provides support for 5V Uno shields and is the default behavior. By switching the jumper position, the voltage translation can be disabled to provide 3.3V operation at the I/O pins.

the Galileo board is also software compatible with the Arduino Software Development Environment (IDE), which makes usability and introduction a snap. In addition to Arduino hardware and software compatibility, the Galileo board has several PC industry standard I/O ports and features to expand native usage and capabilities beyond the Arduino shield ecosystem. A full sized mini-PCI Express slot, 100Mb Ethernet port, Micro-SD slot, RS-232 serial port, USB Host port, USB Client port, and 8MByte NOR flash come standard on the board.

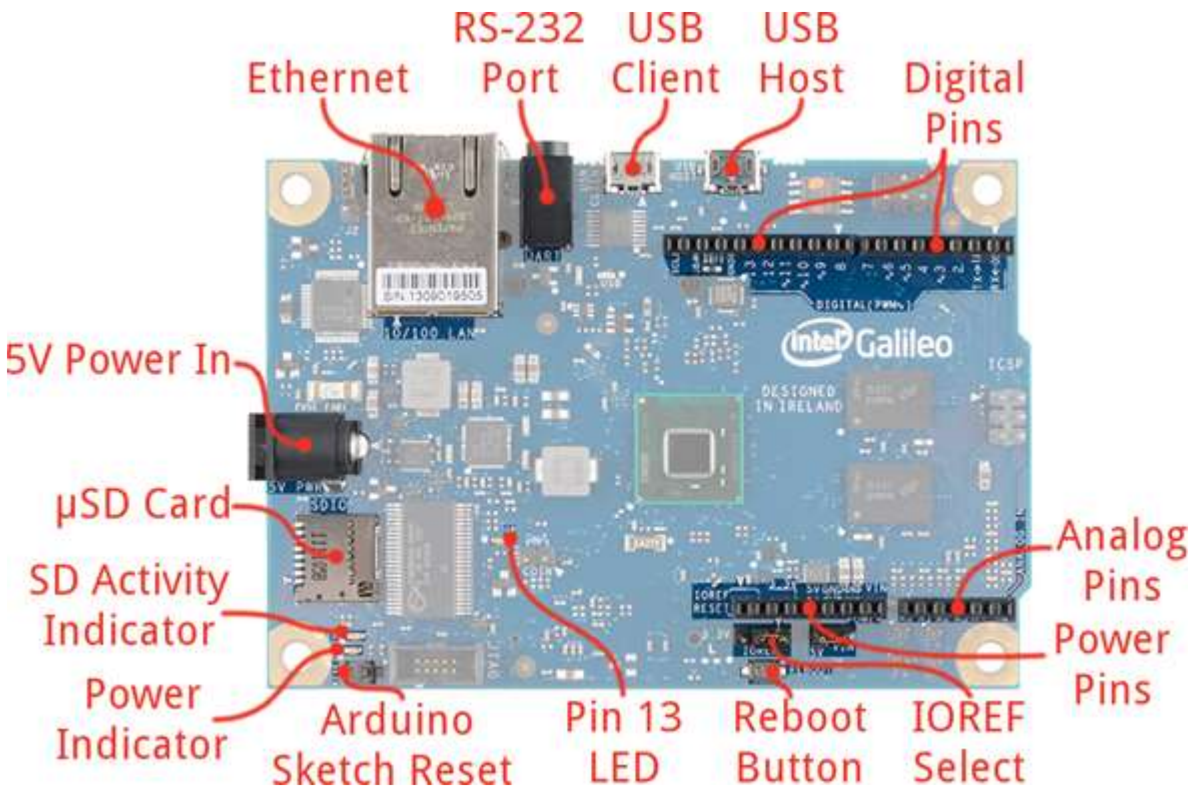


Fig 41.Galileo Quick Review

If you are into programming and technologies, then you guys trust me, this is for you. From a programmer's perspective ,

You can create things for your home/office automation.

- You can use it in Robotics.
- You can take a step towards Artificial Intelligence.

2.9.2 Benefits

- The Galileo board is also software compatible with the Arduino Software Development Environment (IDE), that makes usability and introduction a snap.
- The Galileo board has several PC industry standard I/O ports and features to expand native usage and capabilities beyond the Arduino shield ecosystem.

- A full-sized mini-PCI Express slot, 100Mb Ethernet port, Micro-SD slot, RS-232 serial port, USB Host port, USB Client port, and 8MByte NOR flash come standard on the board.
- Using an Intel Galileo Board you can create your own world of Internet of Things (IoT) by writing just a few sketches.

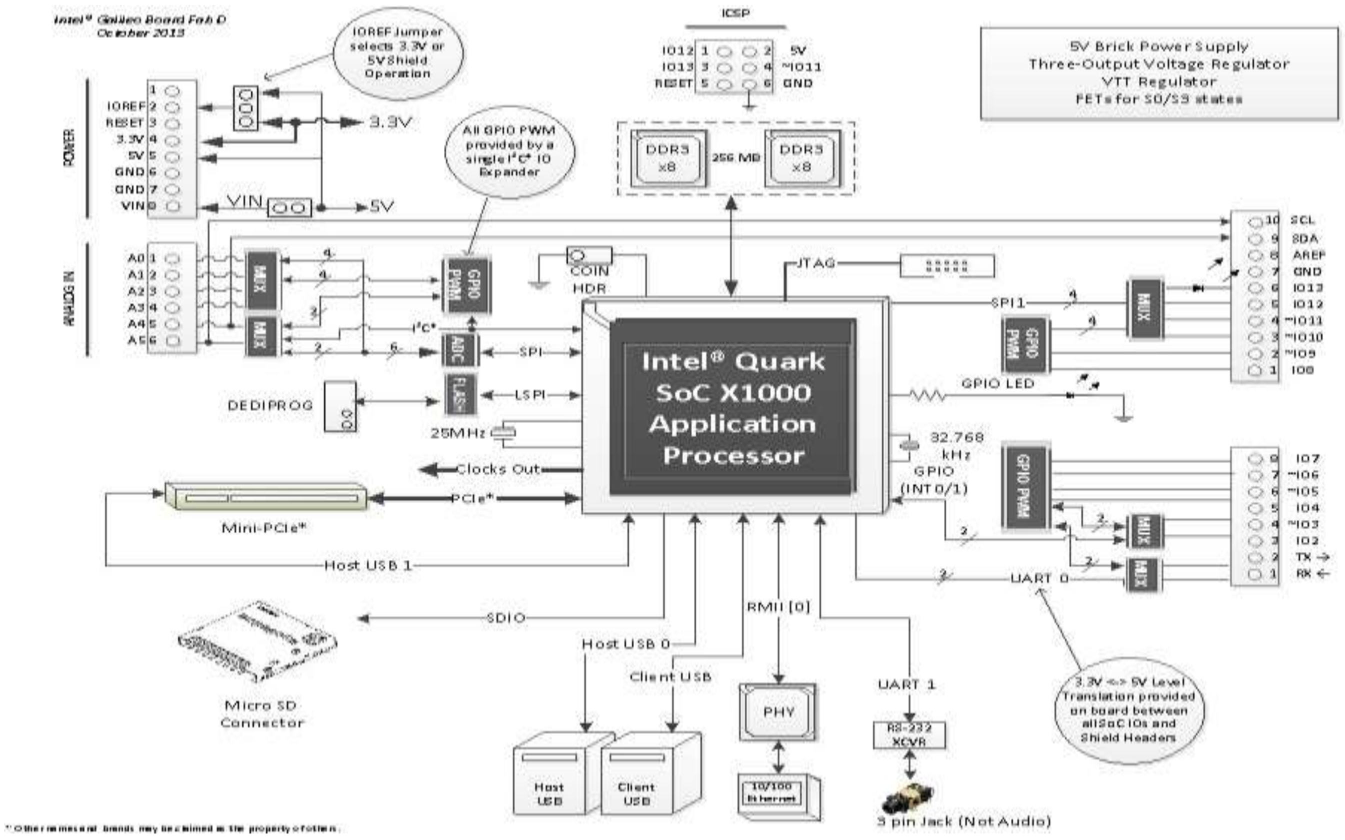


Fig 42.Intel Architecture

2.9.3 Details of Intel Architecture Supported Features:

The genuine Intel processor and surrounding native I/O capabilities of the Clanton SoC provides for a fully featured offering for both the maker community and students alike. It will also be useful to professional developers who are looking for a simple and cost effective development environment to the more complex Intel® Atom processor and Intel® Core processor-based designs.

- 512 KBytes of on-die embedded SRAM
- Simple to program: Single thread, single core, constant speed
- ACPI compatible CPU sleep states supported
- An integrated Real Time Clock (RTC), with an optional 3V “coin cell” battery for operation between turn on cycles.
-

- 10/100 Ethernet connector
- Full PCI Express* mini-card slot, with PCIe 2.0 compliant features
    - Works with half mini-PCIe cards with optional converter plate
    - Provides USB 2.0 Host Port at mini-PCIe connector
- USB 2.0 Host connector
    - Support up to 128 USB end point devices
- USB Device connector, used for programming
    - Beyond just a programming port - a fully compliant USB 2.0 Device controller
- 10-pin Standard JTAG header for debugging
- Reboot button to reboot the processor
- Reset button to reset the sketch and any attached shields

2.9.4 Storage options:

- Default - 8 MByte Legacy SPI Flash main purpose is to store the firmware (or bootloader) and the latest sketch. Between 256KByte and 512KByte is dedicated for sketch storage. The download will happen automatically from the development PC, so no action is required unless there is an upgrade that is being added to the firmware.

- Default 512 KByte embedded SRAM, enabled by the firmware by default. No action required to use this feature.
- Default 256 MByte DRAM, enabled by the firmware by default.
- Optional micro SD card offers up to 32GByte of storage
- USB storage works with any USB 2.0 compatible drive
- 11 KByte EEPROM can be programmed via the EEPROM library.

2.9.5 Power

Galileo is powered via an AC-to-DC adapter, connected by plugging a 2.1mm center-positive plug into the board's power jack. The recommended output rating of the power adapter is 5V at up to 3Amp.

2.9.6 Electrical Summary

Input Voltage (recommended)	5V
Input Voltage (limits)	5V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
Total DC Output Current on all I/O lines	80 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA

2.9.7 Communication

Galileo has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. Galileo provides UART TTL (5V/3.3V) serial communication, which is available on digital pin 0 (RX) and 1 (TX). In addition, a second UART provides RS-232 support and is connected via a 3.5mm jack. The USB Device ports allows for serial (CDC) communications over USB. This provides a serial connection to the Serial Monitor or other applications on your computer. It also enables Galileo to act as a USB mouse or keyboard to an attached computer. To use these features, see the Mouse and Keyboard library reference pages. The USB Host port allows Galileo act as a USB Host for connected peripherals such as mice, keyboards, and smartphones. To use these features, see the USBHost reference pages. Galileo is the first Arduino board to provide a mini PCI Express (mPCIe) slot. This slot allows full size and half size (with adapter) mPCIe modules to be connected to the board and also provides an additional USB Host port via the slot. Any standard mPCIe module can be connected and used to provide applications such as WiFi, Bluetooth or Cellular connectivity. Initially, the Galileo mPCie slot provides support for the WiFi Library. An Ethernet RJ45 Connector is provided to allow Galileo to connect to wired networks. When connecting to a network, you must provide an IP address and a MAC address. Full support of on-board Ethernet interface is fully supported and does not require the use of the SPI interface like existing Arduino shields. The onboard microSD card reader is accessible through the SD Library. The communication between Galileo and the SD card is provided by an integrated SD controller and does not require the use of the SPI interface like other Arduino boards. The Arduino software includes a Wire library to simplify use of the TWI/I2C bus.

2.9.8 Programming

Galileo can be programmed with the Arduino software. When you are ready to upload the sketch to the board, program Galileo through the USB Client port by selecting "Intel Galileo" as your board in the Arduino IDE. Connect Galileo's port labelled USB Client (the one closest to the Ethernet) to your computer. Rather than requiring a physical press of the reset button before an upload, Galileo is designed to be reset by software running on a connected computer.

When the board boots up two scenarios are possible:

- If a sketch is present in persistent storage, it is executed.
- If no sketch present, the board waits for upload commands from the IDE.

If a sketch is executing, you can upload from the IDE without having to press the reset button on the board. The sketch is stopped; the IDE waits for the upload state, and then starts the newly uploaded sketch. Pressing the reset button on the board restarts a sketch if it is executing and resets any attached shields.

2.9.9 Properties of Pins Configured as OUTPUT

Pins configured as OUTPUT with pinMode() are said to be in a low-impedance state. On Galileo, when a pin is configured as OUTPUT, the functionality is provided via an I2C-based Cypress I/O expander. Digital pins 0 to 13 and Analog pins A0 to A5 can be configured as OUTPUT pins on Galileo.

The I/O expander's pins, when configured as OUTPUT, can source (provide positive current) up to 10 mA (milliamps) and can sink (provide negative current) up to 25 mA of current to other devices/circuits. The individual per pin current sourcing capability of 10 mA is subject to an overall limit of 80 mA combined between all OUTPUT pins. The per pin capability current sinking capability is subject to an overall limit of 200 mA. The following table provides a breakdown of the overall OUTPUT capabilities of the pins.

	Current Source (mA)	Current Sink (mA)
Per Pin Capability	10	25
Digital Pins 3,5,9,10,12, 13 Combined	40	100
Digital Pins 0,1,2,4,6,7,8,11 and Analog Pins A0,A1,A2,A3,A4, A5 Combined	40	100
Digital Pins 0-13 and Analog Pins A0-A5 Combined	80	200

2.9.10 Galileo Jumper Configuration

There are three jumpers on Galileo that are used to vary the configuration of the board. IOREF Jumper To allow Galileo support both 3.3V and 5V shields, the external operating voltage is controlled via a jumper. When the jumper is connected to 5V, Galileo is configured to be compatible with 5V shields and IOREF is set to 5V. When the jumper is connected 3.3V, Galileo is configured to be compatible with 3.3V shields and IOREF is set to 3.3V. The input range of the Analog pins is also controlled by the IOREF jumper and must not exceed the chosen operating voltage. However, the resolution of AnalogRead() remains at 5 V/1024 units for the default 10-bit resolution or, 0.0049V (4.9mV) per unit regardless of IOREF jumper setting.

The IOREF jumper should be used to match the board and shield operating voltages. Incorrectly setting the voltage could damage the board or the shield. I2C Address Jumper To prevent a clash between the I2C Slave address of the on board I/O expander and EEPROM with any external I2C Slave devices, jumper J2 can be used to vary the I2C address of the on-board devices. With J2 connected to pin 1 (marked with white triangle), the 7-bit I/O Expander address is 0100001 and the 7-bit EEPROM address is 1010001. Changing the jumper position changes the I/O Expander address to 0100000 and the EEPROM address to 1010000. VIN Jumper On Galileo, the VIN pin can be used to supply 5V from the regulated power supply connected at the power jack to attached shields or devices. If there is a need to supply more than 5V to a shield using VIN then the VIN jumper should be removed from Galileo to break the connection between the on-board 5V supply and the VIN connection on the board header.

If the VIN jumper is not removed and more than 5V is connected to VIN, it may damage the board or lead to unreliable operation.

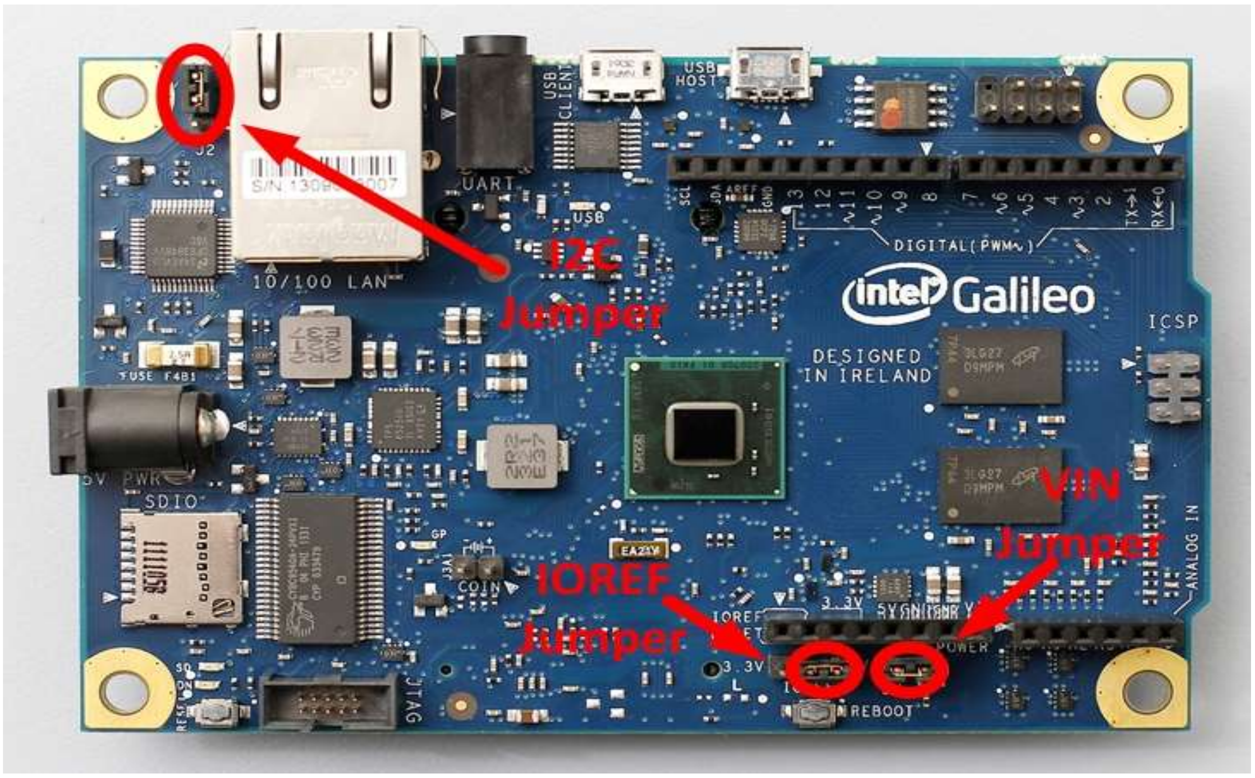


Fig 42.Galileo Jumper

2.9.11 Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, Galileo is designed in a way that allows it to be reset by software running on a connected computer. USB CDC-ACM control signals are used to transition Galileo from run-time to bootloader mode. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment.

2.9.12 Physical Characteristics

Galileo is 4.2 inches long and 2.8 inches wide respectively, with the USB connectors, UART jack, Ethernet connector, and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), is not an even multiple of the 100 mil spacing of the other pins

2.9.13 New tools for innovation

Introducing the Intel® Galileo development board, the first product in a new family of Arduino-compatible development boards featuring Intel® architecture. The platform is easy to use for new designers and for those looking to take designs to the next level. Intel recognizes that the people who take creating beautiful things into their own hands with innovation and technology are the same ones that propel us forward.

The Intel Galileo development board is a great tool for quickly prototyping simple, interactive designs such as LED light displays that respond to social media, or for tackling more complex projects, from automating home appliances to building life-size robots controlled by a smartphone.

This platform provides the ease of Intel architecture development through support for the Microsoft Windows*, Mac OS* and Linux* host operating systems. It also brings the simplicity of the Arduino software integrated development environment (IDE). It's all about delivering Intel performance and quality to the DIY maker community to support invention and creativity.

2.10 ARM Cortex Processor:

2.10.1 Introduction

System-on-chip solutions based on ARM embedded processors address many different market segments including enterprise applications, automotive systems, home networking and wireless technologies. The ARM Cortex™ family of processors provides a standard architecture to address the broad performance spectrum required by these diverse technologies. The ARM Cortex family includes processors based on the three distinct profiles of the ARMv7 architecture; the A profile for sophisticated, high-end applications running open and complex operating systems; the R profile for real-time systems; and the M profile optimized for cost-sensitive and microcontroller applications. The Cortex-M3 processor is the first ARM processor based on the ARMv7-M architecture and has been specifically designed to achieve high system performance in power- and cost-sensitive embedded applications, such as microcontrollers, automotive body systems, industrial control systems and wireless networking, while significantly simplifying programmability to make the ARM architecture an option for even the simplest applications.

There have been several generations of the ARM design. The original ARM1 used a 32-bit internal structure but had a 26-bit address space that limited it to 64 MB of main memory. This limitation was removed in the ARMv3 series, which has a 32-bit address space, and several additional generations up to ARMv7 remained 32-bit. Released in 2011, the ARMv8-A architecture added support for a 64-bit address space and 64-bit arithmetic with its new 32-bit fixed-length instruction set.Arm Ltd. has also released a series of additional instruction sets for different rules; the "Thumb" extension adds both 32- and 16-bit instructions for improved code density, while Jazelle added instructions for directly handling Java bytecodes, and more recently, JavaScript. More recent changes include the addition of simultaneous multithreading (SMT) for improved performance or fault tolerance.

Due to their low costs, minimal power consumption, and lower heat generation than their competitors, ARM processors are desirable for light, portable, battery-powered devices — including smartphones, laptops and tablet computers, as well as other embedded systems. However, ARM processors are also used for desktops and servers, including the world's fastest supercomputer. With over 180 billion ARM chips produced, as of 2021, ARM is the most widely used instruction set architecture (ISA) and the ISA produced in the largest quantity. Currently, the widely used Cortex cores, older "classic" cores, and specialised SecurCore cores variants are available for each of these to include or exclude optional capabilities.

2.10.2 ARM1

Acorn chose VLSI Technology as the "silicon partner",as they were a source of ROMs and custom chips for Acorn. Acorn provided the design and VLSI provided the layout and production. The first samples of ARM silicon worked properly when first received and tested on 26 April 1985. Known as ARM1, these versions ran at 6 MHz. The first ARM application was as a second processor for the BBC Micro, where it helped in developing simulation software to finish development of the support chips (VIDC, IOC, MEMC), and sped up the CAD software used in ARM2 development. Wilson subsequently rewrote BBC BASIC in ARM assembly language. The in-depth knowledge gained from designing the instruction set enabled the code to be very dense, making ARM BBC BASIC an extremely good test for any ARM emulator.

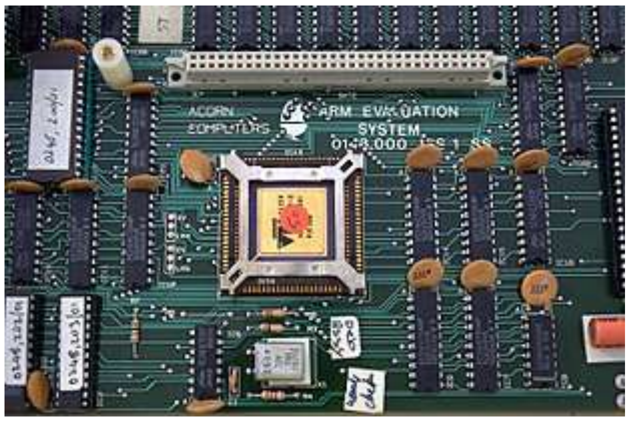


Fig 43. ARM1 2nd processor for the BBC Micro

2.10.3 ARM2

The result of the simulations on the ARM1 boards led to the late 1986 introduction of the ARM2 design running at 8 MHz, and the early 1987 speed-bumped version at 10 to 12 MHz.[b] A significant change in the underlying architecture was the addition of a Booth multiplier, whereas previously multiplication had to be carried out in software.[37] Additionally, a new Fast Interrupt reQuest mode, FIQ for short, allowed registers 8 through 14 to be replaced as part of the interrupt itself. This meant FIQ requests did not have to save out their registers, further speeding interrupts. The ARM2 was roughly seven times the performance of a typical 7 MHz 68000-based system like the Commodore Amiga or Macintosh SE. It was twice as fast as a Intel 80386 running at 16 MHz, and about the same speed as a multi-processor VAX-11/784 supermini. The only systems that beat it were the Sun SPARC and MIPS R2000 RISC-based workstations.[39] Further, as the CPU was designed for high-speed I/O, it dispensed with many of the support chips seen in these machines, notably, it lacked any dedicated direct memory access (DMA) controller which was often found on workstations. The graphics system was also simplified based on the same set of underlying assumptions about memory and timing. The result was a dramatically simplified design, offering performance on par with expensive workstations but at a price point similar to contemporary desktops. The ARM2 featured a 32-bit data bus, 26-bit address space and 27 32-bit registers. The ARM2 had a transistor count of just 30,000, compared to Motorola's six-year-older 68000 model with around 40,000.

This simplicity enabled low power consumption, yet better performance than the Intel 80286. A successor, ARM3, was produced with a 4 KB cache, which further improved performance. The address bus was extended to 32 bits in the ARM6, but program code still had to lie within the first 64 MB of memory in 26-bit compatibility mode, due to the reserved bits for the status flags.

2.10.4 Higher performance through better efficiency

In order to achieve higher performance, processors can either work hard or work smart. Pushing higher clock frequencies may increase performance but is also accompanied by higher power consumption and design complexity. On the other hand, higher compute efficiency at slower clock speeds results in simpler and lower power designs that can perform the same tasks. At the heart of the Cortex-M3 processor is an advanced 3-stage pipeline core, based on the Harvard architecture, that incorporates many new powerful features such as branch speculation, single cycle multiply and hardware divide to deliver an exceptional Dhrystone benchmark performance of 1.25 DMIPS/MHz. The Cortex-M3 processor also implements the new Thumb®-2 instruction set architecture, helping it to be 70% more efficient per MHz than an ARM7TDMI-S® processor executing Thumb instructions, and 35% more efficient than the ARM7TDMI-S processor executing ARM instructions, for the Dhrystone benchmark.

2.10.5 Ease of use for quick and efficient application development

Reducing time-to-market and lowering development costs are critical criteria in the choice of microcontrollers, and the ability to quickly and easily develop software is key to these requirements.

The Cortex-M3 processor has been designed to be fast and easy to program, with the users not required to write any assembler code or have deep knowledge of the architecture to create simple applications. The processor has a simplified stack-based programmer’s model which still maintains compatibility with the traditional ARM architecture but is analogous to the systems employed by legacy 8- and 16-bit architectures, making the transition to 32-bit easier. Additionally a hardware based interrupt scheme means that writing interrupt service routines (handlers) becomes trivial, and that start-up code is now significantly simplified as no assembler code register manipulation is required.

Key new features in the underlying Thumb-2 Instruction Set Architecture (ISA) implement C code more naturally, with native bitfield manipulation, hardware division and If/Then instructions. Further, from a development perspective, Thumb-2 instructions speed up development and simplify long term maintenance and support of compiled objects through automatic optimization for both performance and code density, without the need for complex interworking between code compiled for ARM or Thumb modes.

2.10.6 ARM6

In the late 1980s, Apple Computer and VLSI Technology started working with Acorn on newer versions of the ARM core. In 1990, Acorn spun off the design team into a new company named Advanced RISC Machines Ltd.,which became ARM Ltd. when its parent company, Arm Holdings plc, floated on the London Stock Exchange and NASDAQ in 1998.[46] The new Apple- ARM work would eventually evolve into the ARM6, first released in early 1992. Apple used the ARM6-based ARM610 as the basis for their Apple Newton PDA.

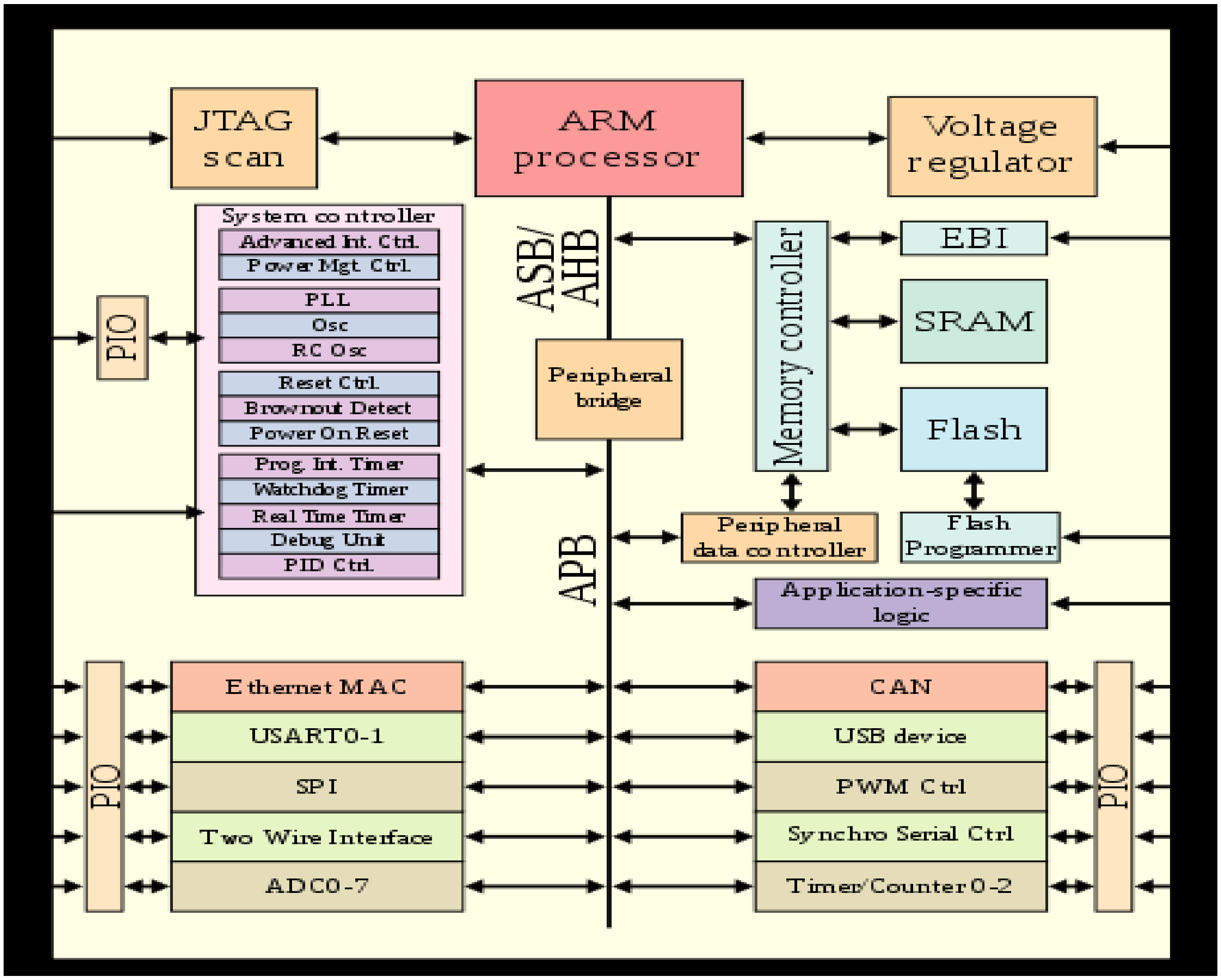


Fig 44. Microprocessor-based system on a chip

2.10.7 Reduced costs and lower power for sensitive markets

A constant barrier to the adoption of higher performance microcontrollers has always been cost. Advanced manufacturing technologies are expensive and therefore smaller silicon area requirements can reduce costs significantly. The Cortex-M3 processor reduces system area by implementing the smallest ARM core to date, with just 33,000 gates in the central core (0.18um G) and by efficiently incorporating tightly coupled system components in the processor. Memory requirements are minimized by implementing unaligned data storage, atomic bit manipulation and the Thumb-2 instruction set that reduces instruction memory requirements for the Dhrystone benchmark by more than 25% compared to ARM instructions.

In order to address the increasing need for energy conservation in markets like white goods and wireless networking, the Cortex-M3 processor supports extensive clock gating and integrated sleep modes. Enabled by these features, the processor delivers a power consumption of just 4.5mW and a silicon footprint of 0.30mm2 when implemented at a target frequency of 50MHz on the TSMC 0.13G process using ARM Metro™ standard cells.

2.10.8 Integrated debug and trace for faster time to market

Embedded systems typically have no graphical user interface making software debug a special challenge for programmers. In-circuit Emulator (ICE) units have traditionally been used as plug-in devices to provide a window into the system through a familiar PC interface. As systems get smaller and more complex, physically attaching such debug units is no longer a viable solution. The Cortex-M3 processor implements debug technology in the hardware itself with several integrated components that facilitate quicker debug with trace & profiling, breakpoints, watchpoints and code patching, significantly reducing time to market.

2.11 WEARABLE DEVELOPMENT BOARDS

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

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

- Hardware / electronic design and development expertise for developing products with minimum form factor size. Moreover, power management expertise is a must-have to ensure that devices can last several days / weeks / months on a single battery charge.
- BSP / firmware development expertise ensuring memory optimized driver development with minimum footprint, enhanced power management and highest performance in spite of low-end microcontroller.
- Smartphone application development on Android and iPhone platforms to allow connectivity with the wearable device.
- Cloud / web application for enabling M2M (machine-to-machine) / IoT (Internet of things) for remote monitoring and control.

2.11.1 Challenges

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

- Complex hardware designing abilities smaller sized form factor development.
- Knowledge of choosing the right components for size and efficient power management.
- Certifications to ensure radiations do not affect human body.

- Expertise in driver development for developing optimized drivers for new types of sensors.
- Ability to develop code that fits on lower capacity RAM / flash.
- Domain knowledge for efficient application development.
- Ability to develop iPhone / Android / Windows applications.
- Ability to develop Cloud / web / smartphone connectivity applications.

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

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

2.11.2 BSP / firmware / operating system development services

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

  Firmware is a software program permanently etched into a hardware device such as a keyboards, hard drive, BIOS, or video cards. It is programmed to give permanent instructions to communicate with other devices and perform functions like basic input/output tasks. Firmware is typically stored in the flash ROM (read only memory) of a hardware device.

  Firmware was originally designed for high level software and could be changed without having to exchange the hardware for a newer device. Firmware also retains the basic instructions for hardware devices that make them operative. Without firmware, a hardware device would be non-functional.

  Development Boards for IoT

- Driver development for various sensors such as accelerometers, gyroscopes, motion sensors, proximity sensors, magnetometers etc.
- Driver development for communication interfaces like BLE 4.0, NFC, RF and even using the 3.5 mm audio headphone jack.

- Development of power management algorithms.
- Power management by using the right components and critical designs.

2.11.3 Embedded / real-time application development services

Key aspects of application development include:

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

2.11.4 Smart phone application development services

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

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

2.11.5 Cloud / web application development services

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

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

1. C.H.I.P

CHIP is the new kid on the block. It comes with a powerful 1GHz processor powered by Allwinner R8. The best thing about CHIP is that it comes with embedded Bluetooth 4.0 and WiFi radios, providing out-of-the-box connectivity. The board has 4GB of high-speed storage to run a special Linux distribution based on Debian. You don't need a separate SD Card to install and run the OS. With 8 GPIO pins, CHIP can be connected to a variety of sensors. The board also supports PWM, UART, I2C for connecting motors and other actuators. One of the key advantages of CHIP is the cost and the form-factor. Developers can SSH into the Linux OS and install required packages.

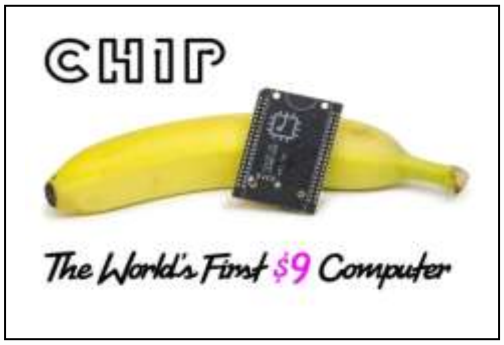


Fig 43. CHIP

2. Mediatek Linkit One

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.

Fig 44. Linkit

3. Particle Photon

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

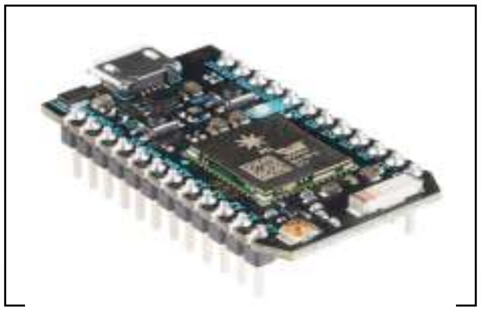


Fig 45.Particle Photon

4. 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.

Fig 46 Tessel

5. Adafruit Flora

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

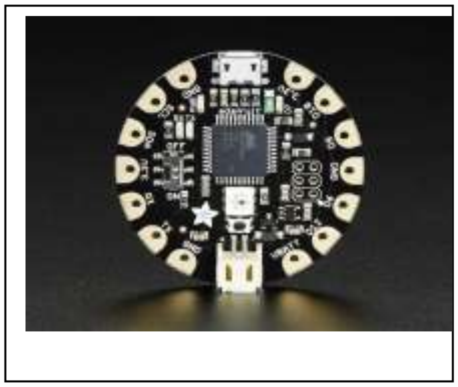


Fig 47.Adafruit Flora

6. 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.

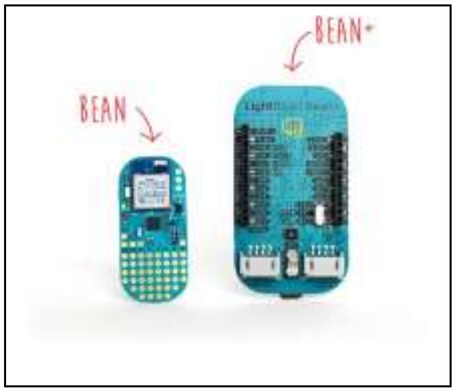


Fig 48 Light Blue Bean

7. Udoo Neo

Udoo Neo is a full-blown computer that also has an Arduino-compatible microcontroller. It's positioned as the combination of Raspberry Pi and Arduino. The board has the same pinout as Arduino Uno. Neo embeds two cores on the same processor – a powerful 1GHz ARM Cortex-A9, and an ARM Cortex-M4 I/O real-time co-processor. It packs a punch with an embedded 9-axis motion sensors and a Wi-Fi + Bluetooth 4.0 module. You can install Android Lollipop or a customized flavor of Debian Linux called UDOObuntu, which is compatible with Ubuntu 14.04 LTS.

When it comes to the power-packed features and specifications, Udoo NEO is nothing short of a desktop computer. With a Freescale i.MX 6SoloX applications processor with an embedded ARM Cortex-A9 core and a Cortex-M4 Core, Neo comes with 1GB RAM. The Micro HDMI port can be connected to an external display and audio sources. The standard Arduino pin layout is compatible with Arduino shields. You can install Node.js, Python, and even Java on Udoo Neo.

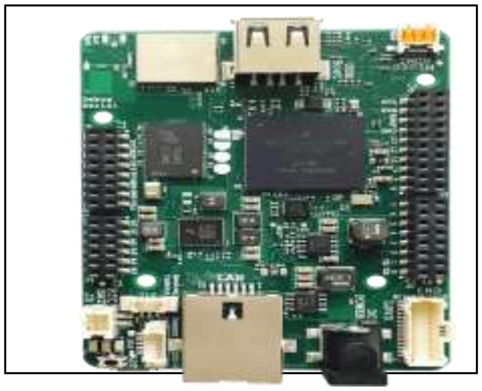


Fig 49.Udoo neo

8.Intel Edison

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

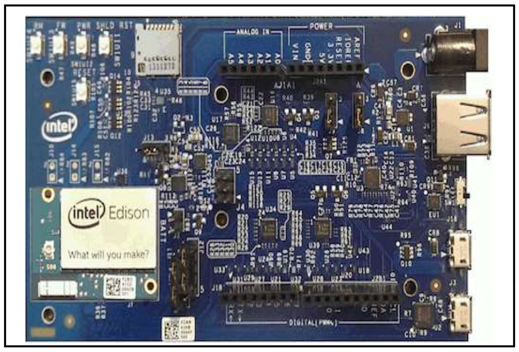


Fig 50.Intel Edison

9. Raspberry Pi

Raspberry Pi is undoubtedly the most popular platform used by many hobbyists and hackers. Even non-technical users depend on it for configuring their digital media systems and surveillance cameras. The recently launched Raspberry Pi 3 included built-in WiFi and Bluetooth making it the most compact and standalone computer. Based on a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor and 1GB RAM, the Pi is a powerful platform. The Raspberry Pi 3 is equipped with 2.4 GHz WiFi 802.11n and Bluetooth 4.1 in addition to the 10/100 Ethernet port. The HDMI port makes it further easy to hook up A/V sources.

Raspberry Pi runs on a customized Debian Linux called Raspbian, which provides an excellent user experience. For developers and hackers, it offers a powerful environment to install a variety of packages including Node.js, the LAMP stack, Java, Python and much more. With four USB ports and 40 GPIO pins, you can connect many peripherals and accessories to the Pi. 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.

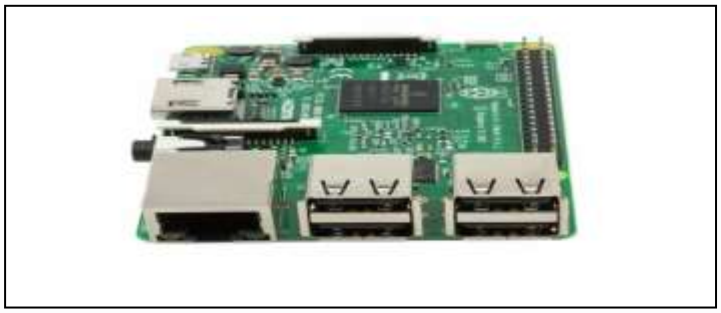


Fig 51.RaspBerry pi

10.Arduino Uno

Arduino Uno remains to be the top favorite of absolute beginners and experts. Considered to be one of the first microcontroller-based development boards, the Arduino Uno R3 is simplest yet the most powerful prototyping environment. It is based on the ATmega328P which has 14 digital input/output pins and six analog inputs. Though it comes with just 32 KB of Flash memory, it can accommodate code that deals with complex logic and operations.Arduino enjoys the best community participation and support. From sensors to actuators to libraries, it has a thriving ecosystem. The board layout has become almost the gold standard for microcontrollers. Almost every prototyping environment tries to be compatible with the Arduino pin breakout. The open source IDE to develop sketches is another reason for its popularity. With a simple syntax based on ‘C’ language, the code is easy to learn. If you are eager to learn basics of electronics and IoT, look no further. Do yourself a favor and get an Arduino Uno R3.

Fig 52.Arduino

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 INFORMATION TECHNOLOGY

UNIT - III

SITA3008 - Internet of Things

UNIT 3 COMMUNICATION AND CONNECTIVE TECHNOLOGIES

1. IoT Communication Model

2. Cloud computing in IoT

3. IoT in cloud architecture

4. Logging on to cloud, Selecting and Creating cloud service

5. Cloud based IoT platforms - IBM Watson, Google cloud.

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. The system is more complex, a number of potential challenges may stand in the way of developing smart IoT – particularly in the areas of security, privacy, interoperability and standards, legal, regulatory, and rights issues, as well as the inclusion of emerging economies.

The Internet of Things involves a complex and evolving set of technological, social and policy considerations across a diverse set of stakeholders.

IoT devices will be found everywhere and enable ambient intelligence in the future. The various communication models used in IOT, given below such as

i. **Device-to-Device communication model**
ii. **Device-to-Cloud communication model**
iii. **Device-to-Gateway model**
iv. **Back-end Data-sharing model**

The IoTs allow people and things to be connected anytime, anyplace, with anything and anyone, using any path/network and any service. From an operational perspective, it is useful to think about how IoT devices connect and communicate in terms of their technical communication models, before making decisions:

i. Device –to- Device communication model

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.

Device-to-device communication model for a number of devices in a wireless network.

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

ii. Device –to- Cloud communication model

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. 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.

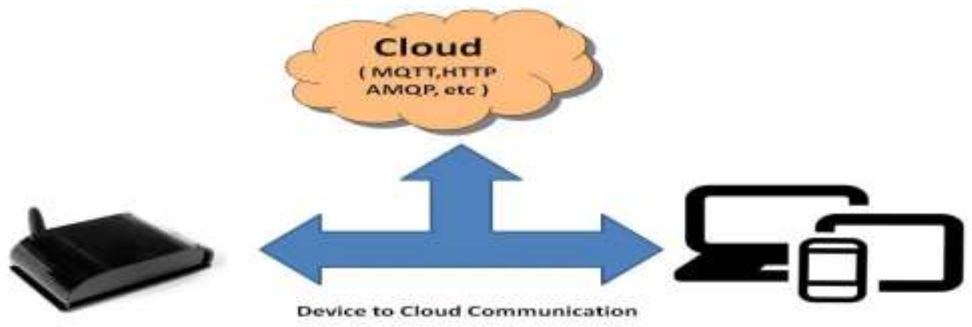


Fig 1 Device to Cloud Communication

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).

3.1.1 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.

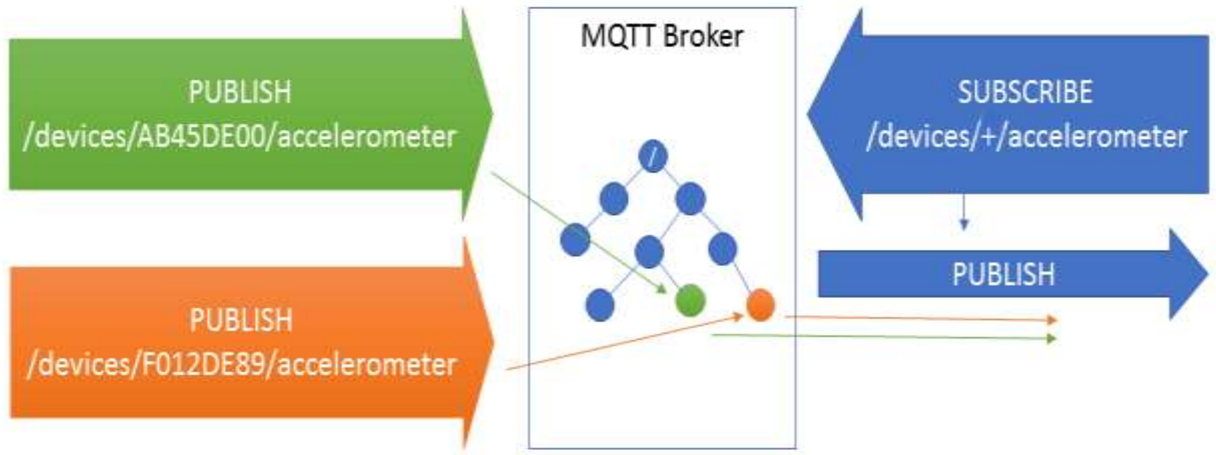


Fig 2 MQTT Protocol

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 nonconfirmable 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.

3.1.2 Attack Surfaces on Device to Cloud Communication

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.

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 unvalidated

or unencoded user input within the output it generates. By leveraging XSS, an attacker does not target a victim directly. Instead, an attacker would exploit a 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. **Username and password enumeration attacks**

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.

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.

iii. Device –to- Gateway communication model

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

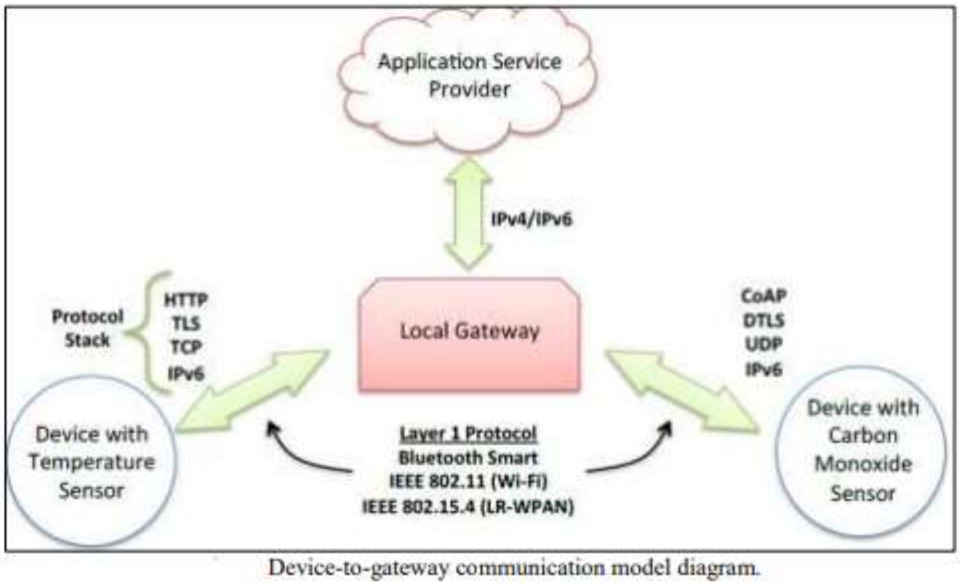

Application Service Provider
IPv4/IPv6
Protocol Stack
HTTP
TLS
TCP
IPv6
Local Gateway
CoAP
DTLS
UDP
IPv6
Device with Temperature Sensor
Layer 1 Protocol
Bluetooth Smart
IEEE 802.11 (Wi-Fi)
IEEE 802.15.4 (LR-WPAN)
Device with Carbon Monoxide Sensor
Device-to-gateway communication model diagram.

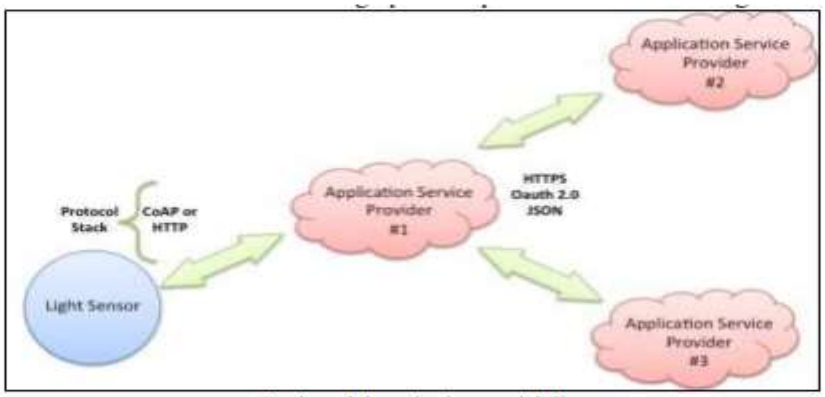
Fig 3 Device to gateway

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.

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.

Attacks include distributed denial-of-service attacks (DDoS) attacks, HTTP floods, SQL injections, cross-site scripting, parameter tampering, and Slowloris attacks. To combat these and more, most organizations have an arsenal of application layer security protections, such as web application firewalls (WAFs), secure web gateway services, and others

iv. The **back-end data-sharing model** refers to a communication architecture that enables users to export and analyze smart object data from a cloud service in combination with data from other sources. This architecture supports "the [user's] desire for granting access to the uploaded sensor data to third parties". This approach is an extension of the single device-to-cloud communication model, which can lead to data silos where "IoT devices upload data only to a single application service provider''. A back-end sharing architecture allows the data collected from single IoT device data streams to be aggregated and analyzed and suggests a federated cloud services approach, or cloud applications programmer interfaces (APIs), to achieve interoperability of smart device data hosted in the cloud.


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

Back-end data sharing model diagram.

Fig 4 Backend sharing model

3.2 IOT in Cloud Computing

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.

3.2.1 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.

3.3 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.

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:

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

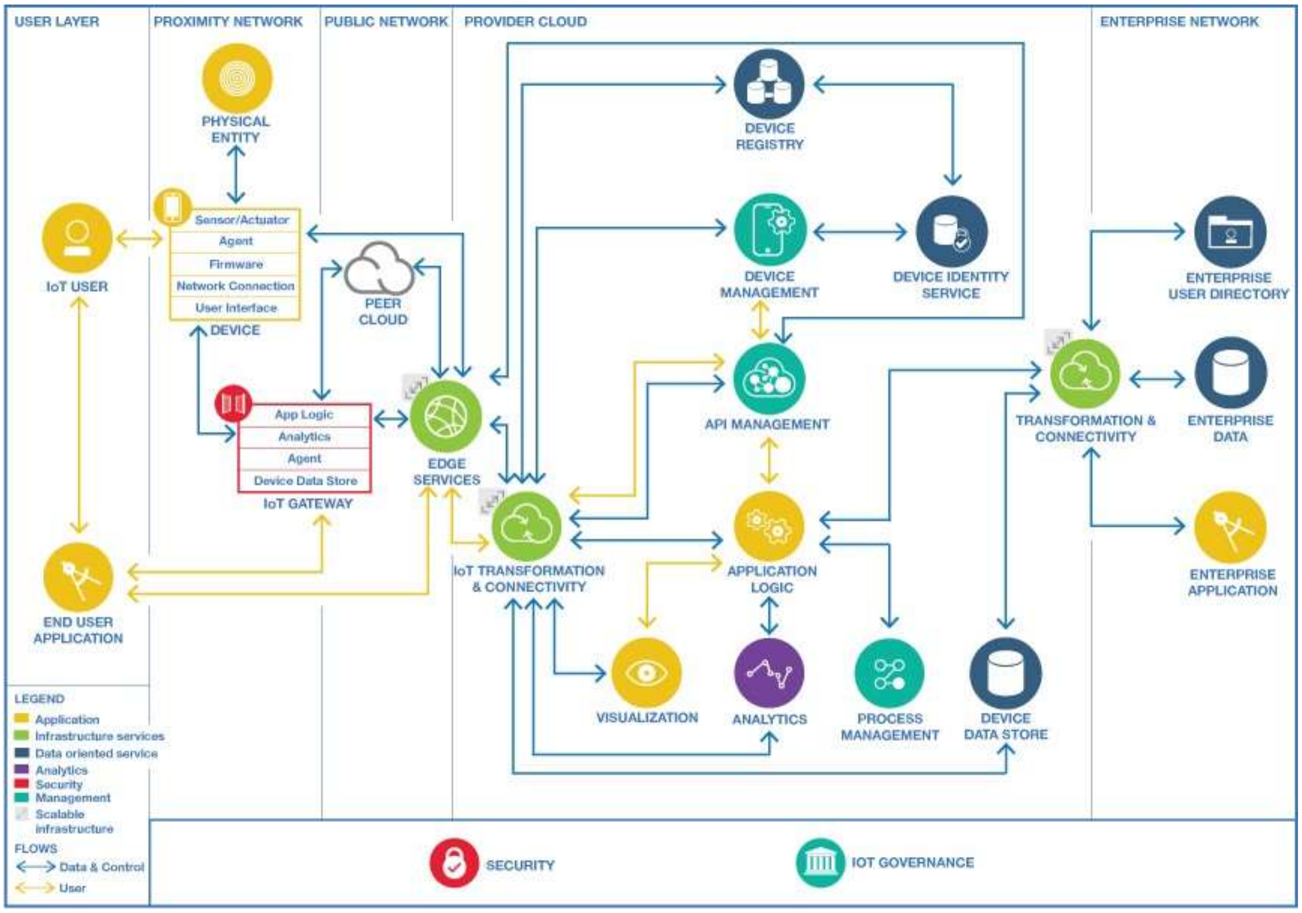


Fig 5 Cloud Components for IoT

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.

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 and

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.

3.4 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.

3.4.1 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.

3.4.2 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.

3.4.3 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.
- We 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.
- You 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.

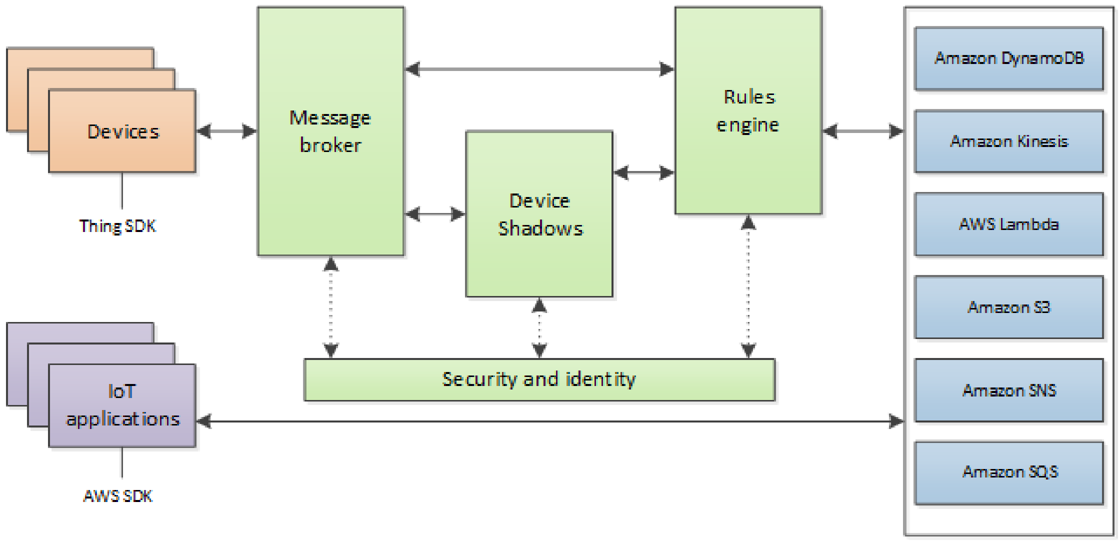


Fig 6 AWS IOT Architecture

- 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.

3.5 Managing Cloud Account Credentials

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

3.5.1 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**.

3.5.2 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**.

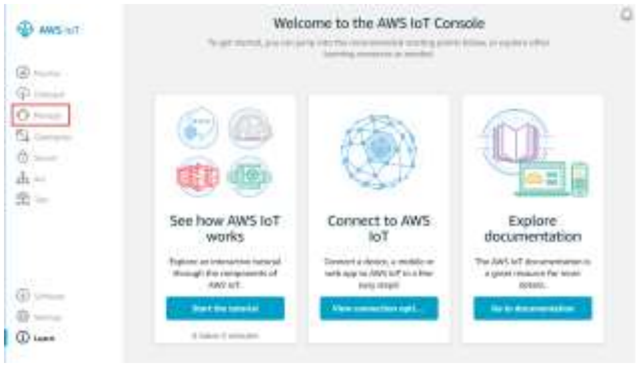


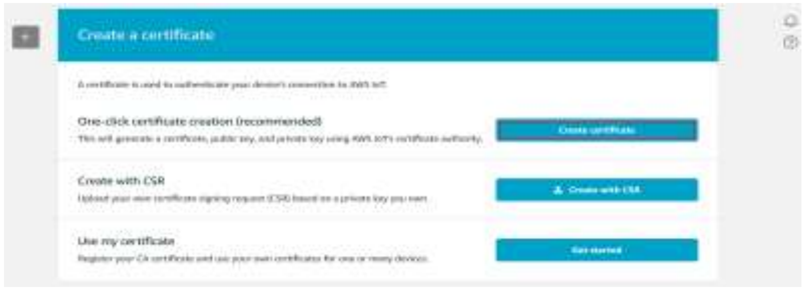
Fig 7 Registration

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.

3.5.3 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**.



Create a certificate

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)
This will generate a certificate, public key, and private key using AWS IoT's certificate authority.
Create certificate

Create with CSR
Upload your own certificate signing request (CSR) based on a private key you own.
Create with CSR

Use my certificate
Register your CA certificate and use your own certificates for one or many devices.
Get started

Fig 8 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.

3.5.4 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**.

3.5.5 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**.

3.5.6 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**.

3.5.7 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.

3.5.8 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.

3.5.9 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**.

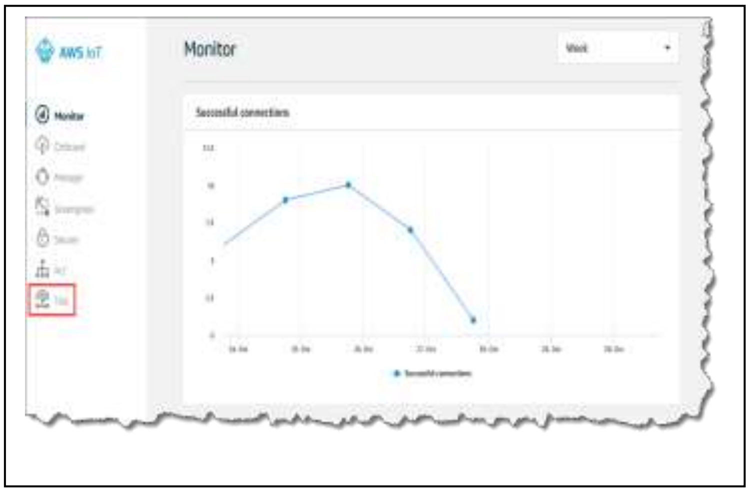


Fig 9 MQTT Messages

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).

3.5.8 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.

3.6. Cloud based IoT platforms

3.6.1 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.

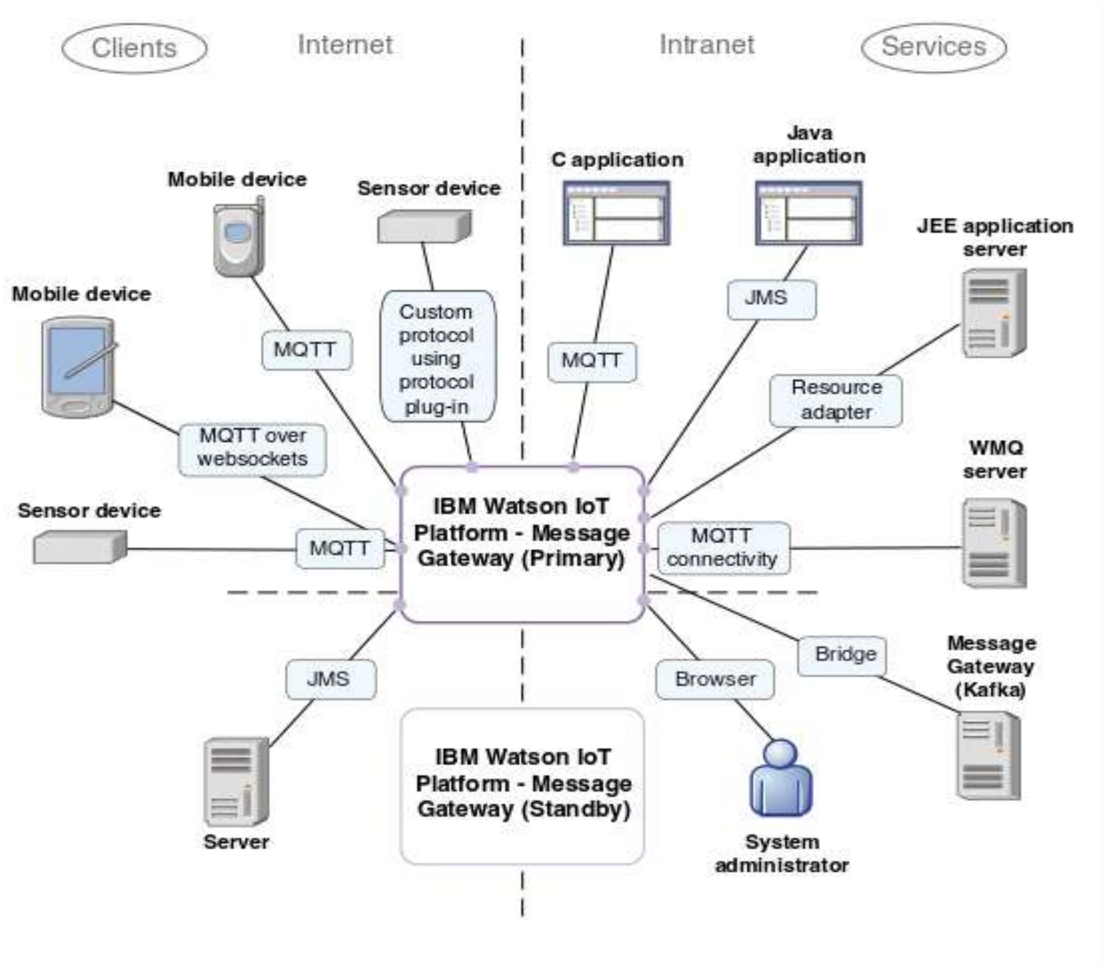


Fig 10 IBM Watson Platform

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.

3.6.2 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.

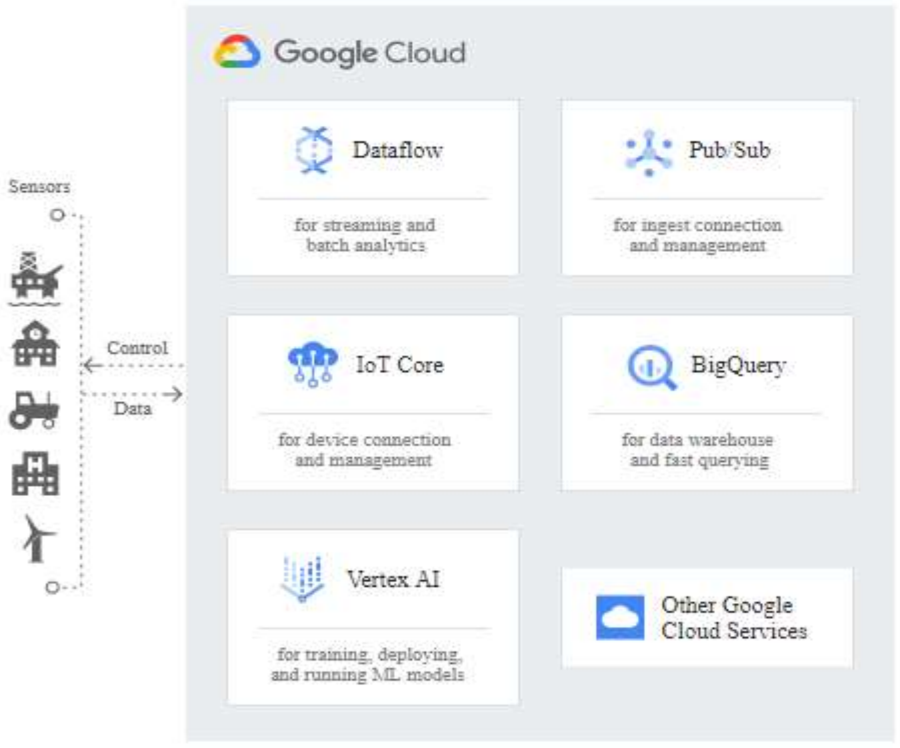

Google Cloud
Sensors
Control
Data
Dataflow
for streaming and batch analytics
Pub/Sub
for ingest connection and management
IoT Core
for device connection and management
BigQuery
for data warehouse and fast querying
Vertex AI
for training, deploying, and running ML models
Other Google Cloud Services

Fig 11 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

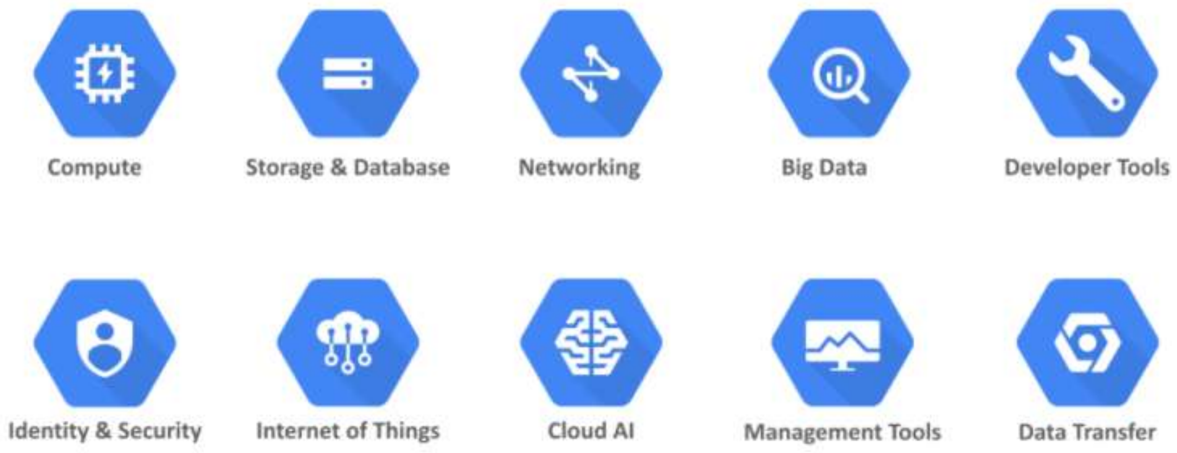
Perform fleet management, inventory tracking, cargo integrity monitoring, and other business-critical functions with Google Cloud IoT’s logistics solution.

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**



- 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

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 INFORMATION TECHNOLOGY

UNIT - IV

SITA3008 - Internet of Things

UNIT IV

1. **Big Data Analytics**
2. **Architecture**
3. **Apache Hadoop**
4. **Using Hadoop MapReduce for Batch Data Analysis**
5. **Apache Storm**
6. **Data Visualization and Visualization tools for IoT**

4.1 Big Data Analytics

Big Data Analytics is "the process of examining large data sets containing a variety of data types – i.e., Big Data – to uncover hidden patterns, unknown correlations, market trends, customer preferences, and other useful information." Companies and enterprises that implement Big Data Analytics often reap several business benefits, including more effective marketing campaigns, the discovery of new revenue opportunities, improved customer service delivery, more efficient operations, and competitive advantages. Companies implement Big Data Analytics because they want to make more informed business decisions. Big Data Analytics gives analytics professionals, such as data scientists and predictive modelers, the ability to analyze Big Data from multiple and varied sources, including transactional data

4.1.1 Characteristics

Big Data is often defined in terms of **"3V's"** i.e.

Volume - the amount of data generated, stored and analysed. The amount of data stored determines the level of insight that can be obtained from that data;

Variety - type and nature of data. Historically data was structured and from a single source - in which case it would fit readily into 'columns' and 'rows'. Increasingly data is sourced from a variety of sources with many different formats;

Velocity - the speed at which data is generated and processed. Where historically data could reasonably be expected to be uploaded via a daily 'batch' process now data is measured in thousands or even millions of transactions per minute.

Variability - Variations in the data sets. For example is a temperature measured in degrees Celsius, Fahrenheit or Kelvin;

Veracity - Quality of the captured data. Where decisions are being made on data you need to be sure that the data is correct.

4.1.2 Analytics

Analytics is the scientific process of discovering and communicating the meaningful patterns which can be found in data.

It is concerned with turning raw data into insight for making better decisions. Analytics relies on the application of statistics, computer programming, and operations research in order to quantify and gain insight to the meanings of data. It is especially useful in areas which record a lot of data or information.

4.1.3 Types

"**Descriptive**: A set of techniques for reviewing and examining the data set(s) to understand the data and analyze business performance.

Diagnostic: A set of techniques for determine what has happened and why

Predictive: A set of techniques that analyze current and historical data to determine what is most likely to (not) happen

Prescriptive: A set of techniques for computationally developing and analyzing alternatives that can become courses of action – either tactical or strategic – that may discover the unexpected

Decisive: A set of techniques for visualizing information and recommending courses of action to facilitate human decision-making when presented with a set of alternatives

4.1.4 Challenges for IoT Big Data

Some of the key challenges for IoT Big Data, which have a bearing on the design of architectures suitable for service delivery include

1. **The number of IoT devices**: With forecasted growth in the number of connected "things"

expected into the billions world-wide there will be masses of devices which may be a data source, and which may be subject to third party control;

2. **The variety of IoT device**s: There will be enormous variety in the devices which may provide data, even in the case of similar devices e.g. an electronic thermostat. Data from any individual device manufacturer or model may be quite dissimilar from that of nominally identical devices in such areas as field names, units, and data structures;

3. **Intelligence of IoT devices:** IoT devices have more and more compute resources and integrate several technologies like Graphics Processing Unit (GPU) and Solid State Drive (SSD) storage. Simple sensors are evolving to autonomous systems which will be able to manage their own analytics and be part of large analytics networks;

4. **Risk of IoT device malfunction:** With a great number of IoT devices and manufacturers it is reasonable to assume there will be many occasions where IoT devices malfunction in various ways. In the most drastic situations devices will fail completely but it should be expected that more subtle malfunctions will occur which might result in aberrations of data coming from those devices, or a failure of the device to perform a required control function;

5. **Update frequency**: Though some devices (e.g. remote sensors) will produce data reports at a low frequency there may be substantial quantities of data streaming from more sophisticated Internet connected things such as cars;

6. **Historical data**: It is expected that many Big Data insights will derive from historical data recorded from IoT devices. This may be processed alone to derive analytics/ intelligence or considered alongside current data particularly to enable smart monitoring and control;

7. **Context data**: Much IoT data will make more sense when put in context with other data. Context data might be generally "static" (or at least with a slow update period) such as geographical data, or could be more dynamic e.g. weather forecast data. Another important source of context data can be information gathered from the mobile networks themselves e.g. mobile user location or usage dynamics;

8. **Privacy issues:** With so many expected IoT devices acquiring data there could be a substantial risk relating to the disclosure of data which is considered personal to end users. When IoT data is stored in a Big Data system and made available to third parties there is a need to implement strong safeguards to ensure end users remain in control of their personal information. Mobile Network Operators (MNOs) are in a strong position to help users remain in control of

their data and to make data available in the best way via consent, aggregation or anonymisation.

4.2 General Architecture for IoT Big Data

4.2.1 Context Data Layer

This functional unit is concerned with obtaining external non IoT data ("Context data") which is either available to the third party application or used during the processing of IoT data e.g. "mashing up" IoT data with context data. The Context Data Layer is also able to communicate with the external data sources, e.g. to start and stop data feeds.

Examples of context data might include geographic/ mapping information, weather forecasts, schedules e.g. for transportation, or information generated from mobile networks/ users. This allows IoT data to be associated with further context data e.g. a moisture sensor reports the current moisture level whilst a weather forecast for the same geographical area identifies whether rain is predicted - allowing a decision to be made as to whether to water a crop.

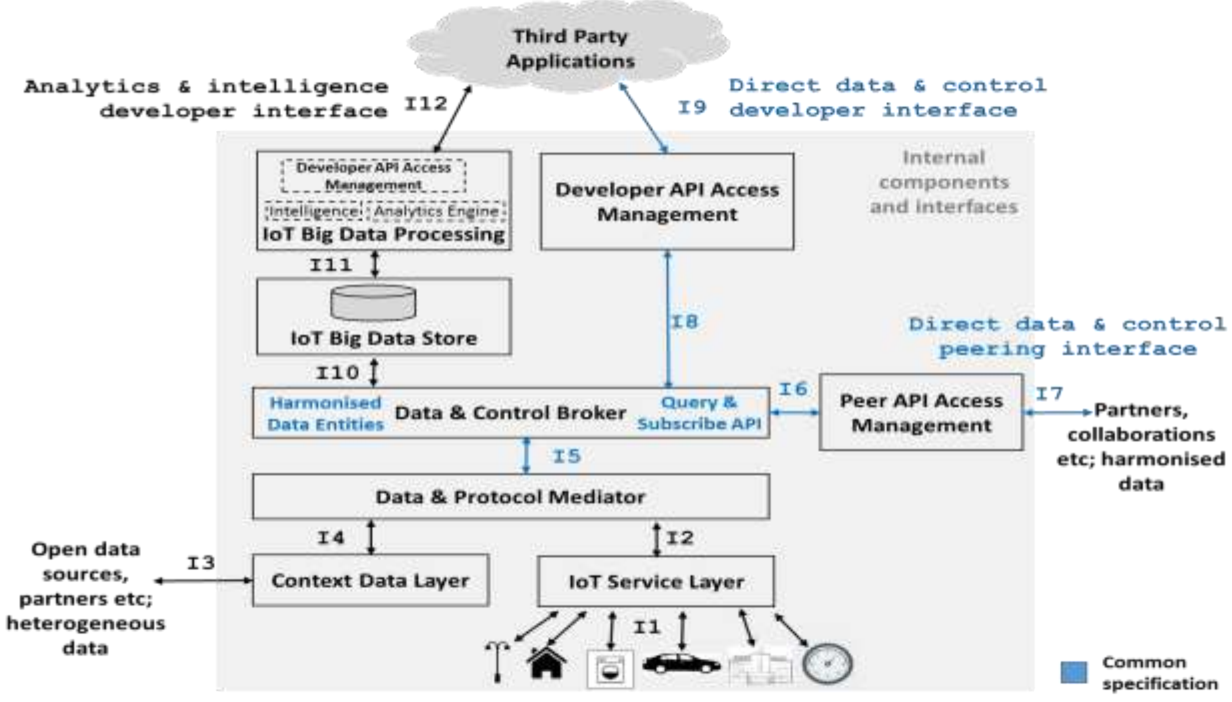

Third Party Applications
Analytics & intelligence developer interface
I12
I9
Direct data & control developer interface
Developer API Access Management
Intelligence
Analytics Engine
IoT Big Data Processing
Developer API Access Management
Internal components and interfaces
I11
IoT Big Data Store
I8
Direct data & control peering interface
I10
Harmonised Data Entities
Data & Control Broker
Query & Subscribe API
I6
Peer API Access Management
I7
Partners, collaborations etc; harmonised data
I5
Data & Protocol Mediator
I4
I2
Open data sources, partners etc; heterogeneous data
I3
Context Data Layer
IoT Service Layer
I1
Common specification

Fig 1: General Architecture for IoT Big Data

Context data might be received in different ways e.g. via Hypertext Transfer Protocol (HTTP) based APIs which request data from external servers, information received within an email, via batch file by making an outgoing File Transfer Protocol (FTP) request or by a batch file being deposited via FTP, or data received using removable media. This unit is principally concerned with implementing the relevant adapters in order to receive the various types of context data. The diagram below shows the general architecture for delivery of IoT Big Data services. This is explained in the following narrative.

4.2.2 IoT Service Layer

The IoT service layer is concerned with handling the device specific interactions required for obtaining data from IoT devices and sending control commands (where relevant) to those IoT devices. Therefore this layer is required to handle bi-directional communications both to IoT devices and to the upper layers of the architecture.

The IoT Service Layer is expected to handle the lower level interactions with IoT devices. Those devices might be connected using a variety of protocols and low level communication technologies including (but not limited to) oneM2M [3], Hypercat [4], Constrained Application Protocol (CoAP), MQ Telemetry Transport (MQTT), Real Time Streaming Protocol (RTSP), or device specific interfaces such as JavaScript Object Notation (JSON)/Extensible Markup Language (XML) over HTTP.

The IoT Service Layer is expected to handle authentication and security aspects regarding the interfacing with IoT devices.

4.2.3 Data and Protocol Mediator

The Data and Protocol Mediator is responsible for ingesting information from IoT devices as well as other external sources ("context data"). It ensures that data is transformed to the Harmonised Entity Definition before being stored and published by the Data & Control Broker. The harmonisation process itself may be partially implemented in the 'Context Data

Layer' function or the 'IoT Service Layer' function but the Data & Protocol Mediator will ensure that harmonisation is complete before data is exposed to the higher level parts of the architecture.

The harmonisation process includes:

- Conversion of payload encoding e.g. converting between an XML format payload of the IoT or context data and the JSON based structures defined in the Harmonised Entity Definitions;
- Mapping of the data structures and data fields between the lower level IoT device structures and fields and the Harmonised Entity Definitions e.g. the IoT device might store a temperature value in a field named 'temp' whereas the Harmonised Entity Definition in a field named 'currentTemperature';
- Unit conversion from the values and ranges of the lower level IoT devices to the Harmonised Entity Definition e.g.

  - The Harmonised Entity Definition might represent a switch as the Boolean true or false value whereas the IoT device could represent as the integer 1 or 0 respectively;
  - The Harmonised Entity Definition might represent a temperature in degrees Centigrade as a double precision float whereas the IoT device might record in degrees Fahrenheit.

- Data quality verification e.g. identifying a situation where an IoT sensor is apparently faulty such as delivering a sensor reading which is outside of the expected range. For example an outdoor temperature sensor transmitting a value which is significantly outside of the normal weather temperature range;
- Combining ("mash up") or linking relevant context data with IoT data e.g. associating a specific IoT sensor with its geographic location or weather forecast data to form a richer entity definition;
- Cross referencing related entity data e.g. different sensors with say the car to which they belong.

The Data & Protocol Mediator will also enable control requests to be processed - performing broadly a 'reverse' process compared with data harmonisation:

- Verifying the control request to make sure that the request is valid e.g.

Refers to a valid IoT device;

  - The control action is relevant to that IoT device e.g. a fixed position sensor cannot be asked to move to a different position;
  - The control action is valid according to the current state of the device/ system (which should be maintained by the control broker);
  - The parameter values supplied in the request are valid both in terms of individual parameter range and in combination.
- Transforming the high level control request into the equivalent device specific request payload e.g. generating an XML format payload if that is required by the IoT device;
- Mapping of the data structures and data fields in the control request between the high level structures and fields of the Harmonised Entity Definitions and the lower level IoT device structures and fields;

4.2.4 Data & Control Broker

Data & Control Broker is responsible for enabling third party application access to harmonised data entities through a query and subscribe API, allowing applications to gain access to such data in a standard way. The broker may store data in the short to medium term, coming from multiple devices and data sources via the Data and Protocol Mediator. This function also transforms control requests coming from the application layer to be passed onwards to the Data & Protocol Mediator.

The control process itself may be partially implemented in the 'IoT Service Layer' function but the Data & Control Broker in collaboration with the Data & Protocol Mediator will ensure responsibility for providing third party application access to control services in a consistent (harmonised) and controlled way. Control brokering will perform broadly a 'reverse' process compared with data harmonisation, receiving high level control requests from the third party application - normally formatted as a JSON based request communicated over HTTPS, and adapting this through the Data & Protocol Mediator and IoT Service Layer.

The Data & Control Broker is expected to have access to a data store which may act as a short to medium term buffer space for control actions or a short to medium term store for harmonised data entities. The expected use of this is:

- Retention of current instances of harmonised data entities processed from IoT devices and external sources (context data);
- Storage of control requests and any status information relevant to the request;
- Storage of a window of historical harmonised data entities that may be queried directly via the third party application. Note that it is expected that such a data store would be for short to medium term harmonised data entities, whereas longer term storage of harmonised data entities would be provided in the "IoT Big Data Store”;
- Storage of any results of Analytics and Intelligence results which become additional context data that can be queried or mashed up with other IoT data or external data sources.

It should be noted that the Data & Control Broker has the option of using its own internal database for data storage or the defined IoT Big Data Store function defined in this architecture i.e. some of the logically separate elements of the defined architecture may be physically implemented together.

4.2.5 Peer API Access Management

The Peer API Access Management function is responsible for interfacing with its peers in other organisations to receive and publish additional relevant harmonised IoT and context data. The policies applied to these trusted interfaces may be different to those applied to the main developer interface provided by the Developer API Access Management function. For example, organizations may be willing to share certain sensitive data with each other but require this sensitive data to be anonymized before being offered to third party developers. See sections 4.2.6 and 4.2.7 on I6 and I7 interfaces for more details.

4.2.6 Developer API Access Management

The Developer API Access Management function controls access to harmonized data entities, covering both IoT and context data, as well as control services to third party applications. It implements authentication, authorization and access control using industry standards to ensure privacy and security around the harmonized data. This function is mainly concerned with managing the privacy and security aspects of the data access by external parties. It is expected that this layer does not perform any actual IoT and context data processing or storage but is ensuring that data and control services from lower layers of the architecture are delivered in a properly managed way. It is assumed that any data processing/ complex queries/ analytics/ intelligence is the responsibility of the third party application.

The Developer API Access Management function access control for the harmonized data, it is expected to perform the following:

- Be responsible for presenting data & control services to third party applications via a RESTful based API over http^3. This interface shall use JSON based encoding of data using the Harmonized Entity Definitions for both data and control and use the NGSIv2 interface to support entity retrieval/ simple queries;
- Implement API access control (i.e. application level security) to ensure only known/ approved applications have access to IoT and context data and control services. Access control should be provided on a per application basis allowing granularity over which application should be able to access what IoT and context data and control functions;
- Implement any usage policies against applications accessing the APIs e.g. applying IP address based access rules or throttling rules to ensure there is relevant fair usage of the platform;
- Provide access to a publish/ subscribe service so that IoT and context data can be pushed to the third party application server as new data is received;
- Log API usage information e.g. number of API calls made by an application, number and type of entity data retrieved, number and type of control requests received.

4.2.7 IoT Big Data Store

The provision of Big Data Analytics and Intelligence is dependent on having access to the relevant mass of data from which the various insights can be obtained. This function provides data storage for this massive data and it may also provide short to medium term storage capabilities for use by the Data & Control Broker, depending on the specific implementation.

For IoT Big Data usage it is considered that the Data Store must be able to handle a data set greater than 50TB in size. For small scale deployments/ prototypes a Relational Database such as MySQL may support IoT data storage. However realistically a NoSQL or 'graph' database is considered more suitable for commercial 'Big Data' deployment particularly because the graph data model is richer and more versatile.

"Big Data" databases address needs such as:

- The need to store vast amounts of data (orders of magnitude higher than Relational Databases reasonably work to);
- Insights are obtained when exploring ad-hoc relationships between data;
- Data is arriving at such a rate that it is impossible to maintain an indexing process;
- Data are not tightly constrained into fixed table/ column formats.

The "Big Data" database is expected to be used to store the harmonised data entities received from the IoT devices and/or the external data sources. As it is expected there could be many millions of IoT devices generating data frequently, the required storage space may be vast (i.e. of the order of many terabytes to many petabytes of data). It is expected the "Big Data" database could be implemented using products such as Apache Cassandra, Apache Hadoop, MongoDB, Neo4j, Titan or DynamoDB. To achieve high performance the database component may employ substantial quantities of memory to hold copies of data that is persistently stored on "hard disk".

4.2.8 IoT Big Data Processing

The processing of stored IoT data to perform analytics and intelligence is identified as the responsibility of the IoT Big Data Processing function. The IoT Big Data Processing function also provides related Developer API Access Management to control access to the intelligence and analytics by implementing authentication, authorisation and access control to ensure privacy and security. A broad division is made between analytics and intelligence. In practice both analytics and intelligence will be processing subsets of the mass of IoT data retained in the IoT Big Data Store.

- Analytics - principally involves relatively conventional methods (by human analysts and normal programming techniques) of exploring links and statistical relationships between data and then the analytics engine will produce its output based on the execution of a defined process;
- Intelligence - encompassing machine learning / artificial intelligence, it would be expected that algorithms 'adapt' to the observed data and the match between predicted and desired outcomes.

The outputs from Analytics and Intelligence are expected to be in a wide range of different formats, many of which will not conform to a uniform 'API' based approach e.g. the generation of a PDF report or the generation of a data set to be FTP'd to the third party developer's platform.

Relevant products for Analytics & Intelligence provision include:

- **Apache Spark**

  Apache Spark[15] is a powerful data processing system based upon Cassandra or Hadoop[16] for the data storage component and provides several powerful tools for building applications around it such as an SQL interface, graph data library and a job server.

  Spark is not a complete solution out of the box, but does provide a powerful big data platform with great performance. Spark is considered the leading solution for high performance Big Data analytics.

  A Spark solution could be delivered over a RESTful interface or a websockets connection (for better notification and real time services). More usually however developers would use the standard programming interfaces available to Java, Python, Scala and R programming languages.

- **Apache TinkerPop3 + Titan + Elastic Search + Gremlin**

  Titan provides Casandra backends, integration with Elastic search[17], Apache Lucene[18] / Solr[19], Spark and others which allows it to support Geo searches, full text searches, graph traversals and regular 'SQLesque' queries making it ideal for the IoT Big Data project.

Apache TinkerPop3[20] is a graph computing framework which is seeing a rapid adoption in data driven applications. Many projects are seeking to incorporate the TinkerPop specification into their interfaces for interoperability in graph databases and servers. There are several implementations of graph databases which expose a Gremlin[21] querying interface which makes it easier to query the graph database. Two such databases are Titan and Google Cayley.

- **Apache Mahout**

Mahout is designed for the development of high performance and scalable machine learning applications. It builds for example on top of Apache Spark / Hadoop and supports a range of machine learning algorithms. Uses include

Collaborative filtering – mines user behaviour and makes product recommendations (e.g. Amazon recommendations);
Clustering – takes items in a particular class (such as web pages or newspaper articles) and organizes them into naturally occurring groups, such that items belonging to the same group are similar to each other;
Classification – learns from existing categorizations and then assigns unclassified items to the best category;
Frequent itemset mining – analyses items in a group (e.g. items in a shopping cart or terms in a query session) and then identifies which items typically appear together.

- **Tensorflow**

Another open source set of tools for machine learning - developed originally by the Google Brain Team to support advances in search ranking algorithms as well as other Google research activities.

Tensorflow[23] could be used for example in IoT applications such as developing high accuracy automated number plate recognition algorithms based on images captured from CCTV cameras. This can then be applied in the IoT Big Data system to applications such as security, congestion or traffic planning.

Tensorflow can also be coupled with Apache Spark which is used to obtain the select the data from the IoT Big Data store to use with the tensorflow algorithms.

4.3 Big Data Analytical Tools Classification

- **Data Storage and Management**
- **Data Cleaning**
- **Data Mining**
- **Data Analysis**

4.3.1 Data Storage and Management

4.3.1.1 Hadoop

Apache Hadoop[6] is a highly scalable storage platform designed to process very large data sets across hundreds to thousands of computing nodes that operate in parallel. It provides a very cost effective storage solution for large data volumes with no particular format requirements. MapReduce [6] is the programming paradigm that allows for this massive scalability, is at the heart of Hadoop. The term MapReduce refers to two separate and distinct tasks that Hadoop programs perform. Hadoop has two main components - HDFS and YARN.

HDFS – the Hadoop Distributed File System is a distributed file system designed to run on commodity hardware. It differs from other distributed file systems in that HDFS is highly fault- tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets.

YARN - YARN is a large-scale, distributed operating system for big data applications that runs on top of HDFS. It provides a framework for job scheduling and cluster resource management.

4.3.1.2 Cassandra

Cassandra is a scalable database for large scale data storage from the Apache foundation and is used by many of the world's leading tech companies including github, Netflix, Apple and Instagram. The largest known deployment of Cassandra contains 75000 nodes (cloud servers) and stores over 10PB (Petabytes) of data. Cassandra is a NoSQL data store, which provides a robust means of storing data which spans many nodes, however it does not provide a very powerful query interface; it's highly inefficient to query on anything other than Cassandra's equivalent of a 'primary key'. Several solutions can be combined with Cassandra to provide a more powerful query interface. Apache Spark is one of the most powerful of these.

4.3.1.3 Cloudera

Cloudera is essentially a brand name for Hadoop with some extra services stuck on. They can help your business build an enterprise data hub, to allow people in your organization better access to the data you are storing. While it does have an open source element, Cloudera is mostly and enterprise solution to help businesses manage their Hadoop ecosystem. Essentially, they do a lot of the hard work of administering Hadoop for you. They will also deliver a certain amount of data security, which is highly important if you're storing any sensitive or personal data.

4.3.1.4 MongoDB

MongoDB[9] is a hybrid open source and closed source database, where the core of the database is available freely on an open source license, although some features which may be required on larger commercial deployments are commercially supported add-ons. This model has made MongoDB arguably one of the most popular document oriented databases in use today. A 'document' in MongoDB is a 'binary' representation of a JSON document. This allows arbitrary JSON encoded data to be stored in the database and then queried using a rich JSON based querying interface.

4.3.1.5 Graph Databases

Other databases such as Neo4J[10] or Titan[11] are a powerful way for structuring data which allows for easily traversing relationships as well as retrieving attributes about a particular node. It is worth clarifying that a Graph Database works efficiently where there are ad-hoc relationships between data whereas a Relational Database is efficient for more structured relationships between data. The key strength of these systems is that they're very well adapted for traversing different data types to perform ad-hoc mash-ups.

4.3.2 Data Cleaning Tool

4.3.2.1 OpenRefine

OpenRefine (formerly GoogleRefine) is an open source tool that is dedicated to cleaning messy data. You can explore huge data sets easily and quickly even if the data is a little unstructured. As far as data softwares go, OpenRefine is pretty user-friendly. Though, a good knowledge of data cleaning principles certainly helps. The nice thing about OpenRefine is that it has a huge community with lots of contributors meaning that the software is constantly getting better and better.

4.3.2.2 Data Cleaner

DataCleaner recognises that data manipulation is a long and drawn out task. Data visualization tools can only read nicely structured, "clean" data sets. DataCleaner does the hard work for you and transforms messy semi-structured data sets into clean readable data sets that all of the visualization companies can read. DataCleaner also offers data warehousing and data management services. The company offers a 30-day free trial and then after that a monthly subscription fee.

4.3.3 Data Mining Tool

4.3.3.1 IBM SPSS Modeler

The IBM SPSS Modeler offers a whole suite of solutions dedicated to data mining. This includes text analysis, entity analytics, decision management and optimization. Their five products provide a range of advanced algorithms and techniques that include text analytics, entity analytics, decision management and optimization. SPSS Modeler is a heavy-duty solution that is well suited for the needs of big companies. It can run on virtually any type of database and you can integrate it with other IBM SPSS products such as SPSS collaboration and deployment services and the SPSS Analytic server.

4.3.3.2 Oracle data mining

Another big hitter in the data mining sphere is Oracle. As part of their Advanced Analytics Database option, Oracle data mining allows its users to discover insights, make predictions and leverage their Oracle data. You can build models to discover customer behavior, target best customers and develop profiles. The Oracle Data Miner GUI enables data analysts, business analysts and data scientists to work with data inside a database using a rather elegant drag and drop solution. It can also create SQL and PL/SQL scripts for automation, scheduling and deployment throughout the enterprise.

4.3.3.3 Framed Data

If you’re after a specific type of data mining there are a bunch of startups which specialize in helping businesses answer tough questions with data. If you’re worried about user churn, we recommend Framed Data, a startup which analyzes your analytics and tell you which customers are about to abandon your product.

4.3.4 Data Analysis Tool

4.3.4.1 Qubole

Qubole simplifies, speeds and scales big data analytics workloads against data stored on AWS, Google, or Azure clouds. They take the hassle out of infrastructure wrangling. Once the IT policies are in place, any number of data analysts can be set free to collaboratively “click to query” with the power of Hive, Spark, Presto and many others in a growing list of data processing engines. Qubole is an enterprise level solution. They offer a free trial that you can sign up to at this page.The flexibility of the program really does set it apart from the rest as well as being the most accessible of the platforms.

4.3.4.2 BigML

BigML is attempting to simplify machine learning. They offer a powerful Machine Learning service with an easy-to-use interface for you to import your data and get predictions out of it. You can even use their models for predictive analytics. A good understanding of modeling is certainly helpful, but not essential, if you want to get the most from BigML. They have a free version of the tool that allows you to create tasks that are under 16mb as well as having a pay as you go plan and a virtual private cloud that meet enterprise-grade requirements.

4.3.4.3 Statwing

Statwing takes data analysis to a new level providing everything from beautiful visuals to complex analysis. They have a particularly cool blog post on NFL data! It’s so simple to use that you can actually get started with Statwing in under 5 minutes. This allows you to use unlimited datasets of up to 50mb in size each. There are other enterprise plans that give you the ability to upload bigger datasets.

4.4 Apache Hadoop

Apache Hadoop is one of the main supportive element in Big Data technologies. It simplifies the processing of large amount of structured or unstructured data in a cheap manner. Hadoop is an open source project from apache that is continuously improving over the years.

"Hadoop is basically a set of software libraries and frameworks to manage and process big amount of data from a single server to thousands of machines

It provides an efficient and powerful error detection mechanism based on application layer rather than relying upon hardware." In December 2012 apache releases Hadoop 1.0.0, more information and installation guide can be found at Apache Hadoop Documentation. Hadoop is not a single project but includes a number of other technologies in it.

4.4.1 Hadoop

1. Hadoop is an open source framework that supports the processing of large data sets in a distributed computing environment.

2. Hadoop consists of MapReduce, the Hadoop distributed file system (HDFS) and a number of related projects such as Apache Hive, HBase and Zookeeper. MapReduce and Hadoop distributed file system (HDFS) are the main component of Hadoop.

3. Apache Hadoop is an open-source, free and Java based software framework offers a powerful distributed platform to store and manage Big Data.

4. It is licensed under an Apache V2 license.

5. It runs applications on large clusters of commodity hardware and it processes thousands of terabytes of data on thousands of the nodes. Hadoop is inspired from Google's MapReduce and Google File System (GFS) papers. 6. The major advantage of Hadoop framework is that it provides reliability and high availability.

4.4.2 Use of Hadoop

There are many advantages of using Hadoop:

1. Robust and Scalable – We can add new nodes as needed as well modify them.

2. Affordable and Cost Effective – We do not need any special hardware for running Hadoop. We can just use commodity server.

3. Adaptive and Flexible – Hadoop is built keeping in mind that it will handle structured and unstructured data.

4. Highly Available and Fault Tolerant – When a node fails, the Hadoop framework automatically fails over to another node

4.4.3 Core Hadoop Components

There are two major components of the Hadoop framework and both of them does two of the important task for it.

1. **Hadoop MapReduce** is the method to split a larger data problem into smaller chunk and distribute it to many different commodity servers. Each server have their own set of resources and they have processed them locally. Once the commodity server has processed the data they send it back collectively to main server. This is effectively a process where we process large data effectively and efficiently

2. **Hadoop Distributed File System (HDFS)** is a virtual file system. There is a big difference between any other file system and Hadoop. When we move a file on HDFS, it is automatically split into many small pieces. These small chunks of the file are replicated and stored on other servers (usually 3) for the fault tolerance or high availability.

3. **Namenode:** Namenode is the heart of the Hadoop system. The NameNode manages the file system namespace. It stores the metadata information of the data blocks. This metadata is stored permanently on to local disk in the form of namespace image and edit log file. The NameNode also knows the location of the data blocks on the data node. However the NameNode does not store this information persistently. The NameNode creates the block to DataNode mapping when it is restarted. If the NameNode crashes, then the entire Hadoop system goes down. Read more about Namenode

4. **Secondary Namenode**: The responsibility of secondary name node is to periodically copy and merge the namespace image and edit log. In case if the name node crashes, then the namespace image stored in secondary NameNode can be used to restart the NameNode.

5. **DataNode:** It stores the blocks of data and retrieves them. The DataNodes also reports the blocks information to the NameNode periodically.

6. **Job Tracker:** Job Tracker responsibility is to schedule the client's jobs. Job tracker creates map and reduce tasks and schedules them to run on the DataNodes (task trackers). Job Tracker also checks for any failed tasks and reschedules the failed tasks on another DataNode. Job tracker can be run on the NameNode or a separate node.

7. **Task Tracker:** Task tracker runs on the DataNodes. Task trackers responsibility is to run the map or reduce tasks assigned by the NameNode and to report the status of the tasks to the NameNode.

Besides above two core components Hadoop project also contains following modules as well.

1. Hadoop Common: Common utilities for the other Hadoop modules

2. Hadoop Yarn: A framework for job scheduling and cluster resource management

Traditional Enterprise Systems normally have a centralized server to store and process data. The following illustration depicts a schematic view of a traditional enterprise system. Traditional model is certainly not suitable to process huge volumes of scalable data and cannot be accommodated by standard database servers. Moreover, the centralized system creates too much of a bottleneck while processing multiple files simultaneously.

Google solved this bottleneck issue using an algorithm called MapReduce. MapReduce divides a task into small parts and assigns them to many computers. Later, the results are collected at one place and integrated to form the result dataset.

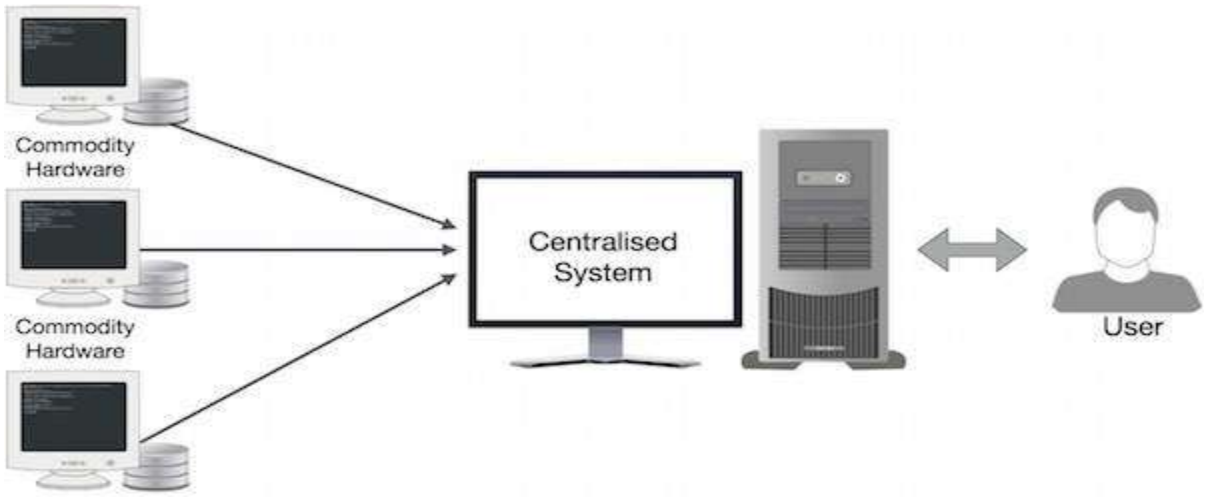


Fig 2 Map Reduce

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

- The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).
- The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples.

The reduce task is always performed after the map job. Let us now take a close look at each of the phases and try to understand their significance.

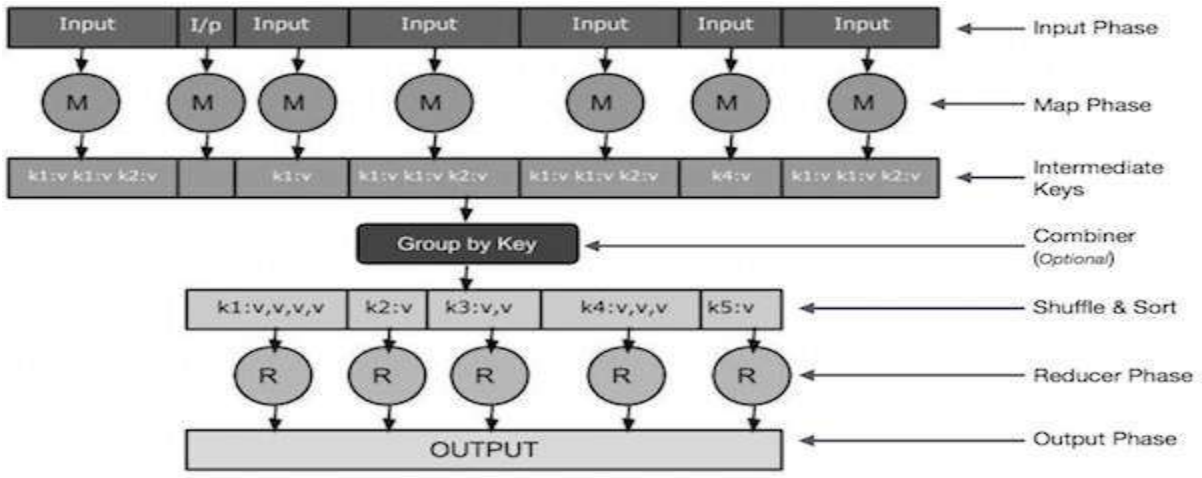


Fig 3 Map Reduce mechanism

- **Input Phase** − Here we have a Record Reader that translates each record in an input file and sends the parsed data to the mapper in the form of key-value pairs.
- **Map** − Map is a user-defined function, which takes a series of key-value pairs and processes each one of them to generate zero or more key-value pairs.
- **Intermediate Keys** − They key-value pairs generated by the mapper are known as intermediate keys.
- **Combiner** − A combiner is a type of local Reducer that groups similar data from the map phase into identifiable sets. It takes the intermediate keys from the mapper as input and applies a user-defined code to aggregate the values in a small scope of one mapper. It is not a part of the main MapReduce algorithm; it is optional.
- **Shuffle and Sort** − The Reducer task starts with the Shuffle and Sort step. It downloads the grouped key-value pairs onto the local machine, where the Reducer is running. The individual key-value pairs are sorted by key into a larger data list. The data list groups the equivalent keys together so that their values can be iterated easily in the Reducer task.
- **Reducer** − The Reducer takes the grouped key-value paired data as input and runs a Reducer function on each one of them. Here, the data can be aggregated, filtered, and combined in a number of ways, and it requires a wide range of processing. Once the execution is over, it gives zero or more key-value pairs to the final step.
- **Output Phase** − In the output phase, we have an output formatter that translates the final key-value pairs from the Reducer function and writes them onto a file using a record writer.

Let us try to understand the two tasks Map &f Reduce with the help of a small diagram −

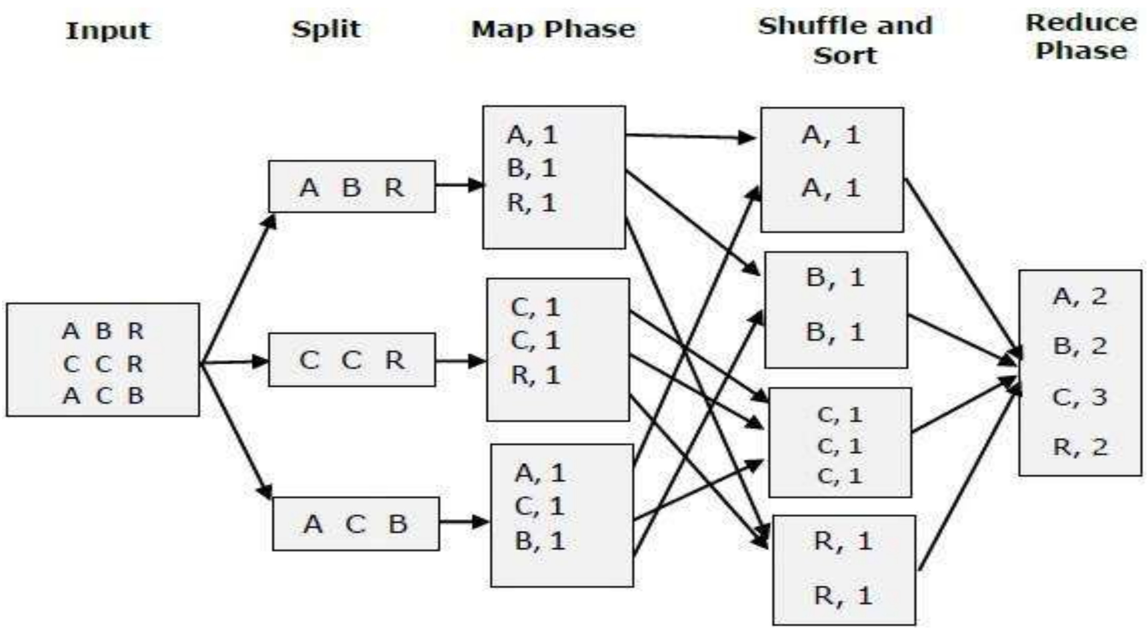


Fig 4 Map Reduce Task

4.4.4 MapReduce-Example

Let us take a real-world example to comprehend the power of MapReduce. Twitter receives around 500 million tweets per day, which is nearly 3000 tweets per second. The following illustration shows how Tweeter manages its tweets with the help of MapReduce.

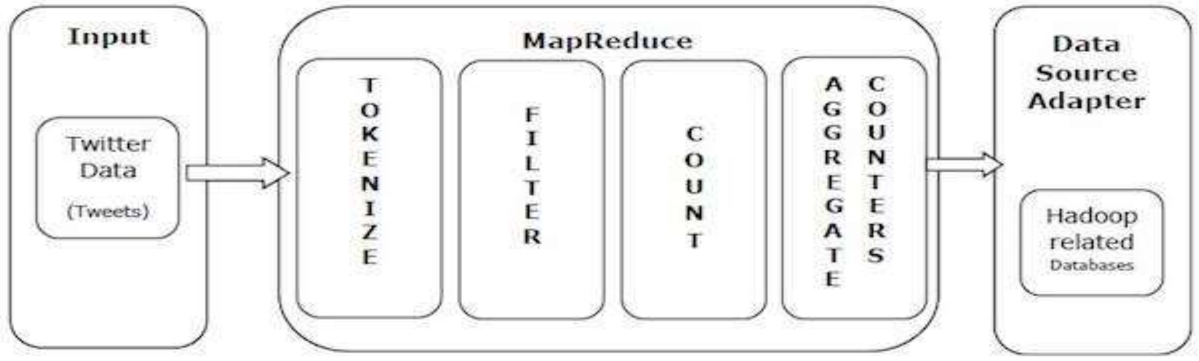


Fig 5 Map Reduce Algorithm

As shown in the illustration, the MapReduce algorithm performs the following actions −

- **Tokenize** − Tokenizes the tweets into maps of tokens and writes them as key-value pairs.

- **Filter** − Filters unwanted words from the maps of tokens and writes the filtered maps as key-value pairs.
- **Count** − Generates a token counter per word.
- **Aggregate Counters** − Prepares an aggregate of similar counter values into small manageable units.

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

- The map task is done by means of Mapper Class
- The reduce task is done by means of Reducer Class.

Mapper class takes the input, tokenizes it, maps and sorts it. The output of Mapper class is used as input by Reducer class, which in turn searches matching pairs and reduces them.

MapReduce implements various mathematical algorithms to divide a task into small parts and assign them to multiple systems. In technical terms, MapReduce algorithm helps in sending the Map & Reduce tasks to appropriate servers in a cluster.

4.5 Apache Storm

Storm was originally created by **Nathan Marz** and team at **BackType**. BackType is a social analytics company. Later, Storm was acquired and open-sourced by **Twitter**. In a short time, Apache Storm became a standard for distributed real-time processing system that allows you to process large amount of data, similar to Hadoop. Apache Storm is written in Java and Clojure. It is continuing to be a leader in real-time analytics. This tutorial will explore the principles of Apache Storm, distributed messaging, installation, creating Storm topologies and deploy them to a Storm cluster, workflow of Trident, real-time applications and finally concludes with some useful examples.

Apache Storm is a distributed real-time big data-processing system. Storm is designed to process vast amount of data in a fault-tolerant and horizontal scalable method. It is a streaming data framework that has the capability of highest ingestion rates. Though Storm is stateless, it manages distributed environment and cluster state via Apache ZooKeeper. It is simple and you can execute all kinds of manipulations on real-time data in parallel.

Apache Storm is continuing to be a leader in real-time data analytics. Storm is easy to setup, operate and it guarantees that every message will be processed through the topology at least once.

Basically Hadoop and Storm frameworks are used for analyzing big data. Both of them complement each other and differ in some aspects. Apache Storm does all the operations except persistency, while Hadoop is good at everything but lags in real-time computation. The following table compares the attributes of Storm and Hadoop.

Storm	**Hadoop**
Real-time stream processing	Batch processing
Stateless	Stateful
Master/Slave architecture with ZooKeeper based coordination. The master node is called as **nimbus** and slaves are **supervisors**.	Master-slave architecture with/without ZooKeeper based coordination. Master node is **job tracker** and slave node is **task tracker**.
A Storm streaming process can access tens of thousands messages per second on cluster.	Hadoop Distributed File System (HDFS) uses MapReduce framework to process vast amount of data that takes minutes or hours.
Storm topology runs until shutdown by the user or an unexpected unrecoverable failure.	MapReduce jobs are executed in a sequential order and completed eventually.

Both are distributed and fault-tolerant	
If nimbus / supervisor dies, restarting makes it continue from where it stopped, hence nothing gets affected.	If the JobTracker dies, all the running jobs are lost.

Use-Cases of Apache Storm

Apache Storm is very famous for real-time big data stream processing. For this reason, most of the companies are using Storm as an integral part of their system. Some notable examples are as follows −

Twitter − Twitter is using Apache Storm for its range of “Publisher Analytics products”. “Publisher Analytics Products” process each and every tweets and clicks in the Twitter Platform. Apache Storm is deeply integrated with Twitter infrastructure.

NaviSite − NaviSite is using Storm for Event log monitoring/auditing system. Every logs generated in the system will go through the Storm. Storm will check the message against the configured set of regular expression and if there is a match, then that particular message will be saved to the database.

Wego − Wego is a travel metasearch engine located in Singapore. Travel related data comes from many sources all over the world with different timing. Storm helps Wego to search real-time data, resolves concurrency issues and find the best match for the end-user.

4.5.1 Apache Storm Benefits

Here is a list of the benefits that Apache Storm offers −

- Storm is open source, robust, and user friendly. It could be utilized in small companies as well as large corporations.
- Storm is fault tolerant, flexible, reliable, and supports any programming language.
- Allows real-time stream processing.
- Storm is unbelievably fast because it has enormous power of processing the data.

- Storm can keep up the performance even under increasing load by adding resources linearly. It is highly scalable.
- Storm performs data refresh and end-to-end delivery response in seconds or minutes depends upon the problem. It has very low latency.
- Storm has operational intelligence.
- Storm provides guaranteed data processing even if any of the connected nodes in the cluster die or messages are lost.
- Apache Storm reads raw stream of real-time data from one end and passes it through a sequence of small processing units and output the processed / useful information at the other end.
- The following diagram depicts the core concept of Apache Storm.

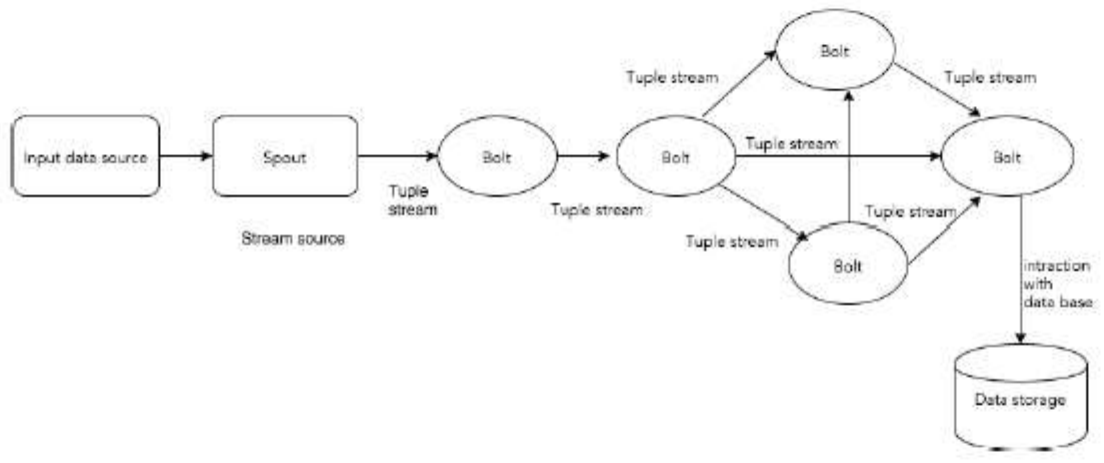


Fig 6 Apache Storm

Let us now have a closer look at the components of Apache Storm −

Components	**Description**
Tuple	Tuple is the main data structure in Storm. It is a list of ordered elements. By default, a Tuple supports all data types. Generally, it is modelled as a

	set of comma separated values and passed to a Storm cluster.
Stream	Stream is an unordered sequence of tuples.
Spouts	Source of stream. Generally, Storm accepts input data from raw data sources like Twitter Streaming API, Apache Kafka queue, Kestrel queue, etc. Otherwise you can write spouts to read data from datasources. “ISpout" is the core interface for implementing spouts. Some of the specific interfaces are IRichSpout, BaseRichSpout, KafkaSpout, etc.
Bolts	Bolts are logical processing units. Spouts pass data to bolts and bolts process and produce a new output stream. Bolts can perform the operations of filtering, aggregation, joining, interacting with data sources and databases. Bolt receives data and emits to one or more bolts. “IBolt” is the core interface for implementing bolts. Some of the common interfaces are IRichBolt, IBasicBolt, etc.

Let’s take a real-time example of “Twitter Analysis” and see how it can be modelled in Apache Storm. The following diagram depicts the structure.

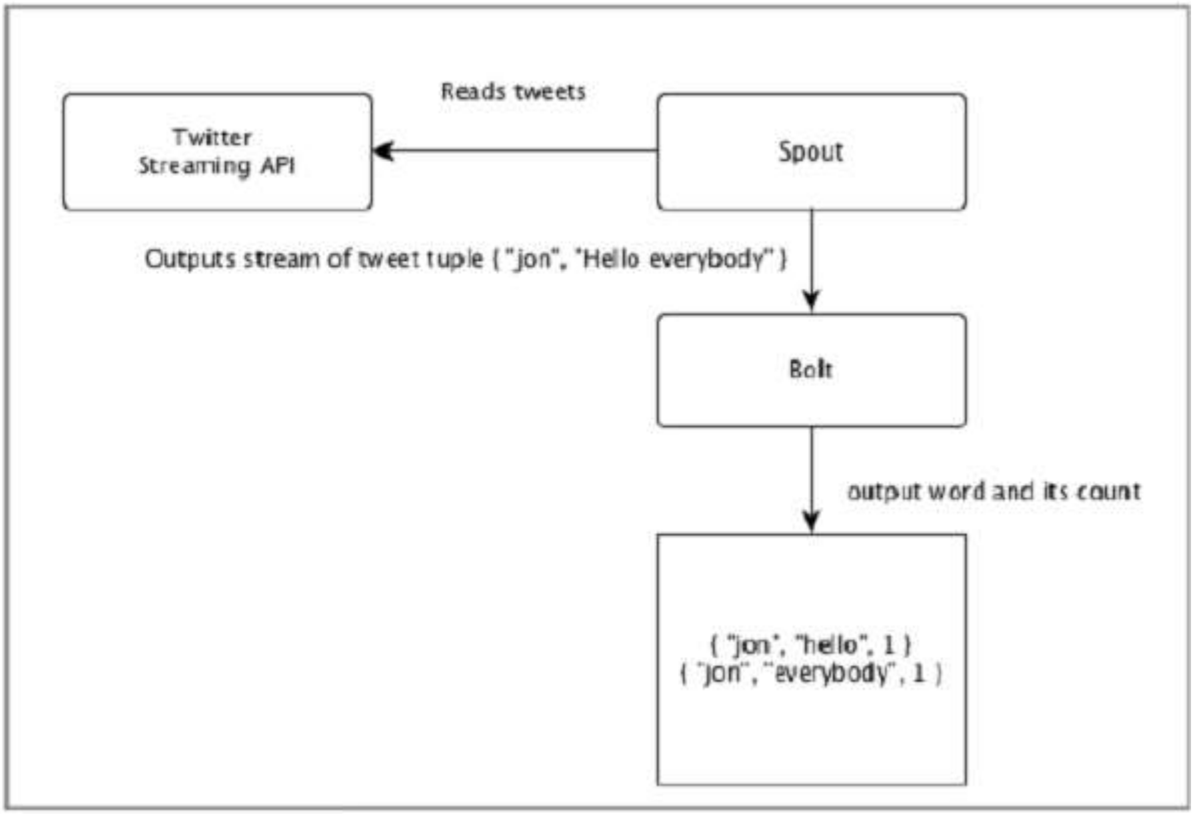

Reads tweets
Twitter Streaming API
Spout
Outputs stream of tweet tuple { "jon", "Hello everybody" }
Bolt
output word and its count
{ "jon", "hello", 1 }
{ "jon", "everybody", 1 }

Fig 7 Twitter Analysis

The input for the “Twitter Analysis” comes from Twitter Streaming API. Spout will read the tweets of the users using Twitter Streaming API and output as a stream of tuples. A single tuple from the spout will have a twitter username and a single tweet as comma separated values. Then, this steam of tuples will be forwarded to the Bolt and the Bolt will split the tweet into individual word, calculate the word count, and persist the information to a configured datasource. Now, we can easily get the result by querying the datasource.

4.5.2 Topology

Spouts and bolts are connected together and they form a topology. Real-time application logic is specified inside Storm topology. In simple words, a topology is a directed graph where vertices are computation and edges are stream of data.

- A simple topology starts with spouts. Spout emits the data to one or more bolts. Bolt represents a node in the topology having the smallest processing logic and the output of a bolt can be emitted into another bolt as input.
- Storm keeps the topology always running, until you kill the topology. Apache Storm's main job is to run the topology and will run any number of topology at a given time.

4.5.3 Tasks

Now you have a basic idea on spouts and bolts. They are the smallest logical unit of the topology and a topology is built using a single spout and an array of bolts. They should be executed properly in a particular order for the topology to run successfully. The execution of each and every spout and bolt by Storm is called as "Tasks". In simple words, a task is either the execution of a spout or a bolt. At a given time, each spout and bolt can have multiple instances running in multiple separate threads.

Workers - A topology runs in a distributed manner, on multiple worker nodes. Storm spreads the tasks evenly on all the worker nodes. The worker node's role is to listen for jobs and start or stop the processes whenever a new job arrives.

Stream Grouping - Stream of data flows from spouts to bolts or from one bolt to another bolt. Stream grouping controls how the tuples are routed in the topology and helps us to understand the tuples flow in the topology. There are four in-built groupings as explained below.

4.5.4 Shuffle Grouping

In shuffle grouping, an equal number of tuples is distributed randomly across all of the workers executing the bolts. The following diagram depicts the structure.

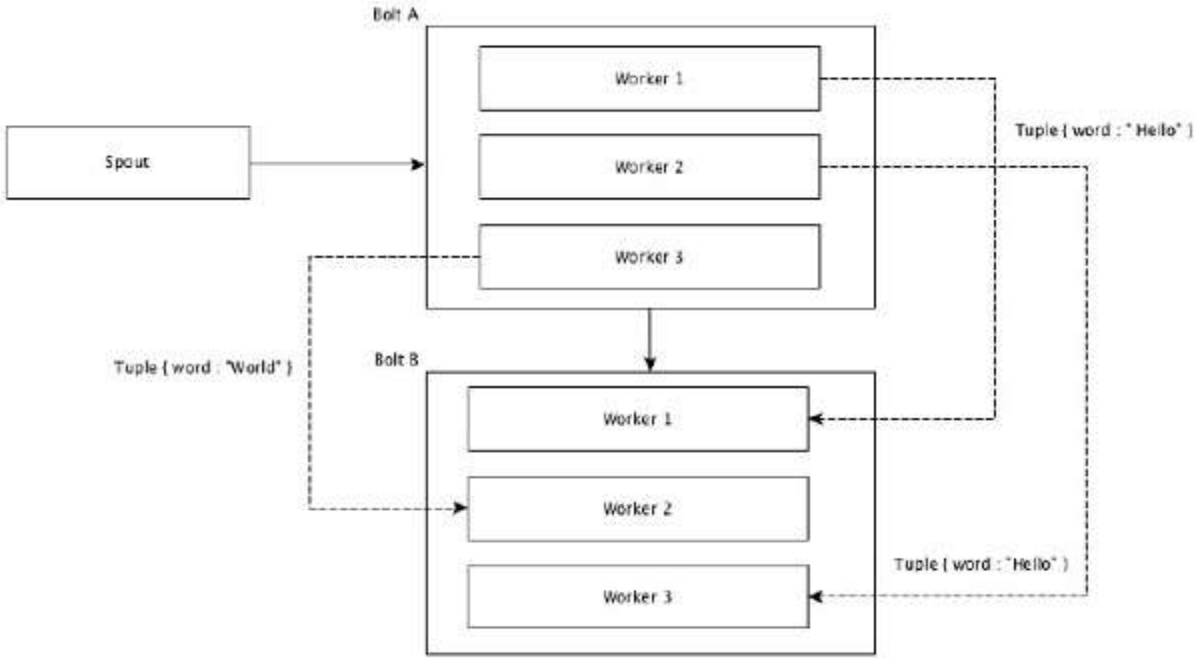


Fig 8 Shuffle Grouping

4.5.5 Field Grouping

The fields with same values in tuples are grouped together and the remaining tuples kept outside. Then, the tuples with the same field values are sent forward to the same worker executing the bolts. For example, if the stream is grouped by the field "word", then the tuples with the same string, "Hello" will move to the same worker. The following diagram shows how Field Grouping works.

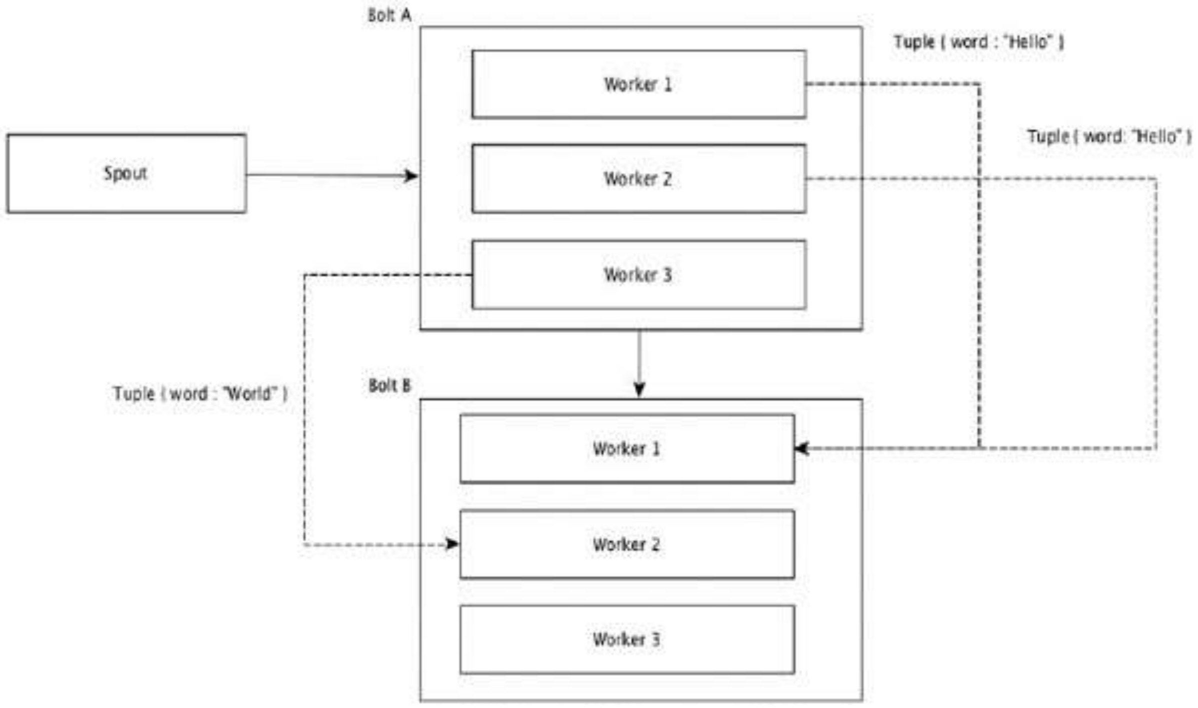


Fig 9 Global Grouping

4.5.6 Global Grouping

All the streams can be grouped and forward to one bolt. This grouping sends tuples generated by all instances of the source to a single target instance (specifically, pick the worker with lowest ID).

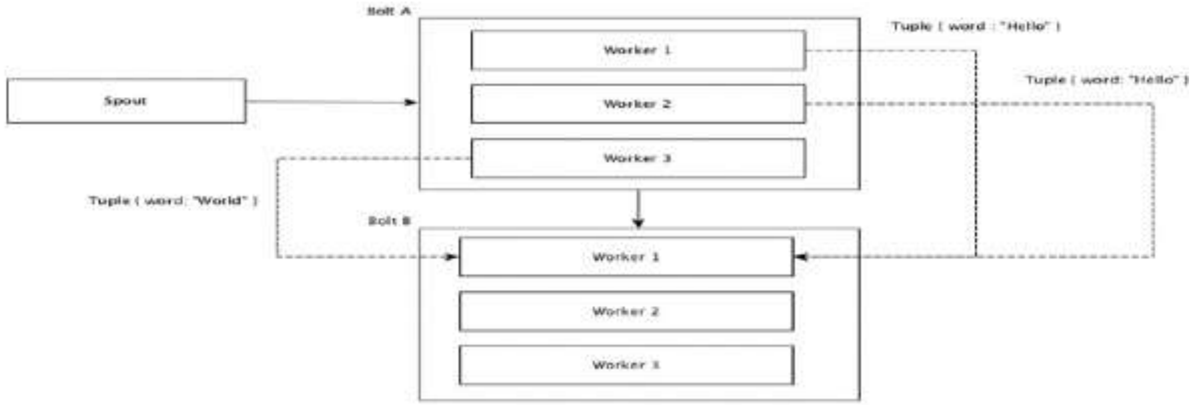


Fig 10 Global Grouping

4.5.7 All Grouping

All Grouping sends a single copy of each tuple to all instances of the receiving bolt. This kind of grouping is used to send signals to bolts. All grouping is useful for join operations.

One of the main highlight of the Apache Storm is that it is a fault-tolerant, fast with no “Single Point of Failure” (SPOF) distributed application. We can install Apache Storm in as many systems as needed to increase the capacity of the application.

Let’s have a look at how the Apache Storm cluster is designed and its internal architecture. The following diagram depicts the cluster design.

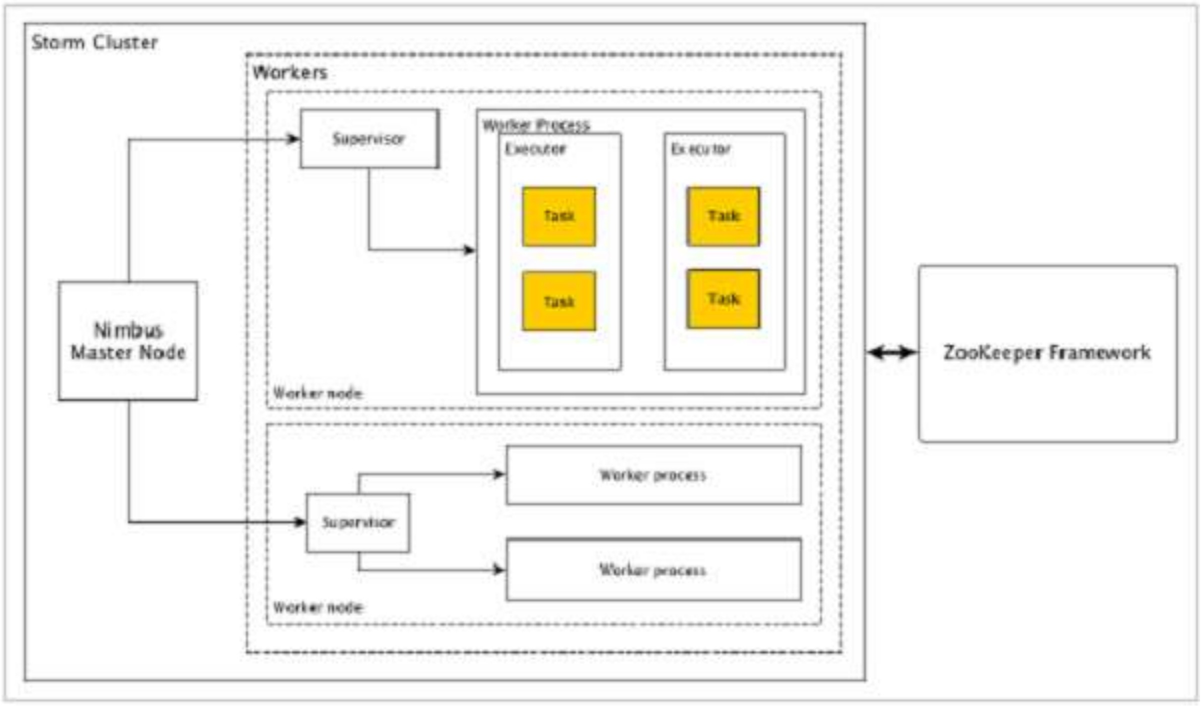


Fig 11 Cluster Design

Apache Storm has two type of nodes, **Nimbus** (master node) and **Supervisor** (worker node). Nimbus is the central component of Apache Storm. The main job of Nimbus is to run the Storm topology. Nimbus analyzes the topology and gathers the task to be executed. Then, it will distributes the task to an available supervisor.

A supervisor will have one or more worker process. Supervisor will delegate the tasks to worker processes. Worker process will spawn as many executors as needed and run the task. Apache

Storm uses an internal distributed messaging system for the communication between nimbus and supervisors.

Components	**Description**
Nimbus	Nimbus is a master node of Storm cluster. All other nodes in the cluster are called as **worker nodes**. Master node is responsible for distributing data among all the worker nodes, assign tasks to worker nodes and monitoring failures.
Supervisor	The nodes that follow instructions given by the nimbus are called as Supervisors. A **supervisor** has multiple worker processes and it governs worker processes to complete the tasks assigned by the nimbus.
Worker process	A worker process will execute tasks related to a specific topology. A worker process will not run a task by itself, instead it creates **executors** and asks them to perform a particular task. A worker process will have multiple executors.
Executor	An executor is nothing but a single thread spawn by a worker process. An executor runs one or more tasks but only for a specific spout or bolt.
Task	A task performs actual data processing. So, it is either a spout or a bolt.
ZooKeeper framework	Apache ZooKeeper is a service used by a cluster (group of

	nodes) to coordinate between themselves and maintaining shared data with robust synchronization techniques. Nimbus is stateless, so it depends on ZooKeeper to monitor the working node status. ZooKeeper helps the supervisor to interact with the nimbus. It is responsible to maintain the state of nimbus and supervisor.

Storm is stateless in nature. Even though stateless nature has its own disadvantages, it actually helps Storm to process real-time data in the best possible and quickest way.

Storm is *not entirely* stateless though. It stores its state in Apache ZooKeeper. Since the state is available in Apache ZooKeeper, a failed nimbus can be restarted and made to work from where it left. Usually, service monitoring tools like **monit** will monitor Nimbus and restart it if there is any failure.

Apache Storm also have an advanced topology called **Trident Topology** with state maintenance and it also provides a high-level API like Pig. We will discuss all these features in the coming chapters.

A working Storm cluster should have one nimbus and one or more supervisors. Another important node is Apache ZooKeeper, which will be used for the coordination between the nimbus and the supervisors.

Let us now take a close look at the workflow of Apache Storm −

- Initially, the nimbus will wait for the "Storm Topology" to be submitted to it.
- Once a topology is submitted, it will process the topology and gather all the tasks that are to be carried out and the order in which the task is to be executed.
- Then, the nimbus will evenly distribute the tasks to all the available supervisors.
- At a particular time interval, all supervisors will send heartbeats to the nimbus to inform that they are still alive.

- When a supervisor dies and doesn't send a heartbeat to the nimbus, then the nimbus assigns the tasks to another supervisor.
- When the nimbus itself dies, supervisors will work on the already assigned task without any issue.
- Once all the tasks are completed, the supervisor will wait for a new task to come in.
- In the meantime, the dead nimbus will be restarted automatically by service monitoring tools.
- The restarted nimbus will continue from where it stopped. Similarly, the dead supervisor can also be restarted automatically. Since both the nimbus and the supervisor can be restarted automatically and both will continue as before, Storm is guaranteed to process all the task at least once.
- Once all the topologies are processed, the nimbus waits for a new topology to arrive and similarly the supervisor waits for new tasks.

By default, there are two modes in a Storm cluster −

- **Local mode** − This mode is used for development, testing, and debugging because it is the easiest way to see all the topology components working together. In this mode, we can adjust parameters that enable us to see how our topology runs in different Storm configuration environments. In Local mode, storm topologies run on the local machine in a single JVM.
- **Production mode** − In this mode, we submit our topology to the working storm cluster, which is composed of many processes, usually running on different machines. As discussed in the workflow of storm, a working cluster will run indefinitely until it is shut down.
- Apache Storm processes real-time data and the input normally comes from a message queuing system. An external distributed messaging system will provide the input necessary for the realtime computation. Spout will read the data from the messaging system and convert it into tuples and input into the Apache Storm. The interesting fact is

that Apache Storm uses its own distributed messaging system internally for the communication between its nimbus and supervisor.

4.5.8 Distributed Messaging System

- Distributed messaging is based on the concept of reliable message queuing. Messages are queued asynchronously between client applications and messaging systems. A distributed messaging system provides the benefits of reliability, scalability, and persistence.
- Most of the messaging patterns follow the **publish-subscribe** model (simply **Pub-Sub**) where the senders of the messages are called **publishers** and those who want to receive the messages are called **subscribers**.
- Once the message has been published by the sender, the subscribers can receive the selected message with the help of a filtering option. Usually we have two types of filtering, one is **topic-based filtering** and another one is **content-based filtering**.
- Note that the pub-sub model can communicate only via messages. It is a very loosely coupled architecture; even the senders don't know who their subscribers are. Many of the message patterns enable with message broker to exchange publish messages for timely access by many subscribers. A real-life example is Dish TV, which publishes different channels like sports, movies, music, etc., and anyone can subscribe to their own set of channels and get them whenever their subscribed channels are available.

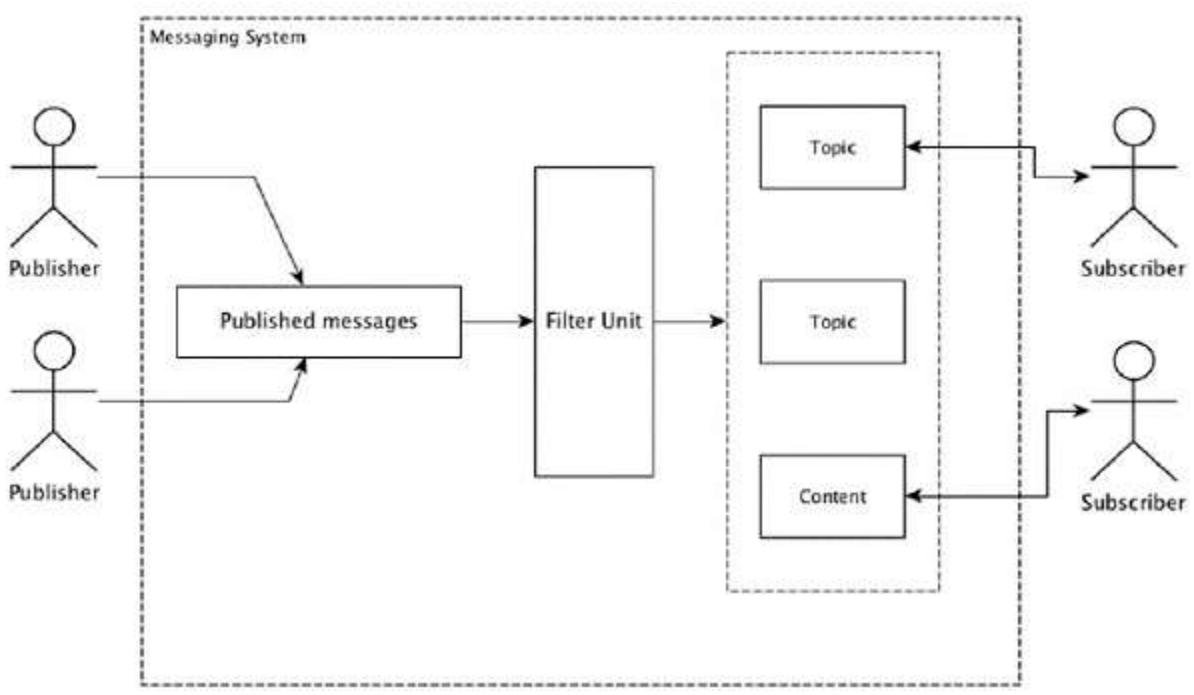


Fig 12 Distributed Messaging system

The following table describes some of the popular high throughput messaging systems –

Distributed messaging system	Description
Apache Kafka	Kafka was developed at LinkedIn corporation and later it became a sub-project of Apache. Apache Kafka is based on brokerenabled, persistent, distributed publish-subscribe model. Kafka is fast, scalable, and highly efficient.
RabbitMQ	RabbitMQ is an open source distributed robust messaging application. It is easy to use and runs on all platforms.
JMS(Java Message Service)	JMS is an open source API that supports creating,

	reading, and sending messages from one application to another. It provides guaranteed message delivery and follows publish-subscribe model.
ActiveMQ	ActiveMQ messaging system is an open source API of JMS.
ZeroMQ	ZeroMQ is broker-less peer-peer message processing. It provides push-pull, router-dealer message patterns.
Kestrel	Kestrel is a fast, reliable, and simple distributed message queue.

4.5.9 Thrift Protocol

- Thrift was built at Facebook for cross-language services development and remote procedure call (RPC). Later, it became an open source Apache project. Apache Thrift is an **Interface Definition Language** and allows to define new data types and services implementation on top of the defined data types in an easy manner.
- Apache Thrift is also a communication framework that supports embedded systems, mobile applications, web applications, and many other programming languages. Some of the key features associated with Apache Thrift are its modularity, flexibility, and high performance. In addition, it can perform streaming, messaging, and RPC in distributed applications.
- Storm extensively uses Thrift Protocol for its internal communication and data definition. Storm topology is simply **Thrift Structs**. Storm Nimbus that runs the topology in Apache Storm is a **Thrift service**.

4.6 Data Visualization

Data visualization is the visual and interactive exploration and graphic representation of data of any size, type (structured and unstructured) or origin. Visualizations help people see things that were not obvious to them before. Even when data volumes are very large, patterns can be spotted quickly and easily. Visualizations convey information in a universal manner and make it simple to share ideas with others.

It's a way to get fast insights through visual exploration, robust reporting and flexible information sharing. It helps a wide variety of users to make sense of the increasing amount of data within your organization. It's a way to present big data in a way business users can quickly understand and use. Data visualization brings the story of your data to life.

Data visualization can be used for different purposes.

Some examples:

- To create and share meaningful reports with anyone anywhere
- To forecast and quickly identify opportunities and anticipate future trends
- To optimize corporate processes & to drive innovation
- To give anyone in the organization the power to visually explore and analyze all available data.

Data visualization was created to visually explore and analyze data quickly. It's designed for anyone in your organization who wants to use and derive insights from data regardless of analytic skill level – from influencers, decision makers and analysts to statisticians and data scientists. It also offers IT an easy way to protect and manage data integrity and security. The amount of data will continue to grow while often time and resources to interpret the data continue to decrease. Data visualization will become one of the few tools able to help us win that challenge.

Data visualization helps uncover insights buried in your data and discover trends within your business and the market that affect your bottom line. Insights in your data can provide competitive advantage and the opportunity to differentiate.

Data visualization lets you read your market intelligently, compare your overall position with the industry trend, define the most appreciated features of your products and adapt development accordingly, combine information about sales with consumer preferences and much more.

Data visualization allows you to spot market trends and grow your business Data visualization allows you now your market's dynamics like never before Knowing your customer better leads to more effective sales and marketing actions and enhances customer experience. Data visualization allows you to know your customers' needs and act on it. Data visualization provides information that is easy to understand and to share. Company KPIs are always under control. Data from a variety of internal and external sources is channeled into one single, shared source of information.

Big Data visualization tool must be able to deal with semi-structured and unstructured data because big data usually have this type of format. It is realized that to cope with such huge amount of data there is need for immense parallelization, which is a challenge in visualization. The challenge in parallelization algorithm is to break down the problem into such independent task that they can run independently. The task of big data visualization is to recognize interesting patterns and correlations. We need to carefully choose the dimensions of data to be visualized, if we reduce dimensions to make our visualization low then we may end up losing interesting patterns but if we use all the dimensions we may end up having visualization too dense to be useful to the users. For example: "Given the conventional displays (1.3 million pixels), visualizing every data point can lead to over-plotting, overlapping and may overwhelm user's perceptual and cognitive capacities

Due to vast volume and high magnitude of big data it becomes difficult to visualize. Most of the current visualization tool have low performance in scalability, functionality and response time .Methods have been proposed which not only visualizes data but processes at the same time. These methods use Hadoop and storage solution and R programming language as compiler environment in the model

4.7 Visualization Tools

Various tools have emerged to help us out from the above pointed problems. The most important feature that a visualization must have is that it should be interactive, which means that user should be able to interact with the visualization. Visualization must display relevant information when hovered over it, zoom in and out panel should be there, visualization should adapt itself at runtime if we select subset or superset of data. We reviewed some of the most popular visualization tools.

4.7.1 Tableau

Tableau is interactive data visualization tool which is focused on Business Intelligence. Tableau provides very wide range of visualization options. It provides option to create custom visualization. It is fast and flexible. It supports mostly all the data format and connection to various servers right from the Amazon Aurora to Cloudera Hadoop and Salesforce. User interface is intuitive, wide variety of charts are available. For simple calculations and statistics one does not require any coding skills but for heavy analytics we can run models in R and then import the results into Tableau. This requires quite a bit of programming skill based upon the task we need to perform.

Some other important big data visualization problems are as follows

Visual noise: Most of the objects in dataset are too relative to each other. It becomes very difficult to separate them.

Information loss: To increase the response time we can reduce data set visibility, but this leads to information loss.

Large image perception: Even after achieving desired me- chanical output we are limited by our physical perception.

4.7.2 Microsoft Power BI

Power BI is a powerful cloud-base business analytics service. Visualization are interactive and rich. Power BI consists of 3 elements, Power BI Desktop, Service(SaaS), Apps. Every service is available to us that is why it makes Power BI flexible and persuasive. With more than 60 types of source integration you can start creating visualization in matter of minutes. Power BI combines the familiar Microsoft tools like Office, SharePoint and SQL Server.

The feature that it distinguishes from other tools is that you can use natural language to query the data. You don't require programming skills for this tool but there is option available to run your R script. You can merge multiple data sources and create models, which comes in handy. Fig. 13 represents 3 visualizations in 3 coordinates, i.e. left, bottom and right. Left represents profit by county and market, bottom represents profit by region and right coordinate represents all over sales and profit.

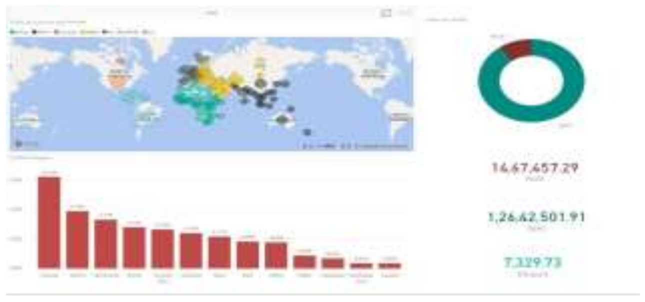


Fig 13 : Microsoft Power BI

4.7.3 Plotly

Plotly is also known as Plot.ly is build using python and Django framework. The actions it can perform are analyzing and visualizing data. It is free for users but with limited features, for all the features we need to buy the professional membership. It creates charts and dashboards online but can be used as offline service inside Ipython notebook, jupyter notebook and panda. Different variety of charts are available like statistical chart, scientific charts, 3D charts, multiple axes, dashboards etc. Plotly uses a tool called "Web Plot Digitizer(WPD)" which automatically grabs the data from the static image .Plotly on premises service is also available, it is like plot.ly cloud but you host data on your private cloud behind your own firewall. This for those wo have concern about the privacy of their data. Python, R, MATLAB and Julia APIs are available for the same.

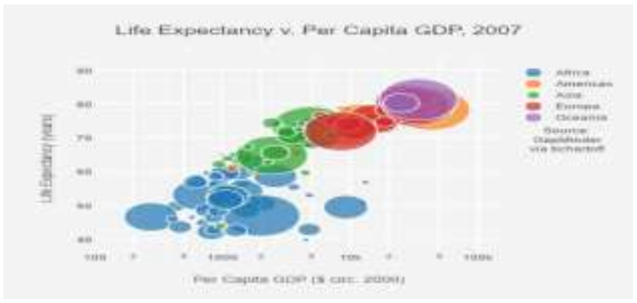


Fig 14 Plotly

4.7.4 Gephi

Gephi is open-source network analysis tool writ- ten in Java and OpenGL. It is used to handle very large and complex datasets. The network analysis includes

- Social Network Analysis
- Link Analysis
- Biological Network Analysis

With its dynamic data exploration Gephi stands out rest of its competition for graph analysis. No programming skills are required to run thin tools but a good knowledge in graphs is necessary. It uses GPU 3D render engine to accelerate the performance and give real time analysis

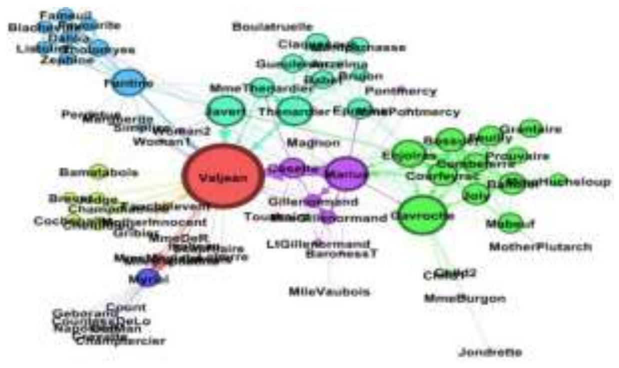


Fig 15 : Gephi

4.7.5 Excel 2016

Microsoft Excel is a spreadsheet developed by Microsoft. It can not only be used for Big Data and statistical analysis but it is also a powerful visualization tool. Using power query excel can connect to most of the services like HDFS, SaaS etc and is capable of managing Semi- Structured data. Combined with visualization techniques like "Conditional Formatting" and interactive graphs makes Excel 2016 a good contender in the ocean of Big Data visualization tools.

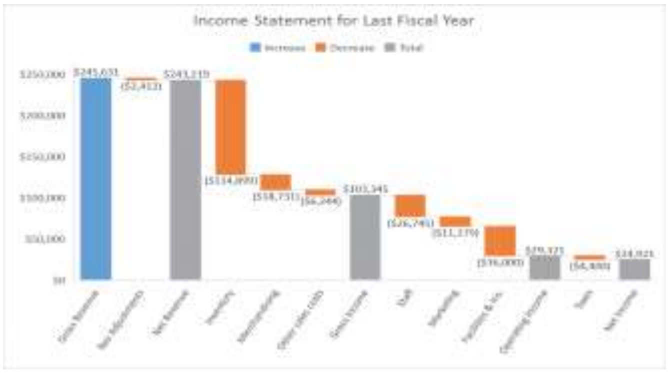


Fig 16 : Excel

4.7.6 Oracle Visual Analyzer

Introduced in 2015, this web-based tool within the Oracle Business Intelligence Cloud Service claimed a spot at the Magic Quadrant Business Intelligence and Analytics Platform report by Gartner. Interactive visuals and highly advanced analysis clubbed with a customizable dashboard are some of the key features of Oracle Visual Analyzer. Being highly scalable, this data visualization tool is very suitable for enterprises with large-scale deployments where deep insights and well curated reports are essential.

Every bit of data carries a story with it and these data visualization tools are the gateway to fathom the story it tries to tell us. It helps us to understand about the current statistics and the future trends of the market.

4.7.7 Data wrapper

Datawrapper is a data visualization tool that’s gaining popularity fast, especially among media companies which use it for presenting statistics and creating charts. It has an easy to navigate user interface where you can easily upload a csv file to create maps, charts and visualizations that can be quickly added to reports. Although the tool is primarily aimed at journalists, it’s flexibility should accommodate a host of applications apart from media usage.

4.7.8 Google Chart

Google is an obvious benchmark and well known for the user-friendliness offered by its products and Google chart is not an exception. It is one of the easiest tools for visualizing huge data sets. Google chart holds a wide range of chart gallery, from a simple line graph to complex hierarchical tree-like structure and you can use any of them that fits your requirement. Moreover, the most important part while designing a chart is customization and with Google charts, it’s fairly Spartan. You can always ask for some technical help if you want to dig deep. It renders the chart in HTML5/SVG format and it is cross-browser compatible. Added to this, it also has adopted VML for supporting old IE browsers and that’s also cross-platform compatible, portable to iOS and the new release of Android. The chart data can be easily exported to PNG format.

4.7.9 Qlikview

Qlik is one of the major players in the data analytics space with their Qlikview tool which is also one of the biggest competitors of Tableau. Qlikview boasts over 40,000 customers spanning across over 100 countries. Qlik is particularly known for its highly customizable setup and a host of features that help create the visualizations much faster. However, the available options could mean there would be a learning curve to get accustomed with the tool so as to use it to its full potential. Apart from its data visualization prowess, Qlikview also offers analytics, business intelligence and enterprise reporting features. The clean and clutter-free user experience is one of the notable aspects of Qlikview. Qliksense is a sister package of Qlikview which is often used alongside the former to aid in data exploration and discovery.

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 INFORMATION TECHNOLOGY

UNIT - V

SITA3008 - Internet of Things

UNIT 5 HANDS-ON PROJECTS

Industry 4.0 concepts. Sensors and sensor Node and interfacing using any Embedded target boards (Raspberry Pi / Intel Galileo/ARM Cortex/ Arduino), DIY Kits – Soil moisture monitoring, Weather monitoring, Air quality Monitoring, Movement Detection.

5.1 Industry 4.0 concepts

Industry 4.0 refers to a new phase in the Industrial Revolution that focuses heavily on interconnectivity, automation, machine learning, and real-time data. Industry 4.0, also sometimes referred to as IIoT or smart manufacturing, marries physical production and operations with smart digital technology, machine learning, and big data to create a more holistic and better connected ecosystem for companies that focus on manufacturing and supply chain management. While every company and organization operating today is different, they all face a common challenge—the need for connectedness and access to real-time insights across processes, partners, products, and people.

That's where Industry 4.0 comes into play.

Industry 4.0 isn't just about investing in new technology and tools to improve manufacturing efficiency—it's about revolutionizing the way your entire business operates and grows. This resource will provide you with an in-depth overview on the topic of Industry 4.0 and IIoT, including information on the following:

5.1.1.Evolution of Industry from 1.0 to 4.0

Before digging too much deeper into the what, why, and how of Industry 4.0, it's beneficial to first understand how exactly manufacturing has evolved since the 1800s. There are four distinct industrial revolutions that the world either has experienced or continues to experience today.

5.1.1.1 The First Industrial Revolution

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

5.1.1.2 The Second Industrial Revolution

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

5.1.1.3 The Third Industrial Revolution

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

5.1.1.4 The Fourth Industrial Revolution or Industry 4.0

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

5.2 Basic IIoT Concepts and Glossary of Terms

There are hundreds of concepts and terms that relate to IIoT and Industry 4.0, but here are 12 foundational words and phrases to know before you decide whether you want to invest in Industry 4.0 solutions for your business:

Enterprise Resource Planning (ERP): Business process management tools that can be used to manage information across an organization.

IoT: IoT stands for Internet of Things, a concept that refers to connections between physical objects like sensors or machines and the Internet.

IIoT: IIoT stands for the Industrial Internet of Things, a concept that refers to the connections between people, data, and machines as they relate to manufacturing.

Big data: Big data refers to large sets of structured or unstructured data that can be compiled, stored, organized, and analyzed to reveal patterns, trends, associations, and opportunities.

Artificial intelligence (AI): Artificial intelligence is a concept that refers to a computer's ability to perform tasks and make decisions that would historically require some level of human intelligence.

M2M: This stands for machine-to-machine, and refers to the communication that happens between two separate machines through wireless or wired networks.

Digitization: Digitization refers to the process of collecting and converting different types of information into a digital format.

Smart factory: A smart factory is one that invests in and leverages Industry 4.0 technology, solutions, and approaches.

Machine learning: Machine learning refers to the ability that computers have to learn and improve on their own through artificial intelligence—without being explicitly told or programmed to do so.

Cloud computing: Cloud computing refers to the practice of using interconnected remote servers hosted on the Internet to store, manage, and process information.

Real-time data processing: Real-time data processing refers to the abilities of computer systems and machines too continuously and automatically process data and provides real-time or near-time outputs and insights.

Ecosystem: An ecosystem, in terms of manufacturing, refers to the potential connectedness of your entire operation—inventory and planning, financials, customer relationships, supply chain management, and manufacturing execution.

Cyber-physical systems (CPS): Cyber-physical systems, also sometimes known as cyber manufacturing, refers to an Industry 4.0-enabled manufacturing environment that offers real-time data collection, analysis, and transparency across every aspect of a manufacturing operation.

Now that you have a better understanding of some of the core concepts related to Industry 4.0, you're ready to dig deeper into how smart manufacturing can revolutionize the way you run and grow your business.

5.2.1 Smart Manufacturing Use Cases

One of the best ways to understand the concept of smart manufacturing better is to think about how it could be applied to your business, or a business similar to your business. Here are three use cases that can help you understand the value of Industry 4.0 in a manufacturing operation:

1. Supply chain management and optimization—Industry 4.0 solutions give businesses greater insight, control, and data visibility across their entire supply chain. By leveraging supply chain management capabilities, companies can deliver products and services to market faster, cheaper, and with better quality to gain an advantage over less-efficient competitors.

2. Predictive maintenance/analytics—Industry 4.0 solutions give manufacturers the ability to predict when potential problems are going to arise before they actually happen. Without IoT systems in place at your factory, preventive maintenance happens based on routine or time. In other words, it's a manual task. With IoT systems in place, preventive maintenance is much more automated and streamlined. Systems can sense when problems are arising or machinery needs to be fixed, and can empower you to solve potential issues before they become bigger problems. These types of analytics can enable manufacturers to pivot from preventive maintenance to predictive maintenance.

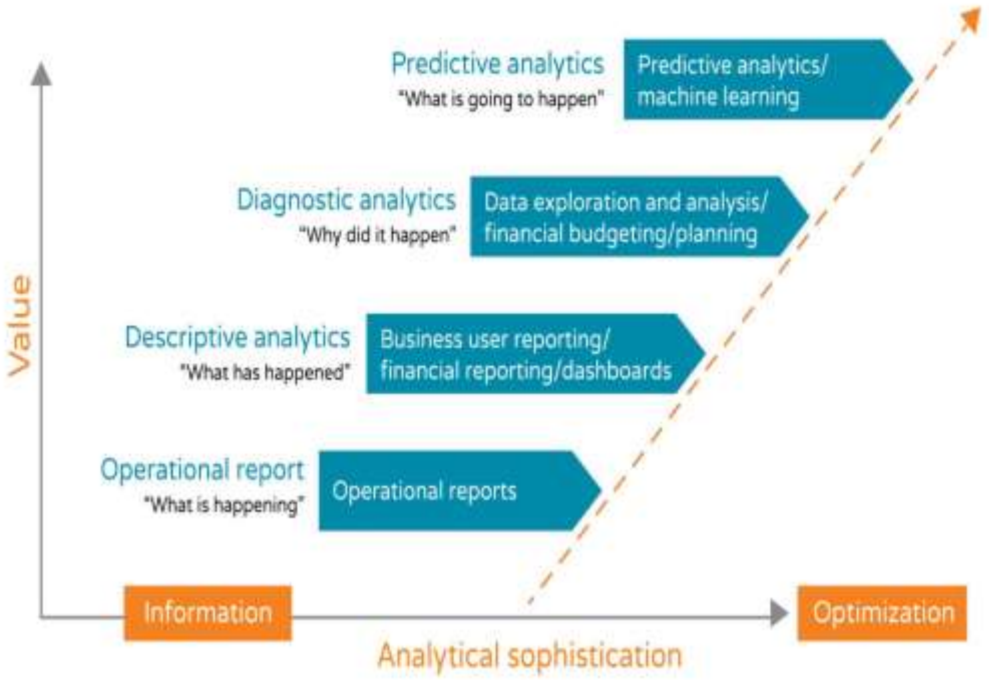

Predictive analytics
"What is going to happen"
Predictive analytics/ machine learning
Diagnostic analytics
"Why did it happen"
Data exploration and analysis/ financial budgeting/planning
Descriptive analytics
"What has happened"
Business user reporting/ financial reporting/dashboards
Operational report
"What is happening"
Operational reports
Value
Information
Optimization
Analytical sophistication

Fig 1 Predictive maintenance/analytics

3. Asset tracking and optimization—Industry 4.0 solutions help manufacturers become more efficient with assets at each stage of the supply chain, allowing them to keep a better pulse on inventory, quality, and optimization opportunities relating to logistics. With IoT in place at a factory, employees can get better visibility into their assets worldwide. Standard asset management tasks such as asset transfers, disposals, reclassifications, and adjustments can be streamlined and managed centrally and in real time.

5.2.2 Benefits of Adopting an Industry 4.0 Model

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

Here is a quick, non-exhaustive list of some of the benefits of adopting an Industry 4.0 model for our business:

It makes you more competitive, especially against disruptors like Amazon. As companies like Amazon continue to optimize logistics and supply chain management, you need to be investing in technology and solutions that help you improve and optimize your own operation. To stay competitive, you have to have the systems and processes in place to allow you to provide the same level of service (or better) to your customers and clients that they could be getting from a company like Amazon.

It makes you more attractive to the younger workforce. Companies that invest in modern, innovative Industry 4.0 technologies are better positioned to attract and retain new workers.

It makes your team stronger and more collaborative. Companies that invest in Industry 4.0 solutions can increase efficiency, boost collaboration between departments, enable predictive and prescriptive analytics, and allow people including operators, managers, and executives to more fully leverage real-time data and intelligence to make better decisions while managing their day-to-day responsibilities.

It allows you to address potential issues before they become big problems. Predictive analytics, real-time data, internet-connected machinery, and automation can all help you be more proactive when it comes to addressing and solving potential maintenance and supply chain management issues.

It allows you to trim costs, boost profits, and fuel growth. Industry 4.0 technologies helps you manage and optimize all aspects of your manufacturing processes and supply chain. It gives you access to the real-time data and insights you need to make smarter, faster decisions about your business, which can ultimately boost the efficiency and profitability of your entire operation.

5.3 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.

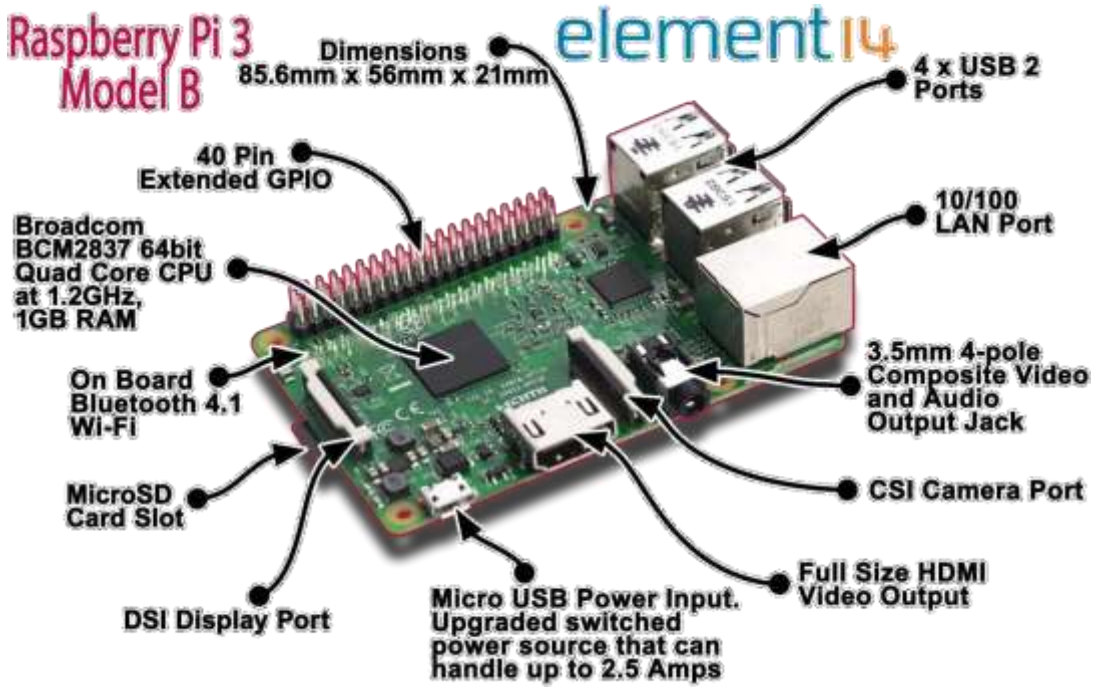

Raspberry Pi 3 Model B
Dimensions 85.6mm x 56mm x 21mm
element14
4 x USB 2 Ports
40 Pin Extended GPIO
10/100 LAN Port
Broadcom BCM2837 64bit Quad Core CPU at 1.2GHz, 1GB RAM
3.5mm 4-pole Composite Video and Audio Output Jack
On Board Bluetooth 4.1 Wi-Fi
MicroSD Card Slot
CSI Camera Port
Full Size HDMI Video Output
DSI Display Port
Micro USB Power Input. Upgraded switched power source that can handle up to 2.5 Amps

Fig 2. Raspberry Pi Element

5.3.1 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 videocore IV
Ethernet	10/100	10/100	None
Wireless	802.11/Bluetooth 4.0	None	None
Video output	HDMI/Composite	HDMI/Composite	HDMI/Composite
GPIO	40	40	40

Fig 3 Configuration

Raspberry Pi Basics: installing Raspbian and getting it up and running

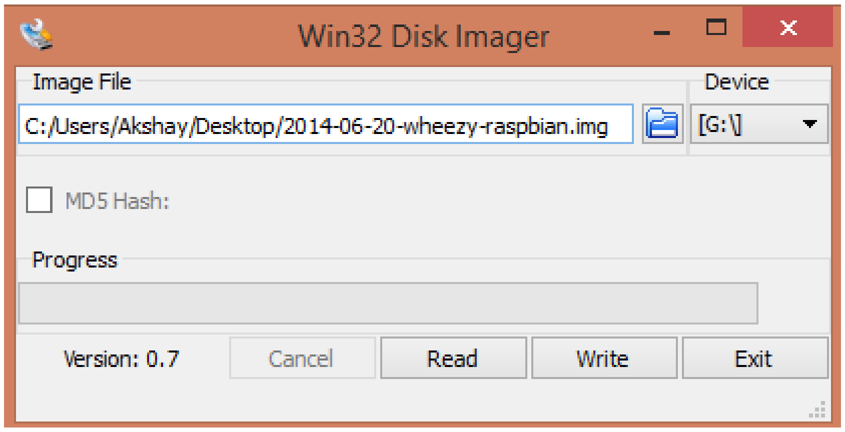
<u>1. Downloading Raspbian and Image writer.</u>

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.

<u>2. Writing the image</u>

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:\"



```
C:/Users/Akshay/Desktop/2014-06-20-wheezy-raspbian.img
[G:\]
Version: 0.7
```

Fig 3 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.

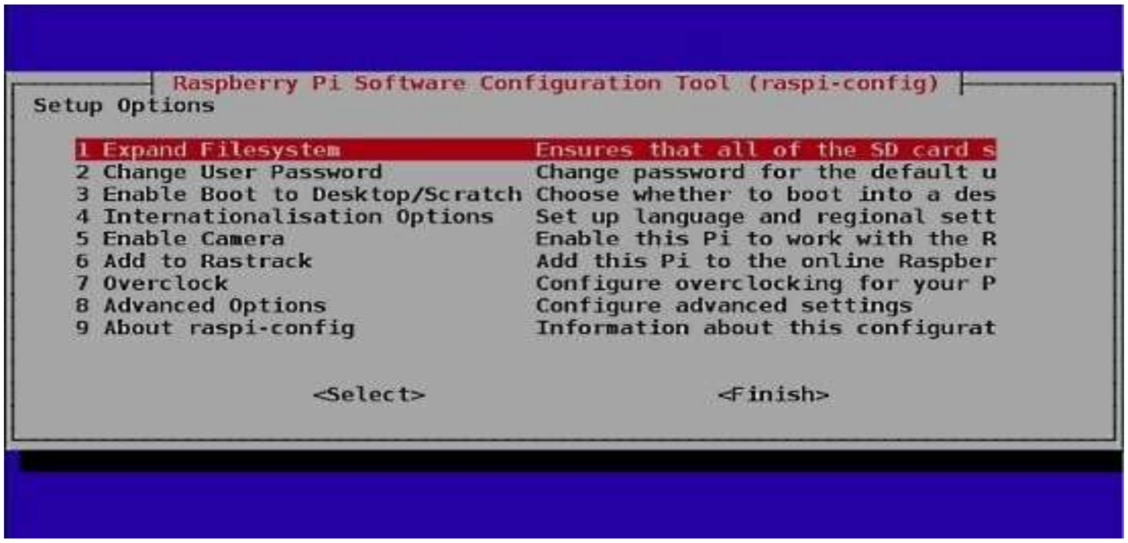
3. Setting up the Pi

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.



```
Raspberry Pi Software Configuration Tool (raspi-config)
Setup Options

  1 Expand Filesystem              Ensures that all of the SD card s
  2 Change User Password           Change password for the default u
  3 Enable Boot to Desktop/Scratch Choose whether to boot into a des
  4 Internationalisation Options   Set up language and regional sett
  5 Enable Camera                  Enable this Pi to work with the R
  6 Add to Rastrack                Add this Pi to the online Raspber
  7 Overclock                      Configure overclocking for your P
  8 Advanced Options               Configure advanced settings
  9 About raspi-config             Information about this configurat

            <Select>                         <Finish>
```

Fig 4 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.

<u>sudoraspi-config</u>

Once you execute this command the "Setup Options" screen will come up as shown in the image above.

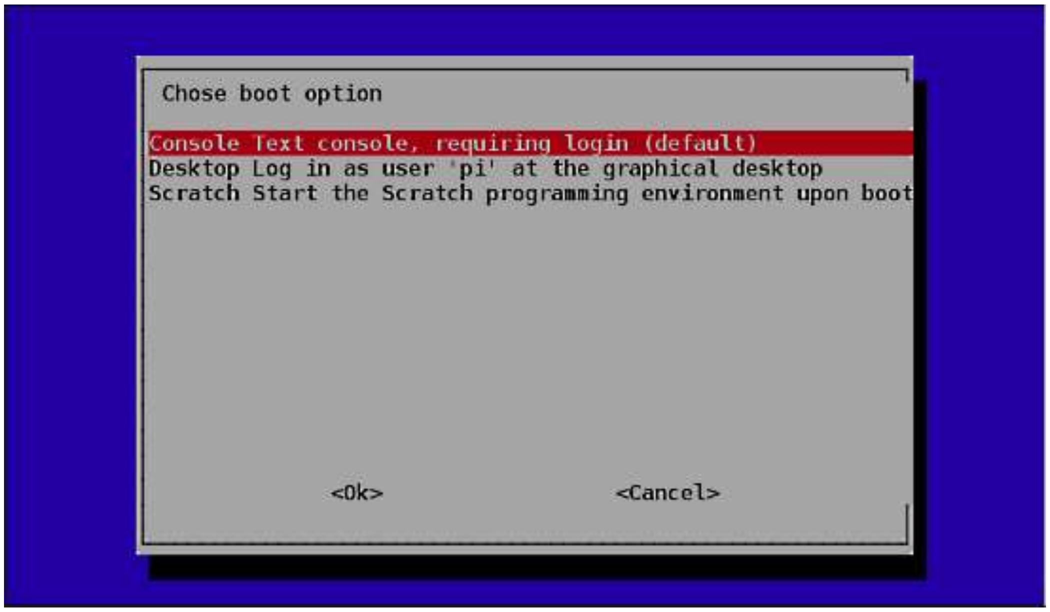
Now that the Setup Options window is up, we will have to set a few things. After completing each of the steps below, if it asks to reboot the Pi, please do so. After the reboot, if you don't get the "Setup Options" screen, then follow the command given above to get the screen/window.

<u>The first thing to do:</u>

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

<u>The second thing to do:</u>

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.



```
Chose boot option

Console Text console, requiring login (default)
Desktop Log in as user 'pi' at the graphical desktop
Scratch Start the Scratch programming environment upon boot
```

Fig 5 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.

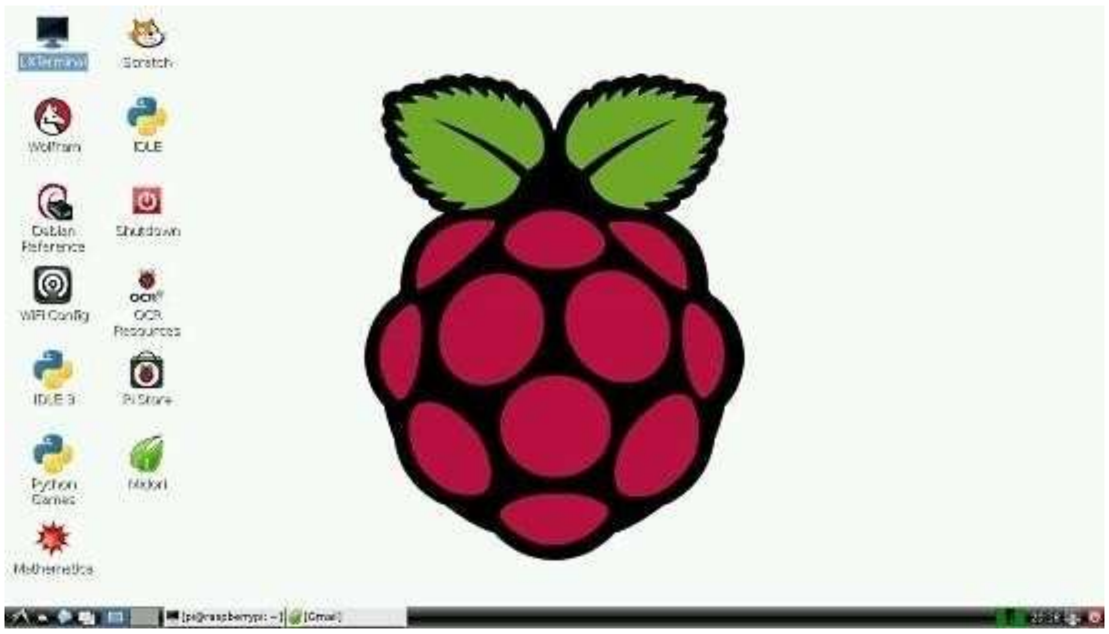


Fig 6 Raspberry Desktop

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.

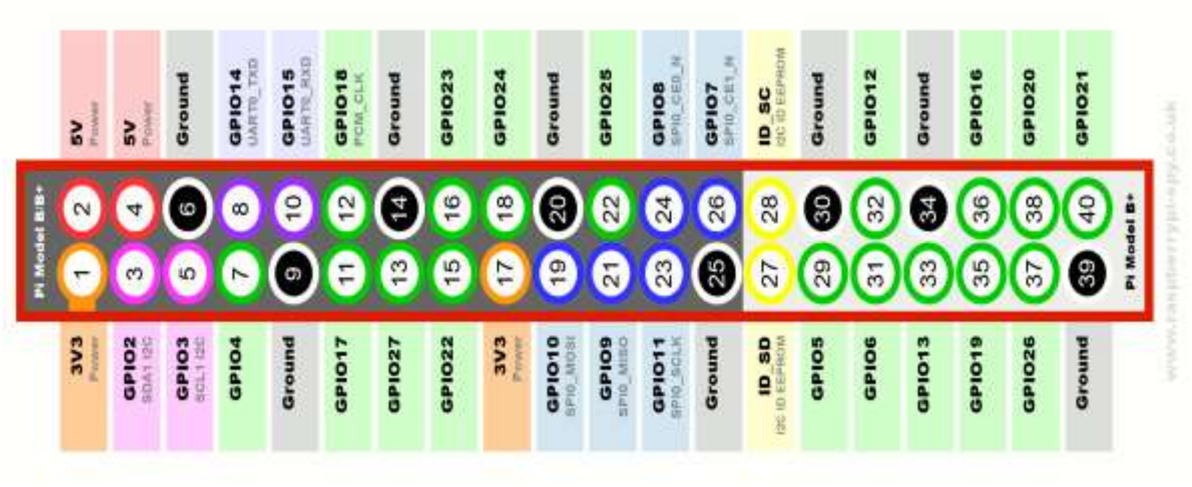


Fig 7. GPIO Pins

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

<u>**Blinking LED Code:**</u>

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.

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

5.4 Intel Galileo Board

The Intel Galileo Board is the first Arduino board based on Intel architecture. The headers (what you connect jumper cables to on the board) are based off the Arduino 1.0 pinout model that's found on the Arduino Uno R3 boards. This provides the ability to use compatible shields (modules that you can plug into headers), allowing you to extend the functionality of the board. Like the Uno, it has 14 digital I/O pins, 6 analog inputs, a serial port, and an ICSP header for serial programming.

Quark

The board features an Intel® Quark SoC X1000 Application Processor, designed for the Internet of Things. It's smaller and more power efficient than the Intel Atom® Processor, making it great for small, low-powered projects.

Ethernet

On the top portion of the board, right next to what looks like an audio jack labeled UART, there is a 100 Mb Ethernet port that allows the Intel Galileo to connect to wired networks. Once your board is connected to the Internet, anything is possible.

Mini-PCIe

The Intel Galileo is the first Arduino Certified board that provides a mini PCI Express (mPCIe) slot. This allows you to connect standard mPCIe modules like Wi-Fi, Bluetooth, and SIM card adapters for cell phones.

Real Time Clock (RTC)

Synchronize data between modules using the boards-integrated Real Time Clock. Using the Arduino Time Library, you can add timekeeping functionality to your program. Wireless projects can synchronize in real time using the Network Time Protocol (NTP) and Global Positioning System (GPS) time data.

To preserve time between system resets, add a coin cell battery to your Intel Galileo Board.

Micro SD

Use the optional onboard micro SD card reader that is accessible through the Secure Digital (SD) Library. Unlike other Arduinos, the Intel Galileo does not save sketches (programs) between power on/off states of the board without an SD card. Using a micro SD card, you can store up to 32 GB of data!

Linux*

Using the Linux image for the Intel Galileo, you can access serial ports, Wi-Fi, and board pins using programming languages like Advanced Linux Sound Architecture (ALSA), Video4Linux (V4L2), Python, Secure Shell (SSH), Node.js, and OpenCV. Using these extra features provided by Linux requires a micro SD card. Take advantage of the Intel Quark processing power and create something amazing.

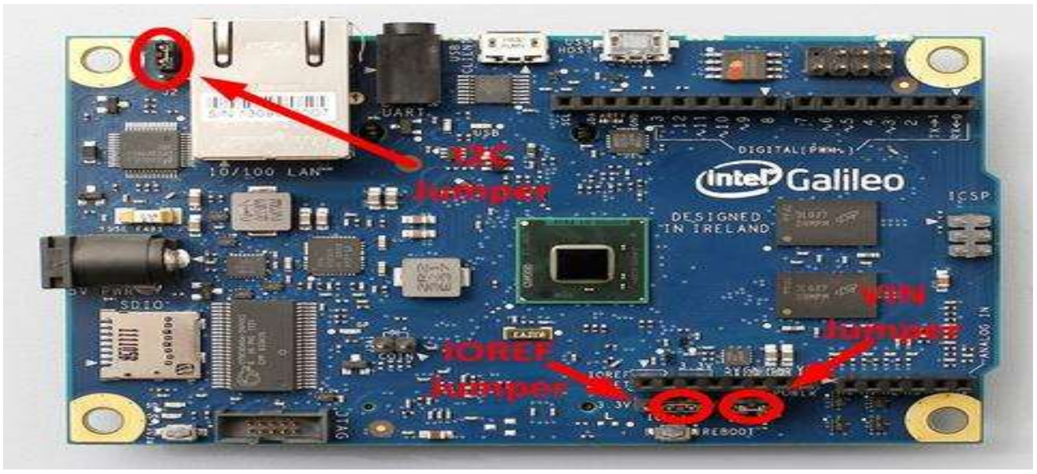


Fig 8 Intel Galileo

Getting Started Using Arduino* IDE for the Intel® Galileo Board

Download and Install

If you already have Arduino* software installed on your computer, you may need to uninstall it. The Intel® Galileo Board uses a special version of the Arduino software to communicate with the board.

To ensure proper use of the Intel® Galileo Board with Arduino, you need to install the latest driver for your operating system from Galileo Software Downloads.

For detailed instructions on how to get started with setting up Arduino for Intel Galileo, see the Getting Started Guide.

For this tutorial, we use Arduino 1.5.3.

Troubleshooting steps

Having issues with the Arduino software and the Intel Galileo Board? Try the following steps.

Once you're set up, let's explore the Arduino Integrated Development Environment.

1. Plug in the power supply

Like most Arduino boards, the Intel Galileo is powered by 5 volts of electricity. The power supply adapter that comes with the Intel Galileo has a maximum of 5 volts. Using a different adapter that is higher than 5V damages the Intel Galileo board.

Plug the adapter into a wall outlet. Universal adapters come included inside the box. Gently plug the adapter into the board. Once the LED on the board labeled USB turns on, the USB serial port is ready for communication.

2. Plug in the micro USB cable

There are two USB cable inputs. Carefully plug the USB cable into the port labeled USB Client. The other port, USB HOST, can be used for things like a keyboard.

3. Select the correct port in Arduino IDE

Start the Arduino software. In the Tools navigation, select Intel® Galileo from the board menu.

Again in Tools, select the serial port option.

You see a list of available serial ports.

The port addresses begin with either try or cu. If you're using Windows, the port begins with COM.

The second portion of the name usually signifies the type of port it is. You want to look for a port starting with cu, followed by .usbmodem and ending in a random letter and number combination like fa141 or fa121.

Once you successfully connected, you should see Intel® Galileo on {your serial port number} at the bottom right side of the Arduino software interface.

Don't see the ports listed?

Try unplugging the USB cable and plugging it back in again. Resetting the connection triggers the USB port on your computer to read incoming data.

Close the Arduino software first, then unplug the USB cable from your computer. Try repeating the process until the port is recognized. Restarting your computer helps you in debugging the issue.

Firmware upgrade

If you haven't done so, you need upgrade the firmware on the Intel Galileo board before continuing. After connecting the board to your computer and selecting the correct serial port, go to the Help menu in the Arduino software and select Firmware Update. This update takes several minutes. Follow the prompts until the process is complete, and then continue.

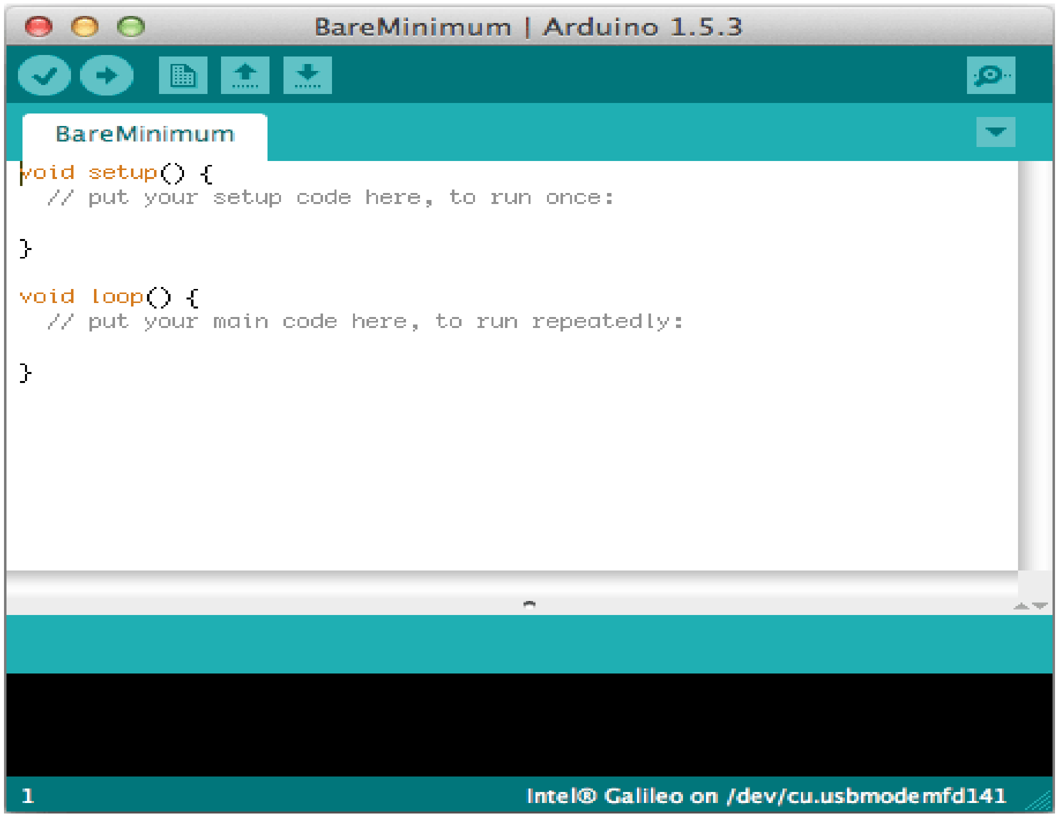
<u>Arduino* IDE Tutorial</u>

Required equipment:

- Intel® Galileo Board
- Power supply (included in the box)
- Micro USB cable (Type B)
- Installed and configured Arduino* software v 1.5.3

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:



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

}

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

}
```

```
1                                   Intel® Galileo on /dev/cu.usbmodemfd141
```

Fig 9 Arduino 1.5.3

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

<u>Pin 13</u>

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.

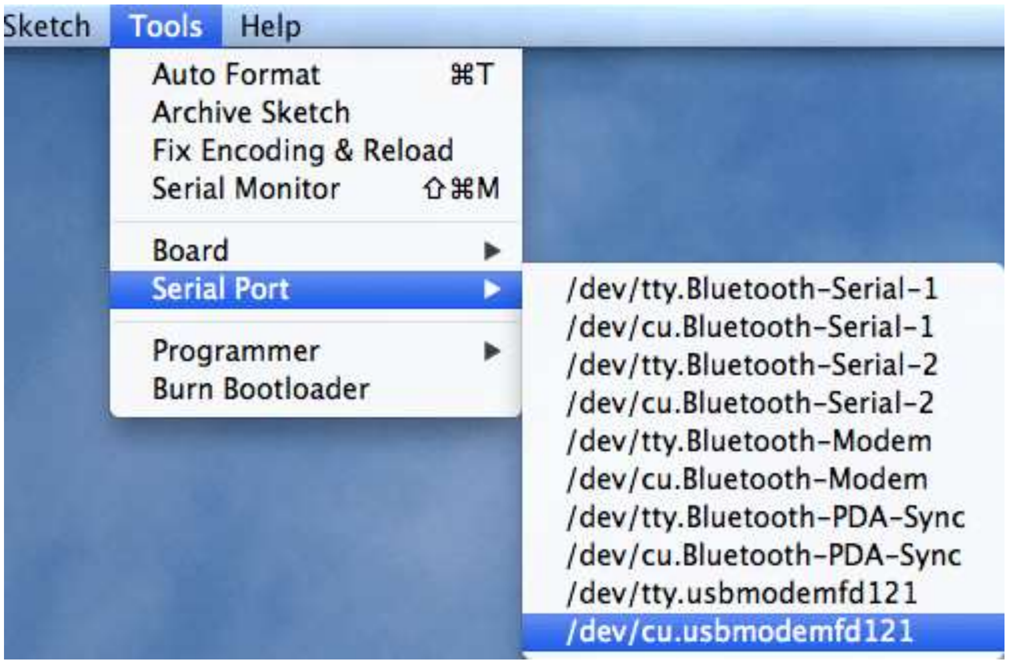


Fig 10 Execution of Arduino

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.

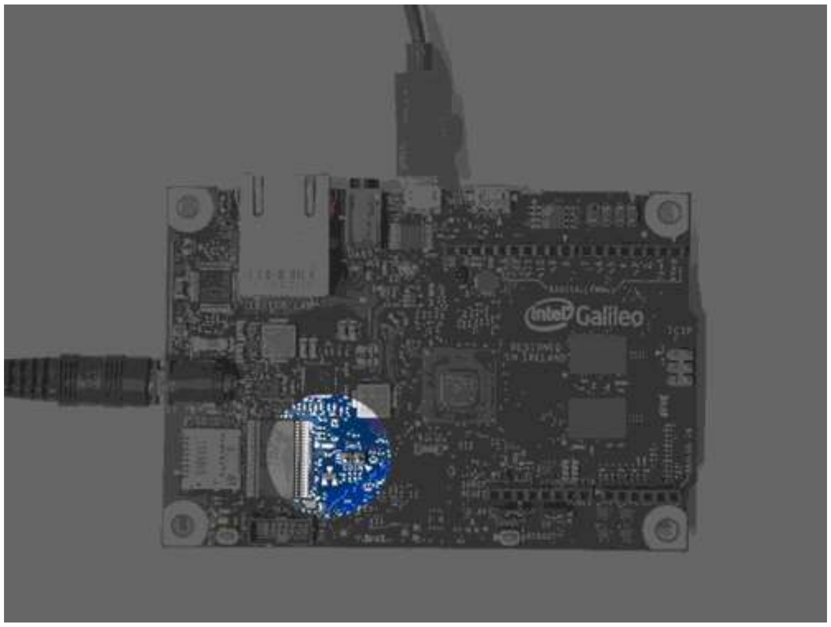


Fig 11 Blink Light

5.5 ARM Cortex

The ARM microcontroller stands for Advance RISC Machine; it is one of the extensive and most licensed processor cores in the world. The first ARM processor was developed in the year 1978 by Cambridge University, and the first ARM RISC processor was produced by the Acorn Group of Computers in the year 1985. These processors are specifically used in portable devices like digital cameras, mobile phones, home networking modules and wireless communication technologies and other embedded systems due to the benefits, such as low power consumption, reasonable performance, etc.

5.5.1 ARM Architecture

The ARM architecture processor is an advanced reduced instruction set computing [RISC] machine and it's a 32bit reduced instruction set computer (RISC) microcontroller. It was introduced by the Acron computer organization in 1987. This ARM is a family of microcontroller developed by makers like ST Microelectronics,Motorola, and so on. The ARM architecture comes with totally different versions like ARMv1, ARMv2, etc., and, each one has its own advantage and disadvantages.

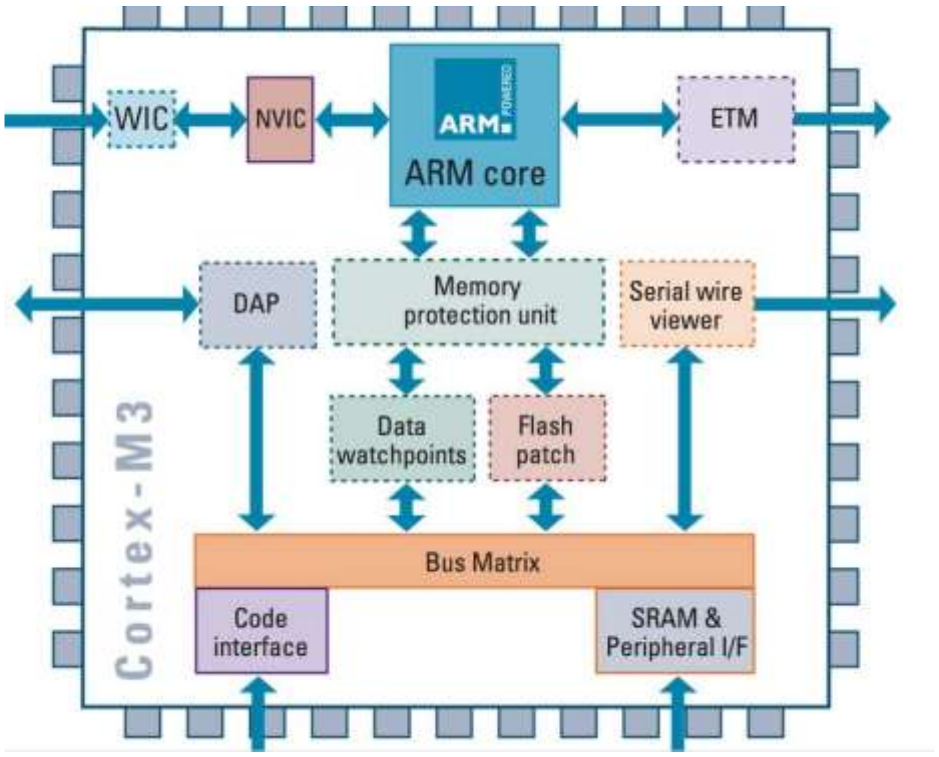


Fig 12 ARM Cortex block diagram

5.5.2 The ARM Architecture

- Arithmetic Logic Unit
- Booth multiplier
- Barrel shifter
- Control unit
- Register file

The ARM processor conjointly has other components like the Program status register, which contains the processor flags (Z, S, V and C). The modes bits conjointly exist within the program standing register, in addition to the interrupt and quick interrupt disable bits; some special registers: Some registers are used like the instruction; memory data read and write registers and memory address register.

Priority encoder: The encoder is used in the multiple load and store instruction to point which register within the register file to be loaded or kept.

Multiplexers: several multiplexers are accustomed to the management operation of the processor buses. Because of the restricted project time, we tend to implement these components in a very behavioral model. Each component is described with an entity. Every entity has its own architecture, which can be optimized for certain necessities depending on its application. This creates the design easier to construct and maintain.

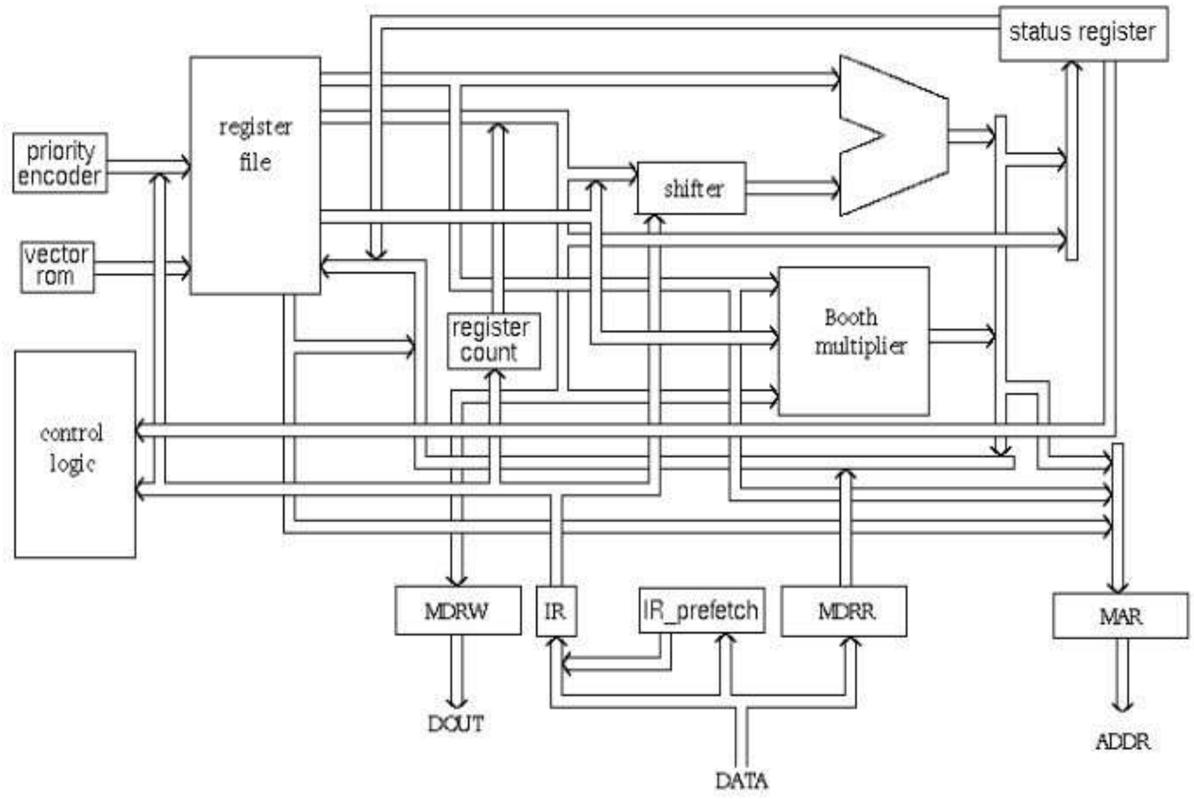


Fig 13 Block diagram

5.5.3 Arithmetic Logic Unit (ALU)

The ALU has two 32-bits inputs. The primary comes from the register file, whereas the other comes from the shifter. Status registers flags modified by the ALU outputs. The V-bit output goes to the V flag as well as the Count goes to the C flag. Whereas the foremost significant bit really represents the S flag, the ALU output operation is done by NORed to get the Z flag. The ALU has a 4-bit function bus that permits up to 16 opcode to be implemented.

5.5.4 Booth Multiplier Factor

The multiplier factor has 3 32-bit inputs and the inputs return from the register file. The multiplier output is barely 32-Least Significant Bits of the merchandise. The entity representation of the multiplier factor is shown in the above block diagram. The multiplication starts whenever the beginning 04 input goes active. Fin of the output goes high when finishing.

5.5.5 Booth Algorithm

Booth algorithm is a noteworthy multiplication algorithmic rule for 2's complement numbers. This treats positive and negative numbers uniformly. Moreover, the runs of 0's or 1's within the multiplier factor are skipped over without any addition or subtraction being performed, thereby creating possible quicker multiplication. The figure shows the simulation results for the multiplier test bench. It's clear that the multiplication finishes only in16 clock cycle.

5.5.6 Barrel Shifter

The barrel shifter features a 32-bit input to be shifted. This input is coming back from the register file or it might be immediate data. The shifter has different control inputs coming back from the instruction register. The Shift field within the instruction controls the operation of the barrel shifter. This field indicates the kind of shift to be performed (logical left or right, arithmetic right or rotate right). The quantity by which the register ought to be shifted is contained in an immediate field within the instruction or it might be the lower 6 bits of a register within the register file.

The shift_val input bus is 6-bits, permitting up to 32 bit shift. The shift type indicates the needed shift sort of 00, 01, 10, 11 are corresponding to shift left, shift right, arithmetic shift right and rotate right, respectively. The barrel shifter is especially created with multiplexers.

5.5.7 Control Unit

For any microprocessor, control unit is the heart of the whole process and it is responsible for the system operation, so the control unit design is the most important part within the whole design. The control unit is sometimes a pure combinational circuit design. Here, the control unit is implemented by easy state machine. The processor timing is additionally included within the control unit. Signals from the control unit are connected to each component within the processor to supervise its operation.

5.5.8 ARM Microcontroller Register Modes

An ARM microcontroller is a load store reducing instruction set computer architecture means the core cannot directly operate with the memory. The data operations must be done by the registers and the information is stored in the memory by an address. The ARM cortex-M3 consists of 37 register sets wherein 31 are general purpose registers and 6 are status registers. The ARM uses seven processing modes to run the user task.

- USER Mode
- FIQ Mode
- IRQ Mode
- SVC Mode
- UNDEFINED Mode
- ABORT Mode
- Monitor Mode

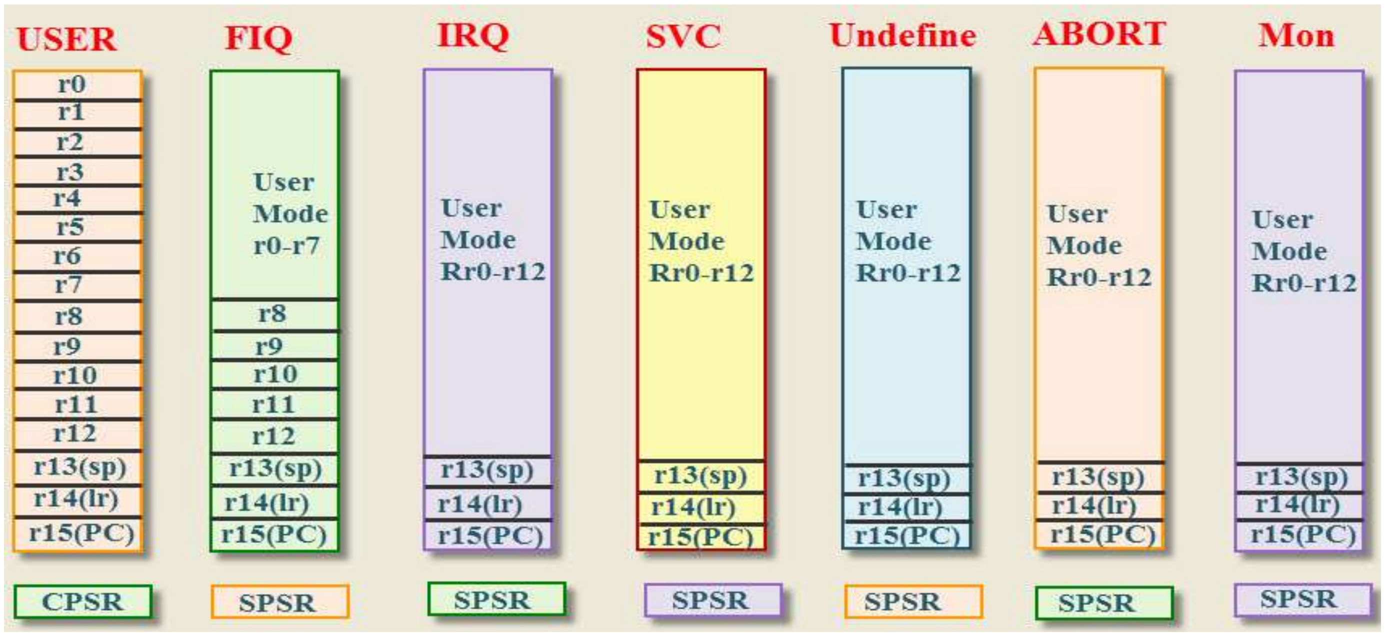


Fig. 14 ARM Microcontroller Register Modes

USER Mode: The user mode is a normal mode, which has the least number of registers. It doesn't have SPSR and has limited access to the CPSR.

FIQ and IRQ: The FIQ and IRQ are the two interrupt caused modes of the CPU. The FIQ is processing interrupt and IRQ is standard interrupt. The FIQ mode has additional five banked registers to provide more flexibility and high performance when critical interrupts are handled.

SVC Mode: The Supervisor mode is the software interrupt mode of the processor to start up or reset.

Undefined Mode: The Undefined mode traps when illegal instructions are executed. The ARM core consists of 32-bit data bus and faster data flow.

THUMB Mode: In THUMB mode 32-bit data is divided into 16-bits and increases the processing speed.

THUMB-2 Mode: In THUMB-2 mode the instructions can be either 16-bit or 32-bit and it increases the performance of the ARM cortex –M3 microcontroller. The ARM cortex-m3 microcontroller uses only THUMB-2 instructions.

Some of the registers are reserved in each mode for the specific use of the core. The reserved registers are

Stack Pointer (SP).

Link Register (LR).

Program Counter (PC).

Current Program Status Register (CPSR).

Saved Program Status Register (SPSR).

The reserved registers are used for specific functions. The SPSR and CPSR contain the status control bits which are used to store the temporary data. The SPSR and CPSR register have some properties that are defined operating modes, Interrupt enable or disable flags and ALU status flag. The ARM core operates in two states 32-bit state or THUMBS state.

5.5.9 ARM-Cortex Microcontroller Programming

In the present days, the microcontroller vendors are offering 32-bit microcontrollers based on ARM cortex-m3 architecture. Many embedded system developers are starting to use these 32-bit microcontrollers for their projects. The ARM microcontrollers supports for both low-level and high level programming languages.

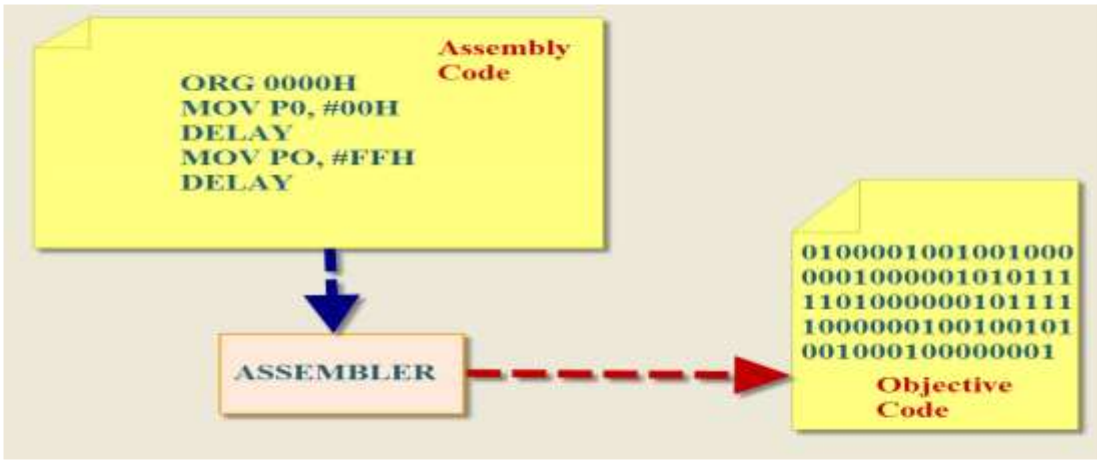


Fig 15 ARM-Cortex Microcontroller Programming

For example the memory size is limited and performance might not be sufficient. The ARM microcontroller's runs at 100 MHz frequency and higher performance, therefore it supports the higher level languages. The ARM microcontroller is programmed with different IDES such as keiluvision3, keiluvision4, coocox and so on. A 8-bit microcontroller use 8-bit instructions and the ARM cortex-M uses a 32-instructions.

Additional Uses of the Cortex Processor

It is a reduced instruction set computing Controller

32-bit high performance central processing unit

3-stage pipeline and compact one

It has THUMB-2 technology

Merges optimally with 16/32 bit instructions

High performance

It supports tools and RTOS and its core Sight debug and trace

JTAG or 2-pin serial wire debugs connection

Support for multiple processors

Low power Modes

It supports sleep modes

Control the software package

Multiple power domains

Nested vectored interrupt controller (NVIC)

Low latency, low noise interrupts response

No need for assembly programming

5.6 Arduino

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. Accepts analog and digital signals as input and gives desired output.

5.6.1 BOARD DETAILS:

- Power Supply:
- USB or power barrel jack
- Voltage Regulator
- LED Power Indicator
- Tx-Rx LED Indicator
- Output power,
- Analog Input Pins
- Digital I/O Pin

ARDUIN0 UN0	
Feature	Value
Operating Voltage	5V
Clock Speed	16MHz
Digital I/O	14
Analog Input	6
PWM	6
UART	1
Interface	USB via ATMega16U2

5.6.2 SET UP:

- Power the board by connecting it to a PC via USB cable
- Launch the Arduino IDE
- Set the board type and the port for the board
- TOOLS -> BOARD -> select your board
- TOOLS -> PORT -> select your port

5.6.3 TYPES:

- Arduino Uno (R3)
- LilyPad Arduino
- RedBoard
- Arduino Mega (R3)

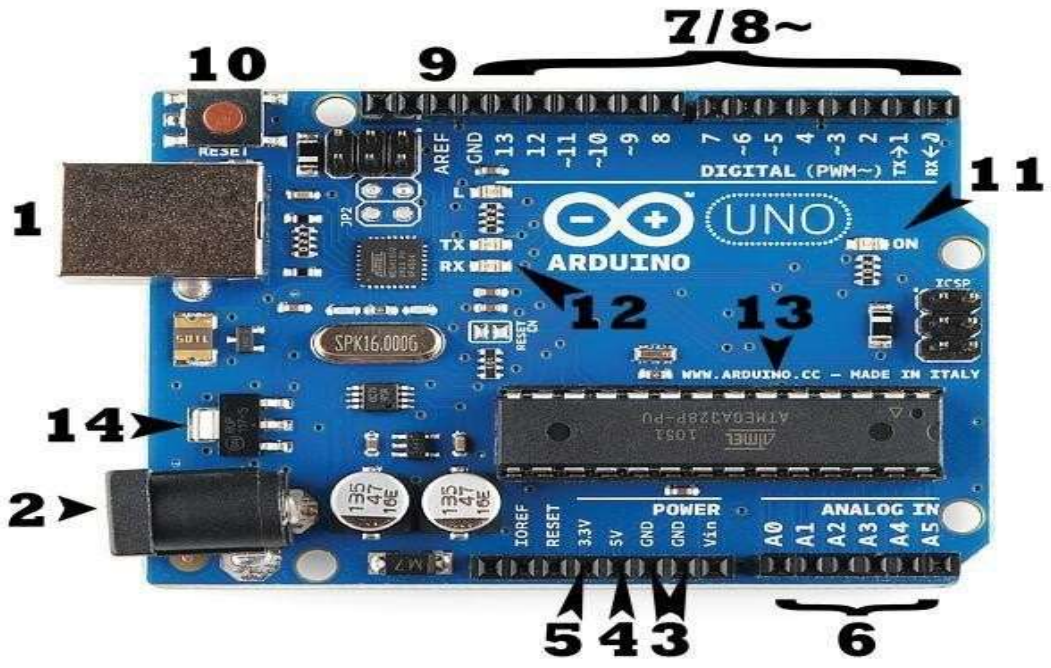


Fig 16 Arduino

5.6.4 Power (USB / Barrel Jack):

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply (like this) that is terminated in a barrel jack. In the picture above the USB connection is labeled (1) and the barrel jack is labeled (2).The USB connection is also how you will load code onto your Arduino board.

NOTE: Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF):

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjunction with a breadboard and some wire. They usually have black plastic ‗headers‘that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

GND (3): Short for ‗Ground‘. There are several GND pins on the Arduino, any of which can be used to ground your circuit.

5V (4) & 3.3V (5): As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.

Analog (6): The area of pins under the ‗Analog in‘label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.

Digital (7): Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

PWM (8): You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).

AREF (9): Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

5.6.5 Reset Button

Just like the original Nintendo, the Arduino has a reset button (10). Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn‘t repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn‘t usually fix any problems.

5.6.6 Power LED Indicator

Just beneath and to the right of the word ―UNO‖ on your circuit board, there‘s a tiny LED next to the word ‗ON‘(11). This LED should light up whenever you plug your Arduino into a power source. If this light doesn‘t turn on, there‘s a good chance something is wrong. Time to re-check your circuit!

5.6.7 TX RX LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs (12). These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we‘re loading a new program onto the board).

5.6.8 Main IC

The black thing with all the metal legs is an IC, or Integrated Circuit (13). Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC‘s from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC‘s, reading the datasheets is often a good idea.

5.6.9 Voltage Regulator

The voltage regulator (14) is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it‘s for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don‘t hook up your Arduino to anything greater than 20 volts.

5.6.10 ARDINO IDE OVERVIEW:

Program coded in Arduino IDE is called a SKETCH

1. To create a new sketch File -> New
2. To open an existing sketch File -> open ->
3. There are some basic ready-to-use sketches available in the EXAMPLES section File -> Examples -> select any program
4. Verify: Checks the code for compilation errors
5. Upload: Uploads the final code to the controller board
6. New: Creates a new blank sketch with basic structure
7. Open: Opens an existing sketch
8. Save: Saves the current sketch

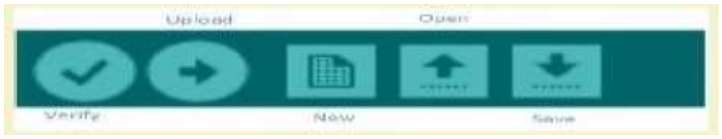


Fig 17 Compilation and Execution

5.6.11 SKETCH STRUCTURE

A sketch can be divided into two parts:

Setup ()

Loop ()

The function setup () is the point where the code starts, just like the main () function in C and C++ I/O Variables, pin modes are initialized in the Setup () function

Loop() function, as the name suggests, iterates the specified task in the program DATA TYPES:

Void, Long, Int, Char, Boolean, Unsigned char, Byte, Unsigned int, Word ,Unsigned long ,Float, Double, Array,String-char array, String-object, Short

5.6.12 Arduino Function libraries

Input/Output Functions:

The arduino pins can be configured to act as input or output pins using the pinMode() function

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

Pin- pin number on the Arduino board Mode- INPUT/OUTPUT

digitalWrite() : Writes a HIGH or LOW value to a digital pin

analogRead() : Reads from the analog input pin i.e., voltage applied across the pin Character functions such as isdigit(), isalpha(), isalnum(), isxdigit(), islower(), isupper(), isspace() return 1(true) or 0(false)

Delay() function is one of the most common time manipulation function used to provide a delay of specified time. It accepts integer value (time in miliseconds)

5.6.13 EXAMPLE BLINKING LED:

Requirement:

- Arduino controller board, USB connector, Bread board, LED, 1.4Kohm resistor, connecting wires, Arduino IDE
- Connect the LED to the Arduino using the Bread board and the connecting wires
- Connect the Arduino board to the PC using the USB connector
- Select the board type and port Write the sketch in the editor, verify andupload Connect the positive terminal of the LED to digital pin 12 and the negative terminal to the ground pin (GND) of Arduino Board

```
void setup()
{
pinMode(12, OUTPUT); // set the pin mode
} void loop()
{
```

digitalWrite(12, HIGH); // Turn on the LED delay(1000); digitalWrite(12, LOW); //Turn of the LED delay(1000);

}

Set the pin mode as output which is connected to the led, pin 12 in this case. Use digitalWrite() function to set the output as HIGH and LOW

Delay () function is used to specify the delay between HIGH-LOW transition of the output

- Connect he board to the PC
- Set the port and board type
- Verify the code and upload, notice the TX – RX led in the board starts flashing as the code is uploaded.

5.7 DIY Kits – Soil moisture monitoring

5.7.1 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.

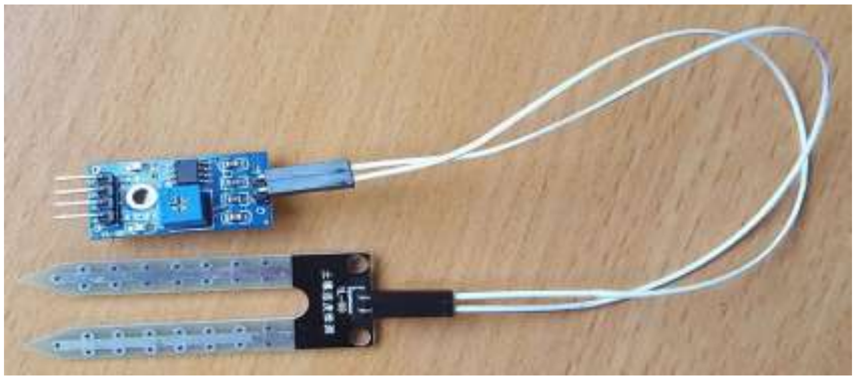


Fig 18 Soil Moisture Sensor

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.

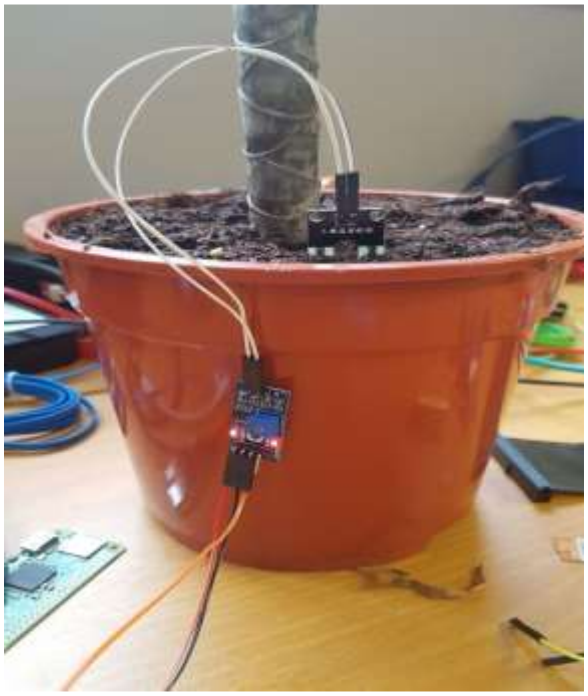


Fig 19 Experimental Setup

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)
```

5.7.2 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.

5.7.2.1 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)

5.7.2.2 Hardware Setup

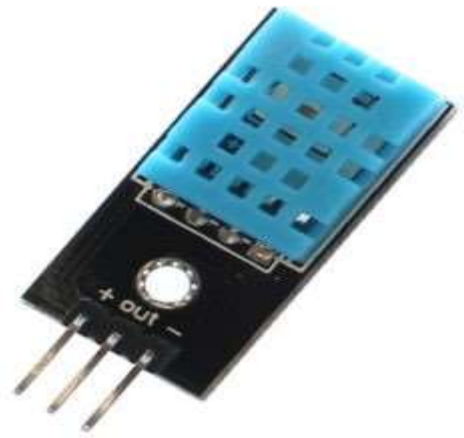


Fig 20 Humidity and Temperature Sensor

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).

5.7.2.3 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!')
```

5.7.3 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

Fig 21 Gas Sensor

.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.

5.7.3.1 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

5.7.3.2 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.

5.7.3.3 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)
```

5.7.3.4 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".

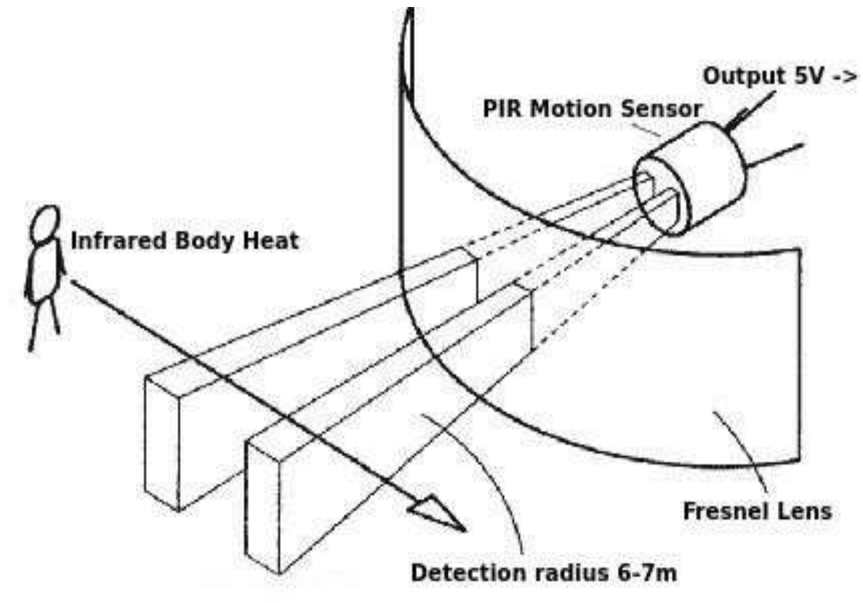


Fig 22 Working of PIR Sensor

5.7.3.5 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
```