LECTURE NOTES ON

INTERNET OF THINGS

B. Tech (R23)

III YEAR I Semester

Prepared by

P.HARI KRISHNA

Assistant Professor M.TECH

UNIT-I
INTRODUCTION OF IOT

IoT comprises things that have unique identities and are connected to internet. By 2020 there will be a total of 50 billion devices /things connected to internet. IoT is not limited to just connecting things to the internet but also allow things to communicate and exchange data.

Definition:
A dynamic global n/w infrastructure with self configuring capabilities based on standard and interoperable communication protocols where physical and virtual –things have identities, physical attributes and virtual personalities and use intelligent interfaces, and are seamlessly integrated into information n/w, often communicate data associated with users and their environments.

Characteristics:

1) **Dynamic & Self Adapting**: IoT devices and systems may have the capability to dynamically adapt with the changing contexts and take actions based on their operating conditions, user_s context or sensed environment.
   Eg: the surveillance system is adapting itself based on context and changing conditions.
2) **Self Configuring:** allowing a large number of devices to work together to provide certain functionality.
3) **Inter Operable Communication Protocols:** support a number of interoperable communication protocols and can communicate with other devices and also with infrastructure.
4) **Unique Identity:** Each IoT device has a unique identity and a unique identifier (IP address).
5) **Integrated into Information Network:** that allow them to communicate and exchange data with other devices and systems.

IoT − Key Features

The most important features of IoT include artificial intelligence, connectivity, sensors, active engagement, and small device use. A brief review of these features is given below −

- **AI** − IoT essentially makes virtually anything "smart", meaning it enhances every aspect of life with the power of data collection, artificial intelligence algorithms, and networks. This can mean something as simple as enhancing your refrigerator and cabinets to detect when milk and your favorite cereal run low, and to then place an order with your preferred grocer.
- **Connectivity** − New enabling technologies for networking, and specifically IoT networking, mean networks are no longer exclusively tied to major providers. Networks can exist on a much smaller and cheaper scale while still being practical. IoT creates these small networks between its system devices.
- **Sensors** − IoT loses its distinction without sensors. They act as defining instruments which transform IoT from a standard passive network of devices into an active system capable of real-world integration.
- **Active Engagement** − Much of today's interaction with connected technology happens through passive engagement. IoT introduces a new paradigm for active content, product, or service engagement.
- **Small Devices** − Devices, as predicted, have become smaller, cheaper, and more powerful over time. IoT exploits purpose-built small devices to deliver its precision, scalability, and versatility.

IoT − Advantages

The advantages of IoT span across every area of lifestyle and business. Here is a list of some of the advantages that IoT has to offer −

- **Improved Customer Engagement** − Current analytics suffer from blind-spots and significant flaws in accuracy; and as noted, engagement remains passive. IoT completely transforms this to achieve richer and more effective engagement with audiences.
- **Technology Optimization** − The same technologies and data which improve the customer experience also improve device use, and aid in more potent improvements to technology. IoT unlocks a world of critical functional and field data.
- **Reduced Waste** − IoT makes areas of improvement clear. Current analytics give us superficial insight, but IoT provides real-world information leading to more effective management of resources.
- **Enhanced Data Collection** − Modern data collection suffers from its limitations and its design for passive use. IoT breaks it out of those spaces, and places it exactly where humans really want to go to analyze our world. It allows an accurate picture of everything.

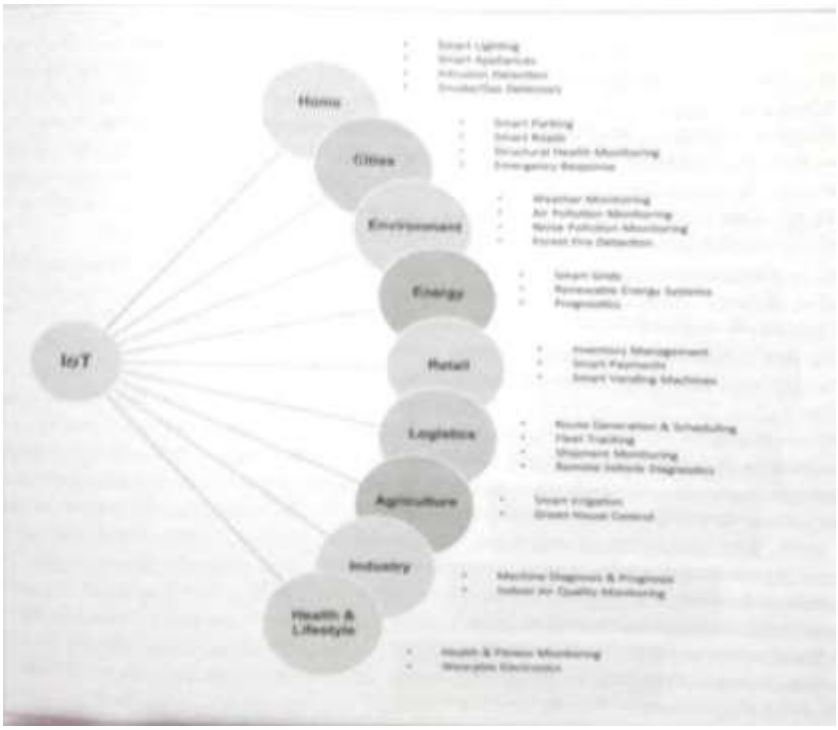
IoT − Disadvantages

Though IoT delivers an impressive set of benefits, it also presents a significant set of challenges. Here is a list of some its major issues −

- **Security** − IoT creates an ecosystem of constantly connected devices communicating over networks. The system offers little control despite any security measures. This leaves users exposed to various kinds of attackers.
- **Privacy** − The sophistication of IoT provides substantial personal data in extreme detail without the user's active participation.
- **Complexity** − Some find IoT systems complicated in terms of design, deployment, and maintenance given their use of multiple technologies and a large set of new enabling technologies.
- **Flexibility** − Many are concerned about the flexibility of an IoT system to integrate easily with another. They worry about finding themselves with several conflicting or locked systems.
- **Compliance** − IoT, like any other technology in the realm of business, must comply with regulations. Its complexity makes the issue of compliance seem incredibly challenging when many consider standard software compliance a battle.

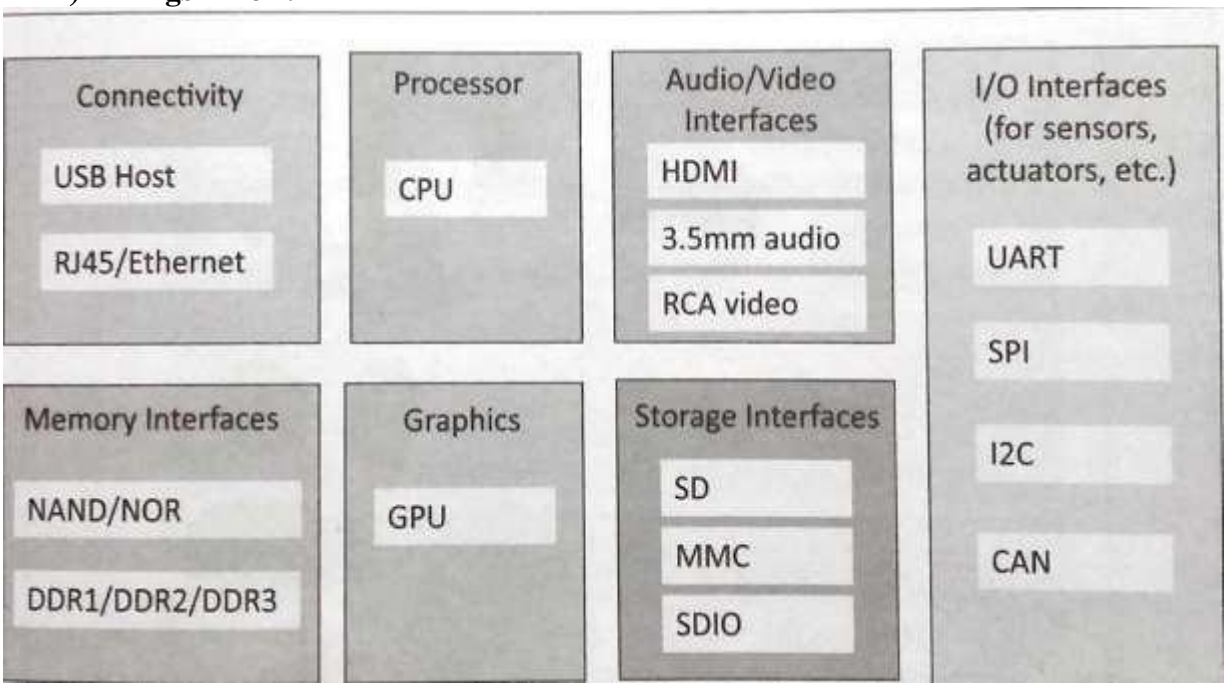
Applications of IoT:

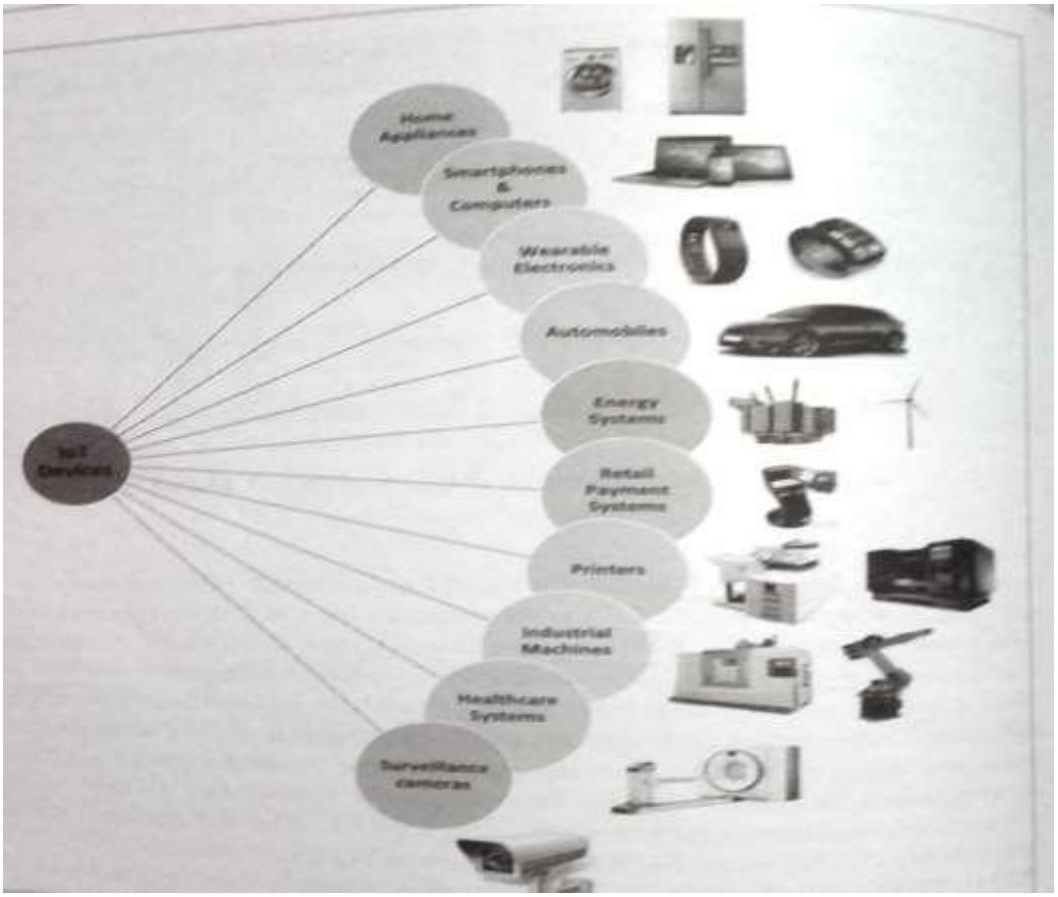
1) Home
2) Cities
3) Environment
4) Energy
5) Retail
6) Logistics
7) Agriculture
8) Industry
9) Health & Life Style



Physical Design of IoT

1) **Things in IoT:**



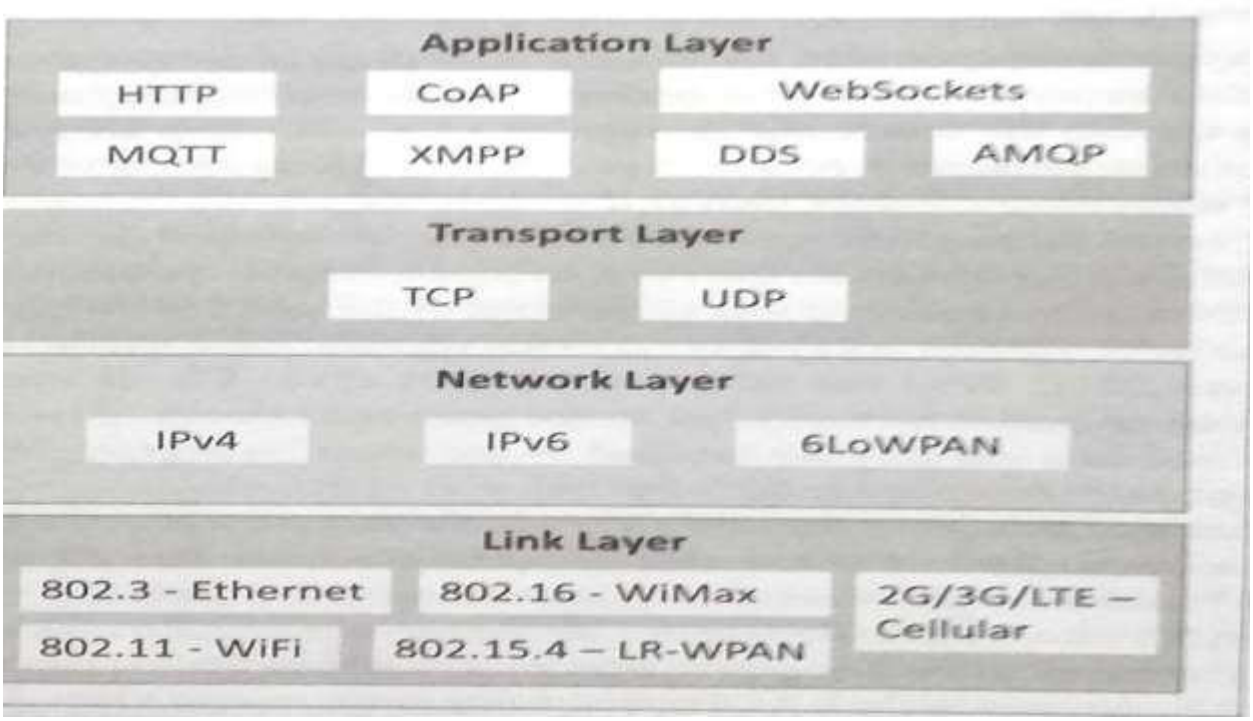


The things in IoT refers to IoT devices which have unique identities and perform remote sensing, actuating and monitoring capabilities. IoT devices can exchange data with other connected devices applications. It collects data from other devices and process data either locally or remotely.

An IoT device may consist of several interfaces for communication to other devices both wired and wireless. These includes (i) I/O interfaces for sensors, (ii) Interfaces for internet connectivity (iii) memory and storage interfaces and (iv) audio/video interfaces.

2) IoT Protocols:

a) Link Layer : Protocols determine how data is physically sent over the network_s physical layer or medium. Local network connect to which host is attached. Hosts on the same link exchange data packets over the link layer using link layer protocols. Link layer determines how packets are coded and signaled by the h/w device over the medium to which the host is attached.



Protocols:

- 802.3-Ethernet: IEEE802.3 is collection of wired Ethernet standards for the link layer. Eg: 802.3 uses co-axial cable; 802.3i uses copper twisted pair connection; 802.3j uses fiber optic connection; 802.3ae uses Ethernet over fiber.
- 802.11-WiFi: IEEE802.11 is a collection of wireless LAN(WLAN) communication standards including extensive description of link layer. Eg: 802.11a operates in 5GHz band, 802.11b and 802.11g operates in 2.4GHz band, 802.11n operates in 2.4/5GHz band, 802.11ac operates in 5GHz band, 802.11ad operates in 60Ghzband.
- 802.16 - WiMax: IEEE802.16 is a collection of wireless broadband standards including exclusive description of link layer. WiMax provide data rates from 1.5 Mb/s to 1Gb/s.
- 802.15.4-LR-WPAN: IEEE802.15.4 is a collection of standards for low rate wireless personal area network(LR-WPAN). Basis for high level communication protocols such as ZigBee. Provides data rate from 40kb/s to250kb/s.
- 2G/3G/4G-Mobile Communication: Data rates from 9.6kb/s(2G) to up to100Mb/s(4G).

B) **Network/Internet Layer:** Responsible for sending IP datagrams from source n/w to destination n/w. Performs the host addressing and packet routing. Datagrams contains source and destination address.

Protocols:

- **IPv4:** Internet Protocol version4 is used to identify the devices on a n/w using a hierarchical addressing scheme. 32 bit address. Allows total of 2**32addresses.
- **IPv6:** Internet Protocol version6 uses 128 bit address scheme and allows 2**128 addresses.

- **6LOWPAN:**(IPv6overLowpowerWirelessPersonalAreaNetwork)operates in 2.4 GHz frequency range and data transfer 250 kb/s.

C) Transport Layer: Provides end-to-end message transfer capability independent of the underlying n/w. Set up on connection with ACK as in TCP and without ACK as in UDP. Provides functions such as error control, segmentation, flow control and congestion control.

Protocols:

- **TCP:** Transmission Control Protocol used by web browsers(along with HTTP and HTTPS), email(along with SMTP, FTP). Connection oriented and stateless protocol. IP Protocol deals with sending packets, TCP ensures reliable transmission of protocols in order. Avoids n/w congestion and congestion collapse.
- **UDP:** User Datagram Protocol is connectionless protocol. Useful in time sensitive applications, very small data units to exchange. Transaction oriented and stateless protocol. Does not provide guaranteed delivery.

D) **Application Layer:** Defines how the applications interface with lower layer protocols to send data over the n/w. Enables process-to-process communication using ports.

Protocols:

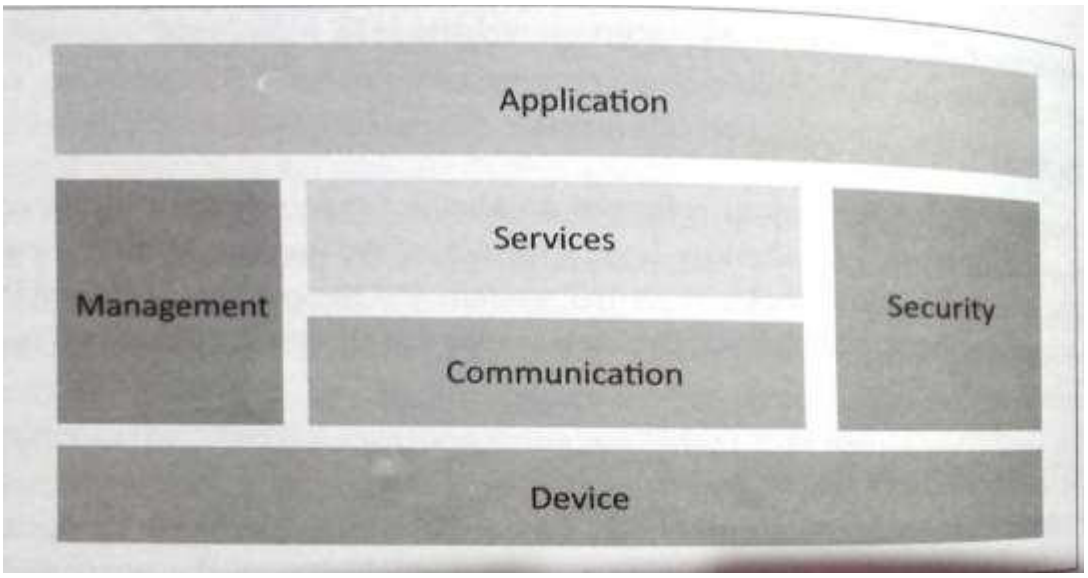
- **HTTP:** Hyper Text Transfer Protocol that forms foundation of WWW. Follow request-response model Stateless protocol.
- **CoAP:** Constrained Application Protocol for machine-to-machine (M2M) applications with constrained devices, constrained environment and constrained n/w. Uses client-server architecture.
- **WebSocket:** allows full duplex communication over a single socket connection.
- **MQTT:** Message Queue Telemetry Transport is light weight messaging protocol based on publish-subscribe model. Uses client server architecture. Well suited for constrained environment.
- **XMPP:** Extensible Message and Presence Protocol for real time communication and streaming XML data between network entities. Support client-server and server-server communication.
- **DDS:** Data Distribution Service is data centric middleware standards for device-to-device or machine-to-machine communication. Uses publish-subscribe model.
- **AMQP:** Advanced Message Queuing Protocol is open application layer protocol for business messaging. Supports both point-to-point and publish-subscribe model.

LOGICAL DESIGN of IoT

Refers to an abstract represent of entities and processes without going into the low level specifies of implementation.

1) IoT Functional Blocks 2) IoT Communication Models 3) IoT Comm. APIs

1) **IoT Functional Blocks:** Provide the system the capabilities for identification, sensing, actuation, communication and management.

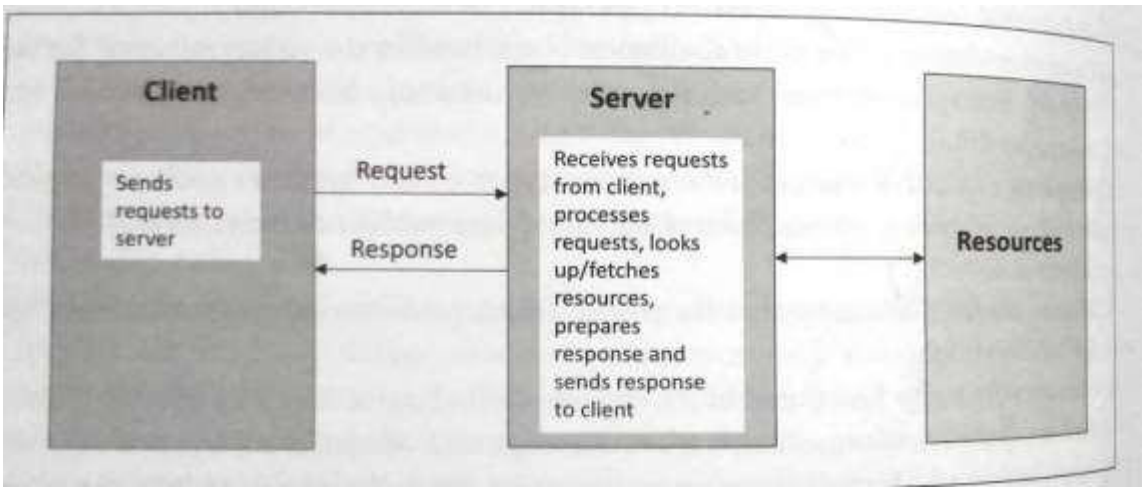


- **Device:** An IoT system comprises of devices that provide sensing, actuation, monitoring and control functions.
- **Communication:** handles the communication for IoTsystem.
- **Services:** for device monitoring, device control services, data publishing services and services for device discovery.
- **Management:** Provides various functions to govern the IoT system.
- **Security:** Secures IoT system and priority functions such as authentication ,authorization, message and context integrity and data security.
- **Application:** IoT application provide an interface that the users can use to control and monitor various aspects of IoT system.

2) IoT Communication Models:

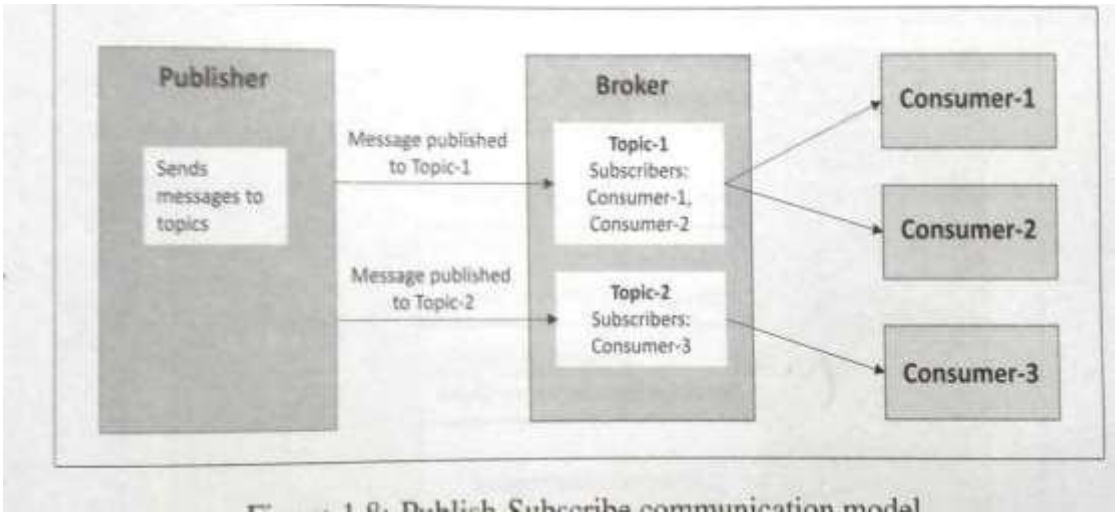
1) Request-Response 2) Publish-Subscibe 3)Push-Pull4) ExclusivePair

1) Request-Response Model:



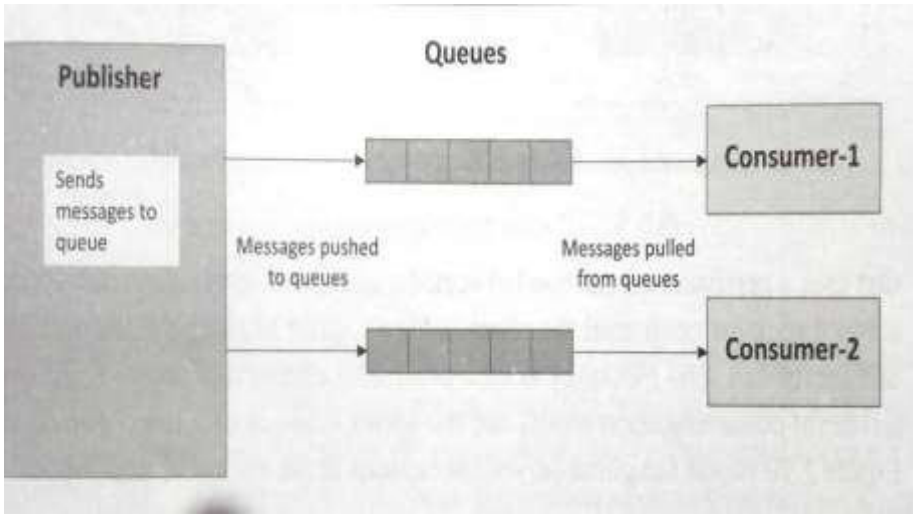
In which the client sends request to the server and the server replies to requests. Is a stateless communication model and each request-response pair is independent of others.

2) **Publish-Subscibe Model:**

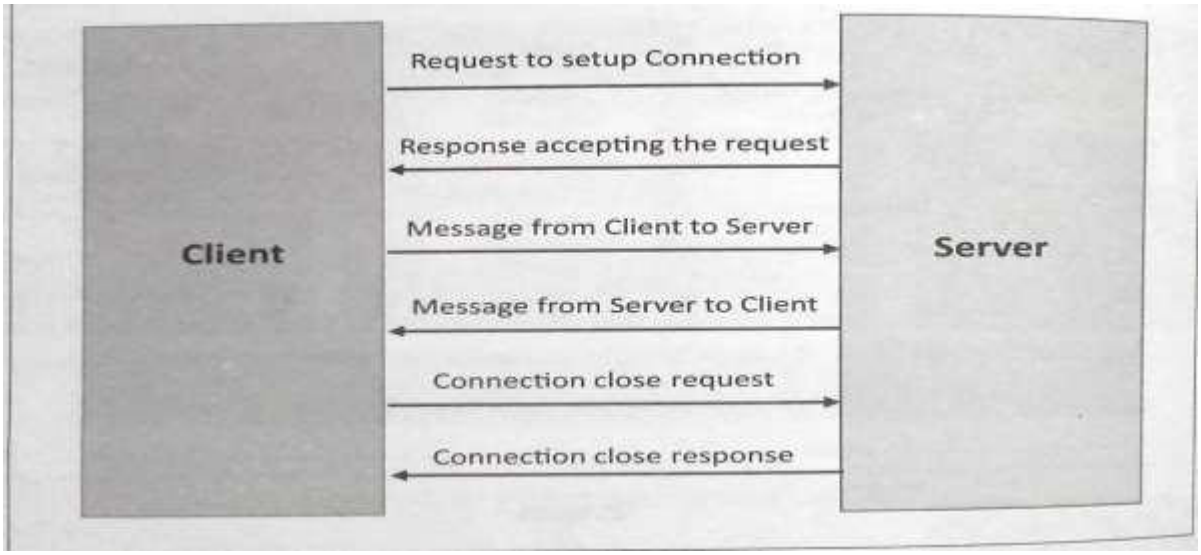


Involves publishers, brokers and consumers. Publishers are source of data. Publishers send data to the topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics which are managed by the broker. When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers.

3) **Push-Pull Model:** in which data producers push data to queues and consumers pull data from the queues. Producers do not need to aware of the consumers. Queues help in decoupling the message between the producers and consumers.



4) **Exclusive Pair:** is bi-directional, fully duplex communication model that uses a persistent connection between the client and server. Once connection is set up it remains open until the client send a request to close the connection. Is a stateful communication model and server is aware of all the open connections.



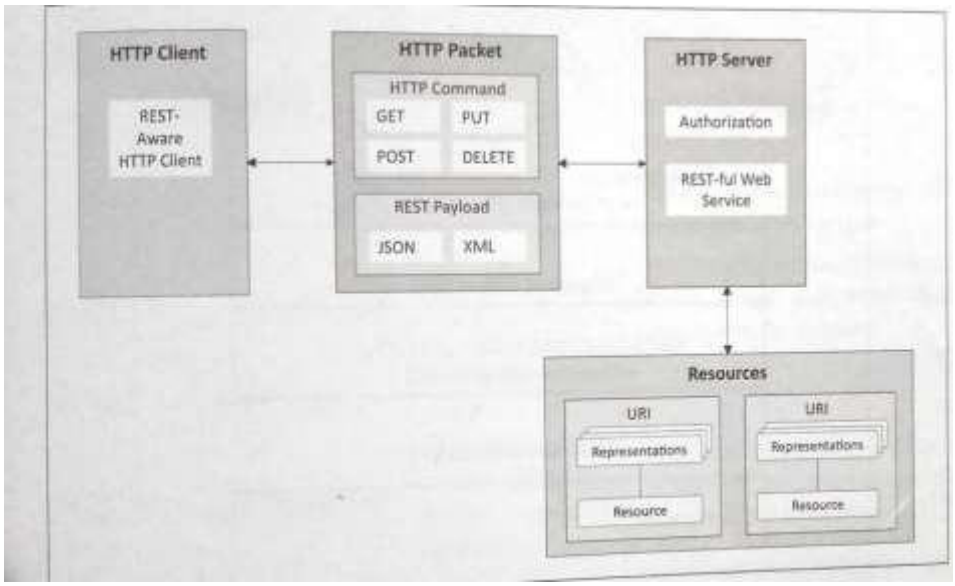
3) IoT Communication APIs:

a) REST based communication APIs(Request-Response Based Model)

b) WebSocket based Communication APIs(Exclusive PairBased Model)

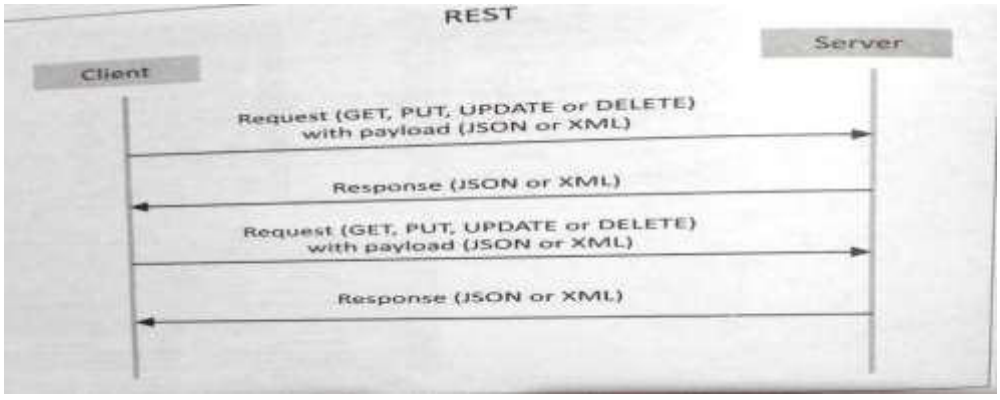
a) REST based communication APIs: Representational State Transfer(REST) is a set of architectural principles by which we can design web services and web APIs that focus on a system_s resources and have resource states are addressed and transferred.

The REST architectural constraints: Fig. shows communication between client server with REST APIs.



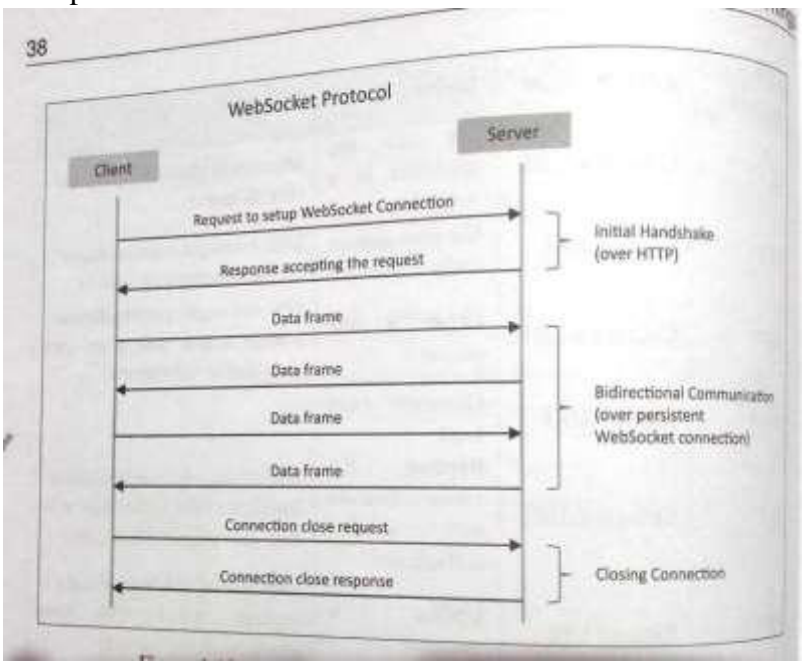
Client-Server: The principle behind client-server constraint is the separation of concerns. Separation allows client and server to be independently developed and updated.
Stateless: Each request from client to server must contain all the info. Necessary to understand the request, and cannot take advantage of any stored context on the server.
Cache-able: Cache constraint requires that the data within a response to a request be implicitly or explicitly labeled as cache-able or non-cacheable. If a response is cache-able, then a client cache is given the right to reuse that response data for later, equivalent requests.
Layered System: constraints the behavior of components such that each component cannot see beyond the immediate layer with which they are interacting.
User Interface: constraint requires that the method of communication between a client and a server must be uniform.
Code on Demand: Servers can provide executable code or scripts for clients to execute in their context. This constraint is the only one that is optional.

Request-Response model used by REST:



RESTful web service is a collection of resources which are represented by URIs. RESTful web API has a base URI(e.g: http://example.com/api/tasks/). The clients and requests to these URIs using the methods defined by the HTTP protocol(e.g: GET, PUT, POST or DELETE). A RESTful web service can support various internet media types.

b) WebSocket Based Communication APIs: WebSocket APIs allow bi-directional, full duplex communication between clients and servers. WebSocket APIs follow the exclusive pair communication model.



IoT Enabling Technologies

IoT is enabled by several technologies including Wireless Sensor Networks, Cloud Computing, Big Data Analytics, Embedded Systems, Security Protocols and architectures, Communication Protocols, Web Services, Mobile internet and semantic search engines.

1) **Wireless Sensor Network(WSN):** Comprises of distributed devices with sensors which are used to monitor the environmental and physical conditions. Zig Bee is one of the most popular wireless technologies used byWSNs.
   WSNs used in IoT systems are described as follows:
   - Weather Monitoring System: in which nodes collect temp, humidity and other data, which is aggregated and analyzed.
   - Indoor air quality monitoring systems: to collect data on the indoor air quality and concentration of various gases.
   - Soil Moisture Monitoring Systems: to monitor soil moisture at variouslocations.
   - Surveillance Systems: use WSNs for collecting surveillance data(motiondata detection).
   - Smart Grids : use WSNs for monitoring grids at variouspoints.

- Structural Health Monitoring Systems: Use WSNs to monitor the health of structures(building, bridges) by collecting vibrations from sensor nodes deployed at various points in the structure.

2) **Cloud Computing:** Services are offered to users in different forms.
   - Infrastructure-as-a-service(IaaS):provides users the ability to provision computing and storage resources. These resources are provided to the users as a virtual machine instances and virtual storage.
   - Platform-as-a-Service(PaaS): provides users the ability to develop and deploy application in cloud using the development tools, APIs, software libraries and services provided by the cloud service provider.
   - Software-as-a-Service(SaaS): provides the user a complete software application or the user interface to the application itself.

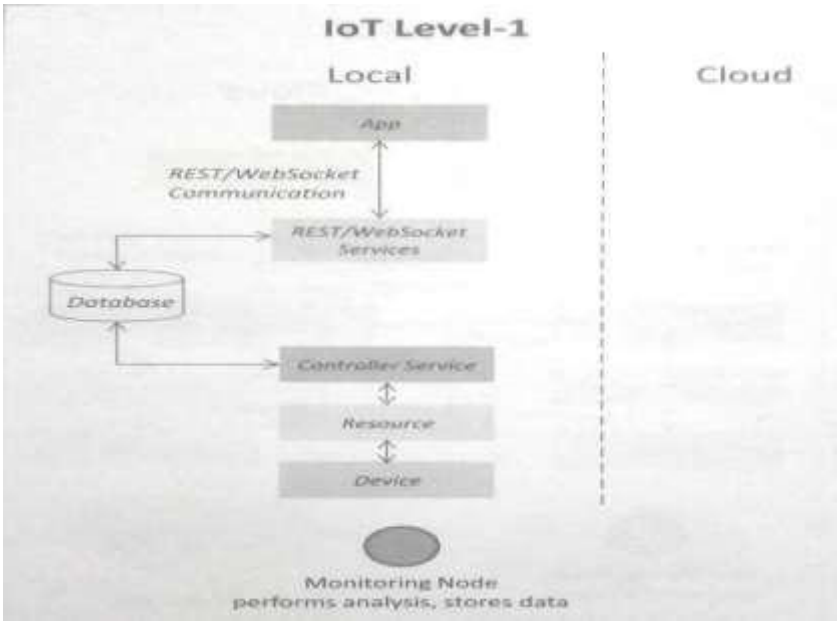
3) **Big Data Analytics:** Some examples of big data generated by IoT are
   - Sensor data generated by IoT systems.
   - Machine sensor data collected from sensors established in industrial and energy systems.
   - Health and fitness data generated IoT devices.
   - Data generated by IoT systems for location and tracking vehicles.
   - Data generated by retail inventory monitoring systems.

4) **Communication Protocols:** form the back-bone of IoT systems and enable network connectivity and coupling to applications.
   - Allow devices to exchange data over network.
   - Define the exchange formats, data encoding addressing schemes for device and routing of packets from source to destination.
   - It includes sequence control, flow control and retransmission of lost packets.

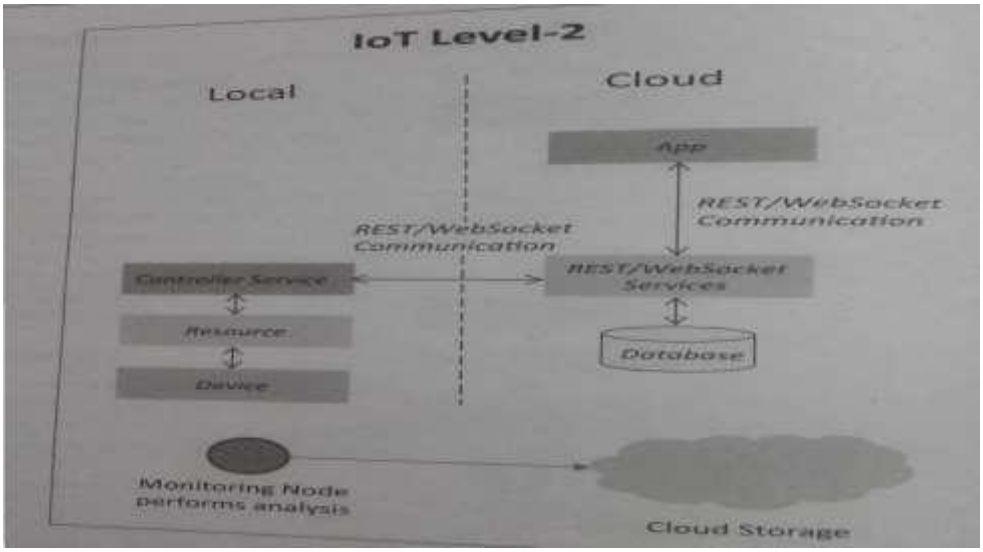
5) **Embedded Systems:** is a computer system that has computer hardware and software embedded to perform specific tasks. Embedded System range from low cost miniaturized devices such as digital watches to devices such as digital cameras, POS terminals, vending machines, appliances etc.,

IoT Levels and Deployment Templates

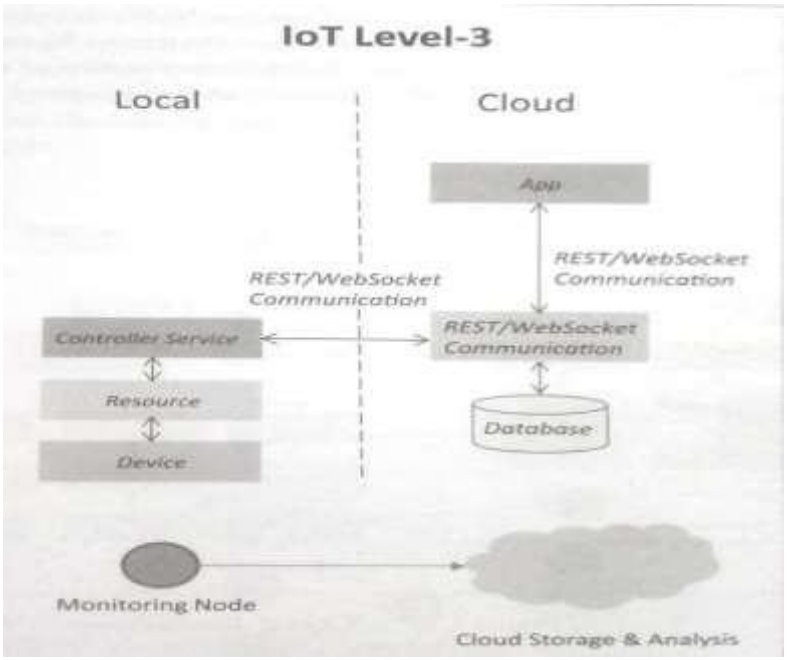
1) **IoT Level1:** System has a single node that performs sensing and/or actuation, stores data, performs analysis and host the application as shown in fig. Suitable for modeling low cost and low complexity solutions where the data involved is not big and analysis requirement are not computationally intensive. An e.g., of IoT Level1 is Home automation.



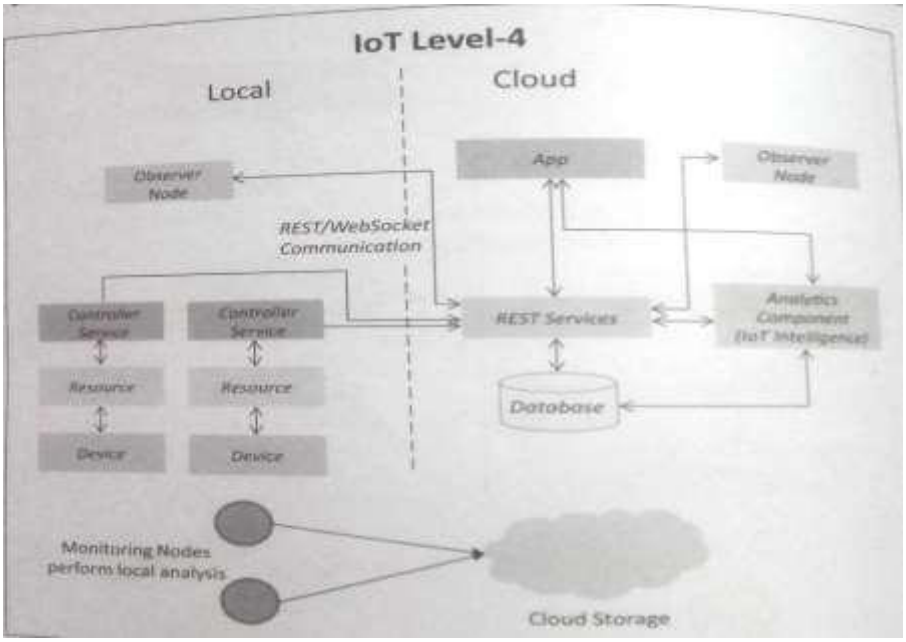
2) **IoT Level2:** has a single node that performs sensing and/or actuating and local analysis as shown in fig. Data is stored in cloud and application is usually cloud based. Level2 IoT systems are suitable for solutions where data are involved is big, however, the primary analysis requirement is not computationally intensive and can be done locally itself. An e,g.**,** of Level2 IoT system for Smart Irrigation.



3) **IoT Level3:** system has a single node. Data is stored and analyzed in the cloud application is cloud based as shown in fig. Level3 IoT systems are suitable for solutions where the data involved is big and analysis requirements are computationally intensive. An example of IoT level3 system for tracking package handling.

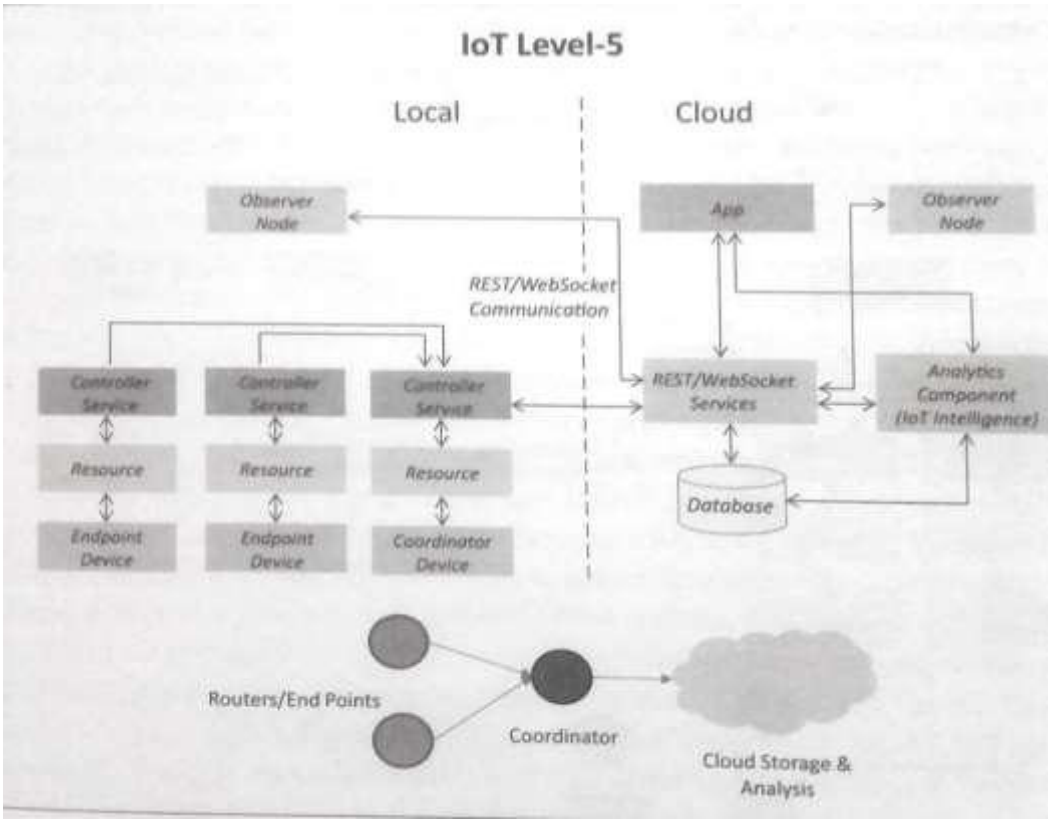


4) **IoT Level4:** System has multiple nodes that perform local analysis. Data is stored in the cloud and application is cloud based as shown in fig. Level4 contains local and cloud based observer nodes which can subscribe to and receive information collected in the cloud from IoT devices. An example of a Level4 IoT system for Noise Monitoring.

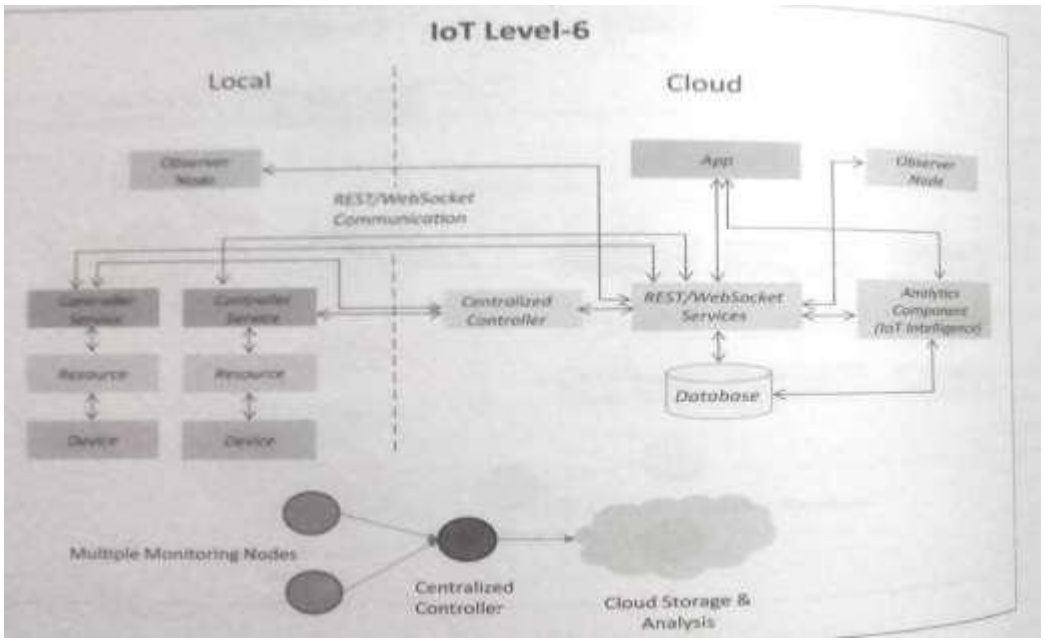


5) **IoT Level5:** System has multiple end nodes and one coordinator node as shown in fig. The end nodes that perform sensing and/or actuation. Coordinator node collects data from theendnodesandsendstothecloud.Dataisstoredandanalyzedinthecloudand

application is cloud based. Level5 IoT systems are suitable for solution based on wireless sensor network, in which data involved is big and analysis requirements are computationally intensive. An example of Level5 system for Forest Fire Detection.



6) **IoT Level6:** System has multiple independent end nodes that perform sensing and/or actuation and sensed data to the cloud. Data is stored in the cloud and application is cloud based as shown in fig. The analytics component analyses the data and stores the result in the cloud data base. The results are visualized with cloud based application. The centralized controller is aware of the status of all the end nodes and sends control commands to nodes. An example of a Level6 IoT system for Weather Monitoring System.



DOMAIN SPECIFIC IoTs

1) **Home Automation:**
   a) **Smart Lighting:** helps in saving energy by adapting the lighting to the ambient conditions and switching on/off or diming the light when needed.
   b) **Smart Appliances:** make the management easier and also provide status information to the users remotely.
   c) **Intrusion Detection:** use security cameras and sensors(PIR sensors and door sensors) to detect intrusion and raise alerts. Alerts can be in the form of SMS or email sent to the user.
   d) **Smoke/Gas Detectors:** Smoke detectors are installed in homes and buildings to detect smoke that is typically an early sign of fire. Alerts raised by smoke detectors can be in the form of signals to a fire alarm system. Gas detectors can detect the presence of harmful gases such as CO, LPGetc.,

2) **Cities:**
   a) **Smart Parking:** make the search for parking space easier and convenient for drivers. Smart parking are powered by IoT systems that detect the no. of empty parking slots and send information over internet to smart application backends.
   b) **Smart Lighting:** for roads, parks and buildings can help in saving energy.
   c) **Smart Roads:** Equipped with sensors can provide information on driving condition, travel time estimating and alert in case of poor driving conditions, traffic condition and accidents.
   d) **Structural Health Monitoring:** uses a network of sensors to monitor the vibration levels in the structures such as bridges and buildings.
   e) **Surveillance:** The video feeds from surveillance cameras can be aggregated in cloud based scalable storage solution.

f) **Emergency Response:** IoT systems for fire detection, gas and water leakage detection can help in generating alerts and minimizing their effects on the critical infrastructures.

3) **Environment:**
   a) **Weather Monitoring:** Systems collect data from a no. of sensors attached and send the data to cloud based applications and storage back ends. The data collected in cloud can then be analyzed and visualized by cloud based applications.
   b) **Air Pollution Monitoring:** System can monitor emission of harmful gases(CO2, CO, NO, NO2 etc.,) by factories and automobiles using gaseous and meteorological sensors. The collected data can be analyzed to make informed decisions on pollutions control approaches.
   c) **Noise Pollution Monitoring:** Due to growing urban development, noise levels in cities have increased and even become alarmingly high in some cities. IoT based noise pollution monitoring systems use a no. of noise monitoring systems that are deployed at different places in a city. The data on noise levels from the station is collected on servers or in the cloud. The collected data is then aggregated to generate noise maps.
   d) **Forest Fire Detection:** Forest fire can cause damage to natural resources, property and human life. Early detection of forest fire can help in minimizing damage.
   e) **River Flood Detection:** River floods can cause damage to natural and human resources and human life. Early warnings of floods can be given by monitoring the water level and flow rate. IoT based river flood monitoring system uses a no. of sensor nodes that monitor the water level and flow rate sensors.

4) **Energy:**
   a) **Smart Grids:** is a data communication network integrated with the electrical grids that collects and analyze data captured in near-real-time about power transmission, distribution and consumption. Smart grid technology provides predictive information and recommendations to utilities, their suppliers, and their customers on how best to manage power. By using IoT based sensing and measurement technologies, the health of equipment and integrity of the grid can be evaluated.
   b) **Renewable Energy Systems:** IoT based systems integrated with the transformers at the point of interconnection measure the electrical variables and how much power is fed into the grid. For wind energy systems, closed-loop controls can be used to regulate the voltage at point of interconnection which coordinate wind turbine outputs and provides power support.
   c) **Prognostics:** In systems such as power grids, real-time information is collected using specialized electrical sensors called Phasor Measurment Units(PMUs) at the substations. The information received from PMUs must be monitored in real-time for estimating the state of the system and for predicting failures.

5) **Retail:**
   a) **Inventory Management:** IoT systems enable remote monitoring of inventory using data collected by RFIDreaders.

b) **Smart Payments:** Solutions such as contact-less payments powered by technologies such as Near Field Communication(NFC) and Bluetooth.
c) **Smart Vending Machines:** Sensors in a smart vending machines monitors its operations and send the data to cloud which can be used for predictive maintenance.

6) **Logistics:**
   a) **Route generation & scheduling:** IoT based system backed by cloud can provide first response to the route generation queries and can be scaled upto serve a large transportation network.
   b) **Fleet Tracking:** Use GPS to track locations of vehicles inreal-time.
   c) **Shipment Monitoring:** IoT based shipment monitoring systems use sensors such as temp, humidity, to monitor the conditions and send data to cloud, where it can be analyzed to detect foods poilage.
   d) **Remote Vehicle Diagnostics:** Systems use on-board IoT devices for collecting data on Vehicle operations(speed, RPMetc.,) and status of various vehicle subsystems.

7) **Agriculture:**
   a) **Smart Irrigation:** to determine moisture amount in soil.
   b) **Green House Control:** to improve productivity.

8) **Industry:**
   a) Machine diagnosis and prognosis
   b) Indoor Air Quality Monitoring

9) **Health and LifeStyle:**
   a) Health & Fitness Monitoring
   b) Wearable Electronics

Reference Books:

1. Jan Holler, VlasiosTsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, ― From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence‖, 1st Edition, Academic Press, 2014.
2. ArshdeepBahga, Vijay Madisetti - Internet of Things: A Hands-On Approach, Universities Press, 2014.
3. The Internet of Things, Enabling technologies and use cases – Pethuru Raj, Anupama C. Raman, CRC Press.
4. Francis daCosta, ―Rethinking the Internet of Things: A Scalable Approach to Connecting Everything‖, 1st Edition, Apress Publications, 2013
5. Cuno Pfister, Getting Started with the Internet of Things, O‟Reilly Media, 2011, ISBN: 9781- 4493- 9357-1
6. DGCA RPAS Guidance Manual, Revision 3 – 2020

7. Building Your Own Drones: A Beginners' Guide to Drones, UAVs, and ROVs, John Baic

UNIT 2

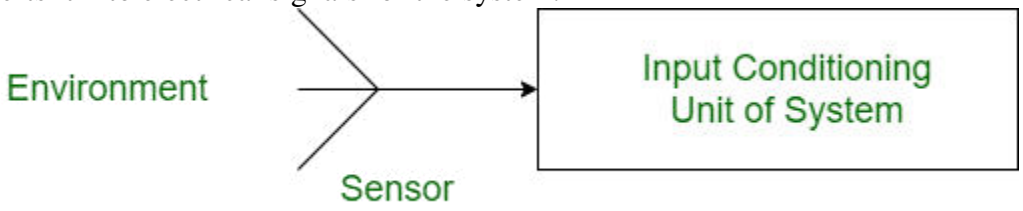
PROTOTYPING IOT OBJECTS USING MICRO PROCESSOR/MICRO CONTROLLER

Sensors and ***actuators*** are critical components of ***embedded systems***. Sensors and actuators differ primarily in their purpose; the sensor is utilized to track environmental changes using measurands, whereas the actuator is utilized when monitoring is combined with control, such as controlling physical changes.

SENSORS:

- A ***sensor*** is a device that detects changes and events in a physical environment. It may convert physical parameters like ***humidity, pressure, temperature, heat, motion, etc.***, into electrical signals.
- This signal can be converted into a human-readable display and sent across a network for additional processing.

A sensor works by sensing a quantity by utilizing a particular detecting device. Each sensor operates on a distinct principle, such as an electromagnetic sensor, a resistive sensor, a capacitor sensor, etc. In general, they sense the matching attribute in the environment and convert it into a proportional magnitude electrical signal.**1. Sensor:** Sensor is a device used for the conversion of physical events or characteristics into the electrical signals. This is a hardware device that takes the input from environment and gives to the system by converting it.
For example, a thermometer takes the temperature as physical characteristic and then converts it into electrical signals for the system.



TYPES OF SENSORS:

1. Active Sensor
2. Passive Sensor

- ***Active sensors*** necessitate a power supply, whereas ***passive sensors*** don't require a power supply.

  Eg: There are various types of sensors available, including temperature, ultrasonic, pressure, and location sensors, among others. They are utilized for detecting and measuring the relevant quantities.

Some important ***sensors*** are as follows:

1. Biosensors

These biosensors utilize electrochemical technology. These sensors are used in medical, food, and water testing devices. These biosensors also aid in analyzing proteins, cells, nucleic acid, etc.

2. Accelerometers

These sensors utilize the Micro Electro Mechanical Sensor Technology. These sensors utilize in patient monitoring, vehicle systems, etc.

3. Image Sensors

These sensors utilize the Complementary Metal Oxide Sensor mechanism. They detect and transfer data that is utilized to make an image. These image sensors are very useful in consumer surveillance and electronic systems.

3. **Chemical Sensors**

These sensors use ultrasonic, microwave, and radar technology, and they are used in security systems, video games, and other applications.

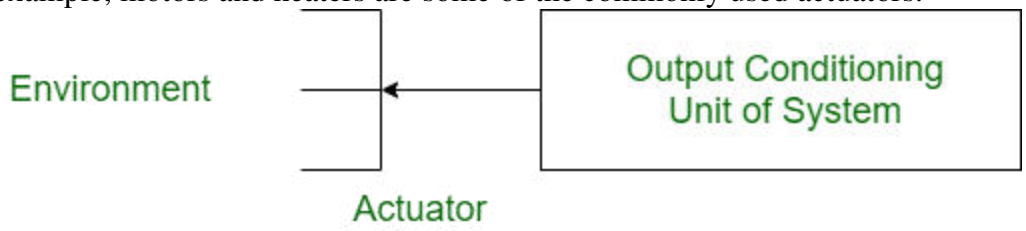
Features of Sensors

There are various features of ***Sensors***. Some main features of Sensors are as follows:

1. A sensor could be either active or passive. Active sensors necessitate a power source, but passive doesn't necessitate a power source.
2. It is a device that monitors and measures changes in the environment.
3. It is responsible for converting physical quantities into electrical signals.
4. It is connected to a system's input.
5. It generates an electrical signal as its output.

2.Actuator:

Actuator is a device that converts the electrical signals into the physical events or characteristics. It takes the input from the system and gives output to the environment. For example, motors and heaters are some of the commonly used actuators.



- A device that changes electrical signals into mechanical work is known as an ***actuator***. It is used to cause movement or a change in the surroundings.
- ***Actuators*** are connected to a system's output. It receives an electrical signal as input and produces mechanical movement as output. It receives input or instruction from a system or a signal conditioning device and outputs it to the environment.
- The ***actuator*** is dependent on the sensor data. The sensor sends data to a signal condition unit, which analyzes the data or information and transmits commands to the actuator depending on that data. A ***"temperature control system"*** is an instance of an actuator system in which a temperature sensor manages the temperature. If the temperature surpasses a specific limit, the device instructs the fan to increase its speed and decrease the temperature.

Types of Actuators

There are various types of ***actuators***. Some of these are as follows:

1. Manual Actuator

This type of actuator is manually operated via gears, levers, and wheels, among other things. They do not need a power source because they are powered by human action.

2. Spring Actuator

It has a loaded spring that is triggered and released to generate mechanical work. It may be triggered in several ways.

3. Hydraulic Actuator

Hydraulic actuators generate pressure by compressing fluid in a cylinder, allowing mechanical movement.

4. Electric Actuators

These actuators require power to function. It utilizes an electric motor to produce movement. They are quick and effective.

Features of Actuators

There are various features of ***Actuators***. Some main features of Actuators are as follows:

1. The actuator assists in managing the environment based on sensor readings.

2. A device that converts electrical signals into mechanical movement is known as an actuator.
3. It requires an additional power source to function.
4. It receives an electrical signal as input.
5. It is connected to a system's output.

It produces mechanical work.**Difference between Sensor and Actuator :**

SENSOR	ACTUATOR
It converts physical characteristics into electrical signals.	It converts electrical signals into physical characteristics.
It takes input from environment.	It takes input from output conditioning unit of system.
It gives output to input conditioning unit of system.	It gives output to environment.
Sensor generated electrical signals.	Actuator generates heat or motion.
It is placed at input port of the system.	It is placed at output port of the system.
It is used to measure the physical quantity.	It is used to measure the continuous and discrete process parameters.
It gives information to the system about environment.	It accepts command to perform a function.
Example: Photo-voltaic cell which converts light energy into electrical energy.	Example: Stepper motor where electrical energy drives the motor.

Setting up the board-programming for iot:

Setting up the board programming for IoT involves several steps, depending on the specific board you are using and the programming language you want to use. Here are some general steps you can follow:

1. Choose your board: There are many IoT boards available, such as Arduino, Raspberry Pi, ESP8266, etc. Choose a board that fits your project requirements.
2. Install the necessary software: You will need to install the necessary software on your computer to program the board. For example, if you are using an Arduino board, you will need to install the Arduino IDE.
3. Connect your board: Connect your board to your computer using a USB cable or another suitable method.

4. Write your code: Write your code in a programming language supported by your board. For example, Arduino boards use a version of C++, while Raspberry Pi boards can use Python or other languages.
5. Upload your code: Upload your code to the board using the software you installed earlier. This will program the board to execute your code.
6. Test your code: Test your code by connecting sensors or other devices to the board and seeing if it works as intended.
7. Deploy your project: Once your code is working correctly, deploy your project to the intended location, such as a remote server or IoT device.

These are the general steps involved in setting up the board programming for IoT. However, the specifics of each step will vary depending on your board, programming language, and project requirements.

Installation of Arduino UNO:

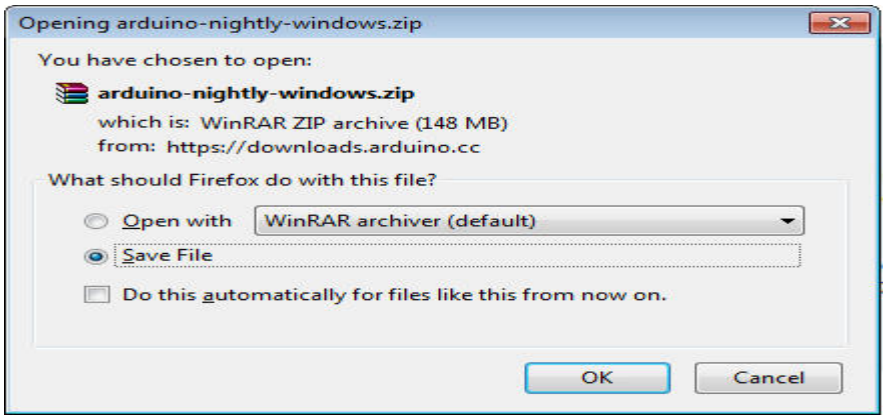
After learning about the main parts of the Arduino UNO board, we go for how to set up the Arduino IDE. we will be ready to upload our program on the Arduino board.our computer and prepare the board to receive the program via USB cable.

Step 1 − First you must have your Arduino board and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.

Step 2 − Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



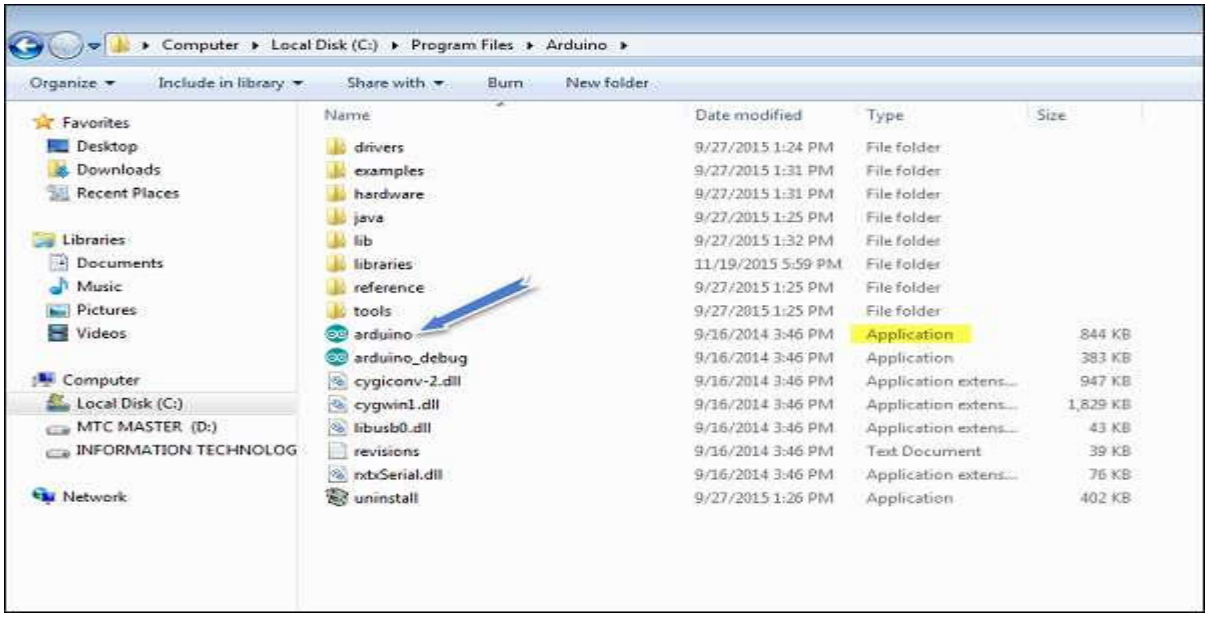
Step 3 − Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED should glow.

Step 4 − Launch Arduino IDE.

After Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

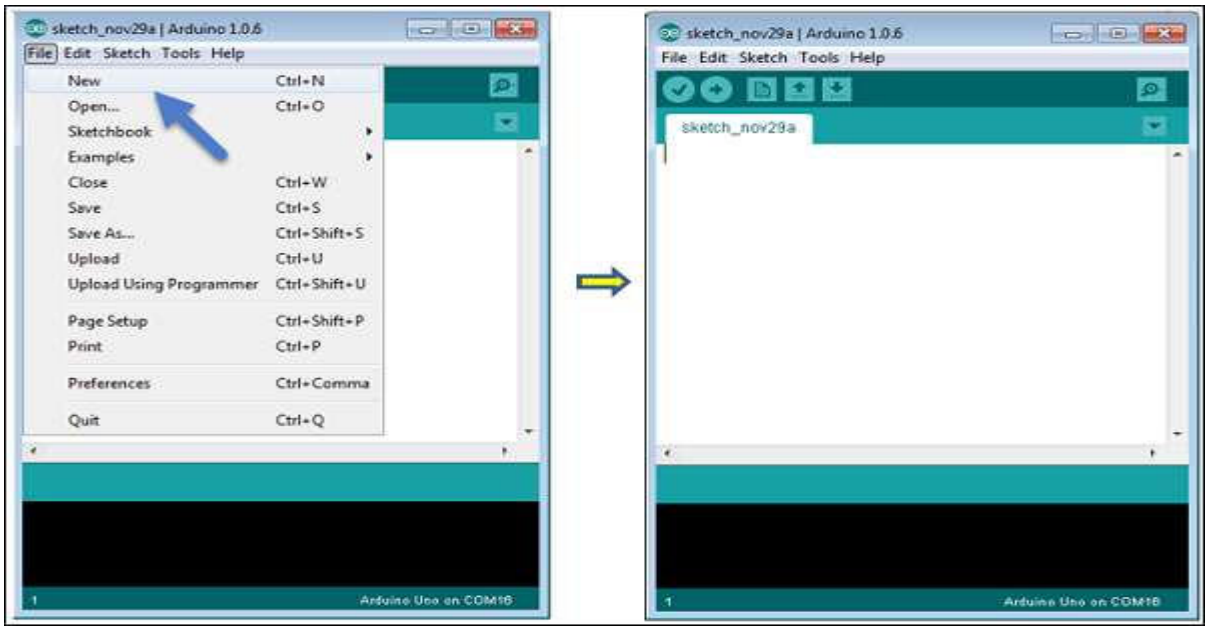


Step 5 − Open your first project.

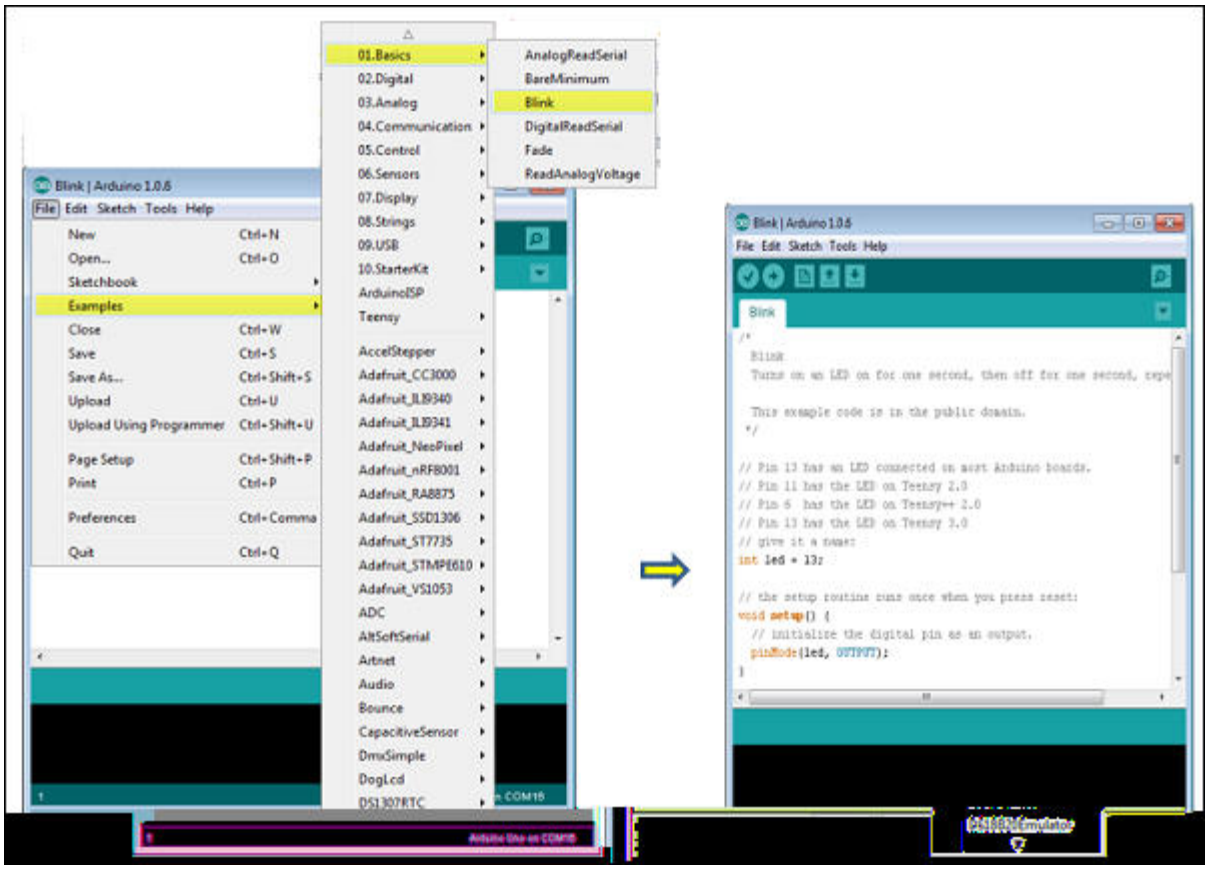
Once the software starts, you have two options −

- Create a new project.
- Open an existing project example.

To create a new project, select File → **New**.



To open an existing project example, select File → Example → Basics → Blink.

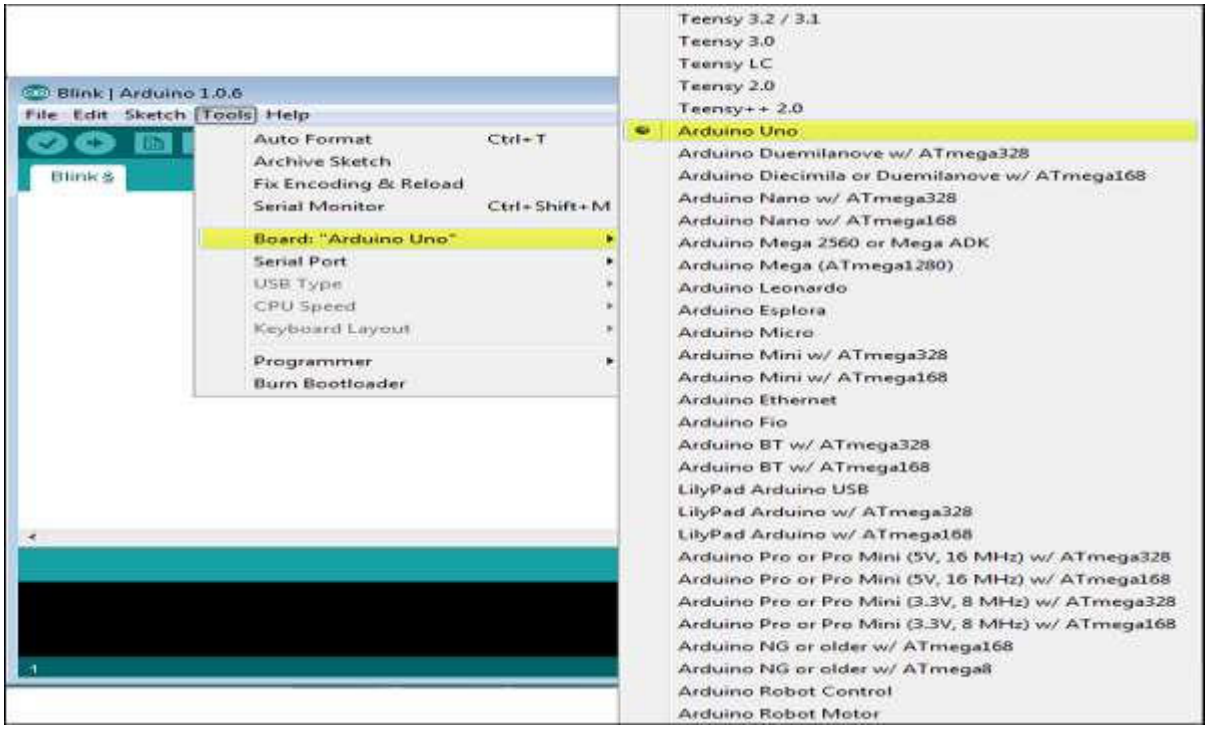


Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6 − Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

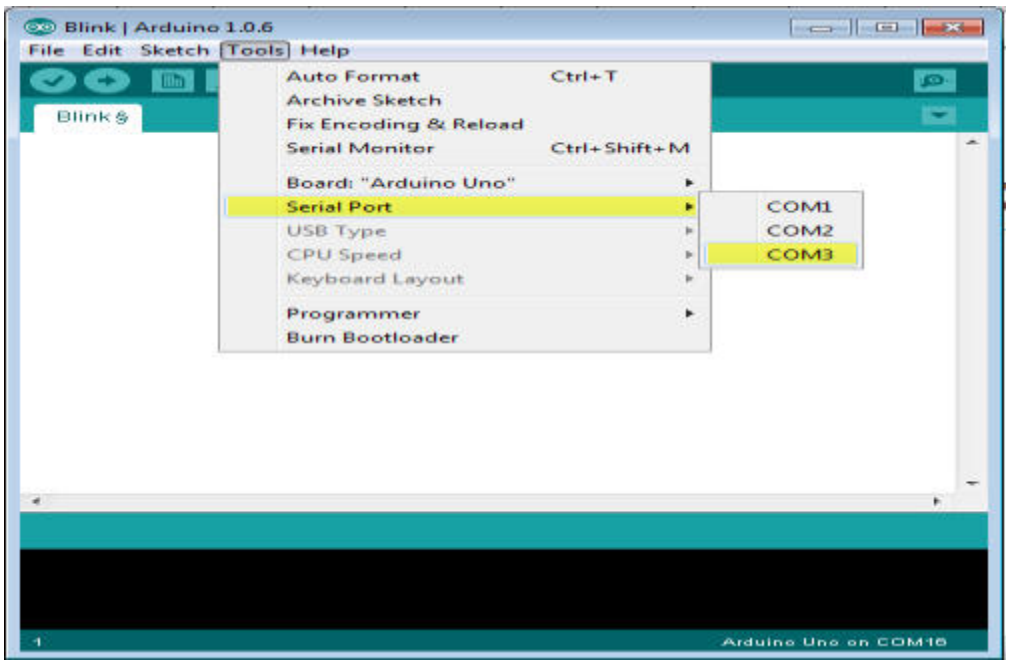
Go to Tools → Board and select your board.



Here, we have selected Arduino Uno board, but you must select the name matching the board that you are using.

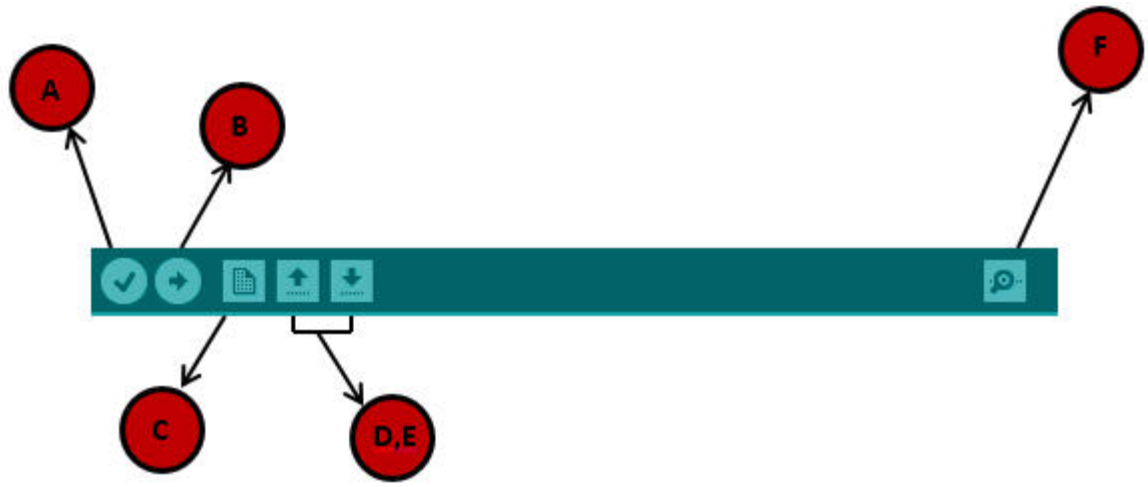
Step 7 − Select your serial port.

Select the serial device of the Arduino board. Go to **Tools → Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 − Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



A − Used to check if there is any compilation error.

B − Used to upload a program to the Arduino board.

C − Shortcut used to create a new sketch.

D − Used to directly open one of the example sketch.

E − Used to save your sketch.

F − Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Readingdata from sensors:

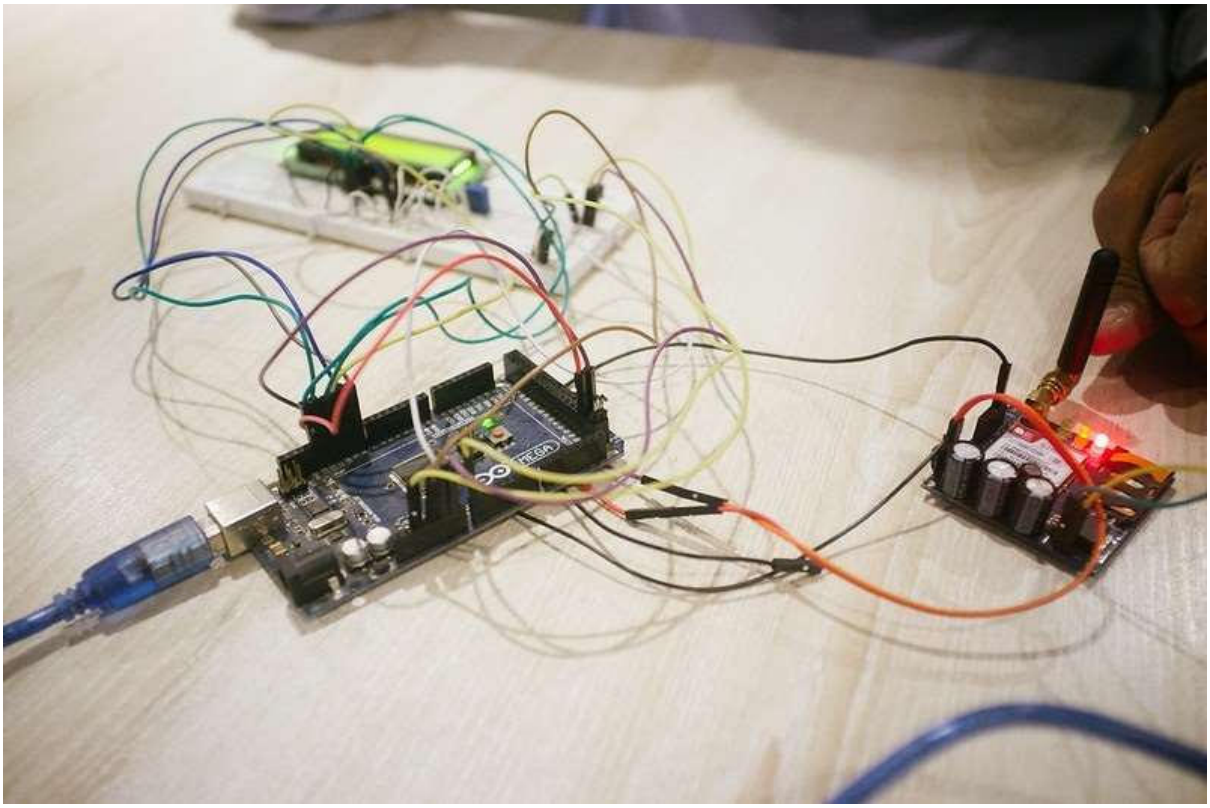
Reading data from a sensor typically involves the following steps:

1. Connect the sensor: The first step is to connect the sensor to the device that will be reading its output. This could be a microcontroller, a single-board computer, or any other device capable of reading analog or digital signals.
2. Configure the device: Once the sensor is connected, the device must be configured to read the sensor's output. This may involve setting up communication protocols, such as I2C or SPI, and configuring the input pins or channels on the device to receive the sensor's output.
3. Read the sensor: Once the device is configured, it can begin reading the sensor's output. Depending on the type of sensor and the device being used, this may involve polling the sensor for data or receiving a continuous stream of data from the sensor.
4. Process the data: Once the data has been read from the sensor, it may need to be processed or analyzed in some way. For example, if the sensor is measuring temperature, the raw data may need to be converted into a meaningful temperature reading using a formula or lookup table.

5. Use the data: Finally, once the data has been processed, it can be used for whatever purpose it was intended. This could be as simple as displaying the data on a screen or as complex as using the data to control a system or make decisions in real-time.

How to read Analog Sensors using Arduino:

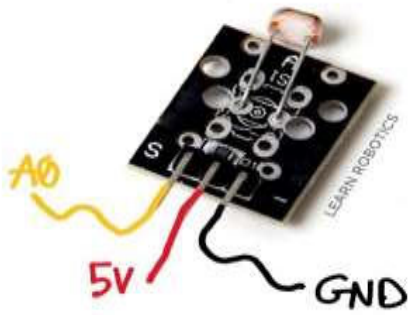
The Arduino has built-in analog and digital input and output (I/O) pins The difference between analog and digital sensors is that an analog sensor collects readings over a range of values, and a digital sensor only reads a HIGH or LOW signal (a bit of data).



The Arduino has a 10-bit Analog-to-Digital-Converter (ADC), which maps sensor readings between 0 and the operating voltage (5V or 3.3V) into integer values between 0 and 1023.

Step 1. Wire Analog Sensors to Arduino

The majority of analog sensors for Arduino are wired and programmed the same. So, once you learn how to wire and read data from one analog sensor, you'll be able to wire and program thousands of additional sensors to collect a whole bunch of data. For this example, I'll walk you through wiring and programming a light-dependent resistor (LDR) also known as a photoresistor.



The first step is to connect the analog sensor to the Arduino. Analog sensors for Arduino have three wires (Ground, Signal, Power). Then, connect the ground wire to GND on the Arduino. Next, attach the Signal wire to an analog pin on the Arduino. Lastly, connect the power wire to the 5V on the Arduino

Step 2. Setup your Arduino Sketch

The next step is to set up the Arduino Sketch. First, configure a global variable for the analog sensor.

```
int ldr = A0;
```

Then, in the setup() method, initialize the sensor as an input and start the Serial monitor.

```
void setup(){
pinMode(ldr, INPUT); //initialize ldr sensor as INPUT
Serial.begin(9600); //begin the serial monitor at 9600 baud
}
```

The Arduino Uno has a baud rate of 9600. We can use the begin method to start the Serial Monitor. Now, we're ready to write the Arduino code to collect readings from our analog sensor.

Step 3. Write code to collect readings from Analog Sensors

Next, collect a sensor reading using the analogRead(ldr) method, and store it in an integer variable. I called this variable "data."

We will use a few print statements to show the readings in the Serial Monitor. Serial.print() will print data horizontally across the screen. Serial.println() will print data vertically down the screen. I used both to label the data while making it easy to read.

```
void loop(){
   int data=analogRead(ldr);
Serial.print("ldr reading=");
Serial.println(data);
delay(1000);
}
```

Finally, add a delay. This prevents the Arduino from taking readings faster than we can see them. Feel free to adjust this delay to whatever interval makes sense for your application. Once

you have the test code written, save the sketch and upload it to the Arduino. Open up the serial monitor and you should see values from 0-1023 depending on how bright or dark the area is.

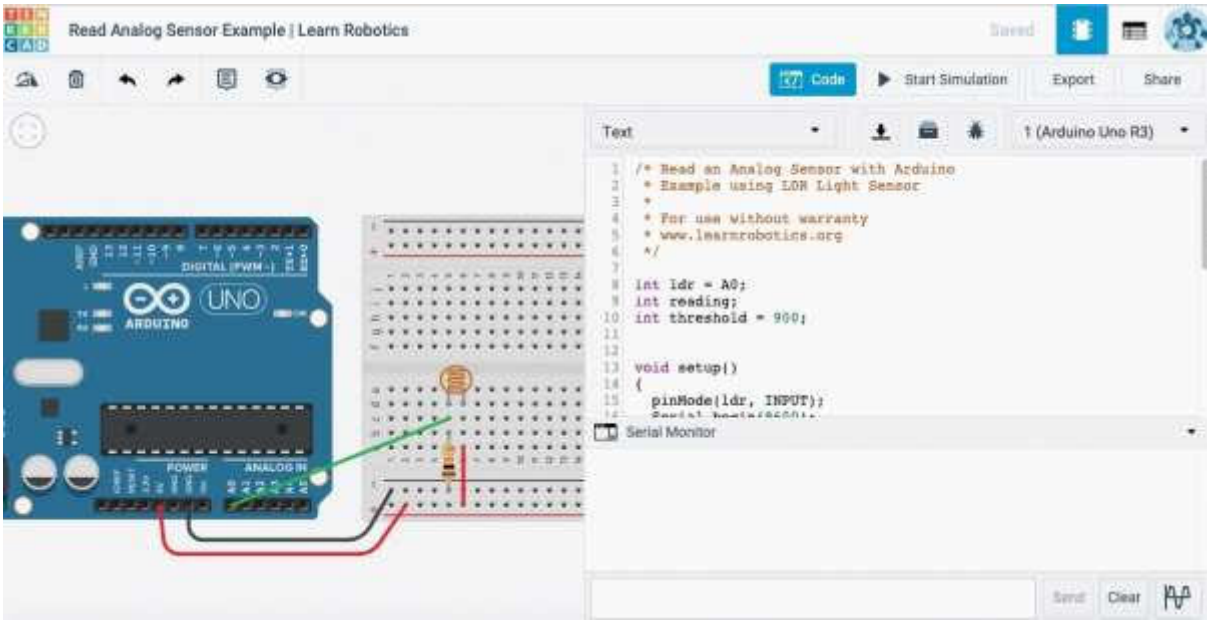
Step 4. Analyze and Convert Sensor Readings as needed

Some sensors require a unit conversion. For example, it's easier to understand what temperature it is when the units are in Celcius or Fahrenheit. Furthermore, with our LDR we could convert the unit readings into a brightness percentage. That way when we analyze the data, we can check for conditions based on 25% bright, 100% bright, or a unit that makes more sense for the application.

Most datasheets specify formulas that you can use to make these conversions. You don't always have to "make up" a unit for your sensor.

Step 5. Use sensor data to make decisions

Once you have an understanding of how data is collected from your analog sensor, you can use the readings to make decisions.



We'll use conditional statements to check to see if a condition is TRUE or FALSE. Then, based on that condition, we'll react accordingly.

Here's an example using the LDR. If the condition is LIGHT, then let's write the word "daylight" to the Serial Monitor. Otherwise, write the word "nighttime" to the Serial Monitor.

First, add a method called lightCheck() to the previous Arduino sketch.

```
//globals to store data
int reading;
int threshold = 900; //range of 0-1023 / higher value = brighter

void lightCheck(){
   reading = analogRead(ldr);
if(reading >= threshold){ //this condition means the readings are light
```

```
Serial.println("daylight");
   }
else{
Serial.println("nighttime");
   }
delay(1000);
}
```

Then, we'll call lightCheck() in our loop() method. Here's how it should work.

Communication through Bluetooth iot:

Bluetooth is a widely used wireless communication technology that is frequently used in Internet of Things (IoT) devices to enable communication between them. Bluetooth enables low-power, short-range wireless communication between devices, making it an ideal choice for many IoT applications.

Bluetooth is often used in IoT devices for various purposes such as transferring data, controlling devices, and enabling communication between devices. For example, a Bluetooth-enabled sensor in a smart home can communicate with a smartphone app to send data on temperature, humidity, and other environmental factors.

One of the main advantages of using Bluetooth in IoT is that it is a low-power technology, which means that it does not consume a lot of battery power. This makes it an ideal choice for IoT devices that are designed to be powered by a battery or other low-power sources. Additionally, Bluetooth is relatively easy to implement and has a high level of compatibility with many devices.

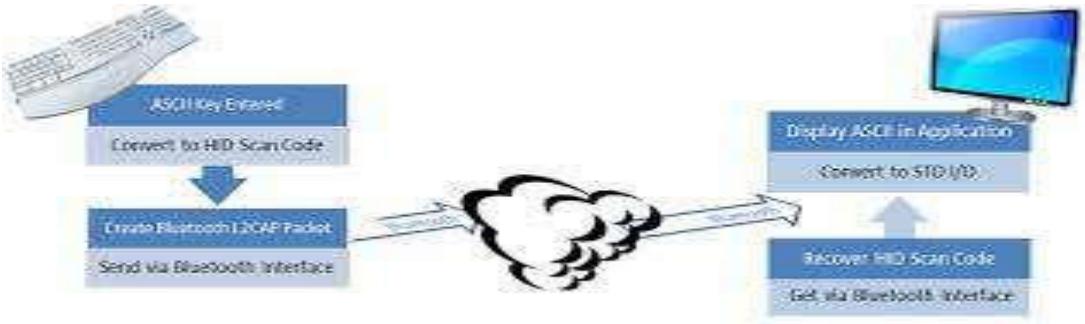
In summary, Bluetooth is an important technology for enabling communication in IoT devices, and its low-power and compatibility features make it a popular choice for many IoT applications

Bluetooth is a short-range wireless communication network over a radio frequency. Bluetooth is mostly integrated into smartphones and mobile devices. The Bluetooth communication network works within 2.4 ISM band frequencies with data rate up to 3Mbps.

There are three categories of Bluetooth technology:

1. Bluetooth Classic
2. Bluetooth Low Energy
3. Bluetooth SmartReady

The features of Bluetooth 5.0 version is introduced as Bluetooth 5 which have been developed entirely for the Internet of Things.



Play Video

Properties of Bluetooth network

- **Standard:** Bluetooth 4.2
- **Frequency:** 2.4GHz
- **Range:** 50-150m
- **Data transfer rates:** 3Mbps

Advantages of Bluetooth network

- It is wireless.
- It is cheap.
- It is easy to install.
- It is free to use if the device is installed with it.

Disadvantages of Bluetooth network

- It is a short-range communication network.
- It connects only two devices at a time.
- Communication through WiFi is a type of wireless communication that enables devices to communicate with each other using radio waves over a local area network (LAN). WiFi (Wireless Fidelity) is a popular wireless networking technology that uses radio waves to provide wireless high-speed Internet and network connections.

- To communicate through WiFi, devices must have a wireless network adapter that can transmit and receive signals over the air. The wireless adapter converts the data into radio waves and sends it to the wireless router or access point, which then sends the data to other devices connected to the same network.

- WiFi communication is commonly used in homes, offices, public places, and other locations where there is a need for wireless connectivity. Some examples of devices that use WiFi communication include smartphones, laptops, tablets, gaming consoles, and smart home devices.

- One of the advantages of WiFi communication is its convenience and flexibility, as it allows devices to connect to the Internet and network without the need for cables or physical connections. However, WiFi communication can be affected by factors such as distance, interference, and network congestion, which can impact its speed and reliability.

communication through wifi:

IoT (Internet of Things) devices often use Wi-Fi to communicate with each other and with the internet. Wi-Fi is a wireless networking technology that uses radio waves to transmit data between devices that are within range of a Wi-Fi network.

To communicate through Wi-Fi in IoT, devices need to have Wi-Fi radios and be connected to a Wi-Fi network. The devices can then send and receive data over the network using various communication protocols such as HTTP, MQTT, CoAP, and others.

One important consideration when using Wi-Fi for IoT communication is security. Wi-Fi networks can be vulnerable to hacking and other security threats, so it's important to implement strong security measures such as encryption, secure authentication, and access controls to protect IoT devices and the data they transmit.

Overall, Wi-Fi is a popular choice for IoT communication due to its widespread availability, relatively low cost, and ease of use. However, other wireless communication technologies such as Bluetooth, Zigbee, and LoRaWAN may also be suitable for specific IoT use cases depending on factors such as range, power consumption, and data transfer speed.

Applications of Wi-Fi :

Wi-Fi has many applications, it is used in all the sectors where a computer or any digital media is used, also for entertaining Wi-Fi is used. Some of the applications are mentioned below –

- Accessing Internet: Using Wi-Fi we can access the internet in any Wi-Fi-capable device wirelessly.

- We can stream or cast audio or video wirelessly on any device using Wi-Fi for our entertainment.
- We can share files, data, etc between two or more computers or mobile phones using Wi-Fi, and the speed of the data transfer rate is also very high. Also, we can print any document using a Wi-Fi printer, this is very much used nowadays.
- We can use Wi-Fi as **HOTSPOTS** also, it points Wireless Internet access for a particular range of area. Using Hotspot the owner of the main network connection can offer temporary network access to Wi-Fi-capable devices so that the users can use the network without knowing anything about the main network connection. Wi-Fi adapters are mainly spreading radio signals using the owner network connection to provide a hotspot.
- Using Wi-Fi or WLAN we can construct simple wireless connections from one point to another, known as Point to point networks. This can be useful to connect two locations that are difficult to reach by wire, such as two buildings of corporate business.
- One more important application is **VoWi-Fi**, which is known as **voice-over Wi-Fi**. Some years ago telecom companies are introduced VoLTE (Voice over Long-Term Evolution). Nowadays they are introduced to VoWi-Fi, by which we can call anyone by using our home Wi-Fi network, only one thing is that the mobile needs to connect with the Wi-Fi. Then the voice is transferred using the Wi-Fi network instead of using the mobile SIM network, so the call quality is very good. Many mobile phones are already getting the support of VoWi-Fi.
- Wi-Fi in offices: In an office, all the computers are interconnected using Wi-Fi. For Wi-Fi, there are no wiring complexities. Also, the speed of the network is good. For Wi-Fi, a project can be presented to all the members at a time in the form of an excel sheet, ppt, etc. For Wi-Fi, there is no network loss as in cable due to cable break.
- Also using W-Fi a whole city can provide network connectivity by deploying routers at a specific area to access the internet. Already schools, colleges, and universities are providing networks using Wi-Fi because of its flexibility.
- Wi-Fi is used as a *positioning system* also, by which we can detect the positions of Wi-Fi hotspots to identify a device location.

Types of Wi-Fi:

Wi-Fi has several types of standards, which are discussed earlier, here just the name of the standards are defined,

Standards	Year of Release	Description
Wi-Fi-1 (802.11b)	**1999**	**This version has a link speed from 2Mb/s to 11 Mb/s over a 2.4 GHz frequency band**
Wi-Fi-2 (802.11a)	**1999**	**After a month of release previous version, 802.11a was released and it provide up to 54 Mb/s link speed over 5 Ghz band**
Wi-Fi-3 (802.11g)	**2003**	**In this version the speed was increased up to 54 to 108 Mb/s over 2.4 GHz**

802.11i	**2004**	**This is the same as 802.11g but only the security mechanism was increased in this version**
802.11e	**2004**	**This is also the same as 802.11g, only Voice over Wireless LAN and multimedia streaming are involved**
Wi-Fi-4 (802.11n)	**2009**	**This version supports both 2.4 GHz and 5 GHz radio frequency and it offers up to 72 to 600 Mb/s speed**
Wi-Fi-5 (802.11ac)	**2014**	**It supports a speed of 1733 Mb/s in the 5 GHz band**

Comparison between WiFi and Bluetooth

The following table highlights the major differences between WiFi and Bluetooth.

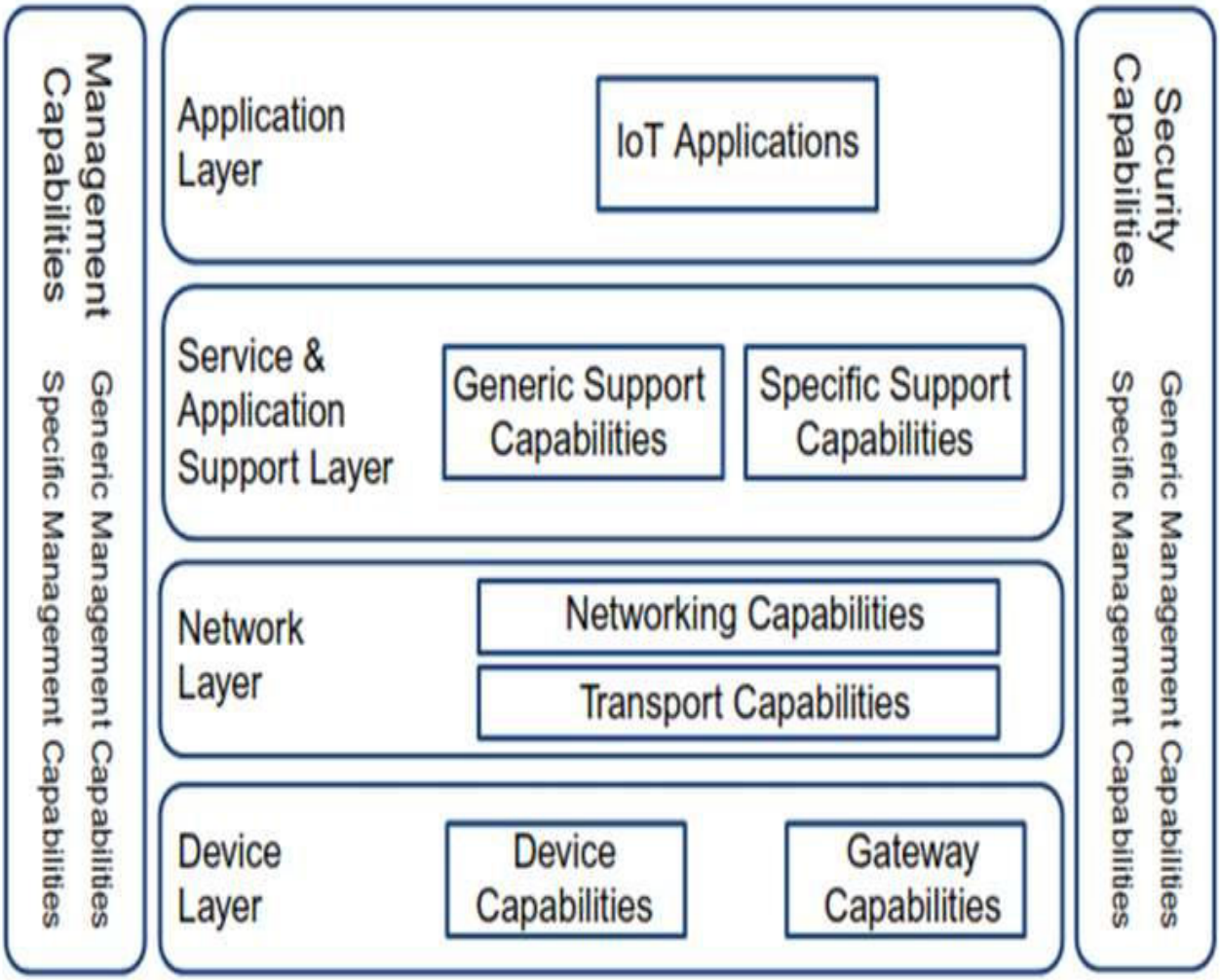
Key	WiFi	Bluetooth
Definition	WiFi stands for Wireless Fidelity. Wi-Fi is a technology that enables devices to connect to the Internet wirelessly.	Bluetooth is a wireless technology that is used to connect devices in short range.
Component	WiFi requires wireless adaptor on all devices and Wireless Router for connectivity.	Bluetooth requires an Bluetooth adaptor on all devices for connectivity.
Power Consumption	WiFi consumes high power.	Bluetooth is easier to use and consumes less power than Wi-Fi because it only requires an adapter on each connecting device.
Security	WiFi is more secure than Bluetooth.	Bluetooth is less secure than other wireless technologies such as WiFi.
Number of Users	Wi-Fi allows more devices and users to communicate at the same time.	Bluetooth restricts the number of devices that can connect at any given moment.
Bandwidth	WiFi needs high bandwidth.	Bluetooth has a low bandwidth.
Coverage	WiFi coverage area is up to 32 meters.	Bluetooth coverage area is about 10 meters.

Reference Books:

1. Jan Holler, VlasiosTsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, ― From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence‖, 1st Edition, Academic Press, 2014.
2. ArshdeepBahga, Vijay Madisetti - Internet of Things: A Hands-On Approach, Universities Press, 2014.
3. The Internet of Things, Enabling technologies and use cases – Pethuru Raj, Anupama C. Raman, CRC Press.
4. Francis daCosta, ―Rethinking the Internet of Things: A Scalable Approach to Connecting Everything‖, 1st Edition, Apress Publications, 2013
5. Cuno Pfister, Getting Started with the Internet of Things, O‟Reilly Media, 2011, ISBN: 9781- 4493- 9357-1
6. DGCA RPAS Guidance Manual, Revision 3 – 2020
7. Building Your Own Drones: A Beginners' Guide to Drones, UAVs, and ROVs, John Baic

Unit-3 IOT ARCHITECTURE&PROTOCOLS

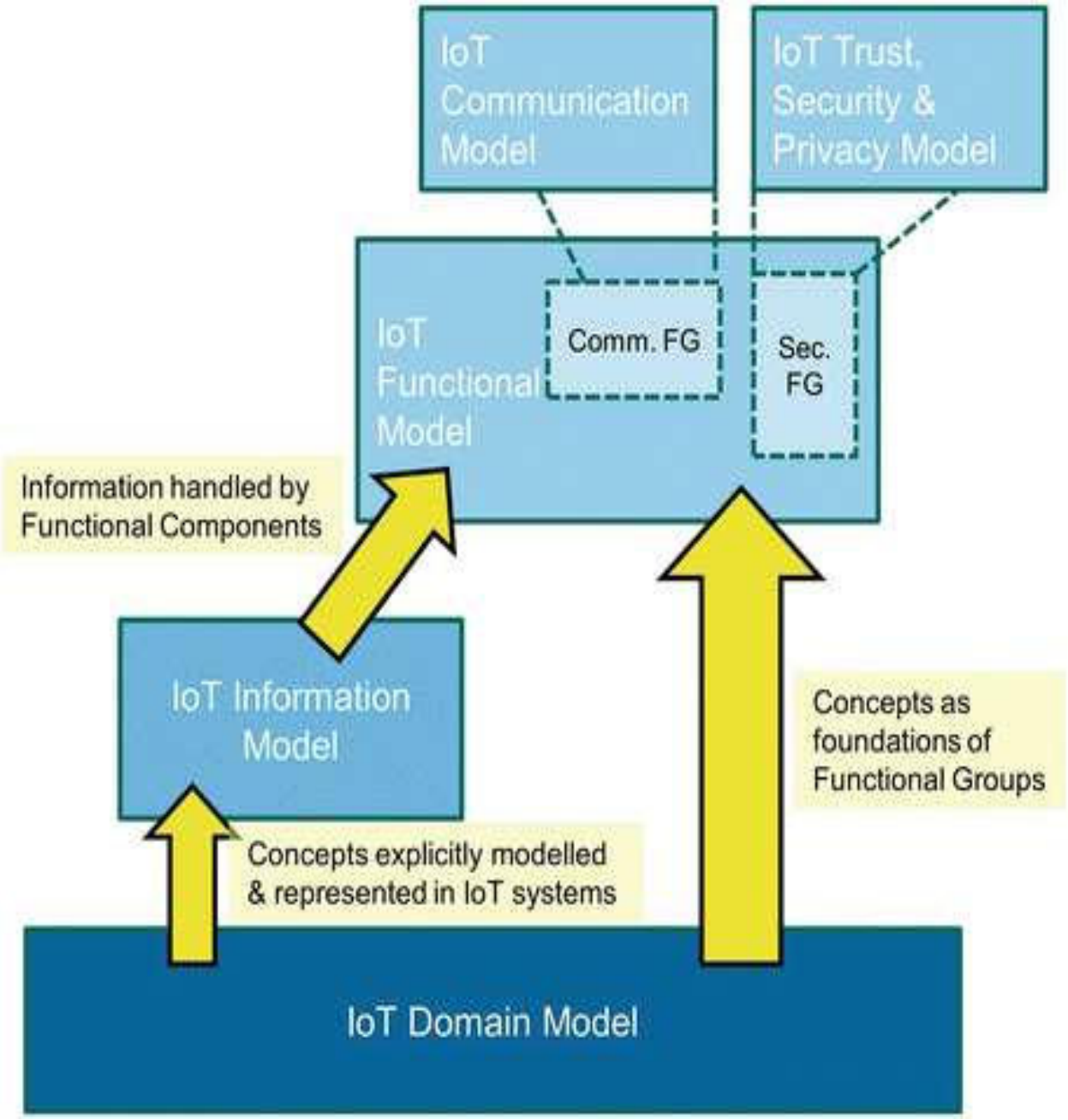
Architecture Reference Model

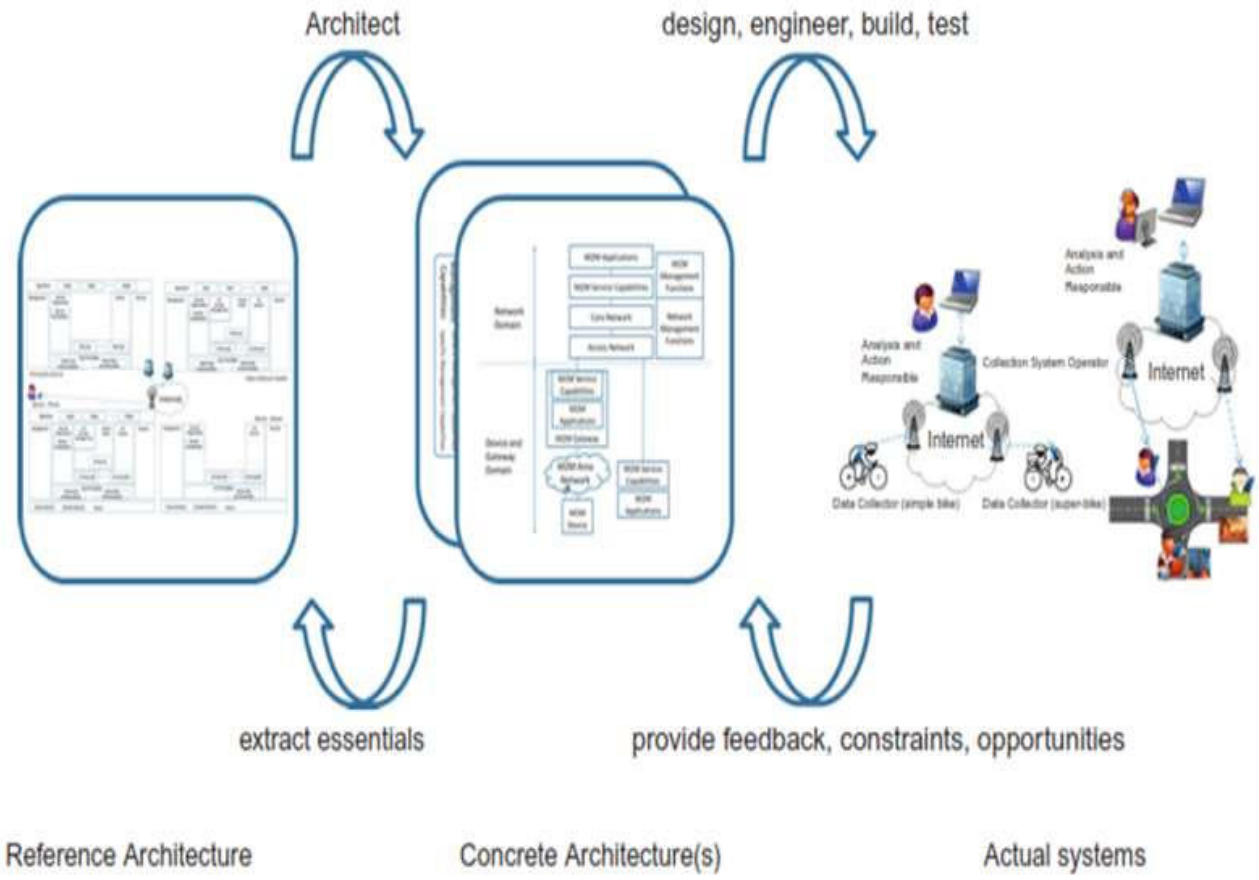


Reference Model and Architecture

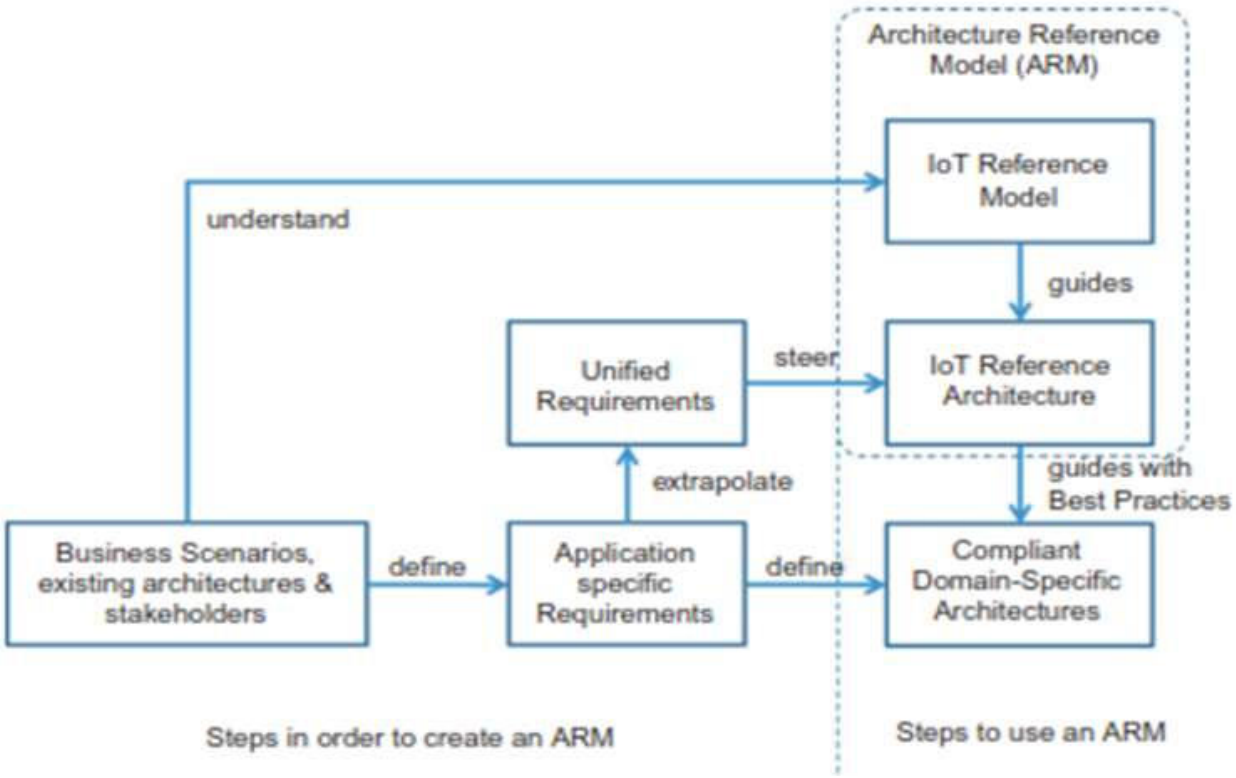
•An ARM consists of two main parts: a Reference model and a Reference Architecture.

•A reference model describes the domain using a number of sub-models


IoT Communication Model
IoT Trust, Security & Privacy Model
IoT Functional Model
Comm. FG
Sec. FG
Information handled by Functional Components
IoT Information Model
Concepts as foundations of Functional Groups
Concepts explicitly modelled & represented in IoT systems
IoT Domain Model



From Reference to concrete architecture and actual system



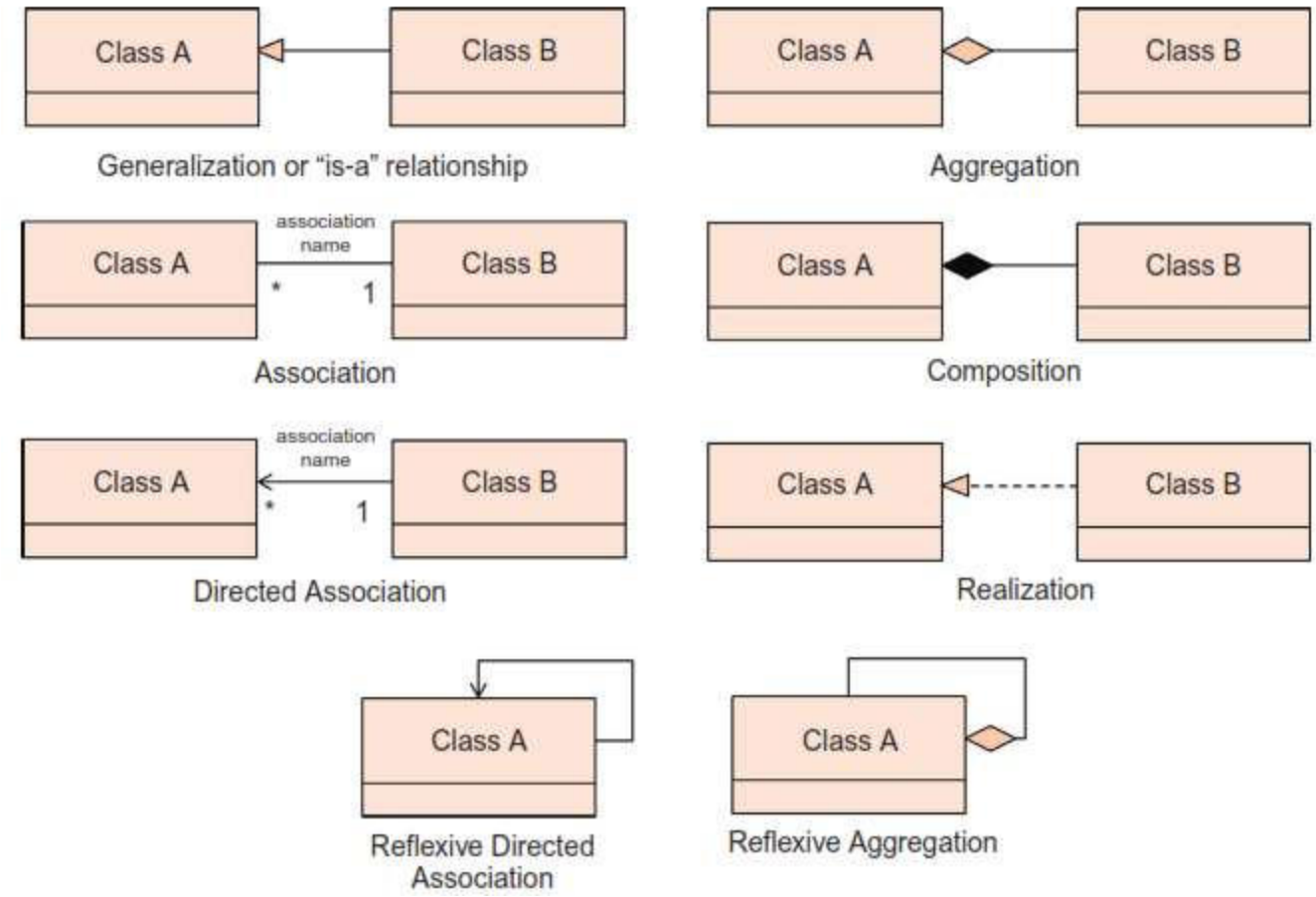
IOT Reference architecture and reference model dependency

IOT Reference Model

IoT domain model

The domain model captures the basic attributes of the main concepts and the relationship between these concepts. A domain model also serves as a tool for human communication between people working in the domain in question and between people who work across different domains.

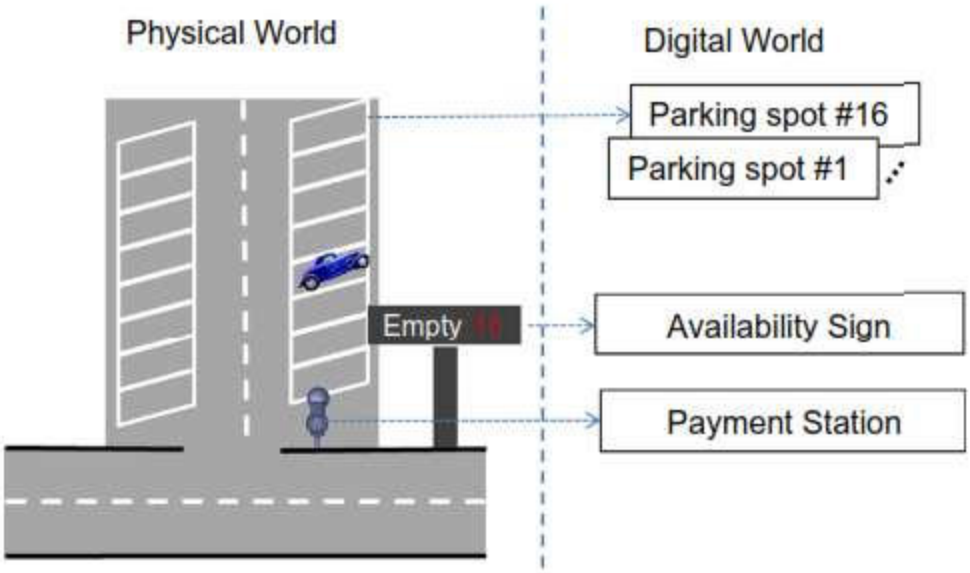
Model notation and semantics



UML Class diagram main modelling concepts

Main concepts

The IoT is a support infrastructure for enabling objects and places in the physical world to have a corresponding representation in the digital world.



Physical vs Virtual World

•The Devices are physical artefacts with which the physical and virtual worlds interact. Devices as mentioned before can also be Physical Entities for certain types of applications, such as management applications when the interesting entities of a system are the Devices themselves and not the surrounding environment. For the IoT Domain Model, three kinds of Device types are the most important:

1. Sensors:

- These are simple or complex Devices that typically involve a transducer that converts physical properties such as temperature into electrical signals.
- These Devices include the necessary conversion of analog electrical signals into digital signals, e.g. a voltage level to a 16-bit number, processing for simple calculations, potential storage for intermediate results, and potentially communication capabilities to transmit the digital representation of the physical property as well receive commands.
- A video camera can be another example of a complex sensor that could detect and recognise people.

2. Actuators:

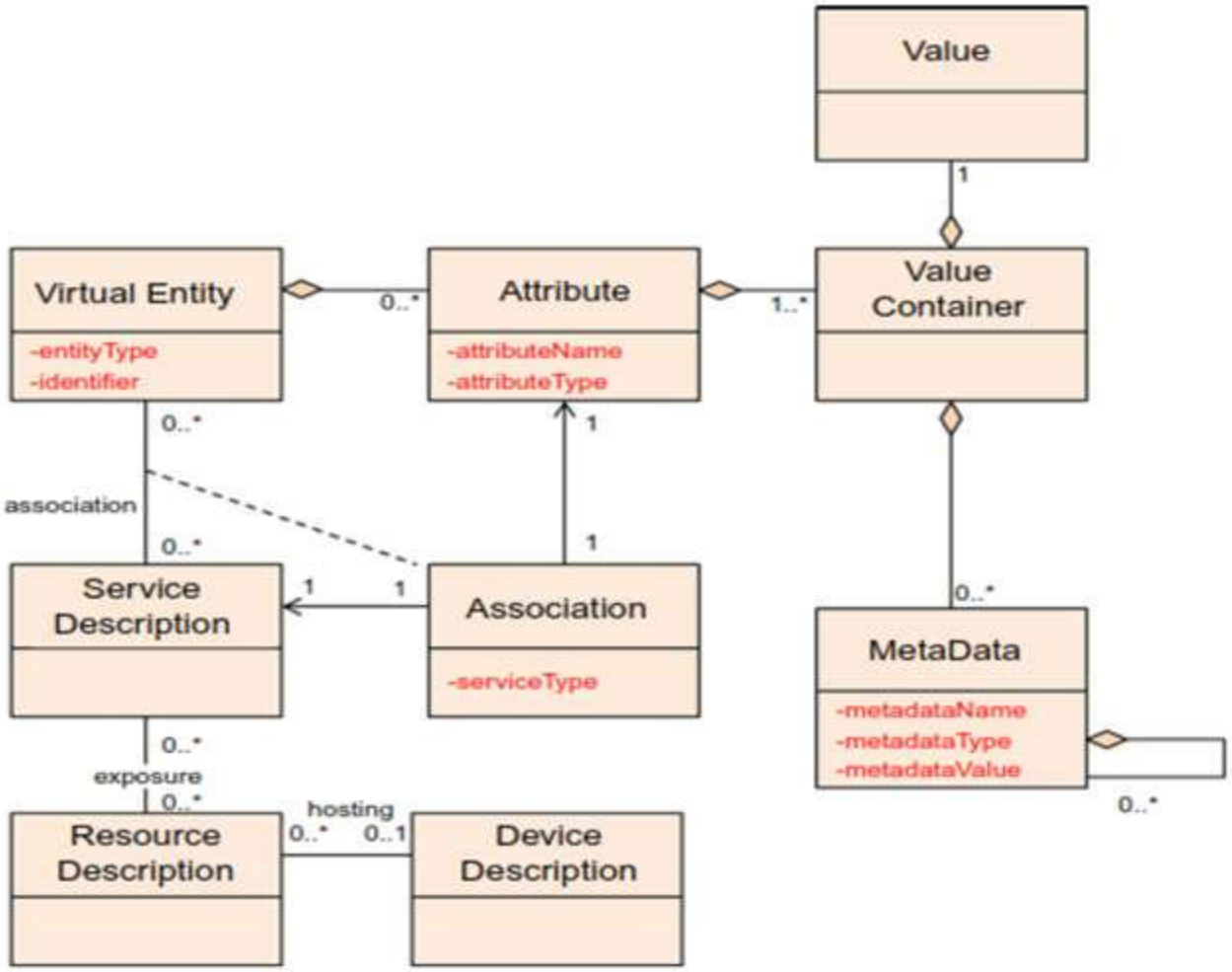
- These are also simple or complex Devices that involve a transducer that converts electrical signals to a change in a physical property (e.g. turn on a switch or move a motor).
- These Devices also include potential communication capabilities, storage of intermediate commands, processing, and conversion of digital signals to analog electrical signals.

3. Tags:

- Tags in general identify the Physical Entity that they are attached to. In reality, tags can be Devices or Physical Entities but not both, as the domain model shows.
- An example of a Tag as a Device is a Radio Frequency Identification (RFID) tag, while a tag as a Physical Entity is a paper-printed immutable barcode or Quick Response (QR) code.
- Either electronic Devices or a paper-printed entity tag contains a unique identification that can be read by optical means (bar codes or QR codes) or radio signals (RFID tags).
- The reader Device operating on a tag is typically a sensor, and sometimes a sensor and an actuator combined in the case of writable RFID tags.

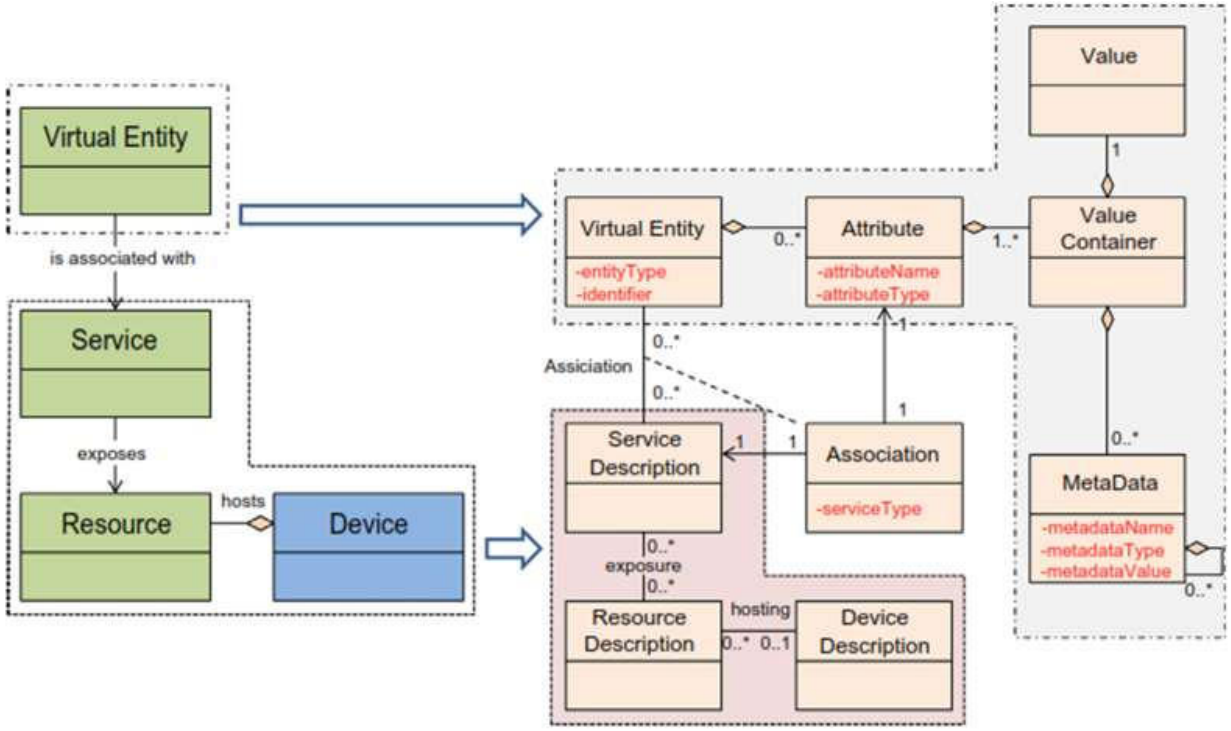
Information Model

Virtual Entity in the IoT Domain Model is the “Thing” in the Internet of Things, the IoT information model captures the details of a Virtual Entity- centric model. Similar to the IoT Domain Model, the IoT Information Model is presented using Unified Modelling Language (UML) diagrams.



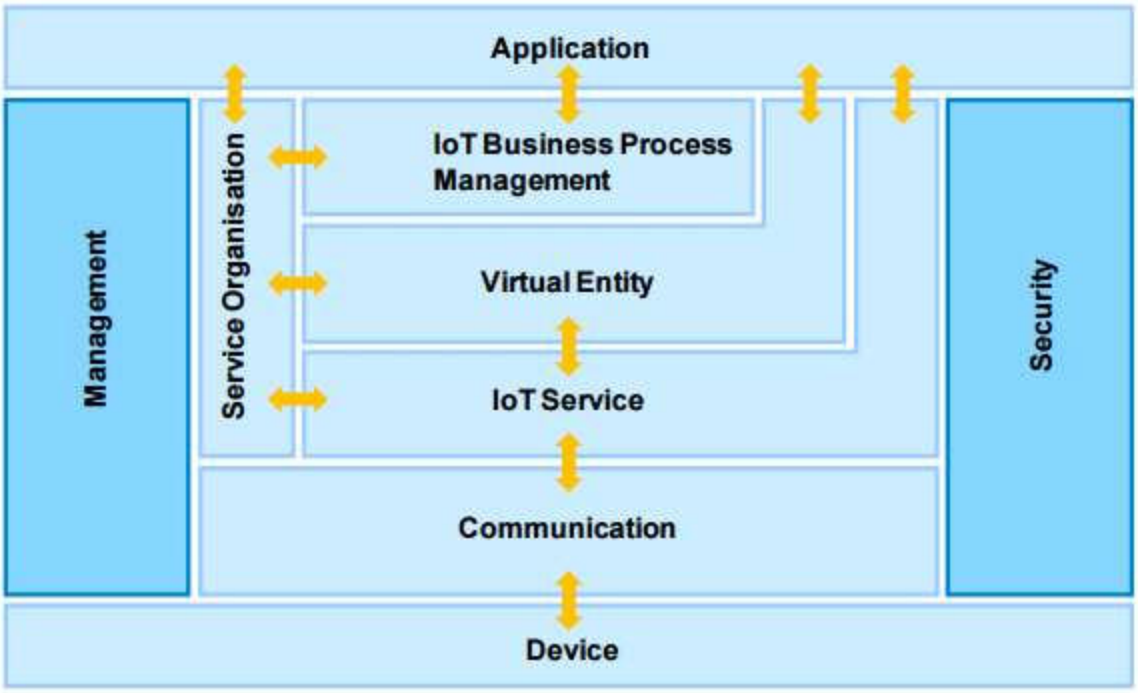
High-level IoT Information Model

Relationship between core concepts of IoT Domain Model and IoT Information Model.



Functional model

The IoT Functional Model aims at describing mainly the Functional Groups (FG) and their interaction with the ARM, while the Functional View of a Reference Architecture describes the functional components of an FG, interfaces, and interactions between the components. The Functional View is typically derived from the Functional Model in conjunction with high-level requirements.



Device functional group

The Device FG contains all the possible functionality hosted by the physical Devices that are used for increment the Physical Entities. This Device functionality includes sensing, actuation, processing, storage, and identification components, the sophistication of which depends on the Device capabilities

Communication functional group

The Communication FG abstracts all the possible communication mechanisms used by the relevant Devices in an actual system in order to transfer information to the digital world components or other Devices.

IoT Service functional group

The IoT Service FG corresponds mainly to the Service class from the IoT Domain Model, and contains single IoT Services exposed by Resources hosted on Devices or in the Network (e.g. processing or storage Resources).

Virtual Entity functional group

The Virtual Entity FG corresponds to the Virtual Entity class in the IoT Domain Model, and contains the necessary functionality to manage associations between Virtual

Entities with themselves as well as associations between Virtual Entities and related IoT Services, i.e. the Association objects for the IoT Information Model. Associations between Virtual Entities can be static or dynamic depending on the mobility of the Physical Entities related to the corresponding Virtual Entities.

IoT Service Organization functional group

The purpose of the IoT Service Organisation FG is to host all functional components that support the composition and orchestration of IoT and Virtual Entity services. Moreover, this FG acts as a service hub between several other functional groups such as the IoT Process Management FG when, for example, service requests from Applications or the IoT Process Management are directed to the Resources implementing the necessary Services.

IoT Process Management functional group

The IoT Process Management FG is a collection of functionalities that allows smooth integration of IoT-related services (IoT Services, Virtual Entity Services, Composed Services) with the Enterprise (Business) Processes.

Management functional group

The Management FG includes the necessary functions for enabling fault and performance monitoring of the system, configuration for enabling the system to be flexible to changing User demands, and accounting for enabling subsequent billing for the usage of the system. Support functions such as management of ownership, administrative domain, rules and rights of functional components, and information stores are also included in the Management FG.

Security functional group

The Security FG contains the functional components that ensure the secure operation of the system as well as the management of privacy. The Security FG contains components for Authentication of Users (Applications, Humans), Authorisation of access to Services by Users, secure communication (ensuring integrity and confidentiality of messages) between entities of the system such as Devices, Services, Applications, and last but not least, assurance of privacy of sensitive information relating to Human Users.

Application functional group

The Application FG is just a placeholder that represents all the needed logic for creating an IoT application. The applications typically contain custom logic tailored to a specific domain such as a Smart Grid

Communication model

Safety

the IoT Reference Model can only provide IoT-related guidelines for ensuring a safe system to the extent possible and controllable by a sys- tem designer.
Eg: smart grid.

Privacy

Because interactions with the physical world may often include humans, protecting the User privacy is of utmost importance for an IoT system. The IoT-A Privacy Model depends on the following functional components: Identity Management, Authentication, Authorisation, and Trust & Reputation

Trust

Generally, an entity is said to ‘trust’ a second entity when the first entity makes the assumption that the second entity will behave exactly as the first entity expects.”

Security

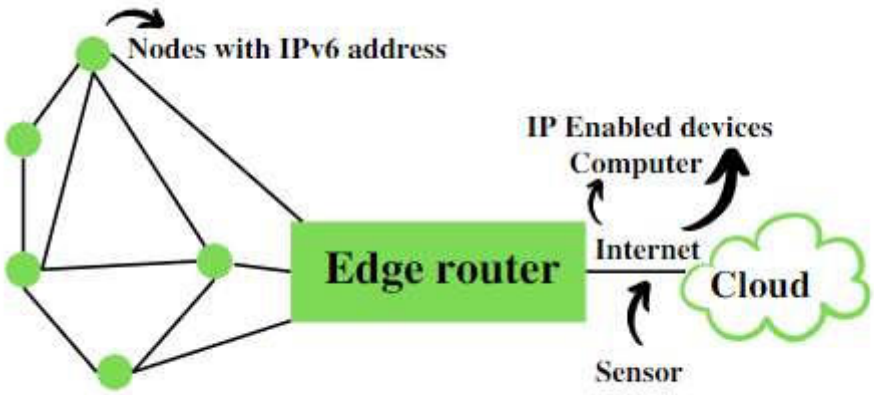
The Security Model for IoT consists of communication security that focuses mostly on the confidentiality and integrity protection of interacting entities and functional components such as Identity Management, Authentication, Authorisation, and Trust & Reputation.

PROTOCOLS:

6LoWPAN

6LoWPAN is an IPv6 protocol, and It's extended from is IPv6 over Low Power Personal Area Network. As the name itself explains the meaning of this protocol is that this protocol works on Wireless Personal Area Network i.e., WPAN.
WPAN is a Personal Area Network (PAN) where the interconnected devices are centered around a person's workspace and connected through a wireless medium. You can read more about WPAN at WPAN. 6LoWPAN allows communication using the IPv6 protocol. IPv6 is Internet Protocol Version 6 is a network layer protocol that allows communication to take place over the network. It is faster and more reliable and provides a large number of addresses.
6LoWPAN initially came into existence to overcome the conventional methodologies that were adapted to transmit information. But still, it is not so efficient as it only allows for the smaller devices with very limited processing ability to establish communication using one of the Internet Protocols, i.e., IPv6. It has very low cost, short-range, low memory usage, and low bit rate.

It comprises an Edge Router and Sensor Nodes. Even the smallest of the IoT devices can now be part of the network, and the information can be transmitted to the outside world as well. For example, LED Streetlights.



- It is a technology that makes the individual nodes IP enabled.
- 6LoWPAN can interact with 802.15.4 devices and also other types of devices on an IP Network. For example, Wi-Fi.
- It uses AES 128 link layer security, which AES is a block cipher having key size of 128/192/256 bits and encrypts data in blocks of 128 bits each. This is defined in IEEE 802.15.4 and provides link authentication and encryption.

Basic Requirements of 6LoWPAN:

1. The device should be having sleep mode in order to support the battery saving.
2. Minimal memory requirement.
3. Routing overhead should be lowered.

Features of 6LoWPAN:

1. It is used with IEEE 802.15,.4 in the 2.4 GHz band.
2. Outdoor range: ~200 m (maximum)
3. Data rate: 200kbps (maximum)
4. Maximum number of nodes: ~100

Advantages of 6LoWPAN:

1. 6LoWPAN is a mesh network that is robust, scalable, and can heal on its own.
1. It delivers low-cost and secure communication in IoT devices.
2. It uses IPv6 protocol and so it can be directly routed to cloud platforms.
3. It offers one-to-many and many-to-one routing.
4. In the network, leaf nodes can be in sleep mode for a longer duration of time.

Disadvantages of 6LoWPAN:

1. It is comparatively less secure than Zigbee.
2. It has lesser immunity to interference than that Wi-Fi and Bluetooth.
3. Without the mesh topology, it supports a short range.

Applications of 6LoWPAN:

1. It is a wireless sensor network.
2. It is used in home-automation,
3. It is used in smart agricultural techniques, and industrial monitoring.

Security and Interoperability with 6LoWPAN:

- **Security**: 6LoWPAN security is ensured by the AES algorithm, which is a link layer security, and the transport layer security mechanisms are included as well.

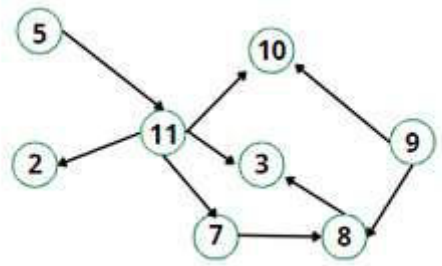
- **Interoperability:** 6LoWPAN is able to operate with other wireless devices as well which makes it interoperable in a network.

RPL (IPv6 Routing protocol):

RPL stands for **Routing Protocol for Low Power and Lossy Networks** for heterogeneous traffic networks. It is a routing protocol for Wireless Networks. This protocol is based on the same standard as by Zigbee and 6 Lowpan is IEEE 802.15.4 It holds both many-to-one and one-to-one communication.

It is a **Distance Vector Routing Protocol** that creates a tree-like routing topology called the *Destination Oriented Directed Acyclic Graph (DODAG)*, rooted towards one or more nodes called the root node or sink node.

The *Directed Acyclic Graphs* (DAGs) are created based on user-specified specific Objective Function (OF). The OF defines the method to find the best-optimized route among the number of sensor devices.



The IETF chartered the ROLL (Routing Over Low power and Lossy networks) working group to evaluate all three routing protocols and determine the needs and requirements for developing a routing solution for IP smart objects. After the study of various use cases and a survey of existing protocols, the consensus was that a new routing protocol should be developed for IP smart objects, given the characteristics and requirements of the constrained network. This new Distance Vector Routing Protocol was named the IPv6 Routing Protocol for Low power and Lossy networks(RPL). The RPL specification was published as RFC 6550 by the ROLL working group.

In an RPL Network, each node acts as a router and becomes part of a mesh network. Routing is performed at the IP Layer. Each node examines every received IPv6 packet and determines the next-hop destination based on the information contained in the IPv6 header. No information from the MAC layer header is needed to perform the next determination.

Modes of RPL:

This protocol defines two modes:
1. Storing mode: All modes contain the entire routing table of the RPL domain. Every node knows how to reach every other node directly.
2. Non-Storing mode: Only the border router(s) of the RPL domain contain(s) the full routing table. All other nodes in the domain maintain their list of parents only and use this as a list of default routes towards the border router. The abbreviated routing table saves memory space and CPU. When communicating in non-storing mode, a node always forwards its packet to the border router, which knows how to ultimately reach the final destination.
RPL is based on the concept of a Directed Acyclic Graph (DAG). A DAG is Directed Graph where no cycle exists. This means that from any vertex or point in the graph,

we cannot follow an edge or a line back to this same point. All of the edges are arranged in a path oriented toward and terminating at one or more root nodes.

A basic RPL process involves building a Destination Oriented Directed Acyclic Graph (DODAG). A DODAG is a DAG rooted in one destination. In RPL this destination occurs at a border router known as the DODAG root. In a DODAG, three parents maximum are maintained by each node that provides a path to the root. Typically one of these parents is the preferred parent, which means it is the preferred next hop for upward roots towards the root. The routing graph created by the set of DODAG parents across all nodes defines the full set of upwards roots. RPL protocol information should ensure that routes are loop-free by disallowing nodes from selected DODAG parents positioned further away from a border router.

Implementation of RPL Protocol:

The RPL protocol is using the Contiki Operating system. This Operating System majorly focuses on IoT devices, more specifically Low Power Wireless IoT devices. It is an Open source Model and was first bought into the picture by Adam Dunkels.

The RPL protocol mostly occurs in wireless sensors and networks. Other similar Operating Systems include T-Kernel, EyeOS, LiteOS, etc.

CoAP Protocol:

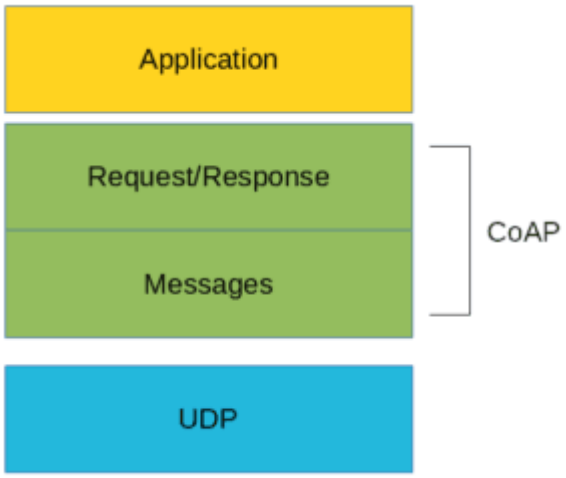
As said before CoAP is an IoT protocol. CoAP stands for Constrained Application Protocol and it is defined in RFC 7252. CoAP is a simple protocol with low overhead specifically designed for constrained devices (such as microcontrollers) and constrained networks. This protocol is used in M2M data exchange and it is very similar to HTTP even if there are important differences that we will cover laters.

The main features of CoAP protocols are:

- Web protocol used in M2M with constrained requirements
- Asynchronous message exchange
- Low overhead and very simple to parse
- URI and content-type support
- Proxy and caching capabilities

As you may notice, some features are very similar to HTTP even if CoAP must not be considered a compressed HTTP protocol because CoAP is specifically designed for IoT and in more details for M2M so it is very optimized for this task.

From the abstraction protocol layer, CoAP can be represented as:



As you can see there are two different layers that make CoAp protocol: Messages and Request/Response. The Messages layer deals with UDP and with asynchronous messages. The Request/Response layer manages request/response interaction based on request/response messages.

CoAP supports four different message types:

- Confirmable
- Non-confirmable
- Acknowledgment
- Reset

Before going deeper into the CoAp protocol structure is useful to define some terms that we will use later:

Endpoint: An entity that participates in the CoAP protocol. Usually, an Endpoint is identified with a host

Sender: The entity that sends a message

Recipient: The destination of a message

Client: The entity that sends a request and the destination of the response

Server: The entity that receives a request from a client and sends back a response to the client

CoAP Messages Model

This is the lowest layer of CoAP. This layer deals with UDP exchanging messages between endpoints. Each CoAP message has a unique id, this is useful to detect message duplicates. A CoAP message is built by these parts:

- a binary header
- a compact options
- payload

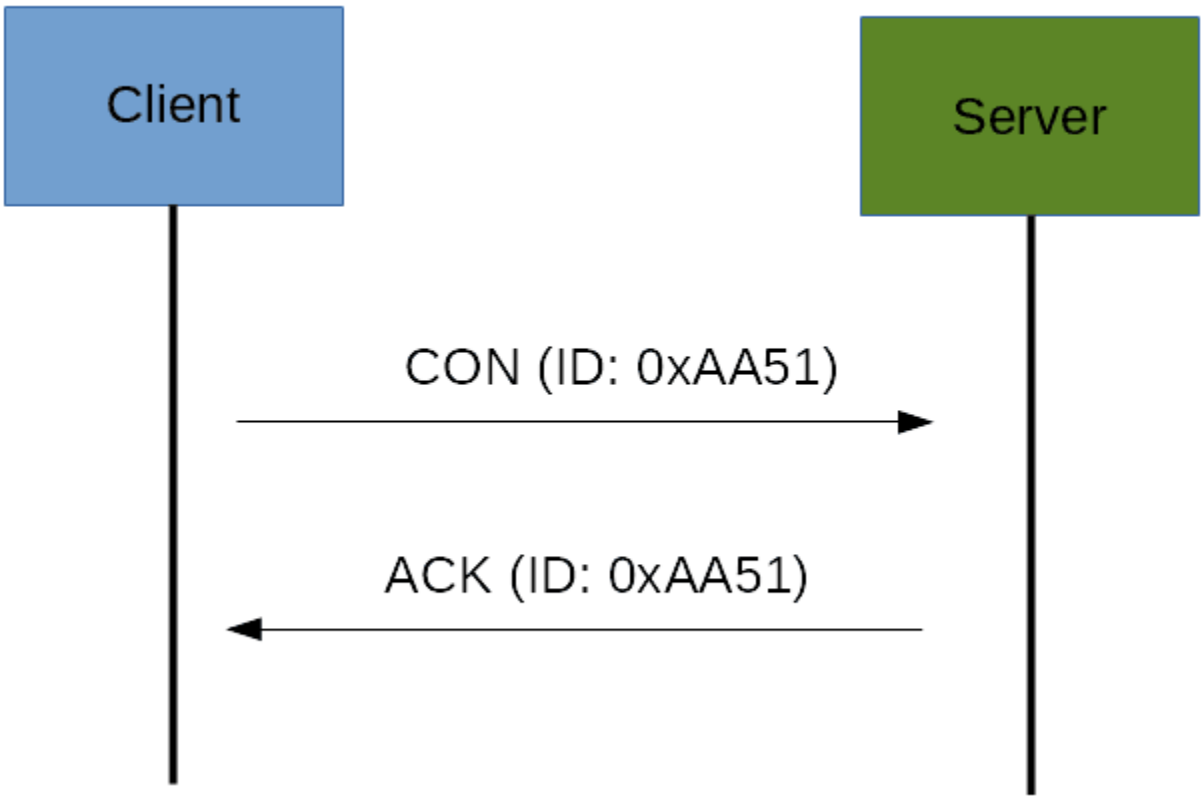
Later, we will describe the message format in more details.

As said before, the CoAP protocol uses two kind of messages:

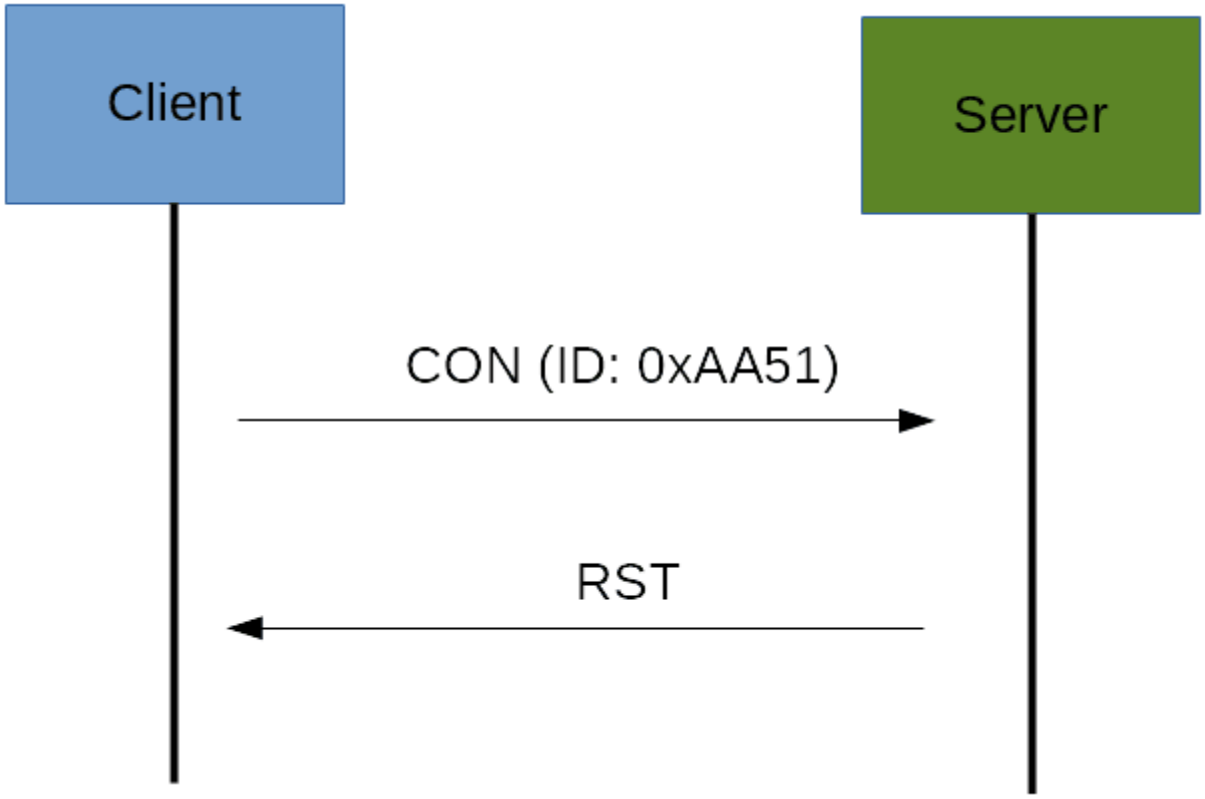
- Confirmable message
- Non-confirmable message

A confirmable message is a reliable message. When exchanging messages between two endpoints, these messages can be reliable. In CoAP a reliable message is obtained using a Confirmable message (CON). Using this kind of message, the client can be sure that the message will arrive at the server. A Confirmable message is sent again and again until the other party sends an acknowledge message (ACK). The ACK message contains the same ID of the confirmable message (CON).

The picture below shows the message exchange process:

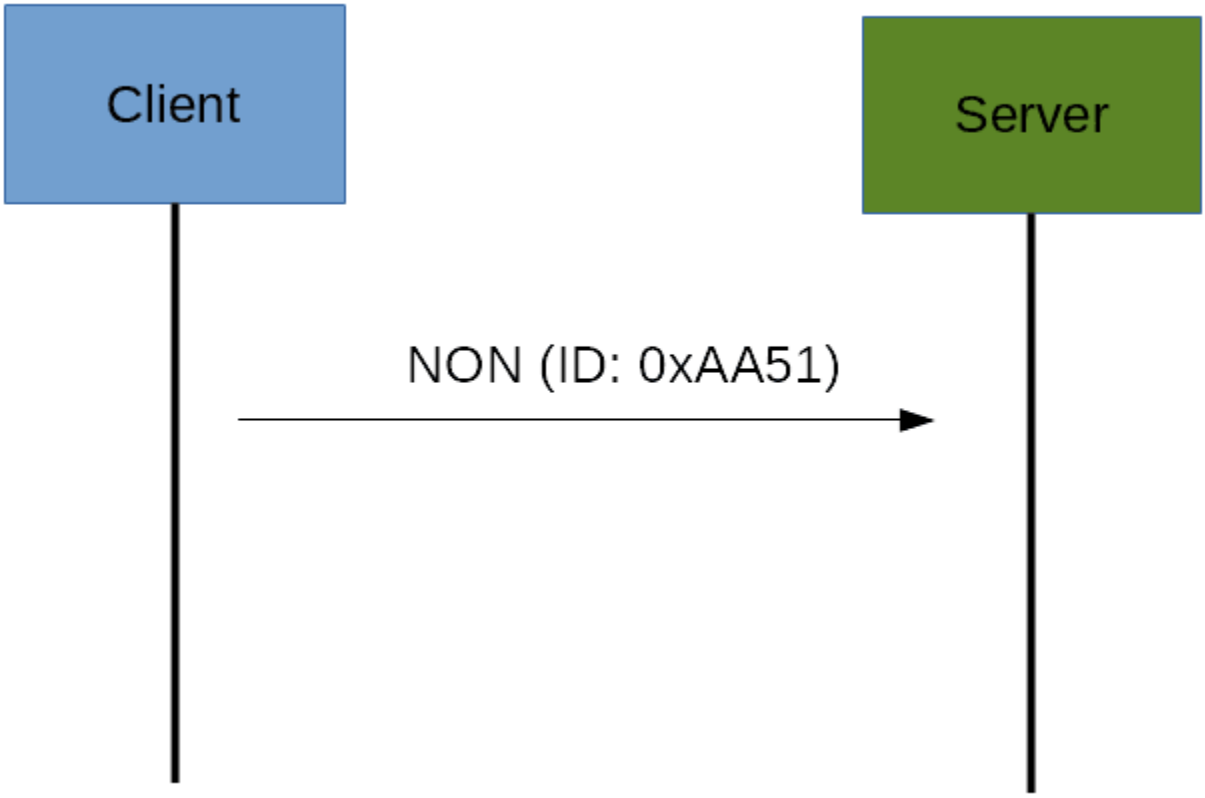


If the server has troubles managing the incoming request it can send back a Rest message (RST) instead of the Acknowledge message (ACK):



The other message category is the Non-confirmable (NON) messages. These are messages that don't require an Acknowledge by the server. They are unreliable messages or in other words messages that do not contain critical information that must be delivered to the server. To this category belongs messages that contain values read from sensors.

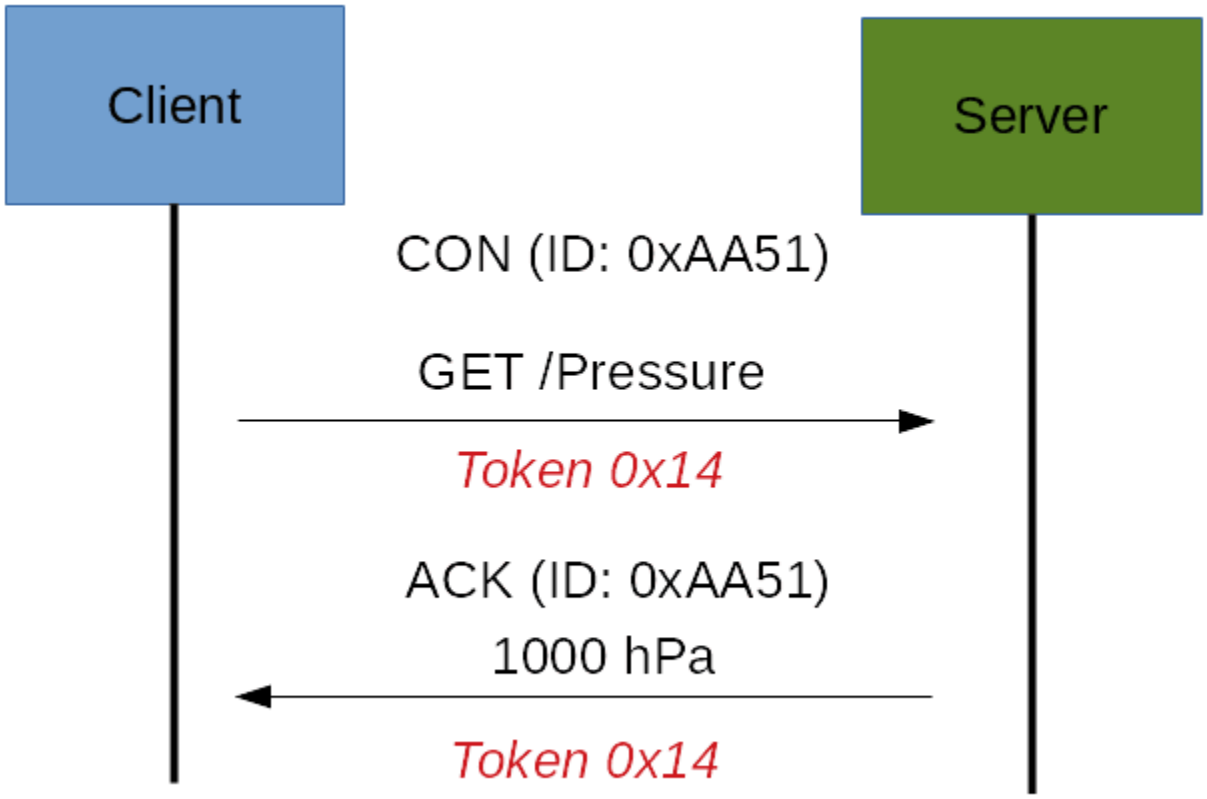
Even if these messages are unreliable, they have a unique ID.



CoAp Request/Response Model

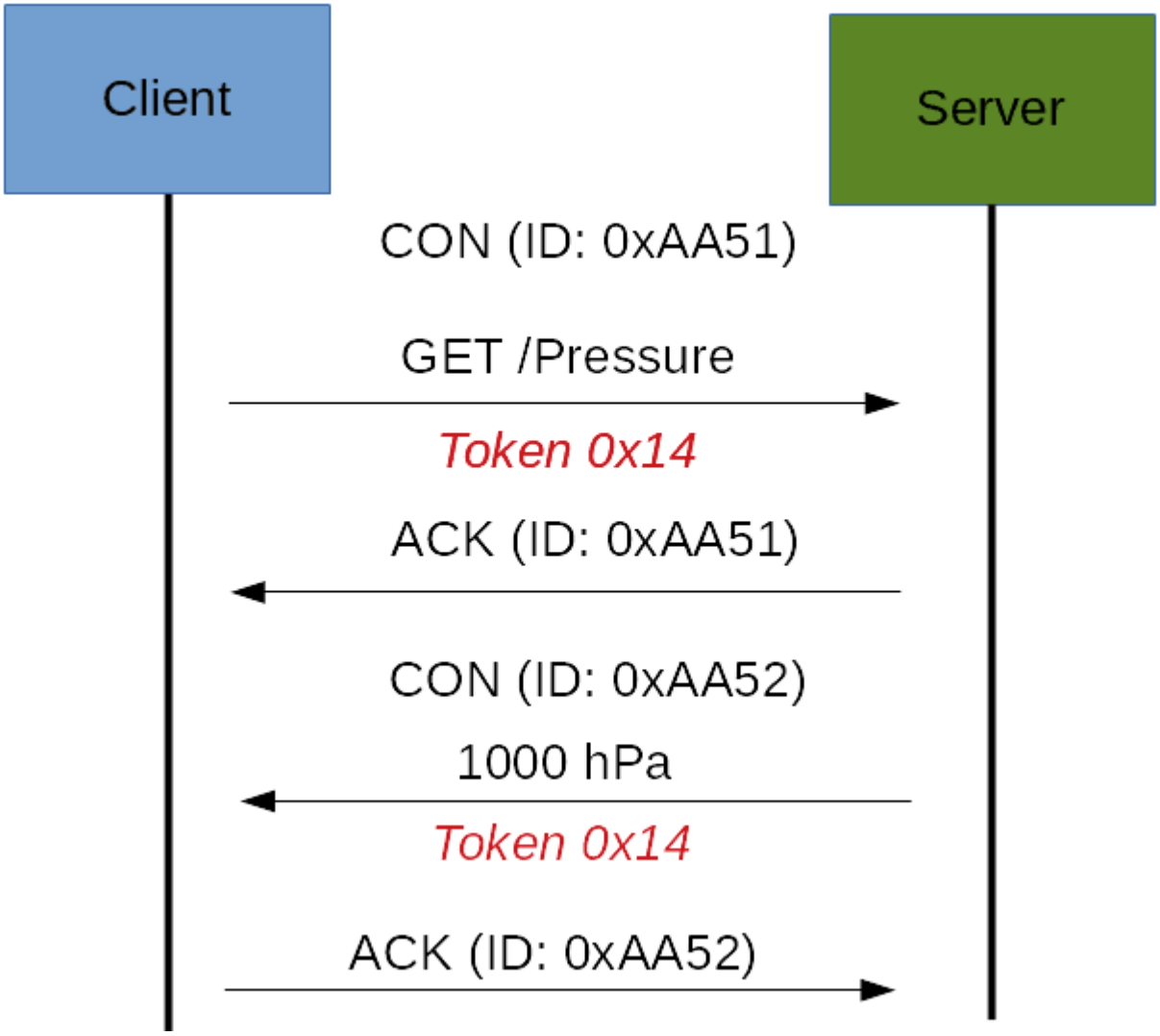
The CoAP Request/Response is the second layer in the CoAP abstraction layer. The request is sent using a Confirmable (CON) or Non-Confirmable (NON) message. There are several scenarios depending on if the server can answer immediately to the client request or the answer if not available:

If the server can answer immediately to the client request then if the request is carried using a Confirmable message (CON) then the server sends back to the client an Acknowledge message containing the response or the error code:



As you can notice in the CoAP message there is a Token. The Token is different from the Message ID and it is used to match the request and the response.

If the server can't answer to the request coming from the client immediately, then it sends an Acknowledge message with an empty response. As soon as the response is available then the server sends a new Confirmable message to the client containing the response. At this point the client sends back an Acknowledge message:

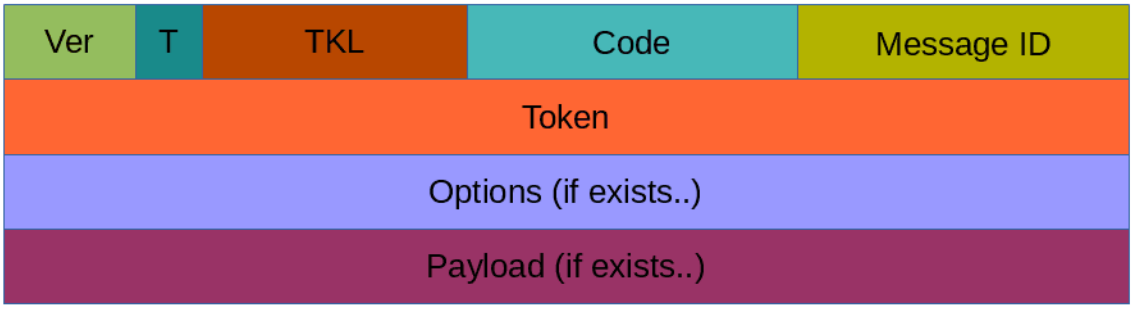


If the request coming from the client is carried using a NON-confirmable message then the server answer using a NON-confirmable message.

CoAp Message Format

This paragraph covers the CoAP Message format. By now we have discussed different kinds of messages exchanged between the client and the server, now it is time to analyze the message format. The constrained application protocol is meat for constrained environments and for this reason, it uses compact messages. To avoid

fragmentation, a message occupies the data section of a UDP datagram. A message is made by several parts:



Where:

Ver: It is a 2 bit unsigned integer indicating the version

T: it is a 2 bit unsigned integer indicating the message type: 0 confirmable, 1 non-confirmable

TKL: Token Length is the token 4 bit length

Code: It is the code response (8 bit length)

Message ID: It is the message ID expressed with 16 bit

and so on.

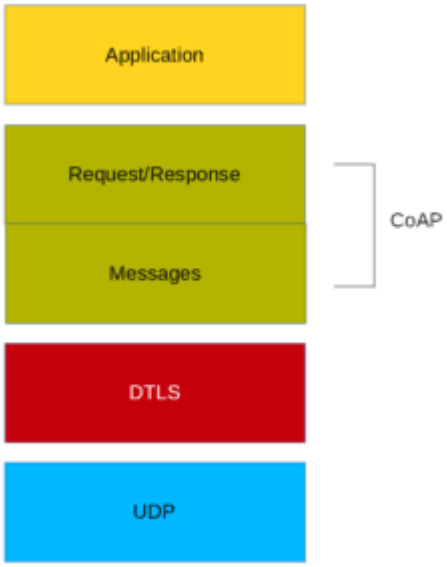
More useful resources:

- MQTT protocol tutorial
- IoT protocols overview

CoAP security aspects

One important aspect when dealing with IoT protocols is the security aspects. As stated before, CoAP uses UDP to transport information. CoAP relies on UDP security aspects to protect the information. As HTTP uses TLS over TCP, CoAP uses *Datagram TLS over UDP*. DTLS supports RSA, AES and so on. Anyway, we should consider that in some constrained devices some of DTLS cipher suits may not be available. It is

important to notice that some cipher suites introduces some complexity and constrained devices may not have resources enough to manage it.



CoAP vs MQTT

An important aspect to cover is the main differences between CoAP and MQTT. As you may know MQTT is another protocol widely used in IoT. There are several differences between these two protocols. The first aspect to notice is the different paradigm used. MQTT uses a publisher-subscriber while CoAP uses a request-response paradigm. MQTT uses a central broker to dispatch messages coming from the publisher to the clients. CoAP is essentially a one-to-one protocol very similar to the HTTP protocol. Moreover, MQTT is an event oriented protocol while CoAP is more suitable for state transfer.

Message Queue Telemetry Transport Protocol (MQTT):

MQTT is simple, lightweight messaging protocol used to establish communication between multiple devices. It is TCP-based protocol relying on the publish-subscribe model. This communication protocol is suitable for transmitting data between resource-constrained devices having low bandwidth and low power requirements. Hence this messaging protocol is widely used for communication in IoT Framework.

Publish-Subscribe Model :
This model involves multiple clients interacting with each other, without having any direct connection established between them. All clients communicate with other clients only via third party known as Broker.

MQTT Client and Broker :
Clients publish messages on different topics to broker. The broker is the central server that receives these messages and filters them based on their topics. It then sends these messages to respective clients that have subscribed to those different topics.

Hence client that has subscribed to a specific topic receives all messages published on that topic.

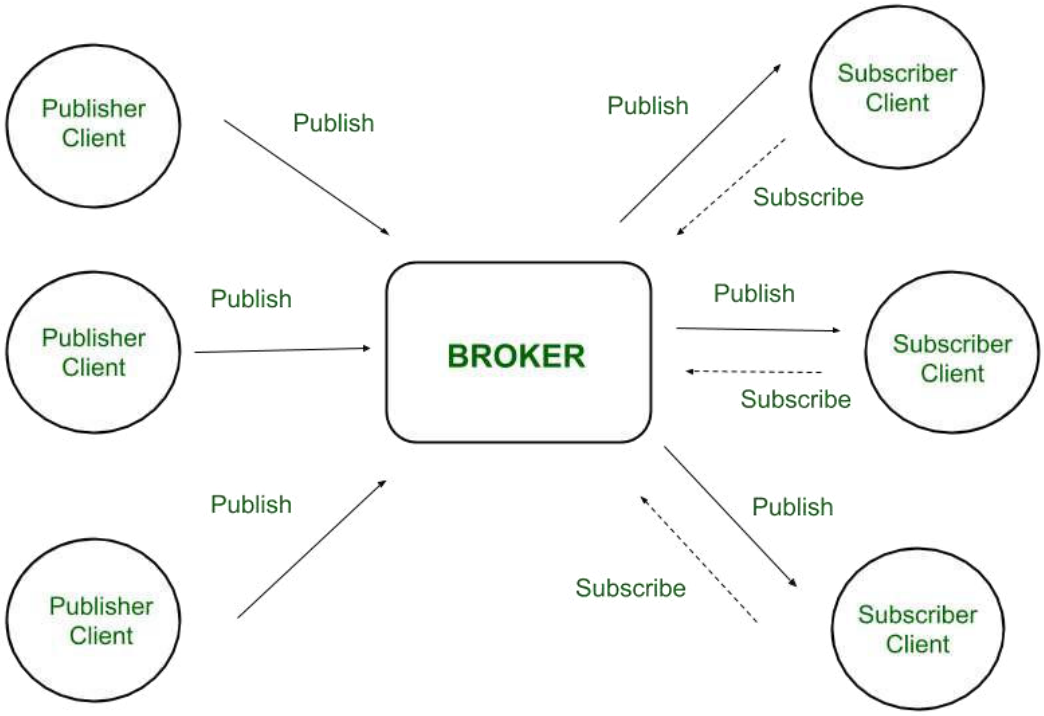


Figure – Publish-Subscribe Model

Here the broker is central hub that receives messages, filters them, and distributes them to appropriate clients, such that both message publishers, as well as subscribers, are clients.

Advantages :

1. **Easy Scalability –**
   This model is not restricted to one-to-one communication between clients. Although the publisher client sends a single message on specific topic, broker sends multiple messages to all different clients subscribed to that topic. Similarly, messages sent by multiple such publisher clients on multiple different topics will be sent to all multiple clients subscribed to those topics. Hence one-to-many, many-to-one, as well as many-to-many communication is possible using this model. Also, clients can publish data and at the same time receive data due to this two-way communication protocol. Hence MQTT is considered to be bi-directional protocol. The default unencrypted MQTT port used for data transmission is 1883. The encrypted port for secure transmission is 8883.
2. **Eliminates insecure connections –**
   In a complex system where multiple devices are connected with each other, each device not only has to manage its connections with other devices but also has to ensure that these connections are secure. But in the publish-subscribe model, the broker becomes central server managing all security aspects. It is responsible for the authentication and authorization of all connected clients.
3. **Lightweight Communication –**
   Data transmission is quick, efficient, and lightweight because MQTT messages have small code footprint. These control messages have a fixed header of size 2 bytes and payload message up to size 256 megabytes.

Topics :
In MQTT, topic is UTF-8 string that the broker uses to filter messages for each individual connected client. Each topic consists of one or more different topic levels. Each topic level is separated by forward slash also called topic level separator. Both topics and levels are case-sensitive.

Example of topic –

```
home/kitchen/table
```

Here, “home”, “kitchen” and “table” are different levels of topic.

Wildcard is an additional feature used in MQTT to make topics and their levels more flexible and user-friendly.

MQTT Topics include two types of wildcards:

1. Single Level: “+”
Single-level wildcard represented by “+” symbol can replace single level in topic.

Example –
If the client wants information about all tables present inside the house, it will subscribe to the topic :

```
home/+/table
```

Hence any information published related to tables, inside the kitchen, living room, bedroom, etc, can be obtained on this topic.

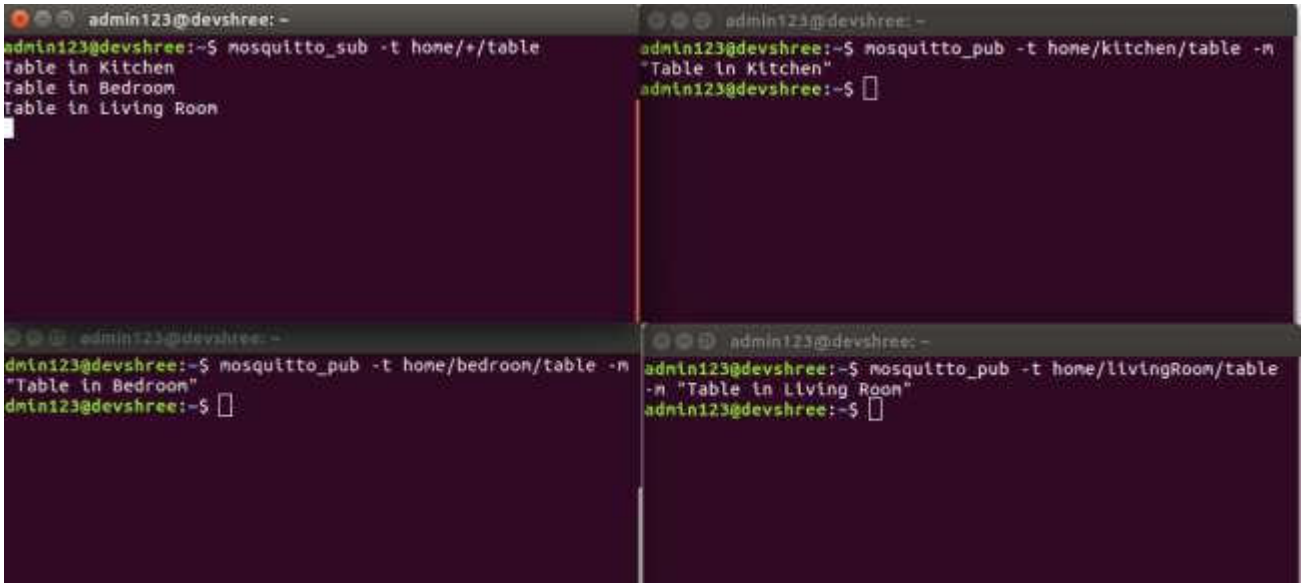


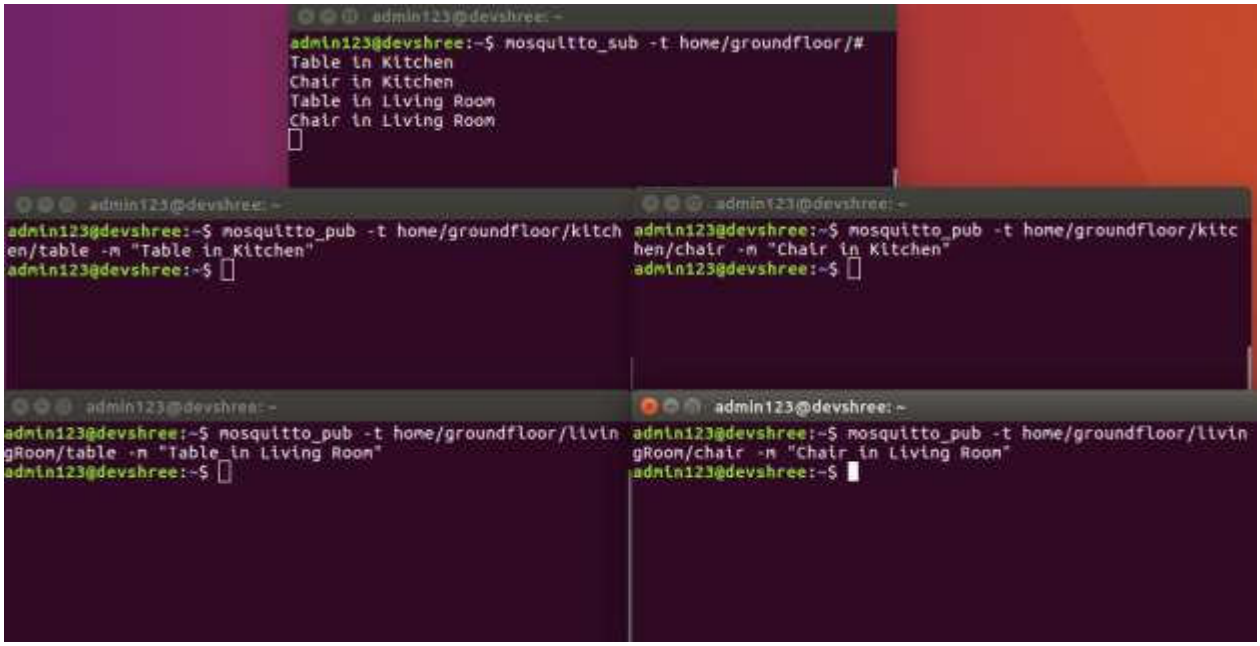
Figure – Single-Level Topics in MQTT

2. Multi-Level: “#”
Multi-level wildcard represented by “#” symbol can replace multiple levels in topic.

Example –
If a client wants information about all objects present inside the kitchen, living room, bedroom, or any other room on ground floor, it will subscribe to topic:

```
home/groundfloor/#
```

Hence any information published on topics related to kitchen items, bedroom items, living room items can be obtained on this topic. Information up to multiple levels can be obtained in this case.



Overview of IoT Framework

IoT (Internet of Things) is a network of devices which are connected to the internet for transferring and sensing the data without much human intervention, the framework used to this is termed as the IoT framework, this framework consists all the required capabilities for the cloud support and other needs which is needed to satisfy the IoT technology, few of the common IoT frameworks that are used frequently are KAA IoT,

Cisco IoT Cloud Connect, ZETTA IoT, SAP IoT, IBM Watson, Hewlett Packard Enterprise, etc

List of IoT Framework

Now we will discuss the IoT Framework one by one

1. KAA IoT

Kaa IoT is one of the most effective and rich Open Source Internet of Things Cloud Platforms, where anyone can freely implement their smart product concepts. You can manage an N number of devices connected to each other with cross-device interoperability on this platform. You can monitor your machine in actual time by providing and configuring remote devices. Kaa enables information exchange between linked devices, the IoT Cloud, information and visualization systems, as well as other elements of IoT Ecosystems

2. Cisco IoT Cloud Connect

Cisco IoT Cloud Connect provides robust, automated, and highly secure connectivity for the enterprise. IoT data management is done by the Cisco Kinetic IoT platform to extract, move and compute the data. As Cisco is very famous for its security services, it protects IoT deployment against threats with a secure **IoT architecture**.

3. ZETTA IoT

Zetta is nothing but a server-oriented platform developed based on the REST, NodeJS, and the Siren hypermedia-API-strip flow-based reactive programming philosophy. After being abstracted as REST APIs they are connected with cloud services. These internet services include tools for visualizing machine analytics and support such as Splunk. It builds a gero-distributed network through connectivity with systems like Heroku to endpoints like Arduino and Linux hackers.

4. Salesforce IoT

Salesforce is power by thunder. Thunder allows companies to unlock earlier unseen ideas and allows anyone to take proactive, personalized activities from any device to bring their clients closer than ever. More than 150,000 clients worldwide were held by Salesforce. Salesforce has a 19.7% market share in the globe of CRM. SAP (12.%1), Microsoft (6.2%), Oracle (9.1%) are far behind its nearest rivals. Many businesses now develop their apps or migrate to Salesforce on the Salesforce platform. This has raised demand for developers and **administrators from Salesforce**.

5. DeviceHive IoT

DeviceHive is another rich IoT open-source platform that is distributed under the Apache 2.0 license and can be used and changed free of charge. It provides deployment options for **Docker and Kubernetes** and can be downloaded and used both by public and personal cloud. You can run batch analysis and machine learning above and beyond

your device information. DeviceHive supports several libraries, including Android and iOS.

6. Oracle IoT

We surely include Oracle, a worldwide software company known to offer its top level of solutions in database management, and business software, as we compare the top Internet-of-Things platforms. Oracle offers its flexible environment outstanding company possibilities to create company applications. Oracle supports the processing and builds large-scale IoT networks with very wide data. The use of advanced security systems to protect IoT systems against external threats is another worth mentioning. Since these systems usually have different devices, some of which have no security tool, it is not sufficiently justifiable to implement centralized security measures.

7. SAP IoT

The SAP Internet of Things cloud platform has everything you need to build and handle an IoT application. The SAP platform provides a convenient environment to remotely manage and monitor all connected devices of your IoT system. In the SAP Platform a remote-devices we can connect directly or through cloud service. Obviously, SAP can use IoT information to create **machine learning and artificial intelligence** applications while maintaining recent technological trends.

8. Microsoft Azure IoT

Without the Microsoft Azure solution, a cloud service giant with **AWS and Google Cloud platform**, the comparison of our IoT platform will be not complete. The Microsoft Azure IoT Suite provides preconfigured solutions and the ability to personalize and develop new solutions to meet the project requirements. The strongest safety mechanisms, superb scalability and simple integration with your current or future systems are achieved through Microsoft Azure Internet of thing Suite.

9. Google Cloud Platform – IoT framework

Things can be done by Google. Google Cloud is one of the best IoT systems available today with its end-to-end platform. Google stands out from the others because it can process the large quantity of information using Cloud IoT Core. Due to Google's Cloud Data Studio and Big Query you get advanced analysis. With the **help of Google Cloud Platform**, you can accelerate your business and with that, you can speed up your

Reference Books:

1. Jan Holler, VlasiosTsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, ― From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence‖, 1st Edition, Academic Press, 2014.
2. ArshdeepBahga, Vijay Madisetti - Internet of Things: A Hands-On Approach, Universities Press, 2014.
3. The Internet of Things, Enabling technologies and use cases – Pethuru Raj, Anupama C. Raman, CRC Press.
4. Francis daCosta, ―Rethinking the Internet of Things: A Scalable Approach to Connecting Everything‖, 1st Edition, Apress Publications, 2013
5. Cuno Pfister, Getting Started with the Internet of Things, O"Reilly Media, 2011, ISBN: 9781- 4493- 9357-1

6. DGCA RPAS Guidance Manual, Revision 3 – 2020
7. Building Your Own Drones: A Beginners' Guide to Drones, UAVs, and ROVs, John Baichtal

P.HARI KRISHNA

Unit-4

What is The Internet of Things or “IoT”?

In the most basic of terms, the Internet of things is just that; it’s every physical device that is connected to the internet. If it has a chip or is considered “smart,” that item is part of the internet of things, otherwise known as the IoT. The IoT encompasses all smart devices, and we do mean ALL. Everything from hairbrushes, smartwatches, kitchen appliances, Industrial sorters, and weather vane data collectors; if it’s connected to the World Wide Web, it’s part of the IoT.

The real beauty of the Internet of Things is it allows devices to communicate and collaborate over huge distances. Gone are the days when each factory, office, and organization had to function within a closed system and physically carry information back and forth. Organizations across the globe can now have their devices updated in real-time.

Three Main Areas of IoT Usage

All physical devices connected to the internet are the definition of the Internet of Things in the broadest sense. But it’s beginning to have a secondary meaning when it comes to systems that are interconnected via the internet and depend on the internet to wirelessly connect to each other within a system or with certain parameters. These systems could be on a small or large scale. There are three areas of IoT technology where the application of The Internet of Things has been widely used.

- Consumer

If you think of your home as the parameter, in this case, every smart device you own is part of your own Internet of Things. This includes everything from wearable devices, smartphones, smart toasters, tablets, televisions, blow dryers, smart blankets, baby monitors, kitchen appliances, and children’s toys. If it’s in your home, connected to your internet, and powered on, it’s part of your very own IoT.

- Enterprise

On the other end of the scale, a large company’s Internet of Things could refer to all of its manufacturing equipment, computers, servers, alarm systems, safety systems, and data collectors across all of its multiple factories.

Public or Government Usage

Government or Public usage of IoT technology has always made people squirm from an ethical standpoint, but the actual applications are far from nefarious and typically streamline systems that have been hopelessly complicated up until IoT became widely available.

Similarly, city, state, and federal governments have used IoT technology to regulate things like gas emissions, water levels in drought-prone areas, sewer effectiveness, and water quality. Cities are now able to detect water contamination much faster than in previous decades, saving many citizens from contamination.

IoT Technologies and Cybersecurity

Though the areas mentioned above are some of the largest applications of IoT technology; there are hundreds of other areas where the internet of things is invaluable. Cybersecurity industries have also found applications, and pitfalls, in the use of the Internet of Things.

- Lots of Data Transfer, Lots of Vulnerability

The Internet of Things was a world-changing invention; however, since its creation in 1999, the IoT has posed a significant security and privacy issue. Every device on the Internet of Things can become a potential security issue. Smart devices are designed to continually gather data and communicate back and forth with other devices. There are two main areas where the cybersecurity threat in IoT technologies is especially apparent.

- Consumer

Let's say the toothbrush tracks your brushing habits, how long you brush, when, and where. The toothbrush records all that data and then communicates it through your wireless internet router and your internet server, which then relays it to multiple points, including a satellite (most likely).

Enterprise/Public

When IoT technology is unprotected on the industry or public scale, the use of IoT technology becomes a serious security issue waiting to happen. Cyberthreat hackers can potentially hijack systems, devices, and machines or shut down public works and demand ransoms. They could infiltrate government offices and mine sensitive data, leading to identity theft and possible threats to national security. Enterprise and Public works attacks are only projected to go up.

Telnet Remote Access Protocol

Unfortunately, the cybersecurity threat to enterprise and public works isn't just theoretical. In the first half of 2021 alone, 1.51 billion IoT data breaches occurred. All of these attacks used the telnet remote access protocol. This protocol is the backbone of IoT technologies and improves upon HTTP and FTP need for end-user approval for data transfer. Telnet automates communications and allows systems to communicate back and forth as needed. It's an extremely useful protocol and widely used. There has been a significant increase in IoT cyberattacks using Telnet remote access protocol during the last five years, peaking during the wind-down of the Covid-19 pandemic. This coincides with the ever-increasing amount of businesses and organizations moving to fully remote or hybrid office settings.

P.HARI KRISHNA

Cybersecurity and IoT Device Discovery

While the vulnerabilities of IoT technologies sound terrifying, there is good news. Cybersecurity companies have been working to mitigate these threats for decades and have even used the Internet of Things framework to create a secure network.

IoT device discovery is the most efficient way to protect IoT technologies from being breached by cyberattacks. The practice of IoT device discovery provides the framework that allows more specific security protocols to function. IoT device discovery is an automated system to onboard, vet, and analyze each endpoint device that connects to it. Information technology departments can then specify what endpoint devices are part of their IoT network and which are potential threats. This allows IT and security departments to see what devices interact with their networks by placing automated gatekeepers when unknown devices try to sign in to the network.

- IoT Device Discovery and Industries/Public Works

When it comes to many types of industry and public works, IoT device discovery is pretty straightforward. These companies, organizations, and government entities are not interacting with consumers and therefore don't have a constant influx of new devices trying to onboard to their network.

The IT department can set strict parameters around its network that only allows for certain types of devices with access tags attached to their IP address. It can also keep track of and monitor the number and type of devices in its network. Only pre-approved devices are allowed onto the network. New devices have to be registered and assigned access tags. It's very similar to having a physical key card to get into your office building, but the key card is a digital code attached to your device.

Simple IoT Device Discovery protocols also come in handy in industry settings where only so many devices should be active on the network. If a new device appears, IoT device discovery flags the device, can quarantine it, and instantly alert their cybersecurity teams.

- IoT Device Discovery and Consumer Interaction

IoT device discovery becomes much more complicated as soon as consumer interaction is involved. This pertains to all areas of IoT technological applications like entertainment, research, healthcare, consumer-facing enterprise, consumer products, and government/public works. In organizations like this, there are countless end-user devices constantly accessing networks. For example, public Wi-Fi, smart devices, and GPS systems. When dealing with varying numbers of devices and no stable parameters, you need a more dynamic version of IoT device discovery software.

- IoTVAS NSE Script

  IoTVAS is short for IoT vulnerability assessment solution and is a more advanced form of IoT device discovery. This system helps enterprises and organizations vet their end-user devices more thoroughly by analyzing device inventories. IoTVAS then reads the 'fingerprint' of the

devices by identifying the device by using one of several device features. These features can include the HTTP and HTTPS services or raw response of the device web server, a Telnet service banner, or an optional MAC (Media Access Control) on the device's network interface.

The Internet of Things: Made Safe Through IoT Devices Discovery

As the world of cyberthreat and data breaches evolves, so will cybersecurity industry professionals who strive to mitigate and repel cyber attacks. Since 1999, the Internet of Things has been bringing devices together, pushing enterprises and STEM pursuits to places humanity could only have dreamt of in the not-so-distant past. As IoT technologies take their next steps into future applications and industries, the IoT Device Discovery protocol will continue to shield it from potential threats.

Why use an IoT platform?

An IoT platform is critical to building an IoT ecosystem, it simplifies IoT, making it more secure, regardless of where you are on your IoT journey to build smart, connected products.

IoT is a complex ecosystem that spans a network of devices and software applications touching multiple parts of the physical and digital landscapes. It is rare for an organization to maintain in-house expertise across all the relevant domains to build a complete set of IoT capabilities. As a result, in the "buy versus build" debate over IoT capabilities, most enterprises see value in buying an IoT platform to provide an out-of-the-box set of key capabilities, on top of which the business can build differentiating logic and applications.

IoT platform capabilities

At a basic level, IoT platforms should allow you to connect and manage your devices with ease, offer application enablement and integration tools, and analyze your IoT data for actionable insights.

IoT connectivity

Connection is at the heart of IoT: devices are connected using protocols to share information and enable new insights. An IoT platform provides out-of-the-box connectivity to many device types and protocols.

For devices that do not support standard IoT protocols, an IoT platform is especially valuable if it offers a software development kit (SDK) to integrate devices with the rest of your ecosystem. Leading IoT platforms enable connectivity and integration using publicly documented APIs.

IoT device lifecycle management

An IoT platform allows you to manage the lifecycle of IoT devices and sensors—from planning and onboarding, monitoring and maintenance, through to retirement—remotely from a centralized location. Robust device lifecycle management processes are often neglected in early-stage IoT projects, when the focus is on building and launching a solution, but they are fundamental to scale a rollout and maintain reliable performance.

Enterprise IoT users need to update and communicate with devices efficiently in a controlled, secured and phased way. One example of this is the bulk registration of devices. Another is updating software and firmware to maintain performance, uptime and security. An IoT platform should allow you to access and monitor critical information easily, such as system resource information, alarms and errors, cellular signal strength or GPS location.

Scalable IoT data management

IoT data is the source of insights. An IoT platform handles data logging, storing, and processing, and manages data transactions. IoT data comes from many devices and locations, and spans many data

types. IoT platforms can orchestrate action based on real-time data, and coordinate the long-term storage and analysis of large data sets to power analytics.

IoT integration

An IoT platform needs to be much more than a passive destination for data from IoT sensors. IoT needs integration to fill its promise, as integrating IoT data with other systems builds value exponentially by helping you use insights from IoT in your existing systems and processes to make better business decisions.

IoT application development

Building and maintaining IoT applications involves technical expertise, time and resources. An IoT platform with application enablement features can help remove the resource technical hurdles to building and deploying applications.

Many businesses see value in enabling their IoT users to develop custom applications with an application builder—or by extending the platform's default applications to meet their specific business needs and requirements.

IoT data analytics

The value of IoT is not in the fleet of devices and sensors an organization is monitoring, but in the accurate and relevant data derived from these IoT devices and sensors. And the value of that data comes from analytics.

An IoT platform with powerful analytic capabilities enables you to access this key data and discover insights. You can create dashboards that pull together data, so you have a single view of the status of all devices and how your project's performing.

An IoT platform with self-service analytics puts key data into the hands of many. The more widely accessible your insights, the greater their value across the entire enterprise.

Registering a IOT device

In order for an appliance or other asset to become "smart" and connect to an IoT backend, it must have sensors that can take device readings and send that information to the cloud. We call these sensors IoT devices.

Field technicians might need to interact with IoT devices in several ways:

1. If a field technician is installing an entirely new asset like an air conditioner, and that air conditioner has an embedded IoT device or devices, they must register that device with the appropriate IoT backend in order for it to start working.
2. Field technicians might need to service an existing asset, and then install a new IoT device that will start sending signals to an IoT backend.
3. While onsite, a field technician might need to interact with and receive data from existing IoT devices in order to better troubleshoot their maintenance or repairs.

In this article, we're going to look at a few ways to register a new IoT device in Connected Field Service, and make sure that IoT device is associated with the correct customer asset in Field Service. While you can use custom IoT providers with Connected Field Service, we're going to use Azure IoT Hub.

Connected Field Service must be connected to Azure IoT Hub or another custom IoT provider. For more information, see the article on getting set up with Azure IoT Hub, or the article on setting up custom IoT providers.

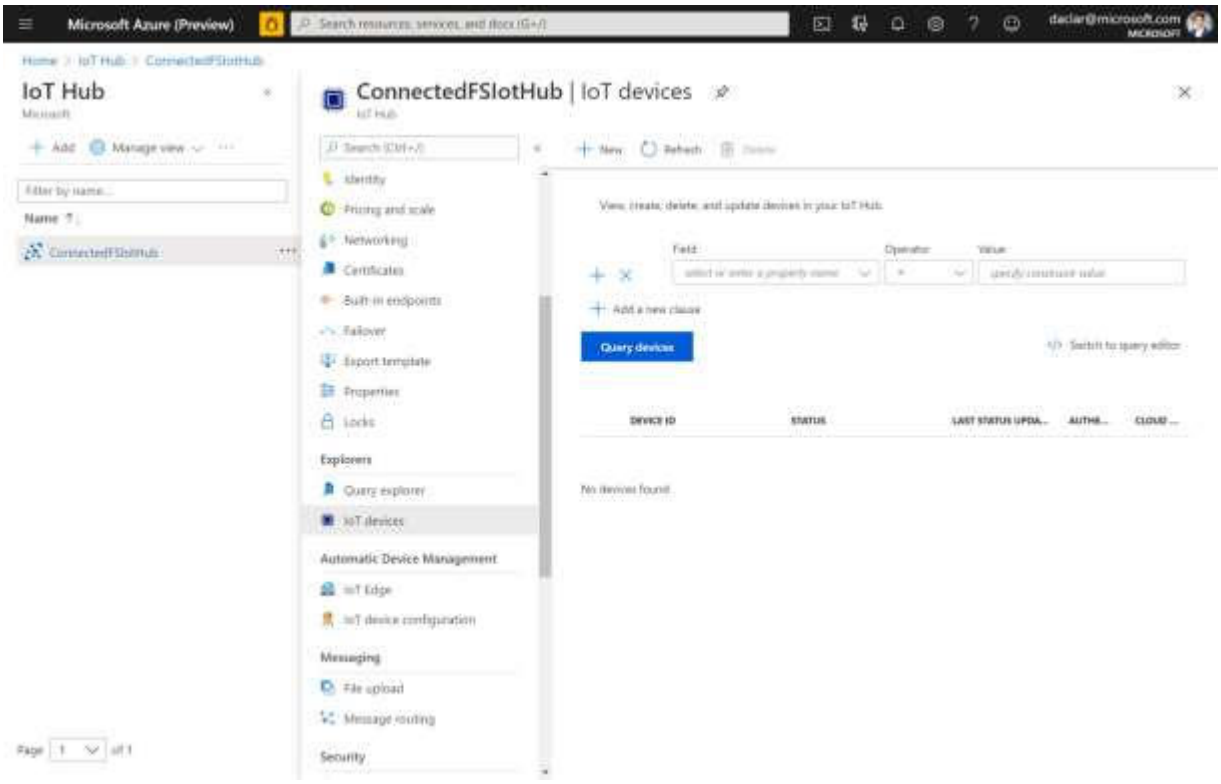
P.HARI KRISHNA

Create and register an IoT device from IoT Hub

First, let's look at how to register a new device from Azure IoT Hub.

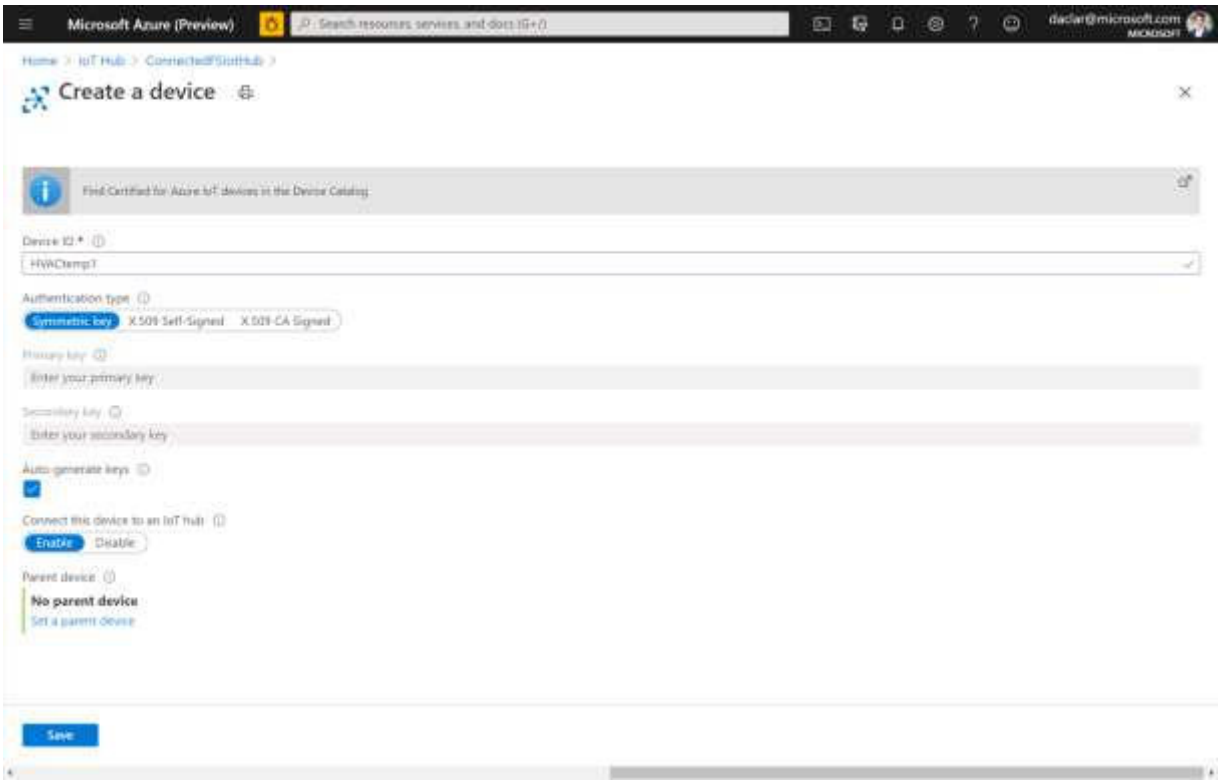
Go to Azure IoT Hub and select an environment.

From the environment, go to **IoT devices > +New**.

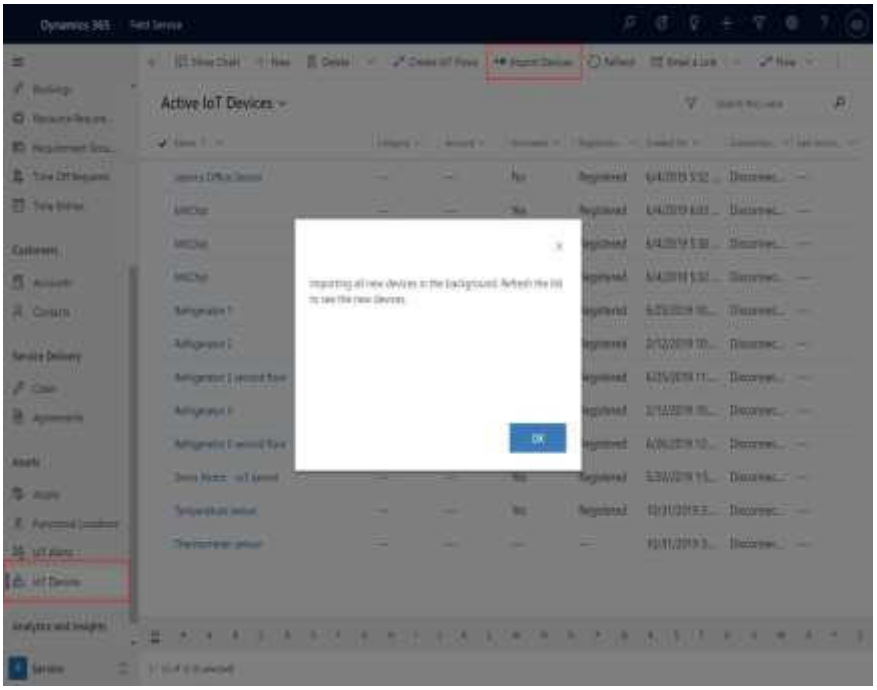


Give the device a descriptive ID (in our example, we name it "HVACtemp1") and **Save**.

P.HARI KRISHNA



Head over to Dynamics 365 Field Service, then to **Assets > IoT Devices**, and select **Import Devices**.



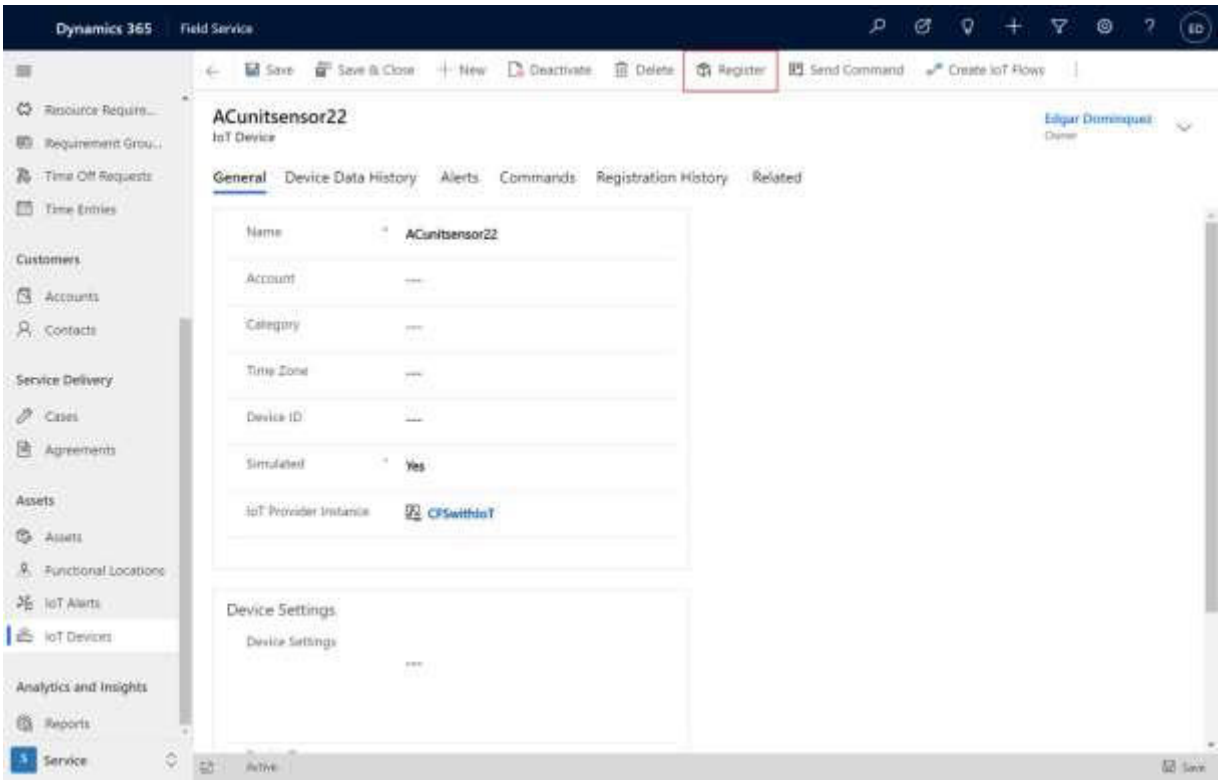
Any new devices registered back in Azure IoT Hub will now appear in the list of active IoT devices in Field Service.

We'll still need to connect this new device to the relevant customer asset, which will we do later in this article.

Create and register an IoT device from Field Service

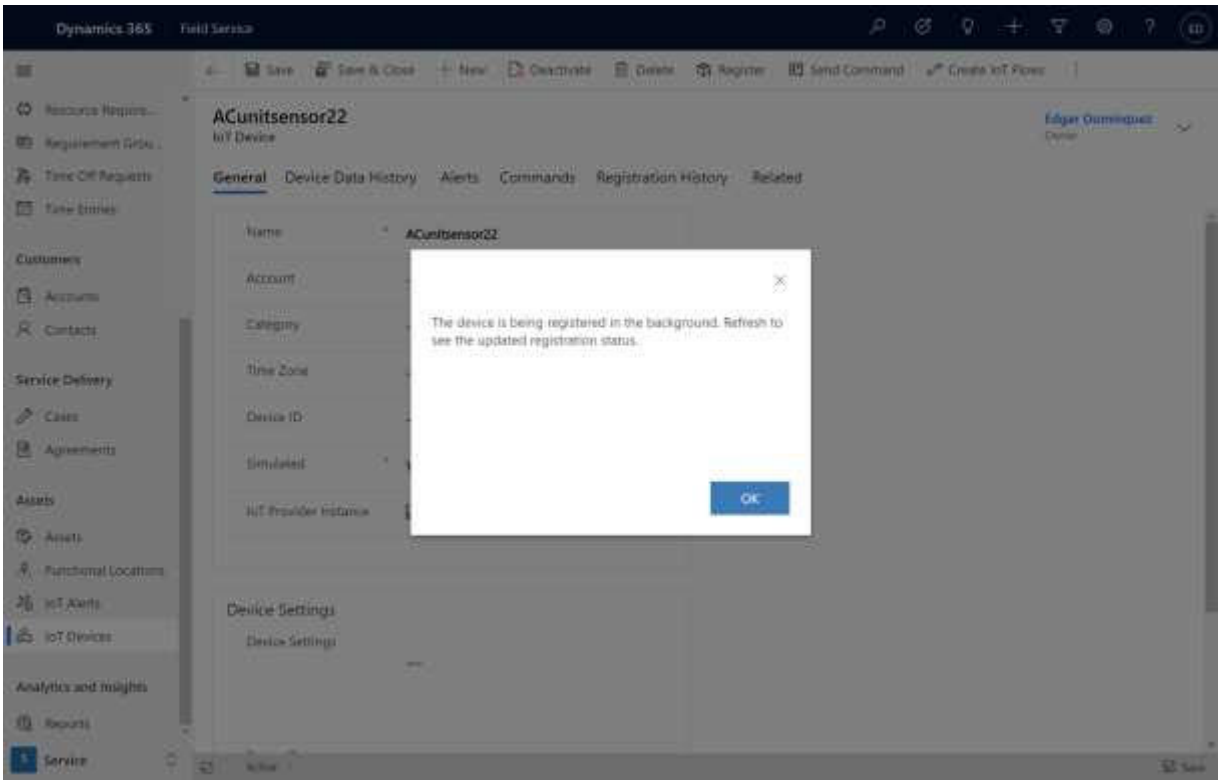
You can also create an IoT device directly in Field Service.

From **Field Service > Assets > IoT Devices > +New**. Give the IoT device a descriptive name, then **Save**. Finally, select **Register** in the top ribbon.

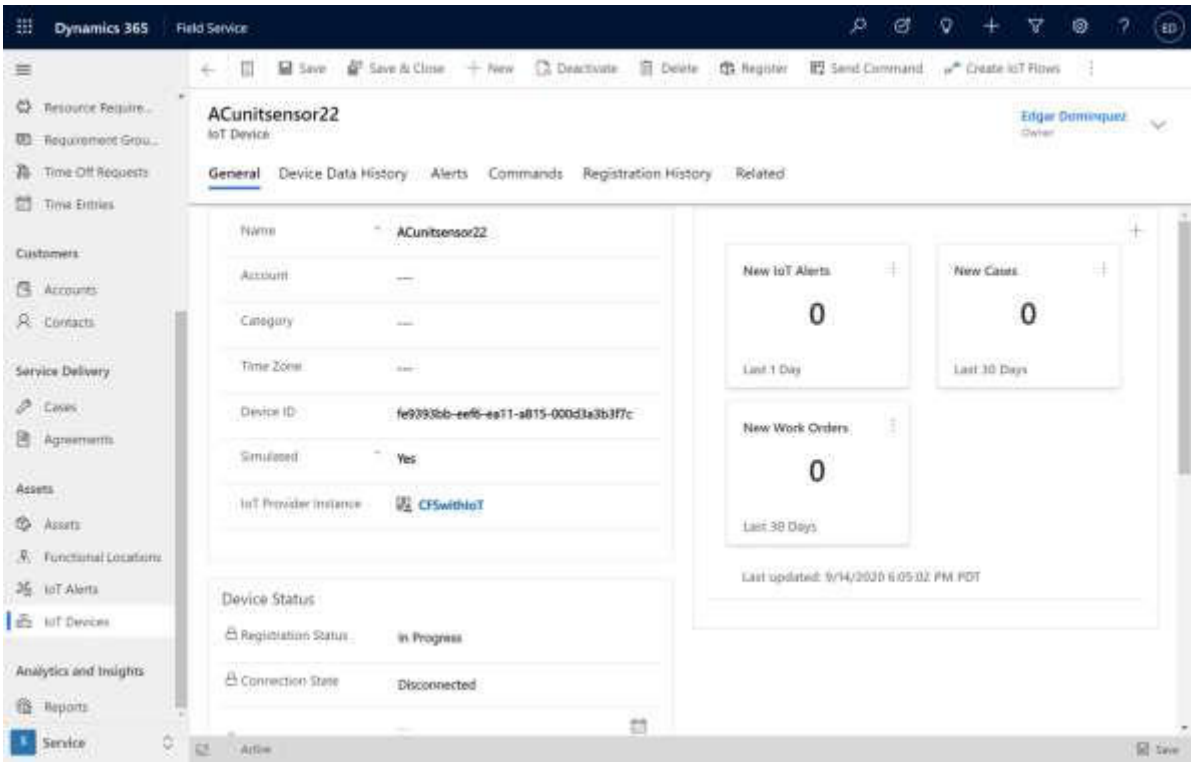


Selecting **Register** sends the new device information back to Azure IoT Hub, which the system tells you with a message seen in the following screenshot.

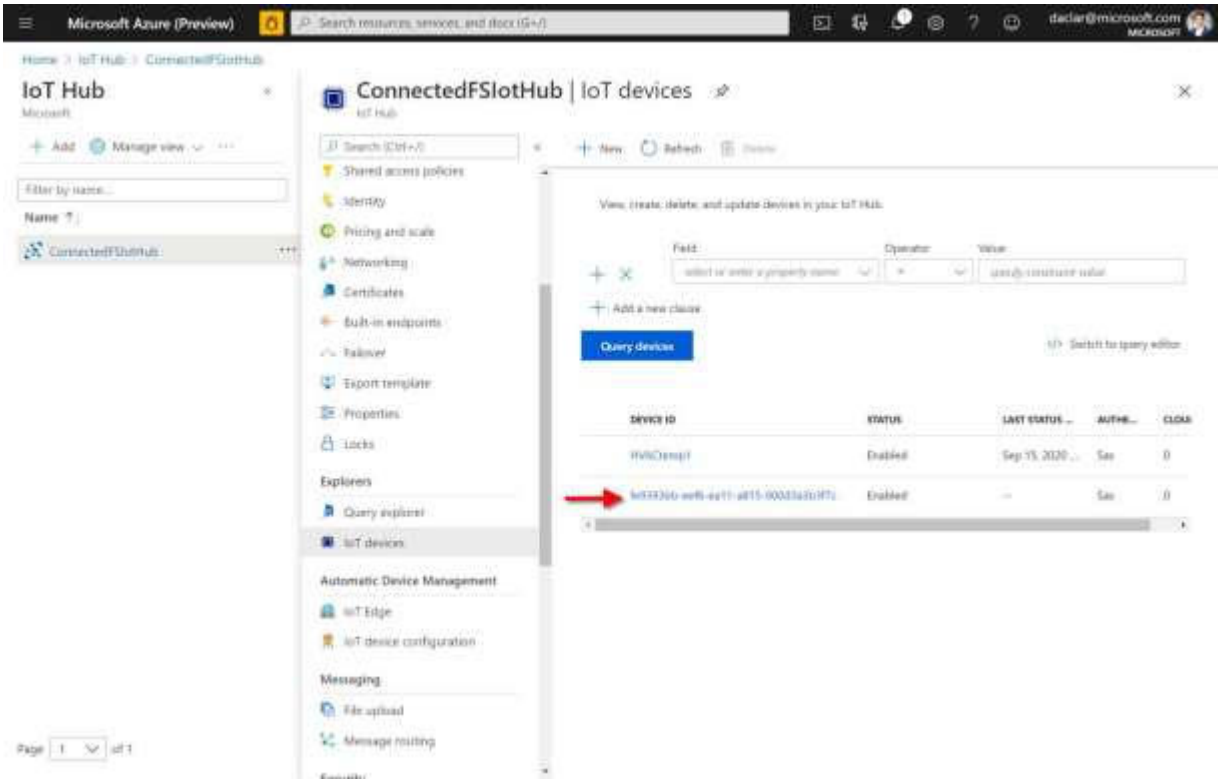
P.HARI KRISHNA



Once the device is synced back to Azure IoT Hub, a device ID will be generated and synced back to the IoT device in Field Service.



Back in Azure IoT Hub, we now see the IoT device we created in Field Service, with its new device ID.



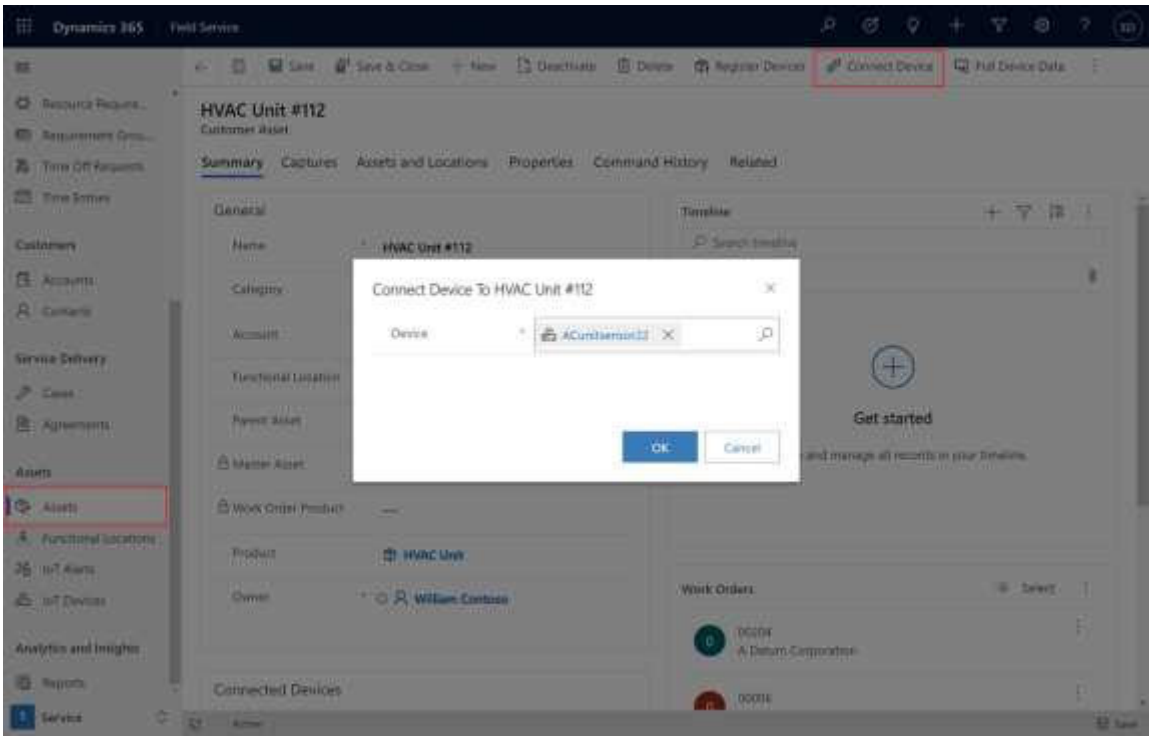
Connect to asset

After we've created the IoT device record, we'll need to associate it with a new or existing customer asset. For instance: in our example, we have a new IoT device sensor that has been installed in an air conditioning unit.

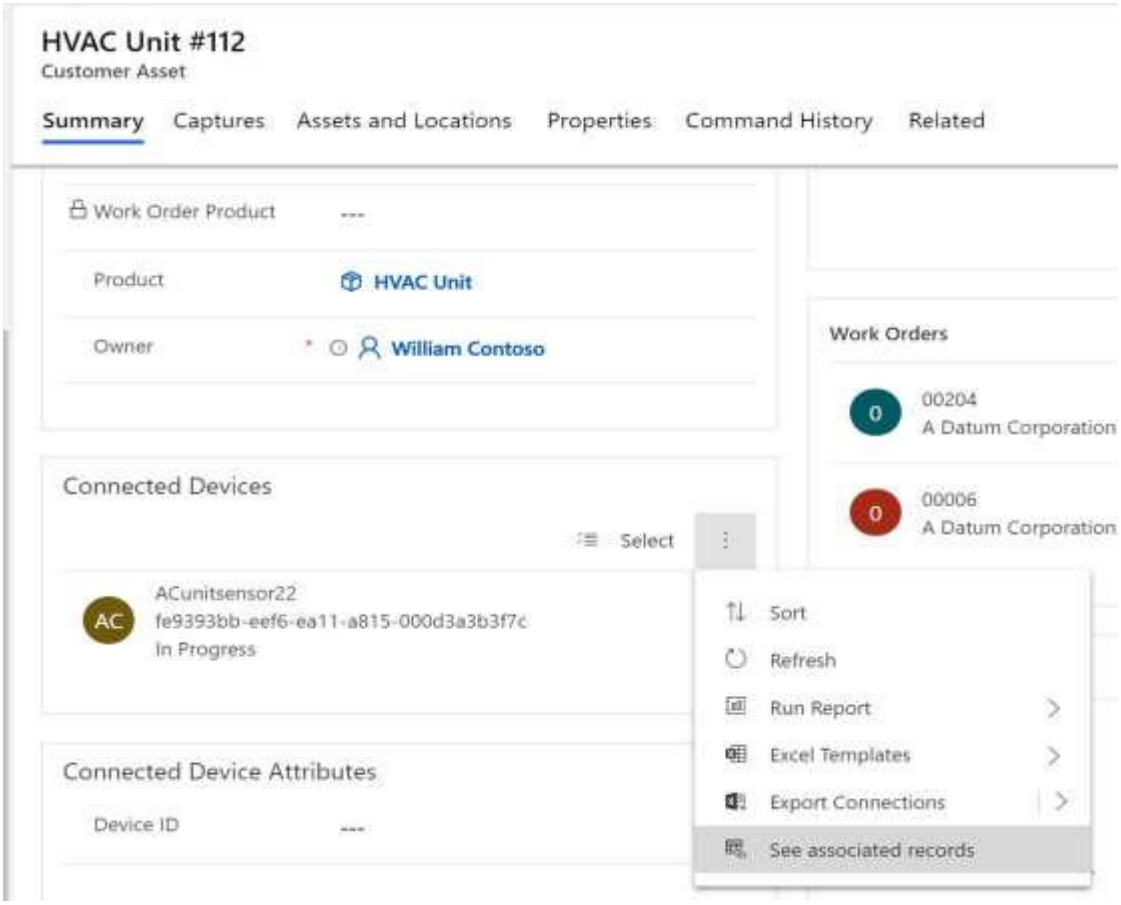
In Field Service, go to **Assets** and find the customer asset you need to associate with the new IoT device.

From the customer asset, select **Connect Device** in the top ribbon, and look up the newly created IoT device.

P.HARI KRISHNA



Once the IoT device has been associated with the asset, a new section will appear called **Connected Devices**, where you can see information about the new IoT device.



Note

- Multiple IoT devices can be associated with a single customer asset. On the asset, go to **Related** > **Connections** to associate additional IoT devices.
- When you're associating a device with an asset, you can set a primary device ID for the asset. If you associate multiple devices with the asset, the primary device ID for the asset won't be displayed in the form; however, a device ID link'll still exist in the background. Also, the first or the oldest device that's associated with the asset will be the primary device ID for the asset. If you delete the link to the primary device ID, then the next oldest-connected device ID will be set as the primary ID.

Registration error

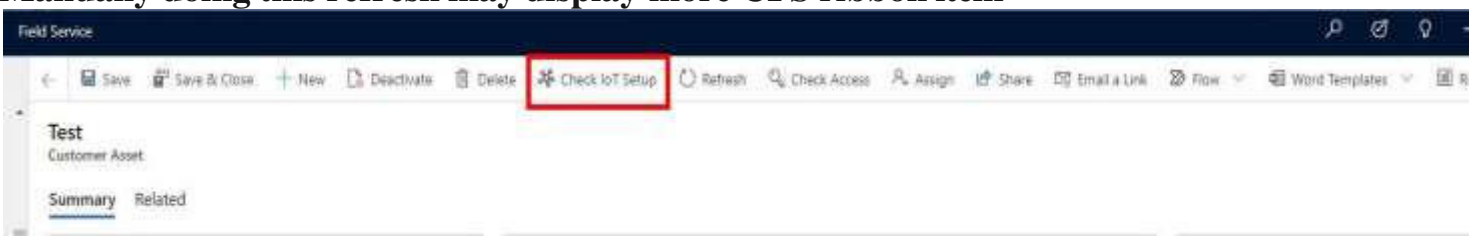
When a device does not register, you'll see an error in the **Registration Status** field of the asset record. For assets with multiple devices, device status will show the error message for each device.

This error may occur if Dynamics 365 is not connected to Azure or if Azure is offline.

Additional notes

P.HARI KRISHNA

Use the Check IoT Setup **ribbon button to check for information about the IoT setup such as, is IoT deployed, are there any devices in the environment. Manually doing this refresh may display more CFS ribbon item**



Deregister a device

To avoid end-user disruption, device deregistration in Windows Autopatch only deletes the Windows Autopatch device record itself. Device deregistration can't delete Microsoft Intune and/or the Azure Active Directory device records. Microsoft assumes you'll keep managing those devices yourself in some capacity.

To deregister a device:

1. Sign into the Intune admin center.
2. Select **Windows Autopatch** in the left navigation menu.
3. Select **Devices**.
4. In either **Ready** or **Not ready** tab, select the device(s) you want to deregister.
5. Once a device or multiple devices are selected, select **Device actions**, then select **Deregister device**.

Warning

Removing devices from the Windows Autopatch Device Registration Azure AD group doesn't deregister devices from the Windows Autopatch service.

Excluded devices

When you deregister a device from the Windows Autopatch service, the device is flagged as "excluded" so Windows Autopatch doesn't try to reregister the device into the service again, since the deregistration command doesn't trigger device membership removal from the **Windows Autopatch Device Registration** Azure Active Directory group.

Important

The Azure AD team doesn't recommend appending query statements to remove specific device from a dynamic query due to dynamic query performance issues.

If you want to reregister a device that was previously deregistered from Windows Autopatch, you must submit a support request with the Windows Autopatch Service Engineering Team to request the removal of the "excluded" flag set during the deregistration process. After the Windows Autopatch Service Engineering Team removes the flag, you can reregister a device or a group of devices.

P.HARI KRISHNA

Hiding unregistered devices

You can hide unregistered devices you don't expect to be remediated anytime soon.

To hide unregistered devices:

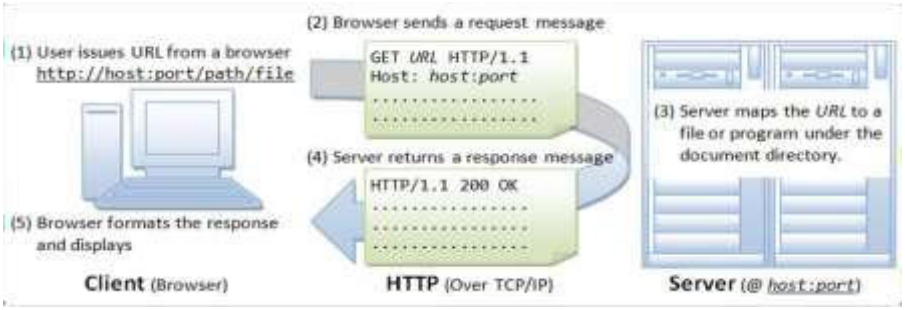
1. Sign into the Intune admin center.
2. Select **Windows Autopatch** in the left navigation menu.
3. Select **Devices**.
4. In the **Not ready** tab, select an unregistered device or a group of unregistered devices you want to hide then select **Status == All**.
5. Unselect the **Registration failed** status checkbox from the list.

What's a web server and HTTP

Any computer that can implement http or https is able to play the role of a web server. Http is a protocol, a way of communication which supplies web pages. It is pretty widely used and easy to implement. Through http you can transfer html and create simple user interfaces, it can implement Java Script and make more complicated web pages and it is available in most of the browsers. One of the great qualities of this protocol is that it replaced complicated and heavy displays with user friendly web pages.

How does it work? The browser sends a request to the server who searches the demanded page and returns it to the browser for the user. The request will consist of information about the kind of browser that is used, about the computer or about the document requested. It will have a method, a URL, a query string and the upload body in case you want data to be sent to the server.

The response will include the status, which tells the browser if the page was found or not (the errors among the 400s are about a not found page, 300 are redirections and 200s are confirmations of the page being found).



Https has two important security roles.

- It encripts the data. The request and the response will be both encripted on sending and decripted when read.
- The server is always asked for a certificate of authenticity before it is asked for a page. This prevents against stolen data through false web pages.

What does a query consist of? It will always look like this: http://address:[port]URL?querystring. The port can be absent, in which case it will be 80 for http and 443 for https. It has to be specified if it is not one the two. Concerning the URL, when it is not written, the default will be /. The available methods in http are: get, post, put and delete. The main ones being the first two.

- Get method needs no upload body. It will only ask for data from the server and send only the headers, the address, the URL.

- Post sends important data to the server, which will be uploaded. Post has the role of modifying data on the server.

  The response of both these methods is the page and any additional information that was requested.
- Put is similar to post, only that in the semantic way, this method only creates an object on the server.
- Delete also plays a semantic part. It needs no upload body and it delete objects on the server. The same action can

  be performed however using get.

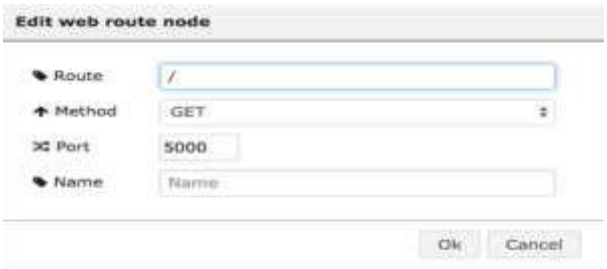
On one server there can be more than one websites,which means that, if the host is not specified in the request, the response may not be the one the browser expects.

Also, the response may have more than text. Any additional feature: images, JavaScript objects and so on will need a new request, so the process will be slowed down.

Webservers on gadgets using Wyliodrin

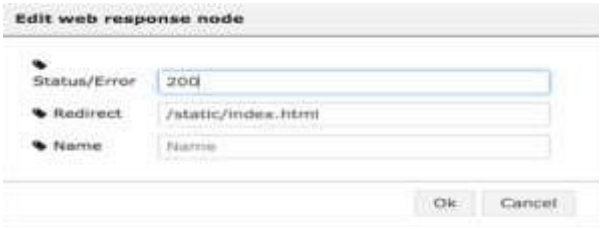
The boards are nonpowerful computers. With wyliodrin there is no need to install any software or make any configuration on the boards to run a webserver on them.

To create a webserver in wyliodrin you will need a *web* node. The simplest way to use a web page in this particular way is to send static files. In the project files, create a new folder *static*. Everything inside it will be sent back to the browser by the server, regardless of the fact that they are html, Java Script or CSS files. Images can be added as well, but they will definitely make the process slower. There are other ways of adding an image. For example, by using a storage system and including the images from there. This method will solve speed and memory issues.
The *web node*: The route option is actually the URL. The webserver will be active when it stumbles upon the specified route. Afterwords you choose the method, and write the port to setup the server. This port will only be used once, in the beginning.



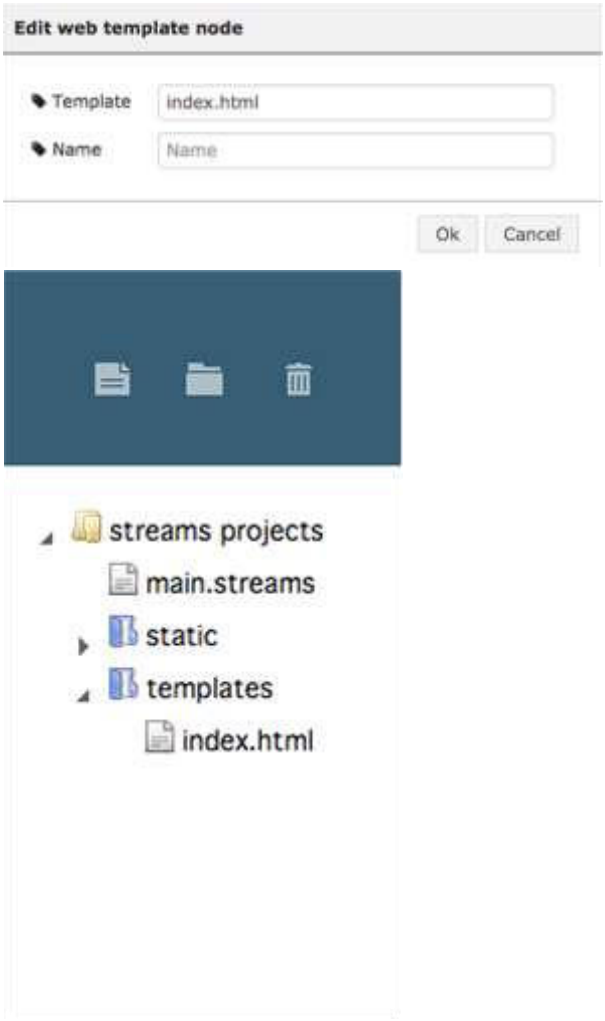
The payload goes either in the query string for the get method or in the upload data for post. The message is built on this payload, on two mandatory variables: *res* which stands for the response and *req* which is the request. Without the last two, the server won't be able to provide a response.

The *web response* node: The message received by this node comes from one web node. For a web response the simplet way is to make a redirection. Which means, in the redirect field, you can write the path to one of the static files and the browser will be sent to this page. On top of these, you will need the board's IP address which might not be public unless it is in the same network with the web server.



As a solution, IOT servers have a public address. The port for these servers can be either 80 for http or 443 for https. The user accesses the public page, through the IOT server which is connected to Wyliodrin as well as the board. Now the problem with the board and the web being in the same network is solved as both can communicate with Wyliodrin.

Web templates: Just as for the static files, you will need a *templates* folder. This time, when you use the node, you don't need the whole path. You can only write the name of the file in the templates folder. What does the node do? It processes the response, meaning it loads the values plus the payload in it and sends it back to the browser. The values need to be in between two sets of curly brackets *{{}}*. Note that the values won't update unless the page is reloaded.



Web services

A long time ago, the web services were more complicated. Now the application only requests the web server for the data, and it is the browser's job to rearrange it so that it is in the right format for the application.

How to implement it into a Wyliodrin application? Using a simple web response and web server node, you send a static page to the user and each time you make a query, instead of a template, you use a web response node and send the payload to the browser, which can be a number, an object or anything else.

JQuery

There is a library called JQuery, based on JavaScript, thus available in any browser, which can make function calls to the server.

Case study: You have the following situation: you change the payload into a variable which stores values from a sensor. You want this variable to be shown in your web page. Practically, when an API gets called, what you will do, will be to make a get request to the server using the web address that you want with the URL */sensor*. The web page will send values that you will store in a variable in your html file.

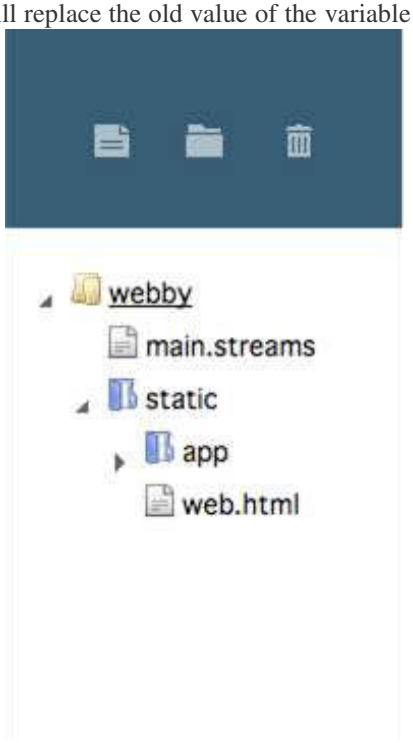
Web sockets

A web socket is based on the http or https protocol. It builds a connection between the browser and the server, so that either one can send data. When the browser makes a request, the server recognises the socket and doesn't close the connection.

The two partie send the packages they need to send. If the server does not know how to work with sockets, the socket io will go back to querying.

AngularJS

AngularJS is a library through which you can build browser applications. In the next example, every web node will create a new socket and serve a static web page. If you include in the response a variable, and this variable changes, wyliodrin controller will be notified every time this kind of novelty appears and AngularJS will replace the old value of the variable with the new one, creating a dynamic web page.



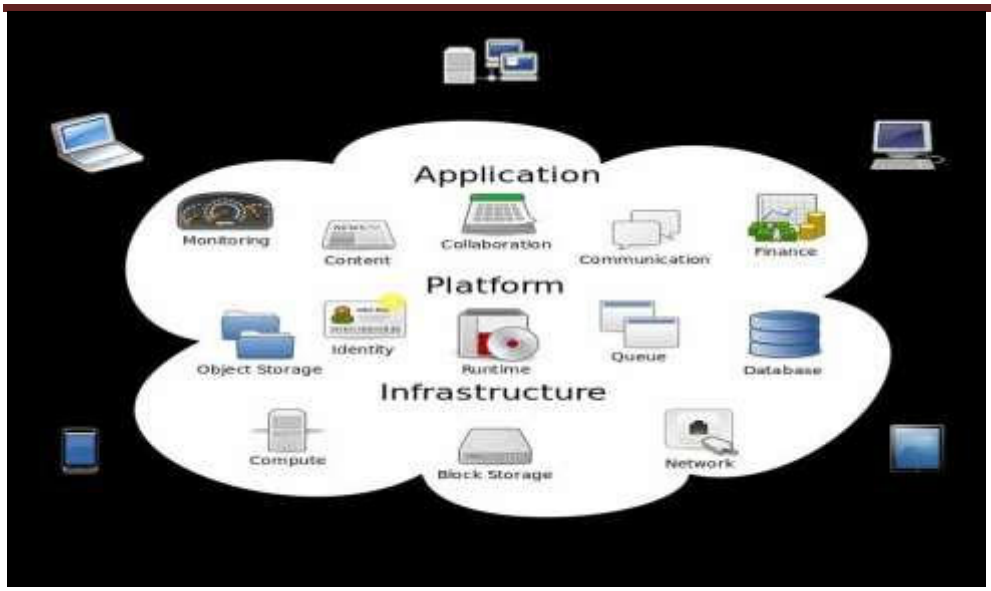

```
<!DOCTYPE html>
<html ng-app="myApp">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script src="/socket.io/socket.io.js"></script>
    <script src="/static/app/controllers/wyliodrincontroller.js"></script>
  </head>
  <body ng-controller="WyliodrinCtrl">
    {{random}}
  </body>
</html>
```

Introduction to Cloud Storage Models & Communication APIs

In truth, cloud computing and IoT are tightly coupled. The growth of IoT and the rapid development of associated technologies create a widespread connection of things. This has lead to the production of large amounts of data, which needs to be stored, processed and accessed. Cloud computing as a paradigm for bigdata storage and analytics. While IoT is exciting on its own, the real innovation will

come from combining it with cloud computing. The combination of cloud computing and IoT will enable new monitoring services and powerful processing of sensory data streams. For example, sensory data can be uploaded and stored with cloud computing, later to be used intelligently for smart monitoring and actuation with other smart devices. Ultimately, the goal is to be able to transform data to insight and drive productive, cost-effective action from those insights. The cloud effectively serves as the brain to improved decision-making and optimized internet-based interactions. However, when IoT meets cloud, new challenges arise. There is an urgent need for novel network architectures that seamlessly integrate them. The critical concerns during integration are quality of service (QoS) and quality of experience (QoE), as well as data security, privacy and reliability. The virtual infrastructure for practical mobile computing and interfacing includes integrating applications, storage devices, monitoring devices, visualization platforms, analytics tools and client delivery.

Cloud computing offers a practical utility-based model that will enable businesses and users to access applications on demand anytime and from anywhere.



Deployment models

Deployment in cloud computing comprises four deployment models: private cloud, public cloud, multi cloud and hybrid cloud.

Cloud storage models

What is Cloud Storage?

Cloud Storage is a mode of computer data storage in which digital data is stored on servers in off-site locations. The servers are maintained by a third-party provider who is responsible for hosting, managing, and securing data stored on its infrastructure. The provider ensures that data on its servers is always accessible via public or private internet connections.

Cloud Storage enables organizations to store, access, and maintain data so that they do not need to own and operate their own data centers, moving expenses from a capital expenditure model to operational. Cloud Storage is scalable, allowing organizations to expand or reduce their data footprint depending on need.

Google Cloud provides a variety of scalable options for organizations to store their data in the cloud. Learn more about Cloud Storage at Google Cloud.
How does Cloud Storage work?

Cloud Storage uses remote servers to save data, such as files, business data, videos, or images. Users upload data to servers via an internet connection, where it is saved on a virtual machine on a physical server. To maintain availability and provide redundancy, cloud providers will often spread data to multiple virtual machines in data centers located across the world. If storage needs increase, the cloud provider will spin up more virtual machines to handle the load. Users can access data in Cloud Storage through an internet connection and software such as web portal, browser, or mobile app via an application programming interface (API).

Cloud Storage is available in four different models:

Public

Public Cloud Storage is a model where an organization stores data in a service provider’s data centers that are also utilized by other companies. Data in public Cloud Storage is spread across multiple regions and is often offered on a subscription or pay-as-you-go basis. Public Cloud Storage is considered to be “elastic” which means that the data stored can be scaled up or down depending on the needs of the organization. Public cloud providers typically make data available from any device such as a smartphone or web portal.

Private

Private Cloud Storage is a model where an organization utilizes its own servers and data centers to store data within their own network. Alternatively, organizations can deal with cloud service

providers to provide dedicated servers and private connections that are not shared by any other organization. Private clouds are typically utilized by organizations that require more control over their data and have stringent compliance and security requirements.

Hybrid

A hybrid cloud model is a mix of private and public cloud storage models. A hybrid cloud storage model allows organizations to decide which data it wants to store in which cloud. Sensitive data and data that must meet strict compliance requirements may be stored in a private cloud while less sensitive data is stored in the public cloud. A hybrid cloud storage model typically has a layer of orchestration to integrate between the two clouds. A hybrid cloud offers flexibility and allows organizations to still scale up with the public cloud if need arises.

Multi cloud

A multi cloud storage model is when an organization sets up more than one cloud model from more than one cloud service provider (public or private). Organizations might choose a multi cloud model if one cloud vendor offers certain proprietary apps, an organization requires data to be stored in a specific country, various teams are trained on different clouds, or the organization needs to serve different requirements that are not stated in the servicers' Service Level Agreements. A multi cloud model offers organizations flexibility and redundancy.

Advantages of Cloud Storage

Total cost of ownership

Cloud Storage enables organizations to move from a capital expenditure to an operational expenditure model, allowing them to adjust budgets and resources quickly.

Elasticity

Cloud Storage is elastic and scalable, meaning that it can be scaled up (more storage added) or down (less storage needed) depending on the organization's needs.

Flexibility

Cloud Storage offers organizations flexibility on how to store and access data, deploy and budget resources, and architect their IT infrastructure.

Security

Most cloud providers offer robust security, including physical security at data centres and cutting edge security at the software and application levels. The best cloud providers offer zero trust architecture, identity and access management, and encryption.

Sustainability

One of the greatest costs when operating on-premises data centres is the overhead of energy consumption. The best cloud providers operate on sustainable energy through renewable resources**.**

Redundancy

Redundancy (replicating data on multiple servers in different locations) is an inherent trait in public clouds, allowing organizations to recover from disasters while maintaining business continuity.

Disadvantages of Cloud Storage

Compliance

Certain industries such as finance and healthcare have stringent requirements about how data is stored and accessed. Some public cloud providers offer tools to maintain compliance with applicable rules and regulations.

Latency

Traffic to and from the cloud can be delayed because of network traffic congestion or slow internet connections.

Control

Storing data in public clouds relinquishes some control over access and management of that data, entrusting that the cloud service provider will always be able to make that data available and maintain its systems and security.

Outages

While public cloud providers aim to ensure continuous availability, outages sometimes do occur, making stored data unavailable.

P.HARI KRISHNA

Communication API'S

A cloud storage API is an application program interface that connects locally-based application to a cloud-based storage system, so that a user can send data to it and access and work with data stored in it. To the application, the cloud storage system is just another target device, like tape or disk-based storage. An application program interface (API) is code that allows two software programs to communicate with each other. The API defines the correct way for a developer to write a program that requests services from an operating system (OS) or other application. APIs are implemented by function calls composed of verbs and nouns. The required syntax is described in the documentation of the application being called

Three basic types of APIs

APIs take three basic forms: local, web-like and program-like.

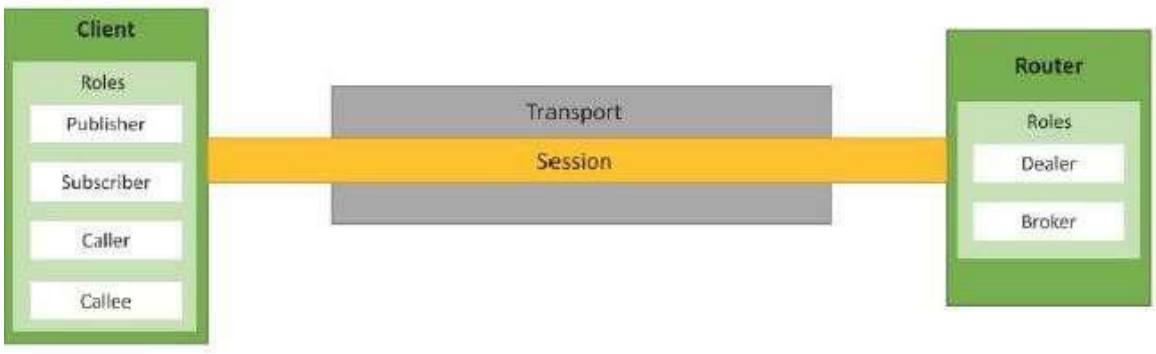
1. **Local APIs** are the origin al form, from which the name came. They offer So r middleware services to application programs. Microsoft's.NET APIs, the TAPI(TelephonyAPI) for voice applications, and database access APIs are examples of the local API form.

2. **Web APIs** are designed to represent widely used resources like HTML pages and are accessed using a simple HTTP protocol. Any web URL activates a web API. Web APIs are often called REST (representational state transfer) or RESTful because the publisher of REST interfaces doesn't save any data internally between requests. As such, requests from many users can be intermingled as they would be on the internet.

3. **Program APIs** are based on remote procedure call (RPC) technology that makes a remote program component appear to be local to the rest of the software. Service oriented architecture (SOA) APIs, such as Microsoft's WS-series of APIs, are program APIs.

WAMP-Auto Bahn for IoT

Web Application Messaging Protocol (WAMP) is a sub-protocol of Websocket which provides publish-subscribe and remote procedure call (RPC) messaging patterns.



Transport: Transport is channel that connects two peers.

• Session: Session is a conversation between two peers that runs over a transport.

• Client: Clients are peers that can have one or more roles. In publish-subscribe model client can have following roles:

– Publisher: Publisher publishes events (including payload) to the topic maintained by the Broker.

– Subscriber: Subscriber subscribes to the topics and receives the events including the payload. In In RPC model client can have following roles: –

1. Caller: Caller issues calls to the remote procedures along with call arguments. –

2. Callee: Callee executestheprocedurestowhichthecallsareissuedbythecallerandreturnstheresultsbacktothe caller. •

3. Router: Routers are peers that perform generic call and event routing. In publish-subscribe model

4. Router has the role of a Broker: – Broker: Broker acts as a router and routes messages published to a topic to all subscribers subscribed to the topic.

In RPC model Router has the role of a Broker:–

1. Dealer: Dealer acts a router and routes RPC calls from the Caller to the Callee and routes results from Callee to Caller.

2. Application Code: Application code runs on the Clients (Publisher, Subscriber, Callee or Caller).

Xively Cloud for IoT

Use of Cloud IoT cloud-based service

•The service provides for the data collection, data points, messages and calculation objects.

• The service also provisions for the generation and communication of alerts, triggers and feeds to the user.

•A user is an application or service. The user obtains responses or feeds from the cloud service.

Pachube platform: for data capture in real-time over the Internet

• Cosm: a changed domain name, where using a concept of console, one can monitor the feeds

•Xively is the latest domain name.

A commercial PaaS for the IoT/M2M

•A data aggregator and data mining website often integrated into the Web of Things

• An IoT PaaS for services and business services.

Xively PaaS services:

- Data visualization for data of connected sensors to IoT devices.
- Graphical plots of collected data.
- Generates alerts.
- Access to historical data
- Generates feeds which can be real-world objects of own or others

Xively HTTP based APIs

- Easy to implement on device hardware acting as clients to Xively web services
- APIs connect to the webservice and send data.

- APIs provides services for logging, sharing and displaying sensor data of all

Xively Support

•The platform supports the REST, WebSockets and MQTT protocols and connects the devices to Xively Cloud Services

- Native SDKs for Android, Arduino, ARMmbed, Java, PHP, Ruby, and Python languages
- Developers can use the work flow of prototyping, deployment and management through the tools provided at Xively

Xively APIs

- Enable interface with Python, HTML5, HTML5 server, tornado
- Interface with WebSocket Server and Web Sockets
- Interface with an RPC (Remote Procedure Call).

XivelyPaaS services

- Enables services
- Business services platform which connects the products, including collaboration products
- Rescue, Bold chat, join.me, and operations to Internet
- Data collection in real-time over Internet

Xively Methods for IoT Devices Data

- Concept of users, feeds, data streams, data points and triggers
- Data feed typically a single location (e.g. a device or devices network),
- Data streams are of individual sensors associated with that location (for example, ambient lights, temperatures, power consumption).
- Pull or Push (Automatic or Manual Feed)

Xively Data formats and Structures

- Number of data formats and structures enable the interaction, data collection and services
- Support exists for JSON, XML and CSV
- Structures: Tabular, spreadsheet, Excel, Data numbers and Text with a comma-separated values in file

Xively Uses in IoT/M2M

- Private and Public Data Access
- Data streams, Data points and Triggers
- Creating and Managing Feeds
- Visualising Data

Django

Django is an opensource web application framework for developing web applications in Python.

• A web application frame work in general is a collection of solutions, packages and best practices that allows development of web applications and dynamic websites.

• Django is based on the Model-Template-View architecture and provides a separation of the data model from the business rules and the user interface.

• Django provides a unified API to a data base backend.

• Thus web applications built with Django can work with different databases without requiring any code changes.

• With this flexibility in web application design combined with the powerful capabilities of the Python language and the Python ecosystem, Django is best suited for cloud applications.

• Django consists of an object- relational mapper, a web templating system and a regular-expression based URL dispatcher**.** Django is Model-Template-View (MTV) framework.

Model

• The model acts as a definition of some stored data and handles the interactions with the database. In a web application, the data can be stored in a relational database, non-relational database, an XML file, etc. A Django model is a Python class that outlines the variables and methods for a particular type of data.

Template

• In a typical Django web application, the template is simply an HTML page with a few extra placeholders. Django's template language can be used to create various forms of text files (XML, email, CSS, Java script, CSV, etc.)

View

• The view ties the model to the template. The view is where you write the code that actually generates the web pages.

View determines what data is to be displayed, retrieves the data from the database and passes the data to the template.

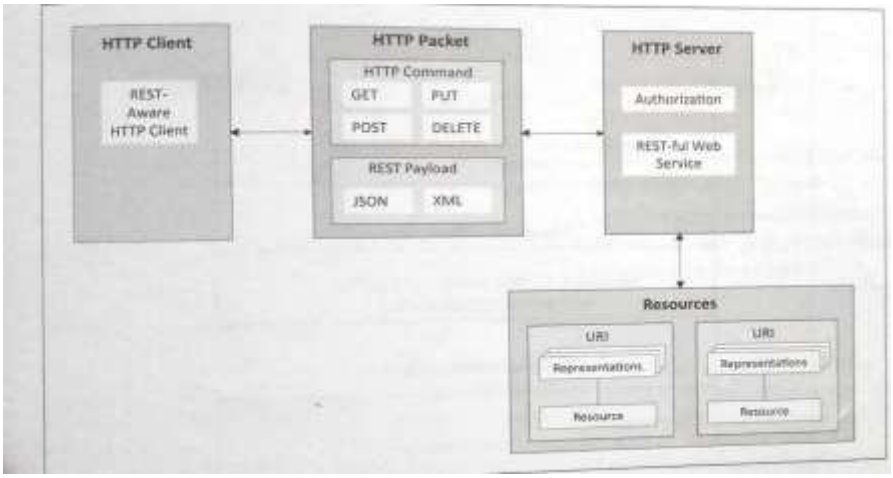
Designing a RESTful Web API

i) REST based communication APIs (Request-Response Based Model)

ii) Web Socket based Communication APIs (Exclusive Pair Based Model)

i) REST based communication APIs: Representational State Transfer (REST) is a set of architectural principles by which we can design web services and web APIs that focus on a system‘s resources and have resource states are addressed and transferred.

The REST architectural constraints are as follows: The below figure shows the communication between client server with REST APIs



Client-Server: The principle behind client-server constraint is the separation of concerns. Separation allows client and server to be independently developed and updated.

Stateless: Each request from client to server must contain all the info. Necessary to understand the request, and cannot take advantage of any stored context on the server.

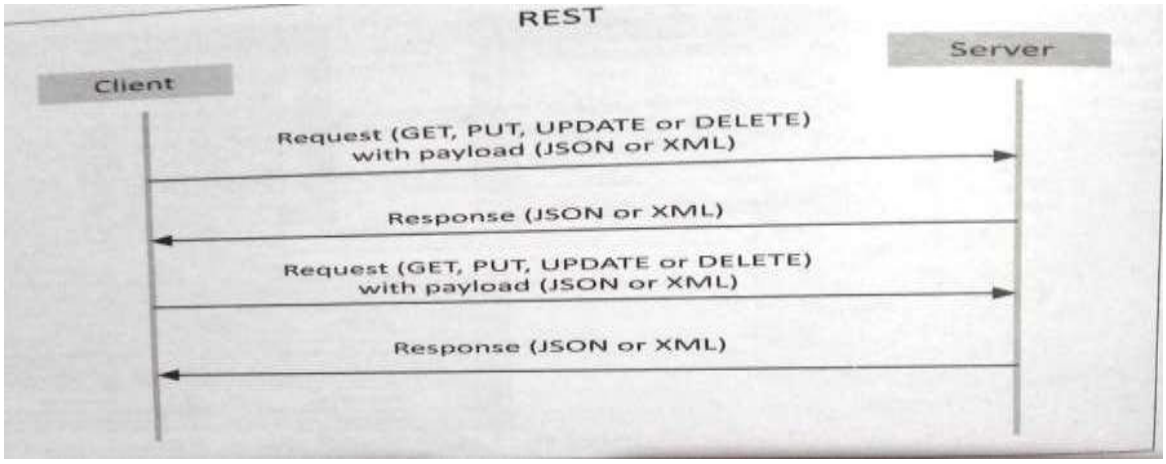
Cache-able: Cache constraint requires that the data within a response to a request be implicitly or explicitly labelled as cache-able or non-cacheable. If a response is cache-able, then a client cache is given the right to reuse that response data for later, equivalent requests.

Layered System: Constraints the behaviour of components such that each component cannot see beyond the immediate layer with which they are interacting.

User Interface: Constraint requires that the method of communication between a client and a server must be uniform.

Code on Demand: Servers can provide executable code or scripts for clients to execute in their context. This constraint is the only one that is optional.

8

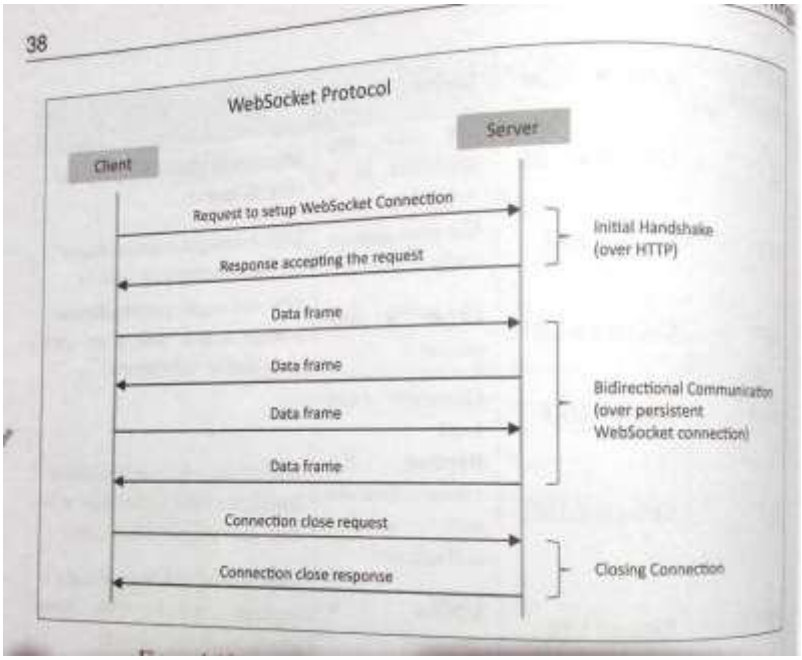


The Request-Response model used by REST:

RESTful web service is a collection of resources which are represented by URIs. RESTful web API has a base URI (e.g: http://example.com/api/tasks/). The clients and requests to these URIs using the methods defined by the HTTP protocol (e.g: GET, PUT, POST or DELETE). A RESTful web service can support various internet media types.

ii) Web Socket Based Communication APIs

WebSocket APIs allow bi-directional, full duplex communication between clients and servers. WebSocket APIs follow the exclusive pair communication model.



Amazon Web Services for IoT

i) AmazonEC2

In this example, a connection to EC2 service is first established by calling boto.ec2.connect_to_region.

• The EC2 region, AWS access key and AWS secret key are passed to this function. After connecting to EC2, a new instance is launched using the conn.run_instances function.

• The AMI-ID, instance type, EC2 key handle and security group are passed to this function.

ii) Amazon AutoScaling

AutoScaling Service

• A connection to Auto Scaling service is first established by calling boto.ec2.autoscale.connect_to_region function.

- **Launch Configuration**

• After connecting to Auto Scaling service, a new launch configuration is created by calling conn.create_launch_configuration.Launchconfigurationcontainsinstructionsonhowtolaunch new instances including the AMI-ID, instance type, security groups, etc.

Auto Scaling Group

• After creating a launch configuration, it is then associated with a new Auto Scaling group. Auto Scaling group is created by calling conn.create_auto_scaling_group. The settings for Auto Scaling group such as the maximum and minimum number of instances in the group, the launch configuration, availability zones, optional load balancer to use with the group, etc.

AutoScaling Policies

• After creating an Auto Scaling group, the policies for scaling up and scaling down are defined.

• In this example, a scale up policy with adjustment type Change In Capacity and scaling_ad justment = 1 is defined.

• Similarly a scale down policy with adjustment type Change In Capacity and scaling_adjustment = -1 is defined.

Cloud Watch Alarms

• With the scaling policies defined, the next step is to create Amazon Cloud Watch alarms that trigger these policies.

• The scale up alarm is defined using the CPU Utilization metric with the Average statistic and threshold greater 70% for a period of 60sec.The scale up policy created previously is associated with this alarm. This alarm is triggered when the average CPU utilization of the instances in the group becomes greater than 70% for more than 60 seconds.

• The scale down alarm is defined in a similar manner with a threshold less than 50%.

iii) AmazonS3:

• In this example, a connection to S3 service is first established by calling boto.connect_s3 function.

• The upload_to_s3_bucket_path function uploads the file to the S3 bucket specified at the specified path.

iv) Amazon RDS

In this example, a connection to RDS service is first established by calling boto.rds.connect_to_region function.

• The RDS region, AWS access key and AWS secret key are passed to this function.

• After connecting to RDS service, the conn.create_db instance function is called to launch a new RDS instance.

• The input parameters to this function include the instance ID, database size, instance type, database username, database password, database port, database engine (e.g. MySQL5.1), database name, security groups, etc.

v) AmazonDynamoDB

In this example, a connection to Dynamo DB service is first established by calling boto.dynamodb.connect_to_region.

• After connecting to Dynamo DB service, a schema for the new table is created by calling conn.create_schema.

• The schema includes the hash key and range key names and types.

• A Dynamo DB table is then created by calling conn.create_table function with the table schema, read units and write units as input parameters.

Sky Net IoT Messaging Platform.

Sky Net is running on a dozen Amazon EC2 servers and has nearly 50,000 registered smart devices including: Arduinos, Sparks, Raspberry Pis, Intel Galileos, and Beagle Boards, Matthieu said. Sky Net runs as an IoT platform-as-a-service (PaaS) as well as a private cloud through Docker, the new lightweight container technology. The platform is written in Node.js and released under an MIT opensource license on GitHub.

The single Sky Net API supports the following IoT protocols: HTTP, REST, Web Sockets, MQTT (Message Queue Telemetry Transport), and CoAP (Constrained Application Protocol) for guaranteed message delivery and low-bandwidth satellite communications, Matthieu said. Every connected device is assigned a 36 character UUID and secret token that act as the device's strong credentials. Security permissions can be assigned to allow device discoverability, configuration, and messaging.

Reference Books:

1. Jan Holler, VlasiosTsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, ― From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence‖, 1st Edition, Academic Press, 2014.
2. ArshdeepBahga, Vijay Madisetti - Internet of Things: A Hands-On Approach, Universities Press, 2014.
3. The Internet of Things, Enabling technologies and use cases – Pethuru Raj, Anupama C. Raman, CRC Press.
4. Francis daCosta, ―Rethinking the Internet of Things: A Scalable Approach to Connecting Everything‖, 1st Edition, Apress Publications, 2013
5. Cuno Pfister, Getting Started with the Internet of Things, O"Reilly Media, 2011, ISBN: 9781- 4493- 9357-1
6. DGCA RPAS Guidance Manual, Revision 3 – 2020
7. Building Your Own Drones: A Beginners' Guide to Drones, UAVs, and ROVs, John Baic

UNIT V IoT Physical Servers & Cloud Offerings

5.1 Introduction to Cloud Storage Models & Communication APIs –

5.2 WAMP - AutoBahn for IoT

5.3 Xively Cloud for IoT

5.4 Django

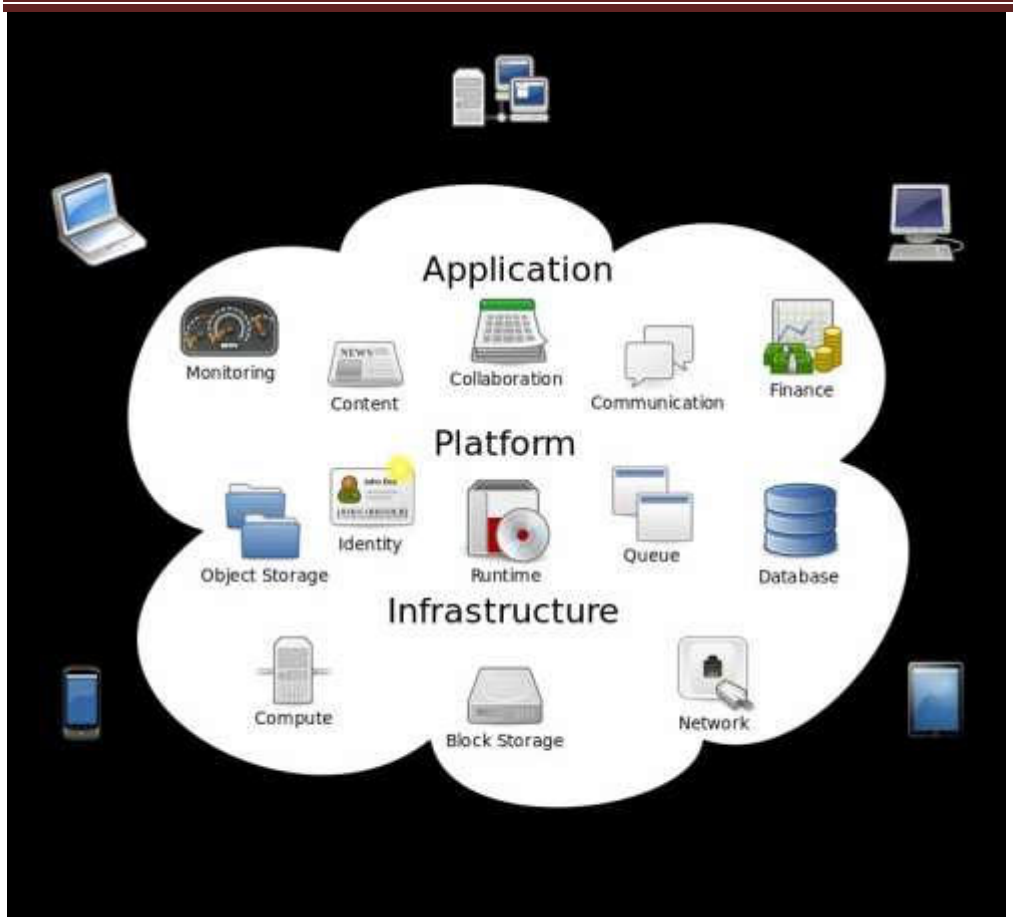
5.5 Designing a RESTful Web API

5.6 Amazon Web Services for IoT

5.7 SkyNetIoT Messaging Platform.

5.1 Introduction to Cloud Storage Models & Communication APIs

In truth, cloud computing and IoT are tightly coupled. The growth of IoT and the rapid development of associated technologiescreate a widespread connection of ―things.‖This has lead to the production of large amounts of data, which needs to be stored, processed and accessed. Cloud computing as a paradigm for big data storage and analytics. While IoT is exciting on its own, the real innovation will come from combining it with cloud computing. The combination of cloud computing and IoT will enable new monitoring services and powerful processing of sensory data streams. For example, sensory data can be uploaded and stored with cloud computing, later to be used intelligently for smart monitoring and actuation with other smart devices. Ultimately, the goal is to be able to transform data to insight and drive productive, cost-effective action from those insights. The cloud effectively serves as the brain to improved decision-making and optimized internet-based interactions.However, when IoT meets cloud, new challenges arise. There is an urgent need for novel network architectures that seamlessly integrate them. The critical concerns during integration are quality of service (QoS) and qualityof experience (QoE), as well as data security, privacy and reliability. The virtual infrastructure for practical mobile computing and interfacing includes integrating applications, storage devices, monitoring devices, visualization platforms, analytics tools and client delivery. Cloud computing offers a practical utility-based model that will enable businesses and users to access applications on demand anytime and fromanywhere.



Deployment models
Deployment in cloud computing comprises four deployment models: private cloud, public cloud, community cloud and hybrid cloud.

A cloud storage API is an application program interface that connects a locally-based application to a cloud-based storage system, so that a user can send data to it and access and work with data stored in it. To the application, the cloud storage system is just another target device, like tape or disk-based storage. An application program interface (API) is code that allows two software programs to communicate with each other. The API defines the correct way for a developer to write a program that requests services from an operating system (OS) or other application. APIs are implemented by function calls composed of verbs and nouns. The required syntax is described in the documentation of the application beingcalled

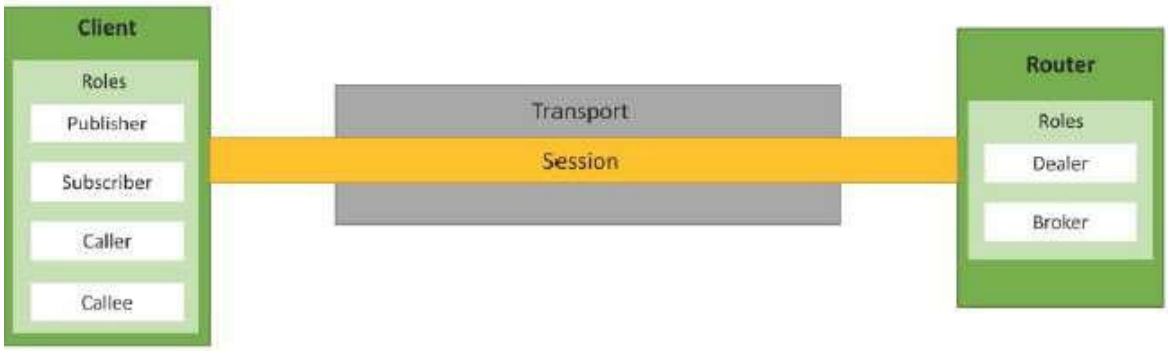
Three basic types of APIs
APIs take three basic forms: local, web-like and program-like.
1. **Local APIs** are the original form, from which the name came. They offer OS or middleware services to application programs. Microsoft's .NET APIs, the TAPI (Telephony API) for voice applications, and database access APIs are examples of the local APIform.

2. **Web APIs** are designed to represent widely used resources like HTML pages and are accessed using a simple HTTP protocol. Any web URL activates a web API. Web APIs are often called REST (representational state transfer) or RESTful because the publisher of REST interfaces doesn't save any data internally between requests. As such, requests from many users can be intermingled as they would be on theinternet.

3. **Program APIs** are based on remote procedure call (RPC) technology that makes a remote program component appear to be local to the rest of the software. Service oriented architecture (SOA) APIs, such as Microsoft's WS-series of APIs, are programAPIs.

5.2 WAMP - AutoBahn for IoT

Web Application Messaging Protocol (WAMP) is a sub-protocol of Websocket which provides publish-subscribe and remote procedure call (RPC) messaging patterns.



Transport: Transport is channel that connects two peers.

• Session: Session is a conversation between two peers that runs over a transport.

• Client: Clients are peers that can have one or more roles. In publish-subscribe model client can have following roles:

– Publisher: Publisher publishes events (including payload) to the topic maintained by the Broker.

– Subscriber: Subscriber subscribes to the topics and receives the events including the payload.

In RPC model client can have following roles:

– Caller: Caller issues calls to the remote procedures along with call arguments.

– Callee: Callee executes the procedures to which the calls are issued by the caller and returns the results back to the caller.

• Router: Routers are peers that perform generic call and event routing. In publish-subscribe model Router has the role of a Broker:

– Broker: Broker acts as a router and routes messages published to a topic to all subscribers subscribed to the topic.

In RPC model Router has the role of a Broker:

– Dealer: Dealer acts a router and routes RPC calls from the Caller to the Callee and routes results from Callee to Caller.

• Application Code: Application code runs on the Clients (Publisher, Subscriber, Callee or Caller).

In RPC model client can have following roles: –

1. Caller: Caller issues calls to the remote procedures along with call arguments. – Callee: Callee executes the procedures to which the calls are issued by the caller and returns the results back to the caller. • Router: Routers are peers that perform generic call and event routing. In publish-subscribe model Router has the role of a Broker: – Broker: Broker acts as a router and routes messages published to a topic to all subscribers subscribed to thetopic.

In RPC model Router has the role of a Broker: –

1. Dealer: Dealer acts a router and routes RPC calls from the Caller to the Callee and routes results from Callee toCaller.
2. Application Code: Application code runs on the Clients (Publisher, Subscriber, Callee or Caller).

5.3 Xively Cloud for IoT

Use of Cloud IoT cloud-based service •The service provides for the data collection, data points, messages and calculation objects. • The service also provisions for the generation and communication of alerts, triggers and feeds to the user. • A user is an application or service. The user obtains responses or feeds from the cloud service.

Pachube platform: for data capture in real-time over the Internet • Cosm: a changed domain name, where using a concept of console, one can monitor the feeds • Xively is the latest domain name.

A commercial PaaS for the IoT/M2M • A data aggregator and data mining website often integrated into the Web of Things • An IoT PaaS for services and business services.

Xively PaaS services:

• Data visualisation for data of connected sensors to IoT devices.

• Graphical plots of collected data.

• Generates alerts.

• Access to historical data

• Generates feeds which can be real-world objects of own or others

Xively HTTP based APIs

• Easy to implement on device hardware acting as clients to Xively web services

• APIs connect to the web service and send data.

• APIs provides services for logging, sharing and displaying sensor data of all

Xively Support

•The platform supports the REST, WebSockets and MQTT protocols and connects the devices to Xively Cloud Services

• Native SDKs for Android, Arduino, ARM mbed, Java, PHP, Ruby, and Python languages

• Developers can use the workflow of prototyping, deployment and management through the tools provided at Xively

Xively APIs

• Enable interface with Python, HTML5, HTML5 server, tornado

• Interface with WebSocket Server and WebSockets

• Interface with an RPC (Remote Procedure Call).

Xively PaaS services

• Enables services

• Business services platform which connects the products, including collaboration products

• Rescue, Boldchat, join.me, and operations to Internet

• Data collection in real-time over Internet

Xively Methods for IoT Devices Data

•Concept of users, feeds, data streams, data points and triggers

• Data feed typically a single location (e.g. a device or devices network),

• Data streams are of individual sensors associated with that location (for example, ambient lights, temperatures, power consumption).

• Pull or Push (Automatic or Manual Feed)

Xively Data formats and Structures

• Number of data formats and structures enable the interaction, data collection and services

• Support exists for JSON , XML and CSV

• Structures: Tabular, spreadsheet, Excel, Data numbers and Text with a comma-separated values in file

Xively Uses in IoT/M2M

• Private and Public Data Access

• Data streams, Data points and Triggers

• Creating and Managing Feeds

• Visualising Data

5.4 Django

Django is an open source web application framework for developing web applications in Python.

• A web application framework in general is a collection of solutions, packages and best practices that allows development of web applications and dynamic websites.

• Django is based on the Model-Template-View architecture and provides a separation of the data model from the business rules and the user interface.

• Django provides a unified API to a database backend.

• Thus web applications built with Django can work with different databases without requiring any code changes.

• With this fiexibility in web application design combined with the powerful capabilities of the Python language and the Python ecosystem, Django is best suited for cloud applications.

• Django consists of an object-relational mapper, a web templating system and a regular-expression based URL dispatcher**.** Django is Model-Template-View (MTV) framework.

Model

• The model acts as a definition of some stored data and handles the interactions with the database. In a web application, the data can be stored in a relational database, non-relational database, an XML file, etc. A Django model is a Python class that outlines the variables and methods for a particular type of data.

Template

• In a typical Django web application, the template is simply an HTML page with a few extra placeholders. Django's template language can be used to create various forms of text files (XML, email, CSS, Javascript, CSV, etc.)

View

• The view ties the model to the template. The view is where you write the code that actually generates the web pages.

View determines what data is to be displayed, retrieves the data from the database and passes the data to the template.

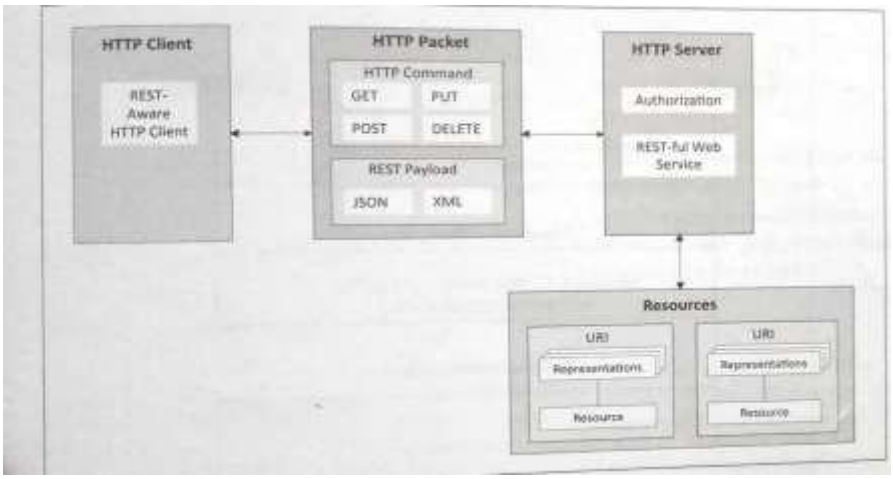
5.5 Designing a RESTful Web API

i) REST based communication APIs(Request-Response Based Model)

ii) WebSocket based Communication APIs(Exclusive Pair Based Model)

i) REST based communication APIs: Representational State Transfer(REST) is a set of architectural principles by which we can design web services and web APIs that focus on a system's resources and have resource states are addressed and transferred.

The REST architectural constraints are as follows: The below figure shows the communication between client server with REST APIs



Client-Server: The principle behind client-server constraint is the separation of concerns. Separation allows client and server to be independently developed and updated.

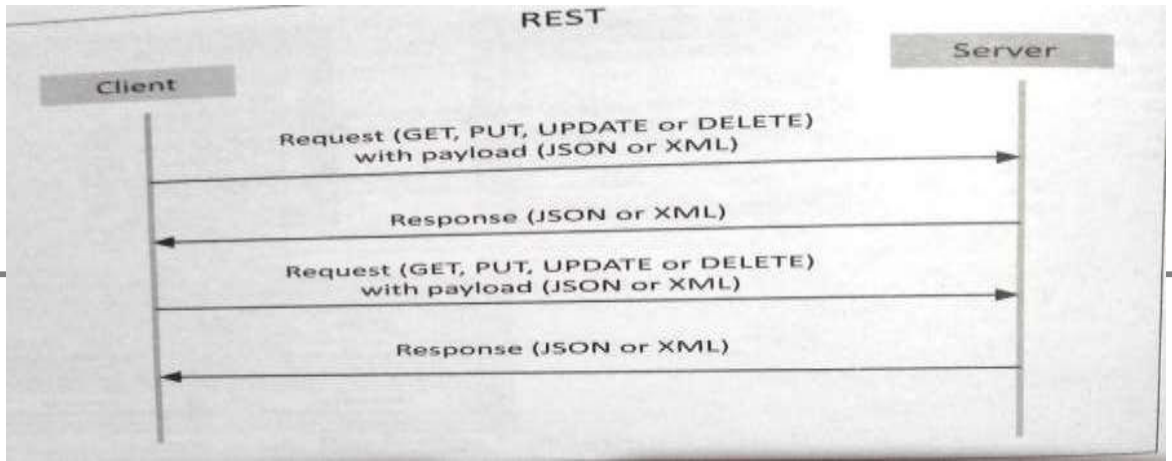
Stateless: Each request from client to server must contain all the info. Necessary to understand the request, and cannot take advantage of any stored context on the server.

Cache-able: Cache constraint requires that the data within a response to a request be implicitly or explicitly labeled as cache-able or non-cacheable. If a response is cache-able, then a client cache is given the right to reuse that response data for later, equivalent requests.

Layered System: constraints the behavior of components such that each component cannot see beyond the immediate layer with which they are interacting.

User Interface: constraint requires that the method of communication between a client and a server must be uniform.

Code on Demand: Servers can provide executable code or scripts for clients to execute in their context. This constraint is the only one that is optional.

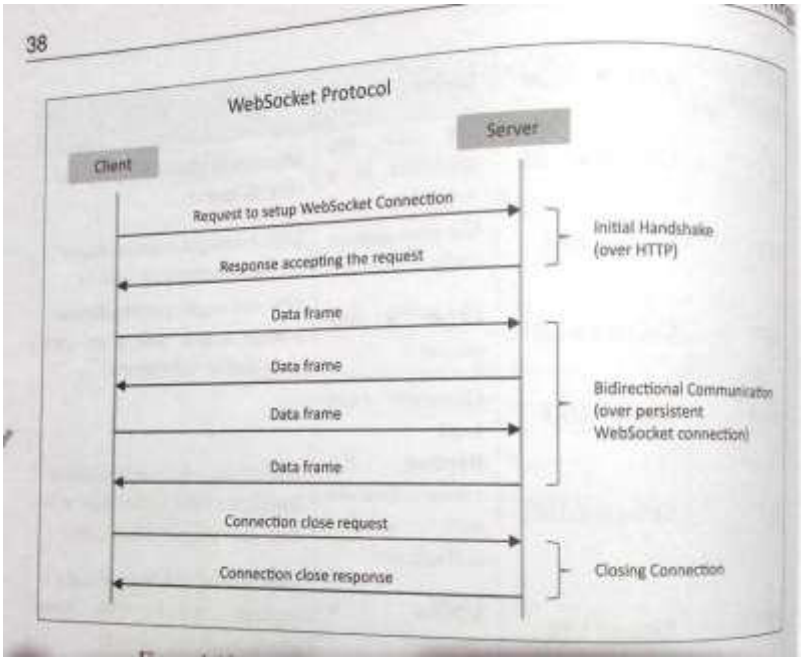


The Request-Response model used by REST:

RESTful web service is a collection of resources which are represented by URIs. RESTful web API has a base URI(e.g: http://example.com/api/tasks/). The clients and requests to these URIs using the methods defined by the HTTP protocol(e.g: GET, PUT, POST or DELETE). A RESTful web service can support various internet media types.

ii) WebSocket Based Communication APIs

WebSocket APIs allow bi-directional, full duplex communication between clients and servers. WebSocket APIs follow the exclusive pair communication model.



5.6 Amazon Web Services for IoT

i) Amazon EC2

In this example, a connection to EC2 service is fi rst established by calling boto.ec2.connect_to_region.

• The EC2 region, AWS access key and AWS secret key are passed to this function. After connecting to EC2 , a new instance is launched using the conn.run_instances function.

• The AMI-ID, instance type, EC2 key handle and security group are passed to this function.

ii) Amazon AutoScaling

AutoScaling Service

• A connection to AutoScaling service is first established by calling boto.ec2.autoscale.connect_to_region function.

• ***Launch Configuration***

• After connecting to AutoScaling service, a new launch configuration is created by calling conn.create_launch_con f iguration. Launch configuration contains instructions on how to launch new instances including the AMI-ID, instance type, security groups, etc.

AutoScaling Group

• After creating a launch configuration, it is then associated with a new AutoScaling group. AutoScaling group is created by calling conn.create_auto_scaling_group. The settings for AutoScaling group such as the maximum and minimum number of instances in the group, the launch configuration, availability zones, optional load balancer to use with the group, etc.

AutoScaling Policies

• After creating an AutoScaling group, the policies for scaling up and scaling down are defined.

• In this example, a scale up policy with adjustment type Change In Capacity and scaling_ad justment = 1 is defined.

• Similarly a scale down policy with adjustment type ChangeInCapacity and scaling_ad justment = -1 is defined.

CloudWatch Alarms

• With the scaling policies defined, the next step is to create Amazon CloudWatch alarms that trigger these policies.

• The scale up alarm is defined using the CPUUtilization metric with the Average statistic and threshold greater 70% for a period of 60 sec. The scale up policy created previously is associated with this alarm. This alarm is triggered when the average CPU utilization of the instances in the group becomes greater than 70% for more than 60 seconds.

• The scale down alarm is defined in a similar manner with a threshold less than 50%.

iii) Amazon S3:

• In this example, a connection to S3 service is first established by calling boto.connect_s3 function.

• The upload_to_s3_bucket_path function uploads the file to the S3 bucket specified at the specified path.

iv) Amazon RDS

In this example, a connection to RDS service is first established by calling boto.rds.connect_to_region function.

• The RDS region, AWS access key and AWS secret key are passed to this function.

• After connecting to RDS service, the conn.create_dbinstance function is called to launch a new RDS instance.

• The input parameters to this function include the instance ID, database size, instance type, database username, database password, database port, database engine (e.g. MySQL5.1), database name, security groups, etc.

v) Amazon Dynamo DB

In this example, a connection to DynamoDB service is first established by calling boto.dynamodb.connect_to_region.

• After connecting to DynamoDB service, a schema for the new table is created by calling conn.create_schema.

• The schema includes the hash key and range key names and types.

• A DynamoDB table is then created by calling conn.create_table function with the table schema, read units and write units as input parameters.

5.7 SkyNetIoT Messaging Platform.

SkyNet is running on a dozen Amazon EC2 servers and has nearly 50,000 registered smart devices including: Arduinos, Sparks, Raspberry Pis, Intel Galileos, and BeagleBoards, Matthieu said. SkyNet runs as an IoT platform-as-a-service (PaaS) as well as a private cloud through Docker, the new lightweight container technology. The platform is written in Node.js and released under an MIT open source license on GitHub.

The single SkyNet API supports the following IoT protocols: HTTP, REST, WebSockets, MQTT (Message Queue Telemetry Transport), and CoAP (Constrained Application Protocol) for guaranteed message delivery and low-bandwidth satellite communications, Matthieu said. Every connected device is assigned a 36 character UUID and secret token that act as the device's strong credentials. Security permissions can be assigned to allow device discoverability, configuration, and messaging.

Part A-Multiple Choice Questions

[Separately discussed]

Part B- 7 Marks

1. Elaborate the WAMP – AutoBahn for IoT.
2. Explain about Xively Cloud for IoT.
3. Draw the Django Architecture with explanation
4. How to Design a RESTful Web API? Explain

Part C- 16 Marks

1. List out the Amazon Web Services for IoT

Reference Books:

1. Jan Holler, VlasiosTsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, ― From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence‖, 1st Edition, Academic Press, 2014.
2. ArshdeepBahga, Vijay Madisetti - Internet of Things: A Hands-On Approach, Universities Press, 2014.
3. The Internet of Things, Enabling technologies and use cases – Pethuru Raj, Anupama C. Raman, CRC Press.
4. Francis daCosta, ―Rethinking the Internet of Things: A Scalable Approach to Connecting Everything‖, 1st Edition, Apress Publications, 2013
5. Cuno Pfister, Getting Started with the Internet of Things, O‟Reilly Media, 2011, ISBN: 9781- 4493- 9357-1
6. DGCA RPAS Guidance Manual, Revision 3 – 2020
7. Building Your Own Drones: A Beginners' Guide to Drones, UAVs, and ROVs, John Baichtal