INTERNET OF THINGS

# LECTURE-1

INTRODUCTION AND CONCEPT OF INTERNET OF THINGS

MODULE-I

Internet of Things (IoT)

Introduction

Internet of Things (IoT) consists of things which have unique identities and they are connected to Internet. Some devices which have already been exist; those are computer networking, 4G-enabled smart mobile phones, have unique identities and are connected to the Internet.

What is Internet of Things (IoT)?

Internet of Things (IoT) is the collection of physical objects, called "things" which are embedded with software, network, & sensors allows the objects for collection and exchange of data. The aim of IoT is to expand the internet connection from a standard device like mobile, computer to relatively small devices.

IoT makes every device "smart," by improving the power of data collection, AI, and networking. IOT allows things for communication & data exchange when executes meaningful information.

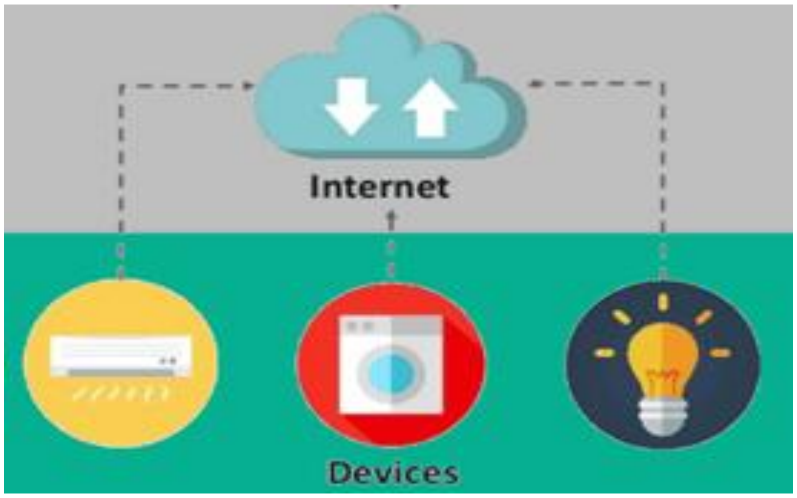
What is IoT?

History of IOT

1970- The idea about connected devices was proposed
1990- John Romkey created a toaster that can be switched on/off through Internet
1995- Cellular module for M2M was introduced by Siemens
1999- Kevin Ashton introduced "Internet of Things" which became widely accepted

2004 – Internet of things was published in famous publications such as the Guardian, Boston Globe, and Scientific American
2005-UN's International Telecommunications Union (ITU) published its first report on Internet of Things.
2008- The Internet of Things was born
2011- One of the research company Gartner consider "The Internet of Things" in their research

How IOT works?



How IoT Works

The whole process begins with the devices themselves like smart phones, smart refrigerator, smart watch, electronic appliances like smart TV, Washing Machine which help us to communicate with IOT platform.

IoT system consists of the following components

1) Sensors/Devices:

Sensors are the devices which collect live data from the environment. These data have various levels of complexities. It may be a temperature monitoring sensor, or may be in the form of video feed. A device has different types of sensors that perform multiple tasks besides sensing. For example, mobile phone consists of multiple sensors like GPS, camera but smart phone is not able to sense these things.

2) Connectivity:

The collected data is sent to a cloud infrastructure. The sensors should be connected to the cloud by different channels of communications. This communication channel includes mobile networks, Bluetooth, WI-FI, WAN, etc.

3) Data Processing:

After data collected, it gets to the cloud; the software that performs processing on these collected data.

4) User Interface:

The information can be send to end-user by triggering alarms on phones or by the notification send through email or text message. The user need an interface which checks their IOT system.

IoT Applications

IoT Applications

IoT applications are given below:

Smart Thermostats

It save the resources on heating bills by knowing the usage patterns.

Connected Cars

IOT automatically helps the automobile companies for handle billing, parking system, insurance, and other related stuff.

Activity Trackers

It can be able to capture the heart rate pattern, calorie expenditure, activity levels, and skin temperature on wrist.

Smart Outlets

Remotely turn any device on or off. It allows a device's for tracking the energy level and send notifications directly to the smart phone.

Parking Sensors

IOT technology helps the users to identify real-time availability of slot for parking spaces on smart phone.

Connect Health

Connected health care means facilitates real-time health monitoring and patient care. It can help for improved medical decision-making based on patient data.

Smart City

Smart city includes traffic management system, rain water drainage & waste management, etc.

Smart home

Smart home refers to the connectivity inside the homes. It consists of smoke detectors, home appliances, light bulbs, windows and door locks, etc.

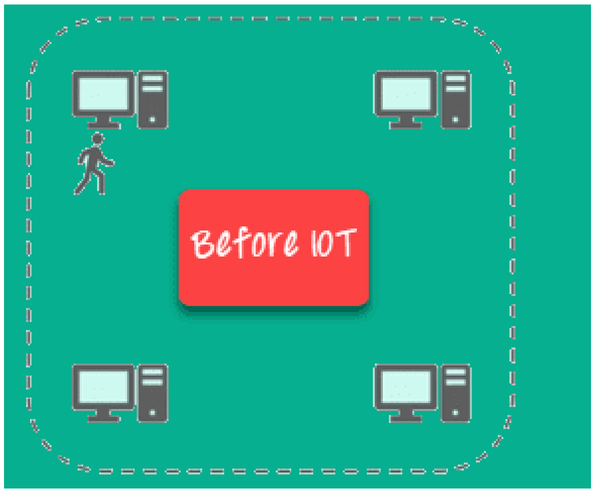
Smart supply chain

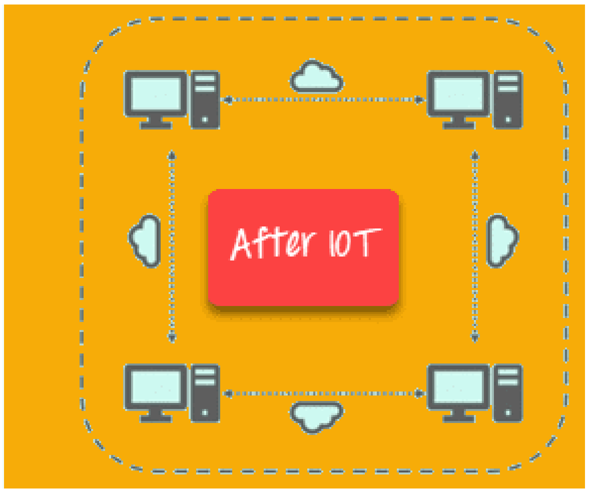
Real time tracking of goods can be identified by IOT when they are on the road, or getting suppliers to exchange the inventory information.

Challenges of IoT

Insufficient testing and updating
Concern regarding data security and privacy
Software complexity
Data volumes and interpretation
Integration with AI and automation
Devices require a constant power supply which is difficult
Interaction and short-range communication

Advantages of IoT





Key benefits of IoT technology are as follows:

Technical Optimization:

IoT helps for improvement of technologies and make them better. Example, with IoT, a car manufacture collects the data from different car sensor and analyzes these data to improve the design and more efficient.

Improved Data Collection:

IoT takes immediate action on data compare to traditional data collection which has its limitation

Reduced Waste:

IoT gives real-time information about data for decision making & management of resources. For example, if a car manufacturer finds some issues in the car engines, then he find the manufacturing plan of the engines and solves with the manufacturing belt.

Improved Customer Engagement:

IoT can improve customer experience by detecting problems by improving the process.

Disadvantages IOT

Security:

IoT technology creates an During the ecosystem of the connected devices, IOT offer authentication control irrespective of sufficient security measures.

Privacy:

IOT technology exposes a substantial amount of personal data, without the user's active participation. Due to this, a lots of privacy issues are created

Flexibility:

Flexibility of the IoT system is mainly due to integrating with another system as there are many diverse systems involved in the process.

Complexity:

IOT system design is very complicated with its deployment and maintenance.

Compliance:

IOT system has its own rules and regulations, because of its complexity, the task of compliance is almost challenging.

IOT Best Practices

Design products for reliability and security
Use strong authentication and security protocols
Disable non-essential services
Ensure Internet-managed, and IoT management hubs & services are secured
Energy efficient algorithms should be designed for the system to be active longer.

Definition of IOT

Network 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 integrated into the information network, and communicate data with each other and their environments.

Characteristics of IOT

- **Dynamic & Self-Adapting:**

IoT devices can dynamically adapt with the changing contexts and take immediate actions by their operating conditions, user's context, or sensed environment. For example, The surveillance cameras can change the modes (to normal or infra-red night modes) by detecting whether it is day or night. Cameras itself change from lower to higher resolution modes if any motion is detected. Here the surveillance system is self adapting based on the context and dynamic conditions.

- **Self-Configuring:**

IoT device can allow a set of large number of devices to work together to provide certain function (weather monitoring). These devices can configure themselves in association with the IoT infrastructure, networking, and fetch software upgrades with user intervention.

- **Interoperable Communication Protocols:**

IoT devices can support a number of interoperable communication protocols which can communicate with other devices and with the infrastructure.

- **Unique Identity:**

IoT device has a unique identity and a unique identifier (IP address or a URI). IoT system have intelligent interfaces which allow the device to communicate with users and the environmental. IoT device allow users to monitor their status, and control remotely, in contest with the control, configuration and management infrastructure.

- **Integrated into Information Network:**

IoT devices are integrated into the information network which allows them to communicate and exchange data with other devices and the systems. IoT devices can be dynamically discovered with the network, For example, a weather monitoring system, it can monitor other devices so that they can communicate and exchange data with each other. Integrated into the information network can help the loT systems "smarter" due to collective information about individual devices in collaboration with the infrastructure. Hence, the aggregated data from large number of connected weather monitoring loT nodes can be analyzed to predict the weather condition

# LECTURE-2

PHYSICAL DESIGN OF IOT; IOT PROTOCOLS, OPEN SYSTEM INTERCONNECTION MODEL (OSI), DATA NETWORKING.

Physical Design of IoT

Things in IoT

An IoT device has a no of interfaces for connections with the other devices, both wired and wireless. These are

(1) I/O interfaces for sensors,

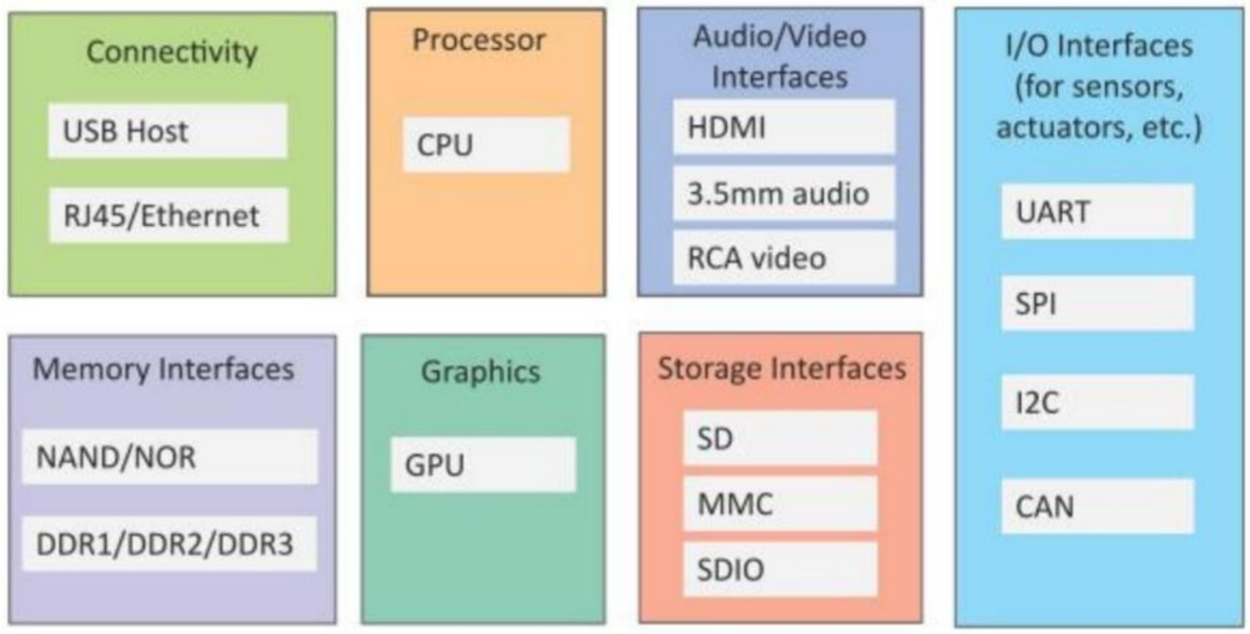
(2) (ii) Internet connectivity,

(3) (iii) memory and storage interfaces and

(4) (iv) audio /video interfaces.

IoT device collects different types of data from sensors, such as temperature, humidity, and light intensity.

These sensed data are communicated to other devices or to the cloud servers.



Block diagram of an IoT Device

IoT Protocols

Link Layer

- The transmission of data over the network's physical layer determine by link layer protocol(e.g., copper wire, coaxial cable, or a radio wave).
- Link layer is the local network connection to which host is attached.
- Hosts can exchange the data packets using link layer protocols.
- Link layer determines how the packets are coded and signaled by the hardware device to which the host is attached (such as a coaxial cable).

802.3 - Ethernet:

- IEEE 802.3 is a collection of wired Ethernet standards for the link layer.
- 802.3 is the standard for 10BASE5 Ethernet that uses coaxial cable as a shared medium, 8
- 02.3.i is the standard for 10BASE-T Ethernet over copper twisted-pair connections,
- 802.3.j is the standard for 10BASE-F Ethernet over fiber optic connections,
- 802.3ae is the standard for 10 Gbit/s Ethernet over fiber. These standards provide data rates from 10 Mb/s to 40 Gb/s and higher.
- The shared medium in Ethernet can be a coaxial cable, twisted-pair wire or an optical fiber.

802.11 - Wi-Fi:

- IEEE 802.11 is a collection of wireless local area network (WLAN) communication standards,. For example,
- 802.11a operates in the 5 GHz band, 802.11b and 802.11g operate in the 2.4 GHz band,
- 802.11n operates in the 2.4/5 GHz bands, 802.11ac operates in the 5 GHz band
- 802.11ad operates in the 60 GHz band. These standards provide data rates from 1 Mb/s to up to 6.75 Gb/s.

802.16 - WiMax:

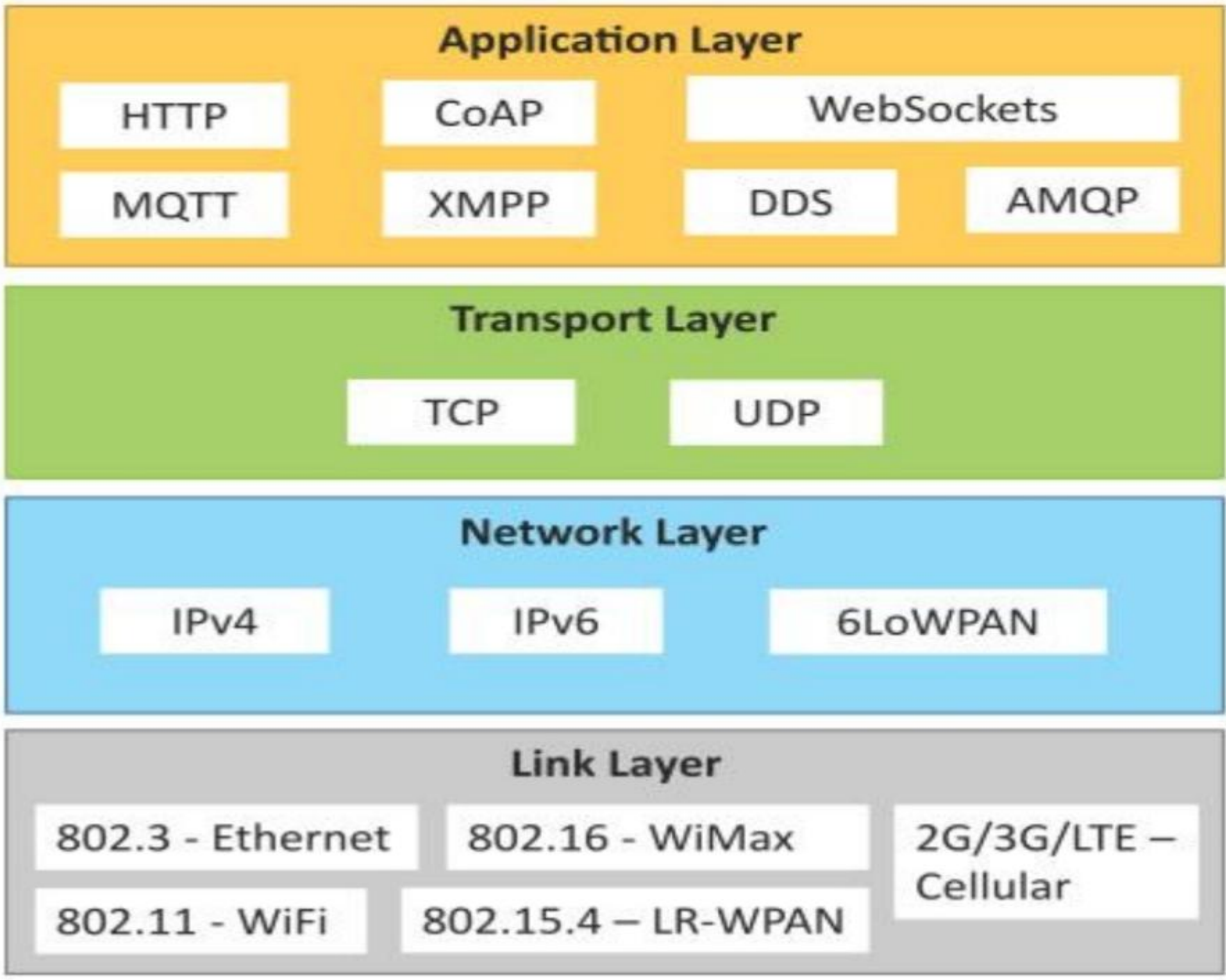
- IEEE 802.16 is a collection of wireless broadband standards, (also called WiMax).
- WiMax standards provide data rates from 1.5 Mb/s to 1 Gb/s

802.15.4 - LR-WPAN:

- IEEE 802.15.4 is the collection of standards for low-rate wireless personal area networks (LR-WPANs).
- These standards create the specifications for high level communication protocols such as ZigBee.
- This provides data rates starting from 40 Kb/s 250 Kb/s. These standards provide low-cost and low-speed communication for power constrained devices.

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

- The generations of mobile communication standards including second generation (2G including GSM and CDMA), third generation (3G - including UMTS and CDMA2000) and fourth generation (4G - including LTE).
- IoT devices having these standards can communicate over cellular networks.
- Data rates for these standards range from 9.6 Kb/s (for 2G) to up to 100 Mb/s (for 4G) and are available from the 3GPP websites.



IoT Protocols

Network/Internet Layer

The network layers are responsible for sending of IP datagram from the source network to the destination network. This layer performs the host addressing and packet routing. The datagram contain the source and destination addresses which are used to route them from the source to destination across multiple networks. Host identification is done using hierarchical IP addressing schemes such as IPv4 or IPv6.

IPv4:

- Internet Protocol version 4 (IPv4) is used to identify the devices on a network using a hierarchical addressing scheme & it is the most deployed internet protocol

- IPv4 uses a 32-bit address scheme which allows total of 2^{32} or 4,294,967,296 addresses. These addresses exhausted in the year 2011 as maximum no of devices are connected to the internet. So IPv4 succeeded by IPv6.
- The IP protocol generates connections on packet networks, but without any guarantee for packets delivery. TCP can give guarantee about delivery and data integrity.

IPv6:

- Internet Protocol version 6 (IPv6) is the latest version of Internet protocol and successor to IPv4.
- IPv6 uses 128-bit address scheme which allows total of 2128 or 3.4 x 1038 addresses.

6LOWPAN:

6LOWPAN (IPv6 over Low power Wireless Personal Area Networks) creates IP protocol for low-power devices which have limited processing capability. 6LOWPAN operates in the 2.4 GHz frequency range and provides data transfer rates of 250 Kb/s. 6LOWPAN works with the 802.15.4 link layer protocol and defines compression mechanisms for IPv6 datagrams over IEEE 802.15.4-based networks.

Transport Layer

The transport layer protocols provide end-to-end message transfer capability independent of the underlying network. The message transfer capability can be set up on connections, either using handshakes (as in TCP) or without handshakes/acknowledgements (as in UDP). The transport layer provides functions such as error control, segmentation, flow control and congestion control.

TCP:

Transmission Control Protocol (TCP) is the most widely used transport layer protocol, that is used by web browsers (along with HTTP, HTTPS application layer protocols), email programs (SMTP application layer protocol) and file transfer (FTP). TCP is a connection oriented and stateful protocol. While IP protocol deals with sending packets, TCP ensures reliable transmission of packets in-order. TCP also provides error detection capability so that duplicate packets can be discarded and lost packets are retransmitted. The flow control capability of TCP ensures that rate at which the sender sends the data is not too high for the receiver to process. The congestion control capability of TCP helps in avoiding network congestion and congestion collapse which can lead to degradation of network performance.

UDP:

Unlike TCP, which requires carrying out an initial setup procedure, UDP is a connectionless protocol. UDP is useful for time-sensitive applications that have very small data units to exchange and do not want the overhead of connection setup. UDP is a transaction oriented and stateless protocol. UDP does not provide guaranteed delivery, ordering of messages and duplicate elimination. Higher levels of protocols can ensure reliable delivery or ensuring connections created are reliable.

Application Layer

Application layer protocols define how the applications interface with the lower layer protocols to send the data over the network. The application data, typically in files, is encoded by the application layer protocol and encapsulated in the transport layer protocol which provides connection or transaction oriented communication over the network. Port numbers are used for application addressing (for example port 80 for HTTP, port 22 for SSH, etc.). Application layer protocols enable process-to-process connections using ports.

HTTP:

Hypertext Transfer Protocol (HTTP) is the application layer protocol that forms the foundation of the World Wide Web (WWW). HTTP includes commands such as GET, PUT, POST, DELETE, HEAD, TRACE, OPTIONS, etc. The protocol follows a request-response model where a client sends requests to a server using the HTTP commands. HTTP is a stateless protocol and each HTTP request is independent of the other requests. An HTTP client can be a browser or an application running on the client (e.g., an application running on an IoT device, a mobile application or other software). HTTP protocol uses Universal Resource Identifiers (URIS) to identify HTTP resources.

COAP:

Constrained Application Protocol (COAP) is an application layer protocol for machine-to-machine (M2M) applications, meant for constrained environments with constrained devices and constrained networks. Like HTTP, COAP is a web transfer protocol and uses a request-response model, however it runs on top of UDP instead of TCP. COAP uses a client-server architecture where clients communicate with servers using connectionless datagrams. CoAP is designed to easily interface with HTTP Like HTTP, COAP supports methods such as GET, PUT, POST, and DELETE.

WebSocket:

WebSocket protocol allows full-duplex communication over a single socket connection for sending messages between client and server. WebSocket is based on TCP and allows streams of messages to be sent back and forth between the client and server while keeping the TCP connection open. The client can be a browser a mobile application or an IoT device.

MQTT:

Message Queue Telemetry Transport (MQTT) is a light-weight messa protocol based on the publish-subscribe model. MQTT uses a client-server architer where the client (such as an IoT device) connects to the server (also called MOTT Broker) and publishes messages to topics on the server. The broker forwards the messages to the clients subscribed to topics. MQTT is well suited for constrained environments where the devices have limited processing and memory resources and the network bandwidth is low.

XMPP:

Extensible Messaging and Presence Protocol (XMPP) is a protocol for real-time communication and streaming XML data between network entities. XMPP powers wide range of applications including messaging, presence, data syndication, gaming, multi-party chat and voice/video calls. XMPP allows sending small chunks of XML data from one network entity to another in near real-time. XMPP is a decentralized protocol and uses a client-server architecture. XMPP supports both client-to-server and server-to-server communication paths. In the context of IoT, XMPP allows real-time communication between IoT devices.

DDS:

Data Distribution Service (DDS) is a data-centric middleware standard for device-to-device or machine-to-machine communication. DDS uses a publish-subscribe model where publishers (e.g. devices that generate data) create topics to which subscribers (e.g., devices that want to consume data) can subscribe. Publisher is an object responsible for data distribution and the subscriber is responsible for receiving published data. DDS provides quality-of-service (QoS) control and configurable reliability.

AMQP:

Advanced Message Queuing Protocol (AMQP) is an open application layer protocol for business messaging. AMQP supports both point-to-point and publisher/subscriber models, routing and queuing. AMQP brokers receive messages from publishers (e.g., devices or applications that generate data) and route them over connections to consumers (applications that process data). Publishers publish messages to exchanges which then distribute message copies to queues. Messages ar either delivered by the broker to the consumers which have subscribed to the queue or the consumers can pull the messages from the queues.

# LECTURE-3

LOGICAL DESIGN OF IOT, FUNCTIONAL BLOCK AND COMMUNICATION MODELS

Logical Design of IoT

Logical design of an IoT system refers to an abstract representation of the entities and processes without going into the low-level specifics of the implementation. In this section the functional blocks of an IoT system and the communication APIs are used.

Iot Functional Blocks

An IoT system comprises of a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication, and management as shown in Figure. These functional blocks are described as follows:

Device:

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

Communication:

The communication block handles the communication for the IoT system.

Services:

An IoT system uses various types of IoT services such as services for device monitoring, device control services, data publishing services and services for device discovery.

Management:

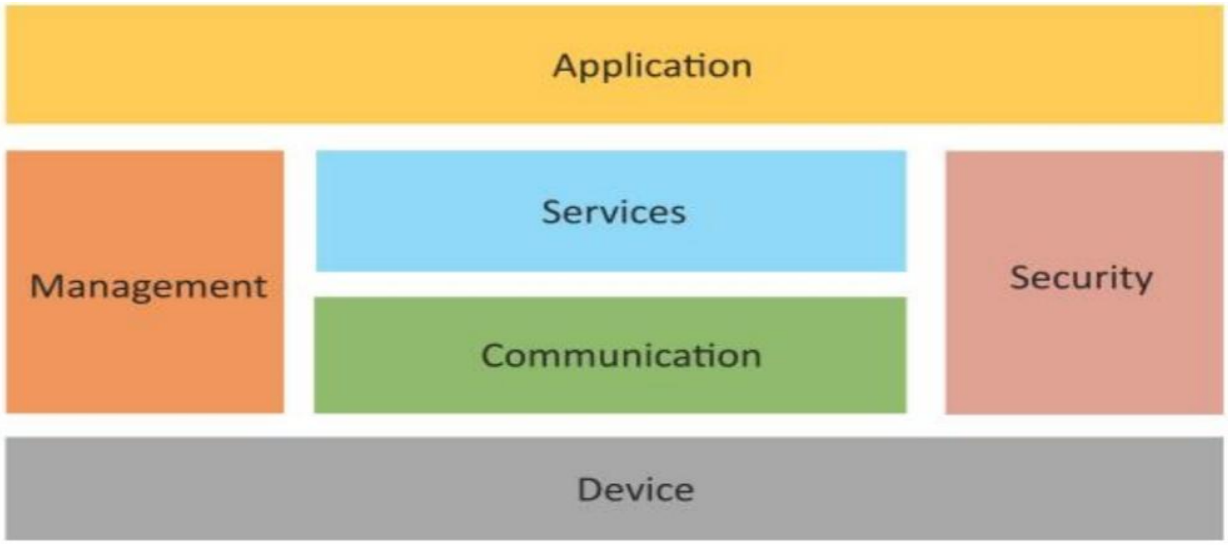
Management functional block provides various functions to govern the IoT system.

Security:

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

Application:

IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view or analyze the processed data.

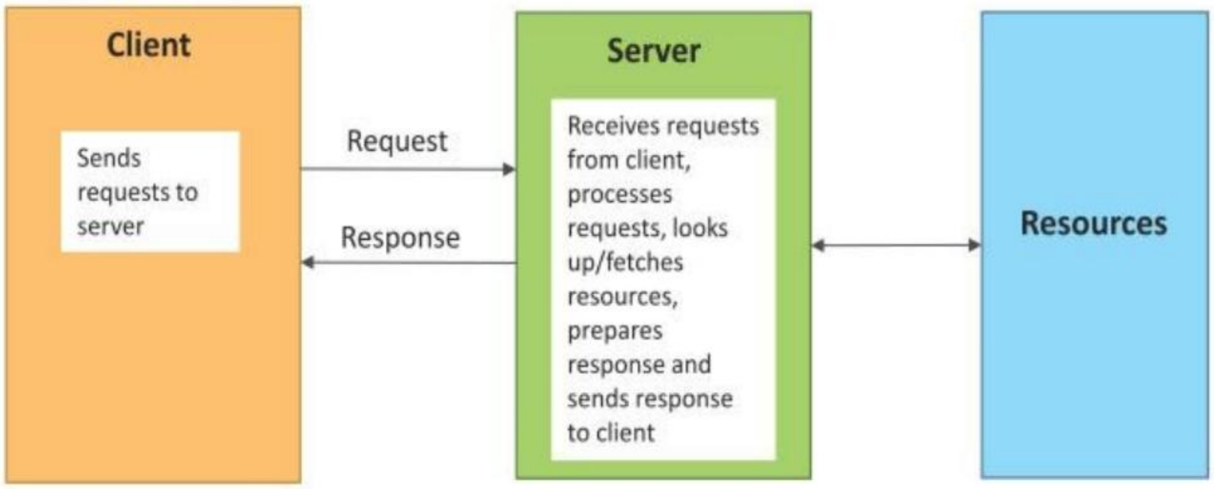


Functional Blocks of IOT

IoT Communication Models

• Request-Response:

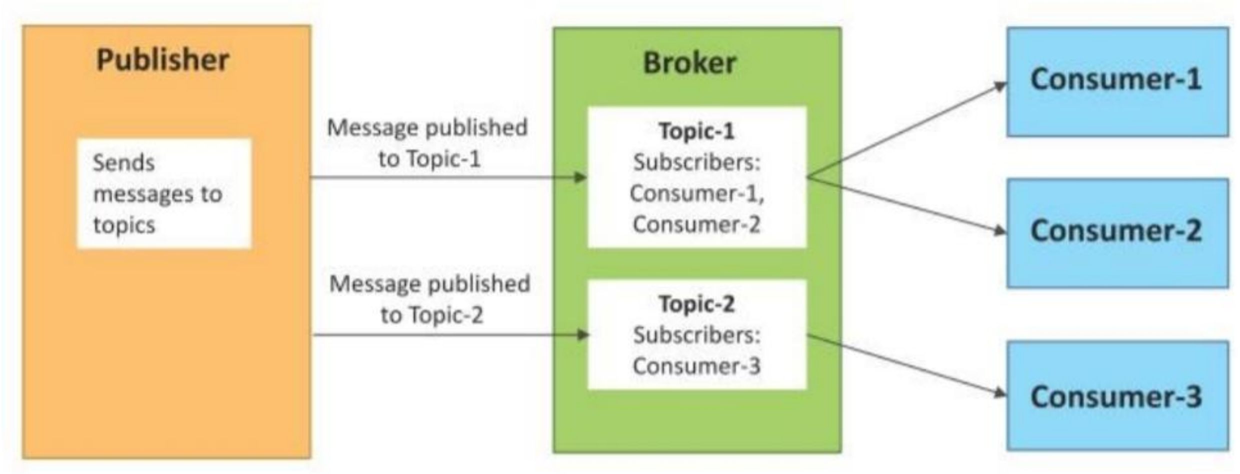
Request-Response is a communication model in which the client sends requests to the server and the server responds to the requests. When the server receives a request, it decides how to respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to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


Client
Sends requests to server
Request
Response
Server
Receives requests from client, processes requests, looks up/fetches resources, prepares response and sends response to client
Resources

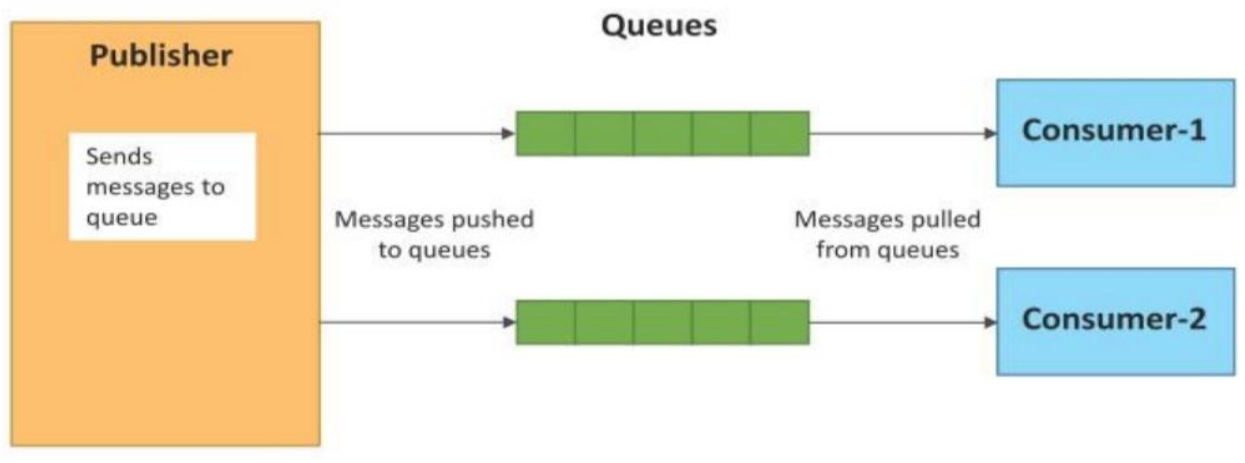
Request-Response communication model

Publish-Subscribe:

Publish-Subscribe is a communication model that involves publishers, brokers and consumers. Publishers are the source of data. Publishers sem the data to the topics which are managed by the broker. Publishers are not aware o the consumers. Consumers subscribe to the topics which are managed by the broke. When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers. Figure shows the publisher-broker-consumer interactions in the publish-subscribe model.



Publish-Subscribe communication model



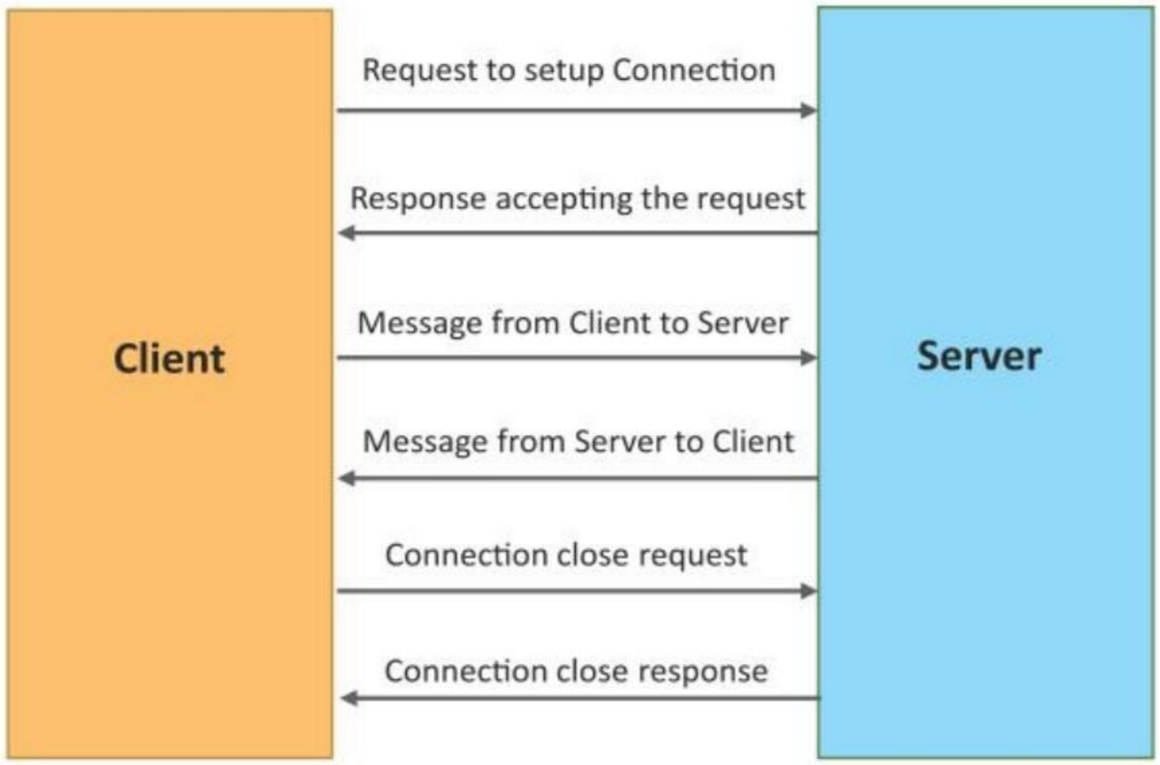
Push-Pull communication model

Push-Pull:

Push-Pull is a communication model in which the data producers push the data to queues and the consumers pull the data from the queues. Producers do not need to be aware of the consumers. Queues help in decoupling the messaging between the producers and consumers. Queues also act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate at which the consumers pull data. Figure shows the publisher-queue-consumer interactions in the push-pull model.

Exclusive Pair:

Exclusive Pair is a bi-directional, fully duplex communication model that uses a persistent connection between the client and server. Once the connection is setup it remains open until the client sends a request to close the connection. Client and server can send messages to each other after connection setup. Exclusive pair is a stateful communication model and the server is aware of all the open connections Figure shows the client-server interactions in the exclusive pair model.


Client
Server
Request to setup Connection
Response accepting the request
Message from Client to Server
Message from Server to Client
Connection close request
Connection close response

Exclusive Pair communication model

IoT Communication APIs

REST-based Communication APIs

Representational State Transfer (REST) is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred. REST APIs follow the request-response communication model described in previous section. The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system. The REST architectural constraints are as follows:

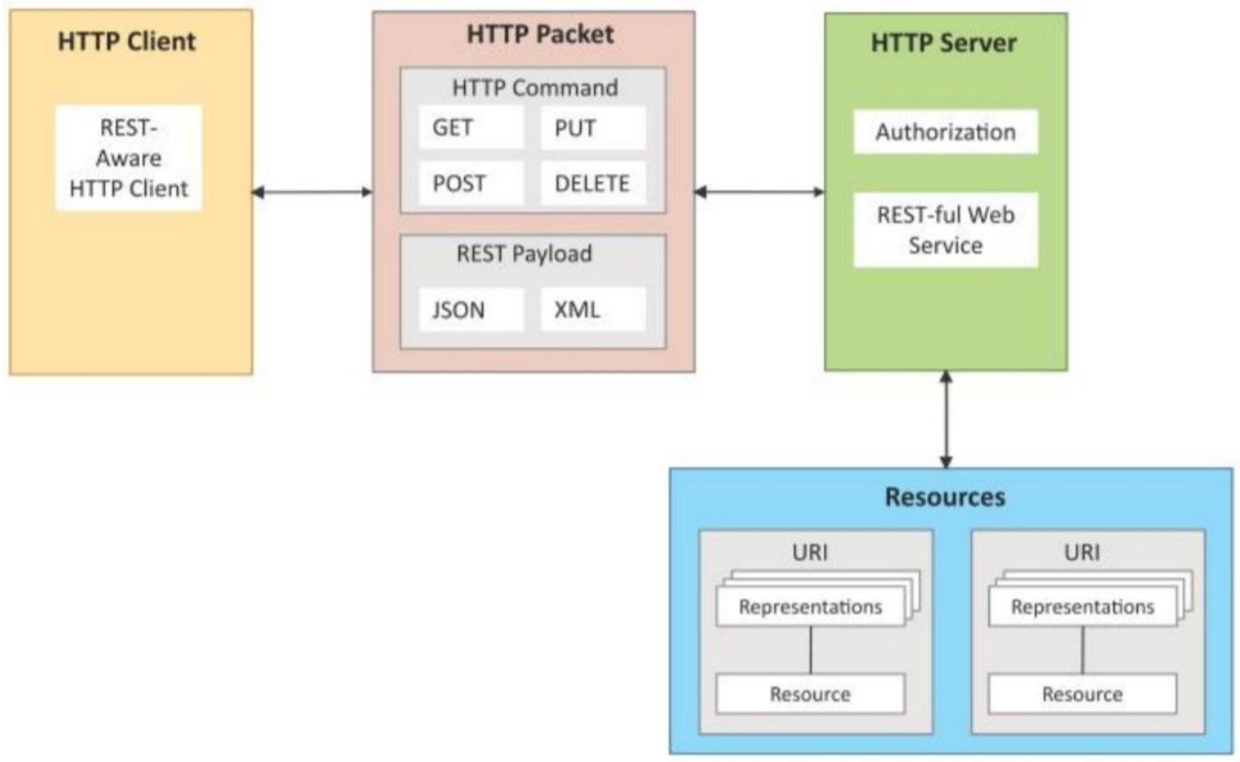


Figure 1: Communication with REST APIs

Client-Server:

The principle behind the client-server constraint is the separation of concerns. For example, clients should not be concerned with the storage of data which is a concern of the server. Similarly, the server should not be concerned about the user interface, which is a concern of the client. Separation allows client and server to be independently developed and updated.

Stateless:

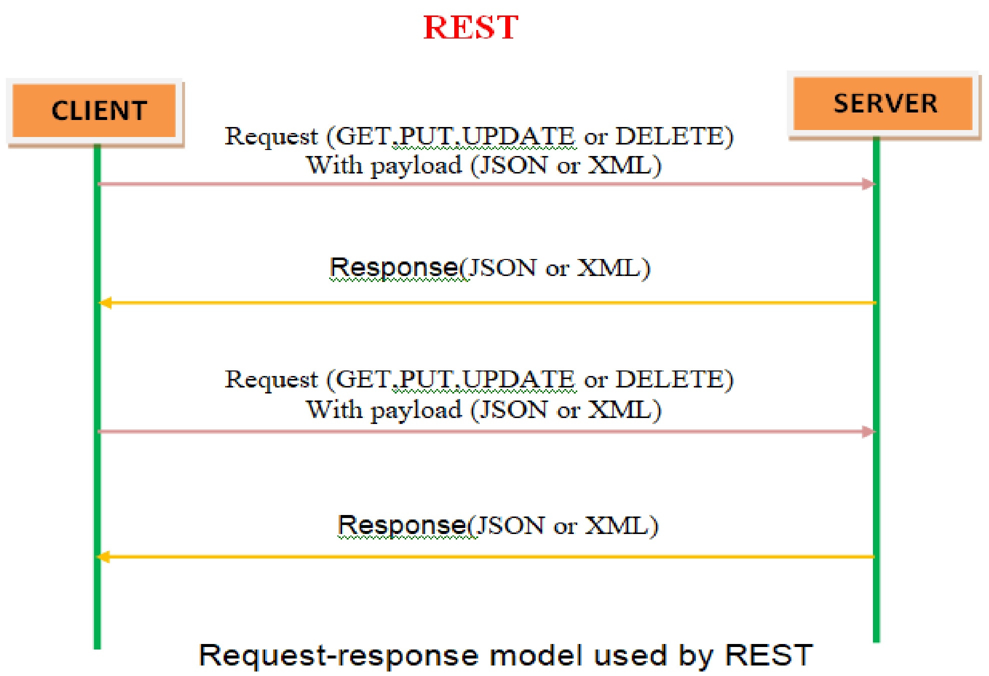
Each request from client to server must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server. The session state is kept entirely on the client.

Cache-able:

Cache constraint requires that the data within a response to a request be implicitly or explicitly labeled as cache-able or non-cache-able. If a response is cache-able, then a client cache is given the right to reuse that response data for later, equivalent requests. Caching can partially or completely eliminate some interactions and improve efficiency and scalability.

Layered System:

Layered system constraint, constrains the behavior of components such that each component cannot see beyond the immediate layer with which they are interacting. For example, a client cannot tell whether it is connected directly to the end server, or to an intermediary along the way. System scalability can be improved by allowing intermediaries to respond to requests instead of the end server, without the client having to do anything different.


REST
CLIENT
SERVER
Request (GET,PUT,UPDATE or DELETE) With payload (JSON or XML)
Response(JSON or XML)
Request (GET,PUT,UPDATE or DELETE) With payload (JSON or XML)
Response(JSON or XML)

Request-response model used by REST

Uniform Interface:

Uniform Interface constraint requires that the method of communication between a client and a server must be uniform. Resources are identified in the requests (by URIs in web based systems) and are themselves separate from the representations of the resources that are returned to the client. When a client holds a representation of a resource it has all the information required to update or delete the resource (provided the client has required permissions). Each message includes enough information to describe how to process the message.

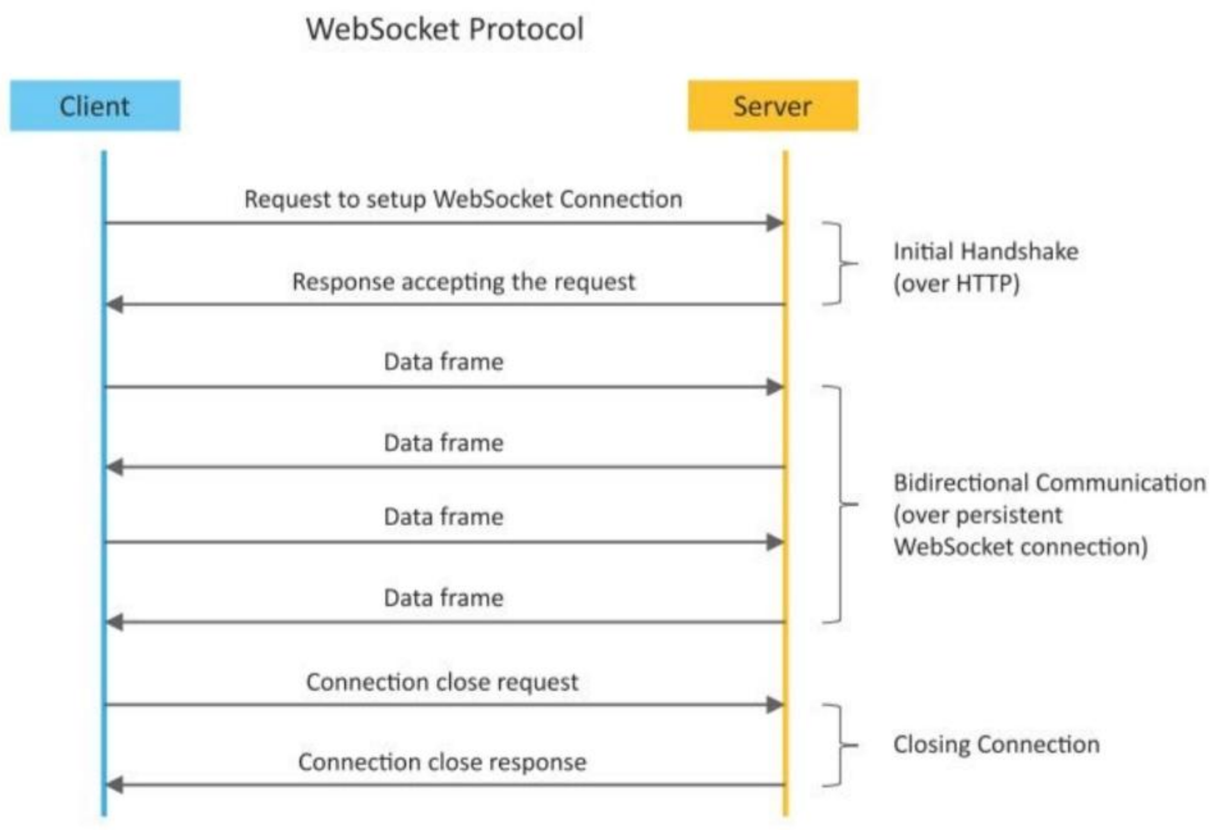
Code on demand:

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

A RESTful web service is a "web API" implemented using HTTP and REST principles. Figure 1 shows the communication between client and server using REST APIs. Figure 2 shows the interactions in 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 send requests to these URIs using the methods defined by the HTTP protocol (e.g., GET, PUT, POST, or DELETE.

WebSocket-based Communication APIs

WebSocket APIs allow bi-directional, full duplex communication between clients and servers. WebSocket APIs follow the exclusive pair communication model n in Figure. Unlike request-response APIs such as REST, the WebSocket APIs allow full duplex communication and do not require a new connection to be setup for each message to be sent. WebSocket communication begins with a connection setup request sent by the client to the server. This request (called a WebSocket handshake) is sent over HTTP and the server interprets it as an upgrade request. If the server supports WebSocket protocol, the server responds to the WebSocket handshake response. After the connection is setup, the client and server can send data/messages to each other in full-duplex mode. WebSocket APIs reduce the network traffic and latency as there is no overhead for connection setup and termination requests for each message. WebSocket is suitable for lol applications that have low latency or high throughput requirements.


WebSocket Protocol
Client
Server
Request to setup WebSocket Connection
Response accepting the request
Initial Handshake (over HTTP)
Data frame
Data frame
Data frame
Data frame
Bidirectional Communication (over persistent WebSocket connection)
Connection close request
Connection close response
Closing Connection

Exclusive pair model used by WebSocket APIs

# LECTURE-4
IOT ENABLING TECHNOLOGIES

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.

Wireless Sensor Networks

A Wireless Sensor Network (WSN) comprises of distributed devices with sensors which are used to monitor the environmental and physical conditions. A WSN consist of a number of end-nodes and routers and a coordinator. End nodes have several sensors attached to them. End nodes can also act as routers. Routers are responsible for routing the data packets from end-nodes to the coordinator. The coordinator collects the data from all the nodes. Coordinator also acts as a gateway that connects the WSN to the Internet. Some examples of WSNs used in IoT systems are described as follows:

- Weather monitoring systems use WSNs in which the nodes collect temperature, humidity and other data, which is aggregated and analyzed.
- Indoor air quality monitoring systems use WSNs to collect data on the indoor air quality and concentration of various gases.
- Soil moisture monitoring systems use WSNs to monitor soil moisture at various locations.
- Surveillance systems use WSNs for collecting surveillance data (such as motion detection data)
- Smart grids use WSNs for monitoring the grid at various points.
- Structural health monitoring systems use WSNs to monitor the health of structures (buildings, bridges) by collecting vibration data from sensor nodes deployed at various points in the structure.

  WSNs are enabled by wireless communication protocols such as IEEE 802.15.4. ZigBee is one of the most popular wireless technologies used by WSNs. ZigBee specifications are based on IEEE 802.15.4. ZigBee operates at 2.4 GHz frequency and offers data rates upto 250 KB/s and range from 10 to 100 meters depending on the power output and environmental conditions. The power of WSNs lies in their ability to deploy large number of low-cost and low-power sensing nodes for continuous monitoring of environmental and physical conditions. WSNs are self-organizing networks. Since WSNs have large number of nodes, manual configuration for each node is not possible. The self-organizing capability of WSN makes the network robust. In the event of failure of some nodes or addition of new nodes to the network, the network can reconfigure itself.

Cloud Computing

Cloud computing is a transformative computing paradigm that involves delivering applications and services over the Internet. Cloud computing involves provisioning of computing, networking and storage resources on demand and providing these resources as metered services to the users, in a "pay as you go" model. Cloud computing resources can be provisioned on-demand by the users, without requiring interactions with the cloud service provider. The process of provisioning resources is automated. Cloud computing resources can be accessed over the network using standard access mechanisms that provide platform-independent access through the use of heterogeneous client platforms such as workstations, laptops, tablets and smart-phones. The computing and storage resources provided by cloud service providers are pooled to serve multiple users using multi-tenancy, Multi-tenant aspects of the cloud allow multiple users to be served by the same physical hardware. Users are assigned virtual resources that run on top of the physical resources.

Cloud computing services are offered to users in different forms

- **Infrastructure-as-a-Service (IaaS):**

  IaaS provides the users the ability to provision computing and storage resources. These resources are provided to the users as virtual machine instances and virtual storage. Users can start, stop, configure and manage the virtual machine instances and virtual storage. Users can deploy operating systems and applications of their choice on the virtual resources provisioned in the cloud. The cloud service provider manages the underlying infrastructure. Virtual resources provisioned by the users are billed based on a pay-per-use paradigm.

- **Platform-as-a-Service (PaaS):**

  PaaS provides the users the ability to develop and deploy application in the cloud using the development tools, application programming interfaces (APIs), software libraries and services provided by the cloud service provider. The cloud service provider manages the underlying cloud infrastructure including servers, network, operating systems and storage. The users, themselves, are responsible for developing, deploying, configuring and managing applications on the cloud infrastructure.

- **Software-as-a-Service (SaaS):**

  SaaS provides the users a complete software application or the user interface to the application itself. The cloud service provider manages the underlying cloud infrastructure including servers, network, operating systems, storage and application software, and the user is unaware of the underlying architecture of the cloud. Applications are provided to the user through a thin client interface (e.g., a browser). SaaS applications are platform independent and can be accessed from various client devices such as workstations, laptop, tablets and smart-phones, running different operating systems. Since the cloud service provider manages both the application and data, the users are able to access the applications from anywhere.

Big Data Analytics

Big data is defined as collections of data sets whose volume, velocity (in terms of its temporal variation), or variety, is so large that it is difficult to store, manage, process and analyze the data using traditional databases and data processing tools. Big data analytics involves several steps starting from data cleansing, data munging (or wrangling), data processing and visualization. Some examples of big data generated by IoT systems are described as follows:

• Sensor data generated by IoT systems such as weather monitoring stations.

• Machine sensor data collected from sensors embedded in industrial and energy systems for monitoring their health and detecting failures.

• Health and fitness data generated by IoT devices such as wearable fitness bands.

• Data generated by IoT systems for location and tracking of vehicles.

• Data generated by retail inventory monitoring systems.

The underlying characteristics of big data include:

Volume:

Though there is no fixed threshold for the volume of data to be considered as big data, however, typically, the term big data is used for massive scale data that is difficult to store, manage and process using traditional databases and data processing architectures. The volumes of data generated by modern IT, industrial, and health-care systems, for example, is growing exponentially driven by the lowering costs of data storage and processing architectures and the need to extract valuable insights from the data to improve business processes, efficiency and service to consumers.

Velocity:

Velocity is another important characteristic of big data and the primary reason for exponential growth of data. Velocity of data refers to how fast the data is generated and how frequently it varies. Modern IT, industrial and other systems are generating data at increasingly higher speeds.

Variety:

Variety refers to the forms of the data. Big data comes in different forms such as structured or unstructured data, including text data, image, audio, video and sensor data.

Communication Protocols

Communication protocols form the backbone of IoT systems and enable network connectivity and coupling to applications. Communication protocols allow devices to exchange data over the network. In section 1.2.2 you learned about various link, network, transport and application layer protocols. These protocols define the data exchange formats, data encoding, addressing schemes for devices and routing of packets from source to destination. Other functions of the protocols include sequence control (that helps in ordering packets determining lost packets), flow control (that helps in controlling the rate at which the sender is sending the data so that the receiver or the network is not overwhelmed) and retransmission of lost packets.

Embedded Systems

An Embedded System is a computer system that has computer hardware and sa embedded to perform specific tasks. In contrast to general purpose computers or per computers (PCs) which can perform various types of tasks, embedded systems are designed to perform a specific set of tasks. Key components of an embedded system include microprocessor or microcontroller, memory (RAM, ROM, cache), networking units (Ether WiFi adapters), input/output units (display, keyboard, etc.) and storage (such as flash memory). Some embedded systems have specialized processors such as digital signal processors (DSPs), graphics processors and application specific processors. Embedded systems run embedded operating systems such as real-time operating systems (RTOS) Embedded systems range from low-cost miniaturized devices such as digital watches to devices such as digital cameras, point of sale terminals, vending machines, appliances (such as washing machines), etc

# LECTURE-5
IOT LEVELS AND DEPLOYMENT TEMPLATES

IoT Levels & Deployment Templates

This section defines various levels of IoT systems with increasing completely. An IoT system comprises of the following components:

- **Device:** An IoT device allows identification, remote sensing, actuating and remote monitoring capabilities.
- **Resource:** Resources are software components on the IoT device for accessing, processing, and storing sensor information, or controlling actuators connected to the device. Resources also include the software components that enable network access for the device.
- **Controller Service:** Controller service is a native service that runs on the device and interacts with the web services. Controller service sends data from the device to the web service and receives commands from the application (via web services) for controlling the device.
- **Database:** Database can be either local or in the cloud and stores the data generated by the IoT device.
- **Web Service:** Web services serve as a link between the IoT device, application, database and analysis components. Web service can be either implemented using HTTP and REST principles (REST service) or using WebSocket protocol (WebSocket service).

  A comparison of REST and WebSocket is provided below:

1. **Stateless/Stateful:**

   REST services are stateless in nature. Each request contains all the information needed to process it. Requests are independent of each other. WebSocket on the other hand is stateful in nature where the server maintains the state and is aware of all the open connections.

2. **Uni-directional/Bi-directional:**

   REST services operate over HTTP and are uni-directional. Request is always sent by a client and the server responds to the requests. On the other hand, WebSocket is a bi-directional protocol and allows both client and server to send messages to each other.

3. **Request Response/Full Duplex:**

   REST services follow a request-response communication model where the client sends requests and the server responds to the requests. WebSocket on the other hand allow full-duplex communication between the client and server, i.e., both client and server can send messages to each other independently.

4. **TCP Connections:**

   For REST services, each HTTP request involves setting up a new TCP connection. WebSocket on the other hand involves a single TCP connection over which the client and server communicate in a full-duplex mode.

5. **Header Overhead:**

   REST services operate over HTTP, and each request is independent of others. Thus each request carries HTTP headers which is an overhead. Due the overhead of HTTP headers, REST is not suitable for real-time applications. WebSocket on the other hand does not involve overhead of headers. After the initial handshake (that happens over HTTP), the client and server exchange messages with minimal frame information. Thus WebSocket is suitable for real-time applications.

6. **Scalability:**

   Scalability is easier in the case of REST services as requests are independent and no state information needs to be maintained by the server. Thus both horizontal (scaling-out) and vertical scaling (scaling-up) solutions are possible for REST services. For WebSockets, horizontal scaling can be cumbersome due to the stateful nature of the communication. Since the server maintains the state of a connection, vertical scaling is easier for WebSockets than horizontal scaling.

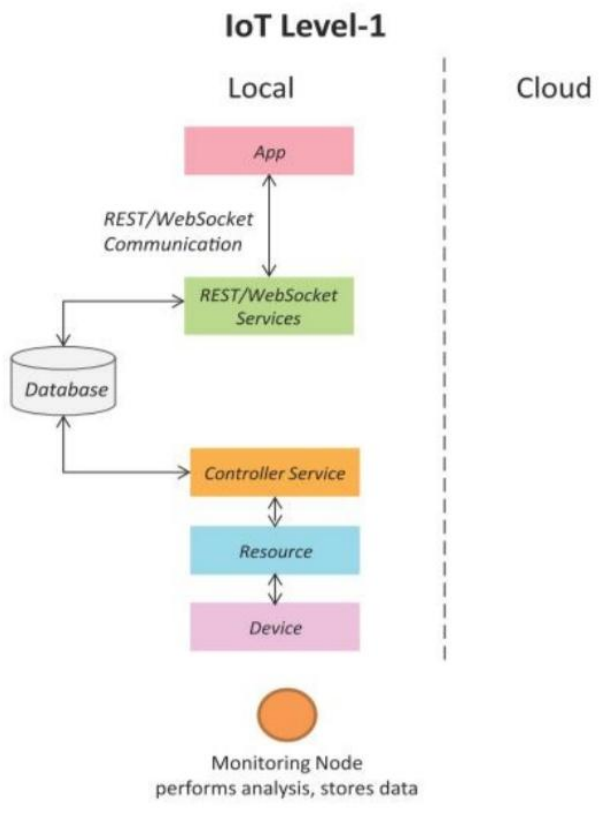
- Analysis Component: The Analysis Component is responsible for analyzing the IoT data and generate results in a form which are easy for the user to understand. Analysis of IoT data can be performed either locally or in the cloud. Analyzed results are stored in the local or cloud databases.

- Application: IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view the processed data.

IoT Level-1

A level-1 IoT system has a single node/device that performs sensing and/or actuations data, performs analysis and hosts the application as shown in Figure Level systems are suitable for modeling low-cost and low-complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive.

Let us now consider an example of a level-1 IoT system for home automation. The system consists of a single node that allows controlling the lights and appliances in a home remotely. The device used in this system interfaces with the lights and appliances usine electronic relay switches. The status information of each light or appliance is maintained in a local database. REST services deployed locally allow retrieving and updating the state of each light or appliance in the status database.



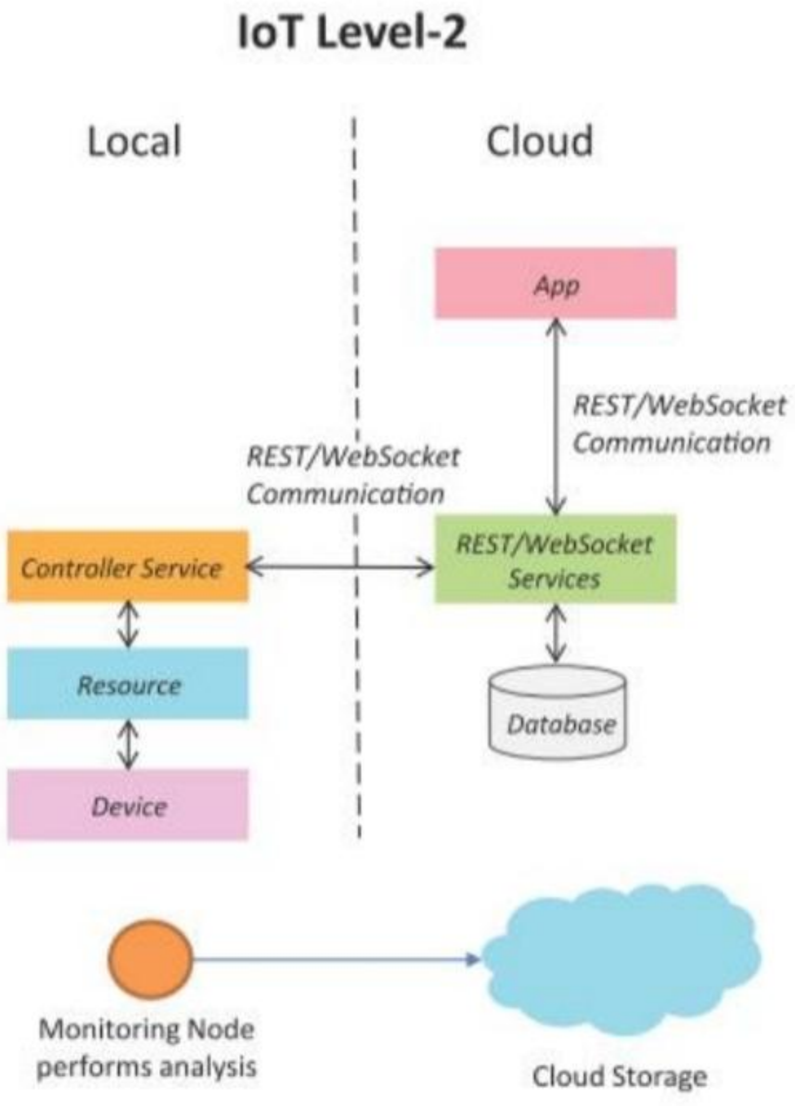
IoT Level-1

The controller service continuously monitors the state of each light or appliance (by retrieving state from the database) and triggers the relay switches accordingly.

The application which is deployed locally has a user interface for controlling the lights or appliances. Since the device is connected to the Internet, the application can be accessed remotely as well.

IoT Level-2

A level-2 IoT system has a single node that performs sensing and/or actuation and local analysis as shown in Figure. Data is stored in the cloud and application is usually cloud-based. Level-2 IoT systems are suitable for solutions where the data involved is big, however, the primary analysis requirement is not computationally intensive and can be done locally itself.



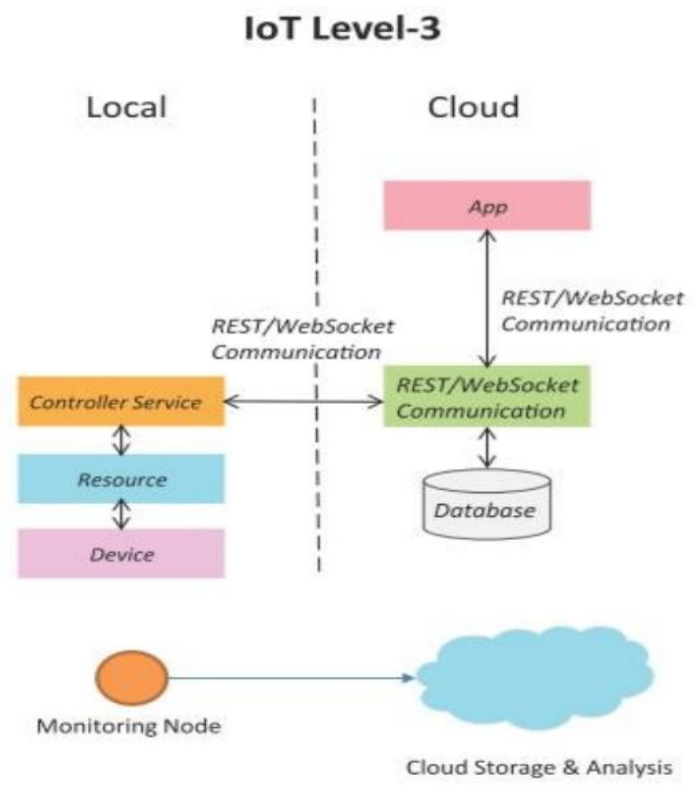
IoT Level-2

Let us consider an example of a level-2 IoT system for smart irrigation. The system consists of a single node that monitors the soil moisture level and controls the irrigation system. The device used in this system collects soil moisture data from sensors. The controller service continuously monitors the moisture levels. If the moisture level drops below a threshold, the irrigation system is turned on. For controlling the irrigation system actuators such as solenoid valves can be used. The controller also sends the moisture data the computing cloud. A cloud-based REST web service is used for storing and retrieving moisture data which is stored in the cloud database. A cloud-based application is used 10 visualizing the moisture levels over a period of time, which can help in making decision about irrigation schedules.

# LECTURE-6
IOT LEVELS AND DEPLOYMENT TEMPLATES CONT.

IoT Level-3

A level-3 IoT system has a single node. Data is stored and analyzed in the cloud application is cloud-based as shown in Figure.Level-3 IoT systems are suitable for solutions where the data involved is big and the analysis requirements are computationally intensive.



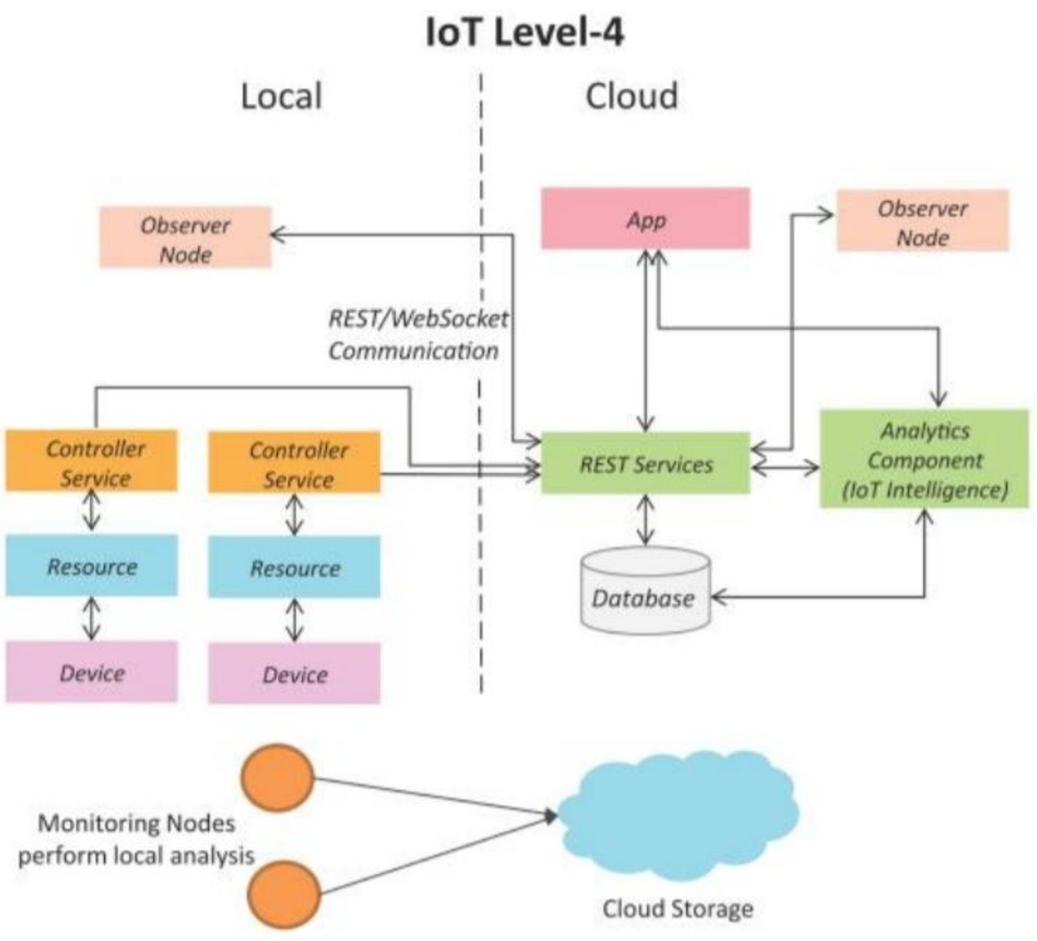
IoT Level-3

Let us consider an example of a level-2 IoT system for tracking package handling. The system consists of a single node (for a package) that monitors the vibration levels for a package being shipped. The device in this system uses accelerometer and gyroscope sensors for monitoring vibration levels. The controller service sends the sensor data to the cloud in real-time using a WebSocket service. The data is stored in the cloud and also visualized using a cloud-based application. The analysis components in the cloud can trigger alerts if the vibration levels become greater than a threshold. The benefit of using WebSocket service instead of REST service in this example is that the sensor data can be sent in real time to the cloud. Moreover, cloud based applications can subscribe to the sensor data feeds for viewing the real-time data.

IoT Level-4

A level 4 IoT system has multiple nodes that perform local analysis. Data is stored in cloud and application is cloud-based as shown in Figure. Level 4 contains local and cloud-based observer nodes which can subscribe to and receive information collected in the cloud from IoT devices. Observer nodes can process information and use it for various applications, however, observer nodes do not perform any control functions. Level-4 IoT systems are suitable for solutions where multiple nodes are required, the data involved is big and the analysis requirements are computationally intensive.

Let us consider an example of a level-4 IoT system for noise monitoring. The system consists of multiple nodes placed in different locations for monitoring noise levels in an area. The nodes in this example are equipped with sound sensors. Nodes are independent of each other. Each node runs its own controller service that sends the data to the cloud. The data is stored in a cloud database. The analysis of data collected from a number of nodes is done in the cloud. A cloud-based application is used for visualizing the aggregated data.

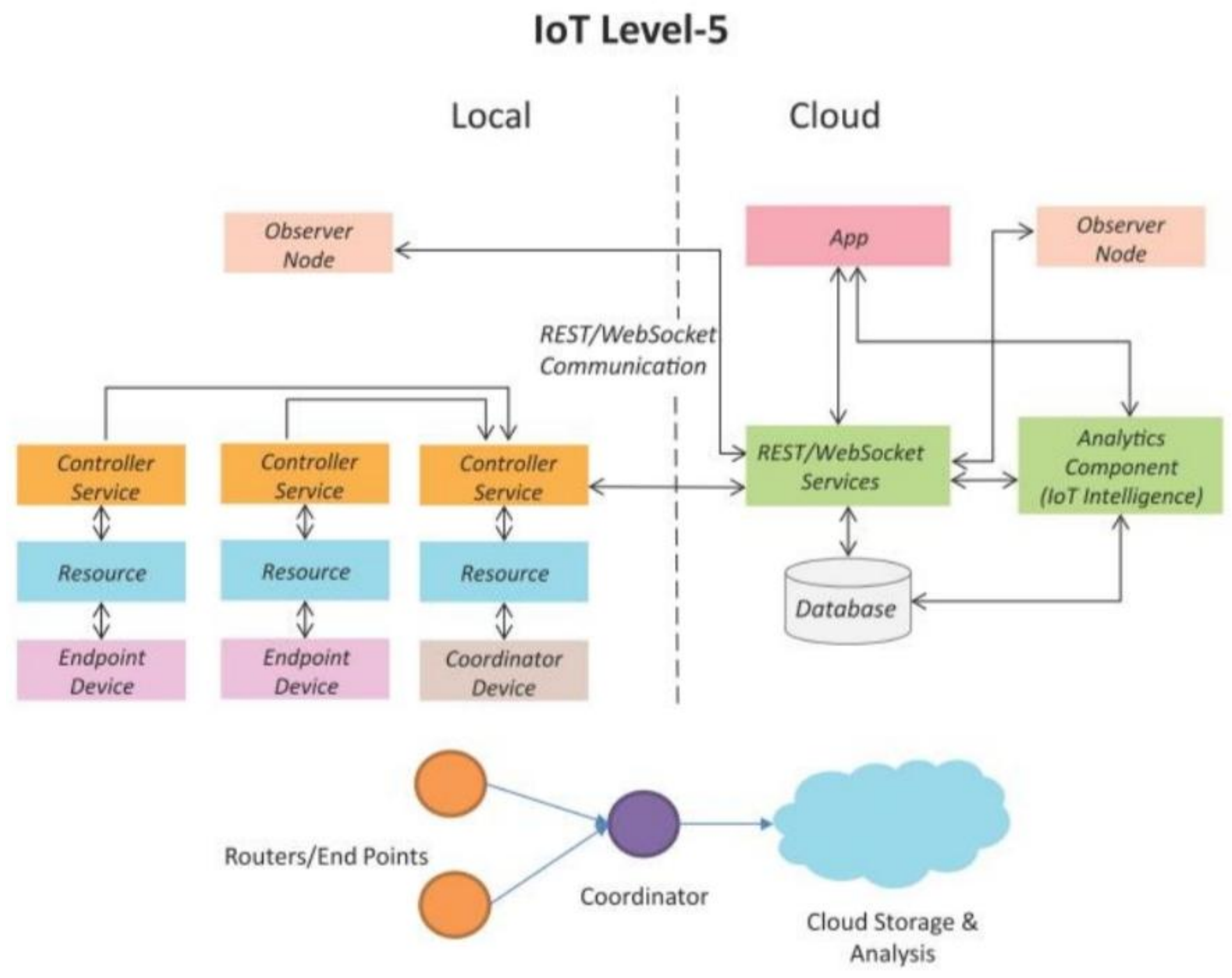


IoT Level-4

IoT Level-5

A level-5 IoT system has multiple end nodes and one coordinator node as shown in Figure. The end nodes that perform sensing and/or actuation. Coordinator node collects data from the end nodes and sends to the cloud. Data is stored and analyzed in the cloud and application is cloud-based. Level-5 IoT systems are suitable for solutions based on wireless sensor networks, in which the data involved is big and the analysis requirements are computationally intensive.

Let us consider an example of a level-5 IoT system for forest fire detection. The system consists of multiple nodes placed in different locations for monitoring temperature, humidity and carbon dioxide (CO2) levels in a forest. The end nodes in this example are equipped with various sensors (such as temperature, humidity and CO2). The coordinator node collects the data from the end nodes and acts as a gateway that provides Internet connectivity to the IoT system. The controller service on the coordinator device sends the collected data to the cloud. The data is stored in a cloud database. The analysis of data is done in the computing cloud to aggregate the data and make predictions. A cloud-based application is used for visualizing the data.

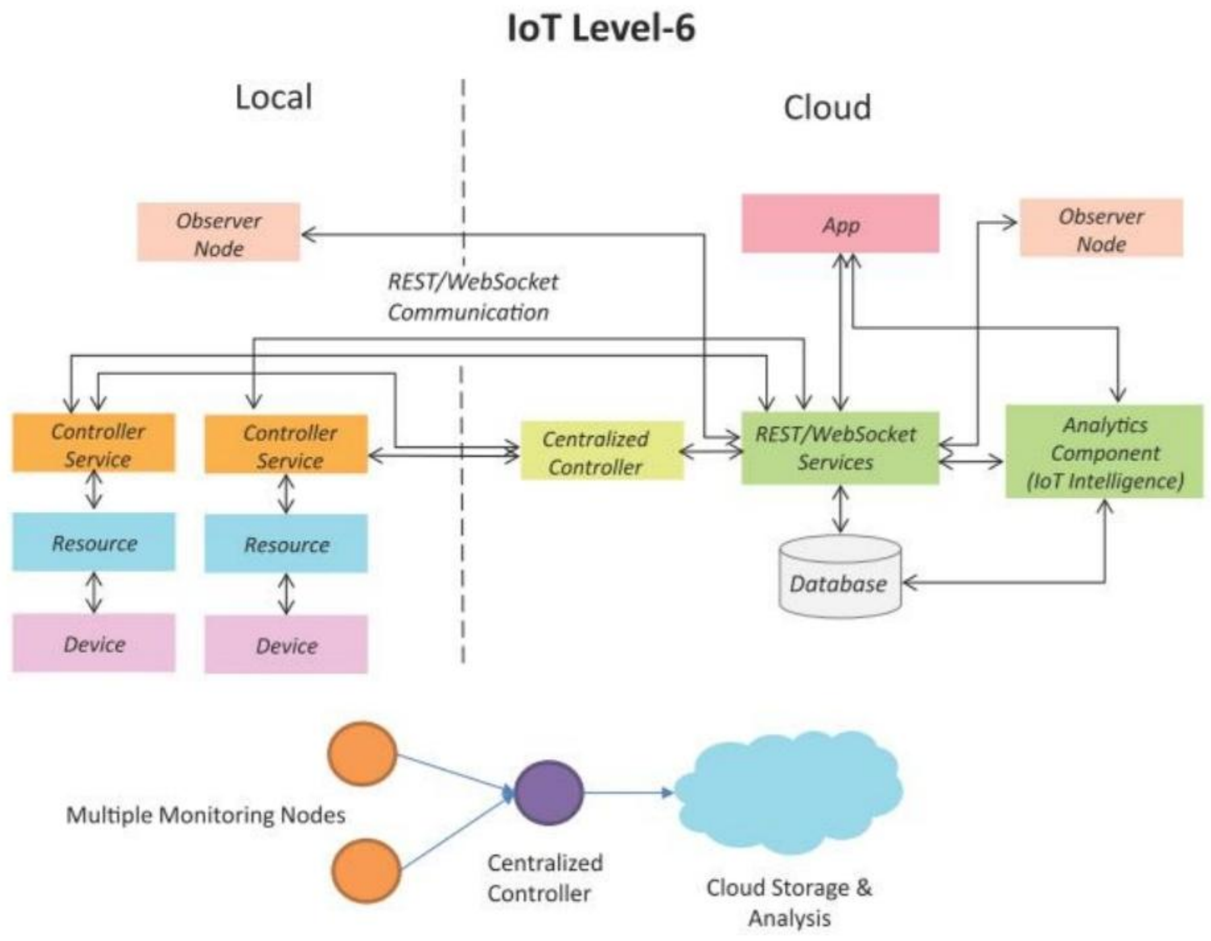


IoT Level-5

IoT Level-6

A level-6 IoT system has multiple independent end nodes that perform sensing and/or actuation and send data to the cloud. Data is stored in the cloud and application is cloud-based as shown in Figure. The analytics component analyzes the data and stores the results in the cloud database. The results are visualized with the cloud-based application. The centralized controller is aware of the status of all the end nodes and sends control commands to the nodes.

Let us consider an example of a level-6 IoT system for weather monitoring system consists of multiple nodes placed in different locations for monitoring temperature humidity and pressure in an area. The end nodes are equipped with various sensors (such temperature, pressure and humidity). The end nodes send the data to the cloud in real-time using a WebSocket service. The data is stored in a cloud database. The analysis of data is done in the cloud to aggregate the data and make predictions. A cloud-based applications used for visualizing the data.

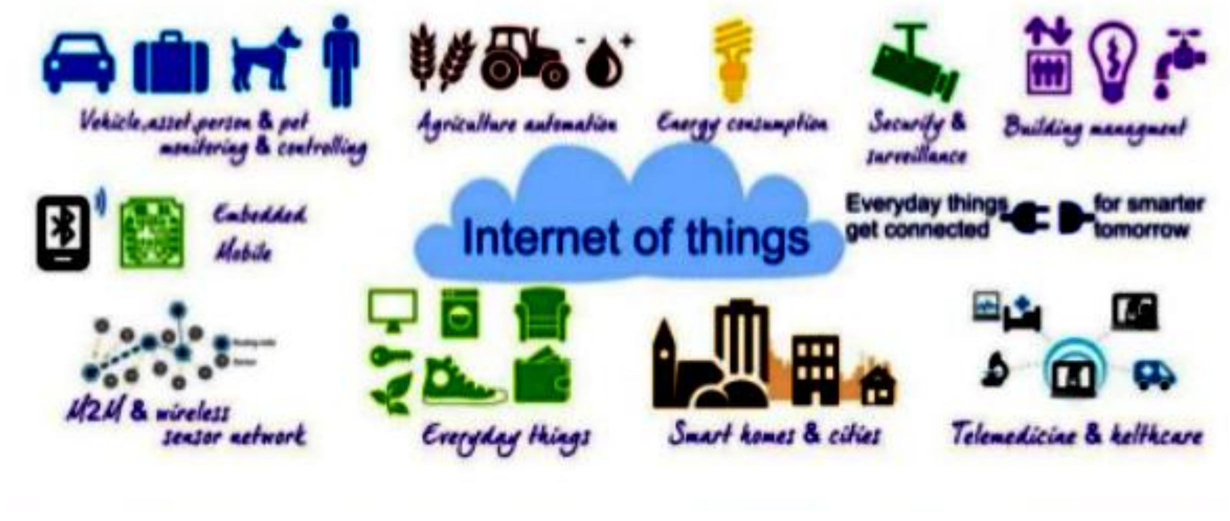


IoT Level-6

# LECTURE-7
DOMAIN SPECIFIC IOTS; HOME AUTOMATION; CITIES

Domain specific IoT

Domain Specific IoTs



Outline IoT Applications for

• Home
• Cities
• Environment
• Energy Systems
• Retail
• Logistics
• Industry
• Agriculture
• Health & Lifestyle

Home Automation IoT applications for smart homes:

Smart Lighting
Smart Appliances
Intrusion Detection
Smoke / Gas Detectors

Smart Lighting

- Smart lighting achieve energy savings by sensing the human movements and their environments and controlling the lights accordingly.
- Key enabling technologies for smart lighting include

  Solid state lighting (such as LED lights)

IP-enabled lights

- Wireless-enabled and Internet connected lights can be controlled remotely from IoT applications such as a mobile or web application.
- LED lighting system that is embedded with ambient intelligence gathered from a distributed smart WSN to optimize and control the lighting System to be more efficient and user-oriented.
- A solid state lighting model is implemented on a wireless sensor network that provide services for sensing illumination changes and dynamically adjusting luminary brightness according to user preferences.

Smart Appliances

- Smart appliances make the management easier and provide status information of appliances to the users remotely. E.g. smart washer/dryer that can be controlled remotely and notify when the washing/drying cycle is complete.
- Open Remote is an open source automation platform for smart home and Build ing that can control various appliances using mobile and web applications.
- It comprises of three components: -

  a Controller -manages scheduling and runtime integration between devices.

  a Designer - allows to create both configuration for the controller and user interface designs.

  Control Panel - allows to interact with devices and control them.

- Smart refrigerators can keep track of the items stored (using RFID tags) and send updates to the user when an item is low on stock. Smart TV allows user to search and stream videos and movies from the internet on a local storage drive, search TV channel schedules and fetch news, weather updates and other content from the internet.

Intrusion Detection

- Home intrusion detection systems use security cameras and sensors to detect intrusions and raise alerts.
- The form of the alerts can be in form:

  - SMS
  - Email
  - Image grab or a short video clip as an email attachment

Smoke / Gas Detectors

- Smoke detectors are installed in homes and buildings to detect smoke that is Typically an early sign of fire.
- It uses optical detection, ionization or air sampling techniques to detect smoke
- The form of the alert can be in form :
- Signals that send to a fire alarm system

- Gas detector can detect the presence of harmful gases such as carbon monoxide (CO), liquid petroleum gas (LPG), etc.

Cities IoT applications for smart cities:

- Smart Parking
- Smart Lighting for Road
- Smart Road
- Structural Health Monitoring
- Surveillance
- Emergency Response

Smart Parking

- Finding the parking space in the crowded city can be time consuming and frustrating
- Smart parking makes the search for parking space easier and convenient for driver.
- It can detect the number of empty parking slots and send the information over the Internet to the smart parking applications which can be accessed by the drivers using their smartphones, tablets, and in car navigation systems.
- Sensors are used for each parking slot to detect whether the slot is empty or not, and this information is aggregated by local controller and then sent over the Internet to database.

Smart Lighting for Roads

- It can help in saving energy
- Smart lighting for roads allows lighting to be dynamically controlled and also adaptive to ambient conditions.
- Smart light connected to the Internet can be controlled remotely to configure lighting schedules and lighting intensity.
- Custom lighting configurations can be set for different situations such as a foggy day, a festival, etc.

Smart Roads

- Smart Roads provides information on driving conditions, travel time estimates and alerts in case of poor driving conditions, traffic congestions and accidents.
- Such information can help in making the roads safer and help in reducing traffic jams.
- Information sensed from the roads can be communicated via internet to cloud-based applications and social media and disseminated to the drivers

who subscribe to such applications.

Structural Health Monitoring

- It uses a network of sensors to monitor the vibration levels in the structures such as bridges and buildings.
- The data collected from these sensors is analyzed to assess the health of the structures.
- By analyzing the data it is possible to detect cracks and mechanical break-downs, locate the damages to a structure and also calculate the remaining life of the structure.
- Using such systems, advance warnings can be given in the case of imminent failure of the structure.

Surveillance

- Surveillance of infrastructure, public transport and events in cities is required to ensure safety and security.
- City wide surveillance infrastructure comprising of large number of distri-buted and Internet connected video surveillance cameras can be created.
- The video feeds from surveillance cameras can be aggregated in cloud-based scalable storage solutions.
- Cloud-based video analytics applications can be developed to search for patterns of specific events from the video feeds.

Emergency Response

- IoT systems can be used for monitoring the critical infrastructure cities such as buildings, gas, and water pipelines, public transport and power substations.
- IoT systems for critical infrastructure monitoring enable aggregation and sharing of information collected from lager number of sensors.
- Using cloud-based architectures, multi-modal information such as sensor data, audio, video feeds can be analyzed I near real-time to detect adverse events.

The alert can be in the form :

Alerts sent to the public
Re-rerouting of traffic
Evacuations of the affected areas

# LECTURE-8
DOMAIN SPECIFIC IOTS; ENVIRONMENT; ENERGY; RETAIL

Environment IoT applications for smart environments:

Weather Monitoring
Air Pollution Monitoring
Noise Pollution Monitoring
Forest Fire Detection
River Flood Detection

Weather Monitoring

- It collects data from a number of sensor attached such as temperature, humidity, pressure, etc and send the data to cloud-based applications and store back-ends.
- The data collected in the cloud can then be analyzed and visualized by cloud-based applications.
- Weather alert can be sent to the subscribed users from such applications.
- AirPi is a weather and air quality monitoring kit capable of recording and uploading information about temperature, humidity, air pressure, light levels, UV levels, carbon monoxide, nitrogen dioxide and smoke level to the Internet.

Air Pollution Monitoring

- IoT based air pollution monitoring system can monitor emission of harmful gases by factories and automobiles using gaseous and meteoro-logical sensors.
- The collected data can be analyzed to make informed decisions on pollutions control approaches.

Noise Pollution Monitoring

- Noise pollution monitoring can help in generating noise maps for cities.
- It can help the policy maker in making policies to control noise levels near residential areas, school and parks.
- It uses a number of noise monitoring stations that are deployed at different places in a city.
- The data on noise levels from the stations is collected on servers or in the cloud and then the collected data is aggregate to generate noise maps.

Forest Fire Detection

- IoT based forest fire detection system use a number of monitoring nodes deployed at different location in a forest.
- Each monitoring node collects measurements on ambient condition including temperature, humidity, light levels, etc.
- Early detection of forest fires can help in minimizing the damage.

River Flood Detection

- IoT based river flood monitoring system uses a number of sensor nodes that monitor the water level using ultrasonic sensors and flow rate using velocity sensors.
- Data from these sensors is aggregated in a server or in the cloud, monitoring applications raise alerts when rapid increase in water level and flow rate is detected.

Energy system **IoT applications for smart energy systems:**

Smart Grid
Renewable Energy Systems
Prognostics

Smart Grids

- Smart grid technology provides predictive information and recommendation s to utilize, their suppliers, and their customers on how best to manage power.
- Smart grid collect the data regarding : - Electricity generation - Electricity consumption - Storage - Distribution and equipment health data
- By analyzing the data on power generation, transmission and consumption of smart grids can improve efficiency throughout the electric system.
- Storage collection and analysis of smarts grids data in the cloud can help in dynamic optimization of system operations, maintenance, and planning.
- Cloud-based monitoring of smart grids data can improve energy usage usage levels via energy feedback to users coupled with real-time pricing information.
- Condition monitoring data collected from power generation and transmission systems can help in detecting faults and predicting outages.

Renewable Energy System

- Due to the variability in the output from renewable energy sources (such as solar and wind), integrating them into the grid can cause grid stability and reliability problems.
- IoT based systems integrated with the transformer at the point of interconnection measure the electrical variables and how much power is fed into the grid
- To ensure the grid stability, one solution is to simply cut off the overproductions.

Prognostics

- IoT based prognostic real-time health management systems can predict performance of machines of energy systems by analyzing the extent of deviation of a system from its normal operating profiles.
- In the system such as power grids, real time information is collected using specialized electrical sensors called Phasor Measurement Units (PMU)
- Analyzing massive amounts of maintenance data collected from sensors in energy systems and equipment can provide predictions for impending failures.
- Open PDC is a set of applications for processing of streaming time-series data collected from Phasor Measurements Units (PMUs) in real-time.

Retail **IoT applications in smart retail systems:**

Inventory Management
Smart Payments
Smart Vending Machines

Inventory Management

- IoT system using Radio Frequency Identification (RFID) tags can help inventory management and maintaining the right inventory levels.
- RFID tags attached to the products allow them to be tracked in the real-time so that the inventory levels can be determined accurately and products which are low on stock can be replenished
- Tracking can be done using RFID readers attached to the retail store shelves or in the warehouse.

Smart Payments

- Smart payments solutions such as contact-less payments powered technologies such as Near field communication (NFC) and Bluetooth.
- NFC is a set of standards for smart-phones and other devices to communicate with each other by bringing them into proximity or by touching them
- Customer can store the credit card information in their NFC-enabled smart-phones and make payments by bringing the smart-phone near the point of sale terminals.
- NFC maybe used in combination with Bluetooth, where NFC initiates initial pairing of devices to establish a Bluetooth connection while the actual data transfer takes place over Bluetooth.

Smart Vending Machines

Smart vending machines connected to the Internet allow remote monitoring of inventory levels, elastic pricing of products, promotions, and contact-less payments using NFC. - Smart-phone applications that communicate with smart vending machines allow user preferences to be remembered and learned with time. E.g: when a user moves from one vending machine to the other and pair the smart-phone, the user preference and favorite product will be saved and then that data is used for predictive maintenance. - Smart vending machines can communicated each others, so if a product out of stock in a machine, the user can be routed to nearest machine - For perishable items, the smart vending machines can reduce the price as the expiry date nears.

# LECTURE-9

DOMAIN SPECIFIC IOTS; LOGISTICS; AGRICULTURE; INDUSTRY

Logistics IoT applications for smart logistic systems:

Fleet Tracking
Shipment Monitoring
Remote Vehicle Diagnostics

Fleet Tracking

- Vehicle fleet tracking systems use GPS technology to track the locations of the vehicles in the real- time.
- Cloud-based fleet tracking systems can be scaled up on demand to handle large number of vehicles.
- The vehicle locations and routers data can be aggregated and analyzed for detecting bottlenecks I the supply chain such as traffic congestions on routes, assignments and generation of alternative routes, and supply chain optimization.

Shipment Monitoring

Shipment monitoring solutions for transportation systems allow monitoring the conditions inside containers. –

E.g : Containers carrying fresh food produce can be monitored to prevent spoilage of food. IoT based shipment monitoring systems use sensors such as temperature, pressure, humidity, for instance, to monitor the conditions inside the containers and send the data to the cloud, where it can be analyzed to detect food spoilage.

Remote Vehicle Diagnostics

It can detect faults in the vehicles or warn of impending faults.

- These diagnostic systems use on-board IoT devices for collecting data on vehicle operation such as speed, engine RPM, coolent temperature, fault code number and status of various vehicle sub- system.
- Modern commercial vehicles support on-board diagnostic (OBD) standard such as OBD-II - OBD systems provide real-time data on the status of vehicle sub-systems and diagnostic trouble codes which allow rapidly identifying the faults in the vehicle.
- IoT based vehicle diagnostic systems can send the vehicle data to centralized servers or the cloud where it can be analyzed to generate alerts and suggest remedial actions.

Agriculture **IoT applications for smart agriculture**:

Smart Irrigation
Green House Control

Smart Irrigation

- Smart irrigation system can improve crop yields while saving water.
- Smart irrigation systems use IoT devices with soil moisture sensors to determined the amount of moisture on the soil and release the flow of the water through the irrigation pipes only when the moisture levels go below a predefined threshold.
- It also collect moisture level measurements on the server on in the cloud where the collected data can be analyzed to plan watering schedules.
- Cultivar's RainCould is a device for smart irrigation that uses water valves, soil sensors, and a WiFi enabled programmable computer. [http://ecultivar.com/rain-cloud-product-project/]

Green House Control

- It controls temperature, humidity, soil, moisture, light, and carbon dioxide level that are monitored by sensors and climatological conditions that are controlled automatically using actuation devices.
- IoT systems play an importance role in green house control and help in improving productivity.
- The data collected from various sensors is stored on centralized servers or in the cloud where analysis is performed to optimize the control strategies and also correlate the productivity with different control strategies.

Industry **IoT applications in smart industry:**

Machine Diagnosis & Prognosis
Indoor Air Quality Monitoring

Machine Diagnosis & Prognosis

- Machine prognosis refers to predicting the performance of machine by analyzing the data on the current operating conditions and how much deviations exist from the normal operating condition.
- Machine diagnosis refers to determining the cause of a machine fault.
- Sensors in machine can monitor the operating conditions such as temperature and vibration levels, sensor data measurements are done on

timescales of few milliseconds to few seconds which leads to generation of massive amount of data.

- Case-based reasoning (CBR) is a commonly used method that finds solutions to new problems based on past experience.
- CBR is an effective technique for problem solving in the fields in which it is hard to establish a quantitative mathematical model, such as machine diagnosis and prognosis.

Air Quality Monitoring

- Harmful and toxic gases such as carbon monoxide (CO), nitrogen monoxide (NO), Nitrogen Dioxide, etc can cause serious health problem of the workers.
- IoT based gas monitoring systems can help in monitoring the indoor air quality using various gas sensors.
- The indoor air quality can be placed for different locations
- Wireless sensor networks based IoT devices can identify the hazardous zones, so that corrective measures can be taken to ensure proper ventilation.

# LECTURE-10
DOMAIN SPECIFIC IOTS; HEALTH & LIFE STYLE

Health & Lifestyle IoT applications in smart health & lifestyle:

Health & Fitness Monitoring
Wearable Electronics

Health & Fitness Monitoring

- Wearable IoT devices allow to continuous monitoring of physiological parameters such as blood pressure, heart rate, body temperature, etc than can help in continuous health and fitness monitoring.
- It can analyze the collected health-care data to determine any health conditions or anomalies.
- The wearable devices may can be in various form such as:
  Belts
  Wrist-bands

Wearable Electronics

- Wearable electronics such as wearable gadgets (smart watch, smart glasses, wristbands, etc) provide various functions and features to assist us in our daily activities and making us lead healthy lifestyles.
- Using the smart watch, the users can search the internet, play audio/video files, make calls, play games, etc.
- Smart glasses allows users to tae photos and record videos, get map directions, check flight status or search internet using voice commands
- Smart shoes can monitor the walking or running speeds and jumps with the help of embedded sensors and be paired with smart-phone to visualize the data.
- Smart wristbands can tract the daily exercise and calories burnt.

Summary

- Internet of Things (IoT) refers to physical and virtual objects that have unique identities and are connected to the Internet. This allows the development of intelligent applicat that make energy, logistics, industrial control, retail, agriculture and many other domains of human endeavour "smarter". IoT allows different types of devices, appliances, users and machines to communicate and exchange data.
- The applications of Internet of Things (IoT) span a wide range of domains including (but not limited to) homes, cities, environment, energy systems, retail, logistics, industry, agriculture and health. Things in IoT refers to IoT devices which have unique identities and allow remote sensing, actuating and remote monitoring capabilities. Almost all IoT devices generate data in some form or the other which when processed by data analytics systems leads to useful information to guide further actions.
- IoT protocols for link, network, transport and application layers. Link layer protocols determine how the data is physically sent over the network. The network/internet layers is responsible for sending of IP datagrams from the source network to the destination network. The transport layer protocols provides end-to-end message transfer capability independent of the underlying network. Application layer protocols define how the applications interface with the lower layer protocols to send the data over the network.
- Functional blocks of an IoT system including device communication, services, management, security and application blocks.
- IoT communication models such as request-response, publish-subscribe, push-pull and exclusive pair.
- REST-based and WebSocket-based communication APIs. REST is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred. A RESTful web service is a web API implemented using HTTP and REST principles. WebSocket APIs allow bi-directional, full duplex communication between clients and servers.

- IOT enabling technologies such as wireless sensor networks, cloud computing, big data analytics, communication protocols and embedded systems.
- IoT levels. A level-1 IoT system has a single node/device that performs sensing and/or actuation, stores data, performs analysis and hosts the application. A level-2 IoT system has a single node that performs sensing and/or actuation and local analysis. A level-3 IoT system has a single node. Data is stored and analyzed in the cloud and application is cloud-based. A level-4 IoT system has multiple nodes that perform local analysis. Data is stored in the cloud and application is cloud-based. A level-5 IoT system has multiple end nodes and one coordinator node. A level-6 IoT system has multiple independent end nodes that perform sensing and/or actuation and send data to the cloud.
- The Internet of Things (IoT) is a network of physical objects or people called "things" that are embedded with software, electronics, network, and sensors which allows these objects to collect and exchange data.
- The actual idea of connected devices was proposed in 1970
- Four Key components of IoT framework are
- 1) Sensors/Devices,
- 2) Connectivity,
- 3) Data Processing,
- 4) User Interface
- Various applications of IoT are Smart Thermostats, Connected Cars, Activity Trackers, Smart Outlets, Connect Health, etc
- Technical Optimization, Improve Data Collection, Reduced Waste, Improved Customer Engagement are key benefits of IoT
- Security, Privacy, Complexity, Compliance, are key challenges of IoT

IoT Platforms Design Methodology

Introduction

IoT systems comprise of multiple components and deployment tiers. In design methodology for IoT system design which is independent of specific product, service or programming language.

IoT systems designed with the proposed methodology have reduced design, testing and maintenance time, better interoperability and reduced complexity. With the proposed methodology, IoT system designers can compare various alternatives for the IoT system components. The methodology based on the IoT-A reference model.

Design Methodology

Figure 1 show the steps involved in the IoT system design methodology. To describe design methodology we have to go for various steps. To explain these steps, we use the example of a smart IoT-based home automation system.

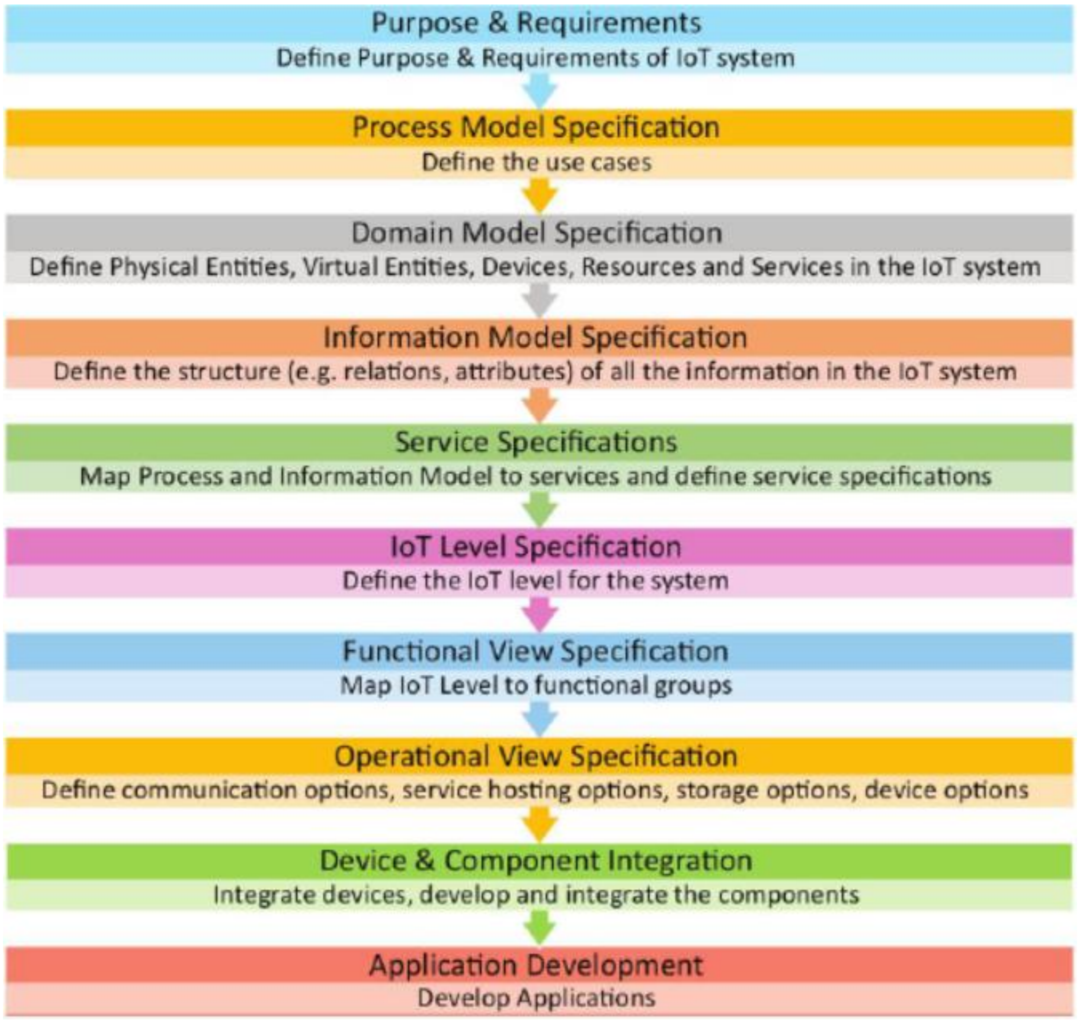

Purpose & Requirements
Define Purpose & Requirements of IoT system
Process Model Specification
Define the use cases
Domain Model Specification
Define Physical Entities, Virtual Entities, Devices, Resources and Services in the IoT system
Information Model Specification
Define the structure (e.g. relations, attributes) of all the information in the IoT system
Service Specifications
Map Process and Information Model to services and define service specifications
IoT Level Specification
Define the IoT level for the system
Functional View Specification
Map IoT Level to functional groups
Operational View Specification
Define communication options, service hosting options, storage options, device options
Device & Component Integration
Integrate devices, develop and integrate the components
Application Development
Develop Applications

Figure 1: Steps involved in IoT system design methodology

Step 1: Purpose & Requirements Specification

The first step in IoT system design methodology is to define the purpose and requirements of the system.

In this step, the system purpose, behavior and requirements (such as data collection requirements, data analysis requirements, system management requirements, data privacy and security requirements, user interface requirements ...) are captured.

Applying this to our example of a smart home automation system, the purpose and requirements for the system may be described as follows:

- **Purpose:** A home automation system that allows controlling of the lights in a home remotely using a web application.
- **Behavior:** The home automation system should have auto and manual modes. In auto mode, the system measures the light level in the room and switches on the light when it gets dark. In manual mode, the system provides the option of manually and remotely switching on/off the light.
- **System Management Requirement:** The system should provide remote monitoring and control functions.
- **Data Analysis Requirement:** The system should perform local analysis of the data.
- **Application Deployment Requirement:** The application should be deployed locally on the device, but should be accessible remotely.
- **Security Requirement:** The system should have basic user authentication capability.

Step 2: Process Specification

The second step in the IoT design methodology is to define the process specification.

In the step, the use cases of the loT system are formally described based on and derived from the purpose and requirement specifications. Figure 2 shows the process diagram for the home automation system.

The process diagram shows the two modes of the system - auto an manual.

In a process diagram, the circle denotes the start of a process, diamond denotes decision box and rectangle denotes a state or attribute.

When the auto mode is chosen, the system monitors the light level. If the light level is low, the system changes the state of the light to "on".if the light level is high, the system changes the state of the light to "off".

When the manual mode is chosen, the system checks the light state set by the user. If the light state set by the user is “on”, the system changes the state of light to "on". Whereas if the light state set by the user is “off”, the system changes the state of light to "off".

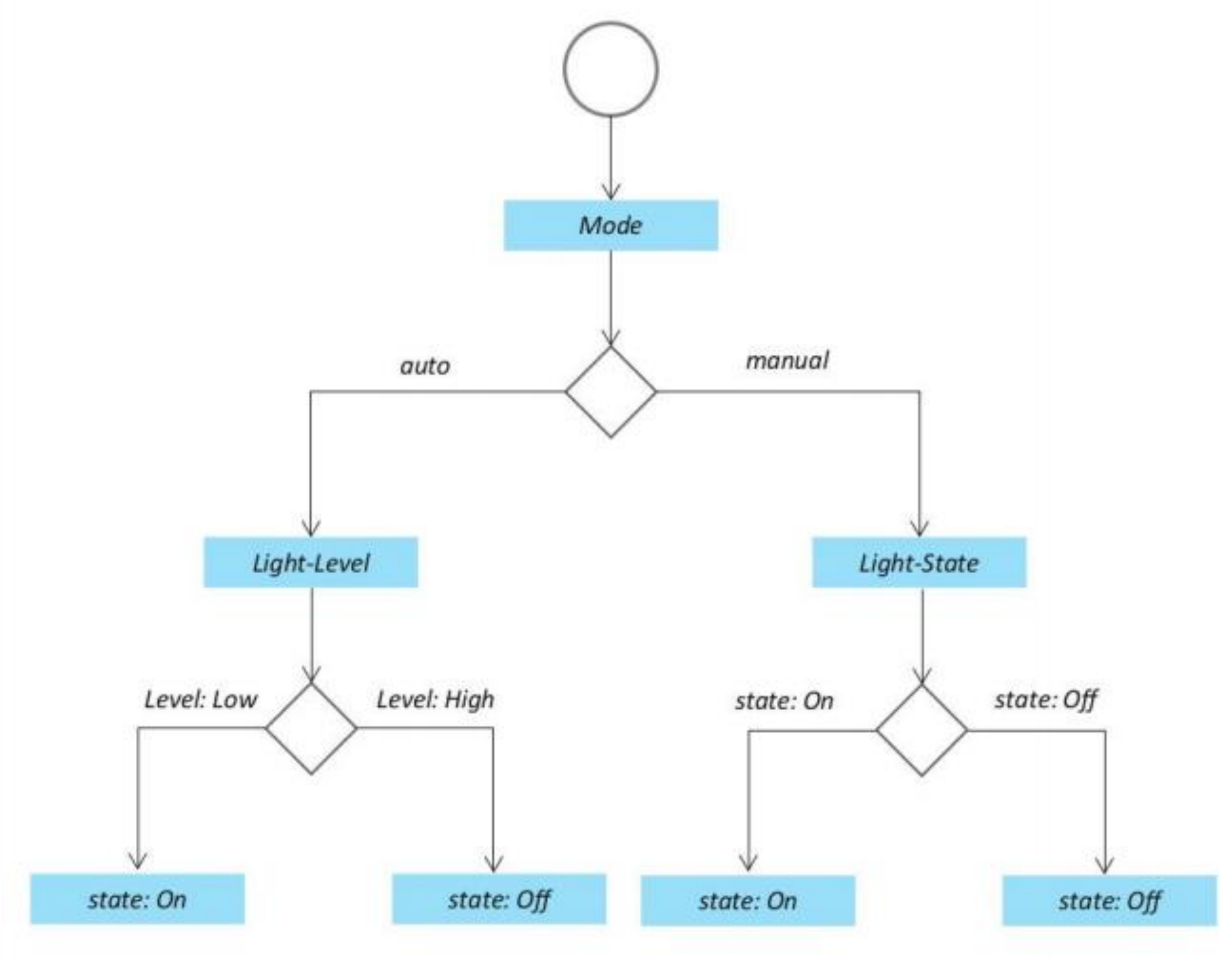


Figure 2: Process specification for home automation IoT system

Step 3: Domain Model Specification

The third step in the IoT design methodology is to define the Domain Model.

The domain model describes the main concepts, entities and objects in the domain of IoT system to be designed.

Domain model defines the attributes of the objects and relationships between objects.

Domain model provides an abstract representation of the concepts, objects and entities in the IoT domain, independent of any specific technology or platform.

With the domain model, the IoT system designers can get an understanding of the IoT domain for which the system is to be designed. Figure 3 shows the domain model for the home automation system example. The entities, objects and concepts defined in the domain model include: .

- **Physical Entity:**

  Physical Entity is a discrete and identifiable entity in the physical environment (e.g. a room, a light, an appliance, a car, etc.). The IoT system provides information about the Physical

Entity (using sensors) or performs actuation upon the Physical Entity (e.g., switching on a light). In the home automation example, there are two Physical Entities involved - one is the room in the home (of which the lighting conditions are to be monitored) and the other is the light appliance to be controlled.

- **Virtual Entity**:

  Virtual Entity is a representation of the Physical Entity in the digital world. For each Physical Entity, there is a Virtual Entity in the domain model. In the home automation example, there is one Virtual Entity for the room to be monitored, another for the appliance to be controlled.

- **Device:**

  Device provides a medium for interactions between Physical Entities and Virtual Entities. Devices are either attached to Physical Entities or placed near Physical Entities. Devices are used to gather information about Physical Entities (e.g., from sensors), perform actuation upon Physical Entities (e.g. using actuators) or used to identify Physical Entities (e.g., using tags). In the home automation example, the device is a single-board mini computer which has light sensor and actuator (relay switch) attached to it.

- **Resource:**

  Resources are software components which can be either "on-device" or "network-resources". On-device resources are hosted on the device and include software components that either provide information on or enable actuation upon the Physical Entity to which the device is attached. Network resources include the software components that are available in network (such as a database). In the home automation example, the on-device resource is the operating system that runs on the single-board minicomputer.

- **Service:**

  Services provide an interface for interacting with the Physical Entity. Services access the resources hosted on the device or the network resources to obtain information about the Physical Entity or perform actuation upon the Physical Entity.

In the home automation example, there are three services:

(1) a service that sets mode to auto or manual, or retrieves the current mode;

(2) a service that sets the light appliance state to on/off, or retrieves the current light state; and

(3) a controller service that runs as a native service on the device.

When in auto mode, the controller service monitors the light level and switches the light on/off and updates the status in the status database. When in manual mode, the controller service retrieves the current state from the database and switches the light on/off. The process of deriving the services from the process specification and information model is described in the later sections.

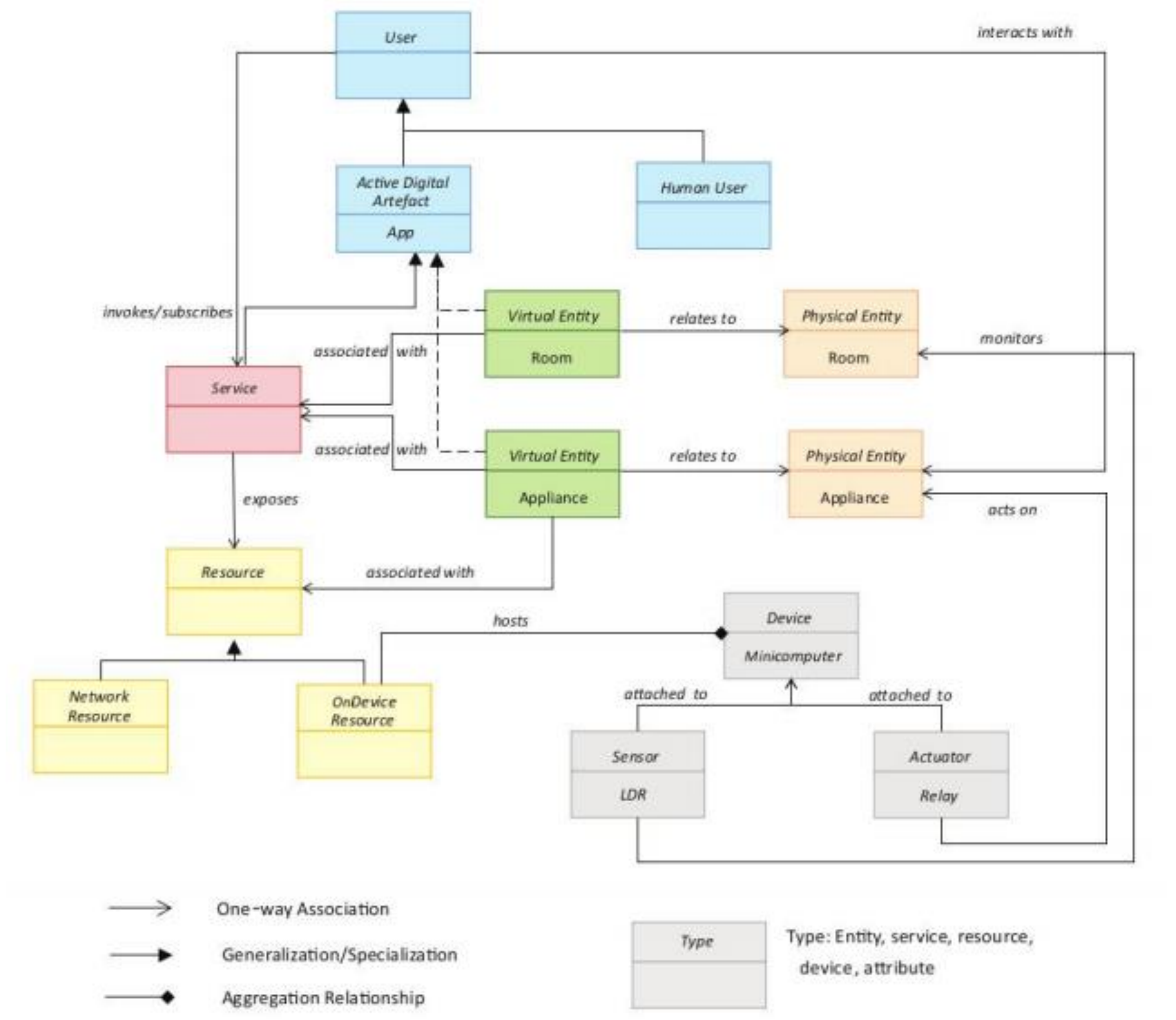


Figure 3: Domain model of the home automation IoT system

Step 4: Information Model Specification

The fourth step in the IoT design methodology is to define the Information Model.

Information Model defines the structure of all the information in the IoT system, for example, attributes of Virtual Entities, relations, etc. Information model does not describe the specifics of how the information is represented or stored.

To define the information model, we first list the Virtual Entities defined in the Domain Model.

Information model adds more details to the Virtual Entities by defining their attributes and relations.

In the home automation example, there are two Virtual Entities - a Virtual Entity for the light appliance (with attribute - light state) and a Virtual Entity for the room (with attribute - light level). Figure 4 shows the Information Model for the home automation system example.

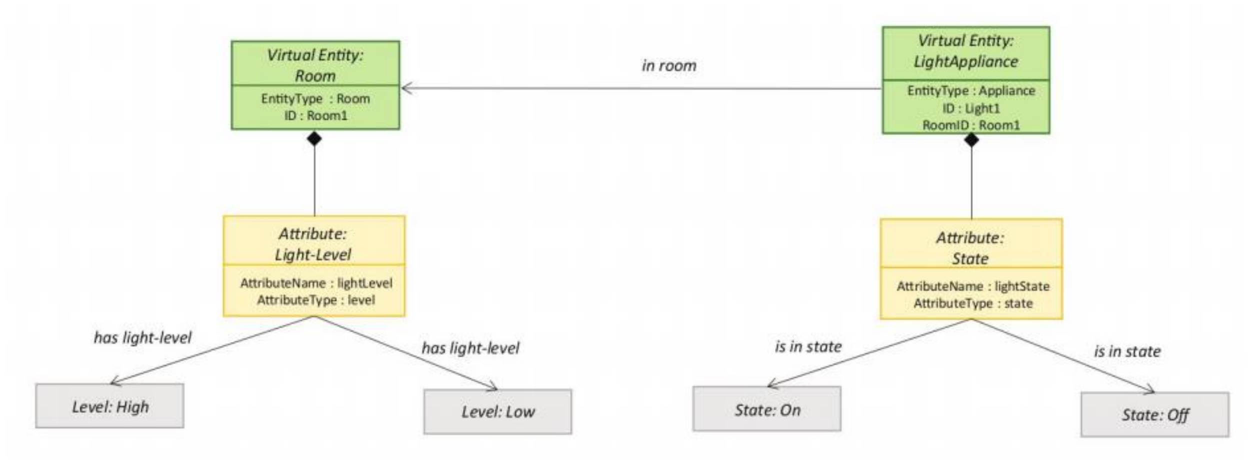


Figure 4: Information model of the home automation IoT system

Step 5: Service Specifications

The fifth step in the IoT design methodology is to define the service specifications.

Service specifications define the services in the IoT system, service types, service inputs/output, service endpoints, service schedules, service preconditions and service effects.

Figure 5 shows an example of deriving the services from the process specification and information model for the home automation IoT system.

From the process specification and information model, we identify the states and attributes. For each state and attribute we define a service. These services either change the state or attribute values or retrieve the current values.

For example, the Mode service sets mode to auto or manual or retrieves the current mode. The State service sets the light appliance state to on/off or retrieves the current light state.

The Controller service monitors the light level in auto mode and switches the light on/off and updates the status in the status database. In manual mode, the controller service, retrieves the current state from the database and switches the light on/off

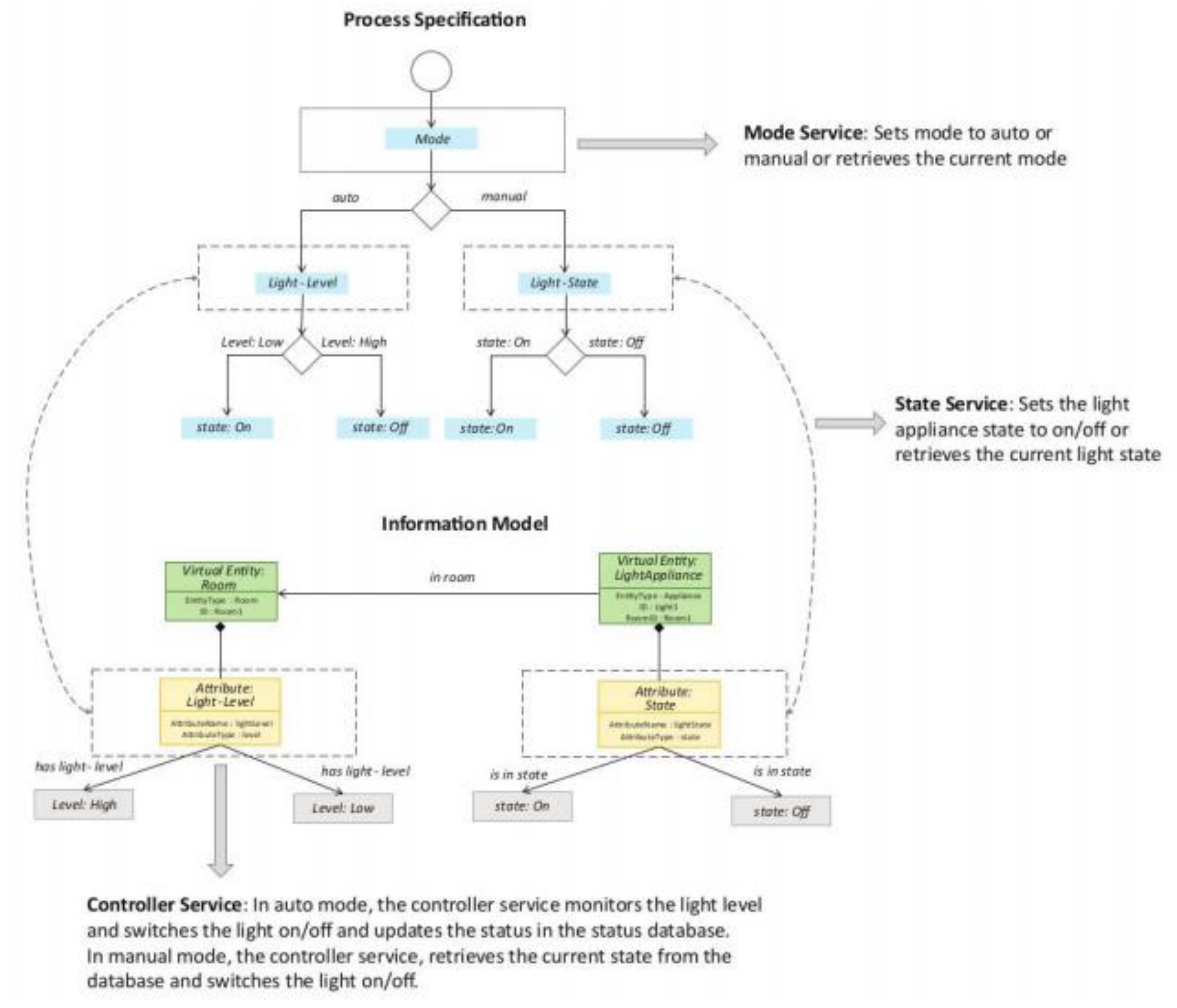

Process Specification
Mode
Mode Service: Sets mode to auto or manual or retrieves the current mode
auto
manual
Light - Level
Light - State
Level: Low
Level: High
state: On
state: Off
state: On
state: Off
state: On
state: Off
State Service: Sets the light appliance state to on/off or retrieves the current light state
Information Model
Virtual Entity: Room
EntityType : Room
ID : Room1
in room
Virtual Entity: LightAppliance
EntityType : Appliance
ID : Light1
RoomID : Room1
Attribute: Light - Level
AttributeName : lightLevel
AttributeType : level
Attribute: State
AttributeName : lightState
AttributeType : state
has light - level
has light - level
Level: High
Level: Low
is in state
is in state
state: On
state: Off
Controller Service: In auto mode, the controller service monitors the light level and switches the light on/off and updates the status in the status database. In manual mode, the controller service, retrieves the current state from the database and switches the light on/off.

Figure 5: Deriving services from process specification and information model for home automation IoT system

Figures 6, 7 and 8 show specifications of the controller, mode and state services the home automation system.

The Mode service is a RESTful web service that sets mode auto or manual (PUT request), or retrieves the current mode (GET request).

The mode updated to/retrieved from the database. The State service is a RESTful web service that see the light appliance state to on/off (PUT request), or retrieves the current light state (GE request).

The state is updated to/retrieved from the status database. The Controller service runs as a native service on the device. When in auto mode, the controller service monitors the light level and switches the light on/off and updates the status in the status database. When in manual mode, the controller service retrieves the current state from the database and switches the light on/off.

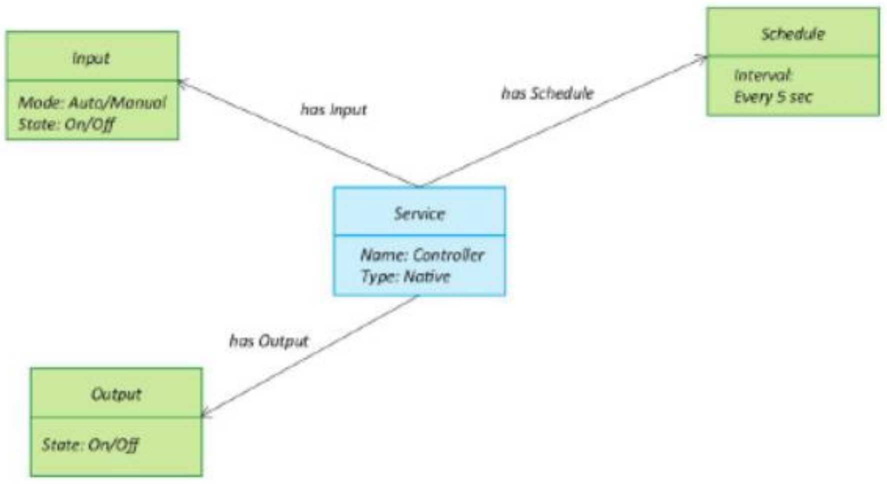


Figure 6: Controller service of the home automation IoT system

Step 6: loT Level Specification

The sixth step in the IoT design methodology is to define the loT level for the system. In module-1, we defined five IoT deployment levels. Figure 9 shows the deployment level of the home automation IoT system, which is level-1.

Step 7: Functional View Specification

The seventh step in the loT design methodology is to define the Functional View.

The Functional View (FV) defines the functions of the loT systems grouped into various Functional Groups (Fs).

Each Functional Group either provides functionalities for interacting with instances of concepts defined in the Domain Model or provides information related to these concepts

The Functional Groups (FG) included in a Functional View include:

- **Device:**

  The device FG contains devices for monitoring and control. In the home automation example, the device FG includes a single board mini-computer, a light sensor and a relay switch (actuator).

- **Communication:**

  The communication FG handles the communication for the IoT system. The communication FG includes the communication protocols that form the backbone of IoT systems and enable network connectivity. The communication FG also includes the communication APIs (such as REST and WebSocket) that are used by the services and applications to exchange data over the network.

  In the home automation example the communication protocols include802.11 (link layer), IPv4/IPv6 (network layer), TCP (transport layer), and HTTP (application layer). The communication API used in the home automation examples is a REST-based API

- **Services:**

  The service FG includes various services involved in the IoT system such as services for device monitoring, device control services, data publishing services and services for device discovery. In the home automation example, there are two REST services (mode and state service) and one native service (controller service).

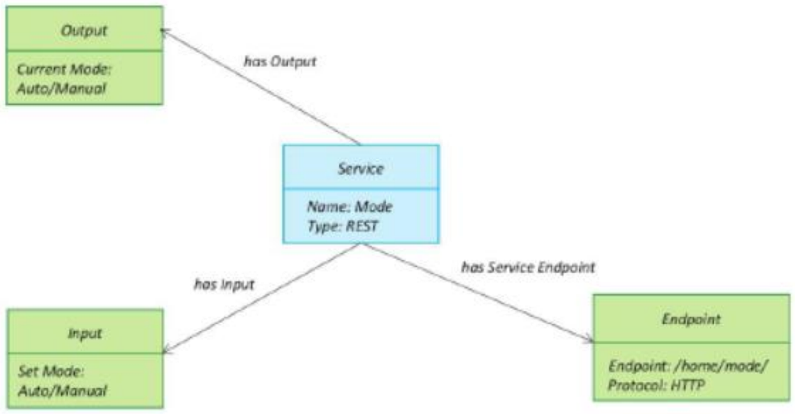


Figure 7: Service specification for home automation IoT system - mode service

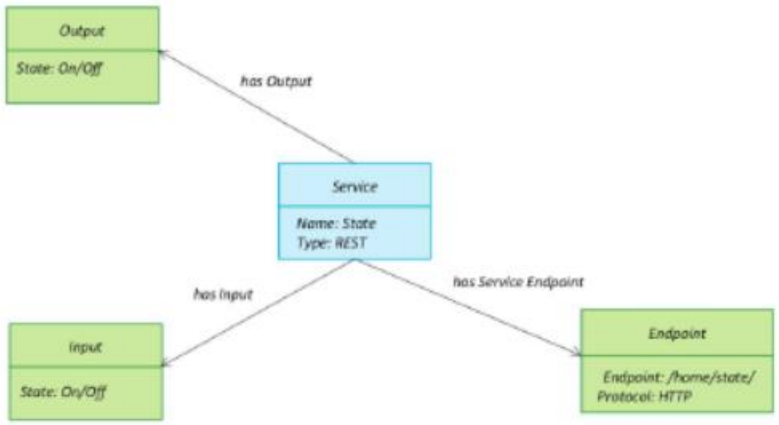


Figure 8: Service specification for home automation IoT system - state service

Sensor and a relay switch (actuator)

- **Management:**

  The management FG includes all functionalities that are needed to configure and manage the IoT system.
- **Security:**

  The security FG includes security mechanisms for the IoT system such as authentication, authorization, data security, etc.

- **Application:**

  The application FG includes applications that provide an interface to the users to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and the processed data.

IoT device maps to the Device FG (sensors, actuators devices, computing devices) and the Management FG (device management). Resources map to the Device FG (on-device resource) and Communication FG (communication APIs and protocols). Controller service maps to the Services FG (native service). Web Services map to Services FG. Database maps to the Management FG (database management) and Security FG (database security). Application maps to the Application FG (web application, application and database servers), Management FG (app management) and Security FG (app security).

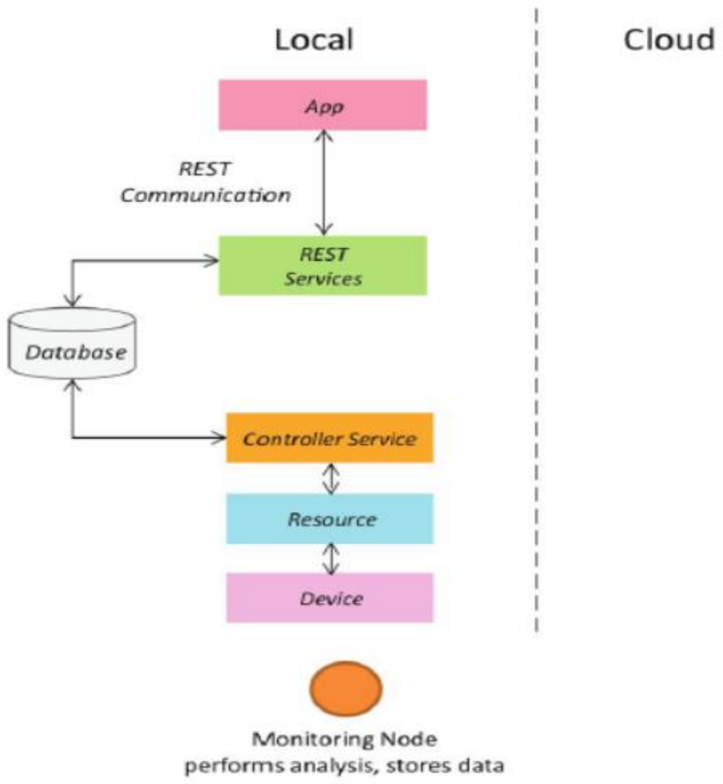


Figure 9: Deployment design of the home automation IoT system

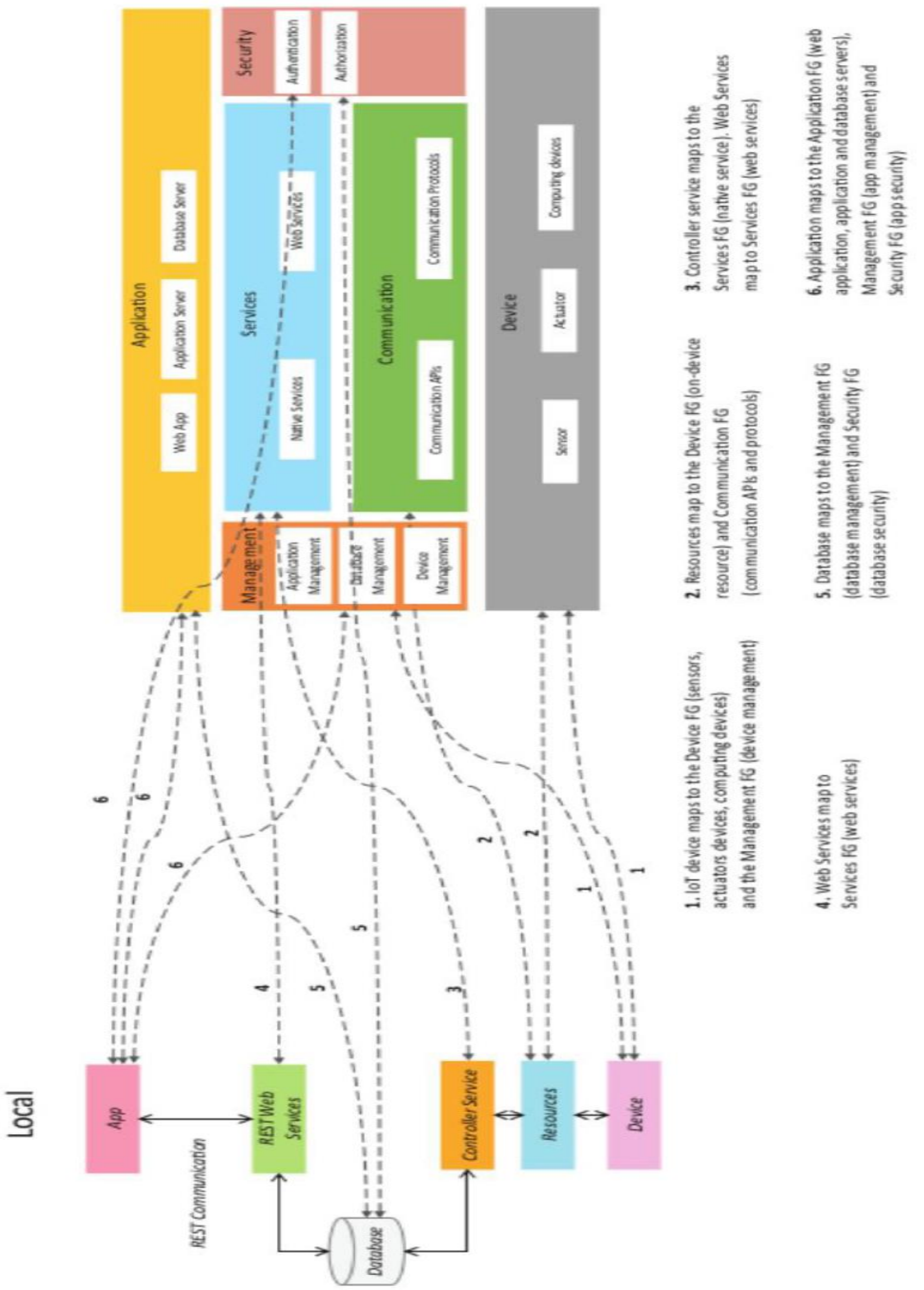

Local
App
REST Web Services
REST Communication
Controller Service
Resources
Device
Database
6
6
6
4
5
5
3
2
2
1
1
Application
Web App
Application Server
Database Server
Management
Application Management
Database Management
Device Management
Services
Native Services
Web Services
Security
Authentication
Authorization
Communication
Communication APIs
Communication Protocols
Device
Sensor
Actuator
Computing devices
1. IoT device maps to the Device FG (sensors, actuators devices, computing devices) and the Management FG (device management)
2. Resources map to the Device FG (on-device resource) and Communication FG (communication APIs and protocols)
3. Controller service maps to the Services FG (native service). Web Services map to Services FG (web services)
4. Web Services map to Services FG (web services)
5. Database maps to the Management FG (database management) and Security FG (database security)
6. Application maps to the Application FG (web application, application and database servers), Management FG (app management) and Security FG (app security)

Figure 10 shows an example of mapping deployment level to functional groups for home automation IoT system.

Step 8: Operational View Specification

The eighth step in the IoT design methodology is to define the Operational View Specifications.

In this step, various options pertaining to the IoT system deployment and operation are defined, such as, service hosting options, storage options, device options, application hosting options, etc.

Figure 11 shows an example of mapping functional groups to operational view specifications for home automation IoT system.

Operational View specifications for the home automation example are as follows:

- **Devices:**

  Computing device (Raspberry Pi), light dependent resistor (sensor), relay switch (actuator).
- Communication APIs:

  REST APIs
- Communication Protocols:

  Link Layer - 802.11, Network Layer - IPv4/IPv6, Transport - TCP, Application - HTTP.
- Services:
  1. Controller Service - Hosted on device, implemented in Python and run as a native service.
  2. Mode service - REST-ful web service, hosted on device, implemented with Django-REST Framework.
  3. State service - REST-ful web service, hosted on device, implemented with Django-REST Framework.
- Application:

  Web Application - Django Web Application,

  Application Server - Django App Server,

  Database Server - MySQL.
- Security:

  Authentication: Web App,

  Database Authorization: Web App, Database
- Management:

  Application Management - Django App Management

  Database Management - MySQL DB Management,

  Device Management - Raspberry Pi device Management.

Step 9: Device & Component Integration

The ninth step in the IoT design methodology is the integration of the devices and components.

Figure 11 shows a schematic diagram of the home automation IoT system.

The devices and components used in this example are Raspberry Pi minicomputer, LDR sensor and relay switch actuator.

Step 10: Application Development

The final step in the IoT design methodology is to develop the IoT application.

Figure 12 shows a screenshot of the home automation web application.

The application has controls for the mode (auto on or auto off) and the light (on or off). In the auto mode, the IoT system controls the light appliance automatically based on the lighting conditions in the room.

When auto mode is enabled the light control in the application is disabled and it reflects the current state of the light. When the auto mode is disabled, the light control is enabled and it is used for manually controlling the light.

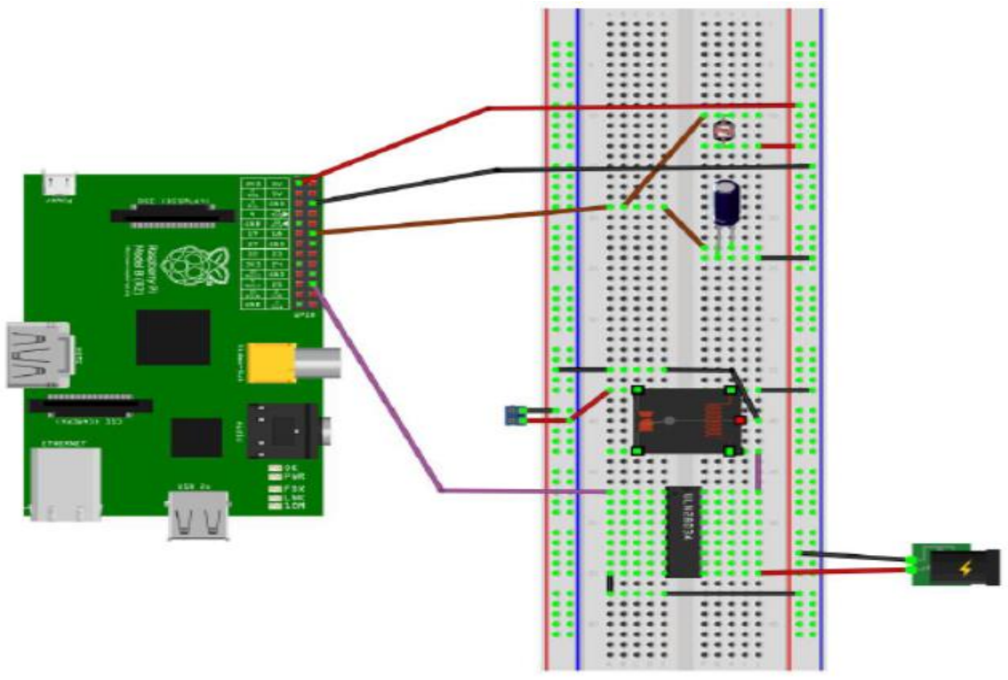


Figure 11: Schematic diagram of the home automation IoT system showing the device, sensor and actuator integrated

Case Study on loT System for Weather Monitoring

The purpose of the weather monitoring system is to collect data on environmental conditions such as temperature, pressure, humidity and light in an area using multiple end nodes.

The end nodes send the data to the cloud where the data is aggregated and analyzed.

Figure 13 shows the process specification for the weather monitoring system. The process specification shows that the sensors are read after fixed intervals and the sensor measurements are stored.

Figure 14 shows the domain model for the weather monitoring system.

In this domain model the physical entity is the environment which is being monitored.

There is a virtual entity for the environment. Devices include temperature sensor, pressure sensor, humidity sensor, light sensor and single-board minicomputer. Resources are software components which can be either on-device or network-resources.

Services include the controller service that monitors the temperature, pressure, humidity and light and sends the readings to the cloud.

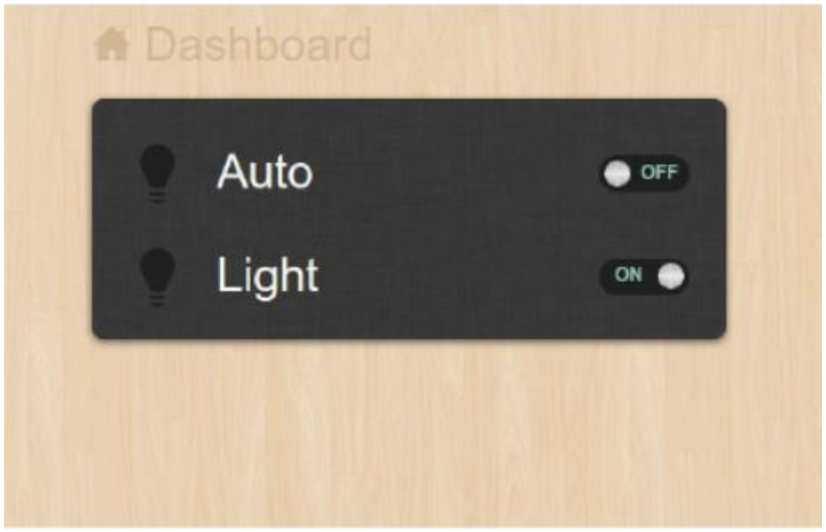


Figure 12: Home automation web application screenshot

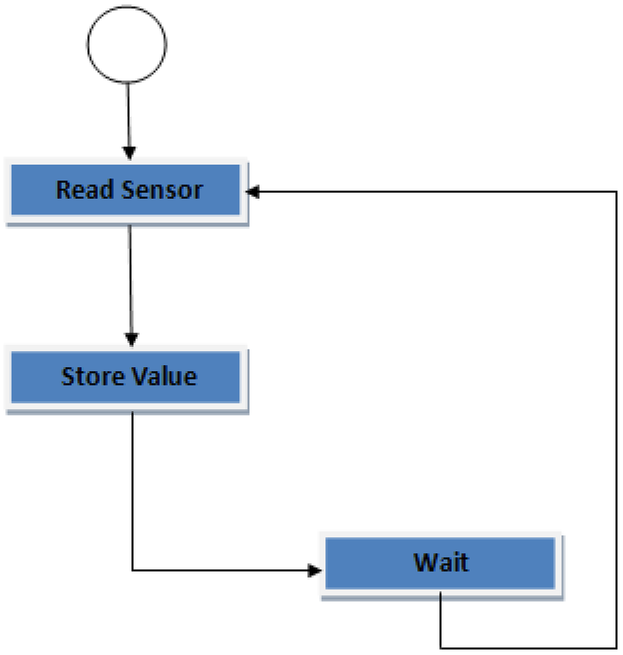


Figure 13: Process specification for weather monitoring IoT system

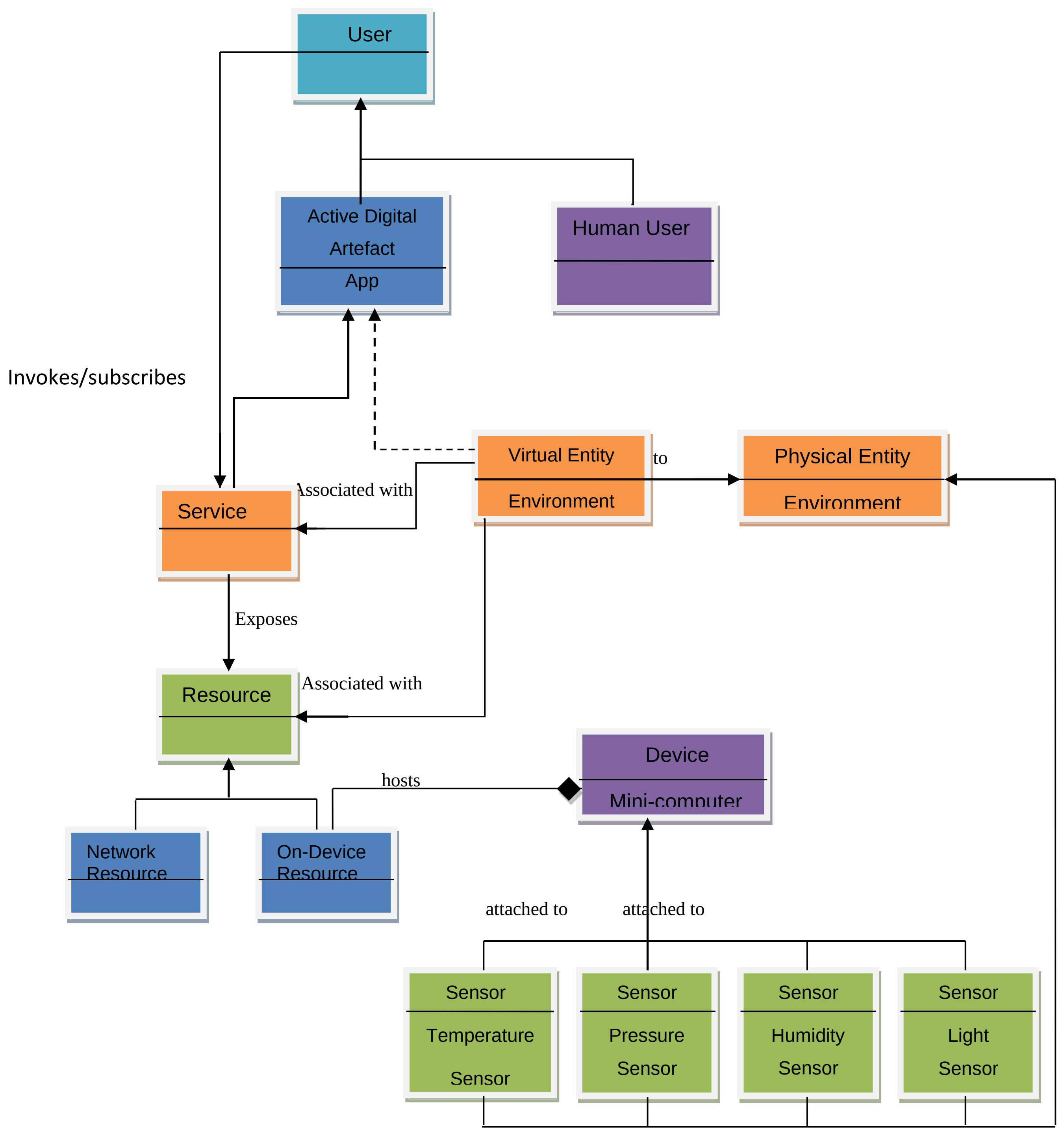


Figure 14: Domain model for weather monitoring IoT system

Figure 18 shows an example of mapping deployment level to functional groups for the weather monitoring system.

Figure 19 shows an example of mapping functional groups to operational view specifications for the weather monitoring system.

Figure 20 shows a schematic diagram of the weather monitoring system.

The devices and components used in this example are Raspberry Pi minicomputer, temperature sensor, humidity sensor, pressure sensor and LDR sensor.

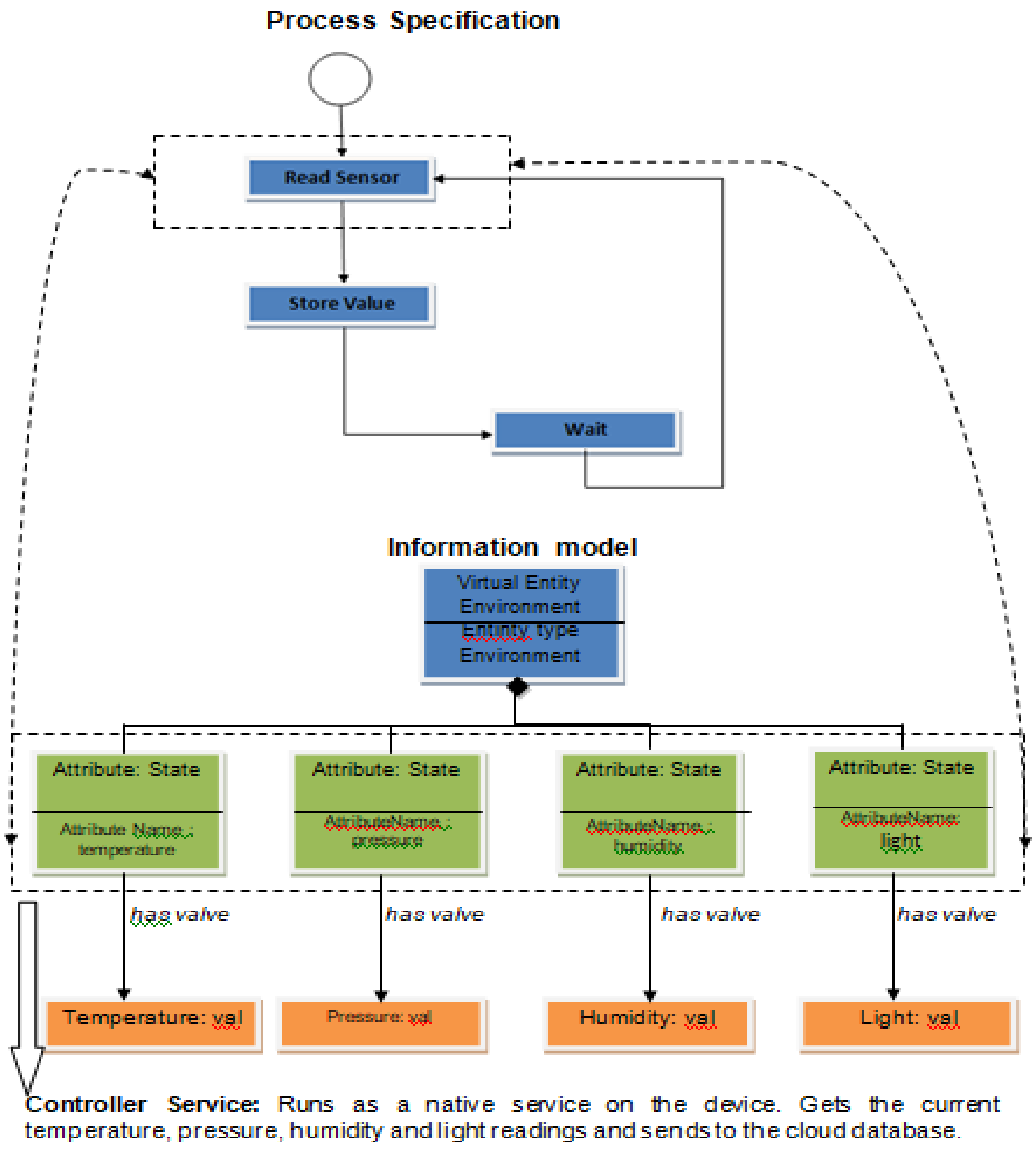


Figure 15: Deriving services from process specification & information model for weather monitoring IOT system

Motivation for Using Python

We explain the motivation for using Python for developing IoT systems. Python is a minimalistic language with English-like keywords and fewer syntactical constructions as compared to other languages. This makes Python easier to learn and understand.

Moreover, Python code is compact as compared to other languages.

Python is an interpreted language and does not require an explicit compilation step.

The Python interpreter converts the Python code to the intermediate byte code, specific to the system.

Python is supported on wide range of platforms, hence Python code is portable

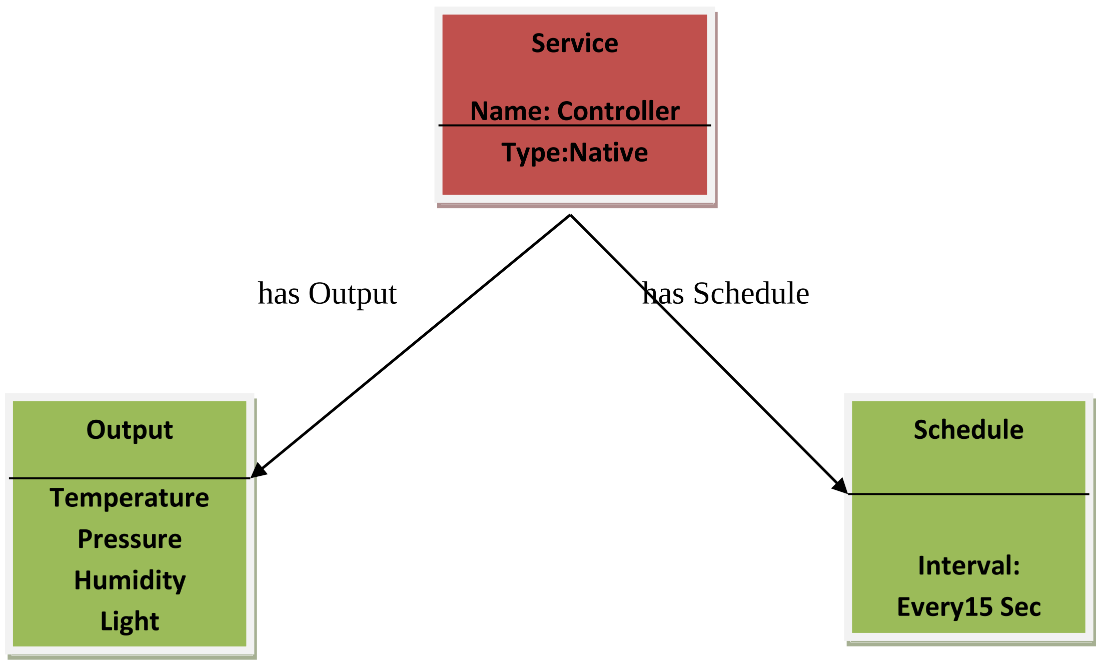


Figure 16: Controller service of the weather monitoring IOT system

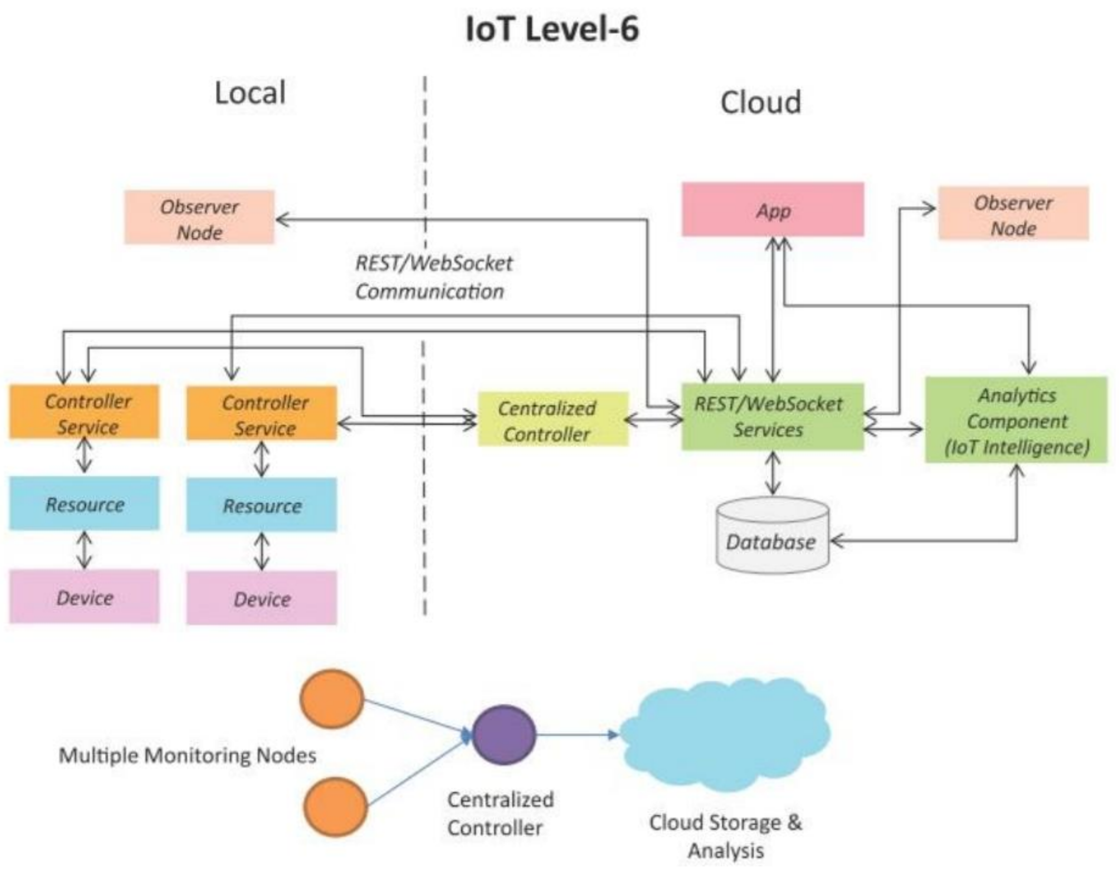


Figure 17: Deployment design of the weather monitoring IOT system

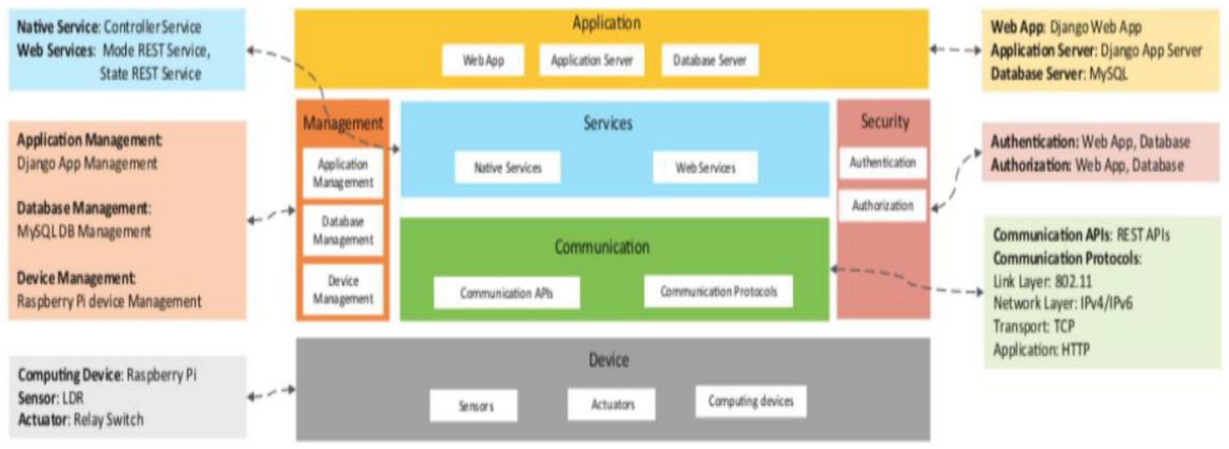


Figure 18: Mapping functional groups to operational view for the weather monitoring IOT system

IOT SYSTEMS - LOGICAL DESIGN USING PYTHON

Python is a general-purpose high level programming language.

There is limited library support for the 3.x versions with operating systems such as Linux and Mac still using Python 2.x as default language.

Here the exercises and examples in have been developed with Python version 2.7.

The main characteristics of Python are:

Multi-paradigm programming language

Python supports more than one programming paradigms including object-oriented programming and structured programming

Interpreted Language

Python is an interpreted language and does not require an explicit compilation step. The Python interpreter executes the program source code directly, statement by statement, as a processor or scripting engine does.

Interactive Language

Python provides an interactive mode in which the user can submit commands at the Python prompt and interact with the interpreter directly.

The key benefits of Python are:

Easy-to-learn, read and maintain

Python is a minimalistic language with relatively few keywords, uses English keywords and has fewer syntactical constructions as compared to other languages. Reading Python programs is easy with pseudo-code like constructs. Python is easy to learn yet an extremely powerful language for a wide range of applications. Due to its simplicity, programs written in Python are generally easy to maintain.

Object and Procedure Oriented

Python supports both procedure-oriented programming and object-oriented programming Procedure oriented paradigm allows programs to be written around procedures or functions that allow reuse of code. Procedure oriented paradigm allows programs to be written an objects that include both data and functionality.

Extendable

Python is an extendable language and allows integration of low-level modules written in languages such as C/C++. This is useful when you want to speed up a critical portion of a program

Scalable

Due to the minimalistic nature of Python, it provides a manageable structure for large programs.

Portable

Since Python is an interpreted language, programmers do not have to worry about compilation, linking and loading of programs. Python programs can be directly executed from source code and copied from one machine to other without worrying about portability. The Python interpreter converts the source code to an intermediate form called byte codes and then translates this into the native language of your specific system and then runs it.

Broad Library Support

Python has a broad library support and works on various platforms such as Windows, Linux, Mac, etc. There are a large number of Python packages available for various applications such as machine learning, image processing, network programming, cryptography, etc.

INSTALLING PYTHON

Python is a highly portable language that works on various platforms such as Windows, Linux, Mac, etc. This section describes the Python installation steps for Windows and Linux:

Windows

Python binaries for Windows can be downloaded from http://www.python.org/getit. For the examples and exercise in this book, you would require Python 2.7 which can be directly downloaded from: http://www.python.org/ftp/python/2.7.5/python-2.7.5.msi Once the python binary is installed you can run the python shell at the command prompt using > python

SUMMARY

- Python is a general-purpose, high level programming language that supports more than one programming paradigms including object-oriented programming and structured programming.
- Python is an interpreted language and does not require an explicit compilation step.
- Python provides an interactive mode in which the user can submit commands at the Python prompt and interact with the interpreter directly.
- The design methodology for IoT system design which is independent of specific product, service or programming language.
- The first step in IoT system design methodology is to define the purpose and requirements of the system.
- In the second step, the use cases of the IoT system are formally described based on the purpose and requirement specifications.
- The third step is to define the Domain Model which describes the main concepts, entities and objects in the domain of IoT system to be designed.
- The fourth step is to define the Information Model which defines the structure of all the information in the IoT system.
- The fifth step is to define the Functional View which defines the functions of the loT systems grouped into various Functional Groups.
- The sixth step is to define the service specifications which define the services in the IoT system, service types, service inputs/output, service endpoints, service schedules, service preconditions and service effects.
- The seventh step is to define the Deployment & Operational View Specifications in which various options pertaining to the IoT system deployment and operation are defined.
- The eight step is the integration of the devices and components.
- The final step in the IoT design methodology is to develop the IoT application.

ASSIGNMENT QUESTIONS

1. What is the difference between a physical and virtual entity?
2. What is an IoT device?
3. What is the purpose of information model?
4. What are the various service types?
5. What is the need for a controller service?

overview of RFID

Radio-Frequency Identification (RFID) is a technology that uses wireless communication to identify, track, and manage objects, people, or animals. It typically involves two main components: RFID tags and RFID readers.

RFID Tags:

Microchip: Contains a unique identifier and possibly additional information.
Antenna: Facilitates communication with the RFID reader through radio waves.
Encapsulation: Protects the microchip and antenna.
RFID Readers:

Antenna: Sends signals to RFID tags and receives responses.
Transceiver: Processes the data received from the tags and communicates with a central system.
Controller: Manages communication between the RFID reader and external systems.
Working Principle:

When an RFID tag comes into the range of an RFID reader, the reader sends an interrogation signal.

The RFID tag, powered by this signal, responds by transmitting its unique identifier and, if applicable, additional data.

The RFID reader captures this information and communicates it to a central database or system.

RFID Frequencies:

RFID operates at various frequencies, including low-frequency (LF), high-frequency (HF), and ultra-high-frequency (UHF).

LF (125-134 kHz), HF (13.56 MHz), and UHF (860-960 MHz) frequencies are commonly used.

Applications:

- Inventory Management: RFID is widely used in retail and logistics for tracking inventory, reducing errors, and improving efficiency.
- Access Control: RFID tags are used in access cards for secure entry to buildings or restricted areas.
- Supply Chain Management: RFID helps monitor the movement of goods, improving visibility and reducing the chances of loss or theft.
- Asset Tracking: Valuable assets, such as equipment or tools, can be tagged and tracked using RFID.
- Passports and ID Cards: Some countries use RFID technology in passports and identification cards for secure authentication.

Advantages:

Automation: RFID enables automation in various processes, reducing the need for manual intervention.

Accuracy:

RFID systems provide accurate and real-time data, minimizing errors.

low power design Bluetooth low energy

Bluetooth Low Energy (BLE) is a wireless communication technology designed for short-range communication with low power consumption. It's commonly used in applications where power efficiency is critical, such as wearable devices, healthcare sensors, and IoT (Internet of Things) devices. Designing a low-power Bluetooth Low Energy system involves several key considerations:

Low Power Microcontrollers:

Choose microcontrollers or systems-on-chip (SoCs) that are specifically designed for low power consumption.

Utilize low-power modes on the microcontroller, such as sleep and standby modes, to reduce power consumption during idle periods.

Transmit Power Control:

Adjust the transmit power based on the communication range requirements. Lowering the transmit power can significantly reduce power consumption.

Data Packet Optimization:

Minimize the amount of data transmitted by optimizing the size of data packets. Smaller packets lead to shorter transmission times and lower power consumption.

Voltage Regulation:

Implement efficient voltage regulation to ensure that the system operates at the lowest possible voltage without compromising functionality.

Range extension technique

Extending the range of a wireless communication system, such as Bluetooth or Wi-Fi, can be essential in various applications.

Mesh Networking:

Mesh networking is a type of network topology where each node in the network can act as a relay for data transmission. This can be particularly useful for extending the range of a network. In a mesh network, nodes are interconnected, and each node can relay data to other nodes. This relay functionality helps in extending the effective range of communication. If a direct communication link between two nodes is not possible due to distance or obstacles, intermediate nodes can relay the data to bridge the gap. if one node fails or is out of range, the network can dynamically reroute data through other available nodes. This adaptive nature contributes to improved reliability and range extension.

Data Mining:

Data mining is the process of discovering patterns, trends, and insights from large datasets. data mining can play a role in optimizing network performance and understanding communication patterns within a mesh network. Predictive modelling using data mining can help anticipate network congestion or potential failures. This information can be used to dynamically adjust the mesh network, ensuring optimal performance and reliability.

Data intensive IoT for continuous recognition application

Implementing a data-intensive Internet of Things (IoT) system for continuous recognition applications involves managing large volumes of data generated by sensors and devices. It involves

1. Sensor Selection and Data Collection:

Choose sensors that provide accurate and reliable data for continuous recognition. Examples include cameras, microphones, accelerometers, and environmental sensors.

2. Data Preprocessing:

Apply filtering techniques to reduce noise and irrelevant data.

3. Cloud Infrastructure:

Utilize scalable cloud storage solutions to accommodate large volumes of data and leverage big data processing tools and frameworks (e.g., Apache Spark) for handling and analysing large datasets.

4. Data Security and Privacy:

Implement end-to-end encryption to secure data during transmission.

5. Continuous Recognition Algorithms:

Develop or deploy machine learning models for continuous recognition tasks.

6. Real-Time Analytics:

Implement real-time analytics for immediate insights from streaming data.

Overview of android

Android is a mobile operating system developed by a consortium of developers known as the Open Handset Alliance and commercially sponsored by Google. It is designed primarily for touchscreen mobile devices such as smartphones and tablets. key aspects of the Android operating system is

Architecture:

Android is built on the Linux kernel, which provides the core system services such as security, memory management, process management, and networking. It includes a set of C/C++ libraries for core system functionality, such as the SQLite database engine, WebKit for web browsing, and OpenGL for graphics rendering. Android applications run in the Android Runtime, which is responsible for executing and managing app code. ART is the default runtime starting from Android 5.0 (Lollipop).

Android provides a rich set of APIs and Java libraries for developing mobile applications. Developers can access features such as UI design, data storage, connectivity, multimedia, and more

Android applications are typically distributed through the Google Play Store, where users can browse, download, and install apps.

Android supports various wireless communication technologies, including Wi-Fi, Bluetooth, NFC, and mobile data (3G, 4G/LTE).

IOS app development tool

iOS app development tools are a set of software tools used by developers to create, design, test, and deploy iOS applications. These tools include programming languages such as Swift and Objective-C, integrated development environments (IDEs) such as Xcode, frameworks such as UIKit and SwiftUI, and performance analysis tools such as Instruments. They allow developers to create a wide range of applications, from simple utilities to complex games and enterprise-level solutions.

Using iOS app development tools, developers have the ability to create robust and efficient applications that run on Apple's operating system and take advantage of the latest technologies and features available on iOS devices.

Internet of everything

The term "Internet of Everything" (IoE) refers to the extension of the Internet of Things (IoT) concept to include not only devices and things but also the connections, data, and interactions among them. It represents a vision of a fully interconnected world where people, processes, data, and things are seamlessly integrated, creating new opportunities for innovation, efficiency, and improved experiences. Cisco is often credited with popularizing the term IoE.

The Internet of Everything (IoE) is a concept that extends the Internet of Things (IoT) emphasis on machine-to-machine (M2M) communications to describe a more complex system that also encompasses people and processes.