EMBEDDED SYSTEMS & IOT

23ECE364B

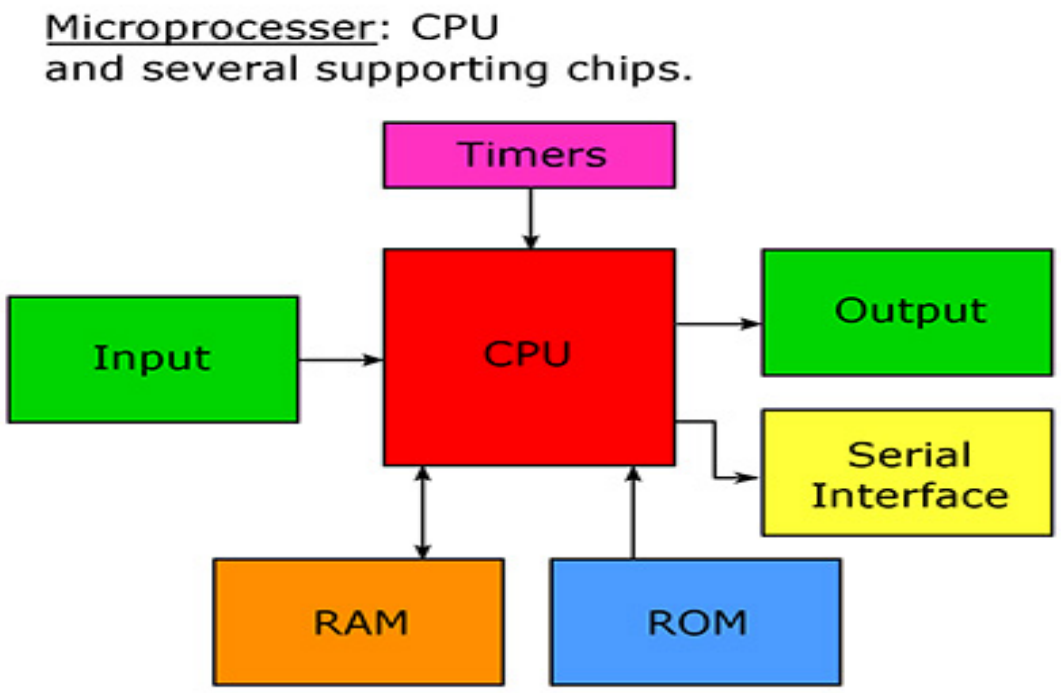
EMBEDDED SYSTEMS AND INTERNET OF THINGS

- Introduction, Hardware & Software Architecture of Embedded Systems, Embedded Systems, Development process, Architecture of Internet of Things, Physical Design & Logical Design of IoT,IoT Enabling Technologies, IoT Levels & Deployment Tools, Applications of Embedded Systems and IoT, Design Methodology for IOT Products.

INTRODUCTION

MICROPROCESSOR

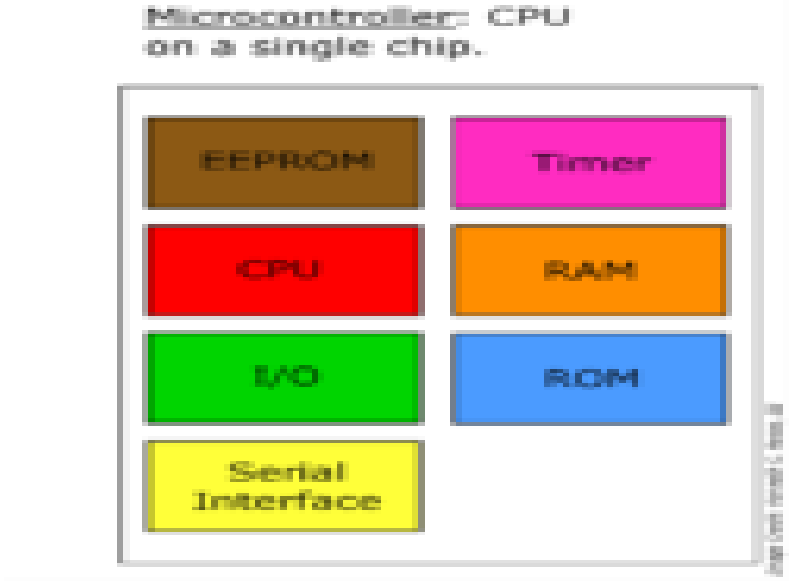
- A microprocessor is a single semiconductor chip
- That integrates the main five functional units of a computer: arithmetic/logical, control, storage, input, and output.
-



INTRODUCTION

MICROCONTROLLER

- A microcontroller is a highly integrated chip that contains many or all of the components comprising a controller.
- This includes a CPU, RAM and ROM, I/O ports, and timers.



Difference between Microprocessor and Microcontroller

Microprocessor	**Microcontroller**
Microprocessor is the heart of Computer system.	Micro Controller is the heart of an embedded system.
It is only a processor, so memory and I/O components need to be connected externally	Micro Controller has a processor along with internal memory and I/O components.
Memory and I/O has to be connected externally, so the circuit becomes large.	Memory and I/O are already present, and the internal circuit is small.
You can’t use it in compact systems	You can use it in compact systems.
Cost of the entire system is high	Cost of the entire system is low

Due to external components, the total power consumption is high. Therefore, it is not ideal for the devices running on stored power like batteries.	As external components are low, total power consumption is less. So it can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the microcontrollers offer power-saving mode.
It is mainly used in personal computers.	It is used mainly in a washing machine, MP3 players, and embedded systems.
Microprocessor has a smaller number of registers, so more operations are memory-based.	Microcontroller has more register. Hence the programs are easier to write.
Microprocessors are based on Von Neumann model	Micro controllers are based on Harvard architecture
It is a central processing unit on a single silicon-based integrated chip.	It is a byproduct of the development of microprocessors with a CPU along with other peripherals.
It has no RAM, ROM, Input-Output units, timers, and other peripherals on the chip.	It has a CPU along with RAM, ROM, and other peripherals embedded on a single chip.
It uses an external bus to interface to RAM, ROM, and other peripherals.	It uses an internal controlling bus.
Microprocessor-based systems can run at a very high speed because of the technology involved.	Microcontroller based systems run up to 200MHz or more depending on the architecture.
It’s used for general purpose applications that allow you to handle loads of data.	It’s used for application-specific systems.
It’s complex and expensive, with a large number of instructions to process.	It’s simple and inexpensive with less number of instructions to process.

Introduction to Embedded System

Embedded System

An Electronic/Electro mechanical system which is designed to perform a specific function and is a combination of both hardware and firmware (Software)

E.g. Electronic Toys, Mobile Handsets, Washing Machines, Air Conditioners, Automotive Control Units, Set Top Box, DVD Player etc…

History of Embedded Systems:

- First Recognized Modern Embedded System: Apollo Guidance Computer (AGC)
- First Mass Produced Embedded System: Autonetics D-17 Guidance computer
- **Classification of Embedded Systems:**
- Based on Generation
- Based on Complexity & Performance Requirements
- Based on deterministic behavior
- Based on Triggering

Embedded Systems - Classification based on Generation

- **First Generation:** The early embedded systems built around 8 bit microprocessor like 8085 and Z80 and 4 bit microcontrollers
- **Second Generation:** Embedded Systems built around 16bit microprocessors and 8 or 16bit microcontrollers, following the first generation embedded systems
- **Third Generation:** Embedded Systems built around high performance 16/32 bit Microprocessors/controllers, Application Specific Instruction set processors like Digital Signal Processors (DSPs), and Application Specific Integrated Circuits ASICs).
- **Fourth Generation:** Embedded Systems built around System on Chips (SoCs),Re-configurable processors and multicore processors

Embedded Systems - Classification based on Complexity &Performance

- **Small Scale:** The early embedded systems built around 8bit microprocessors like 8085 and Z80 and 4bit microcontrollers
- **Medium Scale**: Embedded Systems built around 16bit microprocessors and 8 or 16bit microcontrollers, following the first generation embedded systems
- **Large Scale/Complex:** Embedded Systems built around high performance 16/32 bit Microprocessors/controllers, Application Specific Instruction set processors like Digital Signal Processors (DSPs), and Application Specific Integrated Circuits (ASICs)

Classification Based On Deterministic Behaviour

- It is applicable for **Real time systems**
- **Soft real time systems**: Missing a deadline may not be critical and can be tolerated to a certain degree.
- **Hard real time systems**: Missing a program/task execution time deadline can have catastrophic consequences(financial, human loss of life etc)

Based On Triggering

- **Event Triggered:** Activities within the system are dynamic and depend upon the occurrence of different events
- **Time triggered:** Activities within the system follow statically computed schedule(ie they are allocated time slots during which they can takes place)

Major Application Areas of Embedded Systems

- **Consumer Electronics:** Camcorders, Cameras etc.
- **Household Appliances:** Television, DVD players, Washing machine, Fridge, Microwave Oven etc.
- **Home Automation and Security Systems:** Air conditioners, sprinklers, Intruder detection alarms, Closed Circuit Television Cameras, Fire alarms etc.
- **Automotive Industry:** Anti-lock breaking systems (ABS), Engine Control, Ignition Systems, Automatic Navigation Systems etc.
- **Telecom**: Cellular Telephones, Telephone switches, Handset Multimedia Applications etc.
- **Computer Peripherals**: Printers, Scanners, Fax machines etc.
- **Computer Networking Systems:** Network Routers, Switches, Hubs, Firewalls etc.
- **Health Care:** Different Kinds of Scanners, EEG, ECG Machines etc.
- **Measurement & Instrumentation:** Digital multi meters, Digital CROs, Logic Analyzers PLC systems etc.
- **Banking & Retail:** Automatic Teller Machines (ATM) and Currency counters, Point of Sales (POS)

Purpose of Embedded Systems

• Each Embedded Systems is designed to serve the purpose of any one or a combination of the following tasks.

- ❑ Data Collection/Storage/Representation
- ❑ Data Communication
- ❑ Data (Signal) Processing
- ❑ Monitoring
- ❑ Control
- ❑ Application Specific User Interface

Purpose of Embedded Systems – Data Collection/Storage/Representation

- ✓The collected data can be either analog or digital
- ✓Data collection is usually done for storage, analysis, manipulation and transmission
- ✓The collected data may be stored directly in the system or may be transmitted to some other systems or it may be processed by the system or it may be deleted instantly after giving a meaningful representation

Purpose of Embedded Systems – Data Communication

- ✓ Embedded Data communication systems are deployed in applications ranging from complex satellite communication systems to simple home networking systems
- ✓ Embedded Data communication systems are dedicated for data communication
- The data communication can happen through wired interface (like Wi-Fi, GSM,/GPRS, Bluetooth, ZigBee etc)
- ✓ Network hubs, Routers, switches, Modems etc are typical examples for dedicated data transmission embedded systems

Purpose of Embedded Systems – Data (Signal) Processing

- ✓ Embedded systems with Signal processing functionalities are employed in applications demanding signal processing like Speech coding, synthesis, audio video codec, transmission applications etc
- ✓ Computational intensive systems
- ✓ Employs Digital Signal Processors (DSPs)

Digital hearing Aid employing Signal

Purpose of Embedded Systems – Monitoring

- ✓Embedded systems coming under this category are specifically designed for monitoring purpose
- ✓They are used for determining the state of some variables using input sensors
- ✓Electro Cardiogram (ECG) machine for monitoring the heart beat of a patient is a typical example for this
- ✓The sensors used in ECG are the different Electrodes connected to the patient's body
- ✓Measuring instruments like Digital CRO, Digital Multi meter, Logic Analyzer etc used in Control & Instrumentation applications are also examples of embedded systems for monitoring purpose.

Purpose of Embedded Systems – Control

- ✓ Embedded systems with control functionalities are used for imposing control over some variables according to the changes in input variables
- ✓ Embedded system with control functionality contains both sensors and actuators
- ✓ Sensors are connected to the input port for capturing the changes in environmental variable or measuring variable
- ✓ The actuators connected to the output port are controlled according to the changes in input variable to put an impact on the controlling variable to bring the controlled variable to the specified range
- ✓ Air conditioner for controlling room temperature is a typical example for embedded system with 'Control' functionality
- ✓ Air conditioner contains a room temperature sensing element (sensor) which may be a thermistor and a handheld unit for setting up (feeding) the desired temperature

Purpose of Embedded Systems – Application Specific User Interface

- ✓ Embedded systems which are designed for a specific application
- ✓ Contains Application Specific User interface (rather than general standard UI) like key board, Display units etc
- ✓ Aimed at a specific target group of users
- ✓ Mobile handsets, Control units in industrial applications etc are examples for this

Smart' running shoes from Adidas – The Innovative bonding of Life Style with Embedded Technology

- ✓ Shoe developed by Adidas, which constantly adapts its shock-absorbing characteristics to customize its value to the individual runner, depending on running style, pace, body weight, and running surface
- ✓ It contains sensors, actuators and a microprocessor unit which runs the algorithm for adapting the shock-absorbing characteristics of the shoe
- ✓ A 'Hall effect sensor' placed at the top of the "cushioning element" senses the compression and passes it to the Microprocessor
- ✓ A micro motor actuator controls the cushioning as per the commands from the MPU, based on the compression sensed by the 'Hall effect sensor'

Typical Architecture of an Embedded System

- Typical embedded system consisting of two main parts: embedded hardware and embedded software.
- The embedded hardware primarily includes the processor, memory, bus, peripheral devices, I/O ports, and various controllers.
- The embedded software usually contains the embedded operating system and various applications.

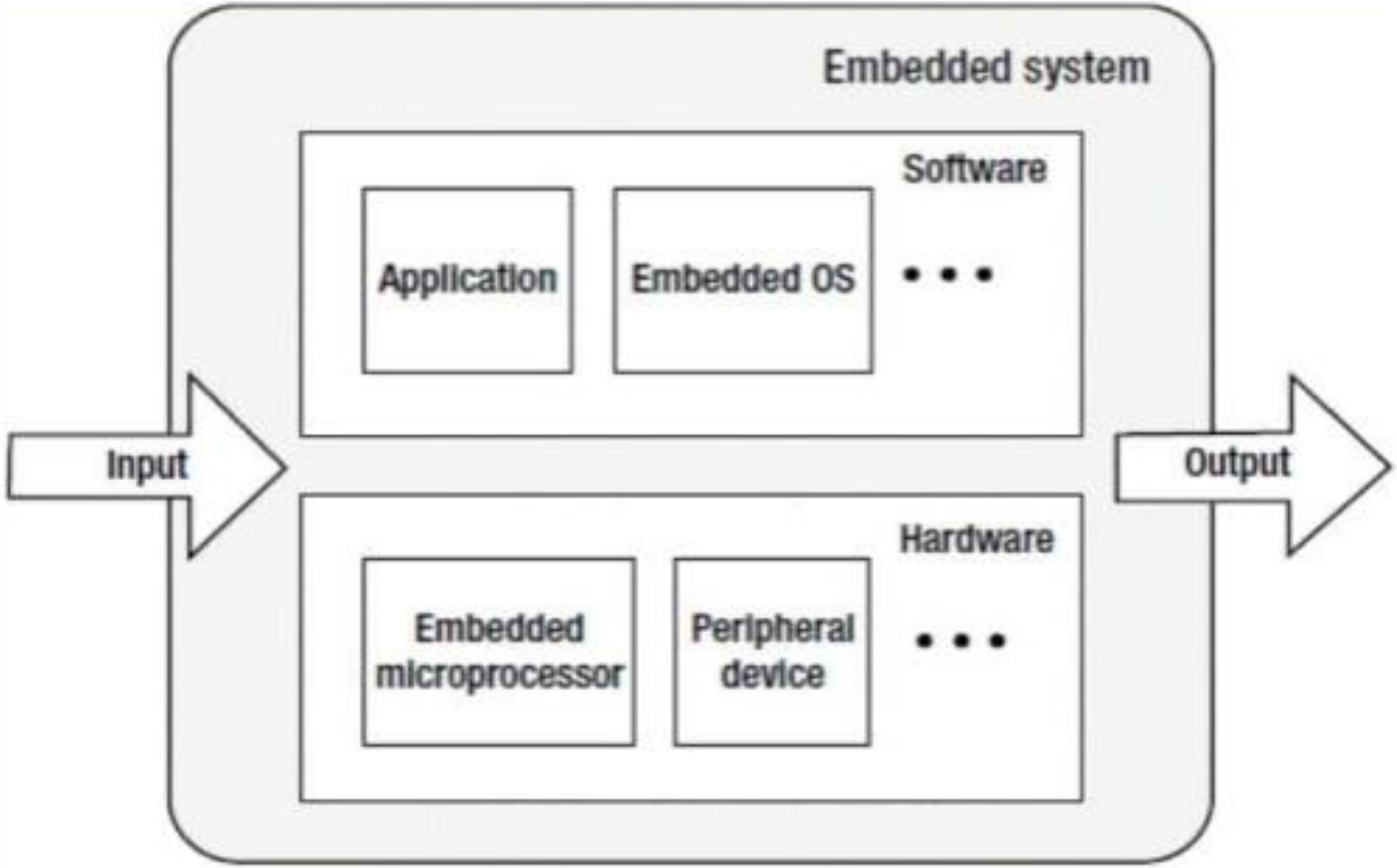


Figure 1-2. Basic architecture of an embedded system

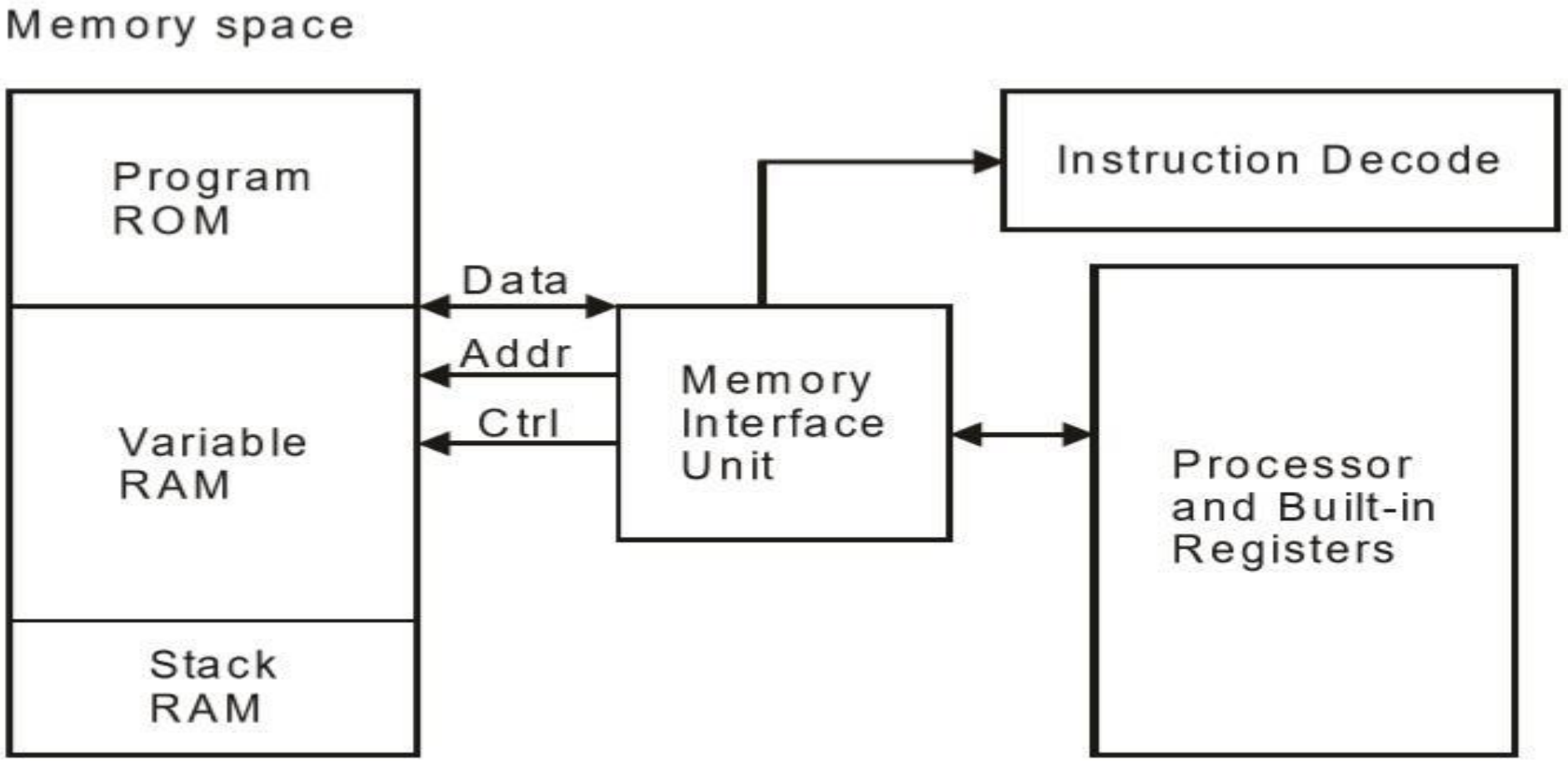
Hardware architecture

- There are basically two types of architecture that apply to embedded systems:
- Von Neumann architecture
- Harvard architecture.

Von Neumann Architecture

- Von Neumann architecture (also known as Princeton architecture) was first proposed by John von Neumann.
- The most important feature of this architecture is that the software and data use the same memory: that is,
- "The program is data, and the data is the program"

Von Neumann Architecture



Von Neumann Architecture

- Path or bus exists for both instruction and data. As a result, the CPU does one operation at a time.
- Instruction fetch and a data operation cannot occur simultaneously, sharing a common bus.
- It supports simple hardware.
- Processing speed very fast but small amount of memory

Von Neumann Architecture

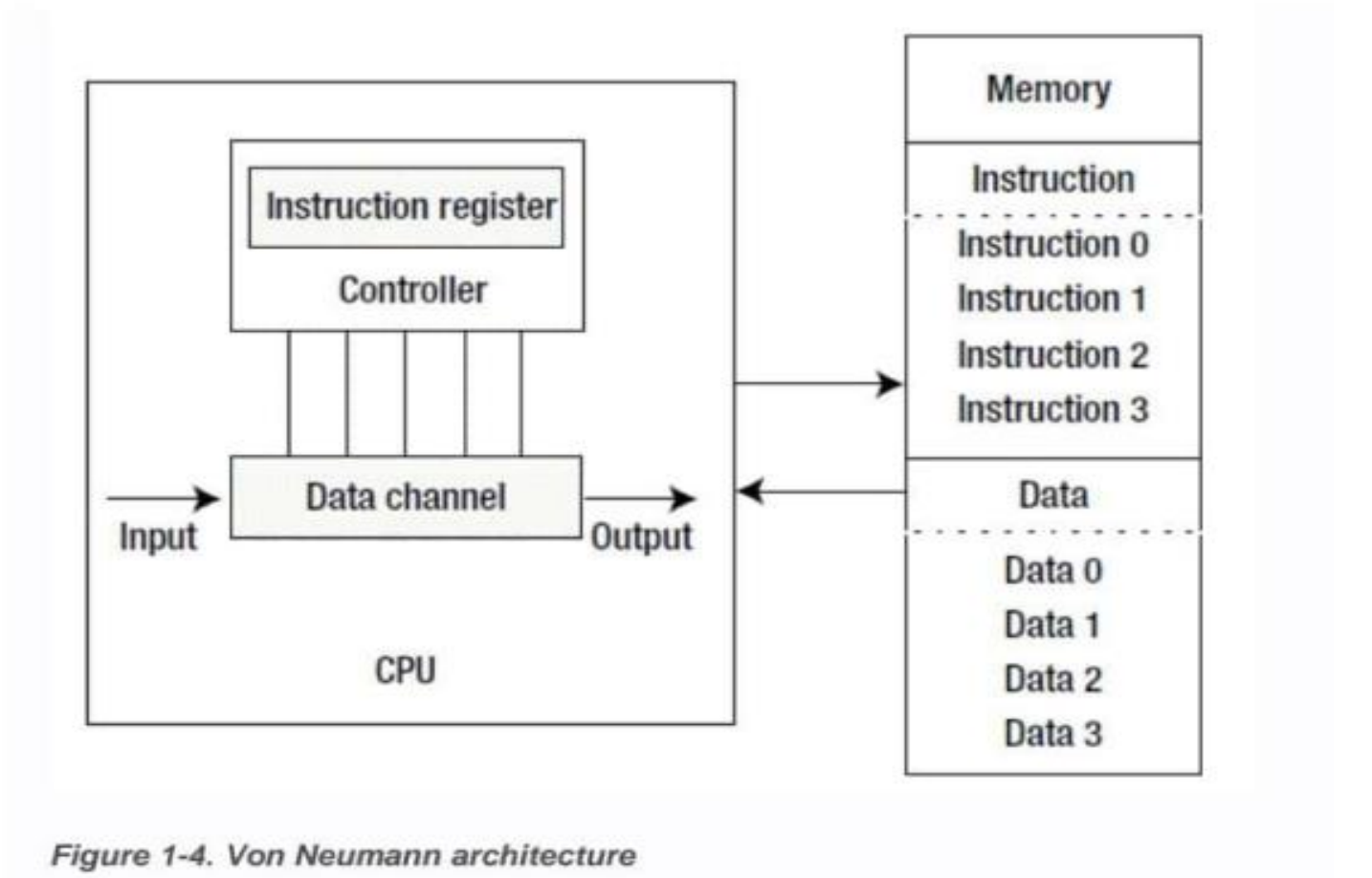
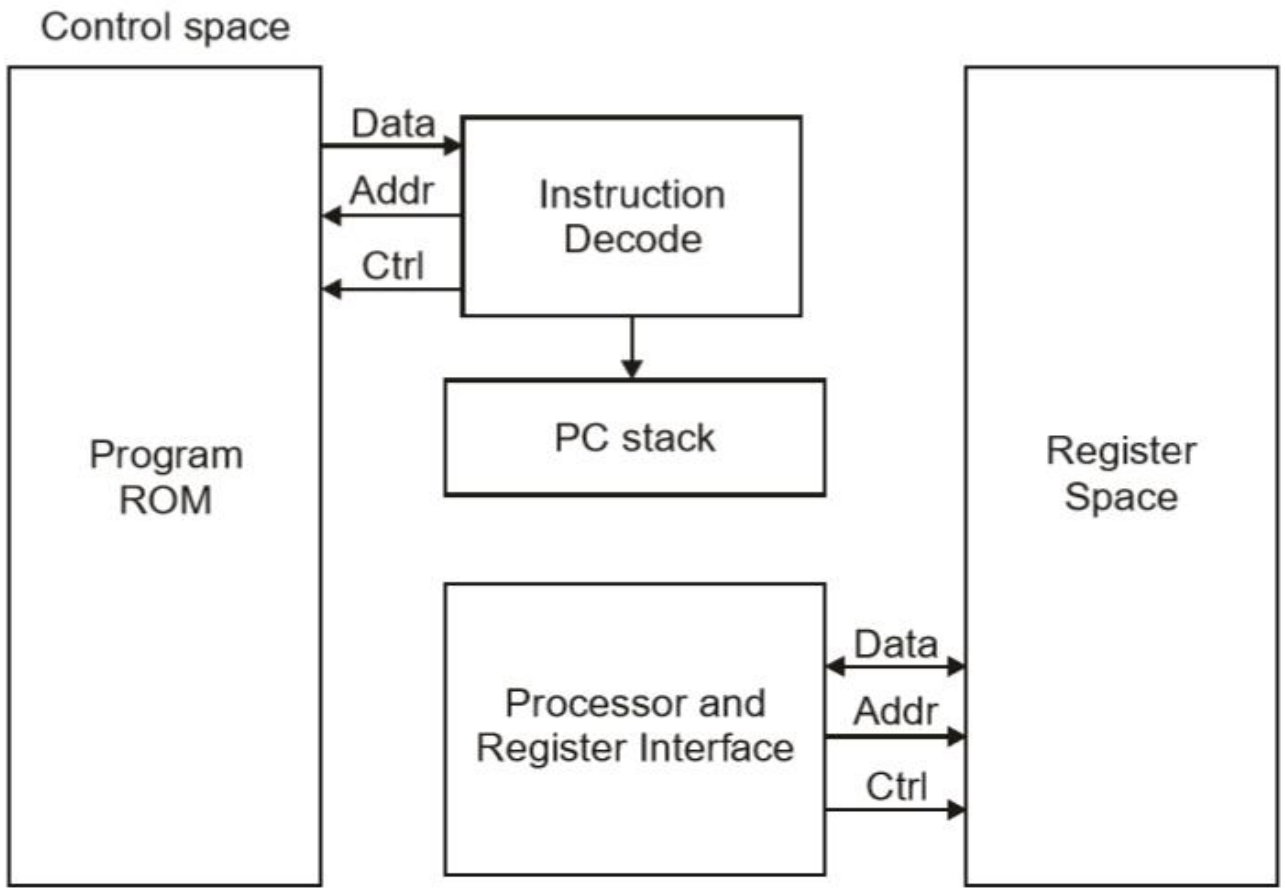


Figure 1-4. Von Neumann architecture

Harvard Architecture



Harvard Architecture

- The Harvard architecture was first named after the Harvard Mark I computer.
- Compared with the Von Neumann architecture, a Harvard architecture processor has two outstanding features.
- First, instructions and data are stored in two separate memory modules; instructions and data do not coexist in the same module.
- Second, two independent buses are used as dedicated communication paths between the CPU and memory; there is no connection between the two buses.

Har

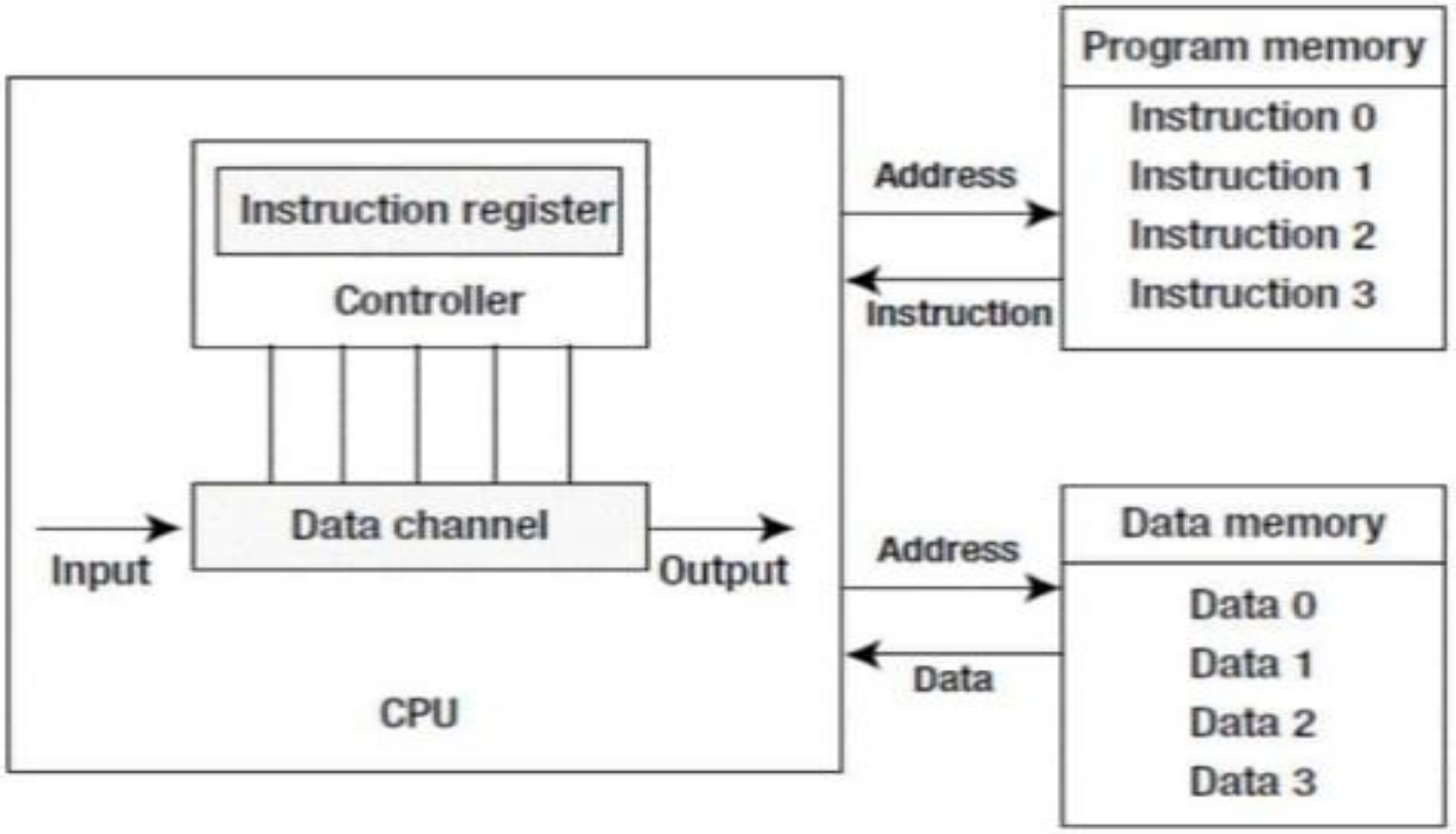


Figure 1-5. Harvard architecture

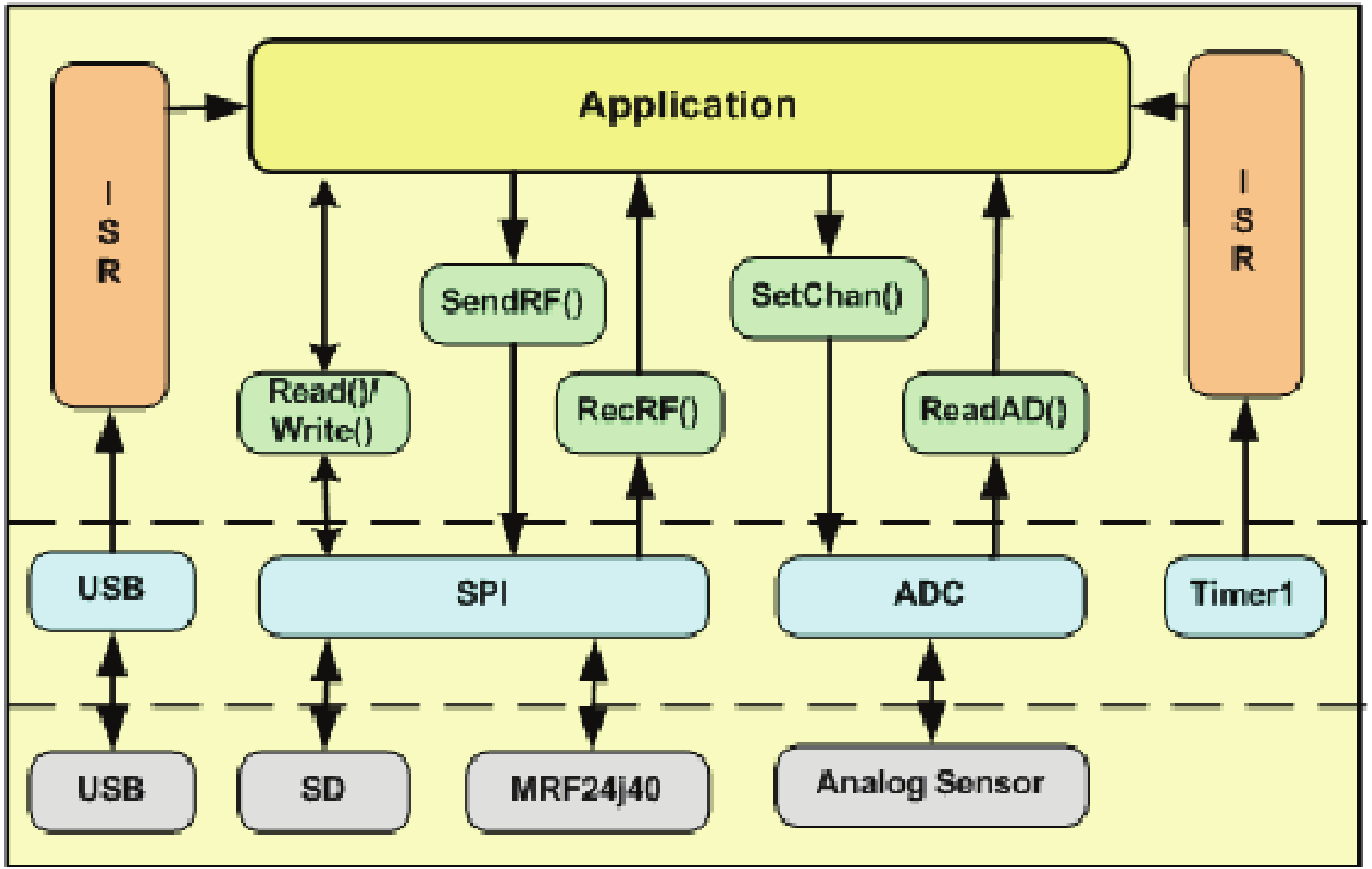
Harvard Architecture

- Because the Harvard architecture has separate program memory and data memory, it can provide greater data-memory bandwidth, making it the ideal choice for digital signal processing.
- Most systems designed for digital signal processing (DSP) adopt the Harvard architecture.
- The Von Neumann architecture features simple hardware design and flexible program and data storage and is usually the one chosen for general purpose and most embedded systems.

Harvard Architecture

- To efficiently perform memory reads/writes, the processor is not directly connected to the main memory, but to the cache.
- Commonly, the only difference between the Harvard architecture and the Von Neumann architecture is single or dual L1 cache.
- In the Harvard architecture, the L1 cache is often divided into an instruction cache (I cache) and a data cache (D cache), but the Von Neumann architecture has a single cache.

EMBEDDED SOFTWARE architecture



EMBEDDED SOFTWARE

- The whole software structure is divided in 4 layers.
- The layers are separated by dotted lines
- Physical level : is the lowest level and it depends on the hardware directly. The modules present in this level correspond to the physical modules of the node; these are the force sensors, accelerometer (not mounted), the USB port, the mini-SD slot (optional) and the RF module, which is controlled by the CPU using the SPI bus.
- Controller level : the functions developed in this level permit the application level to invoke controller functions. The ADC module converts analogical signals from the force sensors to digital, and the SPI allows communications of the CPU with the accelerometer and the mini-SD card and the RF chip. In this layer the set of USB and RTC (Real Time Clock) functions are also included.

Application Layer

- The **topmost layer** where user-defined logic resides.
- Controls system behavior using **high-level function calls** like:
  - Read() / Write() – for data transfer
  - SendRF() / RecRF() – for RF communication
  - SetChan() – to configure RF channel
  - ReadAD() – to read analog sensor data
- Independent of hardware details.

Interrupt Service Routines (ISR)

- Shown on both sides of the diagram.
- Handle **asynchronous events** such as:
  - Timer overflow
  - Data reception
  - Communication completion
- Improve **real-time responsiveness**.

Device Driver / Interface Layer

- Acts as a bridge between application and hardware.
- Includes drivers for:
  - **USB**
  - **SPI**
  - **ADC**
  - **Timer**
- Converts application requests into **hardware-specific commands**.

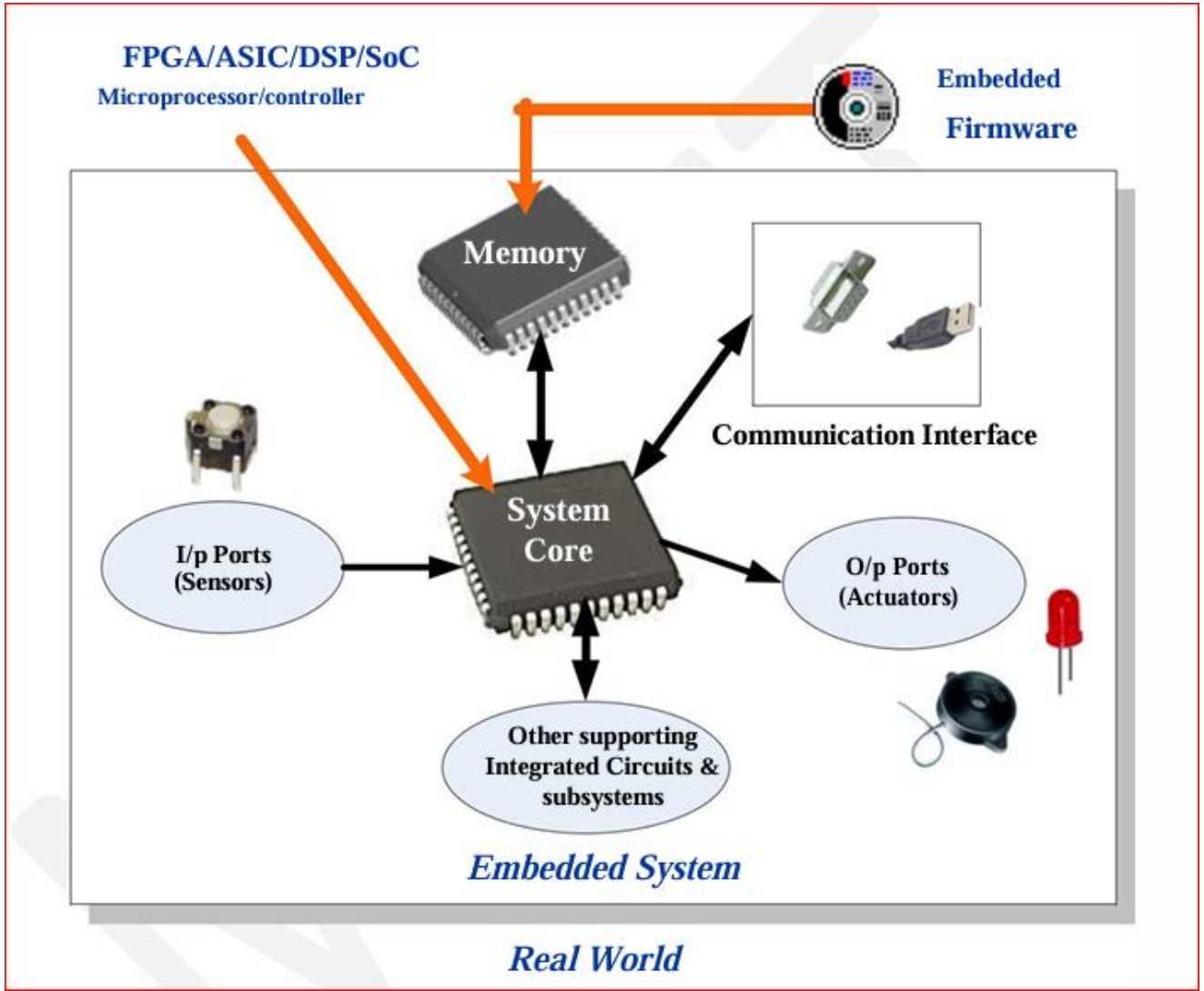
Hardware Abstraction / Interface Layer

- **SPI**: Used to communicate with SD card and RF module (MRF24J40).
- **USB**: Used for external communication or data transfer.
- **ADC**: Converts analog sensor signals to digital data.
- **Timer1**: Generates timing events and interrupts.

Hardware Layer

- Physical components of the embedded system
- Includes:
  - Sensors
  - SD card
  - RF module (MRF24J40)
  - USB device
- Performs actual data sensing and actuation

EMBEDDED SYSTEMS



EMBEDDED SYSTEMS

- An embedded system is a combination of 3 things, Hardware, Software Mechanical Components and it is supposed to do one specific task only.
- A typical embedded system contains a single chip controller which acts as the master brain of the system.

The Core of the Embedded Systems

- The core of the embedded system falls into any one of the following categories.

❑General Purpose and Domain Specific Processors

- o Microprocessors
- o Microcontrollers
- o Digital Signal Processors

❑Programmable Logic Devices (PLDs)

❑Application Specific Integrated Circuits (ASICs)

❑Commercial off the shelf Components (COTS)

Embedded System Development Process

- It is a structured lifecycle involving
- **Requirement Analysis**
- **System Architecture**
- **Hardware/SoftwareDesign,**
- **Prototyping, Implementation (Coding/Building),**
- **Testing, Deployment, and Maintenance**

Embedded System Development Process

- **Requirement Definition & Analysis**: Understand the product's purpose, functions (data collection, control, communication), constraints (power, cost, size, real-time deadlines).
- **System Architecture Design**: Define overall structure, select processor (MCU/MPU), memory, peripherals, and partition tasks between hardware and software

Embedded System Development Process

- **Hardware Design**: Schematic capture, PCB layout, component selection (sensors, actuators, interfaces).
- **Software Design**: Define modules, APIs, select OS (RTOS or bare-metal), and design control logic.
- **Prototyping**: Build initial functional models (Proof of Concept) to validate hardware and software integration.
- **Implementation (Coding & Building)**: Write firmware (C/C++, Assembly), compile, link, and generate executable code for the target.

Embedded System Development Process

- **Testing & Validation**: Verify functionality, performance, safety, and reliability on target hardware (debugging, simulation, field trials).
- **Deployment & Release**: Manufacture, distribute, and launch the product.
- **Maintenance & Support**: Ongoing updates, bug fixes, and improvements in the field.

Internet of Things - IOT

- The Internet of Things (IoT) refers
- to a network of interconnected physical objects such as devices, machines, vehicles,
- or people embedded with sensors, software, and unique identifiers that enable them to collect, exchange, and process data over a network
- without requiring direct human-to-human or human-to-computer interaction.

IoT architecture

- IoT architecture refers to the tangle of components such as sensors, actuators, cloud services, Protocols, and layers that make up IoT networking systems.
- In general, it is divided into layers that allow administrators to evaluate, monitor, and maintain the integrity of the system
- The architecture of IoT is a four-step process through which data flows from devices
- connected to sensors,
- through a network, and
- then through the cloud for processing,
- analysis, and storage.

IoT architecture

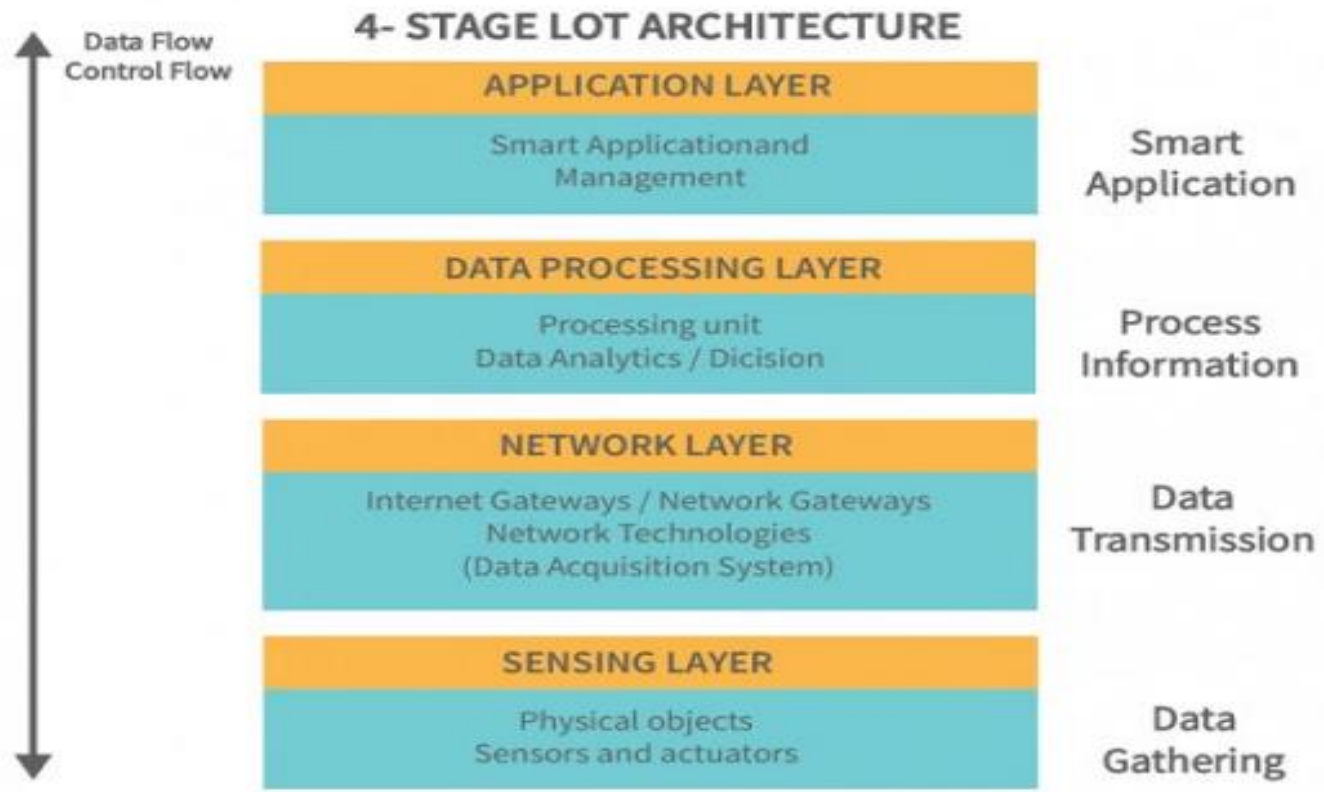


Fig: IoT Architecture

IoT architecture

- There are four layers present
  - i.e., The perception layer,
  - Network layer,
  - Processing layer, and
  - Application layer.

1)Perception/Sensing Layer:

- The first layer of any IoT system involves “things” or endpoint devices that serve as a conduct between the physical and the digital worlds.
- Perception refers to the physical layer, which includes sensors and actuators that are capable of collecting, accepting, and processing data over the network.
- Sensors and actuators can be connected either wirelessly or via wired connections.
- The architecture does not limit the scope of its components nor their location.

2) Network Layer:

- Network layers provide an overview of how data is moved throughout the application.
- This layer contains Data Acquiring Systems (DAS) and Internet/Network gateways.
- A DAS performs data aggregation and conversion functions (collecting and aggregating data from sensors, then converting analog data to digital data, etc.).
- It is necessary to transmit and process the data collected by the sensor devices. That's what the network layer does.
- It allows these devices to connect and communicate with other servers, smart devices, and network devices.
- As well, it handles all data transmissions for the devices.

3) Processing Layer:

- The processing layer is the brain of the IoT ecosystem.
- Typically, data is analyzed, pre-processed, and stored here before being sent to the data center, where it is accessed by software applications that monitor and manage the data as well as prepare further actions.
- This is where Edge IT or edge analytics enters the picture.

4) Application Layer:

- User interaction takes place at the application layer, which delivers application-specific services to the user.
- An example might be a smart home application where users can turn on a coffee maker by tapping a button in an app or a dashboard that shows the status of the devices in a system.
- There are many ways in which the Internet of Things can be deployed such as smart cities, smart homes, and smart health.

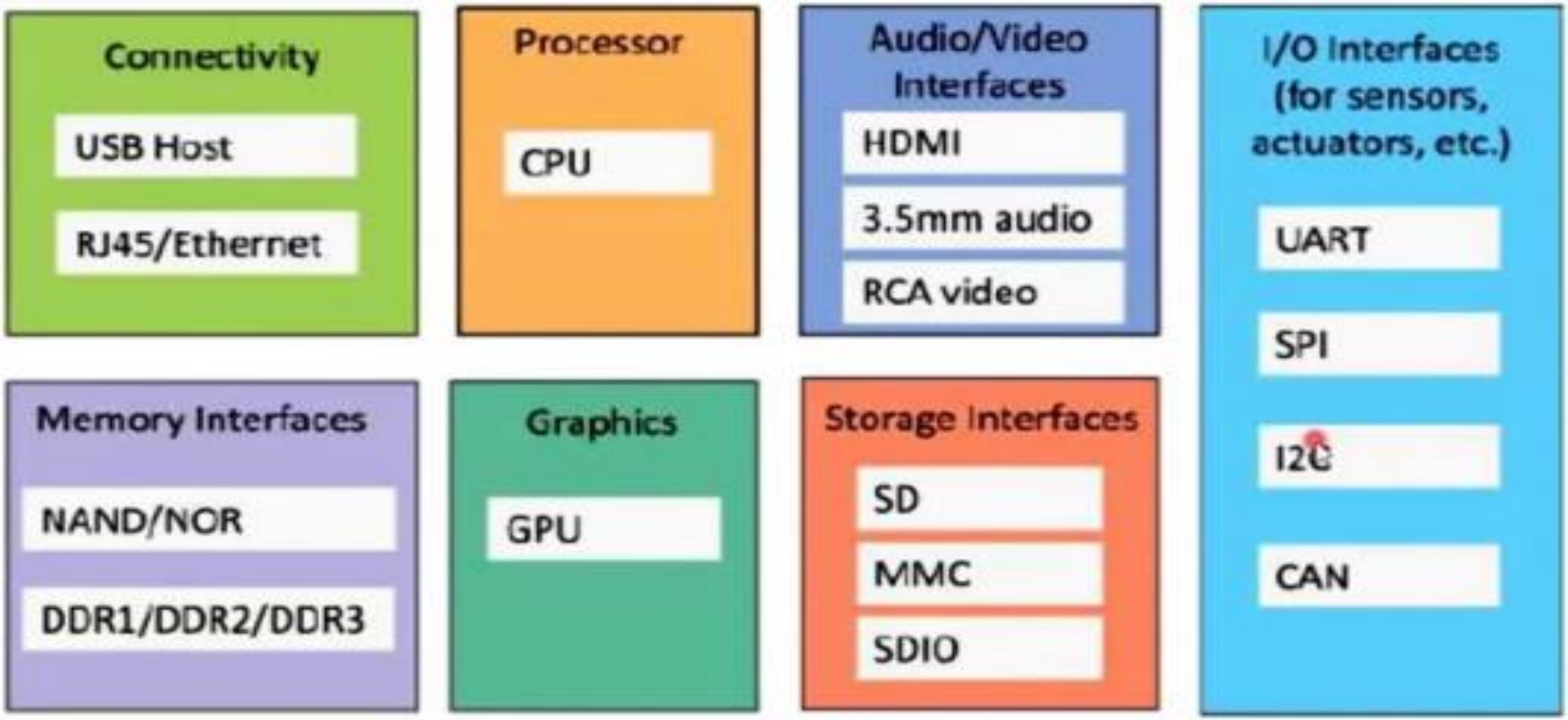
Physical Design of IoT

- The "Things" in IoT usually refers to IoT devices which have unique identities and can perform remote sensing, actuating and monitoring capabilities.
- IoT devices can:
- Exchange data with other connected devices and applications (directly or indirectly), or
- Collect data from other devices and process the data locally or
- Send the data to centralized servers or cloud-based application back-ends for processing the data, or
- Perform some tasks locally and other tasks within the IoT infrastructure, based on temporal and space constraints

Generic block diagram of an IoT Device

- An IoT device may consist of several interfaces for connections to other devices, both wired and wireless.
- I/O interfaces for sensors
- Interfaces for Internet connectivity
- Memory and storage interfaces
- Audio/video interfaces.

Generic Block Diagram of IoT Device



Generic block diagram of an IoT Device

- HDMI: High definition multimedia Interface.
- 3.5mm: Audio Jack which headphone adapter.
- RCA: Radio corporation of America.
- UART: Universal Asynchronous Receiver Transmitter.
- SPI: Serial Peripheral Interface.
- I2C: Inter integrated circuit
- CAN: Controller Area Network used for Micro-controllers and devices to communicate.
- SD: Secure digital (memory card)
- MMC: multimedia card
- SDIO: Secure digital Input Output
- GPU: Graphics processing unit.
- DDR: Double data rate

IoT Protocols:

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

Protocols:

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

b) Network/Internet Layer:

- Responsible for sending IP datagrams from source n/w to destination n/w.
- Performs the host addressing and packet routing.
- Datagrams contains source and destination address.

Protocols:

- IPv4: Internet Protocol version4 is used to identify the devices on a n/w using a hierarchical addressing scheme.
- 32 bit address. Allows total of 2*32addresses.
- IPv6: Internet Protocol version6 uses 128 bit address scheme and allows 2*128 addresses.
- 6LOWPAN:(IPv6 over Low power Wireless Personal Area Network) operates in 2.4 GHz frequency range and data transfer 250 kb/s.

c) Transport Layer:

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

Protocols:

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

d) Application Layer:

- Defines how the applications interface with lower layer protocols to send data over the n/w.
- Enables process-to-process communication using ports.
- **Protocols:**
- HTTP: Hyper Text Transfer Protocol that forms foundation of WWW. Follow request response model Stateless protocol.
- CoAP: Constrained Application Protocol for machine-to-machine(M2M) applications with constrained devices, constrained environment and constrained n/w. Uses client-server architecture.

Protocols:

- Web Socket: allows full duplex communication over a single socket connection.
- MQTT: Message Queue Telemetry Transport is light weight messaging protocol based on publish subscribe model. Uses client server architecture. Well suited for constrained environment.
- XMPP: Extensible Message and Presence Protocol for real time communication and streaming XML data between network entities. Support client-server and server-server communication.
-

Protocols:

- DDS: Data Distribution Service is data centric middleware standards for device-to-device or machine-to-machine communication. Uses publish-subscribe model.
- AMQP: Advanced Message Queuing Protocol is open application layer protocol for business messaging. Supports both point-to-point and publish-subscribe model.

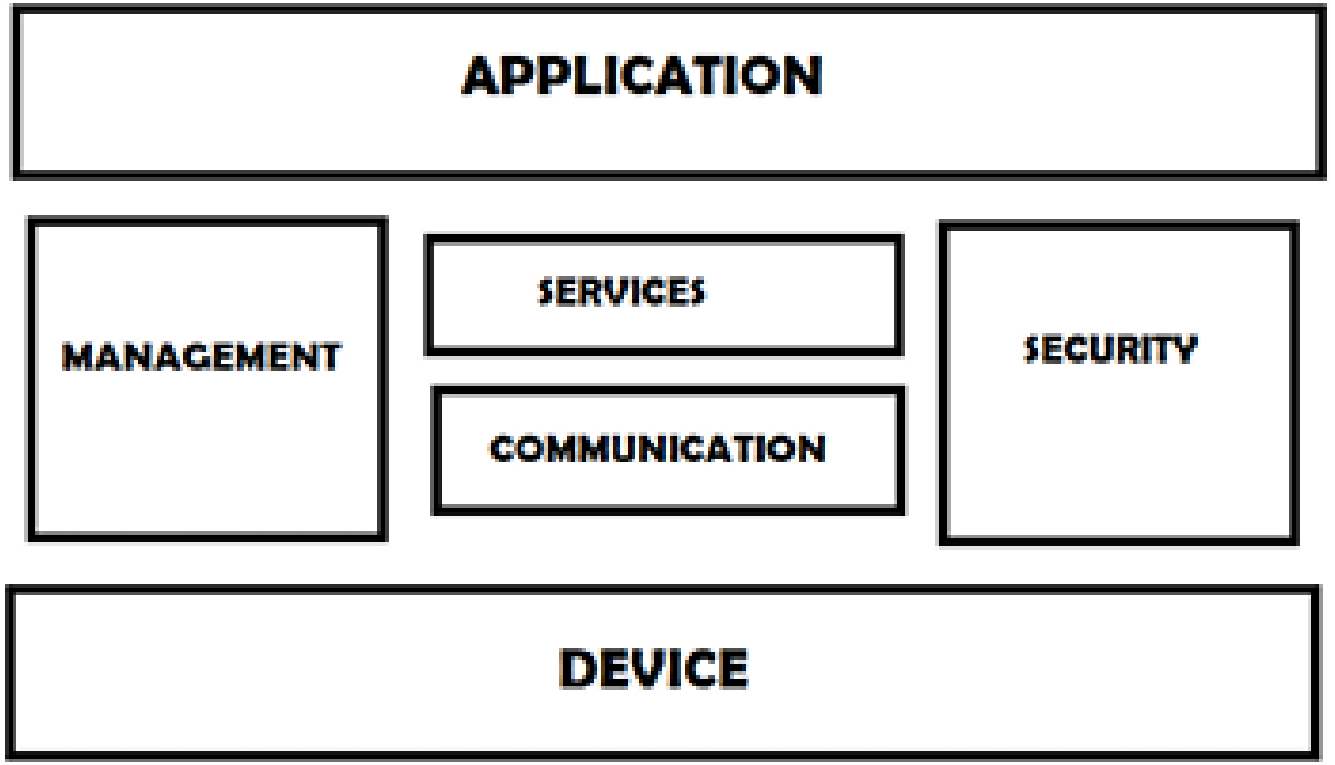
LOGICAL DESIGN of IoT

- Refers to an abstract represent of entities and processes without going into the low level specifies of implementation.
- 1) IoT Functional Blocks
- 2) IoT Communication Models
- 3) IoT Comm. APIs

IoT Functional Blocks:

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

IoT Functional Blocks:



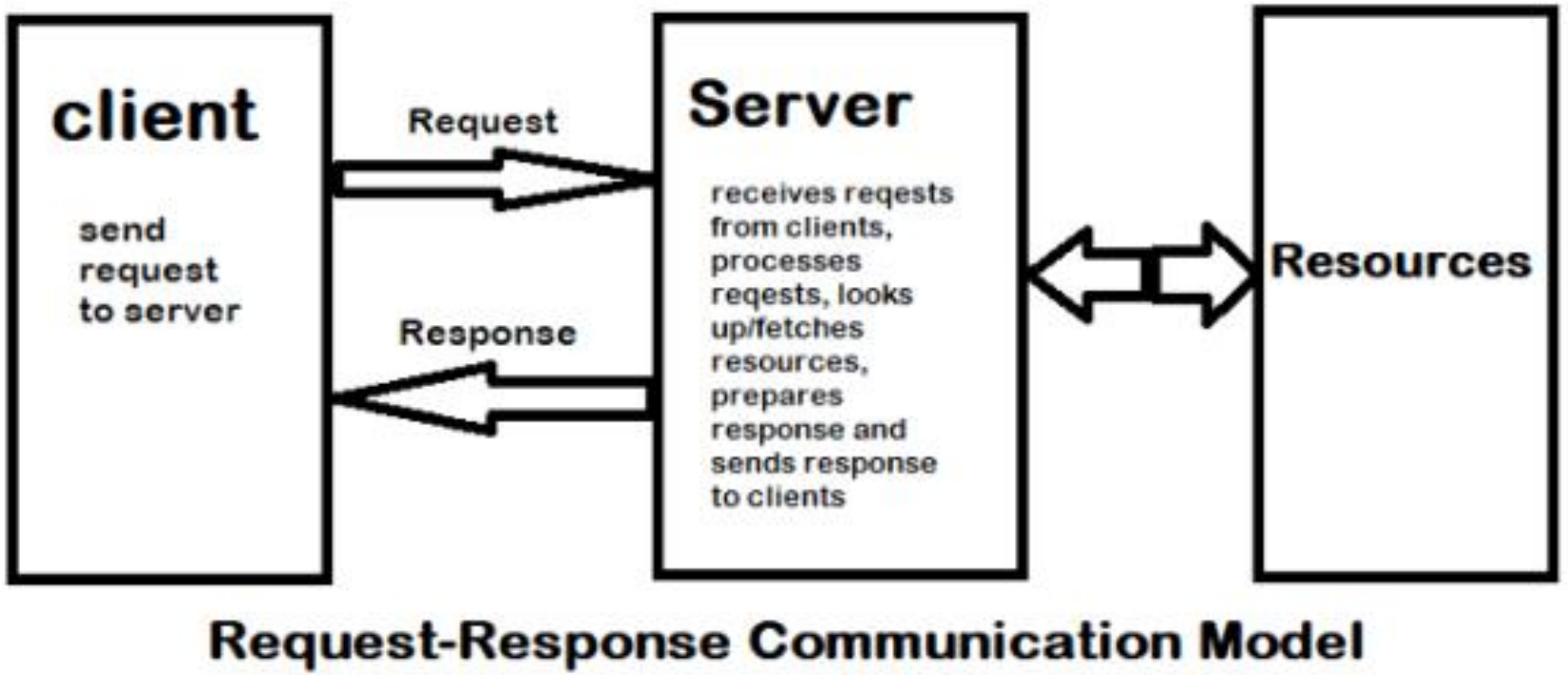
IoT Communication Models:

- A) Request-Response
- B) Publish-Subscribe
- C)Push-Pull
- D) Exclusive Pair

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

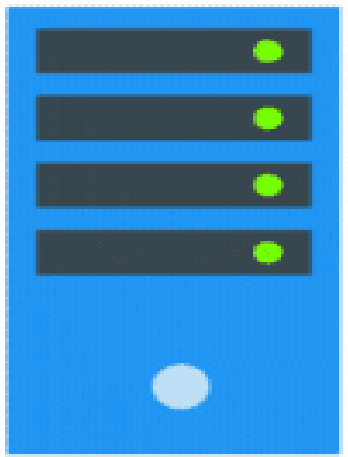


Request-Response Communication Model

REQUEST-RESPONSE ARCHITECTURE

HEY, SERVER!
I WANT TO BOOK
THAT APARTMENT IN
PARIS.

```
{
  apartmentId: "4f233fca",
  startDate: ...,
  endDate: ...,
  ...
}
```



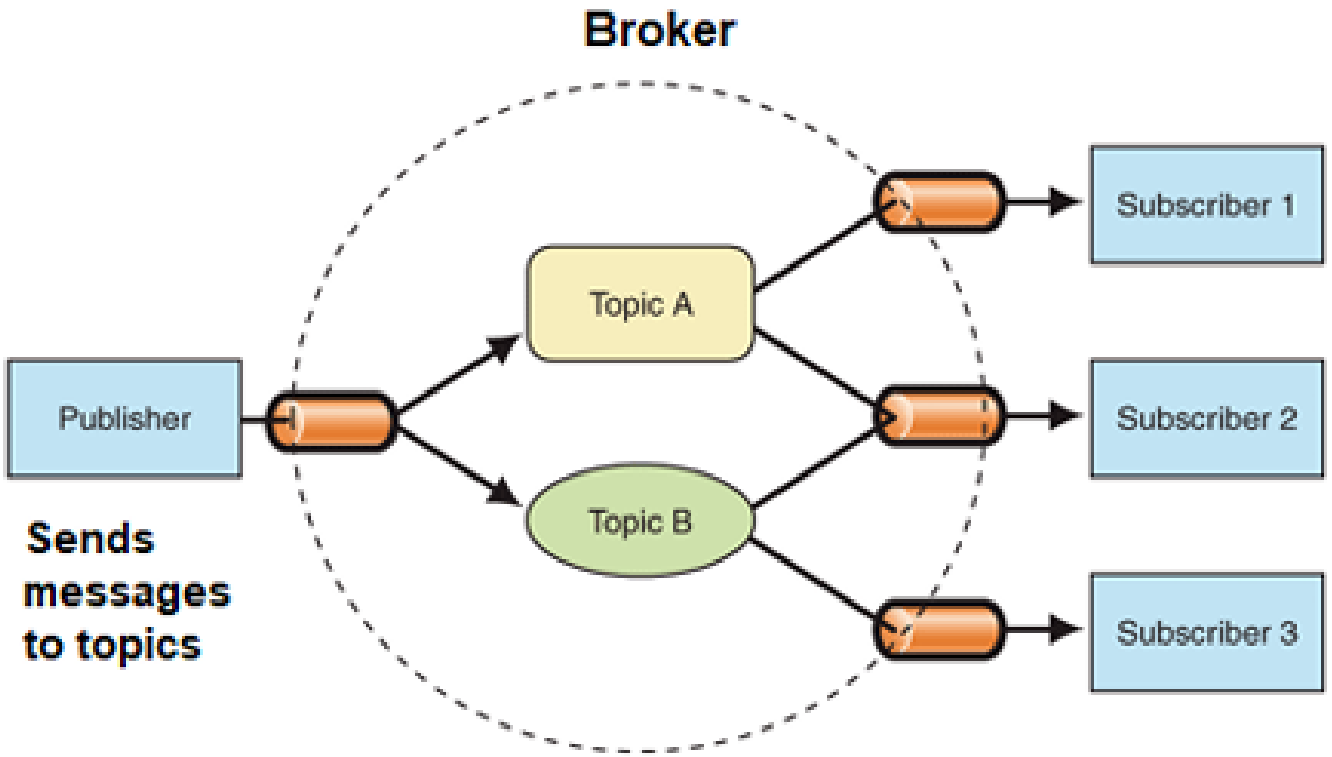
JOHN DOE (CLIENT)

SERVER

B) Publish-Subscribe communication model:

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

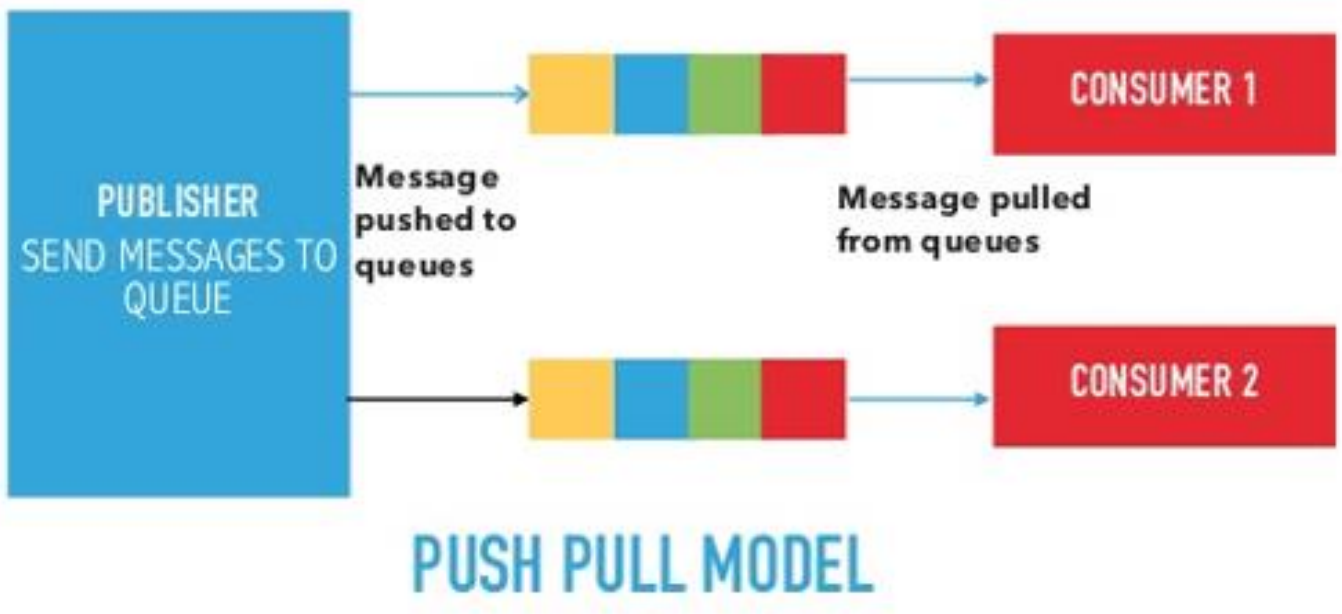
B) Publish-Subscribe communication model:



C) Push-Pull communication model:

- a. 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.
- b. Queues help in decoupling the messaging between the producers and consumers.
- c. 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.

C) Push-Pull communication model:



D) Exclusive Pair communication model:

- a. Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and server.
- b. Once the connection is setup it remains open until the client sends a request to close the connection.
- c. Client and server can send messages to each other after connection setup.

D) Exclusive Pair communication model:

3)IoT Communication APIs:

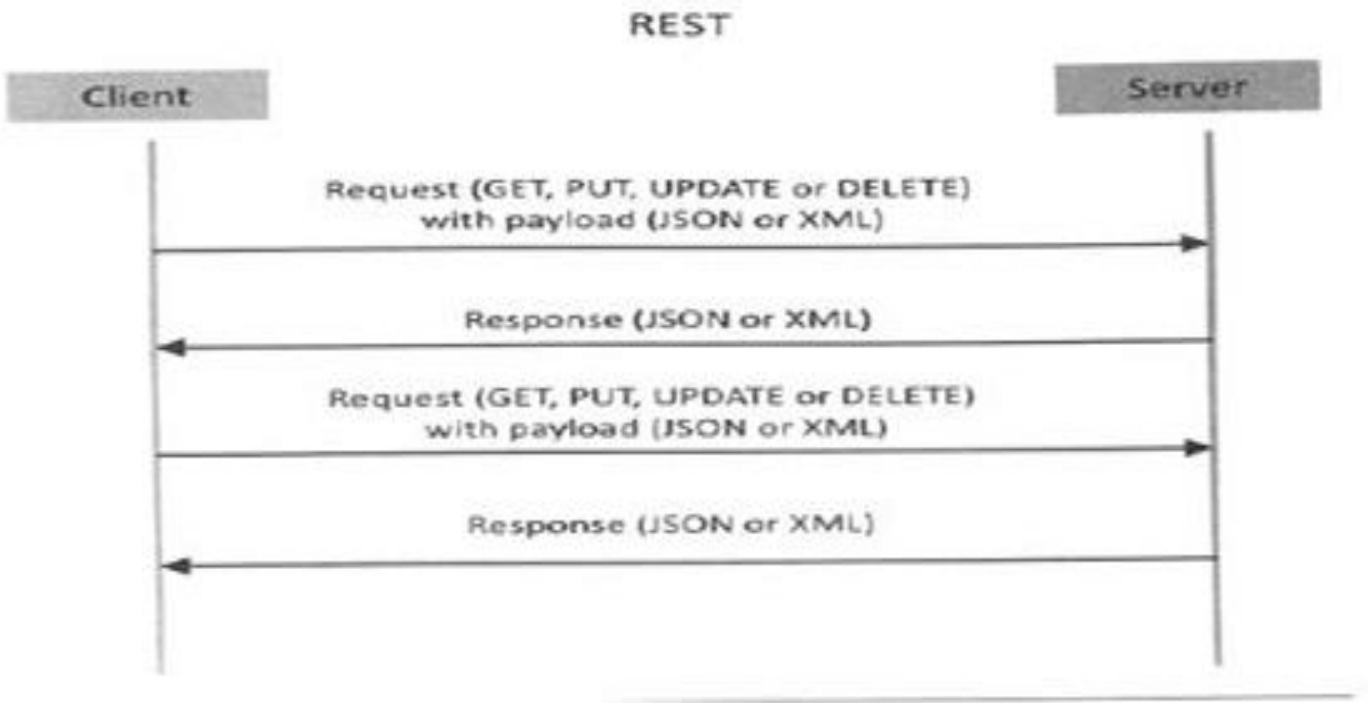
- a) REST based communication APIs(Request-Response Based Model)

  b)WebSocket based Communication APIs(Exclusive PairBasedModel)

Request-Response model used by REST:

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

Request-Response model used by REST:

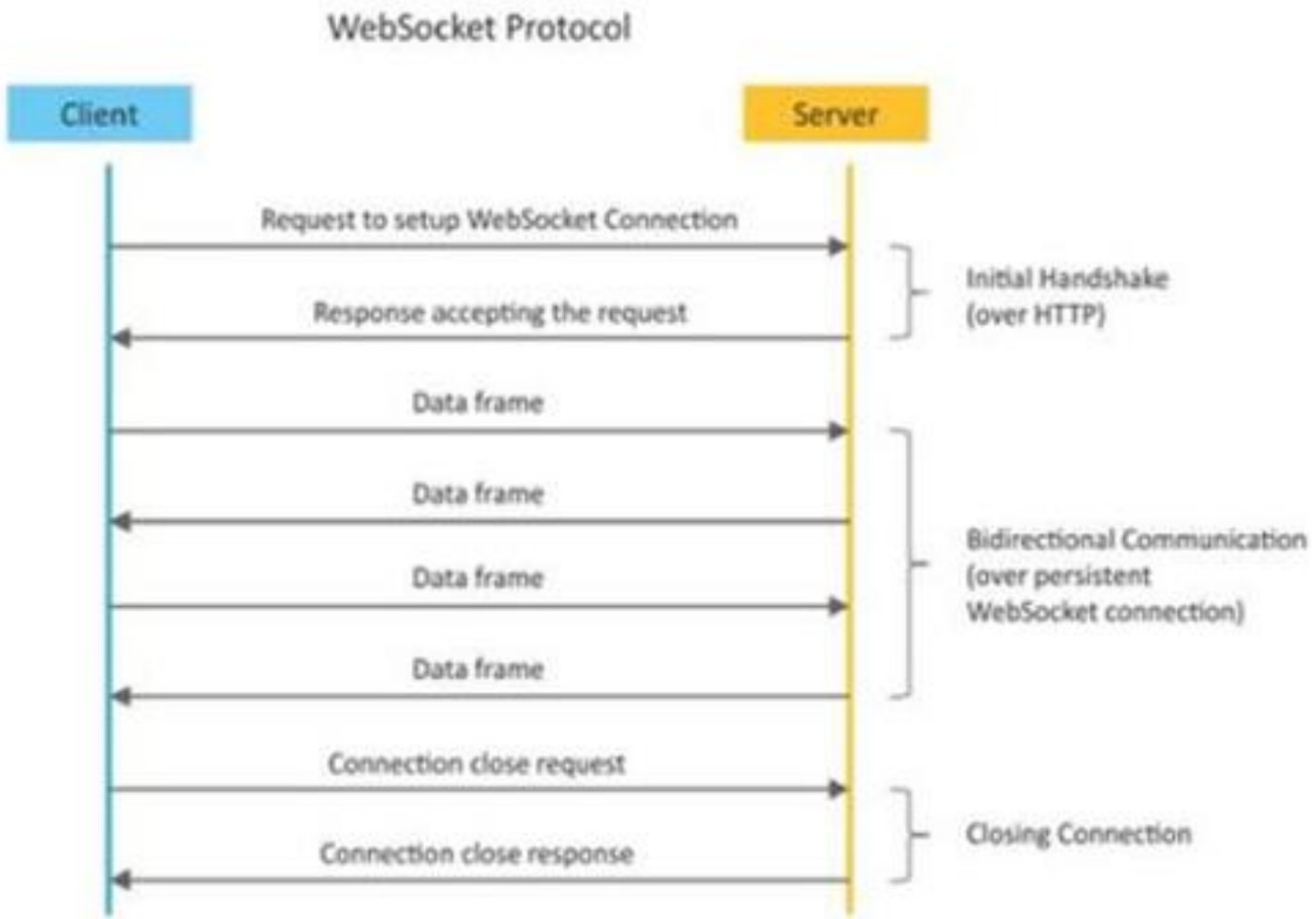


Request-response model used by REST

b) WebSocket Based Communication APIs:

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

b) WebSocket Based Communication APIs:



IoT Enabling Technologies

- IoT is enabled by several technologies including Wireless Sensor Networks, Cloud Computing, Big Data

Wireless Sensor Networks

- A wireless sensor network 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 co-ordinator.
- The coordinator collects the data from all the nodes. Coordinator also acts as a gateway that connects the WSN to the internet.

WSNs used in IoT systems

- Weather Monitoring System: in which nodes collect temp, humidity and other data, which is aggregated and analyzed.
- **Indoor air quality monitoring systems**: to collect data on the indoor air quality and concentration of various gases.
- **Soil Moisture Monitoring Systems:** to monitor soil moisture at various locations.
- **Surveillance Systems:** use WSNs for collecting surveillance data(motion data detection).
- **Smart Grids :** use WSNs for monitoring grids at various points.
- **Structural Health Monitoring Systems:** Use WSNs to monitor the health of structures(building, bridges) by collecting vibrations from sensor nodes deployed at various points in the structure.

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”.
- 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 services are offered to users in different forms.

- Infrastructure-as-a-service(IaaS):Provides users the ability to provision computing and storage resources. These resources are provided to the users as a virtual machine instances and virtual storage.
- Platform-as-a-Service(PaaS): Provides users the ability to develop and deploy application in cloud using the development tools, APIs, software libraries and services provided by the cloud service provider.
- Software-as-a-Service(SaaS): Provides the user a complete software application or the user interface to the application itself. The cloud service provider manages the underlying cloud infrastructure including servers, network, operating systems, storage, and application software.

Big data Analysis

- Big data is defined as collections of data sets whose volume , velocity or variety is so large that it is difficult to store, manage, process and analyze the data using traditional databases and data processing tools.
- Sensor data generated by IoT systems.
- Machine sensor data collected from sensors established in industrial and energy systems.
- Health and fitness data generated IoT devices.
- Data generated by IoT systems for location and tracking vehicles.
- Data generated by retail inventory monitoring systems.

Big data Analysis

- The underlying characteristics of Big Data are
- Volume: There is no fixed threshold for the volume of data for big data. Big data is used for massive scale data.
- Velocity: Velocity is another important characteristics of Big Data and the primary reason for exponential growth of data.
- Variety: Variety refers to the form of data. Big data comes in different forms such as structured or unstructured data including test data, image , audio, video and sensor data .

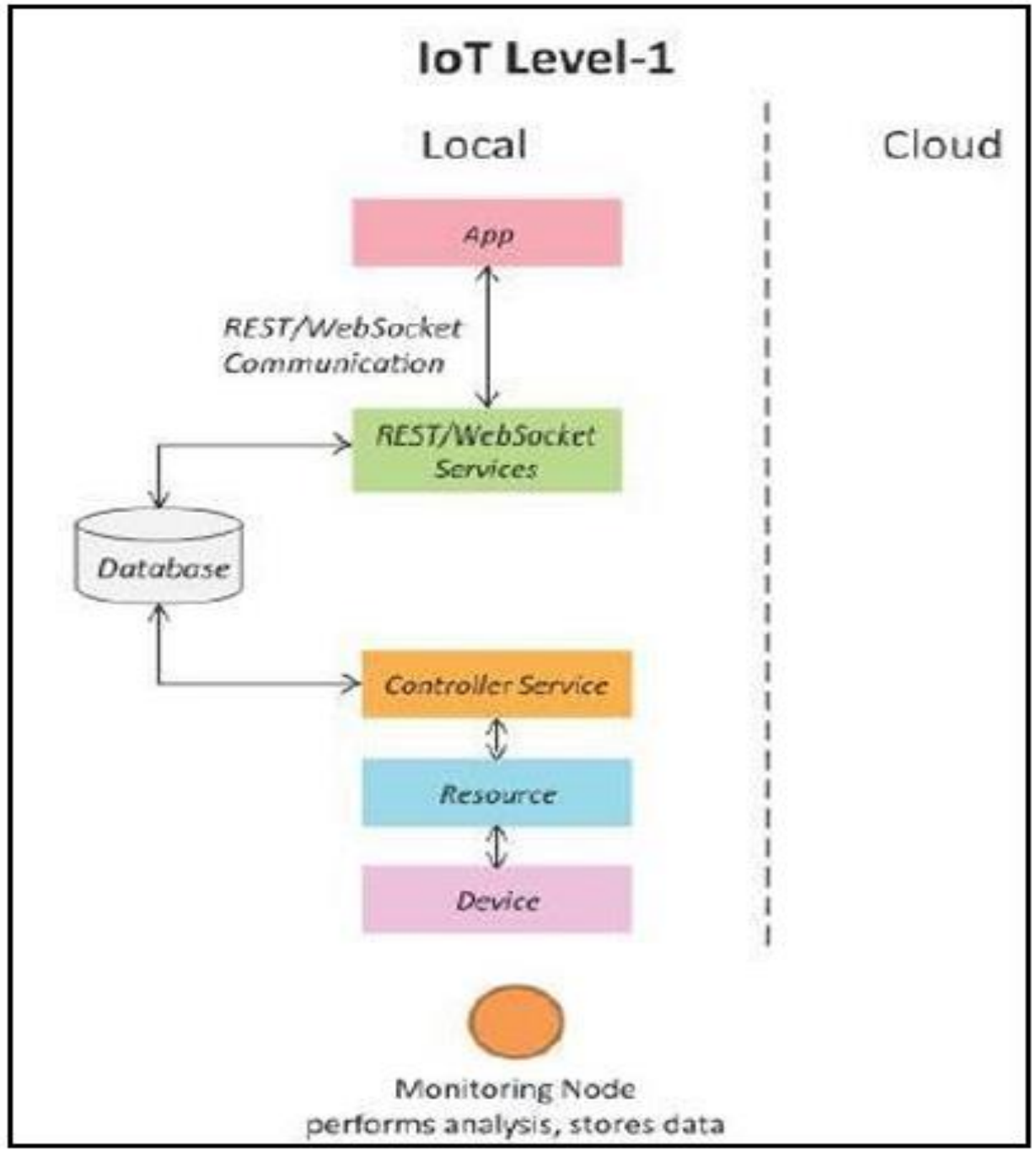
Communication Protocols:

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


Serial communication
We want less connections...
TX
RX
1000110
0
makeagif.com

IOT Levels and Deployment Templates.

- **IoT Level-1**
  - Level-1 IoT systems has a single node that performs sensing and/or actuation, stores data, performs analysis and host the application.
  - Suitable for modeling low cost and low complexity solutions where the data involved is not big and analysis requirement are not computationally intensive.


IoT Level-1
Local
Cloud
App
REST/WebSocket
Communication
REST/WebSocket
Services
Database
Controller Service
Resource
Device
Monitoring Node
performs analysis, stores data

LOCAL SPACE

Application

Electronic Technocrat
Technophilic Holmes

Websocket/REST based communication

Websocket/REST Web Service

Database

Controller Service

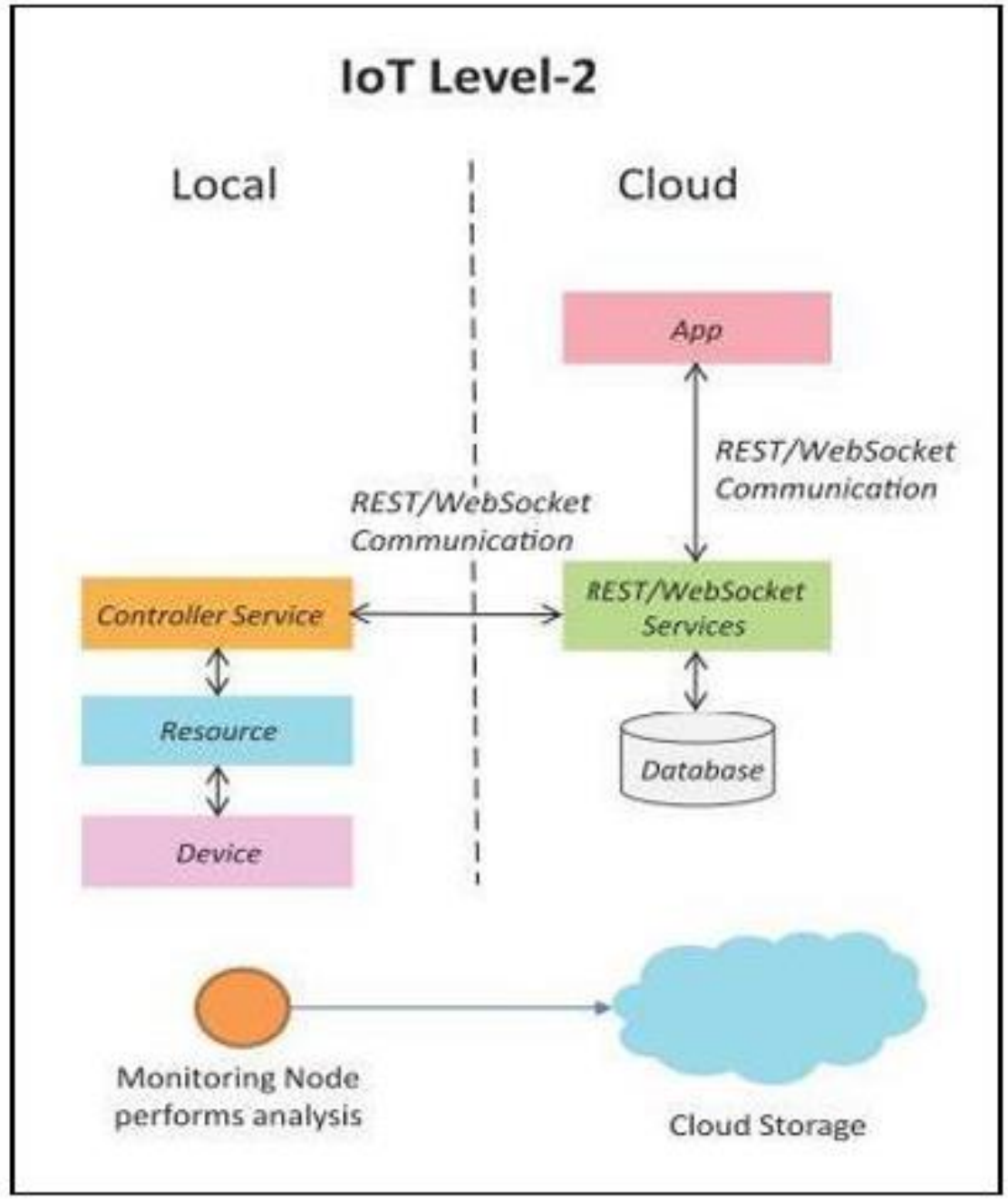
Resource

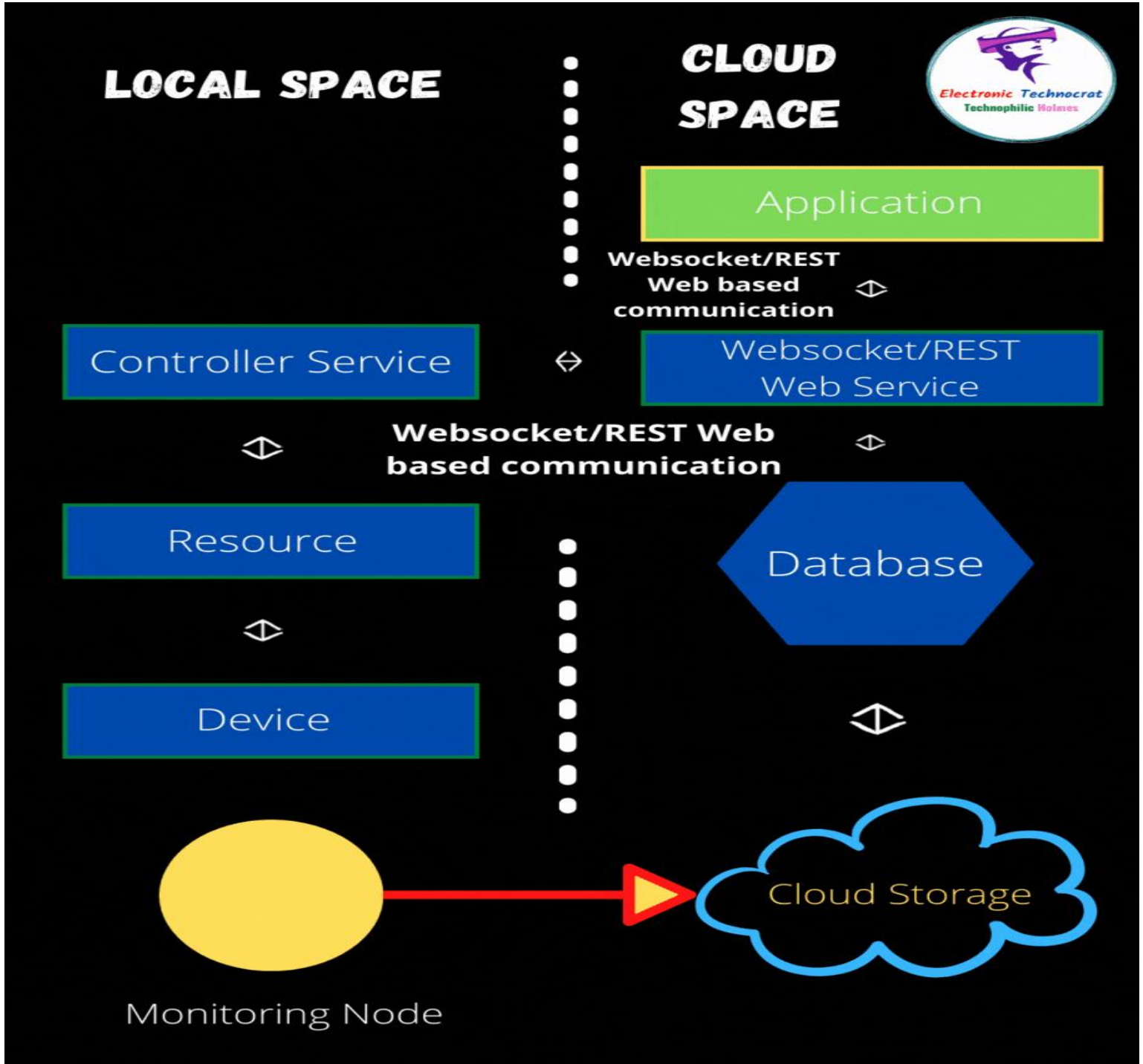
Device

Monitoring Node

IoT Level 2

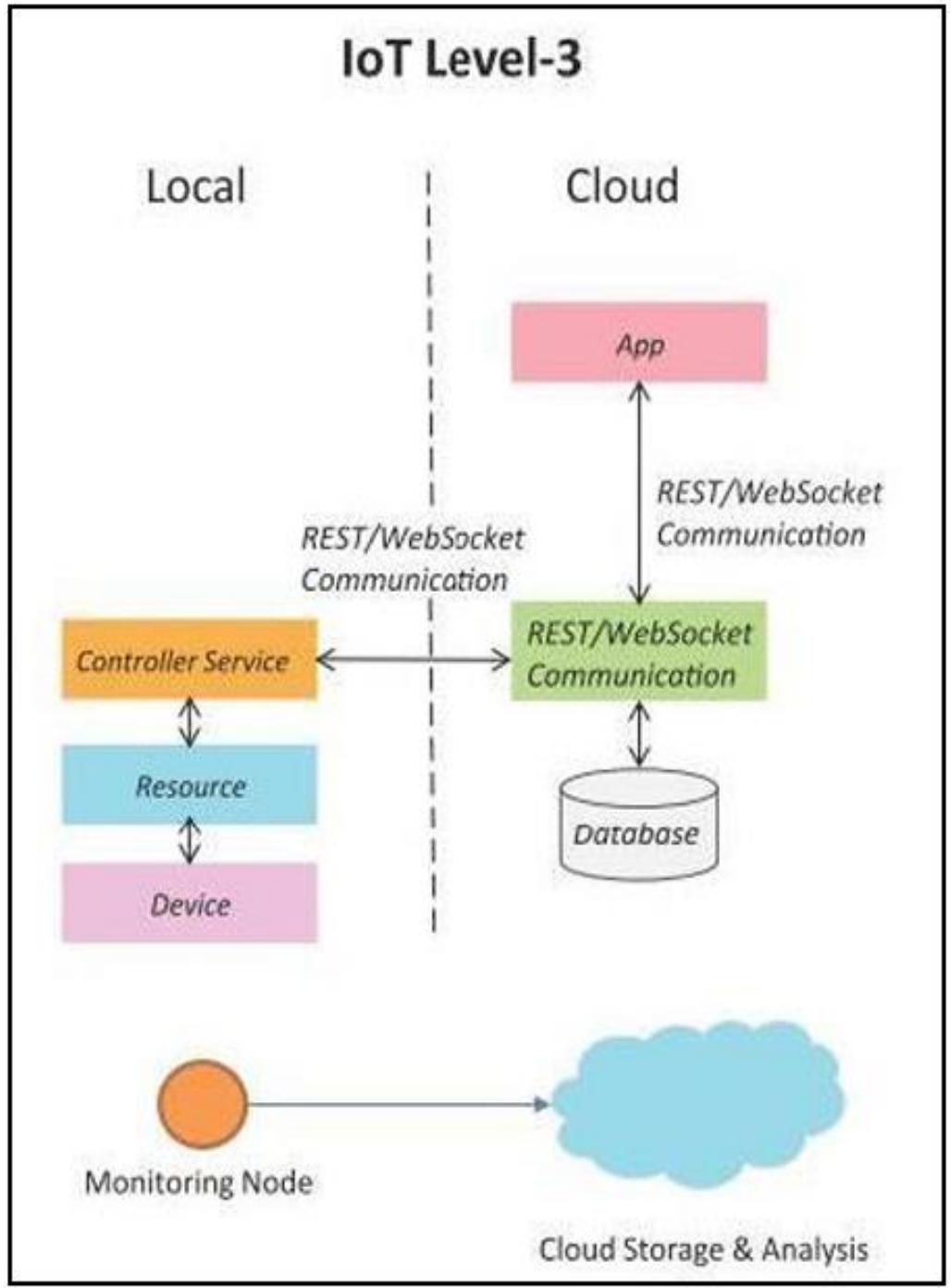
- IoT Level2 has a single node that performs sensing and/or actuating and local analysis .
- Data is stored in cloud and application is usually cloud based.
- Level2 IoT systems are suitable for solutions where data are involved is big, however, the primary analysis requirement is not computationally intensive and can be done locally itself.


IoT Level-2
Local
Cloud
App
REST/WebSocket
Communication
REST/WebSocket
Communication
Controller Service
REST/WebSocket
Services
Resource
Database
Device
Monitoring Node
performs analysis
Cloud Storage


LOCAL SPACE
CLOUD SPACE
Electronic Technocrat
Technophilic Holmes
Application
Websocket/REST
Web based
communication
Controller Service
Websocket/REST
Web Service
Websocket/REST Web
based communication
Resource
Database
Device
Cloud Storage
Monitoring Node

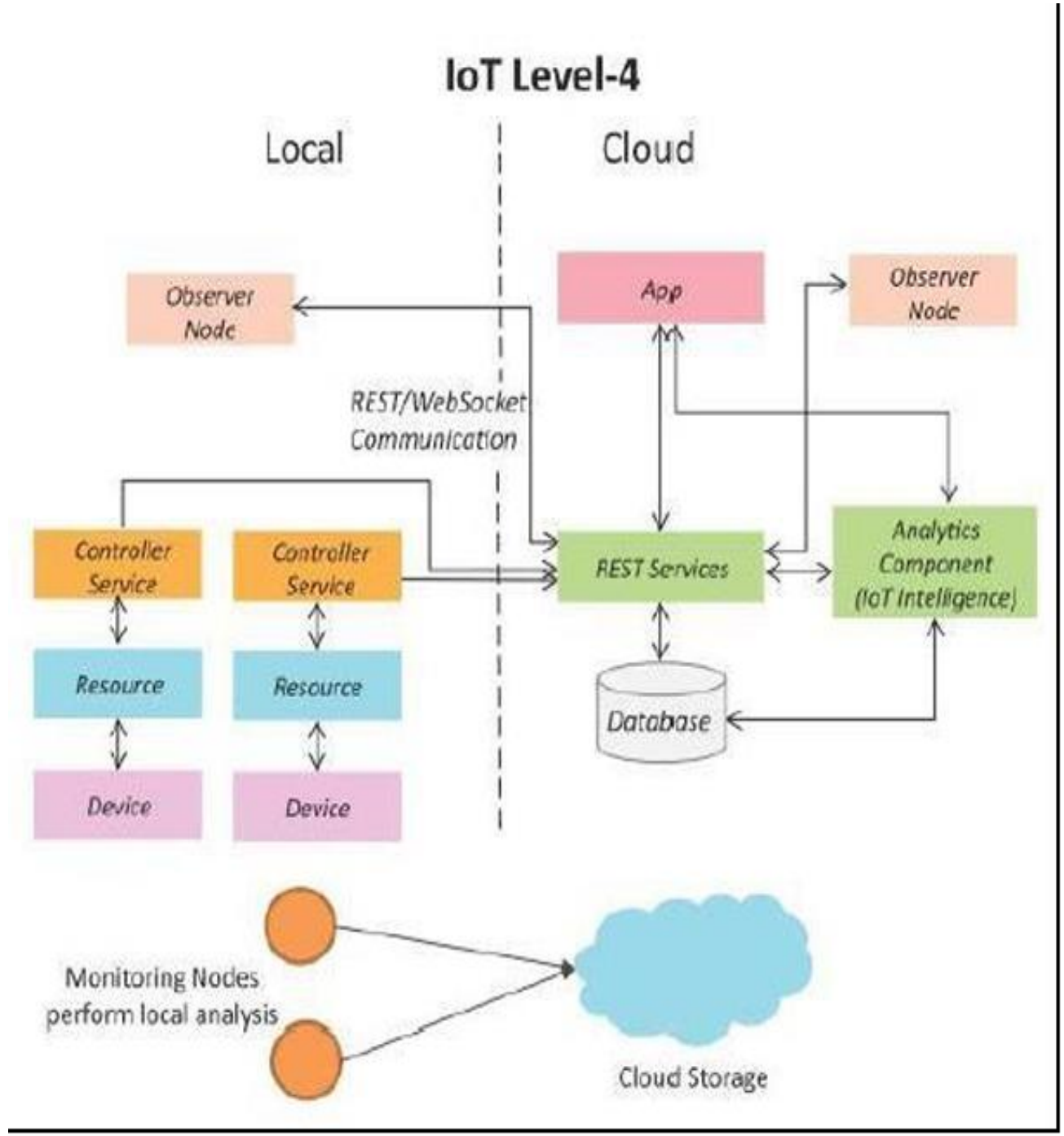
IoT Level 3

- This System has a single node.
- Data is stored and analyzed in the cloud application is cloud.
- Level3 IT systems are suitable for solutions where the data involved is big and analysis requirements are computationally intensive.


IoT Level-3
Local
Cloud
App
REST/WebSocket
Communication
REST/WebSocket
Communication
REST/WebSocket
Communication
Controller Service
Resource
Device
Database
Monitoring Node
Cloud Storage & Analysis

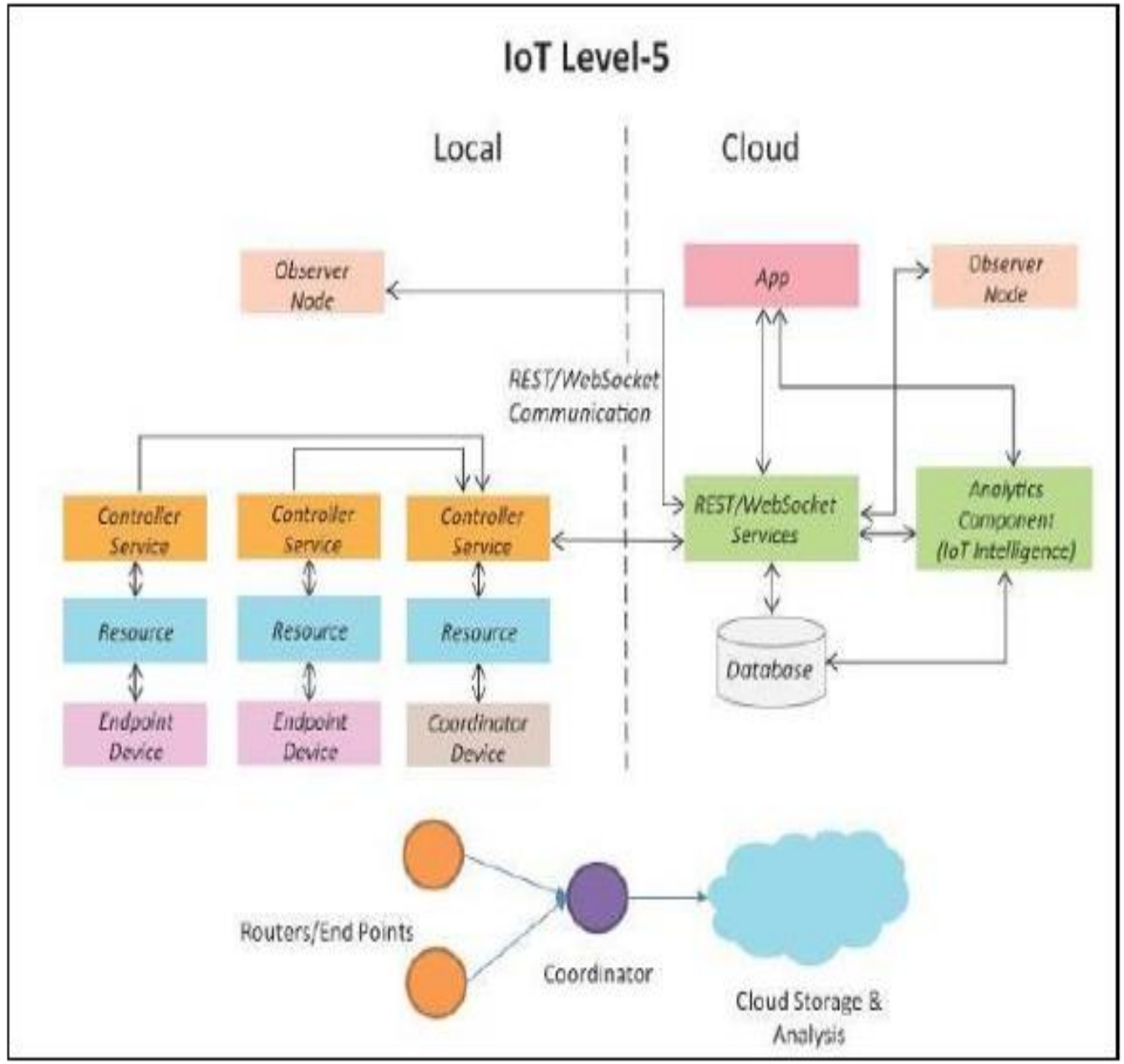
IoT Level 4

- This System has multiple nodes that perform local analysis. Data is stored in the cloud and application is cloud .
- Level4 contains local and cloud based observer nodes which can subscribe to and receive information collected in the cloud from IoT devices.
- Level 4 IoT systems are suitable for solutions where multiple nodes are required, the data involved in big and the analysis requirements are computationally intensive.


IoT Level-4
Local
Cloud
Observer Node
App
Observer Node
REST/WebSocket Communication
Controller Service
Controller Service
REST Services
Analytics Component (IoT Intelligence)
Resource
Resource
Database
Device
Device
Monitoring Nodes perform local analysis
Cloud Storage

IoT Level 5

- System has multiple end nodes and one coordinator node. 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. Level5 IoT systems are suitable for solution based on wireless sensor network, in which data are high intensive.


IoT Level-5
Local
Cloud
Observer Node
App
Observer Node
REST/WebSocket Communication
Controller Service
Controller Service
Controller Service
REST/WebSocket Services
Analytics Component (IoT Intelligence)
Resource
Resource
Resource
Database
Endpoint Device
Endpoint Device
Coordinator Device
Routers/End Points
Coordinator
Cloud Storage & Analysis

Applications of IOT

- **1) Home Automation:**
  - **a) Smart Lighting:** helps in saving energy by adapting the lighting to the ambient conditions and switching on/off or diming the light when needed.
  - **b) Smart Appliances:** make the management easier and also provide status information to the users remotely.
  - **c) Intrusion Detection:** use security cameras and sensors(PIR sensors and door sensors) to detect intrusion and raise alerts. Alerts can be in the form of SMS or email sent to the user.

1) Home Automation

- **d) Smoke/Gas Detectors:** Smoke detectors are installed in homes and buildings to detect smoke that is typically an early sign of fire. Alerts raised by smoke detectors can be in the form of signals to a fire alarm system. Gas detectors can detect the presence of harmful gases such as CO, LPGetc.,

2) Cities:

- **a) Smart Parking:** make the search for parking space easier and convenient for drivers. Smart parking are powered by IoT systems that detect the no. of empty parking slots and send information over internet to smart application back ends.
- **b) Smart Lighting:** for roads, parks and buildings can help in saving energy.
- **c) Smart Roads:** Equipped with sensors can provide information on driving condition, travel time estimating and alert in case of poor driving conditions, traffic condition and accidents.

2) Cities:

- **d) Structural Health Monitoring:** uses a network of sensors to monitor the vibration levels in the structures such as bridges and buildings.
- **e) Surveillance:** The video feeds from surveillance cameras can be aggregated in cloud based scalable storage solution.
- **f) Emergency Response:** IoT systems for fire detection, gas and water leakage detection can help in generating alerts and minimizing their effects on the critical infrastructures.

3) Environment:

- **a) Weather Monitoring:** Systems collect data from a no. of sensors attached and send the data to cloud based applications and storage back ends. The data collected in cloud can then be analyzed and visualized by cloud based applications.
- **b) Air Pollution Monitoring**: System can monitor emission of harmful gases(CO2, CO, NO, NO2 etc.,) by factories and automobiles using gaseous and meteorological sensors. The collected data can be analyzed to make informed decisions on pollutions control approaches.
- **c) Noise Pollution Monitoring:** Due to growing urban development, noise levels in cities have increased and even become alarmingly high in some cities. IoT based noise pollution monitoring systems use a no. of noise monitoring systems that are deployed at different places in a city. The data on noise levels from the station is collected on servers or in the cloud. The collected data is

Retail:

- **a) Inventory Management:** IoT systems enable remote monitoring of inventory using data collected by RFID readers.
- **b) Smart Payments:** Solutions such as contact-less payments powered by technologies such as Near Field Communication(NFC) and Bluetooth.
- **c) Smart Vending Machines:** Sensors in a smart vending machines monitors its operations and send the data to cloud which can be used for predictive maintenance.

Agriculture:

- a) Smart Irrigation: to determine moisture amount in the soil.
- b) Green House Control: to improve productivity.

Industry:

a) Machine diagnosis and prognosis.

b) Indoor Air Quality Monitoring.

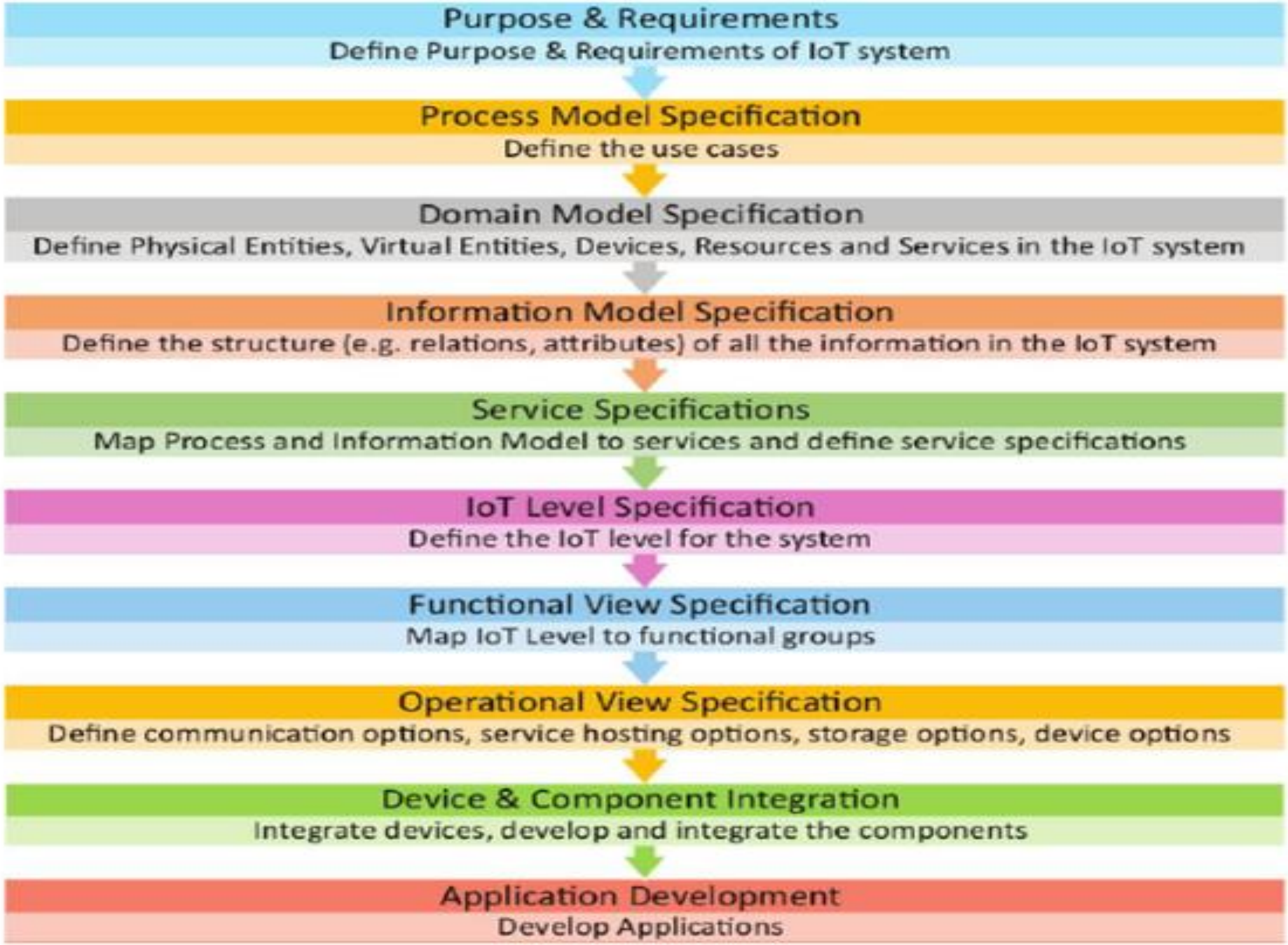
Health and Lifestyle:

a) Health & Fitness Monitoring

b) Wearable Electronics

Design Methodology for IOT Products.

- Designing IoT systems can be a complex and challenging task as these systems involve interactions between various components. A wide range of choices are available for each component. IoT designers often tend to design the system keeping specific products in mind.
- We will look at a generic design methodology which is independent of specific product, service or programming language.
- IoT systems designed with this methodology will have reduced design time, testing time, maintenance time, complexity and better interoperability.


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

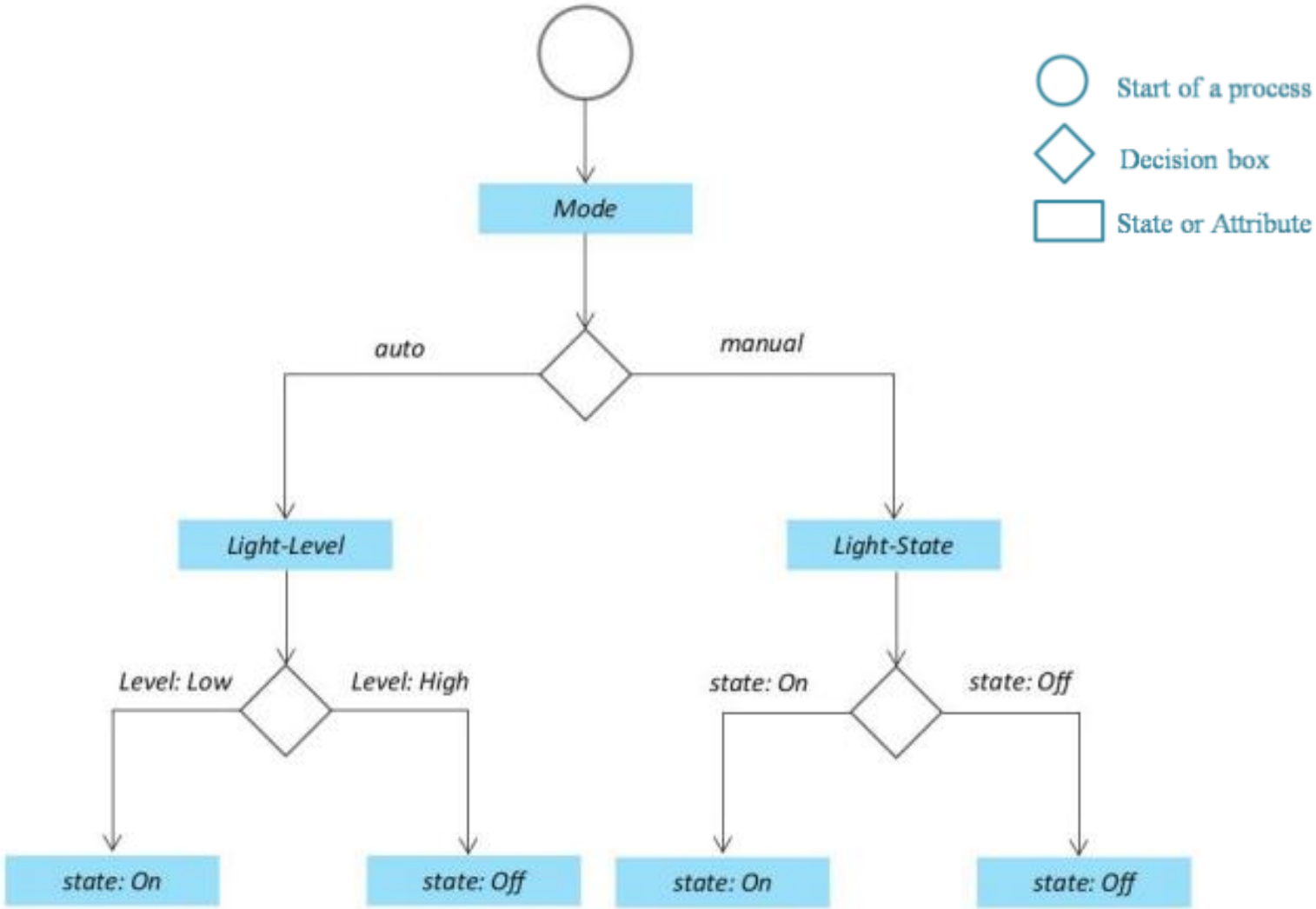
Purpose and Requirements Specification

- In this step, the system purpose, behavior and requirements are captured. Requirements can be:
  - Data collection requirements
  - Data analysis requirements
  - System management requirements
  - Security requirements
  - User interface requirements

For home automation system the purpose and requirements specification is as follows:

Purpose	A home automation system that allows controlling the lights remotely using a web application
Behavior	Home automation system should support two modes: auto and manual **Auto:** System measures the light level in the room and switches on the light when it is dark **Manual:** Allows remotely switching lights on and off
System Management	System should provide remote monitoring and control functions
Data Analysis	System should perform local analysis of the data
Application Deployment	Application should be deployed locally, but should be accessible remotely
Security	Should provide basic security like user authentication

Process Specification

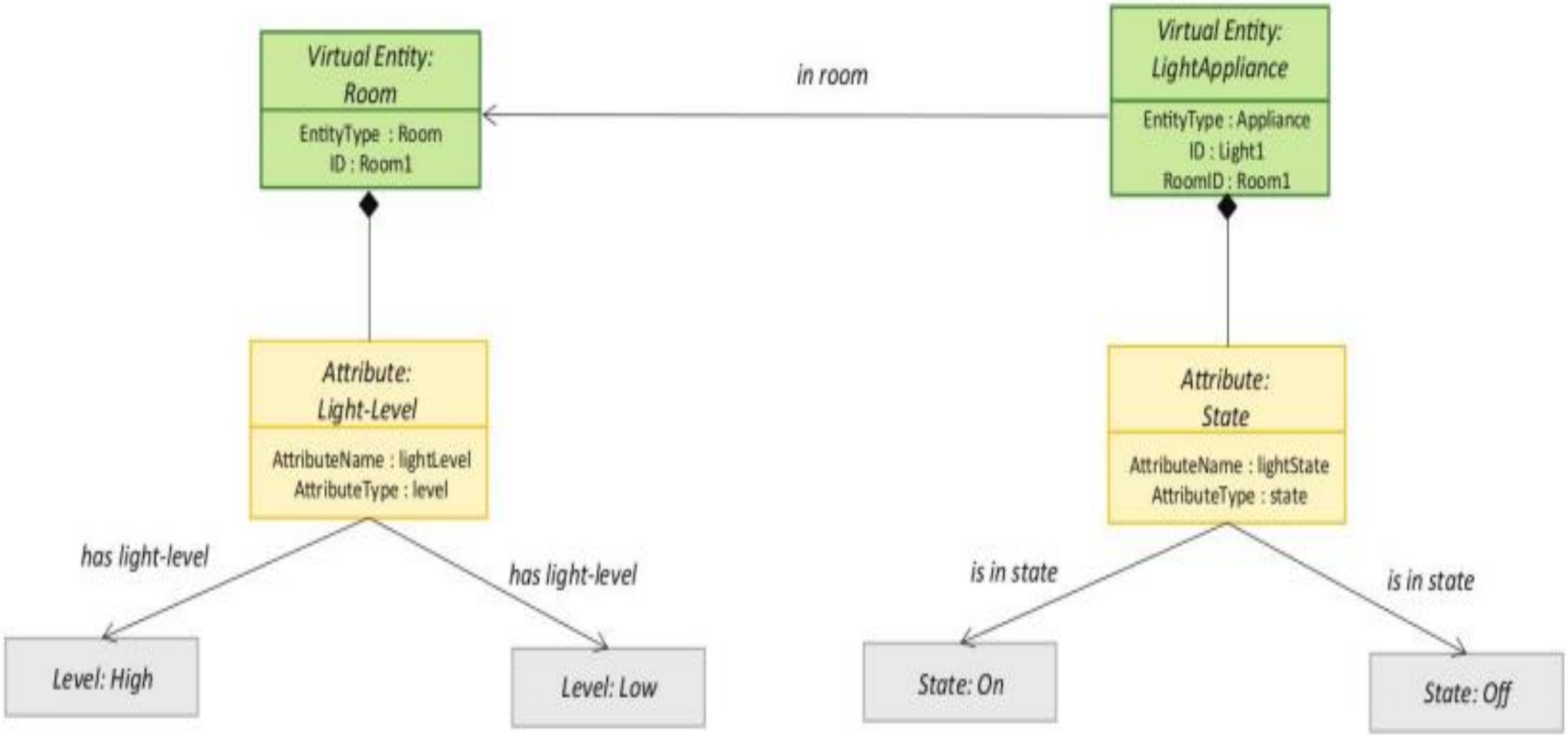


Domain Model Specification

- The domain model describes the main concepts, entities and objects in the domain of the IoT system to be designed.
- Domain model defines the attributes of the objects and relationships between objects. The domain model is independent of any specific technology or platform.

Information Model Specification

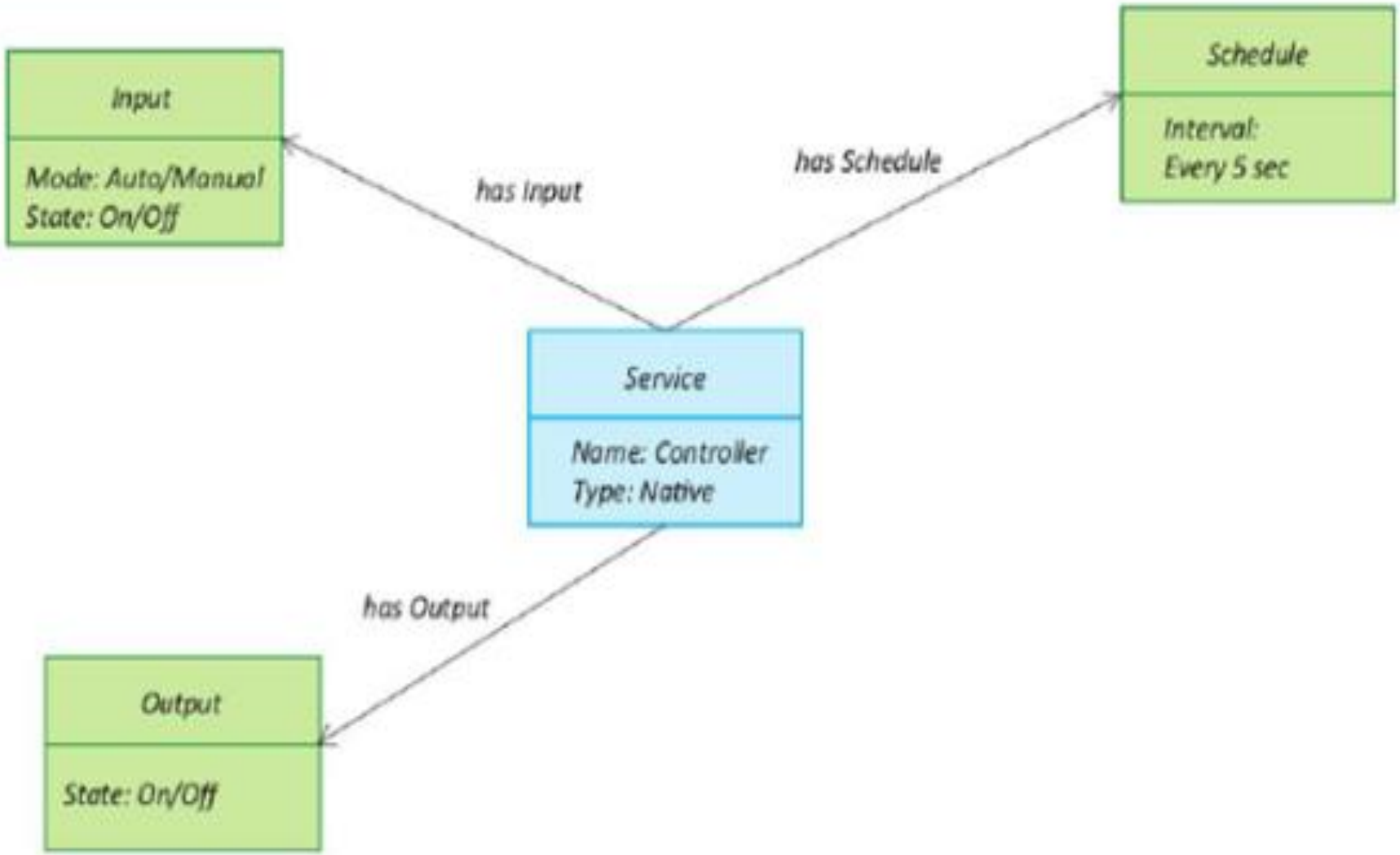
- Information model defines the structure of all the information in the IoT system. Does not describe how the information is stored and represented. To define the information model, we first list the virtual entities. Later more details like attributes and relationships are added.


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
is in state
is in state
Level: High
Level: Low
State: On
State: Off

Service Specifications

The service specification defines the following:

- Services in the system
- Service types
- Service inputs/output
- Service endpoints
- Service schedules
- Service preconditions
- Service effects


Input
Mode: Auto/Manual
State: On/Off
has Input
Schedule
Interval:
Every 5 sec
has Schedule
Service
Name: Controller
Type: Native
has Output
Output
State: On/Off