NO

DATE

Unit 1 - IoT

* SoC - System on Chip
  - It is a small integrated chip that contains all the req. components and circuits of a particular system.
  - components of SoC include CPU, GPU, Memory, I/O devices, etc.
  - Used in various devices
    - smartphones
    - IoT appliances
    - tablets
    - embedded system applications.

---

* Microcontroller ~~[Computer]~~ [CLERPP - VW]
  - Microcontrollers are compact, IC's that serve as the CPU of embedded systems.
  - Characteristics :

    i) Embedded System Integration : Microcontrollers are designed to be embedded within a larger system, often providing control and processing capabilities within a single chip.

NO

DATE

ii) Processing Power: While not as powerful as general-purpose microprocessors, microcontrollers are optimized for specific tasks and offer sufficient processing power for many apps.

iii) Peripheral Integration: They often integrate various peripherals such as timers, ADCs, DACs, communication interfaces (UART, I2C), and pulse-width modulation controllers, reducing the need for external components.

iv) Low Power Consumption: Typically designed to operate on minimal power, making them suitable for battery-powered or energy-efficient applications.

v) Real-Time Operation: Capable of executing tasks within predefined time constraints, making them suitable for time-critical applications.

vii) Cost-Effective: Generally cost-effective due to their integration of components onto a single chip, making them economical for mass production.

viii) Wide Range of Applications: Applications in various fields including consumer electronics, automotive systems, industrial automation, medical devices, IOT devices, robotics, etc.

viii) Variety of Architectures: Available in variety of architectures like ARM, AVR, PIC, and MOS, each with its own features and capabilities.

---

* SOC Architecture [PD-MEN-GPU]

- Processor:

i) Heart of the SoC, usually SoC contains at least one or more processor(s).

ii) It can be a microcontroller, microprocessor, or DSP.

iii) Usually, DSP is used in every SoC as a processor.

NO

DATE

- DSP (Digital Signal Processor):
  i) Included in SOC to perform signal processing operations (data collection, processing, etc.).
  ii) Also used for decoding the images.

- Memory:
  i) Used for the purpose of storage.
  ii) May be volatile or non-volatile memory.
  - RAM → SRAM, DRAM
  - ROM

- Encoder / Decoder:
  i) Used for purpose of interrupting information and converting it into codes.

- NIC:
  i) SOC has an internal interface or bus or network to connect all individual blocks.
  ii) NIC provides a connection of the network to the system.

- GPU :

i) Used in SoC to visualize the interface.

ii) Specially designed to speed up the operations related to image calculations.

iii) Basic blocks → Bus interface
→ Power Management Unit
→ Video Processing Unit
→ Graphics Memory Controller
→ Display interface

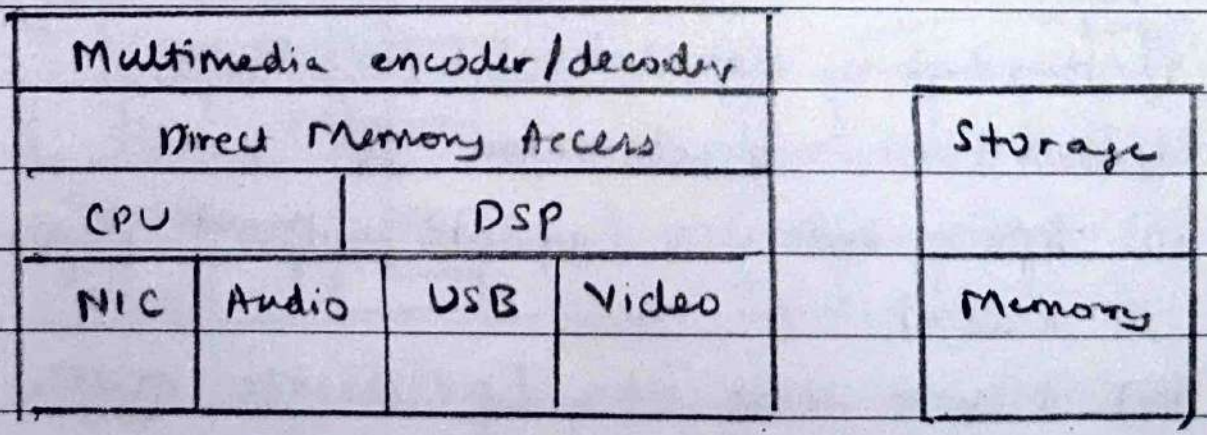
- Peripheral Devices

i) Externally connected devices (USB, HDMI, Wi-Fi, Bluetooth).

ii) Used in SoC to perform various operations.

- UART :

i) Used to transmit or receive data.



SoC Architecture

NO

DATE

* FPGA (Field Programmable Gate Array)

- 3 main components
  - I/O channels (at outermost boundaries)
  - Interconnections (paths between Logic blocks, depending on logic)
  - Logic blocks (inner blocks)
- Capable of reconfiguring the hardware to meet user requirement.
- Contains CLBs (configurable logic blocks) and a set of programmable interconnections.
- Any simple circuit or complex function can be implemented using FPGA.
- Full SoC containing multiple processors can be put on a single FPGA device.
- Programming
  - VHDL
  - Verilog
- Features :

i) Hardware simulation (prototyping)

ii) Hardware acceleration

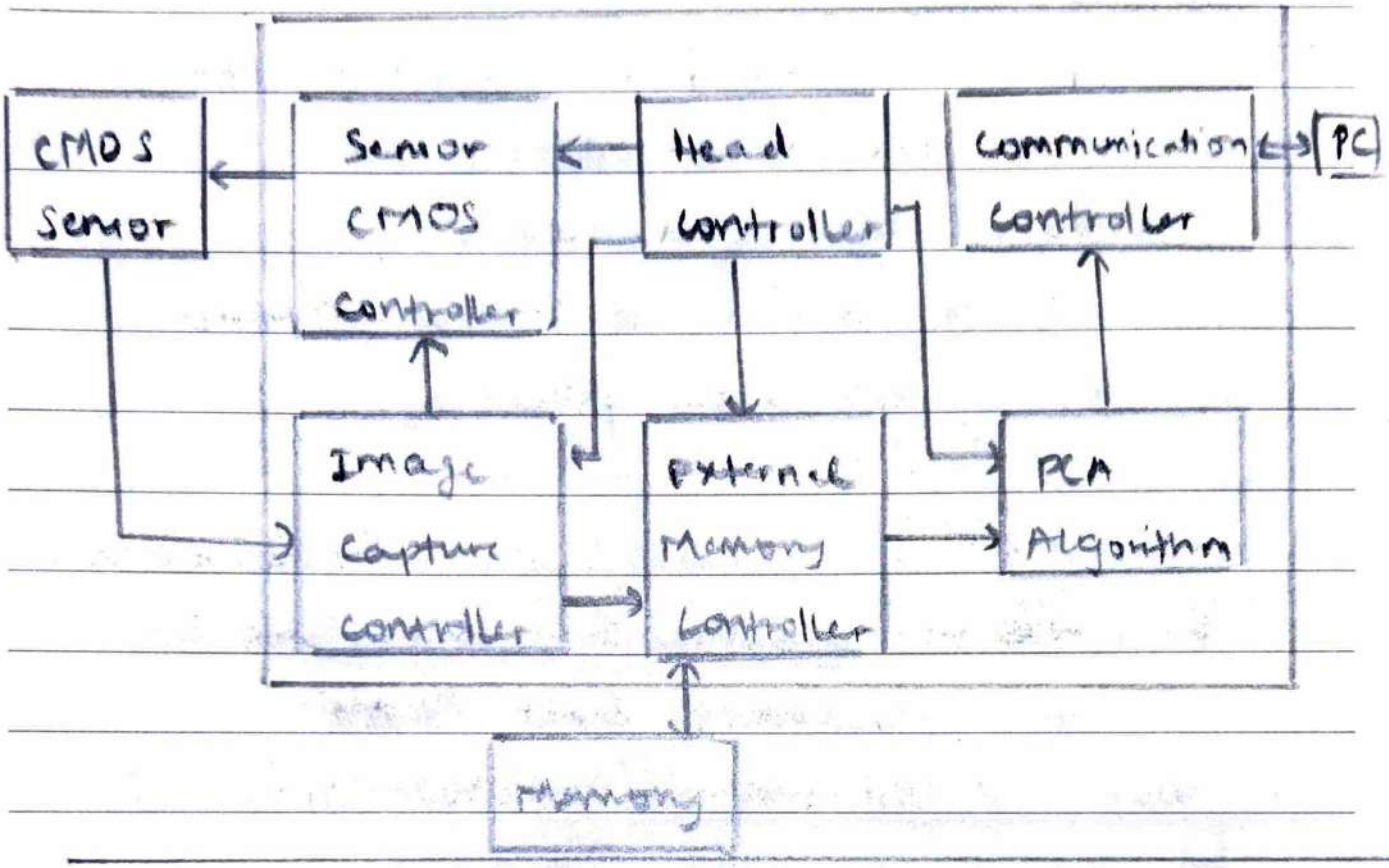
iii) Space avionics (update without physical access).

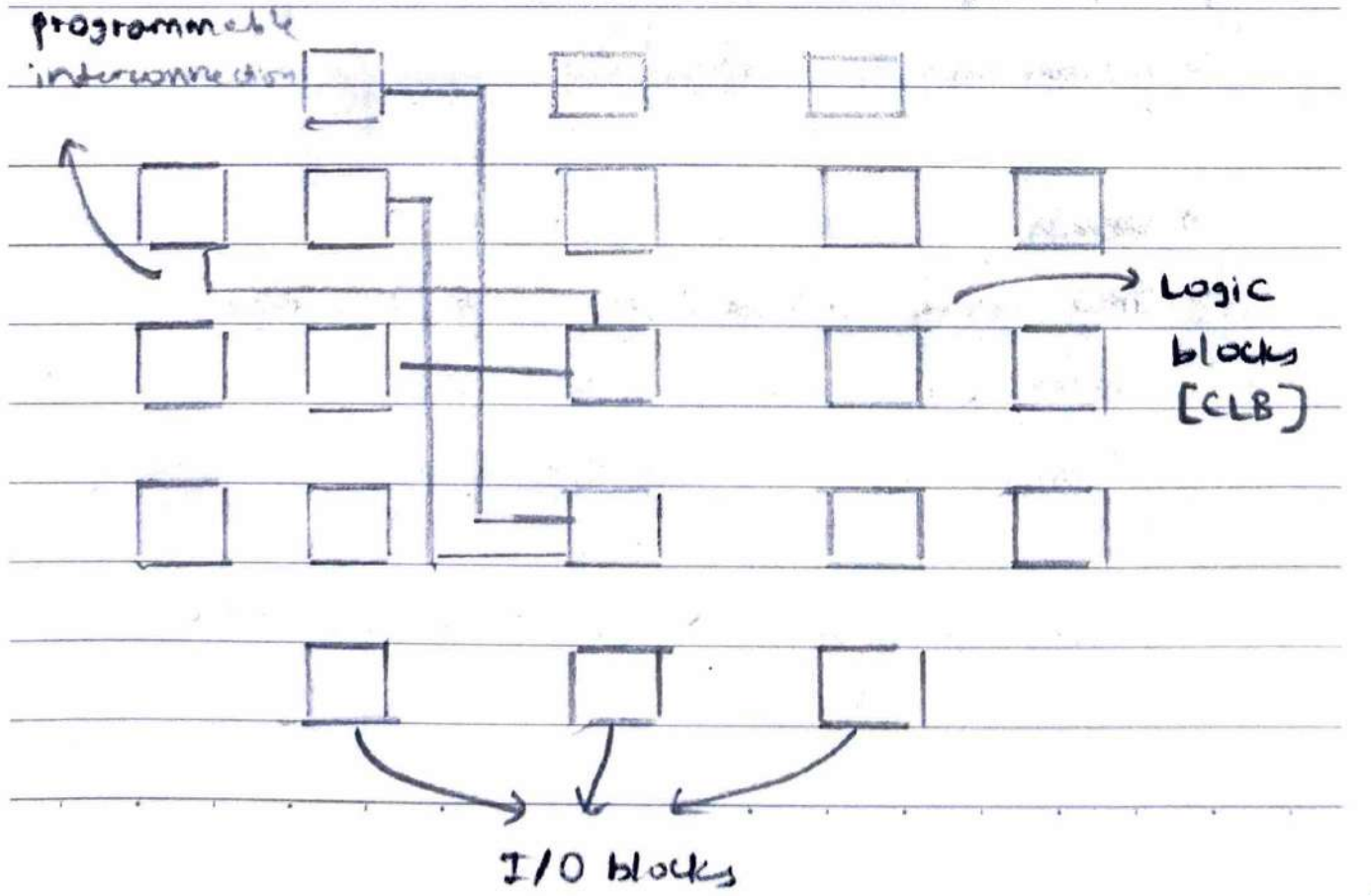
iv) Neural Networks (accelerate matrix multiplication).

NO

DATE

- Low power consumption and performance.
- High initial cost (development and hardware).





NO

DATE

* GPU (Graphics Processing Unit):

- A specialized processor, designed to accelerate graphics rendering.
- Can process many pieces of data simultaneously
- Useful for ML, video editing, gaming.
- May be
  - integrated into CPU
  - offered as discrete hardware unit
- Designed for parallel processing.
- Uses

i) Gaming
   - rendering graphics (2D and 3D)
   - higher resolution and faster frame rates

ii) Video editing and content creation
   - virtual streaming
   - support for large displays

iii) ML
   - high computational capability
   - incredible acceleration in handling workload.

NO

DATE

* APU (Accelerated Processing Unit)

- Combination of GPU and CPU on a single chip.
- Can handle both general computing and graphics-related task.
- Low cost
- Low power consumption (due to small space)
- Used in laptops to increase power factor.
- Used:
  i) Budget friendly gaming PCs.
  ii) e.g. AMD's Ryzen series, Intel Core with Graphics

---

* Compute Units

- Processing elements integrated into IOT devices to perform various computational task.
- Similar to host groups, with added feature of granularity allowing construction of clusterwide structures that mimic network architecture.
- Useful when running communication intensive parallel jobs spanning several hosts.

NO

DATE

- Uses :
  i) Queue
  ii) Host on which queue jobs can be run.
- Encode cluster network topology for jobs.
- Helps in reducing network latency.

---

* ARM 8
- ARM - Advanced RISC Machines
- Part of ARM family of general purpose 32-bit microprocessors, which offer very low power consumption, and price for high-performance devices.
- Based on RISC principles (simpler than CISC)
- Fully static CMOS implementation (increased clock speed).
- Inclusion of branch predicting prefetch unit.
- User optimizable
- Provides fast-onchip memory.
- 32-bit address bus. (can be operated in 26-bit modes for backward compatibility).

* ARM6 Instruction Set

- Uses ARM instruction set.
- Comprises 11 basic instruction types:
  - 2 for high speed operations on data
  - 3 control data transfer between the registers and memory.
  - 3 adjust the flow, privelege level, and system control parameters of execution.
  - 3 dedicated to the control of coprocessors.
- Advantages:
  - i) Straightforward to use when ALP.
  - ii) Good target for compilers of many different high level languages.
  - iii) Employs a prefetch buffer, along with instruction and data pipelining. (so all parts of processing and memory systems can operate continuously).
  - iv) When instruction A is being executed:
    - B is being decoded.
    - C is being fetched from the Prefetch Buffer.
    - Further instructions are being prefetched from memory.
    - Data for prev. instruction is being R/W.

NO

DATE

* ARM 8 Architecture

- Consists of
  - core
  - Prefetch Unit (PU)
  - fast on-chip memory (provide core or PU with two words of data on each cycle)

- Prefetch Unit (PU)

i) Prefetches and buffers instructions.

ii) Makes use of extra bandwidth by removing some of the branches from the instruction stream altogether (thus reducing cycles per instruction).

iii) Predicts whether or not the branch will be taken.

iv) If taken, its destination address is calculated, further instructions are fetched from there.

v) Some branches cannot be predicted (as prediction takes one or more cycles), ∴ the PU is occasionally empty.

vi) 65% of Branches are predictable.

• Core :

i) Compared to earlier ARM processors, it the extension of the data pipeline, making the ARM8 a processor with a 5-stage pipeline.

ii) The adder and shifter operate in parallel rather than in series.

iii) The ARM8 multiplier is bigger than that of ARM7 and operates on 8 bits per cycle.

* ARM8 Hardware Configurations

NO

DATE

* ARM 6 Blocks

- Core
  - Register decode and control logic
  - Register bank
  - Multiplier
  - Write data pipeline
  - Shifter / ALU
  - PSRs (Program Status Register
  - coprocessors
  - Address buffer
- PU
  - Instruction FIFO
  - PC FIFO
  - PC Incrementer

* ARM6 Operating Modes

- Supports
  - byte (8-bit)
  - half-word (16-bit)
  - word (32-bit)

  } data types
- Instructions
  - exactly one word
  - aligned to 4-byte boundaries
- 7 modes: [UFI-S-SAU]

1) User Mode (usr) : Normal program execution state.

ii) FIQ Mode (fiq): Used for fast or high priority interrupt handling.
iii) IRQ Mode (irq): Used for general purpose interrupt handling.
iv) Supervisor Mode (svc): Protected mode for the OS.
v) System Mode (sys): Privileged user mode for the OS.
vi) Abort Mode (abt): Entered after a data or instruction prefetch abort.
vii) Undefined Mode (und): Entered when an undefined instruction is executed.

---

* Introduction to Raspberry Pi
- High performance, low cost microcontroller.
- CPU: Dual ARM Cortex M0 (133 MHz)
- Memory:
  i) 264 KB on-chip SRAM in 6 banks
  ii) Upto 16MB of off-chip flash memory via QSPI bus.
- Architecture:
  1) DMA (Direct Memory Access) Controller for data transfer operations.

NO

DATE

ii) Fully connected AHB (Advanced High Performance Bus) crossbar

iii) Interpolator and integer divider peripherals.

iv) On-chip programmable LDO (low dropout regulator) to generate core voltage.

v) Two on-chip PLLs (Phase locked loops) to generate USB and core clocks.

- Interfacing : 30 GPIO pins (4 as analogue inputs).
- Peripherals :

i) 2 UARTs

ii) 2 SPIs controllers

iii) 2 I2C controller

iv) 16 PWM channels

v) 1 USB 1.1 controller

vi) 8 PIO (Programmable I/O) state machines. for various I/O functions.

vii) 4 ADC inputs

NO

DATE

* Preparing your Raspberry Pi

i) Prepare the Hardware:
- Insert microSD card.
- Connect peripherals to appropriate ports.
- Stable surface and ventilated.

ii) Download OS:
- Visit website and select distribution.
- Download system image (compatible).

iii) Flash OS Image:
- Use tool (Etcher) to flash OS image on microSD.

iv) Boot up RPi:
- Insert microSD.
- Connect to power source
- Initial booting process (bgo)

v) Initial Configuration:
- Follow prompts.
- Set settings.

vi) Update System:
- Open terminal (or via ssh)
- apt

NO

DATE

vii) Additional config

viii) Reboot

ix) Test Config

---

* Programming with Raspberry Pi

i) Install OS (Raspbian with desktop)

ii) configure Pi (basic steps on terminal)

iii) Run the config tool:

- Change user password
- Network options
- Boot options
- Localization options
- Interfacing options
- Overclock
- Advanced options
- Update
- About R-Pi

iv) Use Python 3

v) Install pip

vi) write program.

vii) Connect hardware as per requirement.

viii) Run the program.

* Raspberry Pi and Linux

- sudo
- raspi-config
- ifconfig / ip
- hostname
- passwd
- apt
- dpkg
- apt-get
- systemctl
- journalctl
- iptables / ufw
- nano / vi / vim
- ssh-keygen
- ssh-copy-id
- wget / curl
- scp
- rsync
- chmod / chown
- adduser / deluser
- uname

---

* Introduction to Node.js

- Open-source, cross-platform JS runtime environment that executes JS code outside of a web browser.
- Built on the V8 JS engine (same as Google Chrome).
- App runs in a single process.
- Enables writing server-side application, creating scalable and dynamic web apps.

NO

DATE

- Features:

i) Asynchronous and Event Driven: Efficiently handles multiple tasks simultaneously, making apps highly responsive.

ii) Single-Threaded and Event Loop: Uses single thread to handle multiple client requests, optimizing resource utilization and enabling high throughput.

iii) NPM: Access to a vast ecosystem of libraries and tools.

iv) Cross-Platform: Runs on various OSs, making it easy to develop and deploy apps across different environments.

v) FE/BE: Used for both server and client-side development, enabling developers to use JS across the entire software stack.

* Raspberry Pi Interface

• UART (Universal Asynchronous Receiver and Transmitter):

i) Used for serial communication.

ii) U → applied to any transmitter / receiver.
A → cannot use clock signal for communication of data.
R → Receiver
T → Transmitter.

iii) Two signals → Transmit (TX) TX line
→ Receive (RX) RX line

iv) Transmitted in the form of sequence of bits, along with start / stop bits for framing.

v) Transmitter and receiver agree on a specific baud rate to sync their communication (bits per second).

vi) R-Pi has 2 UARTS
→ UART Tx - GPIO 14
→ UART Rx - GPIO 15

vii) e.g. Printer, microcontrollers, GPS modules.

NO

DATE

- GPIO

i) Refers to a set of pins (or ports) on a microcontroller (or single-board computer) that can be configured to either input or output digital signals.

ii) These pins can be used to interface with external devices such as sensors, LEDs, switches, motors, etc.

iii) These are numbered and can be referred either by their physical number or by their BCM (Broadcom)/logical GPIO number.

iv) The Raspberry Pi has a 40-pin GPIO header.

v) Can be configured for various purposes like digit I/O, or functions such as PWM, SPI, I2C, UART, etc.

vi) Allows the microcontroller to interact with the external environment by sending or receiving signals.

NO

DATE

- I2C (Inter-Integrated Circuit):

i) Protocol for communicating with low speed peripherals. (synchronous)

ii) RPi may have 1 or 2 I2C buses.

iii) Each bus has an I2C controller connected to 2 bidirectional lines
  - SDA : serial data line
  - SCL : serial data clock

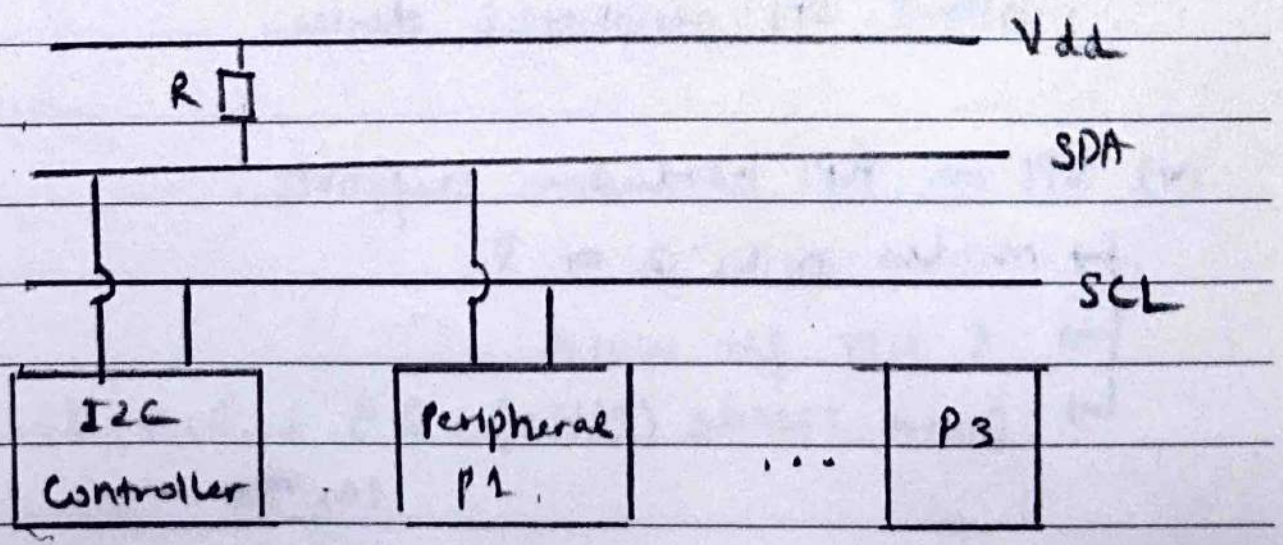
  These lines are connected to GPIO pins.

iv) It is possible to connect multiple I2C devices (LED, sensor) to I2C bus.

v) Each device on I2C must have a unique address.

vi) RPi uses 7-bit addressing scheme.

vii) 128 unique devices can be connected.



NO

DATE

- SPI (Serial Peripheral Interface)

i) Full duplex serial protocol for communicating with high-speed peripherals.

ii) The SPI controller on RPi can drive two SPI peripheral devices.

iii) SPI controller has 4 pins:

- SPI0_SCLK (GPIO 11): outputs a serial clock signal to sync communication.
- SPI0_SDO (GPIO 10): outputs data to the SPI peripheral device.
- SPI0_SDI (GPIO 9): Receives data from the SPI peripheral device.
- SPI0_CE0 (GPIO 8): enables one SPI peripheral device.
- SPI0_CE1 (GPIO 7): enables the other SPI peripheral device.

iv) SPI on RPi hardware supports:

- Modes 0, 1, 2 or 3
- 8 bits per word
- Data speeds (MHz): 0.5, 1, 2, 4, 8, 16, 32

NO

DATE

* Cross Compilation

• Crucial role in developing software for resource-constrained devices with varying architectures and OSs.

• Steps :

i) Setup Environment :

2️⃣

Unit 2 - M2M to IoT - A Market, M2M and IoT Technology Fundamentals

▼ **Syllabus**

- Introduction of M2M: A brief background, M2M communication, A typical M2M solution overview, Key application areas, Trends in information and communications technologies
- M2M to IoT - A Market Perspective, Information marketplaces, Global value chains, IoT value chains
- M2M to IoT - An Architectural Overview, Building an Architecture
- M2M to IoT Technology Fundamentals - Devices and Gateways, Local and Wide Area Networking, Data Management, M2M and IoT Analytics, Knowledge Management
- Architecture Reference Model - IoT Reference Model, Information Model, Functional Model, Communication Model, Safety, Privacy, Trust, Security Model

Introduction to IoT

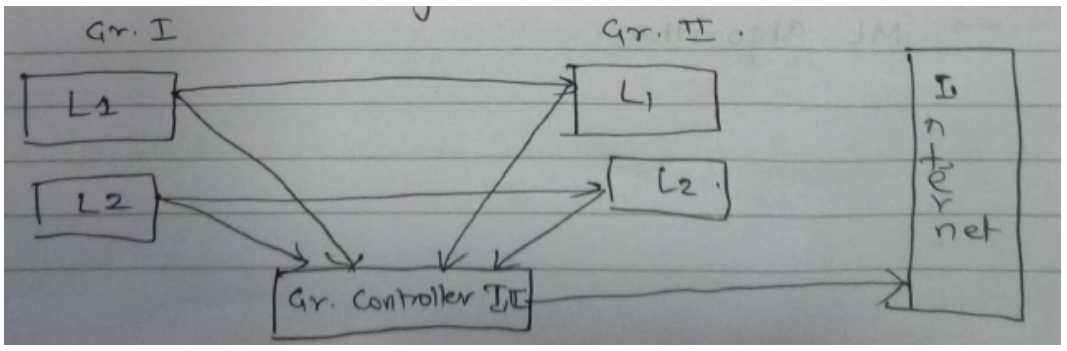
- IoT is a concept which enables the communication between internetworking devices and applications.
- The physical objects or things communicate through internet.
- The concept of IoT begins with things which can be classified as identity communication devices, e.g., RFID.

💡 Internet of Things means a network of physical objects sending, receiving or communication the information using the Internet or other communication technologies.

Hyperconnected Devices

💡 Hyper connectivity is the use of multiple systems and devices that are constantly connected to a network with the help of the Internet.

- Example: A network of city lights connected to the main station using sensors and IoT. All the city lights can be controlled through the main station.

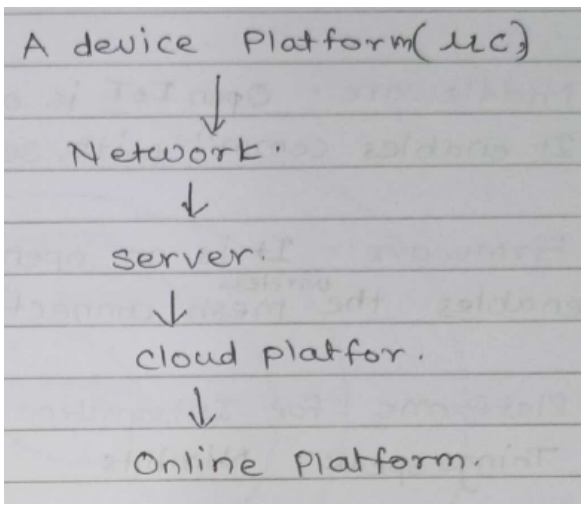


Technology behind IoT

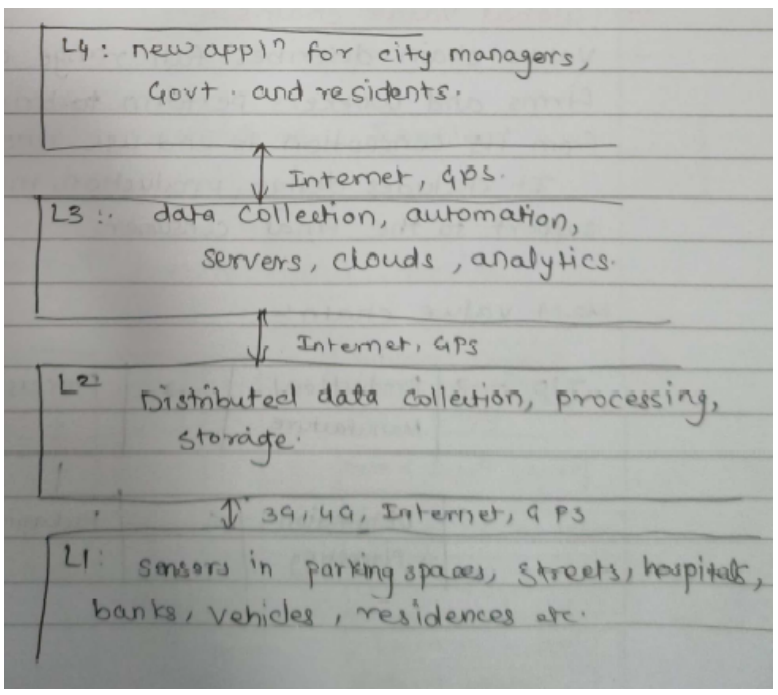
- These entities provide a large and diverse technological environment:
  - Hardware
  - IDE

- Network connection
- Software
- Machine learning algorithms

- **Various Levels in IoT**



- **E.g. Smart City**

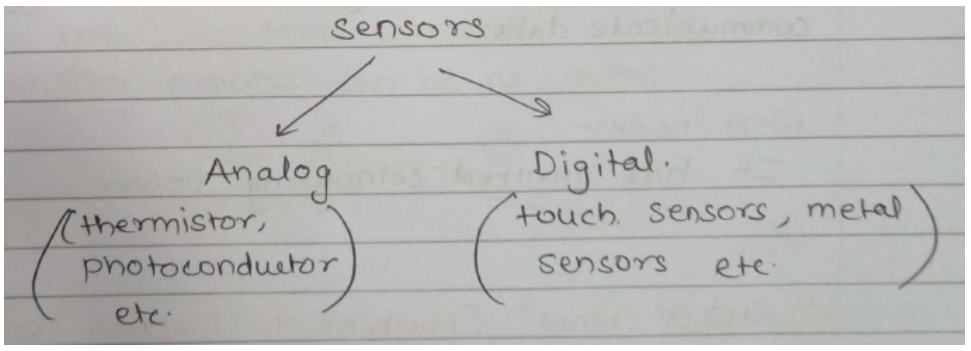


Components of IoT Systems

1. **Sensors and Control Units/Modules**

> 💡 Sensors are electronic components used for sensing physical parameters (like temperature, pressure, sound, etc.).

- Each street light has sensors for measuring light intensity and sends that data over a fixed period of time.
- Most commonly used control units are microcontrollers.
- Sensors can be
  1. **Analog**
     - The output of the sensor is some continuous function of its input parameter.
     - E.g. Thermistor, photoconductor, etc.
  2. **Digital**
     - The output of the sensor is in binary nature.
     - Designed to replace analog.
     - Touch sensors, metal sensors, passive IR sensors, etc.



2. **Communication Module**
    - It enables devices to transmit and receive data over some wired/wireless communication network.
    - It consists of protocol handles, message queues, etc.
    - Wired: Ethernet
    - Wireless: Wi-Fi, Bluetooth, ZigBee
3. **Software**
    - **Device Software**
        - Interacts with sensors, communication module and other network components.
        - Collects and processes data before sending it to the backend system.
    - **Server Software**
        - Includes databased, application server, and middleware components.
        - Manages data storage, access control, authentication, and most importantly, data processing tasks.
4. **Middleware**
    - It is responsible for communication between sensor and cloud.
    - It acts as a bridge between IoT devices and backend systems.
    - Handles normalization, protocol translation, device management and security enforcement.
    - Example: OpenIoT, FiWare
5. **Firmware**
    - It is an open source technology.
    - It enables wireless mesh connecvtivity.

Platforms for Integration

- Thingspeak
- Nimbits

Sources for IoT

- Arduino Uno
- Microduino
- Intel Galileo
- Intel Edison
- Beagle Board
- Raspberry Pi

Wireless Sensor Network

> 💡 It is a network in which each sensor node connects wirelessly with the ability to compute, aggregate and communicate data over it.

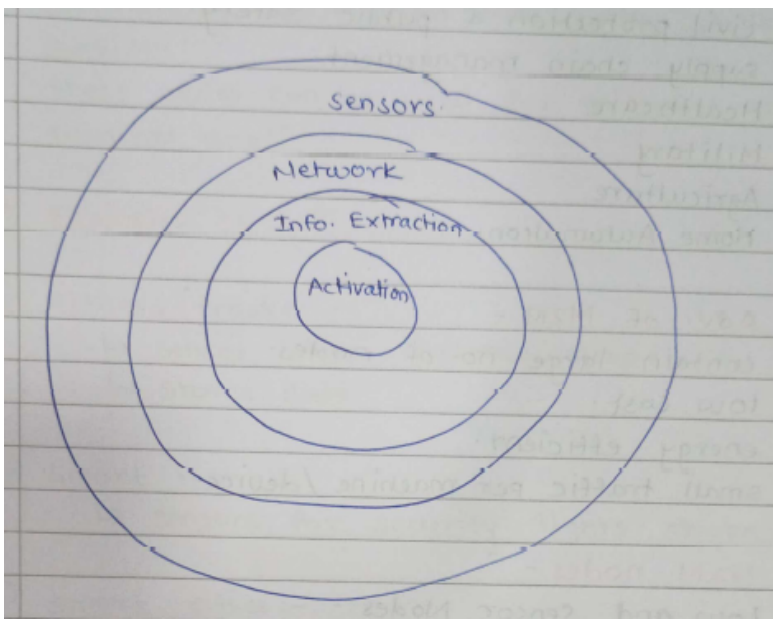
- WSN node has limited computing power.

M2M Communication

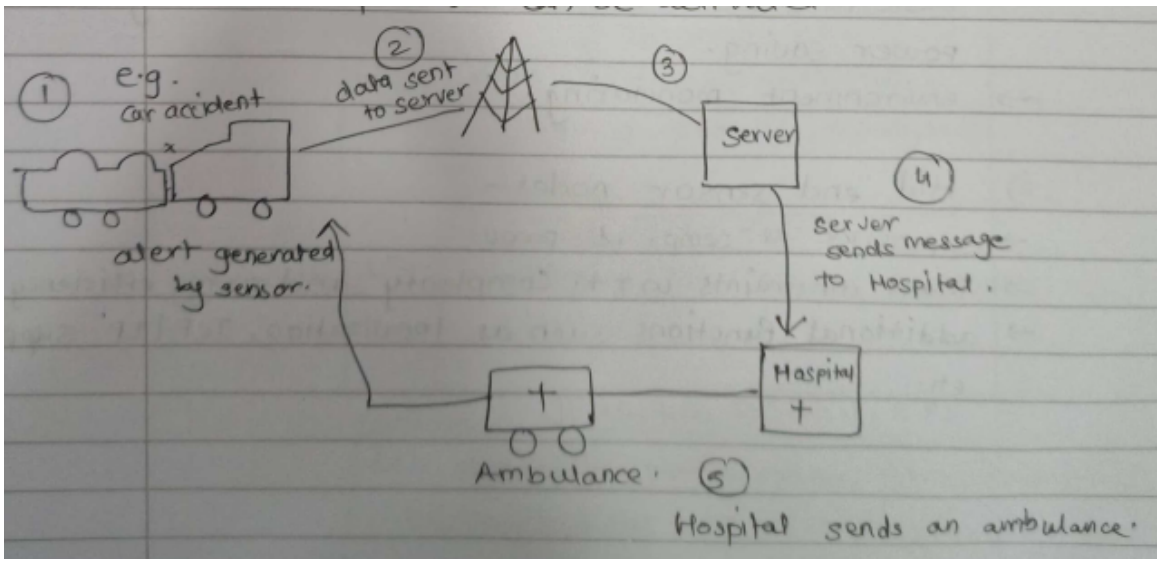
> 💡 Machine-to-Machine Communication refers to the direct communication between devices or machines without human intervention.

> 💡 M2M refers to those solutions that allow communication between devices of the same type and a specific application, all via wired/wireless communication networks.

- M2M allows end-users to capture data about events from assets (temperature, inventory levels).
- It is an important and basic block in IoT.
- Automated Interaction: Involves devices communicating autonomously.
- Data Exchange: To coordinate activities, share status updates, or trigger actions.
- Remote Monitoring and Control: Allows operators to monitor and control devices remotely.
- Efficiency: Share information about machine status, performance, resource allocation, etc.
- Wireless Technologies: Cellular networks, Wi-FI, Zigbee, etc.
- **Advantages**
  - Contains large number of nodes
  - Low cost
  - Energy efficient
  - Small traffic per machine/device
- **Overview**
  - Sensors collect data from remote areas
  - Networks are responsible for transportation of data
  - The required/necessary information is then extracted from the data
  - Then the process is activated using that information



  - Example of M2M communication



M2M Key Application Areas

- **Smart Cities**

  Used for intelligent infrastructure management, which includes
  - smart traffic lights
  - waste management
  - public transportation systems
  - environmental monitoring
- **Environmental Monitoring**
  - Air quality monitoring (AQI)
  - Water quality management
  - Weather forecasting
- **Healthcare**

  Plays crucial role in healthcare, allows professionals to monitor patient health remotely.
  - Remote patient monitoring
  - Wearable health devices
  - Telemedicine systems
- **Agriculture**
  - Real time-monitoring of soil conditions, weather patterns, crop health.
  - This data helps farmers optimize irrigation, fertilization, pest control and other farming practices, increasing crop yield.
- **Home Automation**

  Allows smart-home devices to communicate and be controlled remotely.
  - Thermostats
  - Security cameras
  - Lighting systems
  - Automated appliances
- **Supply chain management**
  - Connection and coordination of machines, sensors, and control systems.
  - Optimizes production processes, maintenance, and overall operational efficiency.
- Civil protection and public safety
- Military

Trends in ICTs

1. **5G**
    - Enables faster and more reliable wireless communication
    - Supports higher data speeds, low latency, and increased connectivity for a wide range of devices.
2. **Edge Computing**
    - Brings processing capabilities closer to the data source, reducing latency and improving efficiency.
    - Crucial for applications requiring real-time data processing (IoT devices, smart sensors).
3. **IoT**
    - Involves connecting and integrating various devices, sensors, and everyday objects to the internet.
    - Drives the creation of smart homes, cities, industries, and healthcare systems.
4. **AI/ML**
    - Provide capabilities such as NLP, image recognition, predictive analysis and automation across industries.
5. **AR/VR**
    - Provide immersive and interactive experiences.
    - Transforming industries such as gaming, healthcare, education and manufacturing.
6. **Blockchain**
    - Known for its role in cryptocurrencies
    - Offers decentralized and secure solutions
    - Supply chain, healthcare, identity verification
7. **Quantum Computing**
    - Explores potential of quantum mechanics
    - Perform complex computations at a much faster rate than classical computers

---

M2M Nodes

- **Low End Sensor Nodes**
    - Cheaper
    - Low-range data-transfer capabilities
    - Basic functions like data collection, auto configuration, power saving
    - e.g. Basic sensors, actuators, in environment monitoring, asset management
- **Mid End Sensor Nodes**
    - More expensive than low-end sensor nodes
    - Fewer constraints with respect to complexity and energy efficiency
    - Additional functions such as localization, TCP/IP support.
    - e.g. Gateways and controllers in energy meters
- **High End Sensor Nodes**
    - Can handle multimedia with quality of service
    - Highest cost
    - Can be used for military and biomedical applications
    - e.g. industrial controllers and IoT gateways in smart cities

Information Marketplaces

- Data can be shared between companies and value chains in internal information marketplaces.
- Alternatively, data could be publicly exchanged on a public information marketplace.
- These marketplaces are based on the exchange of data in order to create information products.
- **AWS IoT Marketplace**: Provides a platform for discovering, purchasing, and deploying IoT applications and devices.
- **Azure IoT Marketplace**: Offers a variety of pre-built solutions, services, and devices for IoT development.
- **IBM Watson IoT Marketplace**: Facilitates the discovery and integration of IoT solutions into projects.
- **Google Cloud IoT Marketplace**: Provides a platform for discovering and deploying IoT technologies.
- **Cisco IoT Marketplace**: Platform for Cisco IoT solutions and services, accelerating IoT deployments with software, hardware, and consulting.
- **IoTize Market**: Focused on connectivity solutions for IoT
- **IoTSense Marketplace**: Provides a marketplace for developers and businesses.
- **Thingiverse**: Platform focused on 3D printing, where users can find and share IoT-related designs and models.

Global Value Chains

> 💡 Value chain describes full range of activities that industries (or firms and workers) perform to bring a product from its conception to end-use and beyond.

1. **Design**
   - Involves conceptualizing and developing products or services
   - Based on market needs, consumer preferences, and technological advancements
2. **Production**
   - Encompasses manufacturing processes and operations for transforming raw materials or components into finished goods or intermediate products.
3. **Marketing**
   - Focuses on promoting and selling products or services to target markets and consumers.
4. **Distribution**
   - Involves the movement and logistics of goods or services from production facilities to distribution channels, retailers, and the end user.
5. **Support to the Consumer**
   - Focuses on providing after-sales services, customer support, and assistance to ensure customer satisfaction and loyalty.
   - Product installation and training
   - Tech support
   - Repair and maintenance

M2M Value Chains

> 💡 M2M Value Chains are specific type of value chain within the broader concept of GVCs.

- Internal to one company and cover one solution.

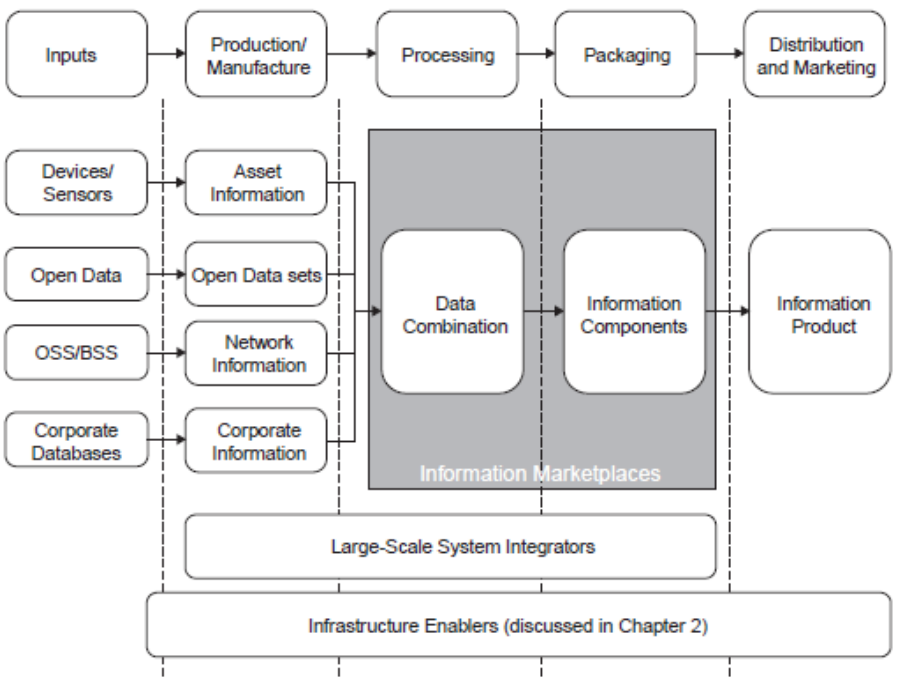
Input → Production → Processing → Packaging → Distribution/Marketing

1. **Inputs**

- Base raw ingredients that are turned into a product.
- Example: cocoa beans for the manufacture of chocolate or data from an M2M device that will be turned into a piece of information.

2. **Production/Manufacture**
    - The process that the raw inputs are put through to become part of a value chain.
    - Example: cocoa beans may be dried and separated before being transported to overseas markets, or Data from an M2M solution needs to be verified and tagged.
3. **Processing**
    - The process whereby a product is prepared for sale.
    - Example: cocoa beans may now be made into cocoa powder, ready for use in chocolate bars Or for an M2M solution, the aggregation of multiple data sources to create an information component.
4. **Packaging**
    - The process whereby a product can be branded as would be recognizable to end-user consumers.
    - Example: the chocolate bar would now be ready to eat and have a red wrapper with the words “KitKat” on it. For M2M solutions, the data will have to be combined with other information from internal corporate databases.
5. **Distribution/Marketing**
    - The channels to market for products.
    - Example: the chocolate bar may be sold at a supermarket, a kiosk, or even online. An M2M solution, however, will have produced an Information Product that can be used to create new knowledge within a corporate environment.

IoT Value Chains

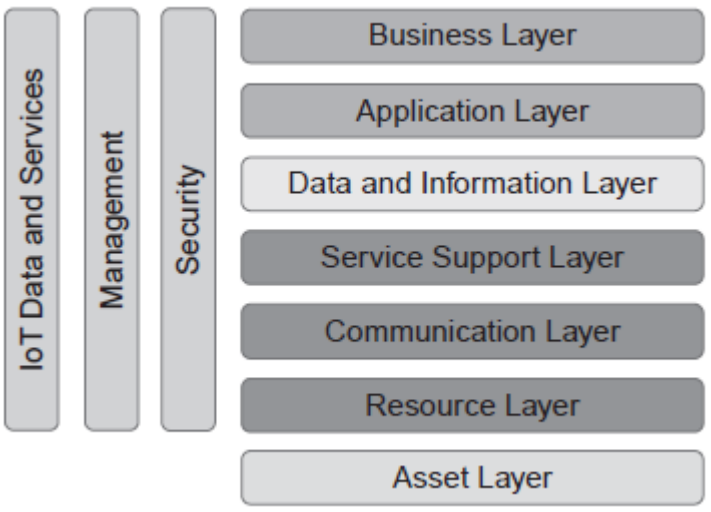


1. **Inputs**
    - Significantly more inputs than for an M2M solution.
    - Devices/Sensors: Data from these devices and sensors provides a different and much broader marketplace than M2M does.
    - Open Data: Data provided by government and city organizations, like city maps.
    - OSS/BSS (Operational and Business Support Systems): OSS/BSS of mobile operator networks are being used in information marketplaces
    - Corporate Databases: Companies of a certain size generally have multiple corporate databases covering various functions.
2. **Production/Manufacture**
    - The raw inputs described above will undergo initial development into information components and products.

- Asset Information: This data relates to whatever the sensor/device has been developed to monitor.
- Open Data Sets: May include maps, rail timetables, or demographics (about a certain area)
- Network Information: Information such as GPS data, services accessed via mobile network.
- Corporate Information: Information like the current state of demand for a particular product.

3. **Processing**
    - The data from the various inputs from the production and manufacture stage are combined together to create information.
    - Extensive use of data analytics.
4. **Packaging**
    - Creates information components, which could be produced as charts or other traditional methods of communicating information to end-users.
5. **Distribution and Marketing**
    - Final stage in creation of an Information Product.
    - Two categories of Information Products
        1. For improving internal decision-making
        2. For resale to other economic actors

M2M to IoT - Architectural Overview



1. **Asset Layer**
    - At the lowest level is the Asset Layer.
    - Consists of real world objects and entities that are subject to being monitored and controlled, as well as having digital representation and identities.
    - Assets are instrumented with embedded technologies that bridge the digital realm with the physical world.
    - e.g. Homes, vehicles, machinery, buildings, etc.
2. **Resource Layer**
    - Provides the main functional capabilities of sensing, actuation, and embedded identities.
    - Gateways of different types are placed that can provide aggregation or other capabilities.
3. **Communication Layer**
    - Provides means for connectivity between the resources on one end and different computing infrastructures on the other.
    - These computing infrastructures host and execute service support and application logic.

Network	Description
LAN (Local Area Network)	Connects devices in close proximity

Network	Description
WAN (Wide Area Network)	Wired and Wireless technologies (public or private), cellular mobile technologies
WPAN (Wireless Personal Area Network)	Fitness and healthcare applications
HAN and BAN (Home and Building Area Network)	Automation and control applications
NAN (Neighborhood Area Network)	Distribution grid of a smart electricity grid
V2V (Vehicle to Vehicle)	Collision avoidance or car platooning
ZigBee	Protocol stack for home automation

4. **Service Support Layer**
    - Provides support services
    - These support services
        - execute in data centers (or server farm) in a cloud environment.
        - provide uniform handling of underlying devices and networks
    - Example: remote device management for remote software updates, diagnostics or recovery
5. **Data and Information Layer**
    - Provides a more abstract set of functions
    - Main purposes are to capture knowledge and provide control logic support.
    - Main focus is on the organization of information.
6. **The Application Layer**
    - Provides the specific IoT applications.
    - Open ended array of different applications
    - Examples: smart meter in Smart Grid, vehicle tracking
7. **Business Layer**
    - Focuses on supporting core business operations of any organization
    - Where integration of IoT applications into business process and enterprise systems take place.
    - Provides exposure to APIs for third parties to get access to data and information
    - Support for business process workflow

Parameters for Building M2M to IoT Architecture

Parameter	M2M	IoT
Scalability	Designed for applications with a limited number of devices	Need to accommodate a massive number of devices and scale seamlessly with network growth
Interoperability	Limited: May use proprietary protocols and interface	Emphasizes interoperability across devices and platforms
Data Variety and Complexity	Involves simple, predefined data exchanges between devices	Deals with diverse data types (structured and unstructured)
Communication Protocols	May use specific communication protocols tailored to their application	Leverage standardized protocols such as MQTT, CoAP, HTTP
Security	May focus on securing point-to-point communication	Introduces a broader attack surface, requiring comprehensive security measures
Device Management	Typically simpler, focusing on basic functionalities	More complex tasks, including firmware updates, configuration and lifecycle

Parameter	M2M	IoT
		management
Standardization	May lack standardized approaches	Crucial in IoT to ensure compatibility and collaboration between different ecosystems

Devices in IoT Systems

1. **Basic Devices**
   - Microcontroller class devices, which can perform simple operations
   - Cannot communicate without using gateways
   - Good for simple processes like alarms, metering, etc.
   - Components used are cheap, usually using SoC, at least two microprocessors and wireless connectivity stack.
   - E.g. landline phones, 2-way radio
2. **Advanced Devices**
   - General purpose class devices, which perform application level logic and support communication protocols.
   - Designed for more complex processes like automated units, processing multiple actions at the same time.
   - May also function as a gateway for devices on the same LAN.
   - E.g. wireless router

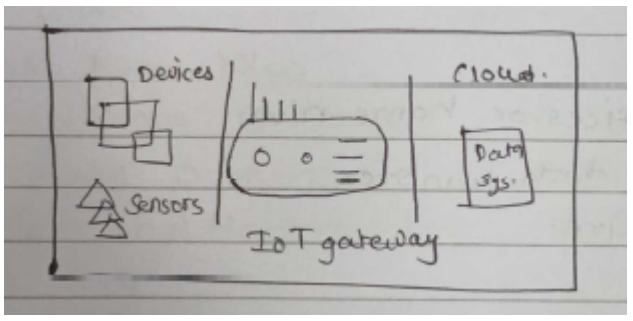
M2M vs IoT

Parameter for Difference	IoT	M2M
	Internet of Things	Machine-to-Machine
Intelligence	Devices have objects responsible for decision making	Limited intelligence observed
Connection	Via network	Point-to-point
Communication Protocol	Internet protocols (HTTP, FTP, Telnet)	Traditional protocol and communication technologies
Internet	Required for communication	Not dependent on Internet
Business Model	B2B and B2C	B2B
Approach	Horizontal enabler approach	Vertical system solution approach
Components	Devices/Sensors Connectivity Data Processing UI	Device Area Networks Gateway Application Server
Deployment	Cloud	In-house
Examples	Smart wearables, home automation	Telemetry, environmental monitoring

IoT Gateways

> 💡 IoT Gateways act as a bridge between different communication technologies.

- Acts as a medium to establish connection between the cloud and controller in IoT.
- Developer-to-developer, or developer-to-cloud communication can be established using gateways.



- **Functions**
  1. Protocol Translation: Translate data between different communication protocols.
  2. Data Aggregation: Collect and aggregate data, preparing them for further processing or transmission.
  3. Local Processing: To extract insights, or execute predefined algorithms.
  4. Data Filtering: Based on predefined criteria, rules, or thresholds to discard redundant data.
  5. Local Data Storage: Temporarily cache or buffer incoming data before transmission.
  6. Autonomous Development Control: Execute predefined scripts, rules, or workflows (server-independent)
  7. Device Security: Implement security controls like encryption and authentication to ensure CIA.
- **Working**
  1. Receive data from sensor network
  2. Perform preprocessing, filtering, and cleaning on unfiltered data
  3. Transports it into standard protocol for communication
  4. Send data to cloud
- **Advantages**
  - Reliable (fault tolerance and remote management)
  - Scalable (modular architecture and distributed processing)
  - Cost-effective

LAN and WAN in IoT

LAN

- Group of network devices that allow communication between those connected devices.
- Advantages
  - Reliable and secure
  - No need for Internet communication
- Wired LAN
  - Used in offices or home networks
  - High speed data transfer
  - Fast and secure
- Wi-Fi
  - Uses radio waves to connect devices
  - Used in homes, shops, public spaces
  - Convenient and easy to use
  - Flexible, suitable for mobile phones
- ZigBee
  - Low-power, low-data rate

    - Used in home automation and sensor networks
    - Reliable

WAN

- Connected collection of telecommunication networks.
- Switched WAN
    - Multiple components of LANs connected via shared network infrastructure
    - Best suited for distributed environments
- Point-to-Point WAN
    - Two LAN nodes are connected via a leased line

LAN vs WAN

LAN	WAN
Local Area Network	Wide Area Network
Ownership is private	Ownership is private/public
Higher speed than WAN	Lower speed than LAN
Propagation delay is short	Propagation delay is long
Easy design and maintenance	Difficult design and maintenance
Covers small area	Covers large geographical area
Principle of broadcasting	Principle of Point-to-Point
High data transfer rate	Low data transfer rate
Ethernet, Token Ring	Frame Relay, X.25
Low setup cost	High setup cost
e.g. computer lab	e.g. pager/beeper

M2M to IoT Analytics

> 💡 The process of analyzing data generated by connected devices to derive meaningful insights and drive informed decision-making.

1. Data Collection
2. Data Integration
3. Data Preprocessing
4. Data Analysis
5. Visualization and Reporting
6. Real-time Analytics
7. Predictive Analysis
8. Continuous Improvement

Data Management (Page 107)

- Effective data management is crucial for handling the vast amount of data generated by connected devices efficiently and securely.
- Needed for data governance, as massive volumes of data are generated.
- **Need of Data Management**

- Product Development: Error detection, performance analysis, offer product insights
- Prediction of Wear and Tear: Equipment lifecycles, make maintenance plans
- Facilitate Resource and System Efficiency

- **Methods**
  1. **Data Generation:** First stage within which data is generated actively or passively from the device, system, or as a result of its interactions.
  2. **Data Acquisition:** Deals with the collection of data (actively or passively) from the device, system, or as a result of its interactions.
  3. **Data Validation:** Data acquired must be checked for correctness and meaningfulness within the specific operating context.
  4. **Data Storage:** Specialized technologies such as massively parallel processing DBs, distributed file systems, cloud computing platforms, etc. are needed.
  5. **Data Processing:** Main goal is to operate on the data at a low level and enhance it for future needs.
  6. **Data Remanence:** Prevention of data remanence using overwriting, degaussing, encryption, and physical destruction.
  7. **Data Analysis:** The data is subjected to analysis with the aim to obtain the information they encapsulate and use it for supporting decision-making processes.

Knowledge Management

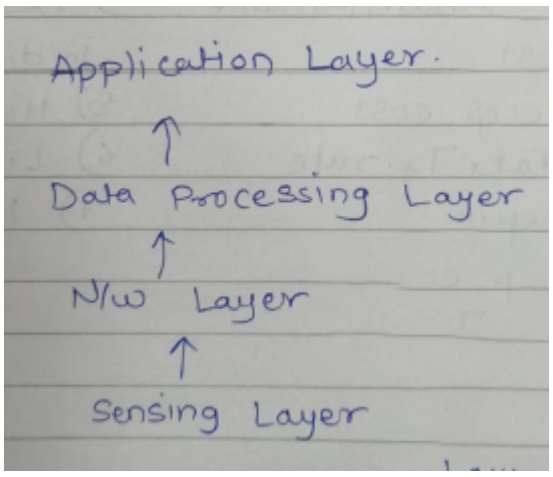
> 💡 Involves the systematic process of capturing, organizing, sharing, and leveraging knowledge and insights derived from connected devices and systems.

1. **Data Capture**: Sensor readings, device statuses, environmental conditions, user interactions.
2. **Data Organization**: Categorizing data based on attributes such as type, source, time, location.
3. **Data Sharing and Collaboration**: Establishing platforms, tools, and processes for sharing data, analyses, best practices, and lessons learned.
4. **Knowledge Discovery and Analysis**: Data mining, AI/ML are employed to uncover patterns, trends, correlations within the data.
5. **Knowledge Repositories**: House structured and unstructured data, analyses, reports, documentation
6. **Continuous Learning and Improvement**: Capture feedback, lessons learned, and insights from past experiences.
7. **Security and Governance**: Implementing robust security measures, access controls, encryption, and compliance with data privacy regulations.

IoT Reference Model

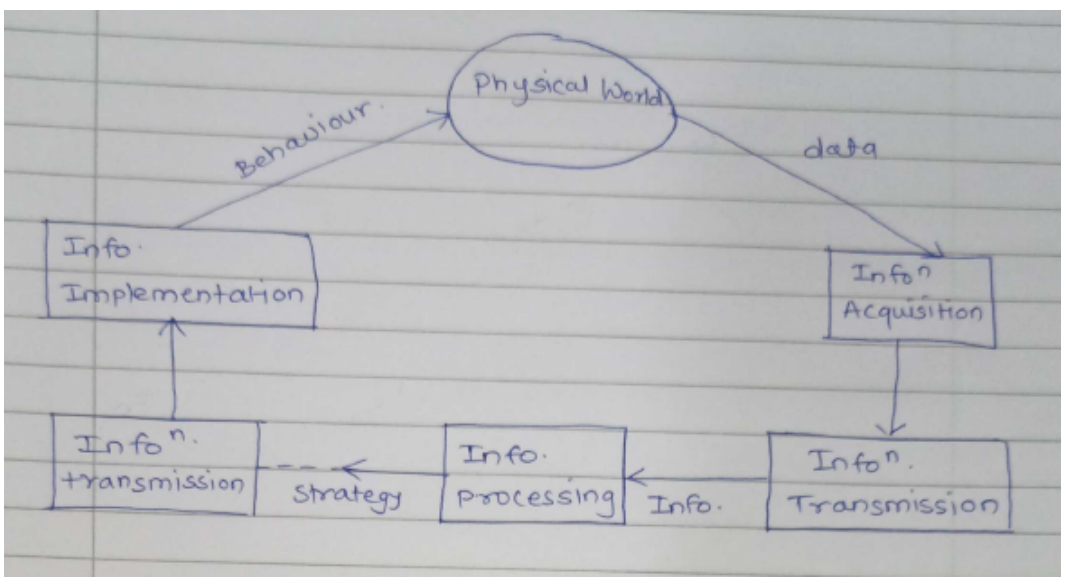
- Application Layer
  - Interacts with end-user, access and control, IoT development
- Data Processing Layer
  - Software/hardware components responsible for collecting, analyzing and interpreting data.
- Network Layer
  - Responsible for providing communication and connectivity
  - Wi-Fi, Bluetooth, Zigbee, 4G, 5G
- Sensing Layer
  - Collect data from different sources

- Sensors/actuators
- Connected to network layer through wired/wireless communication protocols



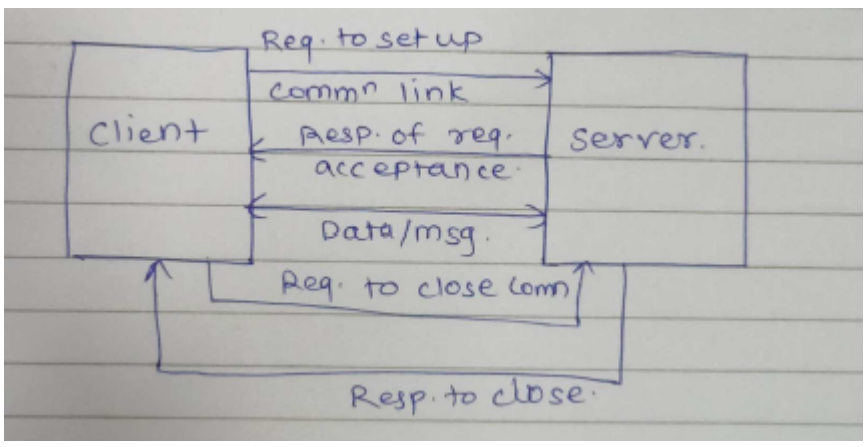
IoT Information Model

- On a high-level, the IoT Information Model maintains the necessary information about Virtual Entities and their properties or attributes.
- The IoT Information Model describes Virtual Entities and their attributes, including metadata.
- Associations between Virtual Entities and their Services are explicitly captured in the Information Model.
- Attributes can be associated with sensor services or actuation services, depending on whether they involve reading sensors or controlling actuators.



IoT Communication Model

- The communication model involves identifying interaction endpoints, traffic patterns, and underlying technology properties.
- Potential endpoints include Users, Resources, and Devices.
- Gateways bridge different communication technologies for interactions involving constrained Devices.



3️⃣

Unit 3 - IoT and Protocols, IoT Security and Interoperability

▼ **Syllabus**

- Introduction to IoT: What is IoT? IoT Examples, Simple IoT LED Program
- IoT and Protocols
- IoT Security: UPnP, CoAP, MQTT, XMPP
- IoT Service as a Platform: Clayster, Thinger.io, SenseIoT, Carriots and NodeRED
- IoT Security and Interoperability: Risks, Modes of Attacks, Tools for Security and Interoperability

What is IoT?

💡 The Internet of Things is a network of interrelated devices that connect and exchange data with other IoT devices and the cloud.

- It refers to a network of interconnected physical devices embedded with sensors, actuators, software, and network connectivity, allowing them to collect and exchange data.
- Also a widely used term for a set of technologies and systems associated with the emerging wave of Internet-connected things that are based on the physical environment.
- A *thing* in the internet of things can be a natural or man-made object that can be assigned an IP address and is able to transfer data over a network.
  - person with a heart monitor implant
  - a farm animal with a biochip transponder
  - an automobile that has built-in sensors to alert the driver when tire pressure is low.
- With IoT, data is transferable over a network without requiring human-to-human or human-to-computer interactions.
- IoT devices are equipped with sensors to collect data from the surrounding environment.
- IoT also facilitates automation by allowing devices to respond to specific conditions or triggers automatically.

IoT Examples

1. **Smart Home Automation**
   - Devices like smart thermostats (Nest), smart lighting systems, and smart security cameras allow homeowners to control and monitor their home environment remotely.
   - These devices can be managed through mobile apps or voice commands, providing convenience and energy efficiency.
2. **Wearable Health Trackers**
   - Fitness trackers and smartwatches collect health related data, including steps taken, heart rate, sleep patterns, and more.
   - Users can track their fitness goals, and healthcare professionals can use that data for remote patient monitoring and preventive healthcare.
3. **Industrial IoT in Manufacturing**

- Manufacturing equipment, machinery, and industrial processes are equipped with sensors and connectivity capabilities that enable
    - real-time monitoring
    - predictive maintenance
    - optimization of production processes
- These features lead to increased productivity. uptime, and cost savings for industrial organizations.

4. **Smart Cities**
    - Smart cities leverage IoT to create smart city solutions for better urban management.
    - Examples
        - smart traffic management systems that use sensors to optimize traffic flow
        - waste management systems with smart bins that notify authorities when full
        - environmental monitoring for air quality and noise levels
5. **Smart Vehicles**
    - Vehicles equipped with sensors and connectivity features enable
        - real-time monitoring of vehicle performance
        - navigation assistance
        - remote diagnostics,
        - remote software updates
    - This enhancing safety, efficiency, and convenience for drivers.
6. **Precision Agriculture/Smart Agriculture**
    - IoT Sensors in the fields monitor soil moisture, temperature, and crop health.
    - Drones equipped with cameras provide aerial views, helping farmers make data driven decisions about irrigation, fertilization, and pest control to optimize their crop yield.

Simple IoT LED Program

1. **Hardware**
    - Raspberry Pi 4 Model B
    - LED
    - Resistor
    - Breadboard and Jumper Wires
    - Power Supply
    - GPIO Pins
2. **Connection Procedure**
    - Connect a female-to-female jumper wire from Pin 15 (GPIO 22) to the LED.
    - Connect a female-to-female jumper wire from Pin 6 (Ground) to the LED’s ground.
3. **Code**

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarning(False)

led = 22 # GPIO BCM number
```

```
GPIO.setup(led, GPIO.out)

try:
    while True:
            GPIO.output(led, GPIO.HIGH)
            time.sleep(0.05)
            GPIO.output(led, GPIO.LOW)
            time.sleep(0.05)
finally:
    GPIO.output(ligtht, GPIO.LOW)
    GPIO.cleanup()
```

IoT Protocols

- UPnP
- CoAP
- MQTT
- XMPP

UPnP (Universal Plug and Play)

> 💡 UPnP is a set of networking protocols which enables devices (like PCs, printers, mobile devices) to discover each other and establish connections for sharing data.

- Intended to be used on residential networks.
- Conceptually extends PnP, which enables users to connect devices directly to a computer without any manual configuration (to the device or to the computer).
- **Working**
  - It assumes that a device is compatible with IP addressing, for using protocols built on it (HTTP, TCP, UDP).
  - It uses these protocols to *advertise* the device's presence and for data transfer.
- **Addressing (IP)**
  1. When initiated, it acts as a DHCP client to assign itself an IP and searches for a DHCP server.
  2. If no DHCP server is found, the device assigns itself an IP using AutoIP (assigns a unique IP to its local network).
  3. During the DHCP transaction, if the device gets a domain name through a DNS server, it uses that instead.
- **Key Components**
  1. Discovery
     - SSDP is used by UPnP devices to discover each other.
     - The devices sends SSDP alive messages after it is added to the network, to advertise it's services to other devices on the network.
     - SSDP also allows a device to passively listen to SSDP alive message from other devices.
     - A discovery message is exchange when two devices discover each other on the network.
     - The message contains information like device type, it's services and it's capabilities.
  2. Device Description
     - After discovery, the devices exchange information in XML format, to learn more about each other.
     - Manufacturer name, model name, manufacturer websites, device's services, etc.

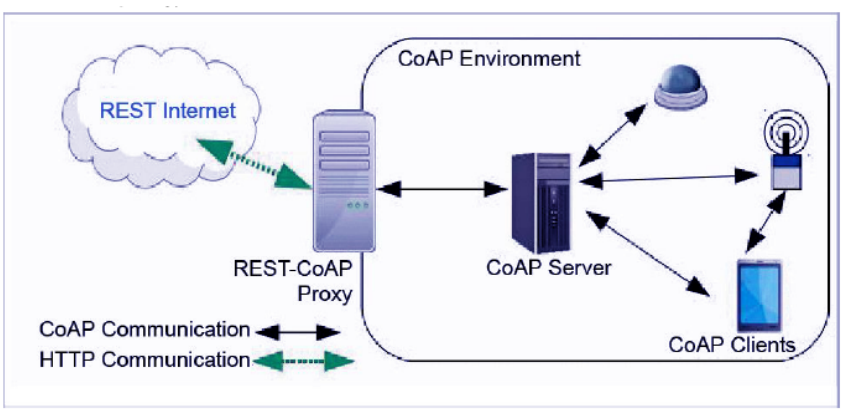
3. Service Calls
    - After receiving information, the control point can call for the service to the URL (provided by manufacturer).
    - This is done using SOAP (which passes XML messages).
4. Event Notification
    - General Event Notification Architecture (GENA) is used.
    - Used by services to respond to service calls.
    - XML format
5. Presentation
    - The presentation URL is used by a control point to retrieve information and by the user to customize the device settings on a web browser.

- **Advantages**
    - Allows PnP compatibility
    - Backed by big vendors (Microsoft, Intel)
    - Industry standard
    - Ideal architecture for home networks
- **Disadvantages**
    - Malicious programs can also use PnP, just as a legitimate program would use it.
    - No authentication is required by the control points, hence anyone can ask to forward a UPnP port.

CoAP (Constrained Application Protocol)

> 💡 CoAP is a session layer protocol that provides the RESTful interface between HTTP client and server.

- Designed to use devices on the same constrained network (by IETF).
- Built for IoT systems primarily based on HTTP protocols.
- **Architecture**
    - CoAP Client
    - CoAP Server
    - REST-CoAP Proxy
    - REST Internet



1. The data is sent from CoAP clients (phones, RFID sensors) to the CoAP server.
2. Same message is routed to the REST-CoAP proxy.
3. The proxy interacts outside the CoAP environment and uploads the data over the REST internet.

- Advantages

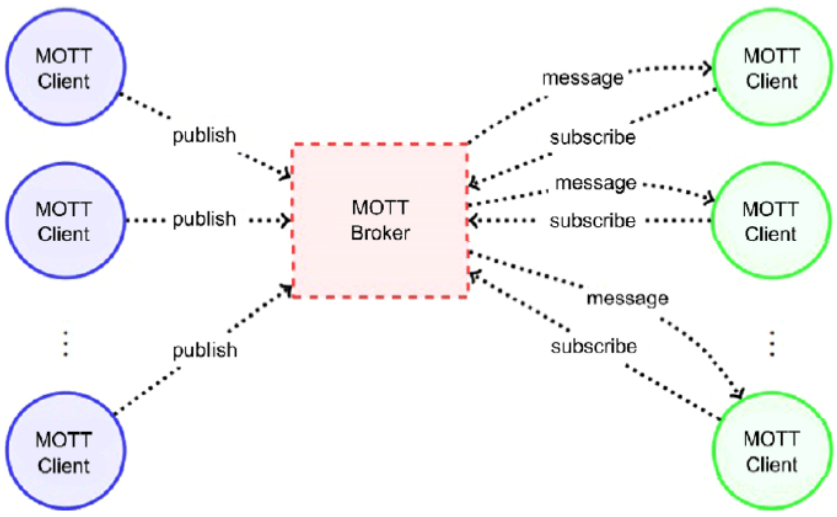
- Low-power consumption
- Optimized for LNNs

MQTT (Message Queue Telemetry Transport)

> 💡 **MQTT** is a simple, lightweight, publish-subscribe TCP messaging protocol used to establish communication between multiple devices.

- **Architecture**
  - Based on publish/subscribe architecture
  - Three major components

1. **Publishers**: Lightweight sensor devices that send their data to connected broker and goes back to sleep whenever possible.
2. **Subscribers**: Applications interested in a certain topic (or sensory data), are connected to brokers to be informed whenever new data is received.
3. **Brokers**: Receives the sensory data and filters them in different topics and sends them to subscribers according to their interest in the topics.



Parameter	CoAP	MQTT
	Constrained Application Protocol	Message Queuing Telemetry Transport
Communication Type	Request-Response model	Publish-Subscribe model
Messaging Mode	Asynchronous and Synchronous	Asynchronous
Transport Layer Protocol	UDP	TCP
RESTful Based	Yes	No
Effectiveness in LNN (Low-power Lossy Network)	Excellent	Low
Communication Model	One-to-one	Many-to-many

XMPP (Extensible Messaging Presence Protocol)

> 💡 XMPP is a protocol for streaming XML elements over a network in order to exchange messages and presence information close to real-time.

- Used by messaging applications like WhatsApp.

- **eXtensible**: XMPP is an open-source project which can be changed or extended according to the need.
- **Messaging**: Designed for sending messages in real-time. Very efficient push mechanism comparatively.
- **Presence**: Determines your state, i.e., online, offline, or busy.
- **Protocol**: Set of standards that allow systems to communicate with each other.
- **Why Messaging Applications use XMPP?**
  - Send and receive messages (to/from other users)
  - Check and share presence status
  - Manage subscriptions to/from other users
  - Manage contact list
  - Block communication to/from specific users
- **Advantages**
  - Free and decentralized (anyone can set up server)
  - Based on open standards
  - Supports multiple implementations of clients and servers
  - Flexible and XML-based
  - Suitable for IM features and custom cloud services
  - Efficient (millions of concurrent users on single service)

IoT Service as a Platform

1. **Clayster**
   - Clayster is an IoT service platform that provides a foundation for building and managing IoT applications.
   - It offers a console application framework for creating services.
   - `Clayster.AppServer.Infrastructure`: Contains the application engine available in the platform. Manages applications, report tools, cluster support, and data sources used in IoT.
   - `Clayster.Library.Abstract`: A data abstraction layer, for efficient object management.
   - `Clayster.Library.Installation`: Defines the concept of packages.
   - `Clayster.Library.Meters`: Contains abstraction model for sensors, actuators, controllers, etc.
2. **Thinger.io**
   - Thinger.io is an open-source IoT platform that allows connecting various devices to the Internet.
   - It supports everything from basic Arduinos to more complex embedded systems like Raspberry Pi.
   - It defines data flow for automations, alerts, and custom KPIs.
   - Built on scalable technology.
3. **SenseIoT**
   - SenseIoT is a cloud-based platform for storing and processing sensor data securely.
   - Tools include data visualization, trigger setting, and remote data management.
   - Developers can integrate it with devices, sensors, or sensor networks via the SenseIoT API.
   - The API conforms to REST design principles, supports HTTP methods like GET, PUT, POST, DELETE and OPTIONS.
4. **Carriots**
   - Carriots is a PaaS (application hosting and development platform) designed for IoT and M2M projects.
   - It collects and stores data from connected objects and facilitates building powerful applications with minimal code.
   - Offers integration with external IT systems.

- Functionalities include rules, triggers, custom alarms, listeners, SDK application engine, etc.

Risks of IoT

1. **Security Vulnerabilities**
2. **Data Privacy Concerns**
3. **Data Integrity and Trustworthiness**
4. **Lack of Interoperability**
5. **Operational Risks**
6. **Supply Chain Risks**
7. **Regulatory and Compliance Challenges**

Modes of Attack

1. **DoS and DDoS**
   - Flood IoT devices or network with an overwhelming volume of traffic.
2. **Botnets**
   - Consist of compromised IoT devices controlled by central C2 server.
   - DDoS, spam campaigns and distributed scanning
3. **Malware and Ransomware**
   - Malware infects and compromises functionality of IoT devices
   - Ransomware encrypts the data and demands random for decryption
4. **Credential**
   - Exploiting weak or default credentials
   - Brute force, dictionary, credential stuffing
5. **MiTM**
   - Intercept communication between IoT devices and their intended recipients
   - Eavesdrop on or manipulate data exchange
   - Wired and wireless networks
6. **Physical**
   - Gaining physical access to IoT devices or infrastructure
   - Tamper with hardware components, implant malware, or extract information
7. **Supply Chain**
   - Target vulnerabilities in the manufacturing, distribution, or procurement process of IoT devices.
   - Compromise components, firmware, or software during production or distribution.

Tools for IoT Security and Interoperability

1. **Firewalls**
   - First layer of defense against malware ,viruses and other threats.
   - Filters both incoming and outgoing data.
   - Customize rules and policies as per requirement.
   - Necessary to create exceptions that allow certain apps to pass through the firewall so that they don’t constantly trigger false alarms.

2. **AV**
    - Signature-based AV software scans files to make sure there aren’t any hidden threats.
    - Remove or quarantine affected files on finding something suspicious.
    - Cannot protect from zero-day threats
3. **Anti-Spyware**
    - Spyware secretly snoops on victims to see where they go online, logs their keystrokes, and any other confidential or personal data.
    - It fights back by (ideally) detecting and removing threats such as key loggers, password recorders.
4. **Palo Alto Networks**
    - Palo Alto Networks IoT security solution makes use of NGFWs.
    - NGFWs go beyond just packet filtering and use behavioral analysis of the network.
5. **AWS IoT Device Management**
    - This tool makes it easy to audit configurations, authenticate devices, and receive alerts to help secure your IoT device fleet.
    - You can seamlessly integrate it with other AWS services (Lambda - serverless computing, CloudWatch for monitoring, SNS - Simple Notification Service).
6. **Metasploit**
    - Metasploit provides a range of tools and modules for conducting penetration tests, exploiting vulnerabilities, and assessing overall security.
    - It enables organizations to simulate real-world attack scenarios.
7. **Microsoft Defender for IoT**
    - It is a unified security solution built specifically for IoT and OT devices, vulnerabilities, and threats.
    - You can seamlessly integrate it with other Azure services (IoT Hub or Security Center for monitoring, Sentinel for threat detection and response).
