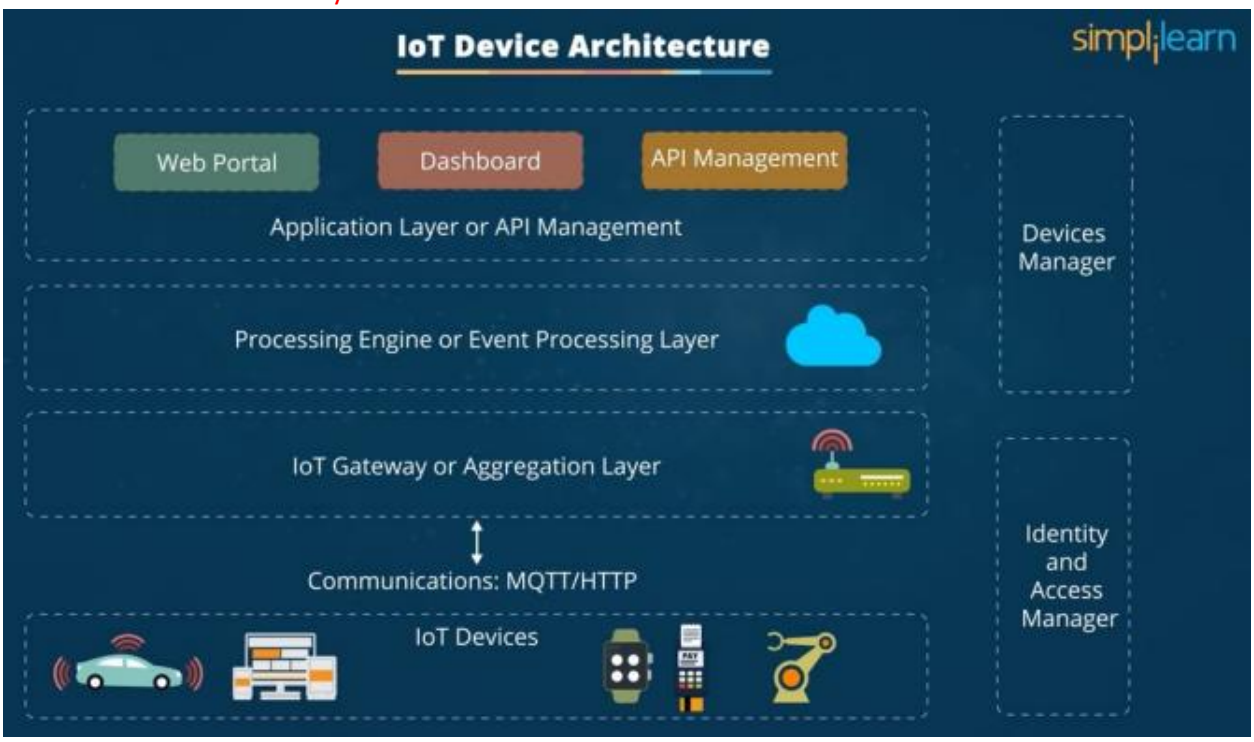
1. Illustrate the IoT architecture layers.



2. Sensors and actuators have design issues such as calibration, nonlinearity, sampling, aliasing, noise, and failures, discuss these issues and how they affect the development.

Calibration

Description: Adjusting the sensor or actuator to ensure accurate output.
Impact: Affects accuracy and reliability; requires regular maintenance.

Nonlinearity

Description: Deviation of the sensor or actuator response from a straight line.
Impact: Leads to errors and complicates signal processing.

Sampling

Description: Recording data at specific time intervals.
Impact: Can result in loss of important information if the sampling rate is inappropriate.

Aliasing

Description: Misinterpretation of high-frequency signals as lower frequencies due to inadequate sampling rates.
Impact: Causes incorrect data interpretation and potential system errors.

Noise

Description: Unwanted disturbances that affect the signal.
Impact: Reduces the accuracy and reliability of measurements and actions.

Failures

Description: Breakdown or malfunction of the sensor or actuator.
Impact: Can lead to system downtime, safety hazards, and increased maintenance costs.

3. For programming models, developers can choose between Looping, Loop and ISR, and Real-time operating systems, discuss the pros and cons of each one.

❑ Simple (Pooling – Loop Structure)

● It goes around repeatedly executing the same sequence of actions.

● Simple - Code is reliable and easy to understand.

● Performance / Scalability / task priorities.

❑ Loop / ISR

● consider placing all the non-critical tasks in the main loop and locating the time-sensitive tasks in Interrupt Service Routines (ISRs).

● Challenge is the distribution of tasks between the main loop and the ISRs.

❑ RTOS (Real Time OS)

● Divide the application into tasks that run concurrently.

● The essence of real-time computing is not only that the computer responds to its environment fast enough, but that it responds reliably fast enough.

4. Developers choose IoT OS based on some parameters such as Footprint, Portability, reliability, connectivity … etc. Discuss these parameters.

❑ Scalability:

The operating system must be scalable for any type of device. That means both integrators and developers need to be familiar with the operating system when it comes to gateways and nodes.

❑ Footprint:

Since the devices will always come with a bag of constraints, it is essential to choose an operating system with low power, processing, and memory requirements. The overheads incurred by you should be minimal at the end of the day.

❑ Reliability:

This is a critical factor to note for mission-critical systems. For instance, Industrial IoT devices are at remote locations and have to work for years without hampering business continuity. Your operating system should be able to fulfill specific certifications for IoT apps.

❑ Portability:

Operating systems isolate apps from the specifics of the hardware, hence leading to greater portability. Usually, an OS is ported to different interface and hardware platforms to the board support package (BSP) in a standardized format, such as POSIX calls.

❑ Connectivity:

This one is obvious, but your operating system should support connectivity protocols such as WiFi, IEEE, Ethernet, and so on. There is no point in IoT apps if they cannot connect without any hassle.

❑ Security:

The operating system of your choice should be safe and secure to use, allowing you to add on some aspects in the form of SSL support, secure boot, components, and encryption drivers.

❑ Modularity:

Every operating system must mandatorily have a kernel core. All other functionalities can be included as add-ons if so required by the IoT app you are building.

5. Provide a summary of Contiki OS

Contiki OS is an open-source operating system tailored for the Internet of Things (IoT) and embedded systems.

Key Features:

1. **Lightweight**: Designed for devices with constrained resources.
2. **Event-Driven Kernel**: Efficient task management and low power consumption.
3. **Protothreads**: Allows blocking operations without full threading overhead.
4. **Networking Support**: Full IP stack with support for IPv4, IPv6, CoAP, MQTT, and 6LoWPAN.
5. **Power Management**: Features mechanisms for extending battery life.
6. **Simulation Tools**: Includes Cooja for emulation and testing.

6. Compare MQTT Quality of Service levels.

❑ QoS level 0 means messages are sent without any confirmation from the receiver. This means it is technically possible for a message to get lost, given an unreliable connection. “fire and forget”-level.

❑ QoS level 1 means the receiver must send a confirmation to let the sender know that the message was received. However, with QoS 1, it is possible that the receiver gets a single message multiple times. This QoS level ensures that a message makes it from sender to receiver but does not ensure that it is received exactly once.

❑ QoS level 2 uses a four-step communication process to ensure a message is sent exactly once only, which can be important depending on the use case.

7. Compare HTPP and MQTT and Show why most IoT platforms use MQTT for communications.

	MQTT	HTTP
Design orientation	Data centric	Document centric
Pattern	Publish/subscribe	Request/response
Complexity	Simple	More complex
Message size	Small, with a compact binary header just two bytes in size	Larger, partly because status detail is text-based
Service levels	Three quality of service settings	All messages get the same level of service
Extra libraries	Libraries for C (30 KB) and Java (100 KB)	Depends on the application (JSON, XML), but typically not small
Data distribution	Supports 1 to zero, 1 to 1, and 1 to n	1 to 1 only

8. Compare LoRa and Zigbee. Comparison points: power consumption, transmission range, Data rate, topology, and cost.

	Lora	zigbee
Power consumption	300bps to 37.5 kbps	Low
transmission range	3 miles to 10 miles	10 to 100 meters
Data rate	Lower	Higher
topology	Star	Star, tree, peer-to-peer and mesh
cost	Low	middle

9. Compare Zigbee, LoRa, and Bluetooth. Comparison points: range, throughput, power consumption, and topology.

	Bluetooth	Zigbee	lora
Range	10m – 1.5k	30m – 100m	2km – 20km
Throughput	125kbps – 2mbps	20kbps- 250kbps	10kbps-50kbps
Power consumption	Low	Low	Low
Topology	P2p,star,mesh,broadcast	Mesh	star

10. AWS is composed of six main components: identity service, device gateway, message broker, rules, device shadow, and registry. Discuss each component and its function
    1. **Identity service** – Provides authentication, authorization, and device provisioning.
    2. **Device gateway** – Securely connects IP-connected devices and edge gateways to the AWS Cloud and other devices at scale.
    3. **Message broker** – Processes and routes data messages to the AWS Cloud.
    4. **Rules** – Invokes actions in the AWS Cloud.
    5. **Device Shadow service** – Maintains a shadow of your device so the device can be accessed and controlled at any time.
    6. **Registry** – Stores information about devices and their attributes.

11. Compare static thing groups and dynamic thing groups.
    Static thing groups organize devices into groups that you specify.
    Dynamic thing groups update group membership through search queries.
12. Security pillars in AWS.

13. What is the role of AWS device defender?
    is a fully managed service that helps you audit the configuration of your devices, monitor connected devices to detect abnormal behavior and mitigate security risks. It gives you the ability to enforce consistent security policies across your AWS IoT device fleet and respond quickly when devices are compromised.
14. Device management role.
    Management eases the organizing, monitoring, and managing of IoT devices at scale. At scale, customers are managing fleets of devices across multiple physical locations. AWS IoT Device Management enables you to group devices for easier management. You can also enable real-time search indexing against the current state of your devices through Device Management Fleet Indexing. Both Device Groups and Fleet Indexing can be used with Over the Air Updates (OTA) when determining which target devices must be updated.

15. Discuss AWS IoT six layers and show the relations between them and the general IoT architecture layer.
    - Edge layer
    - Provision layer
    - Communications layer
    - Ingestion layer
    - Analytics layer
    - Application layer
16. Elaborate provision layer in AWS IoT layers.
    - The private key infrastructure (PKI) used to create unique identities of the devices.
    - The process by which firmware is first installed on devices, and the application workflow that provides configuration data to the device.
    - The ongoing maintenance and eventual decommissioning of devices over time. IoT applications need a robust and automated provisioning layer so that devices can be added and managed by IoT applications in a frictionless way.
    - When you provision IoT devices, you need to install X.509 certificates onto them.
17. Provide a summary of Amazon FreeRTOS.
18. Compare Amazon FreeRTOS and Contiki OS.