Module 1: Introduction to AWS for IoT (3 hours)

1.1 Overview of AWS IoT Core

○ Introduction to AWS IoT Core and its architecture.

○ Setting up an IoT Thing in AWS.

○ Device registration, certificates, and policies.

1.2 AWS IoT Security

○ Secure communication with X.509 certificates.

○ Using AWS IoT policies and authentication.

○ Managing and rotating security certificates.

Module 2: Setting Up ESP32 for AWS IoT (4 hours)

2.1 Connecting ESP32 to AWS IoT Core

○ Using the MQTT protocol with AWS IoT Core for secure data transmission.

○ Implementing device authentication with certificates.

○ Connecting and testing ESP32's connection to AWS IoT Core.

2.2 Sending and Receiving Data with MQTT

○ Sending sensor data (e.g., temperature, humidity) from ESP32 to AWS IoT Core.

○ Subscribing to MQTT topics to receive commands or notifications.

2.3 Hands-on Project

○ Building an MQTT-based temperature monitoring system that sends data to AWS.

Module 3: Advanced Communication Protocols (5 hours)

3.1 Introduction to RS485 Communication

○ Understanding RS485 and its use in industrial environments.

○ Setting up RS485 communication on ESP32 (Master-Slave configuration).

○ Sending and receiving data over RS485 using ESP32.

3.2 Working with I2C

○ Introduction to the I2C protocol and its use cases.

○ Setting up multiple I2C devices (sensors, displays) with ESP32.

○ Communicating with I2C sensors like temperature, humidity, and pressure sensors.

3.3 SPI Communication on ESP32

○ Introduction to SPI protocol and hardware connections.

○ Using SPI to interface with external devices (e.g., SD card, display).

○ Hands-on example: Interfacing an OLED display or SD card with ESP32.

Module 4: AWS Cloud Data Storage and Management (4 hours)

4.1 Storing Data in DynamoDB

○ Introduction to DynamoDB for NoSQL data storage.

○ Sending sensor data from ESP32 to DynamoDB via AWS IoT rules.

○ Querying and managing data in DynamoDB from the cloud.

4.2 Data Visualization with AWS QuickSight

○ Connecting DynamoDB with AWS QuickSight for data analysis and visualization.

○ Creating interactive dashboards to monitor IoT data in real-time.

4.3 Hands-on Project

○ Implementing an IoT data logging system where ESP32 data is stored in DynamoDB and visualized using AWS QuickSight.

Module 5: AWS Lambda for Serverless Computing (4 hours)

5.1 Introduction to AWS Lambda

○ Overview of AWS Lambda and serverless architecture.

○ Setting up a Lambda function to process data from ESP32.

○ Triggering Lambda functions from AWS IoT Core.

5.2 Integrating Lambda with Other AWS Services

○ Sending processed data from Lambda to DynamoDB.

○ Using Lambda to trigger alerts (e.g., sending notifications with AWS SNS).

5.3 Hands-on Project

○ Building a serverless IoT solution: When the ESP32 sends data, it triggers a Lambda function to process and store it.

Module 6: AWS IoT Analytics (3 hours)

6.1 Overview of AWS IoT Analytics

○ Introduction to AWS IoT Analytics and its components (pipelines, channels, datasets).

○ Setting up a data pipeline to ingest and process ESP32 data.

6.2 Data Processing and Analysis

○ Using AWS IoT Analytics to analyze data streams from ESP32.

○ Performing data transformations and filtering.

6.3 Hands-on Project

○ Implementing a data pipeline to collect, process, and analyze sensor data from ESP32.

Module 7: Building IoT Applications with AWS IoT Device Management (4 hours)

7.1 Overview of AWS IoT Device Management

○ Device provisioning and fleet indexing.

○ Organizing and monitoring IoT devices.

7.2 OTA (Over-the-Air) Updates for ESP32

○ Using AWS IoT Device Management to send firmware updates to ESP32 devices over the air (OTA).

○ Automating firmware deployment and updates for multiple devices.

Module 8: Final Project and Deployment (3 hours)

8.1 IoT System Design and Planning

○ Outline a real-world IoT solution that integrates ESP32, AWS IoT Core, DynamoDB, Lambda, and various communication protocols.

○ Design considerations for scalability, security, and performance.

8.2 Implementation, Testing, and Optimization

○ Final project development: Implement the system, debug, and optimize it for real-world usage.