Step 1: Set Up the Hardware

Components Needed:

1. **Raspberry Pi (any model with GPIO support)**
    - **Ensure it has Raspbian OS installed.**
2. **DHT11 Sensor (Temperature and Humidity)**
3. **Soil Moisture Sensor**
    - **Connect through an MCP3008 ADC for analog-to-digital conversion.**
4. **MCP3008 ADC**
    - **Required for soil moisture sensor data.**
5. **Jumper wires and a breadboard for wiring.**

Connections:

1. **DHT11 Sensor:**
    - **VCC → 3.3V**
    - **GND → Ground**
    - **Data → GPIO4**
2. **MCP3008 (SPI):**
    - **VDD/VREF → 3.3V**
    - **GND → Ground**
    - **CLK → GPIO11 (SCLK)**
    - **DOUT → GPIO9 (MISO)**
    - **DIN → GPIO10 (MOSI)**
    - **CS → GPIO8 (CE0)**
3. **Soil Moisture Sensor:**
    - **Connect the analog output to CH0 of MCP3008.**

Step 2: Install Necessary Python Libraries

On the Raspberry Pi:

- **Update and upgrade your system:**

  bash

  Copy code

  ```
  sudo apt update && sudo apt upgrade
  ```

1.

- **Install required Python libraries:**

  bash

  Copy code

  ```
  sudo pip3 install Adafruit-DHT spidev paho-mqtt
  ```

2.

Step 3: Write Python Code for Sensor Data Collection

3.1 DHT11 Sensor Script

- **Use the `Adafruit_DHT` library to read temperature and humidity. Example:**

python

Copy code

- `import Adafruit_DHT`
-
- `DHT_SENSOR = Adafruit_DHT.DHT22`
- `DHT_PIN = 4  # GPIO pin`
-
- `humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)`

- `print(f"Temperature: {temperature}°C, Humidity: {humidity}%")`

3.2 Soil Moisture Script Using MCP3008

- **Use the `spidev` library to read from the MCP3008 ADC. Example:**

python

Copy code

```
import spidev

def read_channel(channel):
    adc = spi.xfer2([1, (8 + channel) << 4, 0])
    data = ((adc[1] & 3) << 8) + adc[2]
    return data

spi = spidev.SpiDev()
spi.open(0, 0)  # Bus 0, Device 0
soil_moisture = read_channel(0)
print(f"Soil Moisture Level: {soil_moisture}")
```

Step 4: Configure AWS IoT Core

4.1 Create an IoT Thing

1. **Log in to AWS Management Console → Navigate to IoT Core.**
2. **Create a new Thing.**
3. **Download the certificates (private key, device certificate, and root CA).**

4.2 Set Up an MQTT Topic

1. Define a topic (e.g., `raspberrypi/sensors`).
2. Attach an IoT policy to allow publish/subscribe actions.

Step 5: Write MQTT Publishing Script

1. Use the `paho-mqtt` library to connect to AWS IoT Core.
2. Publish sensor data as a JSON payload:

python

Copy code

```
import paho.mqtt.client as mqtt
import ssl
import time

# AWS IoT details
ENDPOINT = "<Your-AWS-IoT-Endpoint>"
CLIENT_ID = "RaspberryPi"
TOPIC = "raspberrypi/sensors"

# Certificates
PATH_TO_CERT = "path/to/cert.pem.crt"
PATH_TO_KEY = "path/to/private.pem.key"
PATH_TO_ROOT_CA = "path/to/AmazonRootCA1.pem"

# Connect to AWS IoT
client = mqtt.Client(CLIENT_ID)
```

```
client.tls_set(ca_certs=PATH_TO_ROOT_CA,
               certfile=PATH_TO_CERT,
               keyfile=PATH_TO_KEY,
               tls_version=ssl.PROTOCOL_TLSv1_2)
client.connect(ENDPOINT, 8883, 60)

# Publish Data
def publish_data(temperature, humidity, soil_moisture):
    data = {
        "temperature": temperature,
        "humidity": humidity,
        "soil_moisture": soil_moisture,
        "timestamp": time.strftime("%Y-%m-%d %H:%M:%S")
    }
    client.publish(TOPIC, json.dumps(data))
    print(f"Data published: {data}")
```

Step 6: Deploy AWS Lambda for Data Processing

1. **Create an AWS Lambda function:**
    - **Use Python as the runtime.**
- **Write Lambda code to process incoming MQTT messages:**

  python

  Copy code

```
import json

def lambda_handler(event, context):
```

- `print("Received data:", event)`
- `# Process and store the data (e.g., in DynamoDB)`

2.
3. **Attach the Lambda function to an IoT Rule:**
    - **Trigger the function when data is published to `raspberrypi/sensors`.**

Step 7: Configure AWS IoT TwinMaker

1. **Create a TwinMaker workspace.**
2. **Define entities and components to represent your system:**
    - **Example: RaspberryPiEntity → SensorComponents (temperature, humidity, soil moisture).**
3. **Use the processed data to update digital twin entities in real-time.**

Step 8: Test the System

1. **Run your Python script on the Raspberry Pi:**
    - **Ensure it reads sensor data and publishes it to AWS IoT Core.**
2. **Verify:**
    - **Data in AWS IoT MQTT Test Client.**
    - **Lambda function execution and logs.**
    - **Real-time updates in AWS IoT TwinMaker.**
