Setup Manual for MQTT Server and ESP32 Sensor Integration

Your Name

December 10, 2024

Contents

1 Introduction

This manual provides detailed instructions for setting up an MQTT server on AWS and integrating it with an ESP32-based soil sensor.

2 Setting Up MQTT Server on AWS

2.1 AWS EC2 Instance

1. Launch an EC2 instance using Amazon Linux 2 or Ubuntu Server.
2. Configure security groups to allow traffic on ports 1883, 8883, and 9001.
3. SSH into your EC2 instance.

2.2 Install Docker and Docker Compose

```
sudo apt update
sudo apt install -y docker.io
sudo systemctl start docker
sudo systemctl enable docker
sudo apt install -y docker-compose
```

2.3 Deploy MQTT Broker

1. Transfer your configuration files and certificates to the EC2 instance.
2. Run the Docker container:

   ```
   docker compose up -d
   ```

3 Setting Up ESP32 for Sensor Data

3.1 Environment Setup

1. Install the Arduino IDE or PlatformIO.
2. Install necessary libraries: PubSubClient, WiFi.

3.2 ESP32 Code

```
#include <WiFi.h>
#include <PubSubClient.h>

// WiFi credentials
const char* ssid = "your-SSID";
const char* password = "your-PASSWORD";

// MQTT broker details
const char* mqtt_server = "your-broker-ip";
const int mqtt_port = 1883;
const char* mqtt_user = "admin";
const char* mqtt_password = "mosquitto123";

// Topics
const char* topic = "agri/soil";

// WiFi and MQTT clients
WiFiClient espClient;
PubSubClient client(espClient);
```

```
void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
}

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Read sensor data
  float sensorValue = readSensor(); // Implement this function to read from your

  // Publish sensor data
  char msg[50];
  snprintf(msg, 50, "Sensor value: %f", sensorValue);
  client.publish(topic, msg);

  delay(2000); // Adjust the delay as needed
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP32Client", mqtt_user, mqtt_password)) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

float readSensor() {
  // Implement your sensor reading logic here
  return 0.0; // Replace with actual sensor reading
}
```

4 Data Logging with Python

4.1 Python Script Setup

1. Install the ‘paho-mqtt‘ library:

   ```
   pip install paho-mqtt
   ```

2. Create and run the Python script to log data.

4.2 Python Script

```
import paho.mqtt.client as mqtt
import json
import csv

# MQTT broker details
broker = "localhost"
port = 1883
topic = "agri/soil"

# CSV file to store data
csv_file = "sensor_data.csv"

# Initialize CSV file with headers
with open(csv_file, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["Conductivity", "Moisture", "Temperature", "pH", "Nitrogen"
```

```
# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    try:
        # Decode and parse the JSON message
        data = json.loads(msg.payload.decode())
        # Append data to CSV
        with open(csv_file, mode='a', newline='') as file:
            writer = csv.writer(file)
            writer.writerow([
                data.get("conductivity"),
                data.get("moisture"),
                data.get("temperature"),
                data.get("ph"),
                data.get("nitrogen"),
                data.get("phosphorus"),
                data.get("potassium")
            ])
        print(f"Data saved: {data}")
    except Exception as e:
        print(f"Error processing message: {e}")

client = mqtt.Client()
client.on_message = on_message

client.connect(broker, port, 60)
client.subscribe(topic)

# Blocking call that processes network traffic, dispatches callbacks and handles
client.loop_forever()
```