**Laporan Pratikum Internet of Things (IOT) Pemantauan Suhu dan Kelembapan Berbasis ESP32 Menggunakan MQTT**

**Oleh**

*Maynanda Elisa Pasya T*

*Fakultas Vokasi, Universitas Brawijaya*

*Email:* [*maynandatbg@student.ub.ac.id*](mailto:maynandatbg@student.ub.ac.id)

**ABSTRAK**

Perkembangan teknologi Internet of Things (IoT) memungkinkan berbagai perangkat elektronik saling terhubung untuk mengirim dan menerima data secara real-time. Salah satu penerapannya adalah sistem pemantauan suhu dan kelembapan berbasis mikrokontroler ESP32 yang dilengkapi dengan koneksi WiFi. Pada praktikum ini, digunakan sensor DHT22 untuk membaca suhu dan kelembapan lingkungan, kemudian data dikirimkan ke broker MQTT menggunakan protokol komunikasi ringan dan efisien, yaitu MQTT (Message Queuing Telemetry Transport). ESP32 bertindak sebagai publisher yang mengirimkan data secara berkala ke broker MQTT, sementara client seperti MQTT Explorer atau Node-RED bertindak sebagai subscriber untuk menampilkan data tersebut. Hasil praktikum menunjukkan bahwa sistem dapat bekerja secara real-time, stabil, dan efisien dalam mengirimkan data lingkungan melalui jaringan internet. Praktikum ini membuktikan bahwa ESP32 dan protokol MQTT sangat cocok digunakan dalam pengembangan sistem monitoring berbasis IoT.

*Kata Kunci*: *IoT, ESP32, DHT22, MQTT, Pemantauan Suhu, Kelembapan, MQTT Broker, Monitoring Real-Time.*

***ABTRACT***

*The advancement of Internet of Things (IoT) technology enables electronic devices to connect and communicate data in real time. One of its applications is a temperature and humidity monitoring system based on the ESP32 microcontroller, which is equipped with WiFi connectivity. In this practicum, a DHT22 sensor is used to read environmental temperature and humidity, and the data is transmitted to an MQTT broker using the lightweight and efficient MQTT (Message Queuing Telemetry Transport) communication protocol. The ESP32 acts as a publisher that periodically sends data to the MQTT broker, while clients such as MQTT Explorer or Node-RED serve as subscribers to display the data. The results of the practicum show that the system operates in real time, remains stable, and efficiently transmits environmental data over the internet. This practicum proves that ESP32 and the MQTT protocol are highly suitable for developing IoT-based monitoring systems.*

*Keywords: IoT, ESP32, DHT22, MQTT, Temperature Monitoring, Humidity, MQTT Broker, Real-Time Monitoring.*

1. **Pendahuluan**
   1. Latar Belakang

Internet of Things (IoT) merupakan konsep pengembangan teknologi yang memungkinkan perangkat fisik terhubung dan bertukar data melalui internet. Salah satu implementasi nyata dari IoT adalah sistem pemantauan lingkungan seperti suhu dan kelembapan udara. Pemantauan parameter lingkungan ini penting untuk berbagai aplikasi, seperti pertanian cerdas, gudang penyimpanan, laboratorium, hingga sistem pendingin ruangan. Teknologi komunikasi berbasis MQTT (Message Queuing Telemetry Transport) sangat cocok digunakan dalam sistem IoT karena bersifat ringan, cepat, dan efisien untuk pengiriman data secara berkala.

ESP32, sebagai mikrokontroler yang dilengkapi dengan konektivitas WiFi, menjadi pilihan ideal untuk mengirimkan data suhu dan kelembapan dari sensor ke server atau dashboard pemantauan melalui protokol MQTT. Dengan bantuan broker MQTT seperti Mosquitto atau Cloud MQTT, data dapat diterima dan divisualisasikan secara real-time.

* 1. Tujuan
* Membangun sistem pemantauan suhu dan kelembapan berbasis ESP32.
* Mengirimkan data sensor secara real-time menggunakan protokol MQTT.
* Menampilkan data pada platform monitoring MQTT seperti Node-RED, MQTT Explorer, atau dashboard lainnya.

1. **Metodologi**
   1. Alat dan Bahan

Alat :

1. Laptop
2. Platform Wokwi
3. Platform Visual Studio Code (VSCode)
4. Platform MQTT Explorer

Bahan :

1. ESP32 (Virtual dalam platform wokwi)
2. Sensor DHT22
3. LED
4. Bahasa Pemrograman C++ Pustaka Arduino
5. Library

* PubSubClient.h
* Wifi.h
* DHTesp.h
  1. Langkah Implementasi

1. Buka Visual Studio Code dan pastikan ekstensi PlatformIO sudah terpasang.

2. Buat proyek baru dengan board ESP32 Devkit V1.

3. Tambahkan library DHT sensor library dan PubSubClient melalui platformio.ini.

4. Hubungkan sensor DHT22 ke ESP32 (misalnya pin data ke GPIO 15).

5. Siapkan broker MQTT, bisa menggunakan:

* Broker lokal (Mosquitto)
* Broker cloud (e.g. broker.hivemq.com atau test.mosquitto.org)

6. Tulis program untuk:

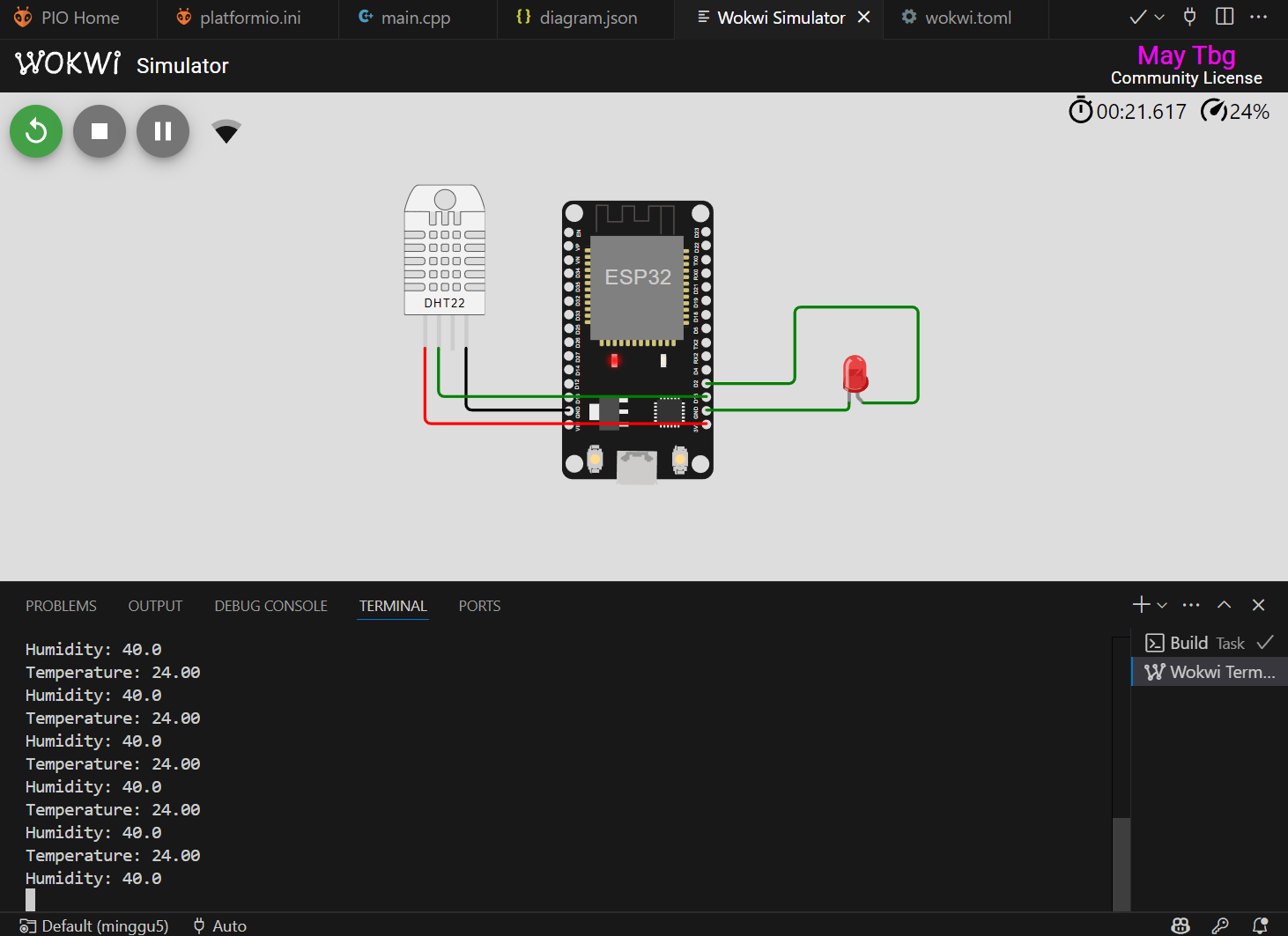
* Membaca data dari sensor suhu dan kelembapan.
* Menghubungkan ESP32 ke WiFi.
* Mengirim data ke broker MQTT dengan format payload JSON atau string.

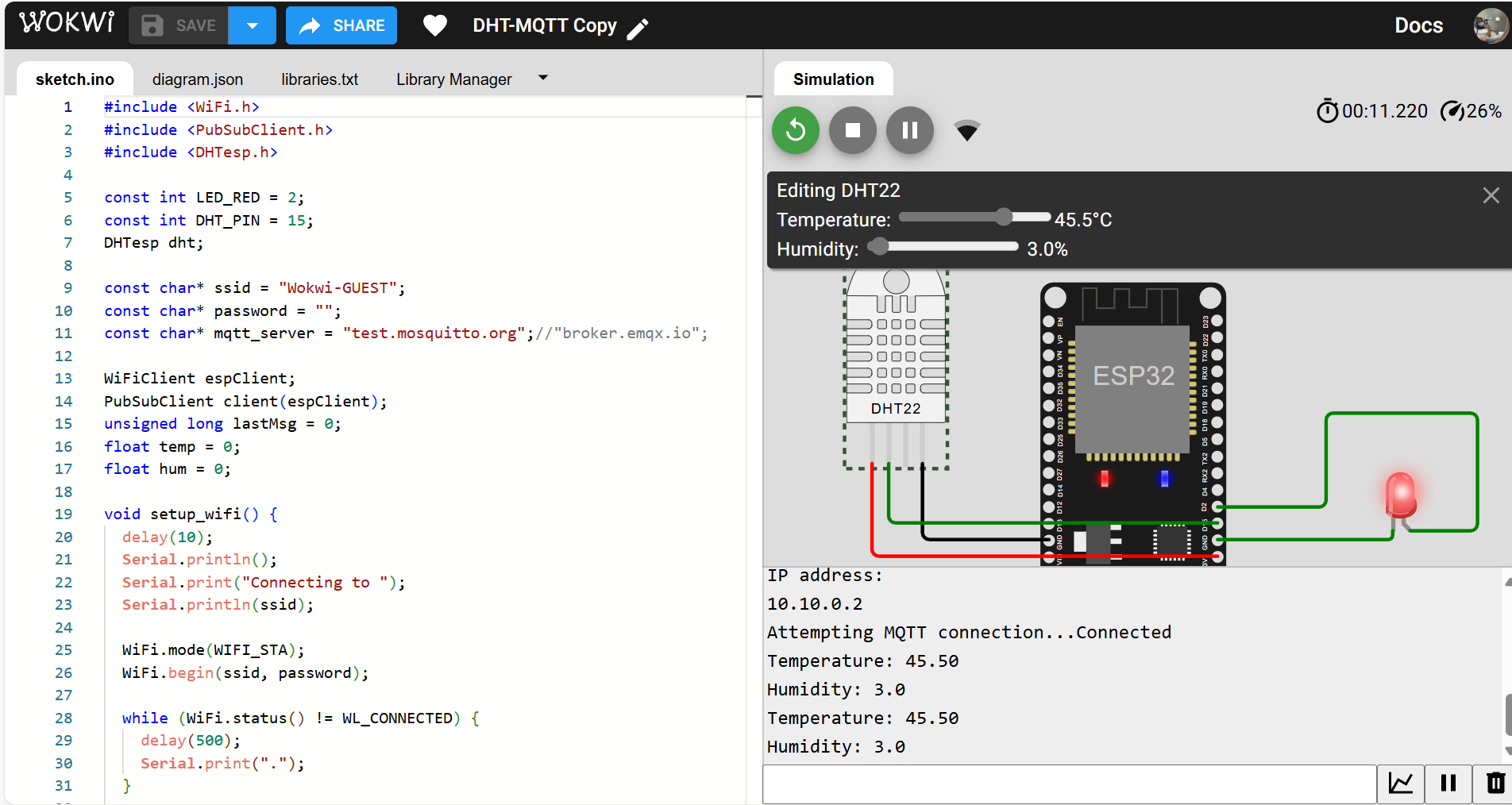
7. Gunakan aplikasi seperti MQTT Explorer atau Node-RED untuk memantau dan menampilkan data.

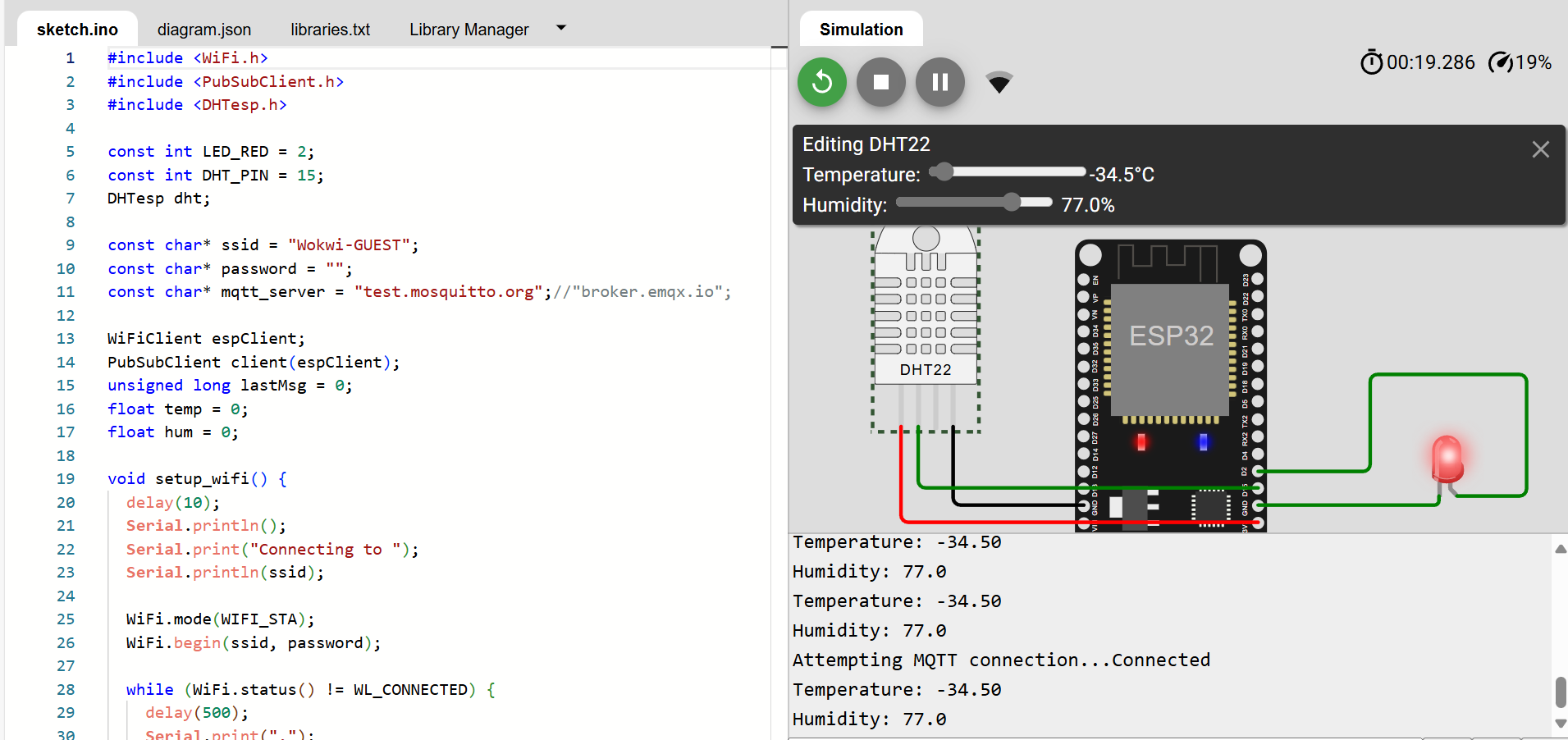
8. Jalankan program dan amati pengiriman data secara berkala ke broker MQTT.

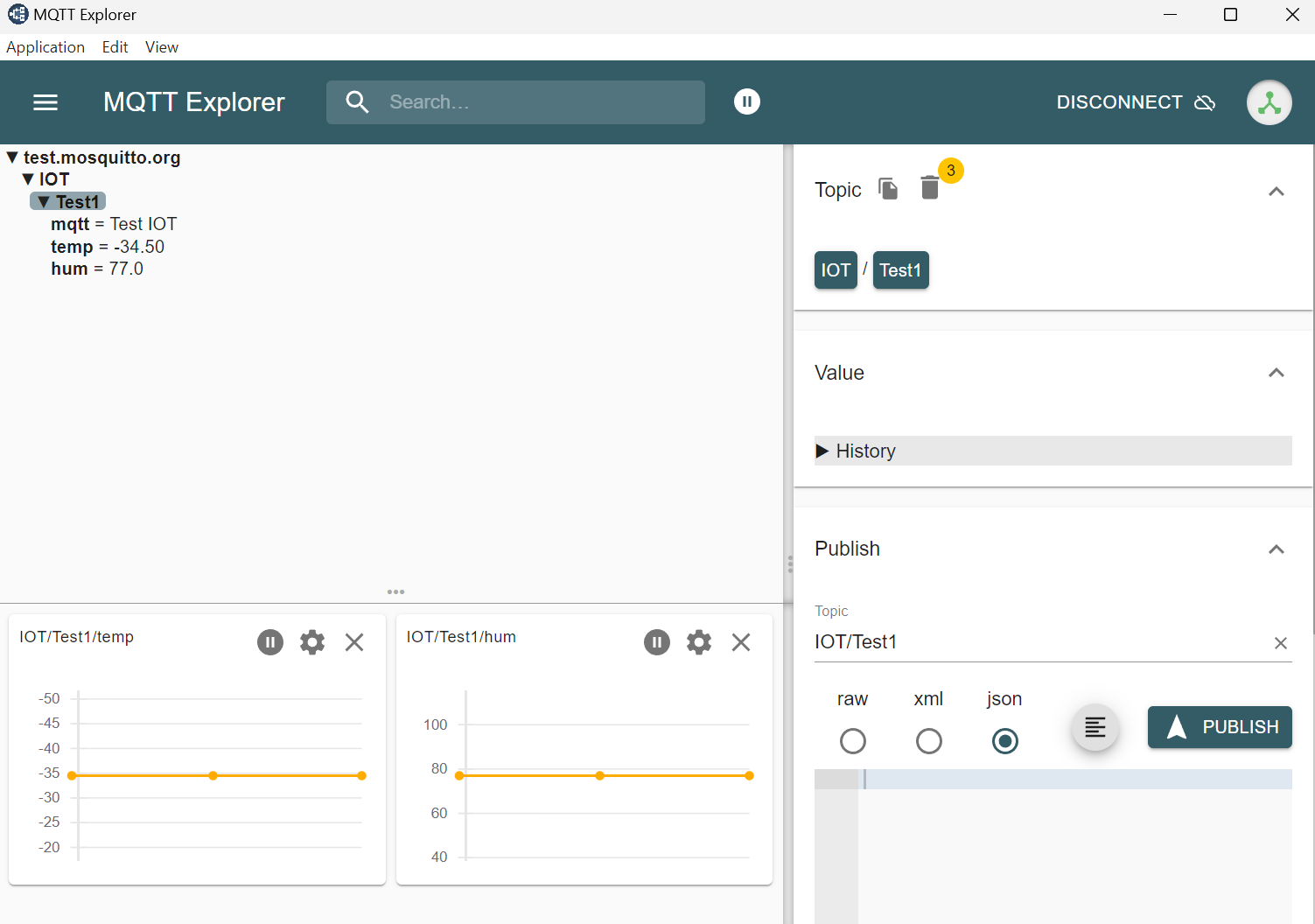
1. **Hasil dan Pembahasan**

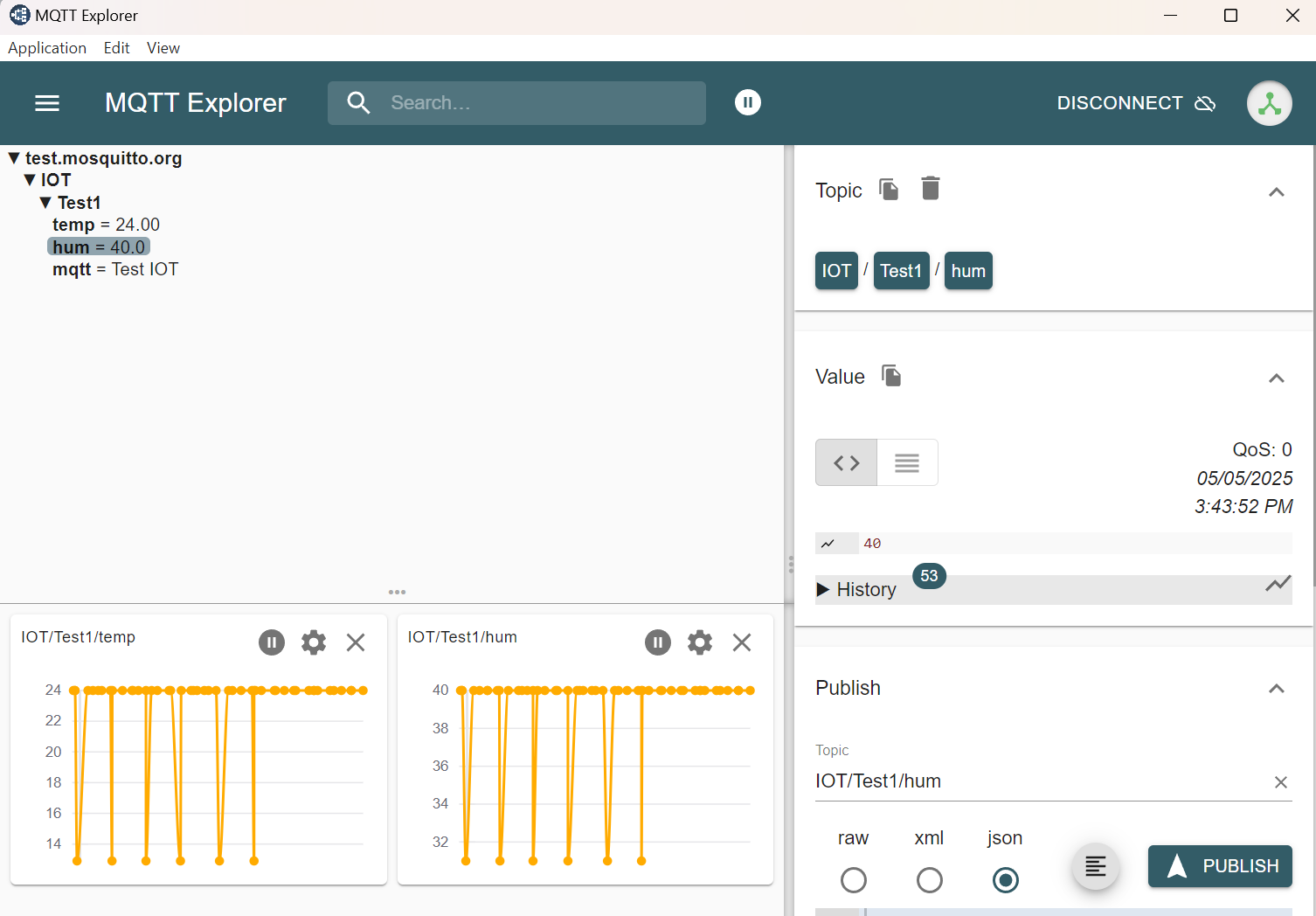
Setelah sistem dijalankan, ESP32 berhasil membaca data dari sensor DHT22 dan mengirimkannya ke broker MQTT menggunakan koneksi WiFi. Pada sisi client, data berhasil diterima dan ditampilkan secara real-time melalui aplikasi MQTT Explorer. Data yang dikirim terdiri dari dua parameter utama, yaitu suhu dalam satuan Celcius dan kelembapan dalam persen.Komunikasi antara ESP32 dan broker menggunakan protokol MQTT yang ringan, sehingga pengiriman data berlangsung cepat dan stabil. Selain itu, ESP32 mampu mempertahankan koneksi secara otomatis meskipun terjadi gangguan jaringan sesaat, dengan melakukan reconnect ke broker. Dalam pengujian, data dikirim setiap lima detik dan ditampilkan langsung di topik yang telah ditentukan.Pada dashboard MQTT Explorer, pengguna dapat melihat update nilai suhu dan kelembapan secara berkala. Hal ini membuktikan bahwa sistem publish-subscribe MQTT dapat bekerja secara efisien dan cocok digunakan untuk aplikasi pemantauan lingkungan berbasis IoT.Salah satu tantangan yang muncul adalah koneksi ke broker publik yang kadang mengalami delay karena banyaknya trafik dari pengguna lain. Namun, untuk tujuan pembelajaran dan praktikum, sistem dapat berjalan dengan baik dan stabil.











1. **Lampiran Kode**

* Kode Program Main.cpp

#include <WiFi.h>#include <PubSubClient.h>#include <DHTesp.h>const int LED\_RED = 2;const int DHT\_PIN = 15;DHTesp dht; // Update these with values suitable for your network.const char\* ssid = "Wokwi-GUEST";const char\* password = "";const char\* mqtt\_server = "test.mosquitto.org";//"broker.emqx.io";WiFiClient espClient;PubSubClient client(espClient);unsigned long lastMsg = 0;float temp = 0;float hum = 0;void setup\_wifi() { //perintah koneksi wifi delay(10); // We start by connecting to a WiFi network Serial.println(); Serial.print("Connecting to "); Serial.println(ssid); WiFi.mode(WIFI\_STA); //setting wifi chip sebagai station/client WiFi.begin(ssid, password); //koneksi ke jaringan wifi while (WiFi.status() != WL\_CONNECTED) { //perintah tunggu esp32 sampi terkoneksi ke wifi delay(500); Serial.print("."); } randomSeed(micros()); Serial.println(""); Serial.println("WiFi connected"); Serial.println("IP address: "); Serial.println(WiFi.localIP());}void callback(char\* topic, byte\* payload, unsigned int length) { //perintah untuk menampilkan data ketika esp32 di setting sebagai subscriber Serial.print("Message arrived ["); Serial.print(topic); Serial.print("] "); for (int i = 0; i < length; i++) { //mengecek jumlah data yang ada di topik mqtt Serial.print((char)payload[i]); } Serial.println(); // Switch on the LED if an 1 was received as first character if ((char)payload[0] == '1') { digitalWrite(LED\_RED, HIGH); // Turn the LED on } else { digitalWrite(LED\_RED, LOW); // Turn the LED off }}void reconnect() { //perintah koneksi esp32 ke mqtt broker baik itu sebagai publusher atau subscriber // Loop until we're reconnected while (!client.connected()) { Serial.print("Attempting MQTT connection..."); // perintah membuat client id agar mqtt broker mengenali board yang kita gunakan String clientId = "ESP32Client-"; clientId += String(random(0xffff), HEX); // Attempt to connect if (client.connect(clientId.c\_str())) { Serial.println("Connected"); // Once connected, publish an announcement... client.publish("IOT/Test1/mqtt", "Test IOT"); //perintah publish data ke alamat topik yang di setting // ... and resubscribe client.subscribe("IOT/Test1/mqtt"); //perintah subscribe data ke mqtt broker } else { Serial.print("failed, rc="); Serial.print(client.state()); Serial.println(" try again in 5 seconds"); // Wait 5 seconds before retrying delay(5000); } }}void setup() { pinMode(LED\_RED, OUTPUT); // inisialisasi pin 2 / ledbuiltin sebagai output Serial.begin(115200); setup\_wifi(); //memanggil void setup\_wifi untuk dieksekusi client.setServer(mqtt\_server, 1883); //perintah connecting / koneksi awal ke broker client.setCallback(callback); //perintah menghubungkan ke mqtt broker untuk subscribe data dht.setup(DHT\_PIN, DHTesp::DHT22);//inisialiasi komunikasi dengan sensor dht22}void loop() { if (!client.connected()) { reconnect(); } client.loop(); unsigned long now = millis(); if (now - lastMsg > 2000) { //perintah publish data lastMsg = now; TempAndHumidity data = dht.getTempAndHumidity(); String temp = String(data.temperature, 2); //membuat variabel temp untuk di publish ke broker mqtt client.publish("IOT/Test1/temp", temp.c\_str()); //publish data dari varibel temp ke broker mqtt String hum = String(data.humidity, 1); //membuat variabel hum untuk di publish ke broker mqtt client.publish("IOT/Test1/hum", hum.c\_str()); //publish data dari varibel hum ke broker mqtt Serial.print("Temperature: "); Serial.println(temp); Serial.print("Humidity: "); Serial.println(hum); }}

* Kode Program diagram.json

{ "version": 1, "author": "Maynanda", "editor": "wokwi", "parts": [ { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 0, "left": 0, "attrs": {} }, { "type": "wokwi-dht22", "id": "dht1", "top": -9.3, "left": -111, "attrs": {} }, { "type": "wokwi-led", "id": "led1", "top": 102, "left": 186.2, "attrs": { "color": "red" } } ], "connections": [ [ "esp:TX0", "$serialMonitor:RX", "", [] ], [ "esp:RX0", "$serialMonitor:TX", "", [] ], [ "dht1:GND", "esp:GND.2", "black", [ "v0" ] ], [ "dht1:VCC", "esp:3V3", "red", [ "v0" ] ], [ "dht1:SDA", "esp:D15", "green", [ "v0" ] ], [ "led1:C", "esp:GND.1", "green", [ "v0" ] ], [ "esp:D2", "led1:A", "green", [ "h61.9", "v-53.6", "h86.4", "v57.6" ] ] ], "dependencies": {}}