**Perancangan Sistem Monitoring IoT Real-Time Menggunakan Protokol MQTT dan Visualisasi Berbasis Web**

*Muhammad Fau Zan Sabani1*

*Fakultas Vokasi, Universitas Brawijaya*

*Email:* [*mfauzan18@student.ub.ac.id*](mailto:mfauzan18@student.ub.ac.id)

# Abstrak:

Laporan praktikum ini membahas tentang pembuatan sistem pemantauan suhu dan kelembapan berbasis *Internet of Things* (IoT) dengan memanfaatkan Wokwi sebagai platform simulasi dan MQTT sebagai protokol komunikasi. Proyek ini bertujuan untuk menerapkan teknologi IoT dalam memantau kondisi lingkungan secara *real-time*, yang datanya tidak hanya dikirim ke *broker* tetapi juga ditampilkan secara visual melalui antarmuka website.

Eksperimen dimulai dengan merancang rangkaian virtual di dalam Wokwi yang mengintegrasikan mikrokontroler ESP32, sensor DHT22, dan sebuah LED sebagai indikator. Kode program kemudian dibuat untuk membaca data sensor, mengirimkannya ke *broker* MQTT, serta mengendalikan nyala LED secara otomatis ketika suhu melewati ambang batas yang ditentukan. Langkah selanjutnya adalah mengembangkan sebuah website sederhana yang berfungsi sebagai *client* MQTT untuk menerima dan menampilkan data suhu dan kelembapan secara langsung.

Hasil utama dari praktikum ini menunjukkan bahwa seluruh alur sistem dapat berjalan dengan sukses, mulai dari pengiriman data oleh ESP32 di simulasi Wokwi, penerimaan oleh *broker* MQTT, hingga visualisasi data pada antarmuka website. Simulasi ini berhasil membuktikan bahwa Wokwi efektif untuk menguji fungsionalitas sistem IoT secara menyeluruh, termasuk integrasinya dengan platform eksternal seperti website, sebelum diimplementasikan pada perangkat keras yang sebenarnya.

Kata Kunci *Suhu, IoT, Kelembapan, ESP32, MQTT, Laporan Praktikum, Web.*

# Abstract: ,

TThis lab report discusses the creation of an Internet of Things (IoT)-based temperature and humidity monitoring system by utilizing Wokwi as a simulation platform and MQTT as a communication protocol. This project aims to apply IoT technology in monitoring real-time environmental conditions, whose data is not only sent to a broker but also displayed visually through a website interface.

The experiment starts by designing a virtual circuit in Wokwi that integrates an ESP32 microcontroller, a DHT22 sensor, and an LED as an indicator. Program code was then created to read the sensor data, send it to the MQTT broker, and automatically control the LED when the temperature crosses a specified threshold. The next step is to develop a simple website that functions as an MQTT client to receive and display temperature and humidity data directly.

The main results of this practicum show that the entire system flow can run successfully, starting from sending data by ESP32 in the Wokwi simulation, receiving by the MQTT broker, to visualizing data on the website interface. This simulation successfully proves that Wokwi is effective for testing the functionality of an IoT system as a whole, including its integration with external platforms such as websites, before it is implemented on actual hardware.

Keywords *Temperature, IoT, Humidity, ESP32, MQTT, Practical Report, Web.*

# 1. Pendahuluan

* 1. **Latar Belakang**

Pemantauan kondisi lingkungan seperti suhu dan kelembapan secara manual seringkali tidak efisien dan rentan terhadap keterlambatan dalam mendeteksi masalah. Di berbagai sektor, seperti ruang server, gudang penyimpanan, atau pertanian dalam ruangan, kegagalan menjaga kondisi yang ideal dapat menyebabkan kerusakan pada aset, penurunan kualitas produk, dan kerugian finansial yang signifikan.

Dengan perkembangan teknologi yang pesat, khususnya di bidang *Internet of Things* (IoT), terbuka peluang untuk menciptakan sistem pemantauan yang bekerja secara otomatis dan *real-time*. Sistem berbasis IoT memungkinkan sensor untuk mengumpulkan data secara terus-menerus dan mengirimkannya melalui jaringan. Hal ini tidak hanya memungkinkan pengambilan tindakan preventif dari jarak jauh, tetapi juga memudahkan visualisasi data melalui antarmuka yang ramah pengguna seperti website, sehingga informasi dapat dipahami dengan cepat dan akurat.

Oleh karena itu, praktikum yang menyimulasikan sistem pemantauan IoT secara *end-to-end* menjadi penting sebagai langkah awal untuk memahami implementasinya. Dalam proyek ini, protokol MQTT digunakan sebagai jalur komunikasi untuk mengirim data dari sensor. Wokwi berperan sebagai platform simulasi yang memungkinkan rangkaian diuji secara virtual, mulai dari pembacaan sensor oleh ESP32 hingga pengiriman data ke *broker* MQTT. Selanjutnya, data tersebut akan diterima dan ditampilkan pada sebuah antarmuka website, membuktikan bagaimana informasi dari sensor dapat diakses dengan mudah oleh pengguna akhir.

* 1. **Tujuan Eksperimen**

Tujuan dari eksperimen simulasi pemantauan IoT menggunakan MQTT dan Wokwi ini adalah sebagai berikut:

* Merancang dan mengimplementasikan sistem pemantauan suhu dan kelembapan secara *end-to-end*, mulai dari akuisisi data oleh sensor DHT22 hingga visualisasi *real-time* pada antarmuka website.
* Menguji dan memvalidasi fungsionalitas pengiriman data dari ESP32 ke *broker* MQTT menggunakan platform simulasi Wokwi sebelum diterapkan pada perangkat keras.
* Memahami arsitektur sistem IoT secara menyeluruh, mencakup mekanisme komunikasi *publish-subscribe* pada protokol MQTT dan cara mengintegrasikannya dengan antarmuka pengguna berbasis web.
* Mengembangkan kemampuan teknis dalam pemrograman mikrokontroler ESP32 serta dasar-dasar pengembangan web untuk visualisasi data yang diterima dari *broker* MQTT.
* Menyediakan dasar untuk membangun aplikasi IoT yang lebih lengkap dan ramah pengguna (*user-friendly*), di mana data tidak hanya dikumpulkan tetapi juga disajikan dalam bentuk informasi yang mudah dipahami.

# 2. Metodologi

* 1. **Alat dan Bahan**

Untuk melakukan praktikum simulasi, alat dan bahan yang digunakan

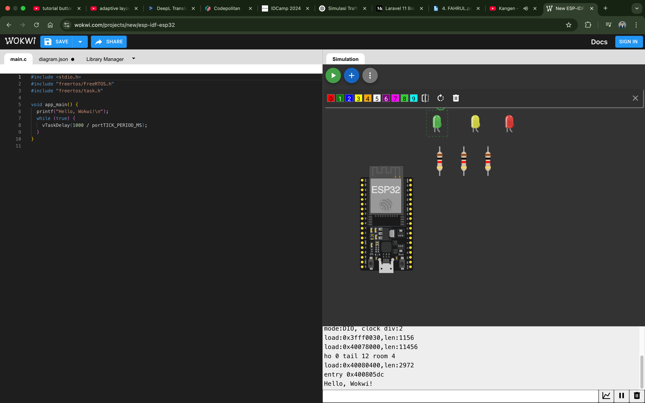
1. Mikrokontroler ESP32
2. Platform wokwi, digunakan untuk membuat simulasi
3. LED Merah
4. DHT 22
5. MQTT Explorer
6. Kabel Jumper
7. Arduino IDE/PlatformIO jika menggunakan Visual Studio Code
   1. **Langkah Implementasi**

Berikut langkah-langkah implementasi simulasi lampu lalu lintas dengan menggunakan Mikrokontroler ESP32 di Wokwi/Arduino:

* Install Arduino IDE atau jika menggunakan VsCode install Extension PlatfromIO di bagian ekstensinya. Buka juga website wokwi untuk tampilan simulasi nya secara virtual (jangan lupa untuk install ekstensi Wokwi juga di VsCode).



* Buat proyek baru di Wokwi dengan memilih ESP32 sebagai mikrokontroler. Tambahkan juga komponen LED dan sensor DHT22.

A computer screen shot of a computer

Description automatically generated

* Untuk pemogramannya menggunakan bahasa Arduino (C/C++) di Arduino IDE/PlatformIO (bisa juga langsung dimasukkan ke file ‘main.c’ di Wokwi).

#include <WiFi.h>

#include <PubSubClient.h>

#include <DHTesp.h>

const int LED\_RED = 2;

const int DHT\_PIN = 15;

DHTesp dht;

const char\* ssid = "Wokwi-GUEST";

const char\* password = "";

const char\* mqtt\_server = "test.mosquitto.org";

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);

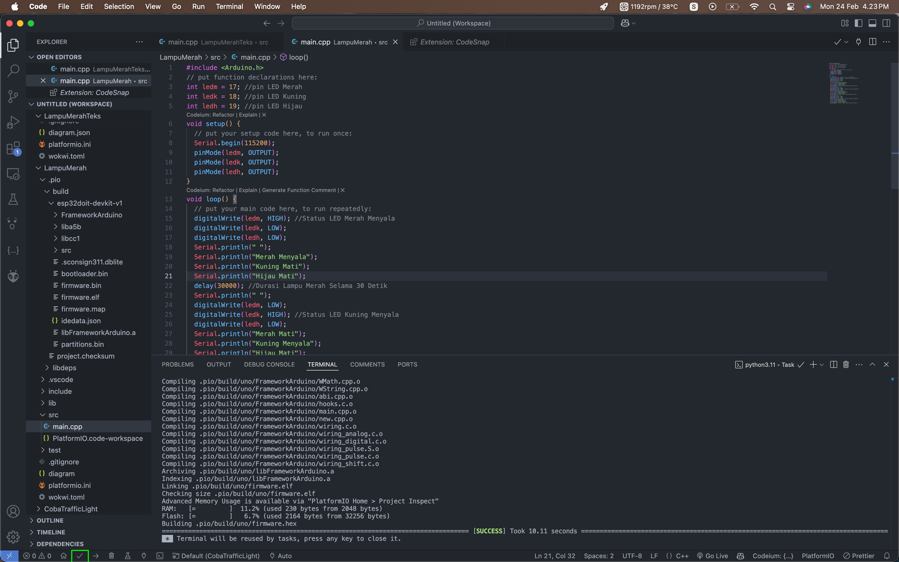
**Serial**.print("Humidity: ");

**Serial**.println(hum);

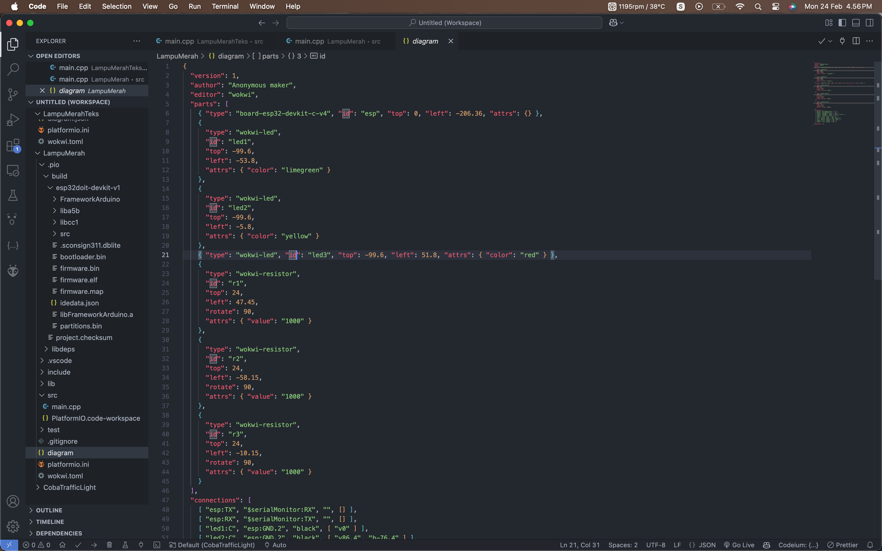
  }

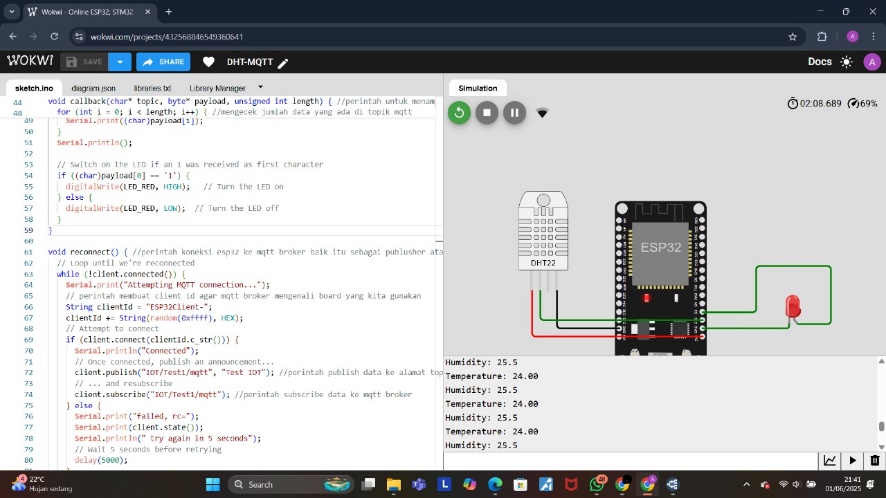
}

* Jika di platform Wokwi file ‘main.c’ tidak bisa ter-compile/dijalankan maka kitab isa mengetes sukses atau tidak kode tersebut di PlatformIO dengan menggunakan tombol centang dibawah.

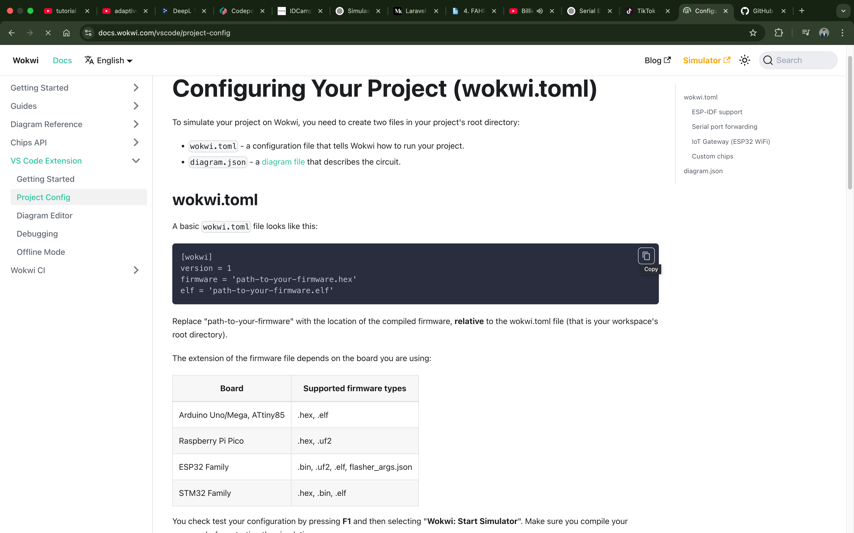


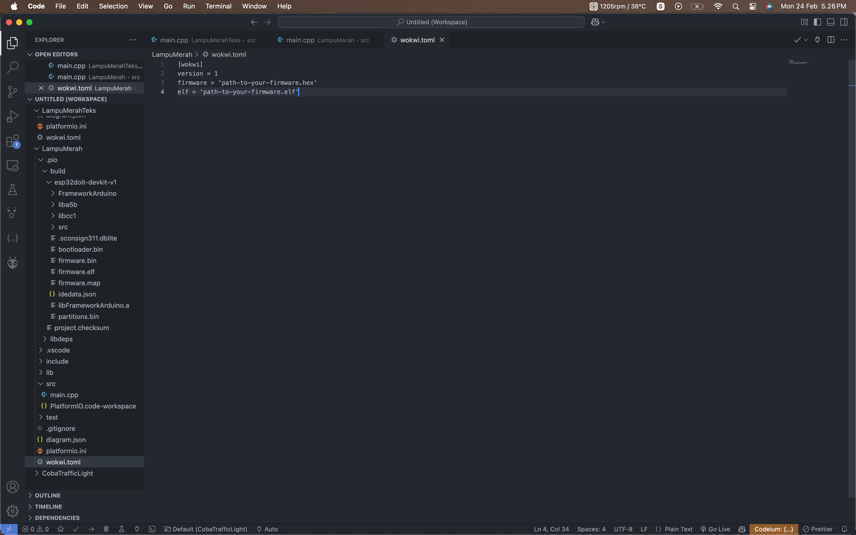
* Jika kode di file ‘main.cpp’ sukses di *Build* maka langkah selanjutnya untuk dapat melakukan virtualisasi simulasi lampu merah adalah membuat file bernama ‘diagram’ yang nanti diisi oleh kode dari file ‘diagram.json’ di projek Wokwi sebelumnya. Lalu kalau sudah, ubah nama file ‘diagram’ sbelumnya menjadi ‘diagram.json’ seperti di platform Wokwi sebelumnya.



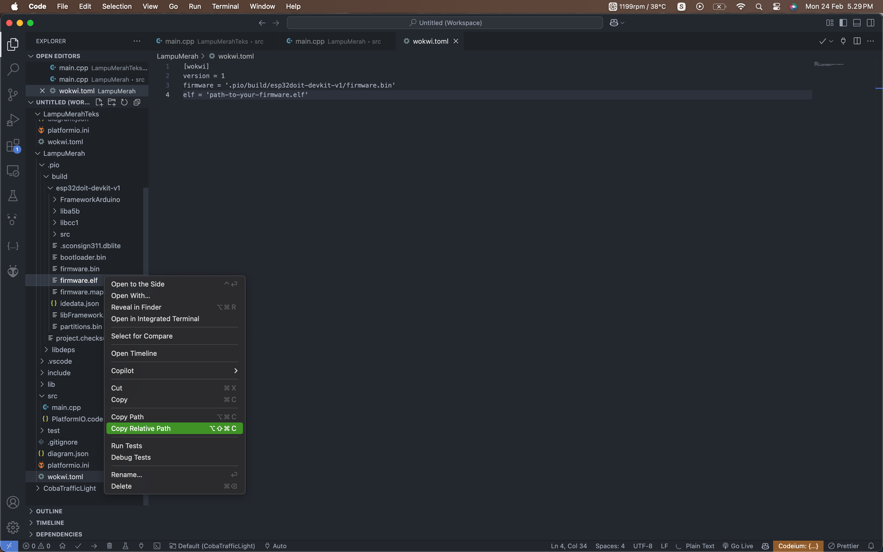


* Untuk dapat menjalankan simulasi dari file ‘diagram.json’ pada platform.io tersebut maka kita harus menambahkan file yang kita namakan ‘wokwi.toml’. Langkah berikutnya cari di web browser dengan keyword ‘Wokwi.toml’ dan salin kode konfigurasinya yang berasal dari halaman web dokumentasi wokwi tersebut ke file ‘wokwi.toml’ yang sudah kita buat.





* Langkah terakhir untuk dapat mengkonfigurasi file ‘wokwi.toml’ tersebut agar dapat membantu menjalankan file diagramnya maka kita perlu untuk mengganti path ‘firmware’ dan ‘elf’ path file yang berada di dalam rute folder ‘pio/build/esp32doit-devkit-v1’. Salin tiap file dengan format file yang bertuliskan ‘.hex’ dan ‘.elf’ dengan menggunakan klik kanan dan klik ‘Copy relative path’ lalu tempel ke path yang harus diisi di file ‘wokwi.toml’.



* Jalankan ulang file ‘diagram.json’ untuk dapat melihat hasil akhir simulasi yang sudah kita buat, dan juga lakukan uji coba pemantauan pada mqtt dengan memasukkan Topic sesuai pad akode yang kita buat sebelumnya, lalu untuk servernya menggunakan test.mosquitto.org.

A computer screen with a white screen

AI-generated content may be incorrect.

* Tambahkan file untuk desain web di direktori folder proyek di platform.io kita, contoh kode:

<!DOCTYPE html>

<html>

<head>

  <meta charset="UTF-8">

  <title>ESP Weather Station</title>

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <script src="https://unpkg.com/mqtt/dist/mqtt.min.js"></script>

  <script src="https://code.highcharts.com/highcharts.js"></script>

  <style>

    body {

      min-width: 310px;

      max-width: 800px;

      margin: 0 auto;

      font-family: Arial, sans-serif;

    }

    h2 {

      text-align: center;

      font-size: 2rem;

    }

    .slider {

      -webkit-appearance: none;

      width: 360px;

      height: 25px;

      background: #FFD65C;

      margin: 14px auto;

      display: block;

    }

    .slider::-webkit-slider-thumb {

      width: 35px;

      height: 35px;

      background: #003249;

      cursor: pointer;

    }

    #textSliderValue {

      text-align: center;

      display: block;

      font-weight: bold;

      font-size: 1.2rem;

    }

    .button-container {

      text-align: center;

      margin-top: 20px;

    }

    .button-container button {

      background-color: #007BFF;

      color: white;

      border: none;

      padding: 12px 24px;

      font-size: 1rem;

      cursor: pointer;

      border-radius: 5px;

    }

    .button-container button:hover {

      background-color: #0056b3;

    }

  </style>

</head>

<body>

  <h2>ESP Weather Station</h2>

  <div id="chart-temperature" class="container"></div>

  <div id="chart-humidity" class="container"></div>

  <p><span id="textSliderValue">PWM VALUE: 0</span></p>

  <input type="range" id="pwmSlider" min="0" max="255" value="0" class="slider" onchange="updateSliderPWM(this)">

  <div class="button-container">

    <button id="toggleButton" onclick="toggleLed()">Turn ON</button>

  </div>

  <script>

    const client = mqtt.connect('wss://test.mosquitto.org:8081');

    let ledState = false; // false = OFF, true = ON

    client.on('connect', () => {

      console.log("MQTT connected");

      client.subscribe('IOT/Test1/temp');

      client.subscribe('IOT/Test1/hum');

    });

    let chartT = Highcharts.chart('chart-temperature', {

      chart: { type: 'line' },

      title: { text: 'DHT Temperature' },

      xAxis: { type: 'datetime' },

      yAxis: { title: { text: 'Temperature (°C)' } },

      series: [{ name: 'Temp', data: [] }]

    });

    let chartH = Highcharts.chart('chart-humidity', {

      chart: { type: 'line' },

      title: { text: 'DHT Humidity' },

      xAxis: { type: 'datetime' },

      yAxis: { title: { text: 'Humidity (%)' } },

      series: [{ name: 'Humidity', data: [] }]

    });

    client.on('message', (topic, message) => {

      const payload = parseFloat(message.toString());

      const time = (new Date()).getTime();

      if (topic === 'IOT/Test1/temp') {

        chartT.series[0].addPoint([time, payload], true, chartT.series[0].data.length > 40);

      }

      if (topic === 'IOT/Test1/hum') {

        chartH.series[0].addPoint([time, payload], true, chartH.series[0].data.length > 40);

      }

    });

    function updateSliderPWM(element) {

      const val = element.value;

      document.getElementById("textSliderValue").innerText = `PWM VALUE: ${val}`;

      client.publish("IOT/Test1/pwm", val.toString());

    }

    function toggleLed() {

      ledState = !ledState;

      const payload = ledState ? "1" : "0";

      client.publish("IOT/Test1/mqtt", payload);

      document.getElementById("toggleButton").innerText = ledState ? "Turn OFF" : "Turn ON";

    }

  </script>

</body>

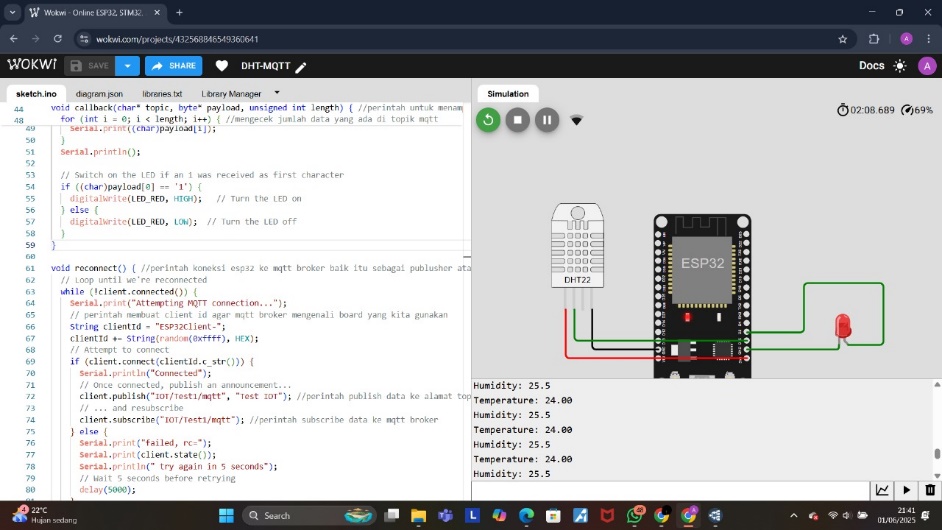
</html>

# 3. Hasil dan Pembahasan

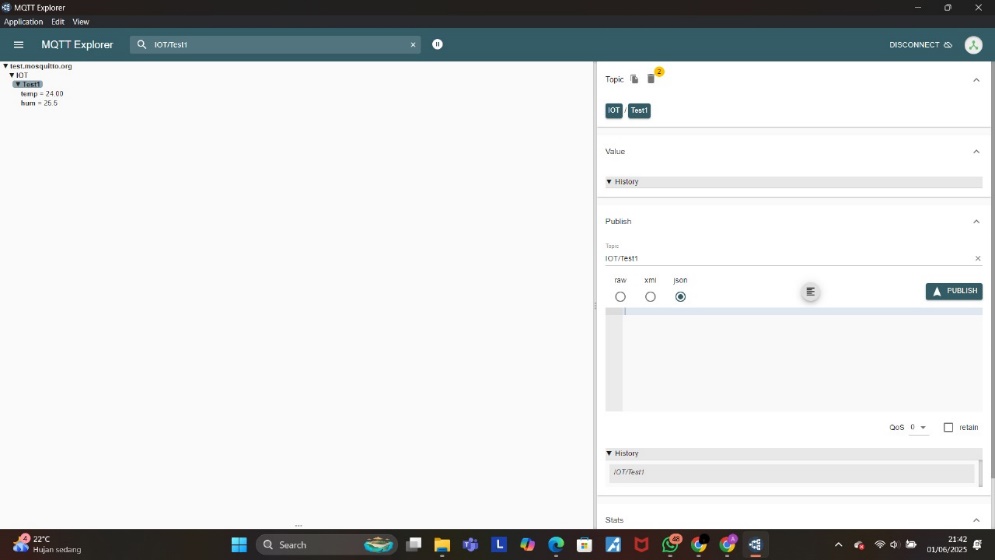
1. **Hasil Eksperimen**

Dari simulasi yang sudah dibuat, di dapatkan hasil yang sudah sesuai dengan kode yang kita buat sebelumnya. Berikut hasil simulasinya:

* Hasil yang ditampilkan di Wokwi:



* Untuk data yang ditampilkan di MQTT Explorer



# 4. Lampiran

* Kode Program .json:

{

  "version": 1,

  "author": "Subairi",

  "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": {}

}