

Laporan

Praktikum Internet of Things (IoT) Membuat Rangkaian
Pendeteksi Temperatur dan Humidity Menggunakan Wokwi
yang Terhubung Ke Mqtt dan Website

Alwulida Nur Aini Umma

Fakultas Vokasi, Universitas Brawijaya

Email : alwulidaumma@student.ub.ac.id

Abstrak

Internet of Things (IoT) merupakan pendekatan teknologi yang memungkinkan berbagai perangkat fisik saling terhubung dan bertukar data melalui jaringan internet. Dalam praktikum ini, dilakukan perancangan dan simulasi rangkaian pendeteksi temperatur dan kelembapan berbasis IoT menggunakan simulator Wokwi, protokol MQTT sebagai jalur komunikasi data, dan integrasi dengan website sebagai media pemantauan.

ESP32 digunakan sebagai mikrokontroler utama yang dikendalikan melalui pemrograman C++ di editor Visual Studio Code (VSCode). Sensor DHT22 berfungsi sebagai pendeteksi suhu dan kelembapan, yang mengirimkan data secara real-time ke broker MQTT. Data tersebut kemudian diteruskan ke website untuk divisualisasikan dalam bentuk dashboard pemantauan yang dapat diakses secara online.

Tujuan dari praktikum ini adalah memberikan pemahaman praktis mengenai integrasi sensor, mikrokontroler, protokol MQTT, serta konektivitas dengan website sebagai bagian dari sistem monitoring lingkungan yang efisien dan responsif. Simulasi menunjukkan bahwa sistem berhasil membaca, mengirim, dan menampilkan data suhu serta kelembapan dengan stabil melalui website.

Kata kunci : *Internet of Things, Temperatur, Humidity, ESP32, MQTT, Wokwi, Visual Studio Code, DHT22, Website, Pemantauan Lingkungan.*

Pendahuluan

Perkembangan teknologi Internet of Things (IoT) telah membawa perubahan besar dalam berbagai sektor kehidupan manusia, khususnya dalam hal otomasi, pengumpulan data, dan pemantauan sistem secara real-time. IoT memungkinkan perangkat fisik seperti sensor, aktuator, serta mikrokontroler untuk saling terhubung dan berkomunikasi melalui jaringan internet tanpa memerlukan intervensi langsung dari pengguna. Teknologi ini memberikan efisiensi tinggi dalam pemantauan kondisi lingkungan dan pengambilan keputusan berbasis data yang akurat.

Salah satu implementasi IoT yang banyak digunakan adalah sistem pemantauan suhu dan kelembapan yang sangat relevan dalam bidang pertanian, industri makanan, rumah pintar, serta sistem kesehatan. Untuk mendukung komunikasi data yang cepat dan efisien antara perangkat IoT, protokol Message Queuing Telemetry Transport (MQTT) menjadi pilihan yang umum digunakan. MQTT menggunakan sistem komunikasi publish-subscribe yang ringan dan cocok untuk perangkat dengan keterbatasan sumber daya, seperti mikrokontroler.

Pada praktikum ini, dilakukan perancangan dan simulasi rangkaian sistem pendeteksi suhu dan kelembapan menggunakan sensor DHT22 dan mikrokontroler ESP32. Seluruh sistem disimulasikan melalui platform Wokwi, yang memungkinkan mahasiswa melakukan pengujian perangkat keras secara virtual tanpa perlu perangkat fisik. Data dari sensor dikirimkan ke broker MQTT secara real-time melalui koneksi internet simulatif yang disediakan oleh Wokwi. Proses pemrograman dilakukan menggunakan bahasa C++ dengan bantuan Visual Studio Code (VSCode) sebagai lingkungan pengembangan.

Tujuan dari praktikum ini adalah untuk memahami proses integrasi antara perangkat IoT dengan protokol MQTT, serta mengembangkan kemampuan dalam merancang sistem pemantauan lingkungan yang dapat ditingkatkan untuk aplikasi nyata. Dengan simulasi ini, mahasiswa diharapkan memperoleh pengalaman praktis dalam membangun solusi IoT berbasis sensor dan komunikasi cloud yang efisien, andal, dan mudah dikembangkan.

Metodologi

Dalam praktikum simulasi pembuatan rangkaian pendeteksi temperatur dan humidity yang berbasis Internet of Things (IoT) ini, digunakan beberapa alat dan bahan utama yang mendukung proses perancangan, pemrograman, serta pengujian sistem. Adapun alat dan bahan yang digunakan meliputi:

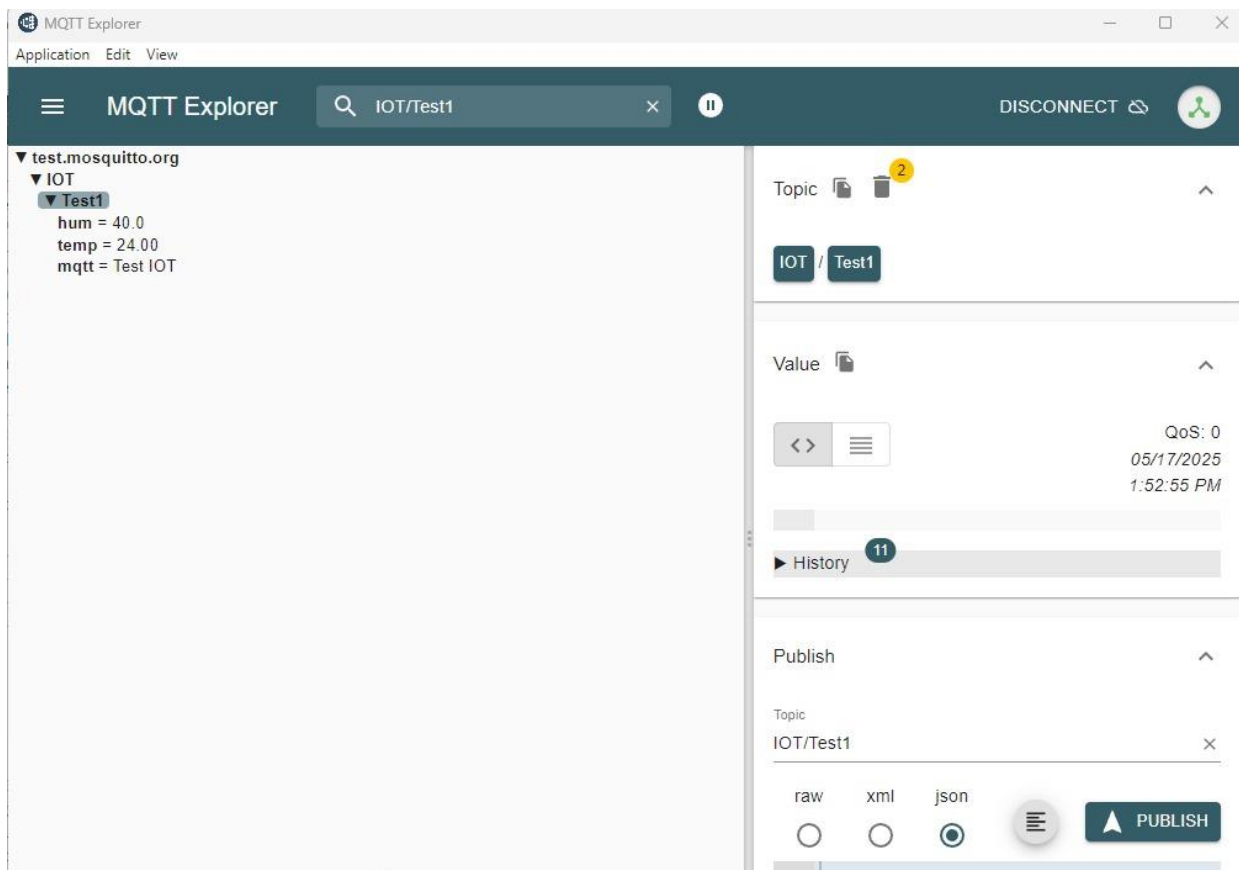
1. Perangkat Lunak dan Kebutuhan Sistem
 - a) Laptop atau Komputer
Digunakan sebagai media untuk mengakses platform Wokwi, menulis kode program, serta memantau proses simulasi dan data yang dikirimkan ke broker MQTT dan website .
 - b) Koneksi Internet
Diperlukan untuk menjalankan simulasi di Wokwi serta menghubungkan ESP32 dengan broker MQTT dan website secara online.
 - c) Wokwi Simulator
Platform simulasi online yang digunakan untuk merancang dan menjalankan sistem secara virtual tanpa perangkat keras fisik. Dalam simulasi ini, komponen-komponen yang digunakan meliputi:
 - ESP32 DevKit V1
Sebagai mikrokontroler utama yang mengelola pembacaan data dari sensor serta koneksi ke broker MQTT melalui jaringan WiFi.
 - Sensor DHT22
Digunakan untuk mendeteksi data suhu dan kelembapan lingkungan. Sensor ini terhubung ke pin digital D15 pada ESP32.
 - LED Merah
Digunakan sebagai indikator sederhana, misalnya untuk menunjukkan status koneksi atau peringatan suhu/kelembapan tinggi. LED ini dikontrol melalui pin D2.
 - Serial Monitor
Digunakan untuk menampilkan hasil pembacaan data sensor dan status koneksi selama proses simulasi berlangsung.
 - d) Visual Studio Code (Vs Code)
Digunakan untuk membuat codingan website dan tampilannya (CSS).
 - e) Platform
Digunakan untuk mengakses website yang telah terhubung dengan wokwi dan Mqtt.
2. Konfigurasi Koneksi pada Wokwi
Berdasarkan simulasi, berikut konfigurasi hubungan antar komponen:
 - a) DHT22 terhubung ke ESP32:
 - VCC → 3V3
 - GND → GND
 - SDA (data) → D15
 - b) LED Merah:
 - Anoda (A) → D2
 - Katoda (C) → GND
 - c) ESP32 juga terhubung ke Serial Monitor melalui pin TX0 dan RX0 untuk keperluan debugging.
3. Pemrograman Sistem
 - a) Visual Studio Code (VSCode) digunakan sebagai editor kode untuk menulis program website dan uji dari wokwi menggunakan bahasa C++ dengan library pendukung seperti DHT.h dan PubSubClient.h.

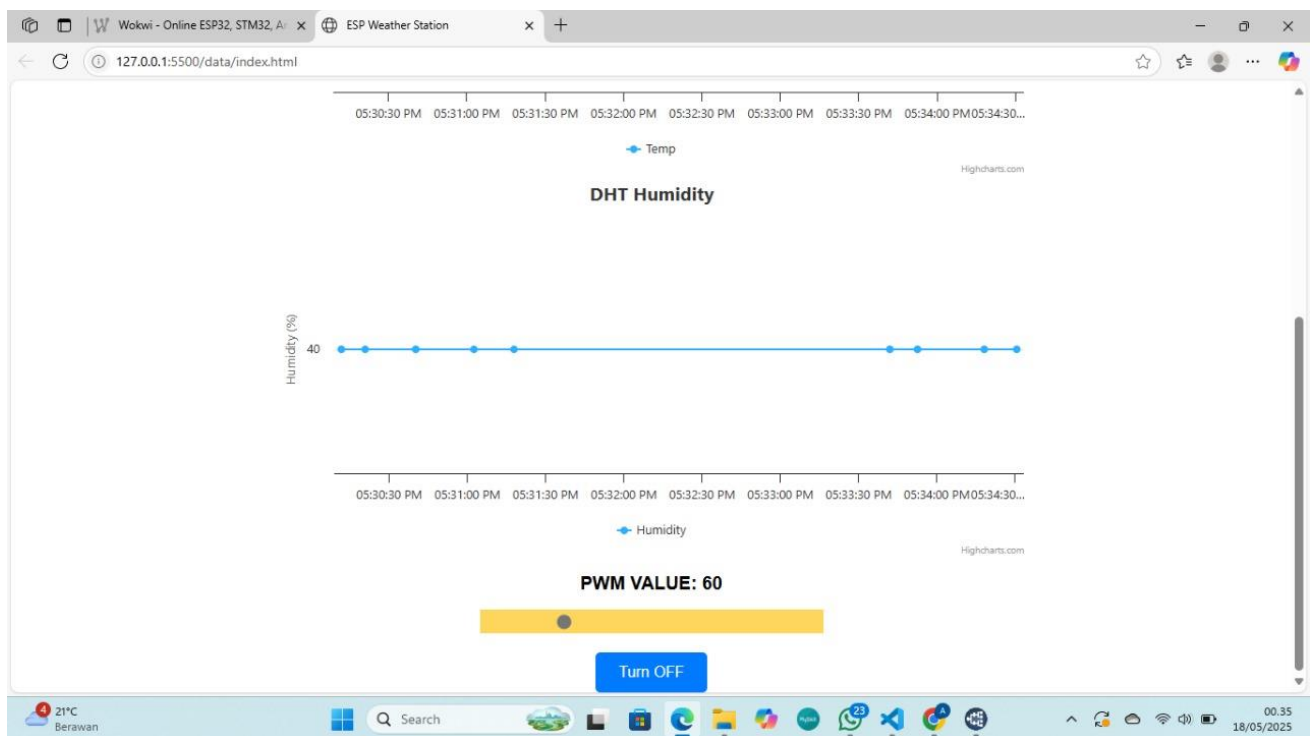
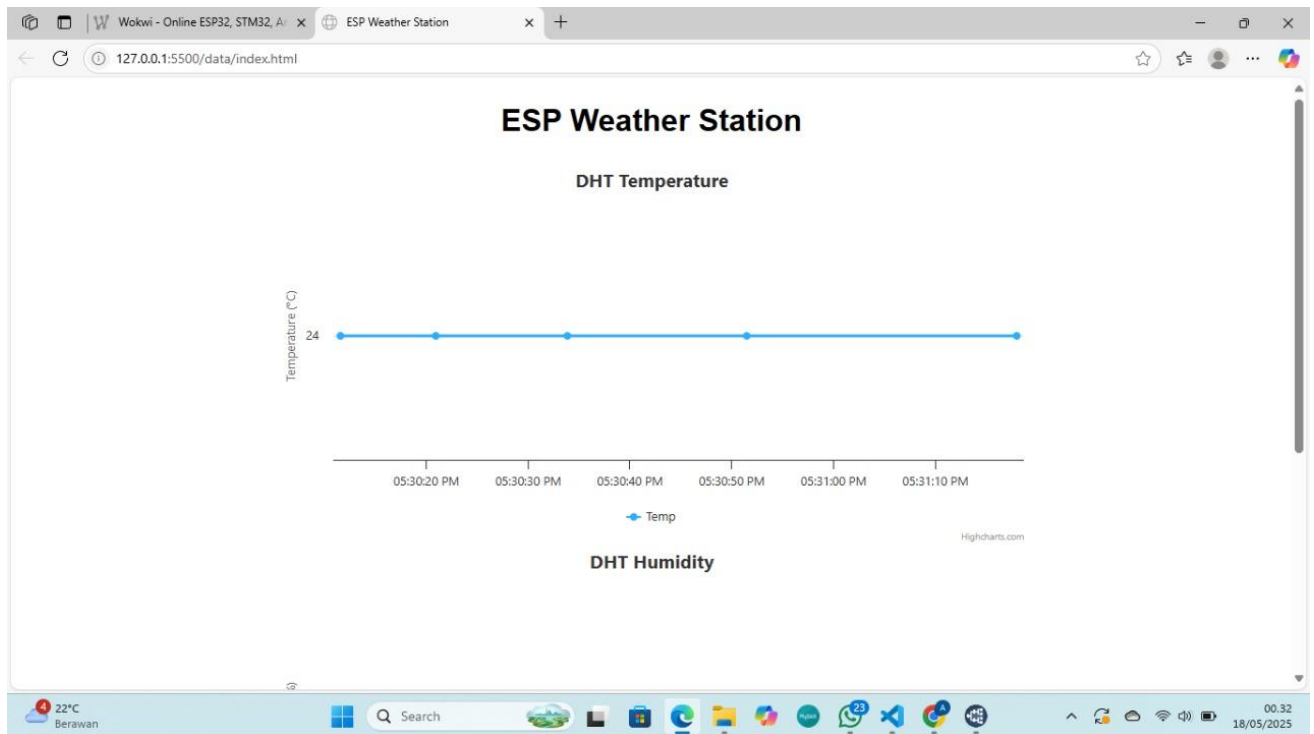
- b) Program ditulis untuk:
- o Membaca data dari sensor DHT22
 - o Menghubungkan ESP32 ke jaringan WiFi simulasi
 - o Mengirim data suhu dan kelembapan ke broker MQTT (seperti broker gratis broker.hivemq.com atau broker lokal)
4. Pengujian dan Simulasi
- Setelah semua koneksi dan kode selesai, simulasi dijalankan di Wokwi.
 - Serial monitor digunakan untuk memastikan bahwa data sensor terbaca dengan benar dan status koneksi MQTT berhasil.
 - LED dapat difungsikan sebagai indikator visual (misalnya menyala saat kelembapan melebihi ambang batas).
 - Vs Code akan menjalankan website di platform seperti chorme yang telah terhubung oleh wokwi yang dihubungkan menggunakan Mqtt.

Hasil dan Pembahasan

- Hasil Eksperimen

Berdasarkan hasil simulasi dari kode yang dibuat di Wokwi yang terhubung ke Aplikasi Mqtt dan Website, hasil dari run kode di wokwi akan muncul juga pada aplikasi Mqtt dan Website.





Untuk mendapatkan hasil tersebut, dibutuhkan beberapa kode program yang kode tersebut memiliki fungsi yang berbeda-beda, seperti berikut ini :

1. Kode sketch.ino pada wokwi yang terdapat Kode yang terhubung ke Mqtt :

```
#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 sampe 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); // inialisasi 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); //inialisasi 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);
}

```

```
}
```

2. Kode diagram.json pada wokwi :

```
{
  "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": { }
}
```

3. Kode index.html di Vs Code :

```
<!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());
}

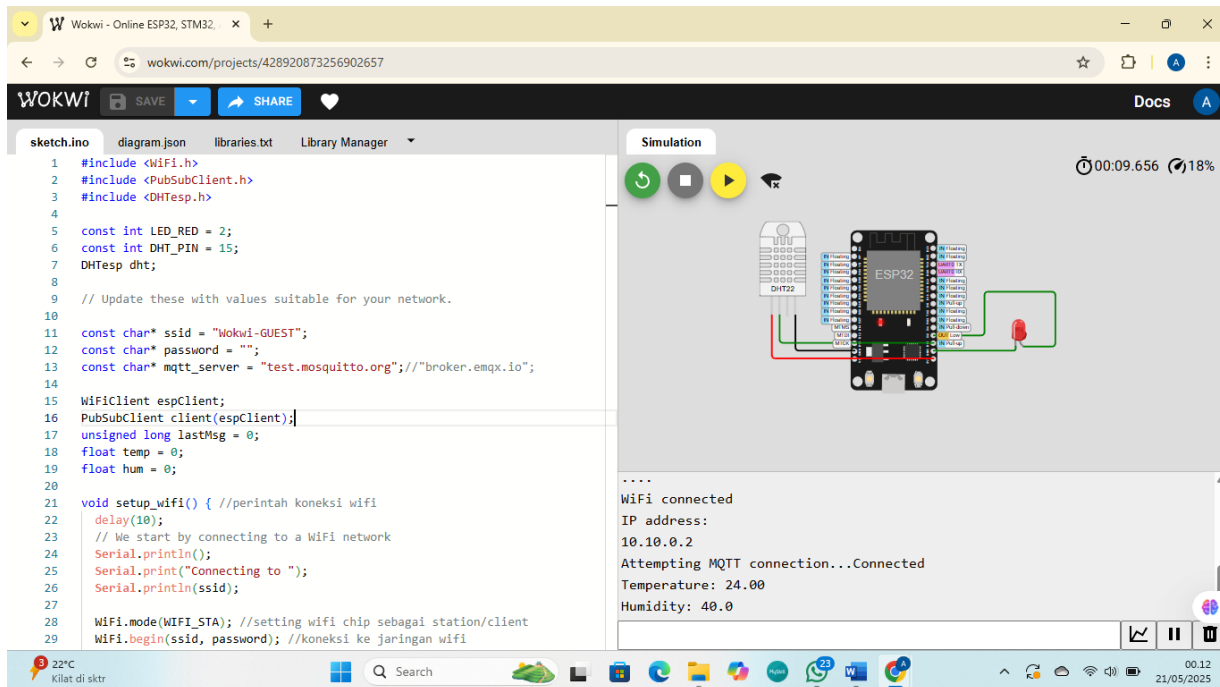
function toggleLed() {
  ledState = !ledState;
  const payload = ledState ? "1" : "0";
  client.publish("IOT/Test1/mqtt", payload);
  document.getElementById("toggleButton").innerText = ledState ? "Turn OFF" : "Turn ON";
}
</script>

</body>
</html>

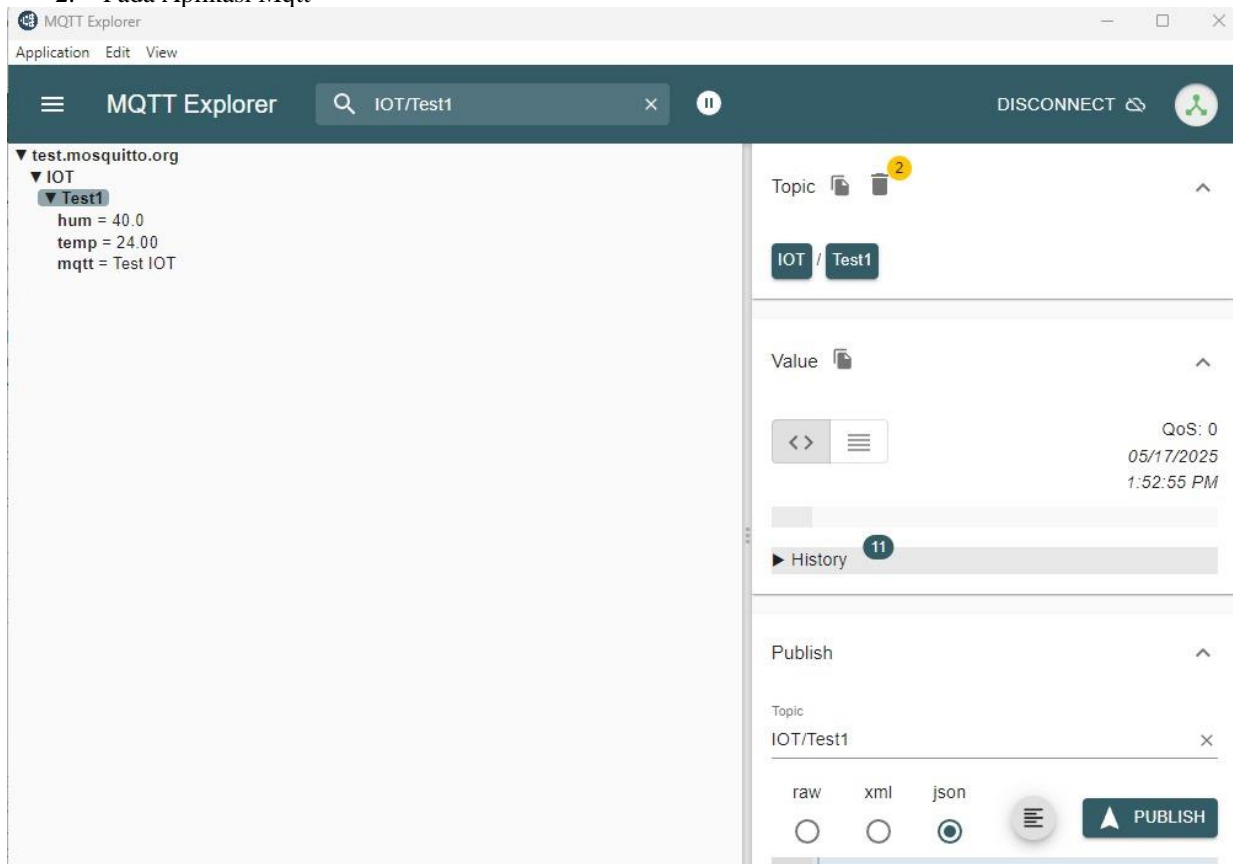
```

- Lampiran

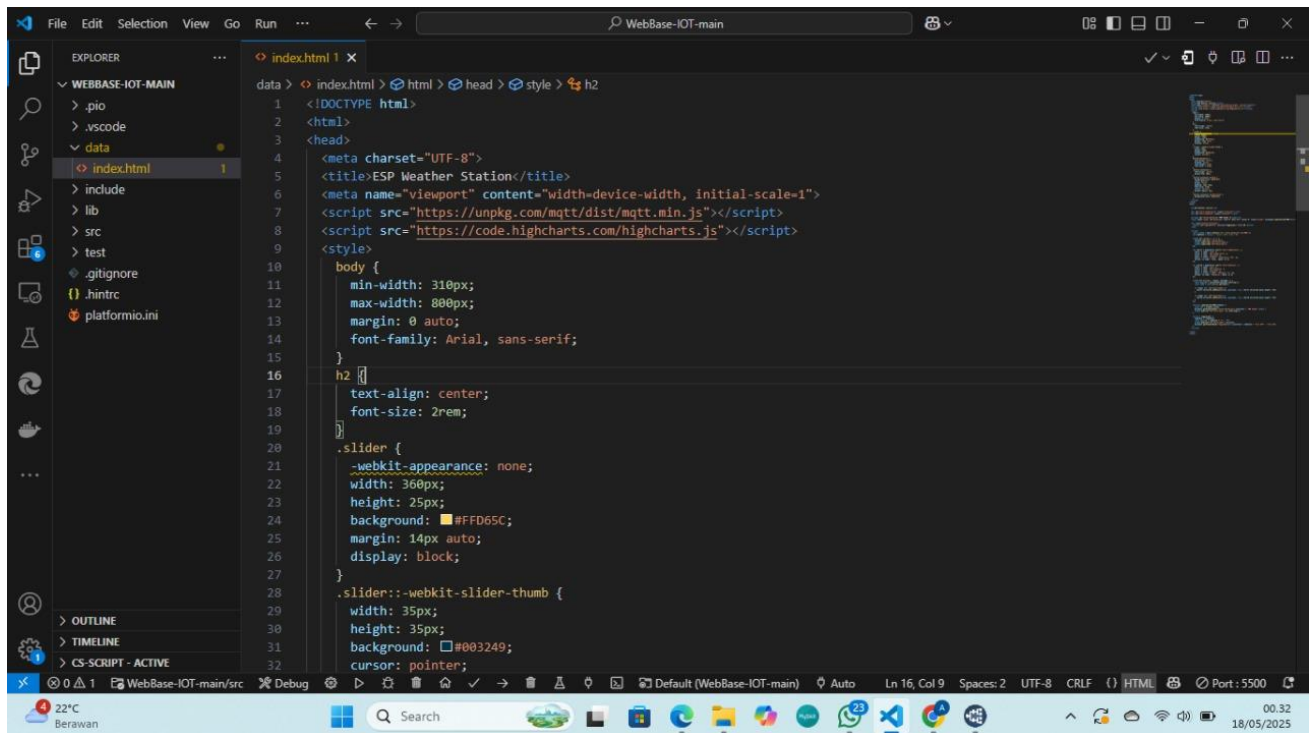
1. Pada Website Wokwi



2. Pada Aplikasi Mqtt



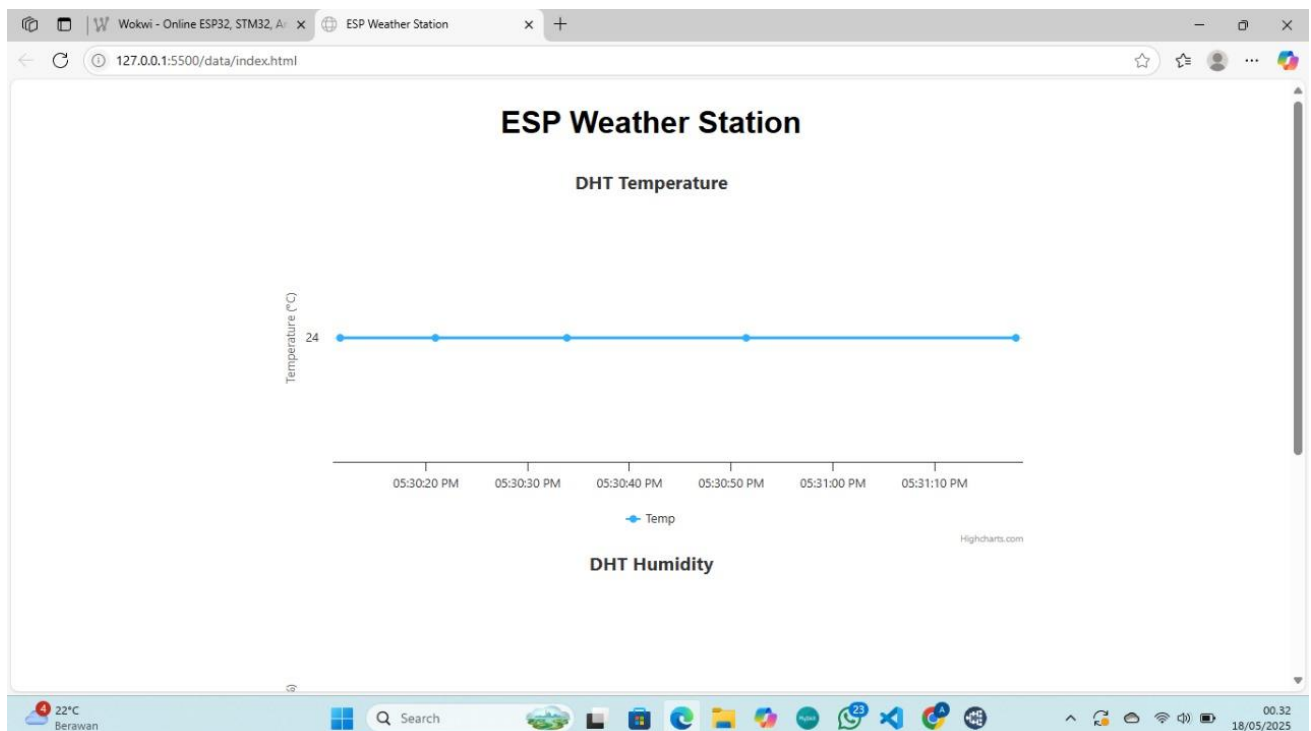
3. Pada Vs Code

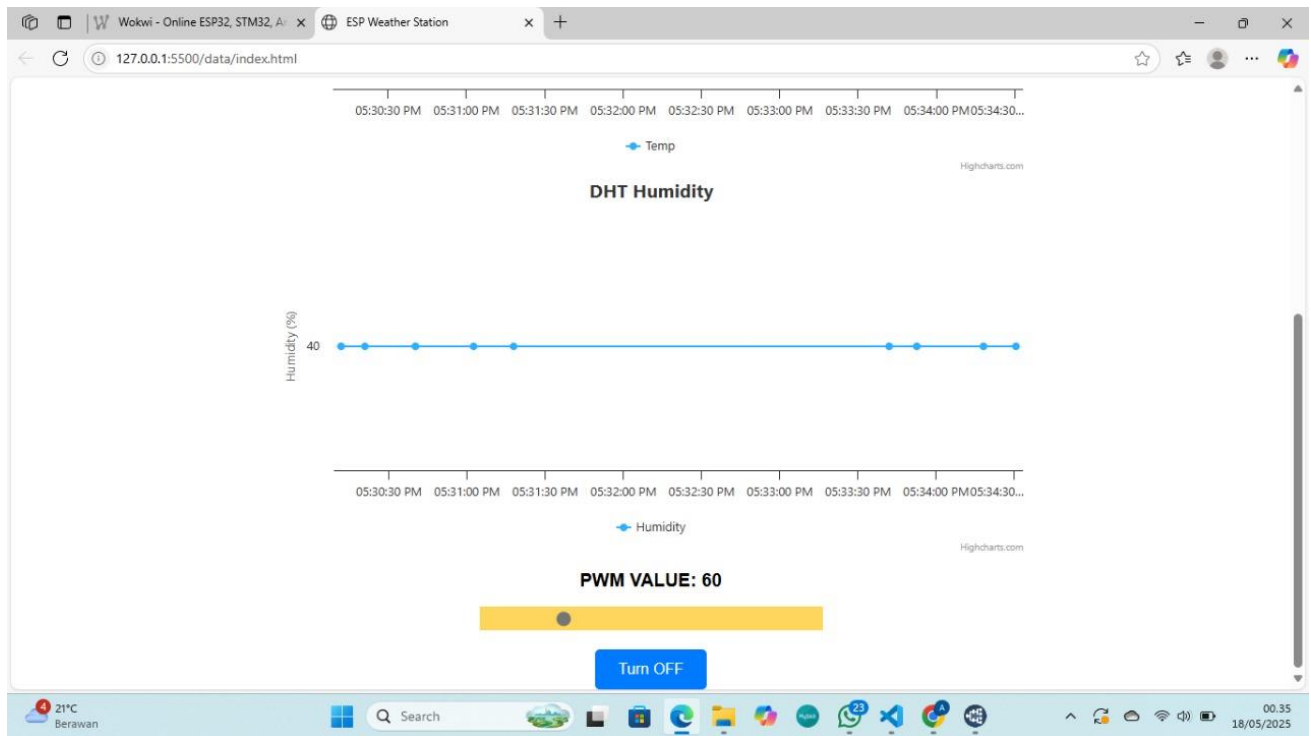


The screenshot shows the VS Code editor interface with the 'index.html' file open. The file contains HTML and CSS code for a web page titled 'ESP Weather Station'. The code includes a meta charset of 'UTF-8', a viewport meta tag, and links to jQuery and Highcharts. The CSS defines a body with a min-width of 310px, a max-width of 800px, and a font-family of Arial, sans-serif. A slider is styled with a width of 360px, height of 25px, and a background color of #FFD65C. The slider thumb is styled with a width of 35px, height of 35px, and a background color of #003249.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>ESP Weather Station</title>
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <script src="https://unpkg.com/jquery/dist/jquery.min.js"></script>
8   <script src="https://code.highcharts.com/highcharts.js"></script>
9   <style>
10     body {
11       min-width: 310px;
12       max-width: 800px;
13       margin: 0 auto;
14       font-family: Arial, sans-serif;
15     }
16     h2 {
17       text-align: center;
18       font-size: 2rem;
19     }
20     .slider {
21       -webkit-appearance: none;
22       width: 360px;
23       height: 25px;
24       background: #FFD65C;
25       margin: 14px auto;
26       display: block;
27     }
28     .slider::-webkit-slider-thumb {
29       width: 35px;
30       height: 35px;
31       background: #003249;
32       cursor: pointer;
```

4. Pada Website





Link Pengerjaan Wokwi

<https://wokwi.com/projects/428920873256902657>

