Laporan Praktikum Implementasi Wokwi dengan MQTT dan Website

Nasywa Anindya Q.E

Fakultas Vokasi, Universitas Brawijaya

Email : nasywaanindya@student.ub.ac.id

Abstrak

Praktikum ini bertujuan untuk menampilkan dan mengirimkan data suhu dan kelembapan secara real-time menggunakan sensor DHT22 yang terhubung dengan mikrokontroler ESP32. Data yang diperoleh dari sensor akan dikirimkan ke broker MQTT melalui koneksi internet menggunakan protokol MQTT, kemudian dapat dipantau melalui aplikasi MQTT Explorer. Selain itu, sistem juga dilengkapi dengan LED berwarna merah yang akan menyala sebagai indikator jika suhu melewati ambang batas tertentu. Implementasi ini menggunakan bahasa pemrograman Arduino dengan PlatformIO sebagai lingkungan pengembangannya, dan Wokwi digunakan sebagai platform simulasi untuk menguji integrasi antara ESP32, sensor DHT22, LED merah, dan konektivitas MQTT sebelum diterapkan pada perangkat fisik. Dengan adanya praktikum ini, diharapkan mahasiswa dapat memahami konsep dasar komunikasi data IoT menggunakan MQTT serta implementasi pemantauan lingkungan secara otomatis melalui sistem berbasis mikrokontroler.

Kata Kunci — *IoT, ESP32, MQTT, DHT22, Wokwi, LED*

Abstact

This practicum aims to display and transmit real-time temperature and humidity data using a DHT22 sensor connected to an ESP32 microcontroller. The data obtained from the sensor will be sent to the MQTT broker via an internet connection using the MQTT protocol, then can be monitored through the MQTT Explorer application. In addition, the system is also equipped with a red LED that will light up as an indicator if the temperature crosses a certain threshold. This implementation uses the Arduino programming language with PlatformIO as the development environment, and Wokwi is used as a simulation platform to test the integration between the ESP32, DHT22 sensor, red LED, and MQTT connectivity before being applied to physical devices. With this practicum, students are expected to understand the basic concepts of IoT data communication using MQTT and the implementation of automatic environmental monitoring through microcontroller-based systems.

Key Words — *IoT, ESP32, MQTT, DHT22, Wokwi, LED*

1. Pendahuluan

Internet of Things (IoT) adalah konsep yang memungkinkan berbagai perangkat elektronik saling terhubung dan bertukar data melalui jaringan tanpa memerlukan interaksi manusia secara langsung. Melalui IoT, perangkat seperti sensor, mikrokontroler, dan aktuator dapat bekerja secara otomatis untuk mengumpulkan, memproses, dan mengirimkan data ke sistem pusat atau pengguna akhir. Salah satu implementasi penting dari IoT adalah sistem pemantauan dan pengendalian lingkungan secara real-time.

Pada praktikum kali ini, dilakukan simulasi pemantauan suhu dan kelembapan menggunakan sensor DHT22 yang terhubung ke mikrokontroler ESP32. Data yang diperoleh akan dikirim secara otomatis ke broker MQTT dan dapat dipantau melalui aplikasi MQTT Explorer. Selain itu, digunakan LED berwarna merah sebagai indikator visual: LED akan menyala jika suhu atau kelembapan melebihi ambang batas tertentu, dan akan mati jika nilainya rendah. Praktikum ini bertujuan untuk memberikan pemahaman kepada mahasiswa mengenai komunikasi data berbasis protokol MQTT serta implementasi aksi otomatis berdasarkan data sensor.

1.1 Latar Belakang

Di era digital saat ini, kebutuhan akan sistem yang dapat memantau dan merespons kondisi lingkungan secara otomatis dan real-time semakin meningkat. Salah satu cara untuk memenuhi kebutuhan tersebut adalah dengan mengembangkan sistem berbasis IoT yang dapat mengumpulkan data lingkungan dan mengirimkannya ke pengguna atau sistem pemrosesan melalui jaringan internet. Data suhu dan kelembapan merupakan dua parameter penting dalam banyak bidang, seperti pertanian, industri, dan sistem kontrol ruangan.

Melalui praktikum ini, dikembangkan sebuah sistem simulasi menggunakan platform Wokwi yang mengintegrasikan sensor DHT22, mikrokontroler ESP32, dan LED sebagai indikator. Data yang dikumpulkan dari sensor dikirimkan ke broker MQTT menggunakan koneksi WiFi, dan dapat dipantau melalui MQTT Explorer. Setelah itu akan masuk ke dalam website yang berisi data yang telah disimpan pada MQTT. Dengan pendekatan ini, mahasiswa diharapkan dapat memahami proses pengumpulan, pengiriman, dan pemanfaatan data sensor secara efisien melalui protokol MQTT dalam sistem IoT.

1.2 Tujuan Praktikum

Ada beberapa tujuan diadakannya praktikum ini sebagai berikut :

1. Mempelajari cara menghubungkan Wokwi/VSCode dengan MQTT.
2. Mempelajari cara menggunakan MQTT untuk mengambil data dari wokwi.
3. Menghubungkan data yang telah disimpan pada MQTT ke dalam Website.
4. Menampilkan data pada website yang telah dibuat.

2. Metodologi

2.1 Alat dan Bahan

Untuk melakukan praktikum simulasi lampu lalu lintas berbasis IoT, alat dan bahan yang digunakan

a) Mikrokontroler ESP32
b) Platform wokwi, digunakan untuk membuat simulasi
c) LED Merah
d) DHT 22
e) MQTT Explorer
f) Kabel Jumper
g) Arduino IDE/PlatformIO jika menggunakan Visual Studio Code

2.2 Langkah Implementasi

Terdapat beberapa Langkah pada praktikum kali ini

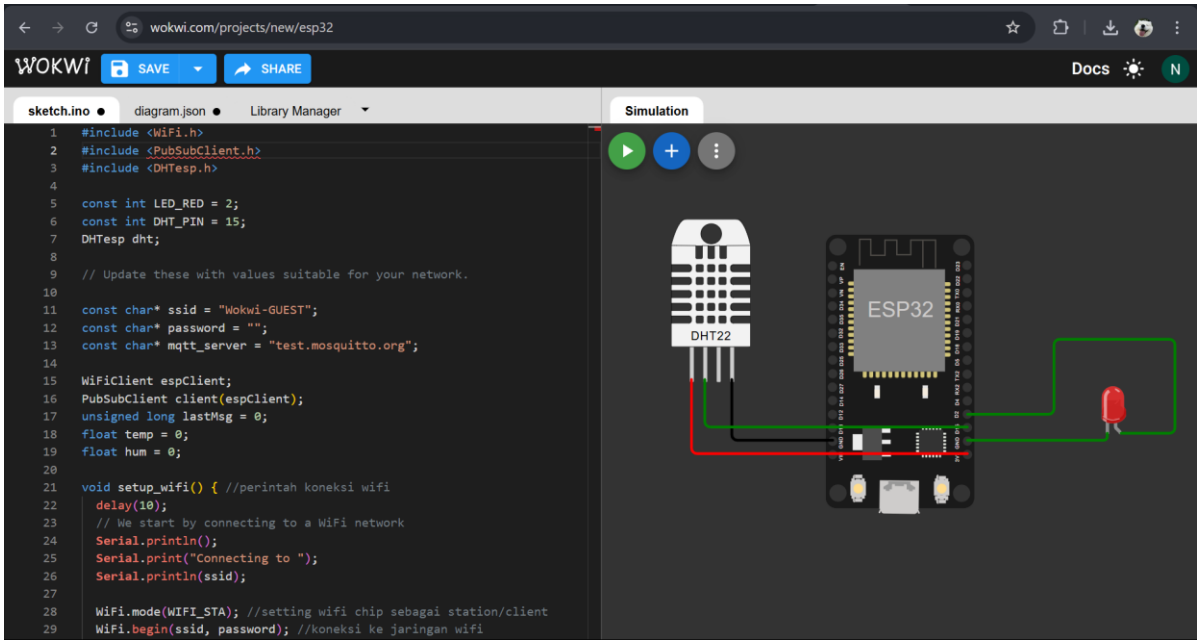
1. Buka platform Wokwi dan buat proyek baru.
2. Tambahkan komponen ESP32, DHT22 dan Lampu berwarna merah.
3. Hubungkan semua komponen ke pin esp sesuai kebutuhan.
4. Tulis kode program dalam bahasa Arduino pada wokwi agar data dapat muncul dan jalannya program
5. Setelah itu download MQTT Explorer.
6. Masuk ke dalam MQTT yang telah di download dan gunakan protocol mqtt:// dengan hots test.mosquitto.org dan port 1883.
7. Lalu connect kan, setelah masuk publish topik dengan nama IOT/Test1.
8. Selanjutnya cari topik yang telah di publish dan nanti data akan masuk ke dalam MQTT.
9. Setelah itu download github yang telah diberikan
10. Buka menggunakan VS Code lalu jalan kan codingan untuk website nya

3. Hasil dan Pembahasan

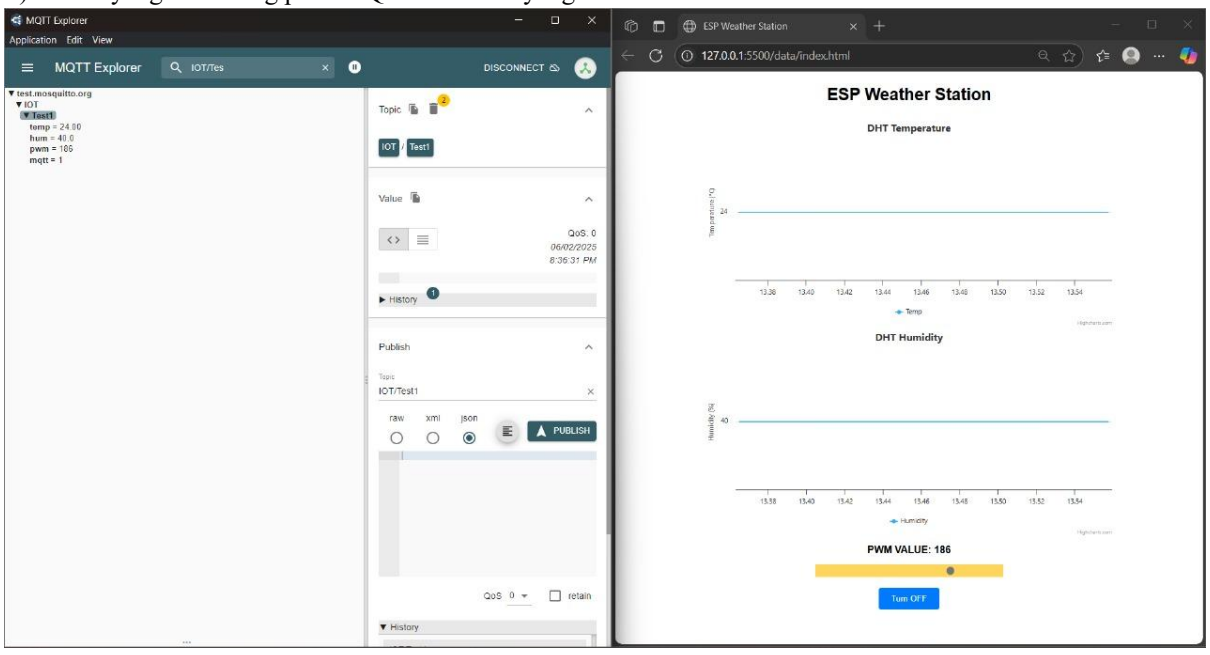
3.1 Hasil Eksperimen

Dalam praktikum kali ini hasil yang didapat sebagai berikut :

1) Pada wokwi



2) Data yang terhubung pada MQTT dan Web yang telah dibuat



URL untuk website yang telah dibuat : http://127.0.0.1:5500/data/index.html

4. Lampiran

Kode Program Pada Wokwi

```
#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";

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 Diagram 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": {}
```

```
}
```

Kode Index untuk Website

```
<!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>