**Laporan Praktikum IoT: Integrasi Wokwi, MQTT, dan Website**

*Difa Aqilah*

*Fakultas Vokasi, Universitas Brawijaya*

*Email :* [*difaaqilah04@student.ub.ac.id*](mailto:difaaqilah04@student.ub.ac.id)

**Abstrak**

Pada praktikum kali ini, kami mencoba membangun sistem monitoring suhu dan kelembapan secara real-time dengan memanfaatkan sensor DHT22 yang terhubung ke mikrokontroler ESP32. Data dari sensor tersebut dikirim ke broker MQTT melalui koneksi internet dan dapat dipantau lewat aplikasi MQTT Explorer.

Sistem ini juga dilengkapi dengan lampu LED merah sebagai penanda jika suhu melewati batas tertentu—lampu akan menyala otomatis. Proyek ini dibangun menggunakan bahasa pemrograman Arduino di PlatformIO, serta diuji coba terlebih dahulu di platform simulasi Wokwi sebelum diterapkan pada perangkat asli. Harapannya, kegiatan ini dapat membantu mahasiswa memahami cara kerja komunikasi data IoT melalui protokol MQTT dan cara membuat sistem pemantauan otomatis.

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

**PENDAHULUAN**

IoT, atau Internet of Things, adalah teknologi yang memungkinkan berbagai perangkat elektronik saling terkoneksi dan bertukar data tanpa campur tangan manusia. Dalam praktiknya, perangkat seperti sensor dan mikrokontroler bisa bekerja otomatis untuk membaca kondisi lingkungan, memproses data, dan mengirimkannya ke sistem pusat. Pada praktikum ini, kami melakukan simulasi monitoring suhu dan kelembapan menggunakan sensor DHT22 yang terhubung dengan ESP32. Data yang terbaca akan dikirim ke broker MQTT dan dapat dipantau lewat MQTT Explorer. Sebagai indikator, LED merah akan menyala ketika suhu atau kelembapan terlalu tinggi. Di masa sekarang, sistem yang mampu membaca dan merespons kondisi lingkungan secara otomatis sangat dibutuhkan, baik di bidang pertanian, industri, maupun manajemen ruangan. Karena itu, sistem pemantauan berbasis IoT menjadi solusi efektif. Melalui praktikum ini, kami menyimulasikan sistem tersebut menggunakan Wokwi, yang menggabungkan sensor DHT22, ESP32, LED, dan protokol MQTT. Data dari sensor dikirim lewat WiFi ke broker MQTT, lalu ditampilkan ke website. Dari kegiatan ini, mahasiswa bisa memahami alur data dari sensor hingga tampil di web.

**TUJUAN**

Beberapa hal yang ingin dicapai dalam praktikum ini:

1. Mempelajari cara menghubungkan Wokwi/VSCode dengan MQTT.
2. Memahami cara mengambil data dari Wokwi menggunakan MQTT.
3. Mengirim data dari MQTT ke website.
4. Menampilkan data suhu dan kelembapan di halaman website.

**METODELOGI**

Alat dan Bahan

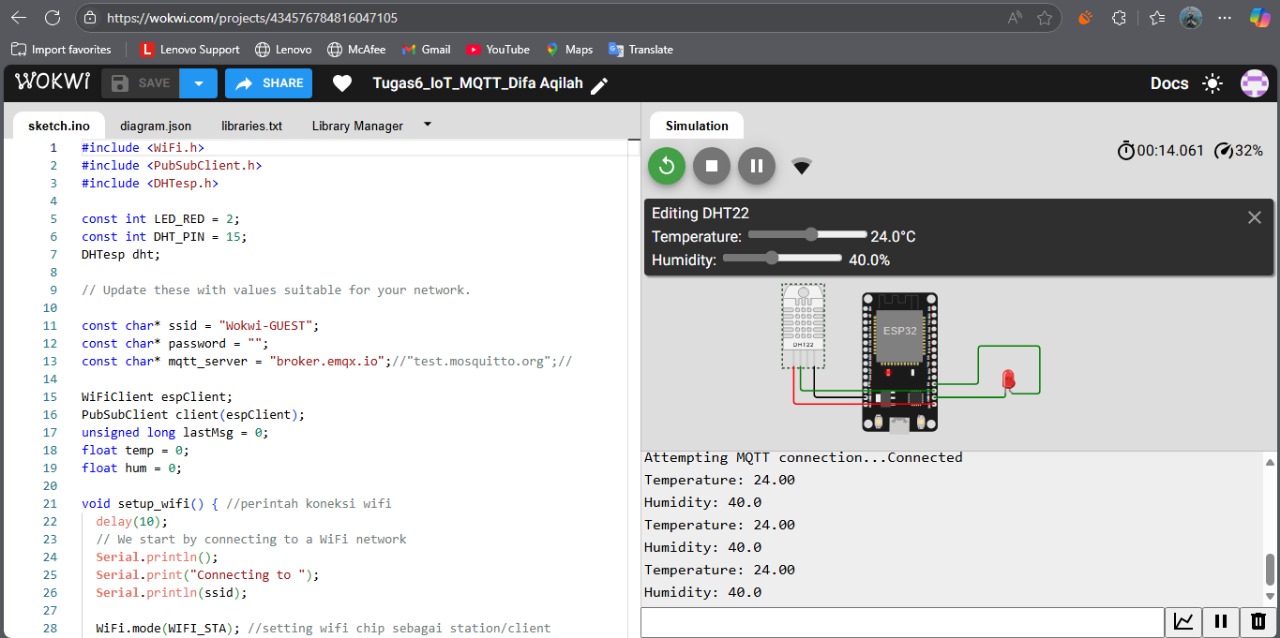
* ESP32
* Wokwi (simulasi)
* Sensor DHT22
* LED merah
* MQTT Explorer
* Kabel jumper
* VSCode dengan PlatformIO

Langkah-langkah

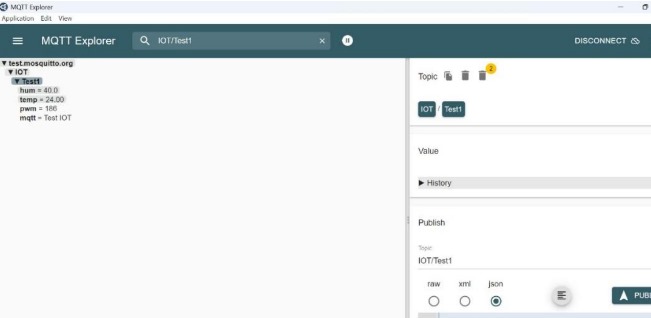
1. Membuat proyek baru di Wokwi.
2. Menambahkan komponen ESP32, DHT22, dan LED merah.
3. Menghubungkan semua komponen ke pin yang sesuai.
4. Menulis kode Arduino untuk membaca sensor dan mengatur LED.
5. Mengunduh dan mengatur aplikasi MQTT Explorer.
6. Menghubungkan MQTT ke server publik (test.mosquitto.org, port 1883).
7. Mem-publish topik data ke “IOT/Test1”.
8. Memantau data dari topik tersebut di MQTT.
9. Mengunduh file website dari GitHub.
10. Menjalankan file website melalui VSCode untuk menampilkan data sensor.

**HASIL DAN PEMBAHASAN**

Hasil eksperimen pada wokwi:



Hasil tampilan MTQQ:



**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>