**Implementasi Sistem Pemantauan Suhu dan Kelembapan Real-Time Berbasis IoT Menggunakan Protokol MQTT**

*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 utama dan MQTT sebagai protokol pengiriman datanya. Proyek ini bertujuan untuk menerapkan teknologi IoT dalam memantau kondisi lingkungan secara *real-time* tanpa harus menggunakan perangkat fisik terlebih dahulu.

Eksperimen dimulai dengan merancang rangkaian virtual di dalam Wokwi, yang mengintegrasikan mikrokontroler ESP32 dengan sensor DHT22 dan sebuah LED sebagai indikator. Kode program kemudian dibuat menggunakan bahasa Arduino untuk membaca data sensor dan mengirimkannya ke *broker* MQTT melalui koneksi internet yang disimulasikan oleh Wokwi. Selain itu, sistem diprogram agar LED merah menyala secara otomatis ketika suhu melewati ambang batas tertentu.

Hasil utama dari praktikum ini menunjukkan bahwa integrasi antara simulasi di Wokwi dengan layanan *broker* MQTT dapat berjalan dengan sukses. Data sensor terbukti dapat dikirim dan dipantau secara *real-time* melalui aplikasi MQTT Explorer. Simulasi ini berhasil membuktikan bahwa Wokwi merupakan platform yang sangat efektif untuk menguji keseluruhan fungsionalitas sistem IoT, terutama dalam hal komunikasi data, sebelum diimplementasikan pada perangkat keras yang sebenarnya.

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

# Abstract:

This practical report discusses the creation of an Internet of Things (IoT)-based temperature and humidity monitoring system by utilizing Wokwi as the main simulation platform and MQTT as the data transmission protocol. This project aims to apply IoT technology in monitoring environmental conditions in real-time without having to use physical devices first.

The experiment starts by designing a virtual circuit inside Wokwi, which integrates an ESP32 microcontroller with a DHT22 sensor and an LED as an indicator. Program code was then created using the Arduino language to read the sensor data and send it to an MQTT broker via an internet connection simulated by Wokwi. In addition, the system is programmed to have a red LED turn on automatically when the temperature crosses a certain threshold.

The main result of this practicum shows that the integration between the simulation in Wokwi and the MQTT broker service can run successfully. Sensor data is proven to be sent and monitored in real-time through the MQTT Explorer application. This simulation successfully proves that Wokwi is a very effective platform for testing the overall functionality of IoT systems, especially in terms of data communication, before being implemented on actual hardware.

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

# 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, sehingga tindakan preventif dapat diambil dengan cepat bahkan dari jarak jauh. Pendekatan ini menawarkan akurasi dan efisiensi yang jauh lebih baik daripada metode konvensional.

Oleh karena itu, praktikum yang menyimulasikan sistem pemantauan berbasis IoT menjadi penting sebagai langkah awal untuk memahami implementasinya. Dalam proyek ini, protokol MQTT digunakan sebagai jalur komunikasi untuk mengirim data dari sensor DHT22. Sementara itu, Wokwi berperan sebagai platform simulasi utama yang memungkinkan seluruh rangkaian—mulai dari pembacaan sensor oleh ESP32 hingga pengiriman data ke *broker* MQTT—diuji sepenuhnya secara virtual tanpa memerlukan perangkat keras fisik.

* 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 berbasis IoT secara *real-time* yang mampu mengirimkan data dari sensor DHT22 ke *broker* MQTT menggunakan mikrokontroler ESP32.
* Menggunakan Wokwi sebagai platform simulasi untuk memvalidasi dan menguji fungsionalitas keseluruhan sistem—mulai dari pembacaan sensor, logika aktuator (LED), hingga konektivitas ke *broker* MQTT—tanpa memerlukan perangkat keras fisik.
* Memahami prinsip dasar dan mekanisme komunikasi data pada sistem IoT, khususnya arsitektur *publish-subscribe* pada protokol MQTT untuk pengiriman dan pemantauan data sensor dari jarak jauh.
* Mengembangkan kemampuan pemrograman mikrokontroler ESP32 dengan bahasa Arduino serta mengeksplorasi cara mengintegrasikan komponen sensor dan aktuator dengan protokol komunikasi IoT dalam sebuah lingkungan virtual.
* Menyediakan dasar untuk pengembangan proyek IoT yang lebih kompleks, serta membuktikan efektivitas simulasi sebagai tahap awal (*prototyping*) sebelum melakukan implementasi pada perangkat keras sesungguhnya.

# 2. Metodologi

* 1. **Alat dan Bahan**

Untuk melakukan praktikum simulasi, alat dan bahan yang digunakan

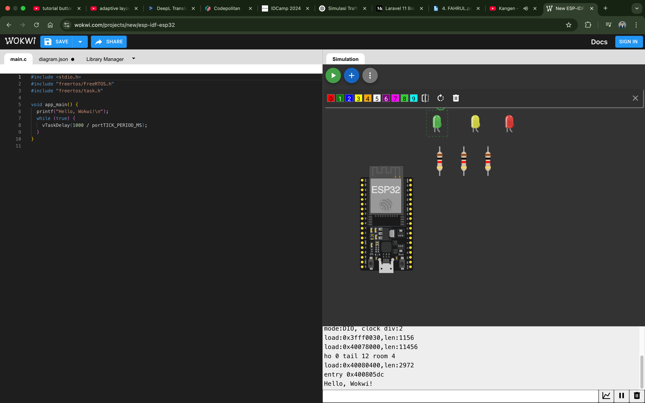
1. Mikrokontroler ESP32
2. Platform wokwi, digunakan untuk membuat simulasi
3. LED Merah
4. DHT 22
5. MQTT Explorer
6. Kabel Jumper
7. Arduino IDE/PlatformIO jika menggunakan Visual Studio Code
   1. **Langkah Implementasi**

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);

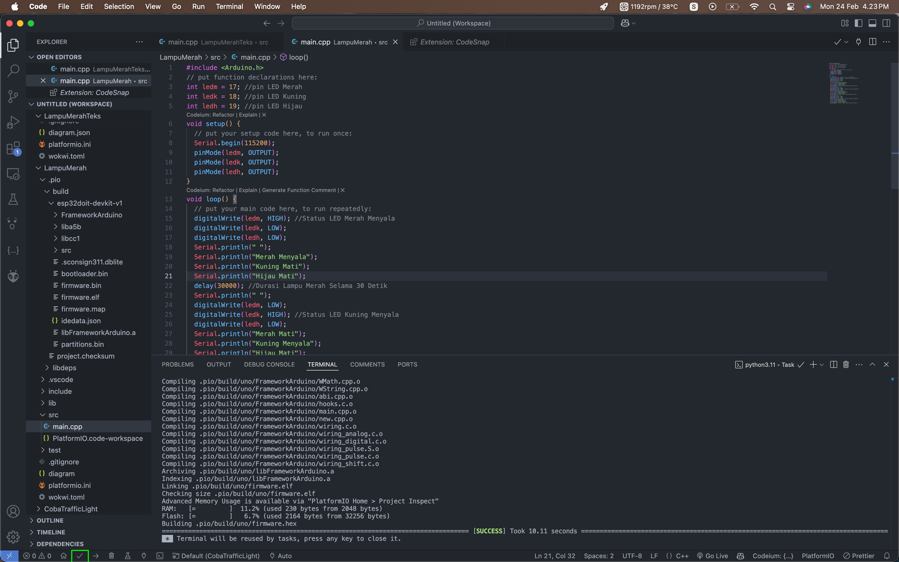
**Serial**.print("Humidity: ");

**Serial**.println(hum);

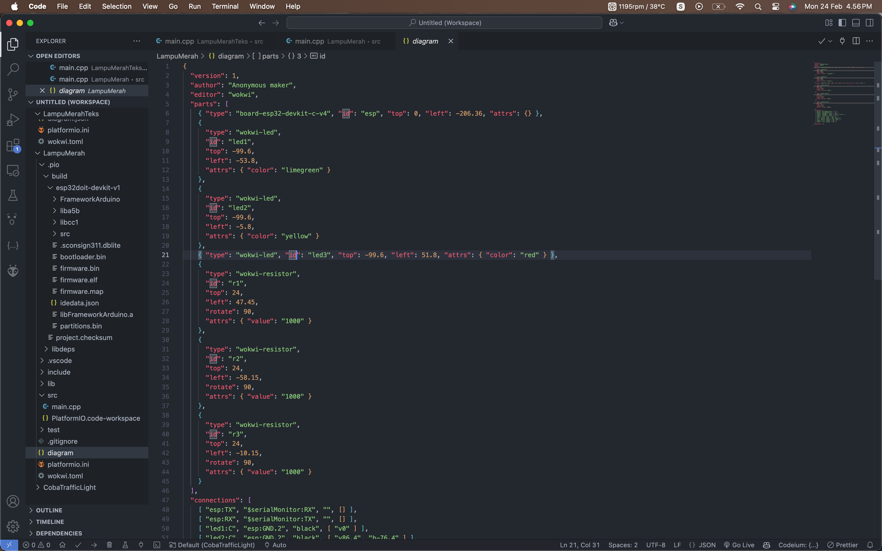
  }

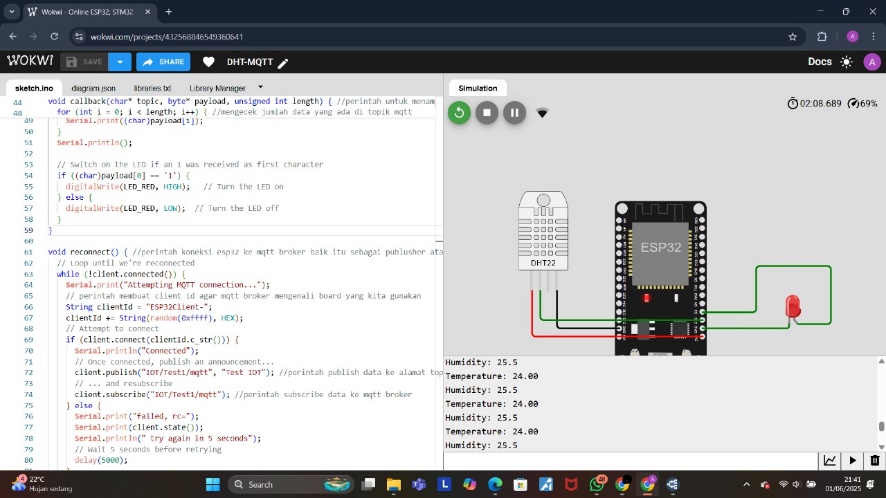
}

* 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.

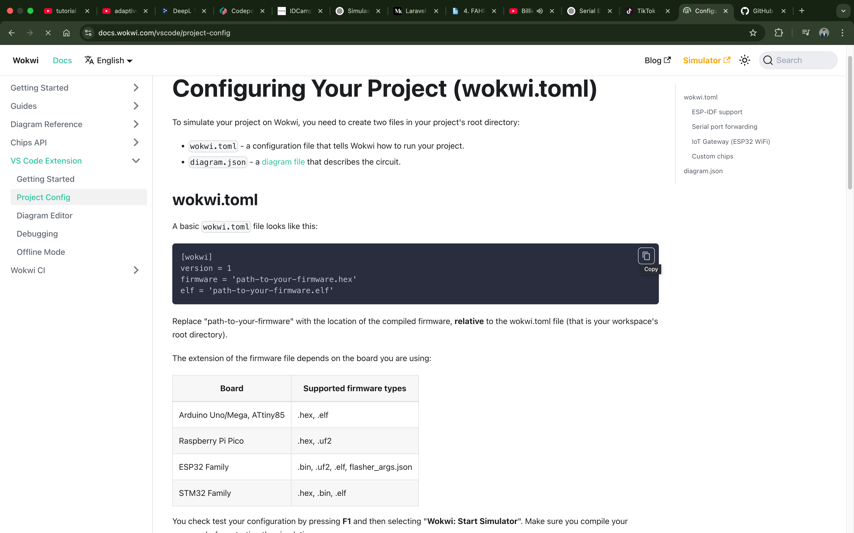


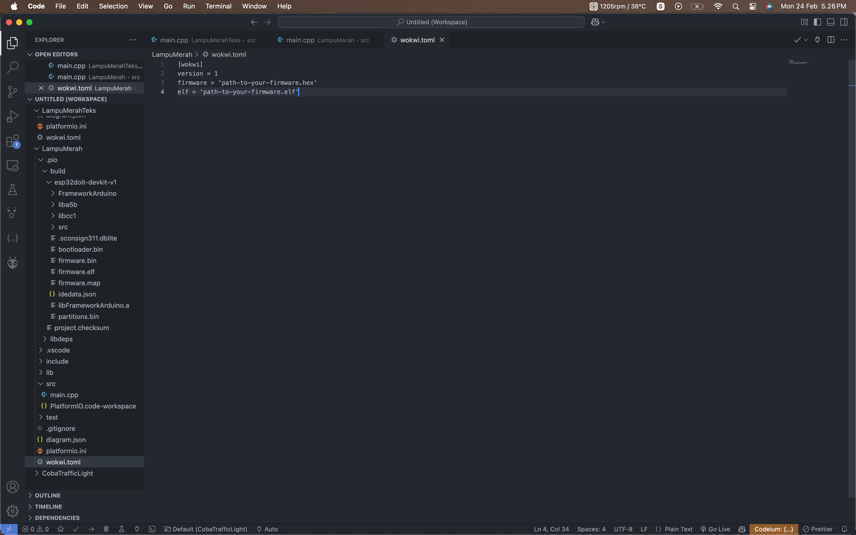
* Jika kode di file ‘main.cpp’ sukses di *Build* maka langkah selanjutnya untuk dapat melakukan virtualisasi simulasi lampu merah adalah membuat file bernama ‘diagram’ yang nanti diisi oleh kode dari file ‘diagram.json’ di projek Wokwi sebelumnya. Lalu kalau sudah, ubah nama file ‘diagram’ sbelumnya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



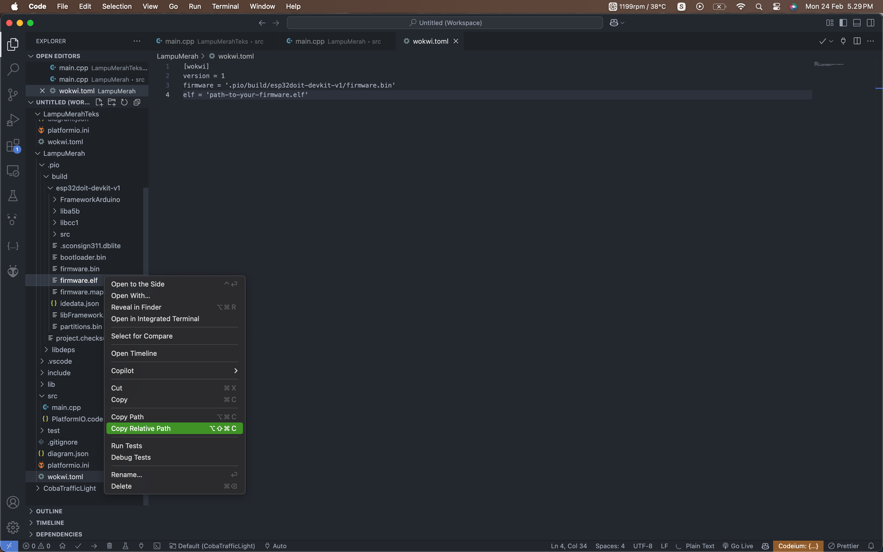


* Untuk dapat menjalankan simulasi dari file ‘diagram.json’ pada platform.io tersebut maka kita harus menambahkan file yang kita namakan ‘wokwi.toml’. Langkah berikutnya cari di web browser dengan keyword ‘Wokwi.toml’ dan salin kode konfigurasinya yang berasal dari halaman web dokumentasi wokwi tersebut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

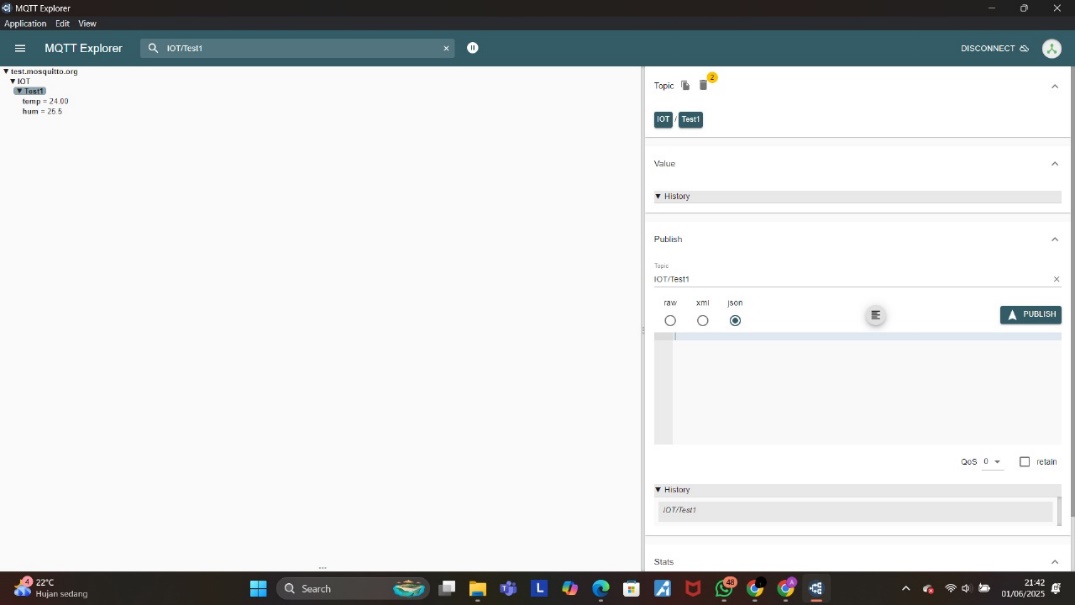




* Langkah terakhir untuk dapat mengkonfigurasi file ‘wokwi.toml’ tersebut agar dapat membantu menjalankan file diagramnya maka kita perlu untuk mengganti path ‘firmware’ dan ‘elf’ path file yang berada di dalam rute folder ‘pio/build/esp32doit-devkit-v1’. Salin tiap file dengan format file yang bertuliskan ‘.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.

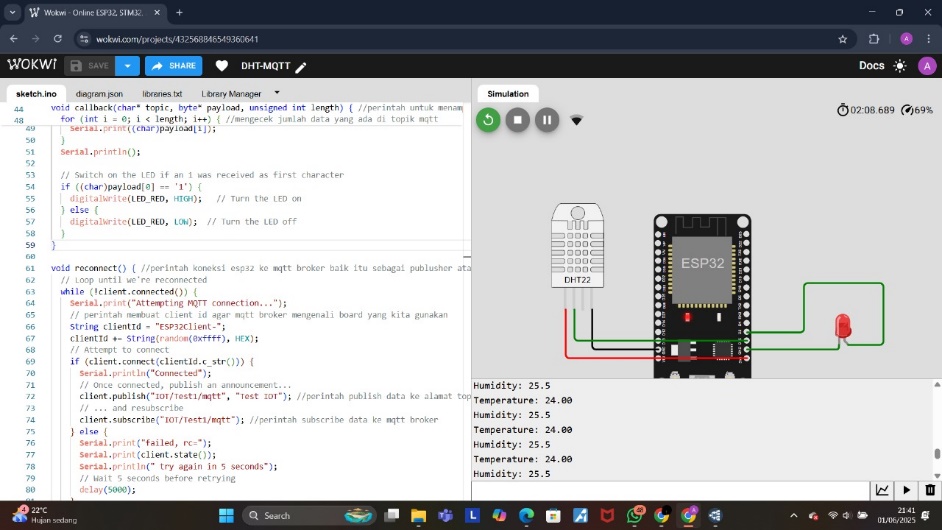


# 3. Hasil dan Pembahasan

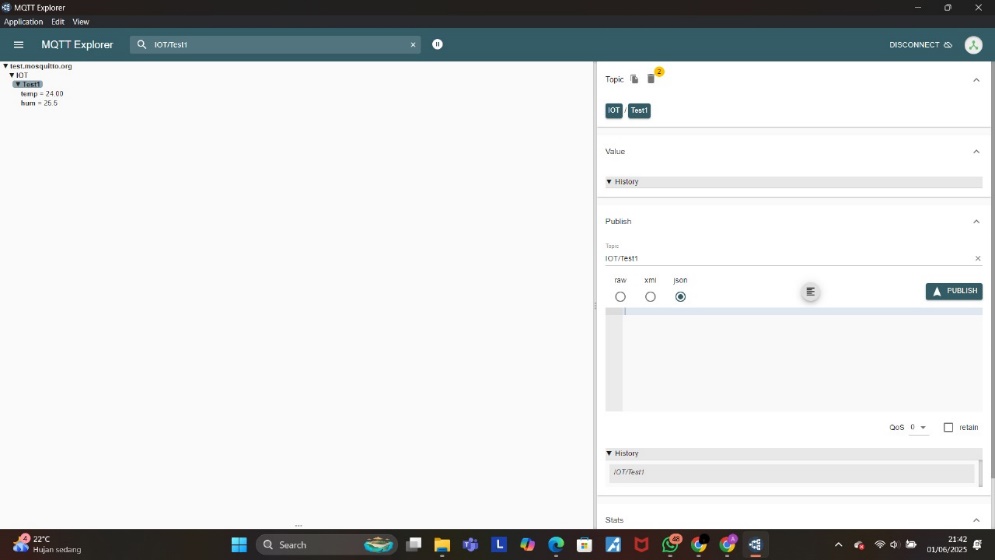
1. **Hasil Eksperimen**

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": {}

}