PRAKTIKUM SIMULASI KOMUNIKASI MQTT MENGGUNAKAN ESP32 DAN PLATFORM WOKWI

TB Rangga Gilang Yanuari
Fakultas Vokasi, Universitas Brawijaya
gilangyanuarirangga@gmail.com

ABSTRAK

Pada praktikum ini dilakukan simulasi komunikasi data antara ESP32 dengan MQTT broker menggunakan protokol publish-subscribe. Platform simulasi Wokwi digunakan untuk menghubungkan ESP32 dengan sensor DHT22 dan LED, di mana data suhu dan kelembapan dipublikasikan ke MQTT broker, dan kontrol LED dapat dilakukan melalui dashboard MQTT seperti MQTT Explorer. Hasil menunjukkan bahwa ESP32 dapat mengirim dan menerima data melalui protokol MQTT dengan lancar, memungkinkan integrasi dengan berbagai aplikasi IoT secara real-time.

Kata kunci: MQTT, ESP32, IoT, Wokwi, Sensor DHT22

ABSTRACT

This practicum simulates data communication between an ESP32 microcontroller and an MQTT broker using the publish-subscribe protocol. The Wokwi platform is used to simulate an ESP32 connected to a DHT22 sensor and an LED, where temperature and humidity data are published to the MQTT broker, and the LED can be controlled via an MQTT dashboard such as MQTT Explorer. The results show that the ESP32 can transmit and receive data using the MQTT protocol effectively, enabling real-time integration with various IoT applications.

Keywords: MQTT, ESP32, IoT, Wokwi, DHT22 Sensor

1. PENDAHULUAN

1.1 Latar Belakang

Internet of Things (IoT) semakin berkembang dalam sistem otomasi, salah satunya melalui protokol MQTT (Message Queuing Telemetry Transport) yang ringan dan efisien. MQTT memanfaatkan arsitektur publish-subscribe untuk pertukaran data antar perangkat. ESP32, dengan dukungan koneksi WiFi, dapat diintegrasikan dengan broker MQTT untuk mengirim (publish) dan menerima (subscribe) data sensor. Dalam praktikum ini, ESP32 disimulasikan di Wokwi, dengan sensor DHT22 untuk pembacaan suhu dan kelembapan serta LED sebagai aktuator yang dikontrol melalui pesan MQTT.

1.2 Tujuan Praktikum

Praktikum ini bertujuan untuk:

1) Memahami konsep komunikasi data MQTT dalam IoT
2) Mengimplementasikan koneksi ESP32 ke broker MQTT

3) Mempublikasikan data sensor DHT22 ke broker
4) Mengontrol LED melalui perintah MQTT dari dashboard

2. METODOLOGI

2.1 Alat dan Bahan

1. **Alat:**
    a) Komputer/laptop sebagai perangkat utama untuk menjalankan simulasi.
    b) Platform Wokwi sebagai simulator rangkaian tanpa perangkat fisik.
    c) Aplikasi MQTT Explorer untuk memonitor dan mengirim data MQTT
    d) Visual Studio Code (VSCode) dengan ekstensi PlatformIO sebagai lingkungan pengembangan.
2. **Bahan:**
    a) ESP32 sebagai mikrokontroler utama (disimulasikan).
    b) Sensor DHT22 untuk membaca suhu dan kelembapan.
    c) LED untuk indikator output.
    d) Jaringan internet untuk koneksi ke broker MQTT.
    e) Broker MQTT publik: test.mosquitto.org, port 1883.
    f) Bahasa pemrograman C++ (Arduino).

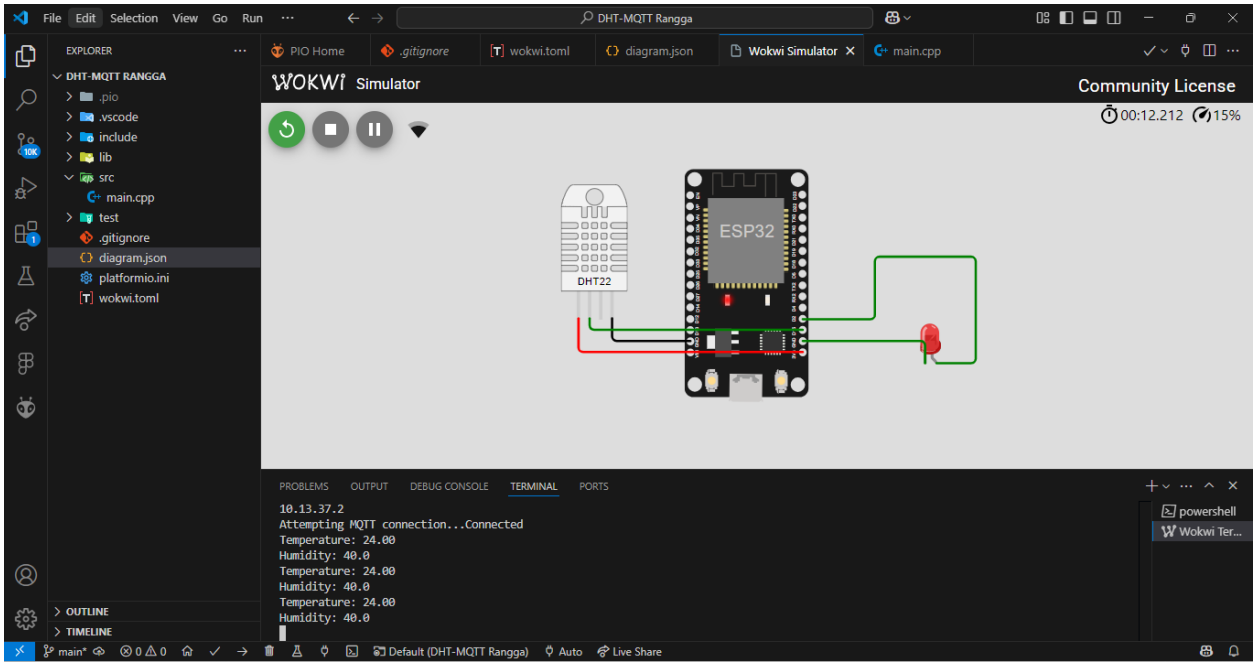
2.2 Langkah Implementasi

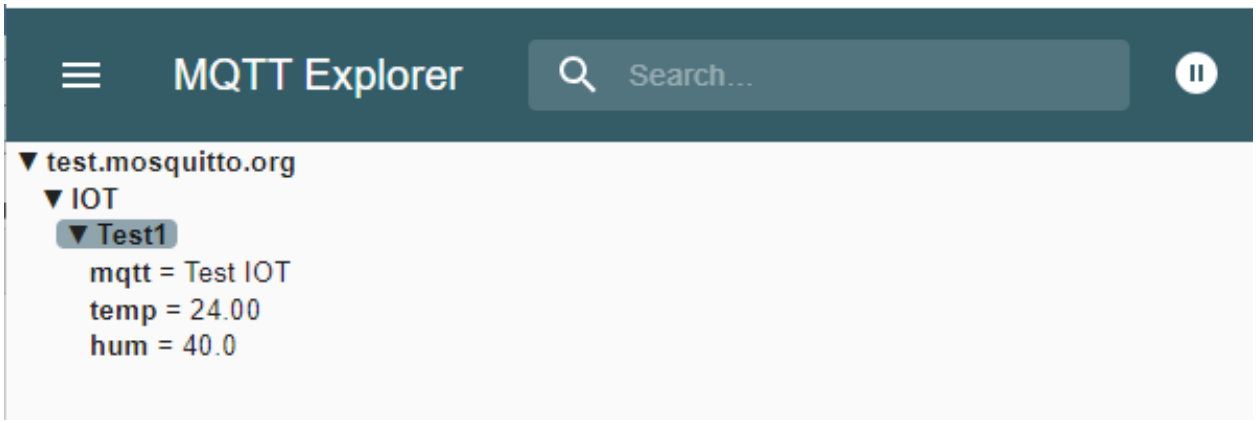
1. **Membuka Wokwi dan Membuat Proyek Baru**
    1) Tambahkan ESP32, sensor DHT22, dan LED.
    2) Hubungkan sesuai pin GPIO:
        3. LED di pin 2
        4. DHT22 di pin 15
2. **Menulis Program ESP32**
    1) Menggunakan library WiFi.h, PubSubClient.h, dan DHTesp.h
    2) Menghubungkan ESP32 ke WiFi Wokwi
    3) Mengatur koneksi ke broker MQTT
    4) Publish data suhu dan kelembapan ke topic MQTT setiap 2 detik
    5) Subscribe ke topic untuk menerima perintah menyalakan/mematikan LED
3. **Menjalankan Simulasi di Wokwi**
    1) Jalankan proyek, buka Serial Monitor
    2) Pastikan koneksi WiFi dan MQTT berhasil
    3) Amati pesan-pesan data suhu, kelembapan, dan perintah LED
4. **Menguji dengan MQTT Explorer**
    1) Buka MQTT Explorer
    2) Tambahkan koneksi ke test.mosquitto.org:1883

3) Subscribe ke topik:
    - IOT/Test1/temp
    - IOT/Test1/hum
1) Publish ke topik IOT/Test1/mqtt dengan payload:
    - 1 → LED menyala
    - 0 → LED mati

3. HASIL DAN PEMBAHASAN

3.1 Hasil





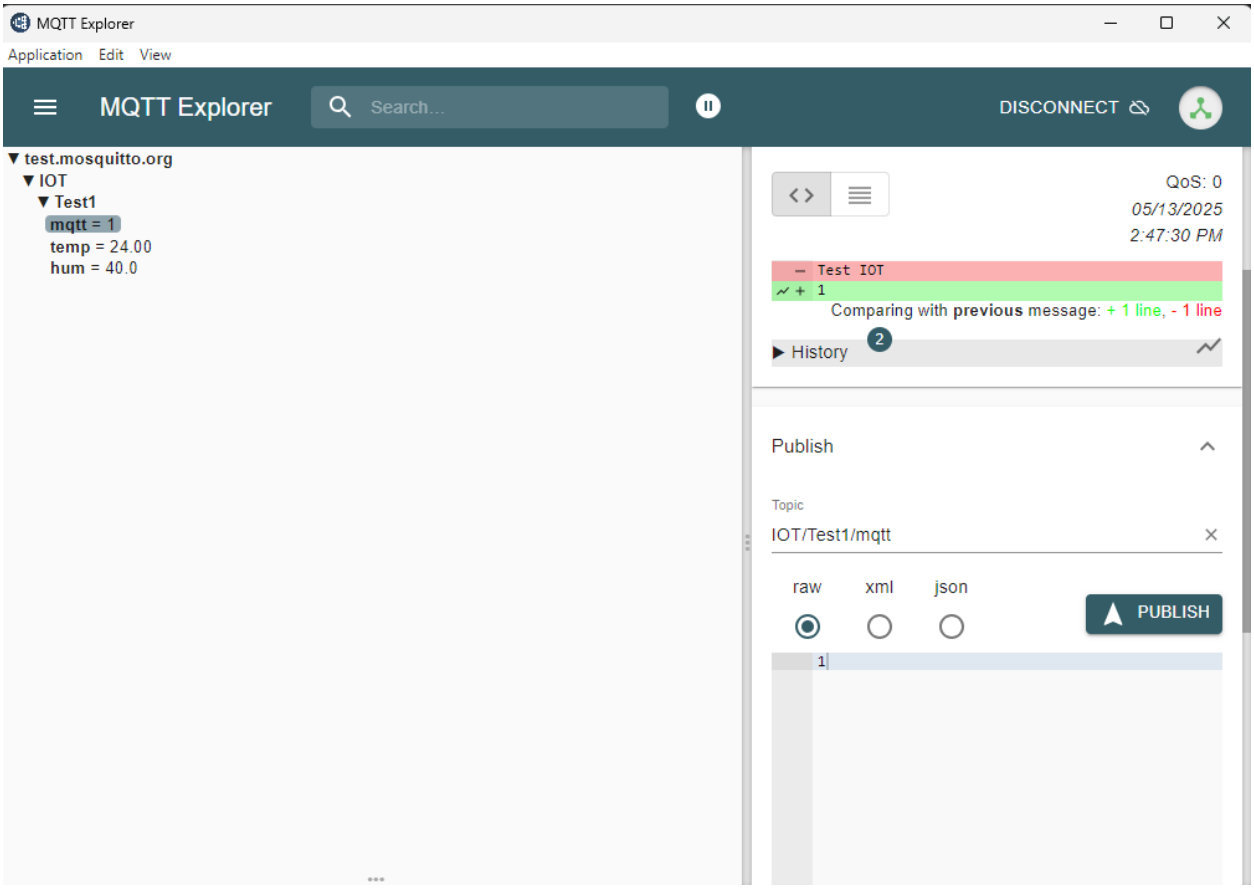
ESP32 berhasil:

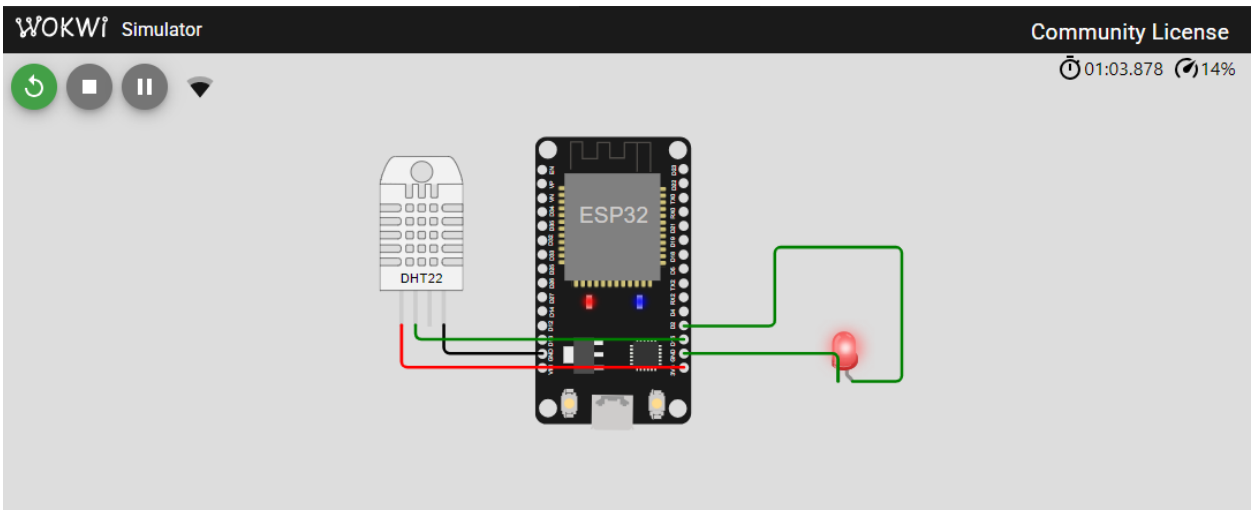
- Terkoneksi ke WiFi dan broker MQTT
- Mengirim data suhu & kelembapan ke topik MQTT

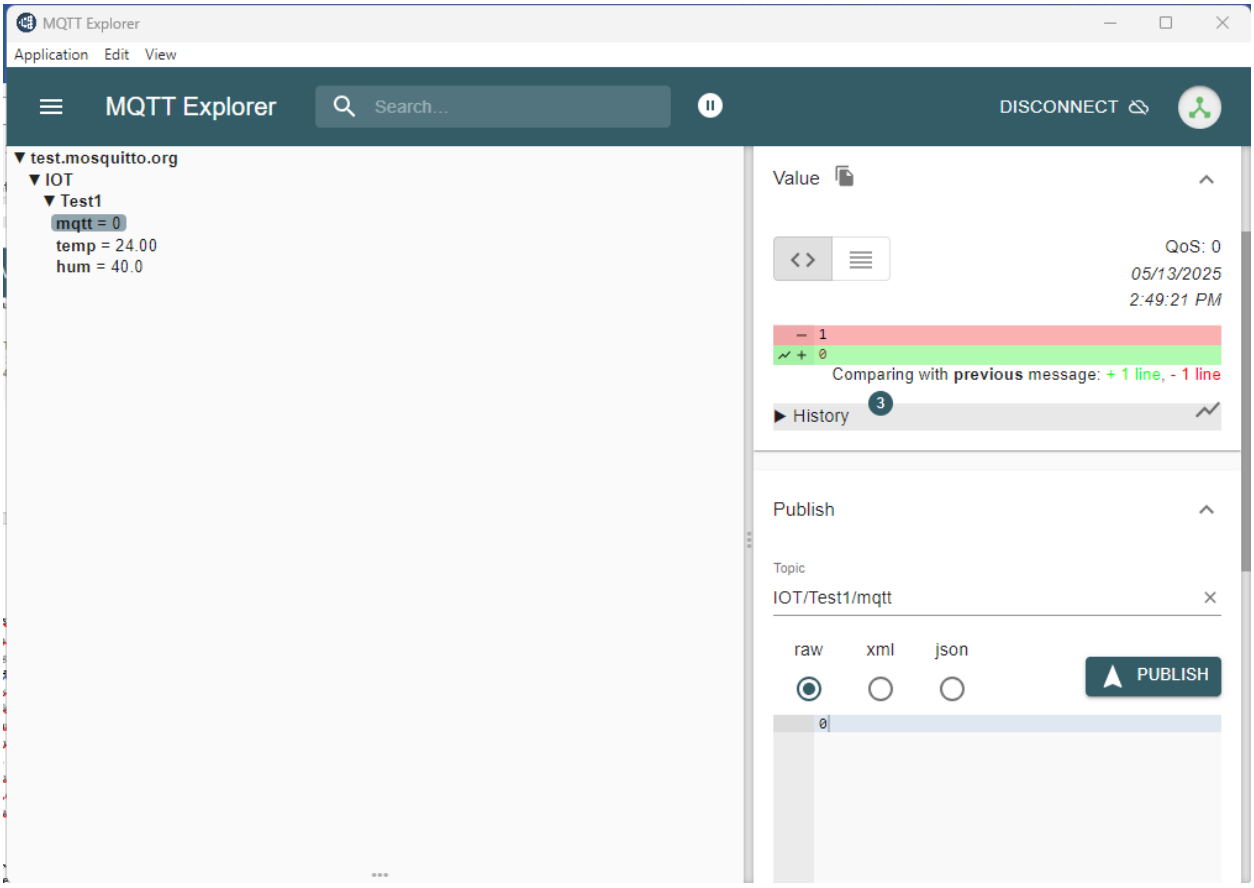
- Menerima perintah dari MQTT untuk mengendalikan LED

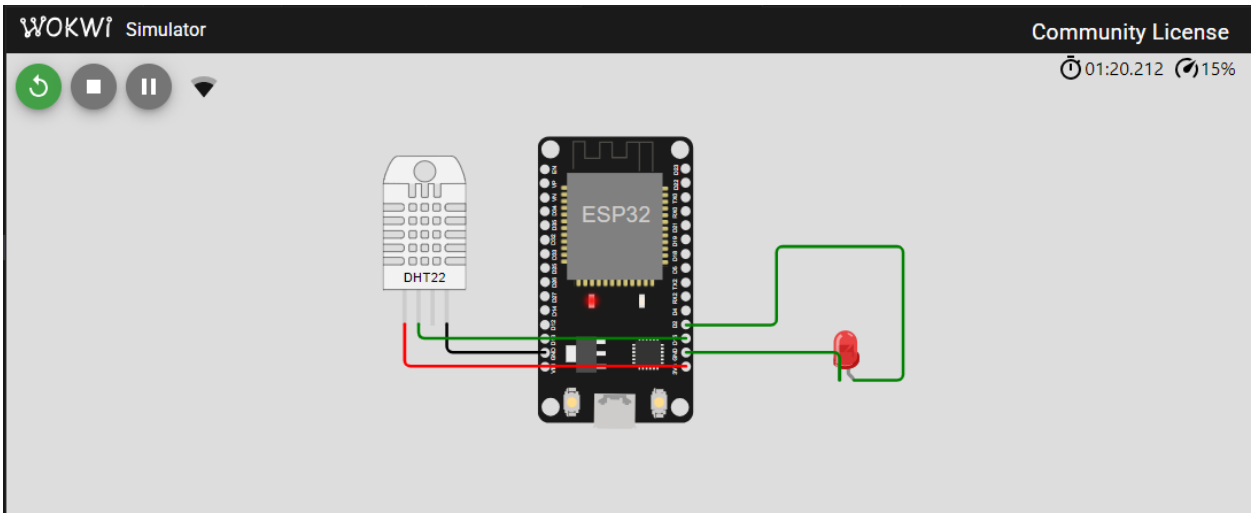
Data yang dikirim:

- IOT/Test1/temp: 24.00 (°C)
- IOT/Test1/hum: 40.0 (%)









LED merespons perintah dari MQTT dengan benar. Saat nilai dipublish dari MQTT Explorer ke topik IOT/Test1/mqtt:

- Dikirim nilai 1 → LED menyala
- Dikirim nilai 0 → LED mati

Pengiriman nilai dilakukan secara manual melalui tampilan MQTT Explorer, menunjukkan komunikasi berhasil antara client MQTT dan ESP32 di platform Wokwi.

3.2 Pembahasan

MQTT terbukti efisien dalam komunikasi antar perangkat IoT karena sifatnya ringan dan real-time. Dalam simulasi ini, data berhasil dikirim dan diterima melalui broker publik tanpa keterlambatan berarti. Sistem ini dapat dikembangkan lebih lanjut ke dashboard visual atau disimpan ke database cloud.

2. LAMPIRAN

4.1 Kode program

```
#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 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);
  }
}
```

4.2 Kode diagram.json

```
{
  "version": 1,
  "author": "Rangga",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -24.1, "left": -5,
"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": {}
}
```

4.3 MQTT EXPLORE