MONITORING LINGKUNGAN BERBASIS MQTT DENGAN ESP32 DAN DHT22

Feby Kurnia Putri

Teknologi Informasi, Fakultas Vokasi, Universitas Brawijaya Jl. Veteran No.10-11, Ketawanggede, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145

febystudentub@gmail.com

Abstrack -- This practicum implements a weather monitoring system based on the Internet of Things (IoT) using the ESP32 microcontroller and the MQTT protocol for real-time data communication. The ESP32 connects to a Wi-Fi network and establishes a connection to a public MQTT broker (test.mosquitto.org) to publish and subscribe to specific topics. Temperature and humidity data measured by the DHT22 sensor are published periodically every two seconds to designated MQTT topics. The system also receives commands from the broker to control an LED indicator using the MQTT callback function. The use of the MQTT protocol enables lightweight, reliable, and efficient message exchange, making it well-suited for IoT applications. Testing is conducted using the Wokwi simulation platform, demonstrating stable connectivity, real-time sensor data transmission, and responsive device control via MQTT messaging.

Keyword : ***Internet of Things, ESP32, MQTT, DHT22, temperature sensor, real-time , MQTT broker.***

Abstrak -- Praktikum ini mengimplementasikan sistem monitoring cuaca berbasis Internet of Things (IoT) menggunakan mikrokontroler ESP32 dan protokol MQTT untuk komunikasi data secara real-time. ESP32 terhubung ke jaringan Wi-Fi dan melakukan koneksi ke broker MQTT publik (test.mosquitto.org) untuk mengirim dan menerima pesan pada topik tertentu. Data suhu dan kelembaban yang diukur menggunakan sensor DHT22 dipublikasikan secara berkala setiap dua detik ke topik MQTT yang telah ditentukan. Sistem juga dapat menerima perintah dari broker untuk mengendalikan LED sebagai indikator dengan menggunakan fungsi callback MQTT. Penggunaan protokol MQTT memungkinkan pertukaran pesan yang ringan, andal, dan efisien, sangat cocok untuk aplikasi IoT. Pengujian dilakukan dengan menggunakan platform simulasi Wokwi, yang menunjukkan kestabilan koneksi, pengiriman data sensor secara real-time, serta kontrol perangkat yang responsif melalui pesan MQTT.

Kata Kunci : ***Internet of Things, ESP32, MQTT, DHT22*****, sensor suhu, *****real-time*****, broker MQTT.***

I. PENDAHULUAN

Perkembangan teknologi Internet of Things (IoT) telah membuka peluang besar untuk pengembangan sistem monitoring dan kontrol perangkat secara jarak jauh dengan koneksi internet. Salah satu protokol komunikasi yang banyak digunakan dalam aplikasi IoT adalah MQTT (Message Queuing Telemetry Transport), yang dirancang untuk pengiriman data ringan, efisien, dan andal pada jaringan yang memiliki bandwidth terbatas atau tidak stabil.

ESP32 merupakan mikrokontroler yang populer untuk aplikasi IoT karena memiliki kemampuan Wi-Fi terintegrasi dan performa yang baik dengan konsumsi daya rendah. Dengan menggabungkan ESP32 dan sensor DHT22, sistem dapat memonitor kondisi lingkungan seperti suhu dan kelembaban secara real-time. Data tersebut kemudian dikirim ke broker MQTT untuk diakses atau dipantau oleh pengguna atau aplikasi lain secara remote.

Pada praktikum ini, implementasi sistem menggunakan ESP32 yang terkoneksi ke jaringan Wi-Fi dan broker MQTT publik untuk mengirim data sensor suhu dan kelembaban setiap beberapa detik. Selain itu, sistem juga mampu menerima perintah untuk mengendalikan LED sebagai indikator melalui pesan MQTT, yang meningkatkan interaktivitas dan kontrol jarak jauh. Pengujian dilakukan secara simulasi menggunakan platform Wokwi untuk memvalidasi fungsi komunikasi MQTT dan pengambilan data sensor secara real-time.

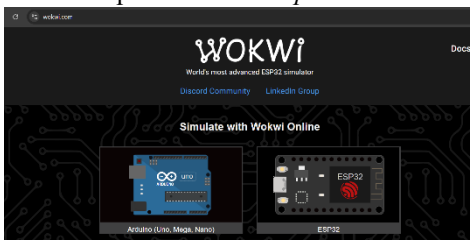
II. METODOLOGI

A. ALAT & BAHAN

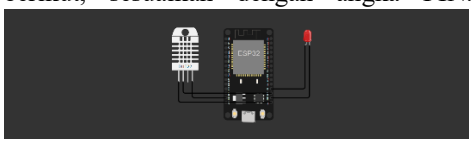
Mikrokontroler *(ESP32),* Sensor DHT22, *LED merah, Kabel jumper, Broker MQTT publik , OpenWeatherMap, Visual Studio Code* (C++*), PlatformIO IDE. Koneksi Wi-Fi (misal: jaringan Wokwi-GUEST), Komputer dengan software Arduino IDE.*

B. LANGKAH IMPLEMENTASI

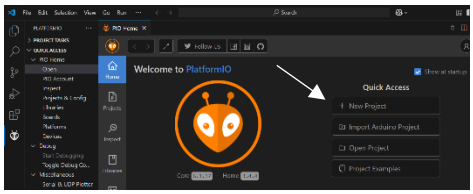
1. Buka https://wokwi.com/ pilih ESP32, kemudian pilih *Starter Templates.*



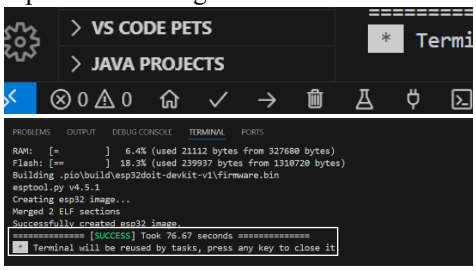
2. Atur rangkain diagram seperti gambar berikut, sesuaikan dengan angka PIN.



3. Buka VScode, kemudian *install PlatformIO IDE.* Buat proyek baru seperti gambar berikut:



4. *Build Wokwi* dengan cara klik centang pada bagian bawah dari *Visual Studio Code,* seperti gambar berikut.



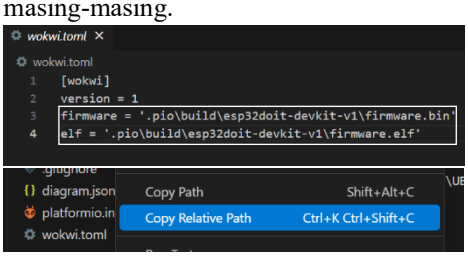
5. Buat *file* baru pada *Visual Studio Code*, kemudian buka *wokwi.toml* pada *google*, kemudian *copy* ke *file* baru yang diberi nama *wokwi.toml* di *VScode.*

wokwi.toml

A basic `wokwi.toml` file looks like this:

```
[wokwi]
version = 1
firmware = 'path-to-your-firmware.hex'
elf = 'path-to-your-firmware.elf'
```

6. Setelah itu ganti menjadi *firmware bin*, dan *esp32 firmware elf* pada *Visual Studio Code* masing-masing.



7. Buat *file* baru bernama *diagram.json* pada *Visual Studio Code.* Kemudian pergi ke *wokwi* untuk meng *copy code diagram.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": 0.3, "left": -111, "attrs": {} },
   {
    "type": "wokwi-led",
    "id": "led1",
    "top": -3.6,
    "left": 157.8,
    "attrs": { "color": "red", "flip": "1" }
   }
 ],
 "connections": [
   [ "esp:TX0", "$serialMonitor:RX", "", []
],
   [ "esp:RX0", "$serialMonitor:TX", "", []
],
   [ "dht1:GND", "esp:GND.2", "black", [
"v0" ] ],
   [ "dht1:VCC", "esp:3V3", "black", [ "v0"
] ],
   [ "dht1:SDA", "esp:D15", "black", [ "v0"
] ],
   [ "led1:A", "esp:D2", "black", [ "v0" ] ],
   [ "led1:C", "esp:GND.1", "black", [ "v0"
] ]
 ],
 "dependencies": {}
}
```

8. *Paste* ke *Visual Studio Code,* pada *file diagram.json. Jika diagram* tidak berfungsi, maka lakukan edit pada *diagram.json*, kemudian *paste code* yang sudah di *copy* pada *wokwi.*

9. Kemudian *wokwi* akan menampilkan hasil dari *diagram.json* pada simulasi *Temperature dan Humidity.*

10. Buat code C++ pada main

```
#include <Arduino.h>
#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() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    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) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  if ((char)payload[0] == '1') {
    digitalWrite(LED_RED, HIGH);
  } else {
    digitalWrite(LED_RED, LOW);
  }
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");

    String clientId = "ESP32Client-";
    clientId += String(random(0xffff), HEX);

    if (client.connect(clientId.c_str())) {
      Serial.println("Connected");

      client.publish("IOT/Test1/mqtt", "Test IOT");
      client.subscribe("IOT/Test1/mqtt");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");

      delay(5000);
    }
  }
}

void setup() {
  pinMode(LED_RED, OUTPUT);
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  dht.setup(DHT_PIN, DHTesp::DHT22);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  unsigned long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;
    TempAndHumidity data = dht.getTempAndHumidity();

    String temp = String(data.temperature, 2);
```

```
    client.publish("IOT/Test1/temp",
temp.c_str());

    String hum = String(data.humidity, 1);
    client.publish("IOT/Test1/hum",
hum.c_str());

    Serial.print("Temperature: ");
    Serial.println(temp);
    Serial.print("Humidity: ");
    Serial.println(hum);
  }
}
```

III. HASIL DAN PEMBAHASAN

1. *ESP32* dapat terhubung ke jaringan *WiFi* dengan alamat *IP 10.13.37.2* dan juga berhasil melakukan koneksi ke server *MQTT.*

2. Informasi yang diperoleh mencakup *Sensor DHT22* secara konsisten mengirimkan data suhu sebesar 24.00°C dan kelembaban sebesar 40.0%, yang ditampilkan secara berulang di terminal. Hal ini menunjukkan bahwa pembacaan sensor berjalan stabil tanpagangguan.

IV. KESIMPULAN

Setelah melakukan perancangan dan pengujian terhadap alat secara keseluruhan. Maka dapat diambil kesimpulan :

1. Perangkat yang digunakan oleh penulis dapat bekerja dengan baik sesuai dengan yang diharapkan.
2. Mikrokontroler *ESP32* yang digunakan sebagai pengendali utama dapat bekerja dalam menjalankan perintah yang diberikan

V. DAFTAR PUSTAKA

Random Nerd Tutorials. (n.d.). *ESP32 with MQTT – Publish DHT11/DHT22 Temperature and Humidity Readings (Arduino IDE)*. https://randomnerdtutorials.com/esp32-mqtt-publish-dht11-dht22-arduino/

Mosquitto. (n.d.). *Eclipse Mosquitto - An open source MQTT broker*. https://mosquitto.org/