Simulasi Komunikasi Data IoT Menggunakan MQTT dan ESP32 Berbasis Platform Wokwi

Azzam Beryl Nemesio Wijoyo
Fakultas Vokasi, Universitas Brawijaya
nemesioberyl@gmail.com

ABSTRAK

Perkembangan teknologi Internet of Things (IoT) telah membawa perubahan signifikan dalam cara perangkat saling berkomunikasi. Salah satu protokol komunikasi yang ringan dan efisien dalam sistem IoT adalah MQTT (Message Queuing Telemetry Transport). Praktikum ini mensimulasikan komunikasi dua arah antara ESP32 dengan broker MQTT melalui publish-subscribe, dengan memanfaatkan platform simulasi Wokwi. Dalam implementasinya, ESP32 terhubung dengan sensor DHT22 dan LED, di mana data suhu dan kelembapan dipublikasikan ke broker MQTT, sementara LED dikendalikan melalui pesan dari dashboard MQTT seperti MQTT Explorer. Hasil menunjukkan komunikasi berjalan optimal, responsif, dan stabil, membuktikan potensi integrasi ESP32 dan MQTT dalam sistem monitoring atau kontrol jarak jauh.

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

ABSTRACT

The development of Internet of Things (IoT) technology has significantly changed how devices communicate. MQTT (Message Queuing Telemetry Transport) is a lightweight and efficient protocol used in many IoT systems. This practicum simulates bidirectional communication between an ESP32 and an MQTT broker using the publish-subscribe model, utilizing the Wokwi simulation platform. The ESP32 is connected to a DHT22 sensor and an LED, where temperature and humidity data are published to the MQTT broker, while the LED is controlled via MQTT commands sent from a dashboard such as MQTT Explorer. The results indicate successful, responsive, and stable communication, demonstrating the potential of integrating ESP32 and MQTT in remote monitoring or control systems.

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

---

1. PENDAHULUAN

1.1 Latar Belakang

Internet of Things (IoT) merupakan konsep integrasi berbagai perangkat melalui jaringan internet untuk bertukar data secara otomatis. Salah satu protokol komunikasi populer dalam IoT adalah MQTT, karena sifatnya yang ringan dan hemat bandwidth. MQTT memanfaatkan pendekatan *publish-subscribe* yang memungkinkan komunikasi data secara asinkron. Dalam praktikum ini, dilakukan simulasi penerapan komunikasi MQTT dengan ESP32 sebagai perangkat utama. Data dari sensor suhu dan kelembapan (DHT22) dipublikasikan secara berkala, sedangkan LED dikontrol melalui topik tertentu sebagai bentuk aktuasi jarak jauh.

1.2 Tujuan Praktikum

Tujuan dari praktikum ini adalah untuk:

1. Memahami cara kerja protokol MQTT dalam komunikasi data IoT.
2. Menghubungkan ESP32 dengan broker MQTT untuk publikasi dan langganan data.
3. Mempublikasikan data suhu dan kelembapan dari sensor DHT22.
4. Mengontrol LED melalui perintah MQTT yang dikirim dari dashboard MQTT Explorer.
5. Melakukan simulasi penuh melalui platform Wokwi dan Visual Studio Code.

---

2. METODOLOGI

2.1 Alat dan Bahan

Perangkat Lunak dan Peralatan:

- Komputer/Laptop dengan koneksi internet
- Platform Wokwi (https://wokwi.com)
- Visual Studio Code dengan ekstensi PlatformIO
- MQTT Explorer

Komponen Virtual yang Disimulasikan:

- ESP32 DevKit V1
- Sensor DHT22
- LED
- MQTT broker publik: test.mosquitto.org (port 1883)

2.2 Langkah Implementasi

Langkah 1: Membuat Simulasi di Wokwi

- Tambahkan ESP32, DHT22, dan LED.
- Hubungkan:
    - LED ke pin D2 dan GND
    - DHT22 ke pin D15 dan 3V3/GND

Langkah 2: Penulisan Program

- Gunakan pustaka: WiFi.h, PubSubClient.h, dan DHTesp.h
- Hubungkan ESP32 ke jaringan WiFi Wokwi-GUEST
- Publikasikan data sensor ke topik:

- IOT/Test1/temp untuk suhu
- IOT/Test1/hum untuk kelembapan

- Subcribe ke topik:
  - IOT/Test1/mqtt untuk mengontrol LED

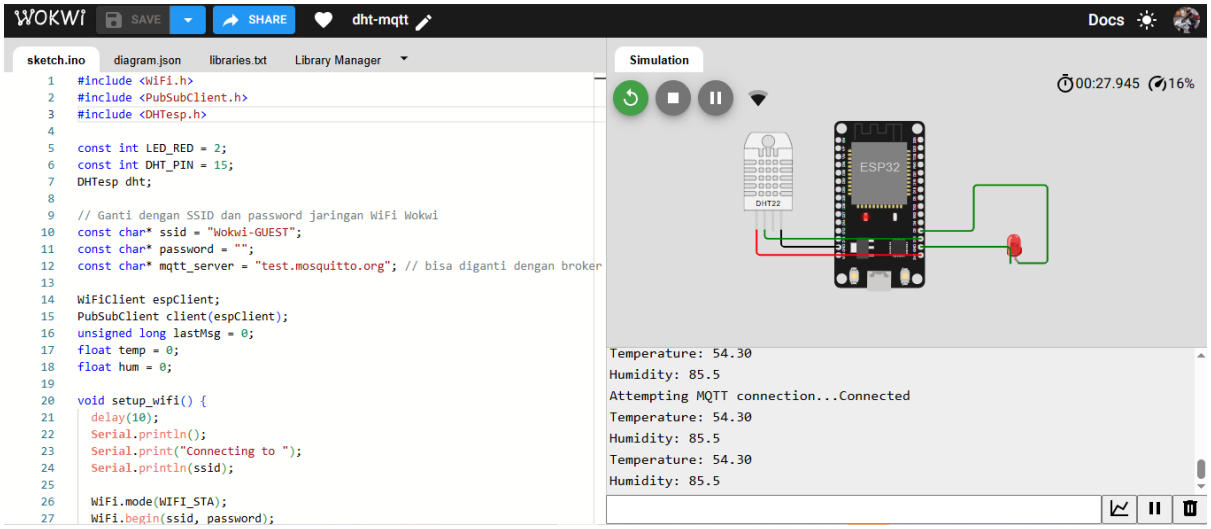
Langkah 3: Pengujian dengan MQTT Explorer

- Buka MQTT Explorer
- Hubungkan ke test.mosquitto.org
- Subscribe ke IOT/Test1/temp dan IOT/Test1/hum
- Publish nilai:
  - "1" ke IOT/Test1/mqtt → LED menyala
  - "0" ke IOT/Test1/mqtt → LED mati

---

3. HASIL DAN PEMBAHASAN

3.1 Hasil Simulasi



ESP32 berhasil melakukan:

- Koneksi ke WiFi dan broker MQTT
- Publikasi data sensor secara berkala
- Respons terhadap perintah dari dashboard MQTT Explorer

Contoh data:

- Topik: IOT/Test1/temp → Nilai: 24.00 °C

- Topik: IOT/Test1/hum → Nilai: 40.0 %
- LED menyala saat pesan 1 diterima
- LED mati saat pesan 0 diterima

Tampilan Serial Monitor:

bash

CopyEdit

Temperature: 24.00

Humidity: 40.0

Message arrived [IOT/Test1/mqtt] 1

3.2 Pembahasan

Protokol MQTT terbukti efisien dan handal untuk komunikasi data antarperangkat IoT. Dalam simulasi ini, sistem menunjukkan stabilitas dalam koneksi, kecepatan respons, dan konsistensi dalam pertukaran data. Dengan pendekatan ringan berbasis publish-subscribe, sistem ini dapat diperluas menjadi jaringan perangkat IoT yang lebih besar seperti smart home atau monitoring lingkungan. Penggunaan broker publik seperti test.mosquitto.org memungkinkan pengujian tanpa infrastruktur lokal, meskipun dalam implementasi nyata sebaiknya menggunakan broker privat untuk keamanan dan kontrol yang lebih baik.

---

4. LAMPIRAN

4.1 Kode Program ESP32 (sketch.ino)

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int LED_RED = 2;
const int DHT_PIN = 15;
DHTesp dht;

// Ganti dengan SSID dan password jaringan WiFi Wokwi
const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* mqtt_server = "test.mosquitto.org"; // bisa diganti dengan broker lain jika perlu
```

```
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);
```

```
  }
}
```

4.2 Kode diagram.json

```
{
  "version": 1,
  "author": "Beryl",
  "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": {}
}
```
