Praktik Implementasi Pengiriman Data Sensor Menggunakan MQTT dan ESP32

oleh

Nadia Alya Paramitha Erwanto[1]

Fakultas Vokasi, Universitas Brawijaya

Email:nadiaalya2729@student.ub.ac.id

Abstrak

Internet of Things (IoT) terus mengalami perkembangan pesat dan memberikan peluang besar dalam menciptakan sistem yang saling terhubung dan dapat dikendalikan secara jarak jauh melalui internet. Salah satu protokol komunikasi yang mendukung pengembangan IoT adalah MQTT (Message Queuing Telemetry Transport), yang dikenal ringan dan efisien untuk komunikasi antar perangkat. Dalam bab ini, dibahas praktik implementasi penggunaan protokol MQTT dengan mikrokontroler ESP32 untuk mengembangkan sistem IoT berbasis komunikasi data ringan. Platform Wokwi digunakan sebagai media simulasi perangkat keras, memungkinkan pengguna untuk menguji dan memvalidasi program sebelum diterapkan pada perangkat fisik. Studi kasus yang diangkat mencakup koneksi ESP32 ke broker MQTT, pengiriman dan penerimaan data sensor, serta visualisasi data melalui dashboard. Hasil praktik ini menunjukkan bahwa kombinasi ESP32, MQTT, dan simulator Wokwi memberikan kemudahan dalam perancangan, pengujian, serta efisiensi dalam pengembangan sistem IoT.

***Kata kunci**: Internet of Things, MQTT, ESP32, Wokwi, komunikasi data, simulasi perangkat keras, sistem IoT*

Abstract

The Internet of Things (IoT) continues to evolve rapidly, offering great opportunities in creating systems that are interconnected and remotely controllable via the internet. One of the communication protocols that supports IoT development is MQTT (Message Queuing Telemetry Transport), known for its lightweight and efficient communication between devices. This chapter discusses the practical implementation of the MQTT protocol using the ESP32 microcontroller to develop an IoT system based on lightweight data communication. The Wokwi platform is used as a hardware simulation tool, allowing users to test and validate programs before applying them to physical devices. The case study includes connecting the ESP32 to an MQTT broker, sending and receiving sensor data, and visualizing the data through a dashboard. The results of this practice show that the combination of ESP32, MQTT, and the Wokwi simulator provides ease in designing, testing, and efficiently developing IoT systems.

Keywords: Internet of Things, MQTT, ESP32, Wokwi, simulation, lightweight communication

Pendahuluan

Perkembangan teknologi Internet of Things (IoT) telah membawa perubahan signifikan dalam berbagai aspek kehidupan, mulai dari rumah pintar (smart home), pertanian cerdas (smart agriculture), hingga industri manufaktur dan sistem kesehatan. IoT memungkinkan berbagai perangkat untuk saling terhubung, bertukar data, serta dikendalikan secara jarak jauh melalui jaringan internet, sehingga menciptakan sistem yang lebih cerdas, efisien, dan responsif.

Salah satu komponen penting dalam pengembangan sistem IoT adalah protokol komunikasi yang efisien dan ringan. MQTT (Message Queuing Telemetry Transport) merupakan salah satu protokol yang banyak digunakan karena kemampuannya dalam mengirim dan menerima data secara efisien, meskipun pada jaringan dengan bandwidth rendah. MQTT dirancang dengan model publish-subscribe, sehingga cocok digunakan untuk aplikasi IoT yang melibatkan banyak perangkat dan memerlukan komunikasi real-time.

ESP32 adalah mikrokontroler yang mendukung koneksi Wi-Fi dan Bluetooth, serta memiliki kemampuan pemrosesan yang cukup tinggi untuk berbagai aplikasi IoT. Dengan harga yang terjangkau dan fitur yang lengkap, ESP32 menjadi pilihan populer dalam implementasi sistem berbasis IoT.

Dalam praktik ini, dilakukan simulasi pengembangan sistem IoT menggunakan ESP32 dan protokol MQTT dengan bantuan platform simulasi Wokwi. Wokwi memungkinkan pengguna untuk menguji dan mengembangkan proyek berbasis mikrokontroler secara virtual tanpa memerlukan perangkat keras fisik. Praktik ini bertujuan untuk memahami proses koneksi antara ESP32 dan broker MQTT, mengirim dan menerima data sensor, serta menampilkannya pada dashboard secara real-time. Melalui simulasi ini, pengguna dapat memperoleh gambaran dan pengalaman praktis dalam merancang sistem IoT yang efisien sebelum diterapkan pada lingkungan nyata.

Metodologi

Metodologi yang digunakan dalam praktik ini mencakup tahapan perencanaan, perancangan sistem, simulasi, dan pengujian. Praktik dilakukan secara virtual menggunakan platform Wokwi, dengan ESP32 sebagai mikrokontroler dan protokol MQTT sebagai media komunikasi data. Berikut langkah-langkah yang dilakukan:

1. Perancangan Sistem

Langkah awal dalam praktik ini adalah merancang sistem IoT sederhana yang melibatkan ESP32 sebagai pusat kendali, sensor sebagai input, serta koneksi ke broker MQTT untuk komunikasi data. Perancangan sistem dilakukan dengan mempertimbangkan komponen virtual yang tersedia pada simulator Wokwi.

2. Pemrograman ESP32

Setelah sistem dirancang, tahap berikutnya adalah membuat kode program untuk ESP32 menggunakan Arduino IDE. Program dibuat untuk:

- Menginisialisasi koneksi Wi-Fi
- Menghubungkan ke broker MQTT
- Membaca data dari sensor (misalnya sensor suhu dan kelembaban)
- Mengirim data sensor ke topik tertentu (publish)
- Menerima perintah dari topik lain (subscribe), jika sistem juga mendukung kontrol

3. Konfigurasi Broker MQTT

Dalam praktik ini digunakan broker MQTT publik seperti **broker.hivemq.com** atau **test.mosquitto.org**. Konfigurasi mencakup:

- Menentukan alamat broker
- Menentukan nama client ID

- Menentukan topik (topic) untuk publish dan subscribe

4. Simulasi pada Wokwi

Selanjutnya dilakukan simulasi menggunakan platform Wokwi. Proyek ESP32 dan sensor virtual dikonfigurasi sesuai dengan rancangan sistem. Wokwi memungkinkan pengguna untuk melihat output program dan aktivitas MQTT secara real-time melalui terminal dan dashboard MQTT (seperti MQTT Explorer atau dashboard Blynk jika dikombinasikan).

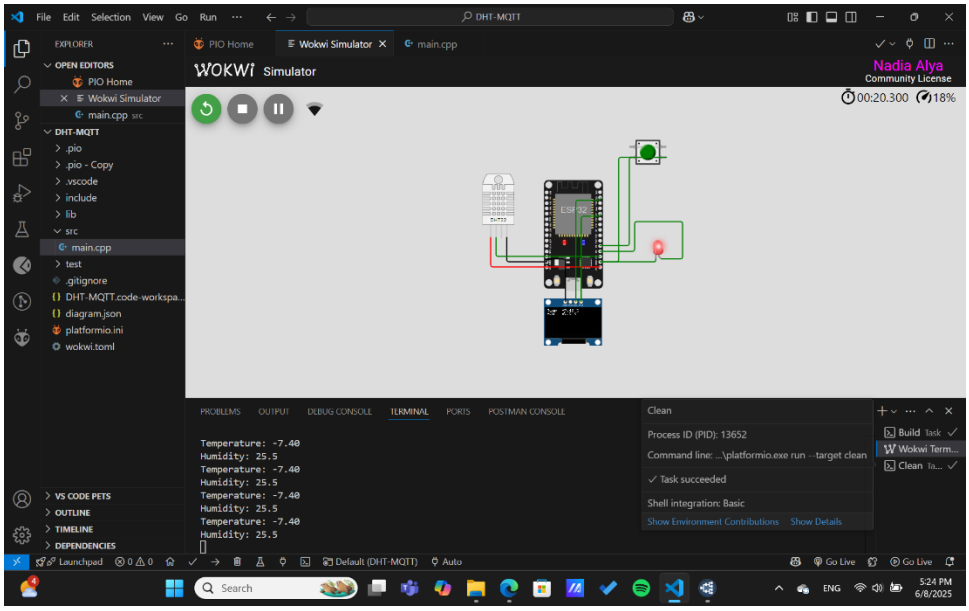
5. Pengujian dan Pemantauan

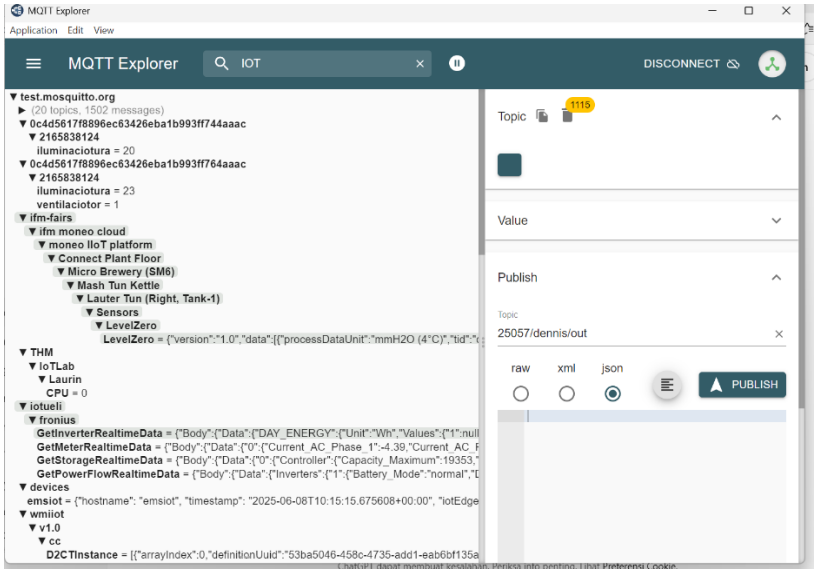
Tahap akhir adalah pengujian sistem. ESP32 dijalankan untuk membaca data sensor, lalu mengirimkannya ke broker MQTT. Data yang terkirim dapat dimonitor menggunakan MQTT dashboard atau serial monitor. Respons dari sistem terhadap data yang dikirim dan diterima juga diamati untuk memastikan komunikasi berjalan sesuai tujuan.

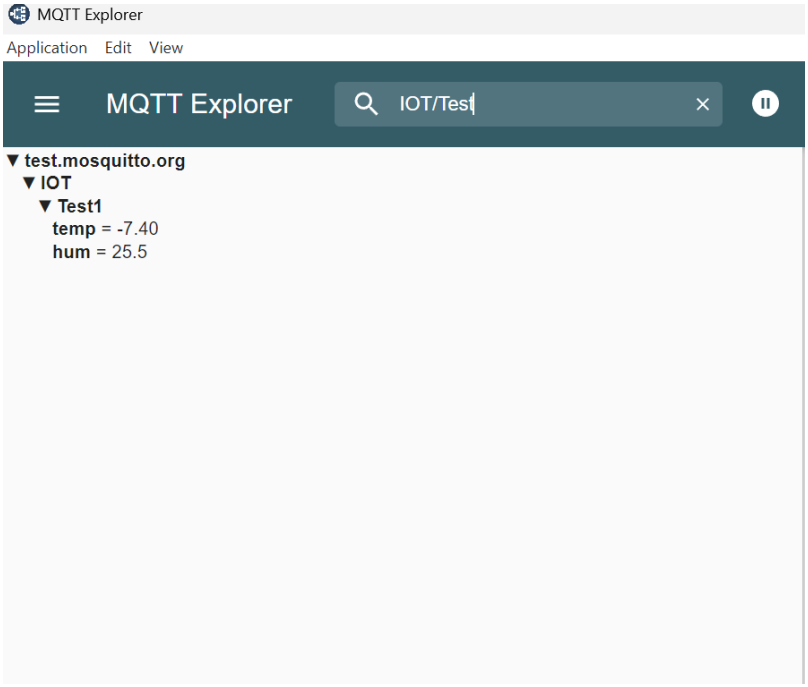
Hasil dan Pembahasan

Data ini dapat dimonitor menggunakan MQTT Explorer atau dashboard berbasis web/mobile yang terhubung dengan broker yang sama.

Jika pada kode juga ditambahkan perintah subscribe untuk menerima data dari topik tertentu (misalnya: iot/perintah/lampu), maka ESP32 juga merespons perintah dari dashboard atau publisher lain untuk mengaktifkan atau menonaktifkan aktuator (misalnya LED atau relay virtual).







Berdasarkan hasil praktik, dapat disimpulkan bahwa kombinasi ESP32, MQTT, dan Wokwi mampu menyediakan solusi simulasi IoT yang cukup andal dan efisien. Beberapa hal penting yang ditemukan selama praktik:

- Koneksi MQTT Stabil: ESP32 dapat terhubung ke broker MQTT dengan cukup stabil dalam simulasi, selama jaringan Wi-Fi virtual terhubung.
- Pengiriman Data Berjalan Lancar: Data dari sensor berhasil dikirim ke broker dan dapat diterima oleh client lain dalam topik yang sama secara real-time.
- Kemudahan Monitoring: Dengan menggunakan MQTT dashboard seperti MQTT Explorer, data dapat dimonitor dengan antarmuka yang user-friendly.
- Respon Sistem Cepat: Ketika perintah dikirim melalui topik subscribe, ESP32 dapat merespons dengan cepat, misalnya menyalakan atau mematikan LED.

Namun demikian, terdapat beberapa keterbatasan dalam simulasi, seperti ketergantungan pada jaringan dan kemampuan visualisasi terbatas jika dibandingkan dengan perangkat nyata. Meski begitu, Wokwi sangat membantu dalam tahap pengujian logika dan komunikasi MQTT sebelum sistem diterapkan secara fisik.

Lampiran

Source code **main.cpp :**

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

const int LED_RED = 2;
const int DHT_PIN = 15;
const int BUTTON_PIN = 4;

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;
unsigned long lastButtonDisplay = 0;
bool lastButtonState = HIGH;
bool showButtonMessage = false;

void setup_wifi() {
```

```
  delay(10);
  WiFi.mode(WIFI_STA);
  Serial.println("Nyari WiFi... Jangan di-skip ya!");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi nyambung coy!");
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Dapet pesan MQTT di topik: ");
  Serial.println(topic);
  if ((char)payload[0] == '1') {
    digitalWrite(LED_RED, HIGH);
    Serial.println("LED nyala! Horeeee!");
  } else {
    digitalWrite(LED_RED, LOW);
    Serial.println("LED mati, tenang bro.");
  }
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Nyoba connect ke MQTT... ");
    String clientId = "ESP32Client-";
    clientId += String(random(0xffff), HEX);

    if (client.connect(clientId.c_str())) {
      Serial.println("Yesss, MQTT connected!");
      client.publish("IOT/Test1/status", "ESP32 connected");
```

```
      client.subscribe("IOT/Test1/mqtt");
    } else {
      Serial.print("Gagal, kode error = ");
      Serial.print(client.state());
      Serial.println(". Coba lagi 5 detik lagi ya.");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(LED_RED, OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);

  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  dht.setup(DHT_PIN, DHTesp::DHT22);

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("OLED gagal dimulai, nangis deh!"));
    while (true);
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println("Starting...");
  display.display();
  delay(1000);
}
```

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  Serial.println("MQTT client loop aktif, santuy...");

  unsigned long now = millis();

  bool buttonState = digitalRead(BUTTON_PIN);
  if (buttonState == LOW && lastButtonState == HIGH) {
    Serial.println("Tombol ditekan, horeee!");
    if (client.connected()) {
      client.publish("IOT/Test1/button", "Tombol ditekan!");
      Serial.println("Pesan tombol berhasil dipublish!");
    }
    digitalWrite(LED_RED, !digitalRead(LED_RED)); // Toggle LED

    showButtonMessage = true;
    lastButtonDisplay = now;
  }
  lastButtonState = buttonState;

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

    String tempStr = String(data.temperature, 2);
    String humStr = String(data.humidity, 1);

    if (client.connected()) {
```

```
    client.publish("IOT/Test1/temp", tempStr.c_str());
    client.publish("IOT/Test1/hum", humStr.c_str());
    Serial.println("Temperature dan Humidity berhasil dipublish!");
   } else {
    Serial.println("MQTT belum terkoneksi, skip publish data sensor.");
   }

   Serial.println("Temperature: " + tempStr + " C");
   Serial.println("Humidity: " + humStr + " %");

   display.clearDisplay();
   display.setCursor(0, 0);
   display.print("Temp: ");
   display.print(tempStr);
   display.println(" C");
   display.print("Hum : ");
   display.print(humStr);
   display.println(" %");

   if (showButtonMessage) {
    display.setCursor(0, 40);
    display.println("Button Pressed!");
   }

   display.display();
  }

  if (showButtonMessage && (now - lastButtonDisplay > 2000)) {
   showButtonMessage = false;
  }
}
```