**Simulasi Komunikasi MQTT pada ESP32 dengan Sensor DHT22 dan Kontrol LED menggunakan Platform Wokwi**

Mohammad Haryo Hammam Q  
233140700111108 **ABSTRAK**

Pada praktikum ini dilakukan simulasi komunikasi data antara mikrokontroler ESP32 dengan broker MQTT menggunakan protokol publish-subscribe berbasis platform Wokwi. ESP32 terhubung dengan sensor DHT22 untuk membaca suhu dan kelembapan, serta LED sebagai aktuator. Data sensor dipublikasikan ke broker MQTT publik (test.mosquitto.org) dan dapat dimonitor melalui dashboard seperti MQTT Explorer. Sebaliknya, ESP32 juga dapat menerima perintah untuk menghidupkan atau mematikan LED melalui topik tertentu. Hasil simulasi menunjukkan bahwa komunikasi data dua arah melalui MQTT berjalan dengan lancar dan efisien. Proyek ini menunjukkan potensi ESP32 dalam mengembangkan sistem IoT yang ringan dan real-time.

**Kata Kunci**: ESP32, MQTT, IoT, Wokwi, DHT22, LED

**ABSTRACT**

This practicum simulates data communication between an ESP32 microcontroller and an MQTT broker using the publish-subscribe protocol on the Wokwi platform. The ESP32 is connected to a DHT22 sensor to read temperature and humidity, and to an LED as an actuator. Sensor data is published to a public MQTT broker (test.mosquitto.org) and monitored using tools like MQTT Explorer. Conversely, commands can be sent from the broker to control the LED. The simulation results confirm successful and efficient bidirectional communication via MQTT. This project demonstrates the potential of ESP32 in building lightweight, real-time IoT systems.

**Keywords**: ESP32, MQTT, IoT, Wokwi, DHT22, LED

**1. PENDAHULUAN**

**1.1 Latar Belakang**

Internet of Things (IoT) menjadi solusi penting dalam sistem otomatisasi dan monitoring modern. Salah satu protokol komunikasi yang banyak digunakan dalam IoT adalah MQTT (Message Queuing Telemetry Transport) yang bersifat ringan dan cepat. ESP32, sebagai mikrokontroler dengan koneksi WiFi bawaan, sangat cocok diintegrasikan dengan MQTT untuk membangun sistem real-time berbasis publish-subscribe.

Dalam praktikum ini, ESP32 disimulasikan di Wokwi untuk mengirimkan data suhu dan kelembapan dari sensor DHT22 ke broker MQTT, serta menerima perintah untuk mengontrol LED. Dashboard MQTT seperti MQTT Explorer digunakan untuk memonitor dan mengirim pesan.

**1.2 Tujuan Praktikum**

* Memahami prinsip kerja protokol MQTT dalam sistem IoT.
* Membangun simulasi komunikasi dua arah antara ESP32 dan broker MQTT.
* Mengirim data suhu dan kelembapan ke broker MQTT.
* Mengontrol LED dari jarak jauh melalui perintah MQTT.

**2. METODOLOGI**

**2.1 Alat dan Bahan**

**Alat:**

* Wokwi IoT Simulator (<https://wokwi.com>)
* MQTT Explorer (Desktop App)

**Bahan:**

* ESP32 Devkit V1 (simulasi Wokwi)
* Sensor DHT22
* LED merah
* Broker MQTT publik: test.mosquitto.org, port 1883
* Bahasa C++ (Arduino Framework)

**2.2 Langkah Implementasi**

1. **Membuat Rangkaian di Wokwi**
   * Tambahkan ESP32, sensor DHT22, dan LED
   * Sambungkan:
     + LED ke GPIO 2
     + DHT22 ke GPIO 15
     + VCC dan GND disesuaikan
2. **Menulis Program ESP32**
   * Gunakan library:
     + WiFi.h
     + PubSubClient.h
     + DHTesp.h
   * Inisialisasi WiFi, MQTT, dan DHT22
   * Publish data ke topic:
     + IOT/Test1/temp
     + IOT/Test1/hum
   * Subscribe ke topic:
     + IOT/Test1/mqtt (untuk kontrol LED)
3. **Testing**
   * Jalankan simulasi di Wokwi
   * Lihat Serial Monitor untuk verifikasi koneksi
   * Buka MQTT Explorer untuk:
     + Subscribe ke IOT/Test1/temp dan IOT/Test1/hum
     + Publish ke IOT/Test1/mqtt dengan payload 1 (LED nyala) dan 0 (LED mati)

**3. HASIL DAN PEMBAHASAN**

**3.1 Hasil**

* ESP32 berhasil terhubung ke WiFi dan broker MQTT.
* Data sensor DHT22 berhasil dikirim ke broker setiap 2 detik:
  + IOT/Test1/temp: 27.50
  + IOT/Test1/hum: 55.0
* LED dapat dikontrol melalui MQTT Explorer:
  + Kirim 1 → LED menyala
  + Kirim 0 → LED mati

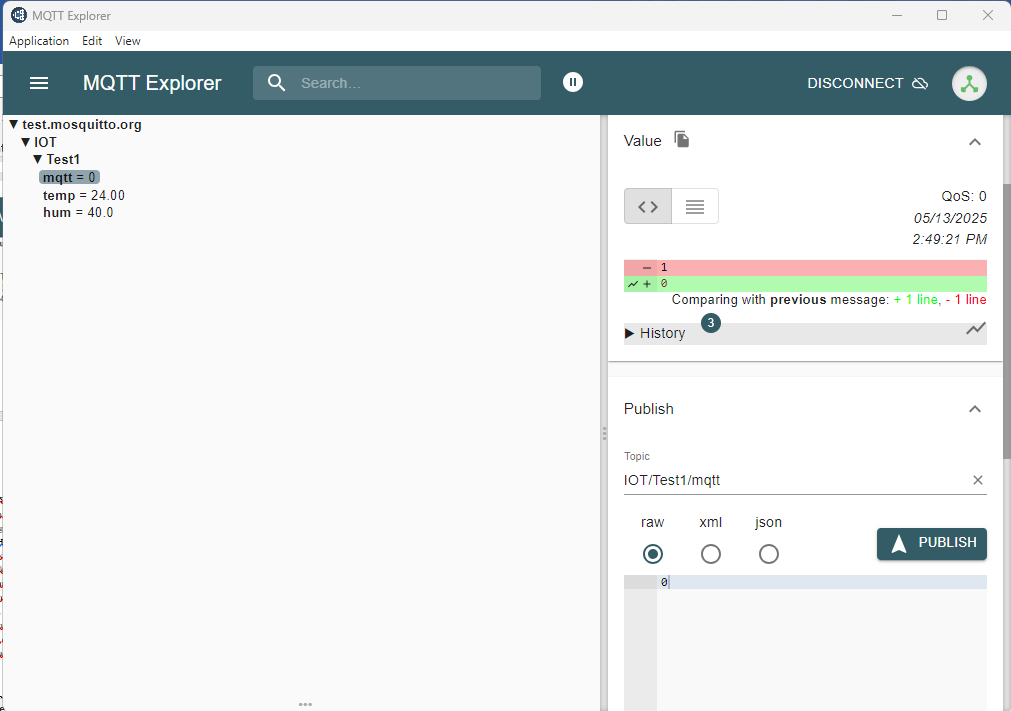
**3.2 Pembahasan**

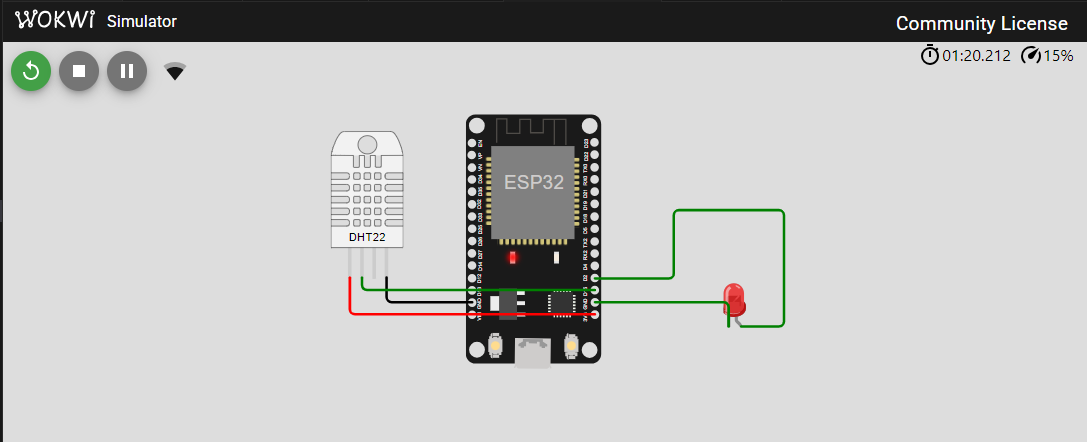
MQTT sebagai protokol komunikasi ringan sangat ideal untuk sistem IoT dengan sumber daya terbatas seperti ESP32. Keunggulan publish-subscribe menjadikan komunikasi data lebih efisien karena tidak perlu koneksi langsung antar perangkat. Simulasi ini menunjukkan penerapan konsep IoT secara nyata meskipun tanpa perangkat fisik, memungkinkan pengembangan awal sebelum deployment.

**4. PENUTUP**

**4.1 Kesimpulan**

Praktikum ini berhasil mensimulasikan komunikasi data IoT antara ESP32 dan broker MQTT menggunakan Wokwi. ESP32 dapat berperan sebagai publisher dan subscriber sekaligus, memungkinkan monitoring data sensor dan kontrol aktuator secara real-time melalui jaringan internet.

****

****

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.

* 1. **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.

1. **LAMPIRAN**
   1. **Kode program**

#include <WiFi.h>

#include <PubSubClient.h>

#include <DHTesp.h>

const int LED\_RED = 2;

const int DHT\_PIN = 15;

DHTesp dht;

const char\* ssid = "KOS Putra Joyoagung 5G";

const char\* password = "joyoagung321";

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() {

  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);

  }

}

* 1. **Kode diagram.json**

{

  "version": 1,

  "author": "awikwok",

  "editor": "wokwi",

  "parts": [

    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -24.1, "left": -5, "attrs": {} },

    { "type": "wokwi-dht22", "id": "dht1", "top": -153.3, "left": 119.4, "attrs": {} },

    {

      "type": "wokwi-led",

      "id": "led1",

      "top": -128.4,

      "left": 205.4,

      "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": {}

}