

Implementasi MQTT pada ESP32 untuk Monitoring Real-Time Sensor DHT22 melalui Web Dashboard

Oleh

Ifadah Aulia Muhti Sinaga

233140707111034

ifadahauliasinaga@gmail.com

Abstrak:

Proyek ini mengimplementasikan sistem monitoring suhu dan kelembapan secara real-time menggunakan mikrokontroler ESP32, sensor DHT22, protokol MQTT, dan dashboard web. Sensor DHT22 digunakan untuk mendeteksi suhu dan kelembapan lingkungan, lalu data dikirim secara berkala melalui jaringan Wi-Fi ke broker MQTT (test.mosquitto.org). Data tersebut kemudian divisualisasikan dalam bentuk angka dan grafik interaktif pada halaman web menggunakan JavaScript dan Chart.js. Pengguna juga dapat mengontrol LED secara langsung melalui dashboard dengan mengirimkan perintah melalui topik MQTT. Sistem ini menunjukkan kemampuan integrasi antara perangkat IoT dan antarmuka pengguna berbasis web untuk monitoring jarak jauh secara efisien dan responsif.

Kata Kunci : ESP32, DHT22, MQTT, IoT, Web Dashboard, Sensor Suhu, Sensor Kelembapan, Real-Time Monitoring, Chart.js, JavaScript

Abstract:

This project implements a real-time temperature and humidity monitoring system using the ESP32 microcontroller, DHT22 sensor, MQTT protocol, and a web-based dashboard. The DHT22 sensor is used to detect environmental temperature and humidity, and the data is periodically transmitted over Wi-Fi to a public MQTT broker (test.mosquitto.org). The transmitted data is then visualized in real-time on a web page using JavaScript and Chart.js, displaying both numerical values and interactive graphs. Users can also control an LED remotely via the dashboard by publishing commands to a specific MQTT topic. This system demonstrates the integration of IoT devices with a responsive web interface for efficient and remote environmental monitoring.

Keyword: ESP32, DHT22, MQTT, IoT, Web Dashboard, Temperature Sensor, Humidity Sensor, Real-Time Monitoring, Chart.js, JavaScript

1. Pendahuluan

1.1 Latar belakang

Internet of Things (IoT) merupakan konsep yang memungkinkan perangkat fisik terhubung ke internet untuk saling bertukar data dan informasi. Dalam implementasi IoT, monitoring suhu dan kelembapan lingkungan menjadi salah satu aplikasi yang umum digunakan, baik untuk kebutuhan rumah pintar, pertanian, industri, maupun ruang server. Mikrokontroler ESP32 yang memiliki konektivitas Wi-Fi terintegrasi sangat cocok digunakan sebagai pengirim data sensor ke server secara real-time.

Protokol Message Queuing Telemetry Transport (MQTT) digunakan karena ringan, cepat, dan efisien dalam mengirimkan data antar perangkat. Untuk memudahkan pengguna dalam memantau data yang dikirimkan oleh sensor, dibutuhkan antarmuka visual yang dapat diakses melalui web browser. Oleh karena itu, eksperimen ini menggabungkan penggunaan ESP32, sensor DHT22, protokol MQTT, dan visualisasi data menggunakan web dashboard berbasis JavaScript.

1.2 Tujuan eksperimen

- Membangun sistem monitoring suhu dan kelembapan secara real-time menggunakan ESP32 dan sensor DHT22.
- Mengirimkan data sensor melalui protokol MQTT ke broker publik.
- Menampilkan data suhu dan kelembapan dalam bentuk angka dan grafik menggunakan web dashboard interaktif.
- Memberikan fitur kendali LED dari dashboard web ke ESP32 melalui MQTT sebagai bentuk komunikasi dua arah (bidirectional).
- Menerapkan teknologi IoT yang efisien dan mudah diakses untuk kebutuhan monitoring lingkungan berbasis web.

2. Metodologi

2.1 Alat dan Bahan

Karena proyek ini berbasis simulasi menggunakan Visual Studio Code (VS Code) dan broker MQTT publik, maka tidak memerlukan perangkat keras fisik. Berikut adalah daftar alat dan bahan yang digunakan:

- Visual Studio Code - Editor untuk menulis dan menjalankan kode
- PlatformIO Extension - Untuk menjalankan kode ESP32 di VS Code
- Mikrokontroler ESP32 - Disimulasikan di lingkungan PlatformIO
- Sensor DHT22 - Sensor suhu dan kelembapan
- Broker MQTT test.mosquitto.org sebagai server MQTT
- MQTT.js - Library JavaScript untuk koneksi MQTT
- Chart.js - Library visualisasi grafik suhu & kelembapan
- Web Browser - Untuk menampilkan dashboard (Chrome/Firefox)
- File HTML, CSS, JS - Antarmuka pengguna berbasis web

2.2 Langkah Implementasi

Langkah implementasi untuk Implementasi MQTT pada ESP32 untuk Monitoring Real-Time Sensor DHT22 melalui Web Dashboard sebagai berikut:

- Buka Wokwi Simulator dan buat proyek baru
- Buat proyek PlatformIO baru untuk board ESP32.
- Tambahkan library WiFi.h, PubSubClient.h, dan DHTesp.h.
- Tulis kode untuk membaca suhu & kelembapan dari sensor DHT22.
- Kirim data ke broker MQTT test.mosquitto.org dengan topik:
IOT/Test1/temp untuk suhu

- IOT/Test1/hum untuk kelembapan
 IOT/Test1/mqtt untuk kontrol LED
- Buat file index.html, style.css, dan script.js.
 - Gunakan MQTT.js untuk berlangganan ke topik:
 IOT/Test1/temp dan IOT/Test1/hum
 - Tampilkan data real-time ke dalam elemen dan grafik interaktif menggunakan Chart.js.
 - Tambahkan tombol untuk mengirim perintah "1" atau "0" ke IOT/Test1/mqtt agar ESP32 dapat mengaktifkan/mematikan LED.
 - Jalankan ESP32 dari PlatformIO, pastikan koneksi WiFi dan MQTT berhasil.
 Buka index.html di browser.
 Amati suhu dan kelembapan ditampilkan dan tergambar secara real-time.
 Uji fungsi tombol "Nyalakan LED" dan "Matikan LED" dari dashboard untuk melihat respons dari ESP32.

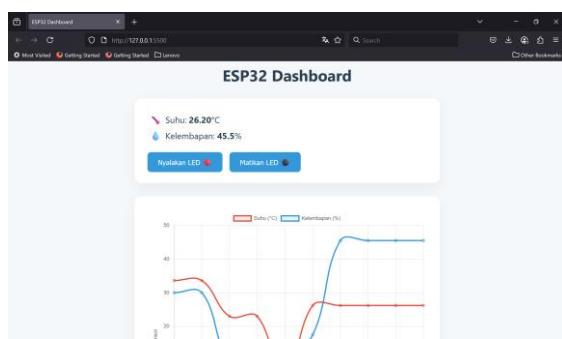
3. Hasil dan Pembahasan

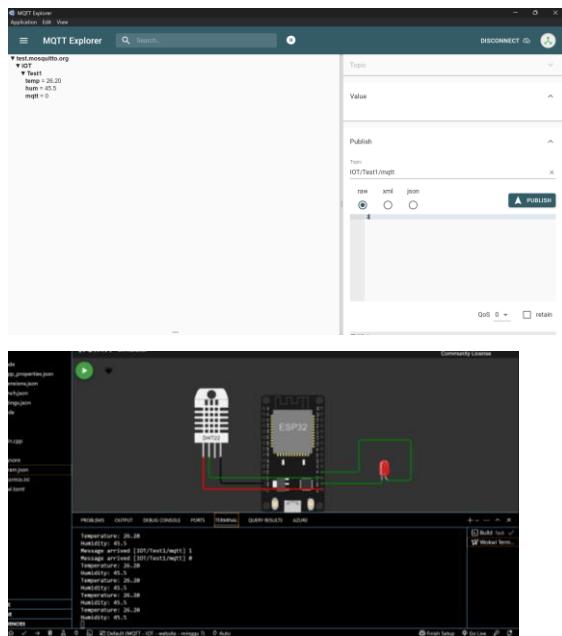
3.1 Hasil Eksperimen

Eksperimen yang dilakukan berhasil menunjukkan integrasi antara perangkat ESP32 dengan protokol komunikasi MQTT dan dashboard web berbasis JavaScript. Sistem dapat berjalan secara real-time dan responsif dalam membaca serta menampilkan data sensor suhu dan kelembapan.

Pada saat dijalankan, ESP32 terkoneksi ke jaringan Wi-Fi dan secara periodik mengirimkan data suhu dan kelembapan dari sensor DHT22 ke broker MQTT test.mosquitto.org. Data tersebut kemudian diterima oleh dashboard web yang berlangganan ke topik yang sama melalui MQTT over WebSocket.

Nilai suhu dan kelembapan berhasil ditampilkan secara real-time dalam bentuk angka dan juga grafik garis dinamis menggunakan Chart.js. Selain itu, fitur kontrol LED juga berfungsi dengan baik; ketika tombol "Nyalakan LED" ditekan, ESP32 menerima pesan 1 dan menyalakan LED, sedangkan tombol "Matikan LED" mengirim pesan 0 dan mematikan LED.





4. Lampiran

- Kode program main.cpp

```

• #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";
•
• 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); //inisialisasi 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);
•     }
• }
```

- }

- Kode program diagram.json

```
• {
•   "version": 1,
•   "author": " Anonymous maker",
•   "editor": "wokwi",
•   "parts": [
•     {
•       "type": "wokwi-esp32-devkit-v1",
•       "id": "esp",
•       "top": 0,
•       "left": 0,
•       "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",
•       [
•         [
•           "dht1:leftPin"
•         ]
•       ]
•     ]
•   ]
• }
```

```

•           "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": {}
• }

```

- Index.html

```

<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <title>ESP32 Dashboard</title>
  <link rel="stylesheet" href="style.css">
  <script src="https://unpkg.com/mqtt/dist/mqtt.min.js"></script>

```

```

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script src="script.js" defer></script>
</head>
<body>
  <h1>ESP32 Dashboard</h1>

  <div class="card">
    <p><span class="icon">🌡</span> Suhu: <span id="temp">---</span> °C</p>
    <p><span class="icon">💧</span> Kelembapan: <span id="hum">---</span> %</p>
    <button onclick="turnOn()">Nyalakan LED 🌟</button>
    <button onclick="turnOff()">Matikan LED ●</button>
  </div>

  <div class="card">
    <canvas id="sensorChart" height="300"></canvas>
  </div>

</body>
</html>

```

- Style.css

```

• :root {
•   --primary: #3498db;
•   --secondary: #2c3e50;
•   --red: #e74c3c;
•   --blue: #3498db;
•   --bg: #f4f8fb;
•   --white: #ffffff;
•   --shadow: 0 10px 25px rgba(0, 0, 0, 0.05);
• }
•
• *
•   margin: 0;
•   padding: 0;
•   box-sizing: border-box;
• }
•
• body {
•   background-color: var(--bg);
•   font-family: 'Segoe UI', sans-serif;
•   padding: 40px 20px;
•   color: var(--secondary);
•   display: flex;
•   flex-direction: column;
•   align-items: center;
• }
•
• h1 {
•   font-size: 36px;
•   margin-bottom: 30px;
• }

```

```
•
•   .card {
•     background-color: var(--white);
•     padding: 25px 30px;
•     border-radius: 16px;
•     box-shadow: var(--shadow);
•     width: 100%;
•     max-width: 700px;
•     margin-bottom: 30px;
•   }
•
•   p {
•     font-size: 20px;
•     margin: 10px 0;
•     display: flex;
•     align-items: center;
•   }
•
•   p span {
•     margin-left: 5px;
•     font-weight: bold;
•   }
•
•   button {
•     padding: 12px 24px;
•     font-size: 16px;
•     border: none;
•     border-radius: 8px;
•     color: white;
•     background-color: var(--primary);
•     cursor: pointer;
•     transition: background-color 0.2s ease;
•     margin-top: 15px;
•     margin-right: 12px;
•   }
•
•   button:hover {
•     background-color: #217dbb;
•   }
•
•   canvas {
•     margin-top: 10px;
•   }
•
•   .icon {
•     margin-right: 8px;
•   }
```

- Script.js

```
const client = mqtt.connect("ws://test.mosquitto.org:8080");
```

```
client.on("connect", () => {
  console.log("Terhubung ke broker MQTT");
  client.subscribe("IOT/Test1/temp");
  client.subscribe("IOT/Test1/hum");
});

client.on("message", (topic, message) => {
  const data = message.toString();
  const time = new Date().toLocaleTimeString();

  if (topic === "IOT/Test1/temp") {
    document.getElementById("temp").innerText = data;
    suhuTerakhir = parseFloat(data);
  } else if (topic === "IOT/Test1/hum") {
    document.getElementById("hum").innerText = data;
    kelembapanTerakhir = parseFloat(data);
  }

  // Update chart hanya jika keduanya sudah ada
  if (suhuTerakhir !== null && kelembapanTerakhir !== null) {
    updateChart(time, suhuTerakhir, kelembapanTerakhir);
    suhuTerakhir = null;
    kelembapanTerakhir = null;
  }
});

function turnOn() {
  client.publish("IOT/Test1/mqtt", "1");
  console.log("Kirim: 1 (LED ON)");
}

function turnOff() {
  client.publish("IOT/Test1/mqtt", "0");
  console.log("Kirim: 0 (LED OFF)");
}

// Variabel penampung nilai sementara
let suhuTerakhir = null;
let kelembapanTerakhir = null;

// Inisialisasi Chart.js
const ctx = document.getElementById("sensorChart").getContext("2d");

const sensorChart = new Chart(ctx, {
  type: "line",
  data: {
    labels: [],
    datasets: [
      {
        label: "Suhu (°C)",
```

```

        borderColor: "#e74c3c",
        backgroundColor: "rgba(231, 76, 60, 0.1)",
        data: [],
        tension: 0.4
    },
    {
        label: "Kelembapan (%)",
        borderColor: "#3498db",
        backgroundColor: "rgba(52, 152, 219, 0.1)",
        data: [],
        tension: 0.4
    }
]
},
options: {
    responsive: true,
    scales: {
        x: {
            title: { display: true, text: "Waktu" }
        },
        y: {
            title: { display: true, text: "Nilai Sensor" },
            min: -20,
            max: 50
        }
    }
}
});

// Fungsi update grafik suhu dan kelembapan
function updateChart(label, suhu, kelembapan) {
    sensorChart.data.labels.push(label);
    sensorChart.data.datasets[0].data.push(suhu);           // suhu
    sensorChart.data.datasets[1].data.push(kelembapan);   // kelembapan

    if (sensorChart.data.labels.length > 10) {
        sensorChart.data.labels.shift();
        sensorChart.data.datasets[0].data.shift();
        sensorChart.data.datasets[1].data.shift();
    }

    sensorChart.update();
}

```

