**LAPORAN PRAKTIKUM**

**Kontrol Tampilan Dot Matrix melalui MQTT dengan ESP32 dalam Lingkungan Simulasi Wokwi dan Node-RED**

****

Disusun Oleh:

Nama : Muhammad Ajra Zemima Muda

NIM : 2023903430022

Kelas : TRKJ 2.C

Jurusan : Teknologi Informasi dan Komputer

Program Studi : Teknologi Rekayasa Komputer Jaringan

Dosen Pembimbing : Attahariq,SST.,M.T



**JURUSAN TEKNOLOGI INFORMASI KOMPUTER**

**PRODI TEKNOLOGI REKAYASA KOMPUTER JARINGAN**

**POLITEKNIK NEGERI LHOKSEUMAWE**

**TAHUN AJARAN 2024-2025**

# **LEMBAR PENGESAHAN**

No Praktikum : 03/TIK/TRKJ-2C/Pengembangan Aplikasi Internet Of

Things

Judul : Laporan Praktikum Kontrol Tampilan Dot Matrix melalui MQTT dengan ESP32 dalam Lingkungan Simulasi Wokwi dan Node-RED

Nama : Muhammad Ajra Zemima Muda

NIM : 2023903430022

Kelas : TRKJ 2.C

Jurusan : Teknologi Informasi dan Komputer

Prodi : Teknologi Rekayasa Komputer Jaringan

Tanggal Praktikum : Rabu, 11 Juni 2025

Tanggal Penyerahan : Kamis, 12 Juni 2025

|  |
| --- |
| Buketrata, 12 Juni 2025 |
| Dosen Pembimbing, |
|  |
|  |
|  |
| **Attahariq, SST., M.T** |
| NIP. 19780724 200112 1 001 |

# **DAFTAR ISI**

[LEMBAR PENGESAHAN i](#_Toc200565320)

[DAFTAR ISI ii](#_Toc200565321)

[BAB I PENDAHULUAN 1](#_Toc200565322)

[A. Dasar Teori 1](#_Toc200565323)

[B. Tujuan 1](#_Toc200565324)

[C. Alat dan bahan 2](#_Toc200565325)

[D. Logika Kerja 2](#_Toc200565326)

[BAB II URAIAN PRAKTIKUM 2](#_Toc200565327)

[A. Langkah Kerja 2](#_Toc200565328)

[1. Rangkain Dot Matrik 3](#_Toc200565329)

[2. Pinout dot matrik 4](#_Toc200565330)

[3. Rancangan di node red 4](#_Toc200565331)

[4. Kode program untuk dot matrik nya 4](#_Toc200565332)

[B. Analisa program 7](#_Toc200565333)

[C. Hasil 14](#_Toc200565334)

[1. Hasil diwokwi 14](#_Toc200565335)

[2. Tampilan dari node red 14](#_Toc200565336)

[BAB III PENUTUP 14](#_Toc200565337)

[A. Kesimpulan 14](#_Toc200565338)

# **BAB I PENDAHULUAN**

1. **Dasar Teori**

Internet of Things (IoT) adalah konsep teknologi yang menghubungkan perangkat fisik ke internet untuk bertukar data secara real-time. Salah satu implementasinya adalah menampilkan informasi ke dot matrix LED, yang terdiri dari susunan LED dalam baris dan kolom untuk menampilkan teks atau simbol.

Dalam praktikum ini, dot matrix dikendalikan oleh ESP32 yang terhubung ke internet melalui WiFi. Data ditransmisikan menggunakan protokol MQTT, yang ringan dan efisien untuk komunikasi IoT. Node-RED digunakan sebagai antarmuka visual untuk mengatur dan mengirim pesan ke ESP32 melalui topik MQTT tertentu.

1. **Tujuan**

Tujuan dari praktikum ini adalah:

1. Memahami cara kerja komunikasi MQTT dalam sistem IoT.
2. Mengimplementasikan tampilan pesan dari Node-RED ke dot matrix melalui ESP32.
3. Menguji konektivitas antara ESP32, broker MQTT, dan Node-RED dalam menampilkan pesan secara real-time.
4. **Alat dan bahan**

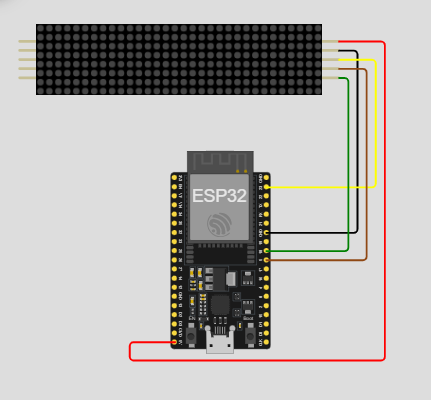
* ESP32 (simulasi via Wokwi)
* Dot Matrix 4-in-1 (MAX7219)
* Node-RED
* Broker MQTT (mqtt.esp32.my.id)
* Laptop/PC dan koneksi internet
* Software simulasi Wokwi

1. **Logika Kerja**

Praktikum ini diawali dengan menyambungkan ESP32 ke WiFi menggunakan SSID bawaan Wokwi. Setelah terhubung, ESP32 akan mengatur koneksi ke broker MQTT dan menunggu pesan dari topik tertentu, yaitu display/text.

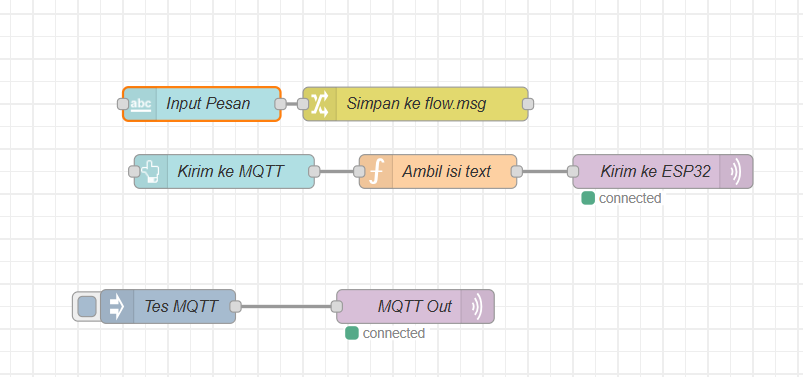
Jika pesan masuk, maka ESP32 akan menjalankan fungsi callback untuk membaca isi pesan dan kemudian menampilkannya ke LED dot matrix menggunakan efek animasi scroll ke kiri. Semua ini dikontrol oleh kode program yang menggunakan library WiFi, PubSubClient, dan MD\_Parola serta MD\_MAX72xx. Node-RED digunakan sebagai antarmuka untuk mengirim pesan ke topik MQTT secara visual dan real-time.

# **BAB II URAIAN PRAKTIKUM**

1. **Langkah Kerja**
2. **Rangkain Dot Matrik**
3. **Pinout dot matrik**

|  |  |  |
| --- | --- | --- |
| **Fungsi** | **MAX7219 Pin** | **ESP32 Pin** |
| **VCC** | **VCC** | **3.3V** |
| **GND** | **GND** | **GND** |
| **DIN** | **Data In** | **GPIO 23** |
| **CS / LOAD** | **CS** | **GPIO 5 (sesuai di #define CS\_PIN 5)** |
| **CLK** | **CLK** | **GPIO 18** |

1. **Rancangan di node red**



1. **Kode program untuk dot matrik nya**

#include <WiFi.h>

#include <PubSubClient.h>

#include <MD\_Parola.h>

#include <MD\_MAX72xx.h>

#include <SPI.h>

// Dot matrix configuration

#define HARDWARE\_TYPE MD\_MAX72XX::FC16\_HW

#define MAX\_DEVICES 4

#define CS\_PIN 5

MD\_Parola disp = MD\_Parola(HARDWARE\_TYPE, CS\_PIN, MAX\_DEVICES);

// WiFi credentials (gunakan WiFi di Wokwi)

const char\* ssid = "Wokwi-GUEST";

const char\* password = "";

// MQTT broker settings

const char\* mqtt\_server = "mqtt.esp32.my.id";

const int mqtt\_port = 7931;

const char\* mqtt\_user = "tamu";

const char\* mqtt\_pass = "tamu2024";

// Topic untuk menerima pesan teks

const char\* topic\_subscribe = "display/text";

WiFiClient espClient;

PubSubClient client(espClient);

String message = "";

void connectWiFi() {

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("\nWiFi Connected!");

}

void callback(char\* topic, byte\* payload, unsigned int length) {

  message = "";

  for (unsigned int i = 0; i < length; i++) {

    message += (char)payload[i];

  }

  Serial.print("Received message: ");

  Serial.println(message);

  disp.displayClear();

  disp.displayText(message.c\_str(), PA\_RIGHT, 100, 0, PA\_SCROLL\_LEFT, PA\_SCROLL\_LEFT);

  disp.displayReset();

}

void connectMQTT() {

  while (!client.connected()) {

    Serial.print("Connecting to MQTT...");

    if (client.connect("ESP32Client", mqtt\_user, mqtt\_pass)) {

      Serial.println("connected");

      client.subscribe(topic\_subscribe);

    } else {

      Serial.print("failed, rc=");

      Serial.print(client.state());

      Serial.println(" retrying in 5 seconds");

      delay(5000);

    }

  }

}

void setup() {

  Serial.begin(115200);

  disp.begin();

  disp.setIntensity(15);

  disp.displayClear();

  connectWiFi();

  client.setServer(mqtt\_server, mqtt\_port);

  client.setCallback(callback);

}

void loop() {

  if (!client.connected()) {

    connectMQTT();

  }

  client.loop();

  if (disp.displayAnimate()) {

    // loop scroll

    disp.displayReset();

  }

}

1. **Analisa program**

***#include <WiFi.h>***

***#include <PubSubClient.h>***

***#include <MD\_Parola.h>***

***#include <MD\_MAX72xx.h>***

***#include <SPI.h>***

Program diawali dengan mengimpor beberapa library penting. Library WiFi.h memungkinkan ESP32 terhubung ke jaringan WiFi, yang merupakan syarat utama agar perangkat dapat mengakses internet dan berkomunikasi dengan server. Untuk komunikasi dengan broker MQTT, digunakan PubSubClient.h, yang berfungsi sebagai pengelola pengiriman dan penerimaan pesan melalui protokol MQTT. Sementara itu, MD\_Parola, MD\_MAX72xx, dan SPI.h digunakan untuk mengendalikan dot matrix LED menggunakan antarmuka SPI. Library Parola sangat membantu dalam menampilkan teks yang dapat bergerak (scroll), berkedip, maupun efek animasi lainnya.

***#define HARDWARE\_TYPE MD\_MAX72XX::FC16\_HW***

***#define MAX\_DEVICES 4***

***#define CS\_PIN 5***

***MD\_Parola disp = MD\_Parola(HARDWARE\_TYPE, CS\_PIN, MAX\_DEVICES);***

Di bagian ini, kita mendefinisikan konfigurasi dari dot matrix LED. `HARDWARE\_TYPE` menyatakan jenis rangkaian yang digunakan, yaitu `FC16\_HW`, yang umum dipakai pada dot matrix murah. `MAX\_DEVICES` berarti ada 4 buah modul dot matrix yang akan digabung menjadi satu baris panjang. `CS\_PIN` adalah pin yang digunakan oleh ESP32 untuk mengatur komunikasi SPI dengan LED, dalam hal ini pin nomor 5. Semua informasi ini kemudian digunakan untuk membuat objek `disp`, yang akan digunakan untuk menampilkan teks ke LED nantinya.

***const char\* ssid = "Wokwi-GUEST";***

***const char\* password = "";***

Ini adalah kredensial WiFi. Karena kita menggunakan simulator Wokwi, maka nama jaringan (`ssid`) adalah `"Wokwi-GUEST"` dan tidak memerlukan password. Jika dijalankan di dunia nyata, kamu tinggal mengganti nilai ini dengan WiFi rumah atau hotspot.

***const char\* mqtt\_server = "mqtt.esp32.my.id";***

***const int mqtt\_port = 7931;***

***const char\* mqtt\_user = "tamu";***

***const char\* mqtt\_pass = "tamu2024";***

Bagian ini menyimpan informasi server MQTT yang akan digunakan untuk menerima pesan dari luar. Server ini bertugas seperti kantor pos: menerima pesan dari pengirim (misalnya dari HP kita), lalu meneruskannya ke ESP32. Diperlukan alamat server, port, dan login agar ESP32 bisa mengakses layanan tersebut.

***const char\* topic\_subscribe = "display/text";***

Topik ini seperti “saluran” dalam komunikasi MQTT. Kita berlangganan ke topik `"display/text"` agar setiap ada pesan baru yang dikirim ke topik ini, ESP32 bisa segera menerimanya dan menampilkannya.

***WiFiClient espClient;***

***PubSubClient client(espClient);***

Di sini kita membuat objek `espClient` untuk koneksi internet, dan `client` untuk koneksi MQTT. `client` menggunakan `espClient` sebagai jembatan ke dunia luar.

***String message = "";***

Variabel ini digunakan untuk menyimpan pesan teks yang diterima dari MQTT sebelum ditampilkan di dot matrix.

***void connectWiFi() {***

***WiFi.begin(ssid, password);***

***while (WiFi.status() != WL\_CONNECTED) {***

***delay(500);***

***Serial.print(".");***

***}***

***Serial.println("\nWiFi Connected!");***

***}***

Fungsi `connectWiFi()` digunakan untuk menyambungkan ESP32 ke jaringan WiFi. Fungsi ini akan terus mengecek apakah WiFi sudah terhubung. Kalau belum, maka akan mencetak titik-titik di Serial Monitor sebagai tanda masih mencoba. Kalau sudah terhubung, baru mencetak “WiFi Connected!”.

***void callback(char\* topic, byte\* payload, unsigned int length) {***

***message = "";***

***for (unsigned int i = 0; i < length; i++) {***

***message += (char)payload[i];***

***}***

***Serial.print("Received message: ");***

***Serial.println(message);***

***disp.displayClear();***

***disp.displayText(message.c\_str(), PA\_RIGHT, 100, 0, PA\_SCROLL\_LEFT, PA\_SCROLL\_LEFT);***

***disp.displayReset();***

***}***

Fungsi `callback()` ini akan dijalankan secara otomatis ketika ada \*\*pesan baru masuk dari MQTT\*\*. Pesan ini datang dalam bentuk array byte (`payload`). Setiap karakter akan dikonversi ke bentuk teks dan dimasukkan ke variabel `message`. Setelah pesan diterima, akan dicetak ke Serial Monitor, lalu ditampilkan ke LED matrix menggunakan animasi scroll dari kanan ke kiri (`PA\_SCROLL\_LEFT`).

***void connectMQTT() {***

***while (!client.connected()) {***

***Serial.print("Connecting to MQTT...");***

***if (client.connect("ESP32Client", mqtt\_user, mqtt\_pass)) {***

***Serial.println("connected");***

***client.subscribe(topic\_subscribe);***

***} else {***

***Serial.print("failed, rc=");***

***Serial.print(client.state());***

***Serial.println(" retrying in 5 seconds");***

***delay(5000);***

***}***

***}***

***}***

Fungsi `connectMQTT()` bertugas menyambungkan ESP32 ke server MQTT. Jika koneksi berhasil, ESP32 akan \*\*berlangganan ke topik `display/text`\*\*. Kalau gagal, program akan memberitahu status error dan mencoba lagi setiap 5 detik. Ini penting agar koneksi tetap stabil walaupun sempat terputus.

***void setup() {***

***Serial.begin(115200);***

***disp.begin();***

***disp.setIntensity(15);***

***disp.displayClear();***

***connectWiFi();***

***client.setServer(mqtt\_server, mqtt\_port);***

***client.setCallback(callback);***

***}***

Fungsi `setup()` adalah bagian yang pertama kali dijalankan saat ESP32 dinyalakan. Di sini, koneksi serial diaktifkan agar kita bisa melihat status di Serial Monitor. Lalu LED matrix diinisialisasi dan dikosongkan. Setelah itu, ESP32 disambungkan ke WiFi menggunakan fungsi `connectWiFi()`. Kemudian, kita mengatur alamat server MQTT dan fungsi `callback()` yang akan dijalankan ketika pesan masuk.

***void loop() {***

***if (!client.connected()) {***

***connectMQTT();***

***}***

***client.loop();***

***if (disp.displayAnimate()) {***

***disp.displayReset();***

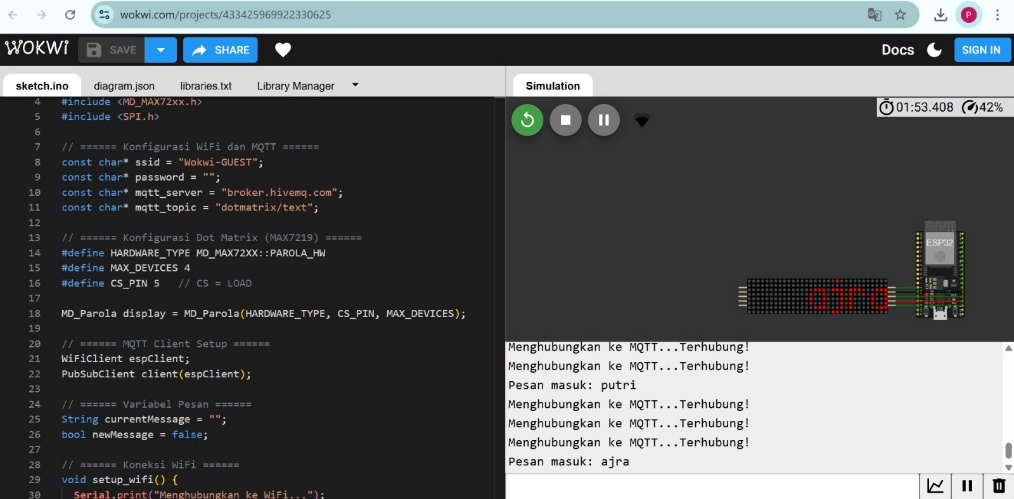
***}***

***}***

Fungsi `loop()` berjalan terus menerus. Setiap kali ESP32 kehilangan koneksi ke MQTT, ia akan mencoba menyambung lagi. Fungsi `client.loop()` digunakan untuk menjaga komunikasi dengan server MQTT tetap aktif dan bisa menerima pesan. Lalu, bagian `disp.displayAnimate()` digunakan untuk menjalankan efek animasi pada LED. Setelah satu kali scroll selesai, animasi akan di-reset agar bisa ditampilkan ulang.

Jadi kesimpulannya, program ini bertugas menerima teks dari internet melalui MQTT, lalu menampilkannya ke dot matrix LED dengan efek scroll. Semua proses dilakukan otomatis, mulai dari menyambung ke WiFi, menerima pesan, hingga menampilkannya. Kamu cukup kirim pesan ke topik MQTT yang sesuai, maka pesanmu akan muncul di LED.

1. **Hasil**



1. Hasil diwokwi
2. Tampilan dari node red



# **BAB III PENUTUP**

1. **Kesimpulan**

Sistem komunikasi antara Node-RED, MQTT, dan ESP32 berhasil menampilkan pesan ke dot matrix LED secara efektif. MQTT terbukti efisien sebagai penghubung dalam aplikasi IoT, dan penggunaan simulator Wokwi sangat membantu memahami konsep tanpa memerlukan perangkat keras langsung.