LAPORAN PRAKTIKUM

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

Disusun Oleh:

Nama	: L Hafidl Alkhair
NIM	: 2023903430060
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	: L Hafidl Alkhair
NIM	: 2023903430060
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

BAB I PENDAHULUAN

A. Dasar Teori

Internet of Things (IoT) adalah konsep teknologi yang memungkinkan perangkat fisik terhubung ke internet dan saling berkomunikasi untuk mengirimkan data secara real-time. Salah satu bentuk implementasi sederhana dari IoT adalah menampilkan data dari internet ke perangkat visual seperti **dot matrix LED**.

Dot matrix LED adalah kumpulan LED kecil yang tersusun dalam baris dan kolom, sehingga dapat digunakan untuk menampilkan karakter, angka, bahkan animasi. Dalam praktikum ini, dot matrix dikendalikan menggunakan **modul ESP32** yang terhubung ke internet melalui **WiFi**, dan menggunakan **protokol MQTT** sebagai penghubung untuk menerima pesan yang akan ditampilkan.

MQTT (Message Queuing Telemetry Transport) merupakan protokol komunikasi ringan yang banyak digunakan dalam proyek IoT karena efisien dan cepat. Data dikirim melalui topik tertentu dan dikendalikan melalui **Node-RED**, sebuah platform visual berbasis web yang sangat memudahkan dalam pengembangan dan simulasi aplikasi IoT.

B. 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.

C. 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

D. Logika Kerja

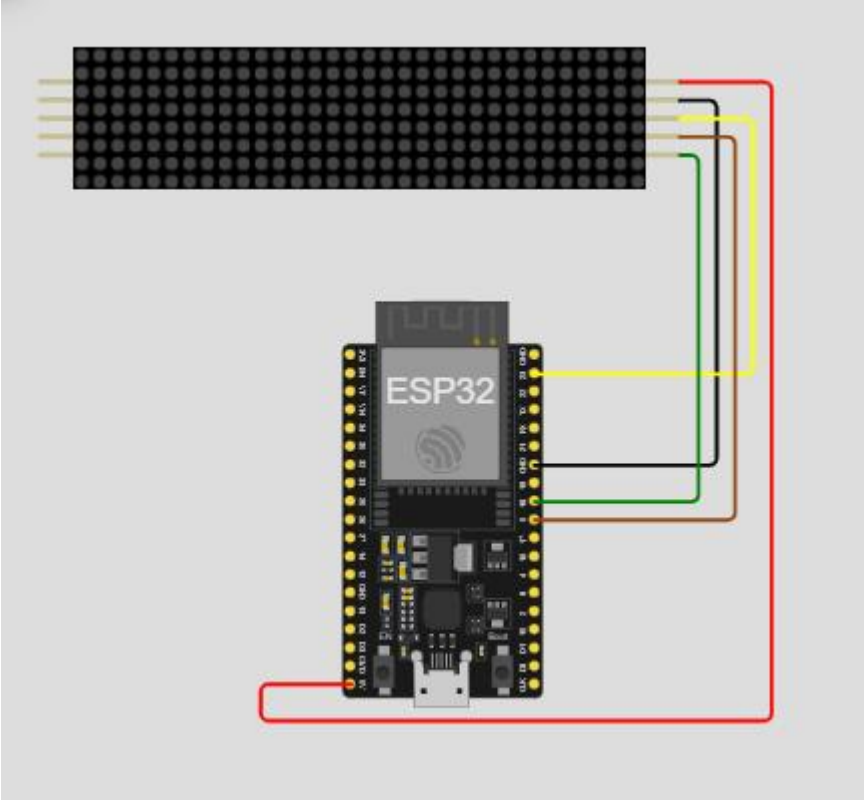
Praktikum ini diawali dengan menyambungkan ESP32 ke WiFi menggunakan SSID bawaan Wokwi. Setelah terhubung, ESP32 akan mengatur koneksi ke broker MQTT dan menunggu pesan dari topik tertentu, yaitu display/text.

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

A. Langkah Kerja

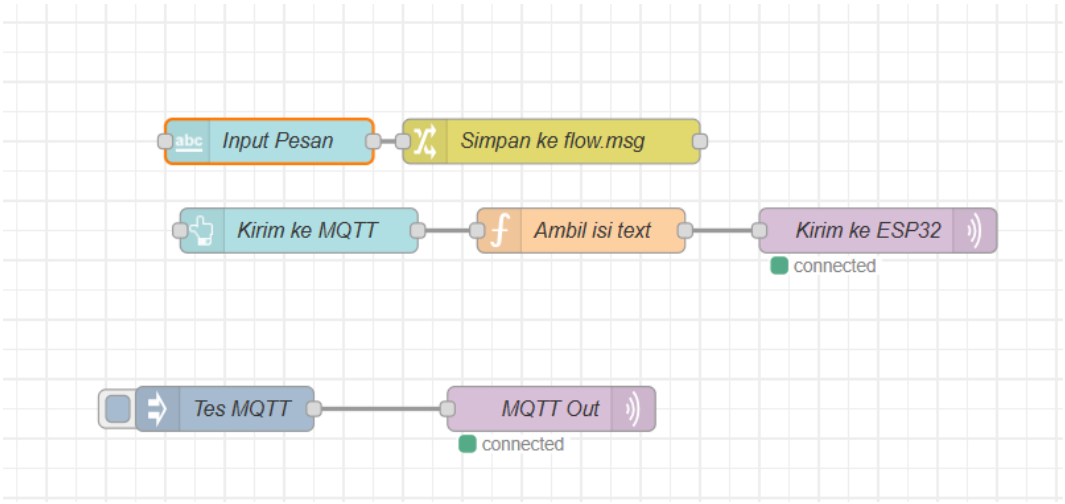
1. **Rangkain Dot Matrik**



2. 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

3. Rancangan di node red



4. 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();
  }
}
```

B. Analisa program

#include <WiFi.h>
#include <PubSubClient.h>
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

Program ini dimulai dengan mengimpor beberapa pustaka penting. `WiFi.h` digunakan agar ESP32 bisa terhubung ke jaringan WiFi. Tanpa ini, ESP32 tidak bisa mengakses internet atau berkomunikasi dengan server. `PubSubClient.h` digunakan untuk menangani komunikasi dengan broker MQTT, yaitu server yang akan menerima dan mengirim pesan-pesan dari dan ke ESP32. Sedangkan

`MD_Parola`, `MD_MAX72xx`, dan `SPI.h` digunakan untuk mengontrol dot matrix LED melalui komunikasi SPI. Library Parola sangat berguna untuk menampilkan teks yang bisa digerakkan (scrolling), berkedip, atau bergaya 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.

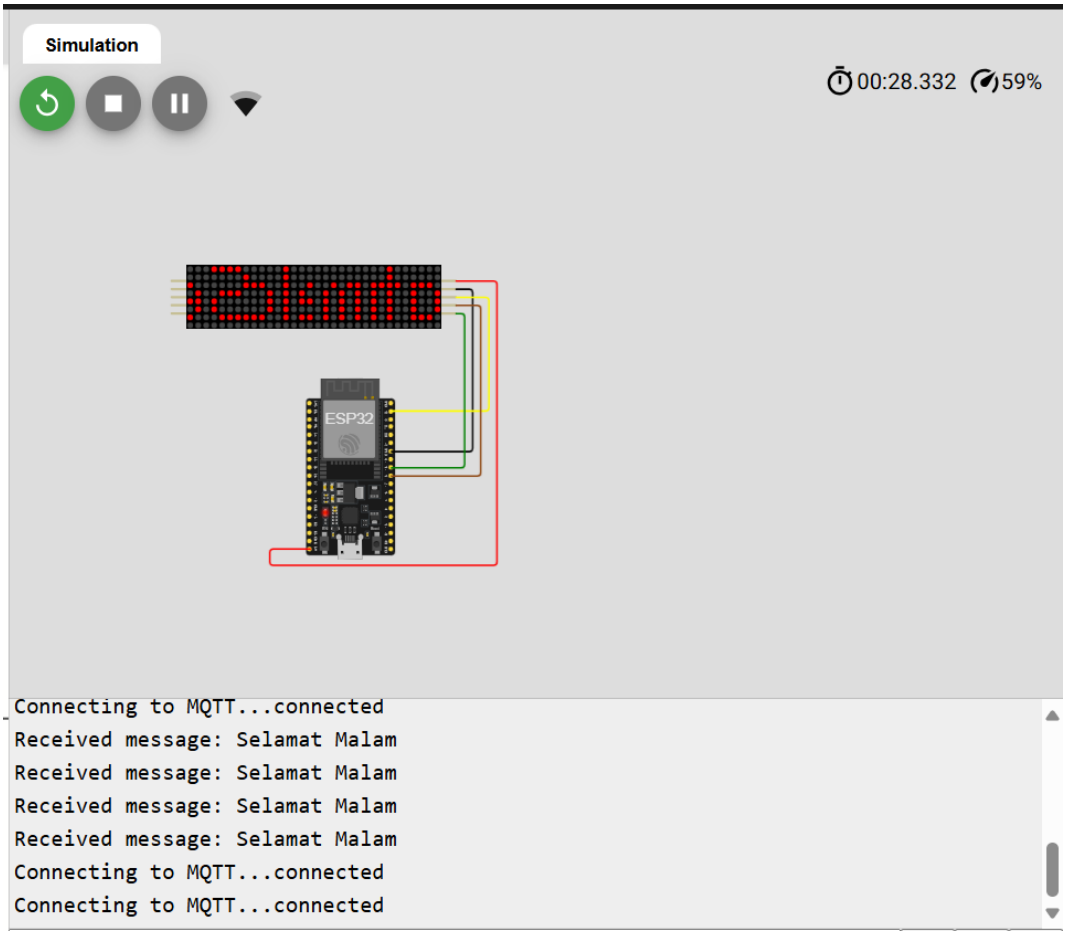
```
void loop() {
  if (!client.connected()) {
    connectMQTT();
  }
  client.loop();

  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.

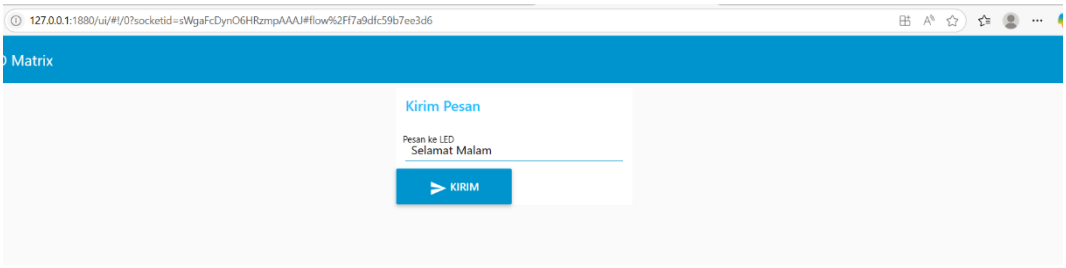
Jadi kesimpulannya, program ini bertugas menerima teks dari internet melalui MQTT, lalu menampilkannya ke dot matrix LED dengan efek scroll. Semua proses dilakukan otomatis, mulai dari menyambung ke WiFi, menerima pesan, hingga menampilkannya. Kamu cukup kirim pesan ke topik MQTT yang sesuai, maka pesanmu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

C. Hasil

1. Hasil diwokwi



2. Tampilan dari node red



BAB III PENUTUP

A. Kesimpulan

Berdasarkan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa sistem komunikasi antara Node-RED, MQTT, dan ESP32 dapat bekerja dengan baik dalam menampilkan pesan ke dot matrix LED. Proses dimulai dari pengiriman pesan di Node-RED, diteruskan oleh broker MQTT, lalu diterima oleh ESP32, dan akhirnya ditampilkan secara dinamis di dot matrix.

Praktikum ini membuktikan bahwa penggunaan MQTT sebagai penghubung antarperangkat sangat efisien untuk aplikasi Internet of Things. Selain itu, penggunaan Wokwi sebagai simulator sangat membantu dalam memahami konsep dan implementasi sistem IoT tanpa harus menggunakan perangkat keras secara langsung.