

**LAPORAN EKSPERIMENT MIKROKONTROLER UNTUK KONTROL
LED DAN KONTROL MOTOR DC BERBASIS MQTT
MENGGUNAKAN ESP32**

Dosen Pengampu:

Dr. Basuki Rahmat, S.Si, M.T



Disusun oleh:

Arif Nur Cahyo 23081010192

**MIKROKONTROLER A – 081
PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”
JAWA TIMUR
2025**

PROGRAM 1

1. Nama Program

Program Pengendalian Kecerahan LED (Fade Effect) dengan Komunikasi Serial

2. Kode Program

```
import time
import serial

PORT = "COM8"
BAUD = 115200
DELAY_BETWEEN_STEPS = 1.0 # 1 detik (bisa diubah sesuai kebutuhan)

ser = serial.Serial(PORT, BAUD, timeout=1)
time.sleep(2)

while ser.in_waiting:
    print(ser.readline().decode(errors="ignore").strip())

def send(cmd: str):
    ser.write((cmd.strip() + "\n").encode())

def read_lines(seconds=0.2):
    time.sleep(seconds)
    out = []
    while ser.in_waiting:
        out.append(ser.readline().decode(errors="ignore").strip())
    return out

# LED ON
send("LED 1")
print("\n".join(read_lines()))

print("Starting fade from 100% to 0%...")
for p in range(100, -1, -20):
    print(f"\nSetting power to {p}%")
    send(f"POWER {p}")

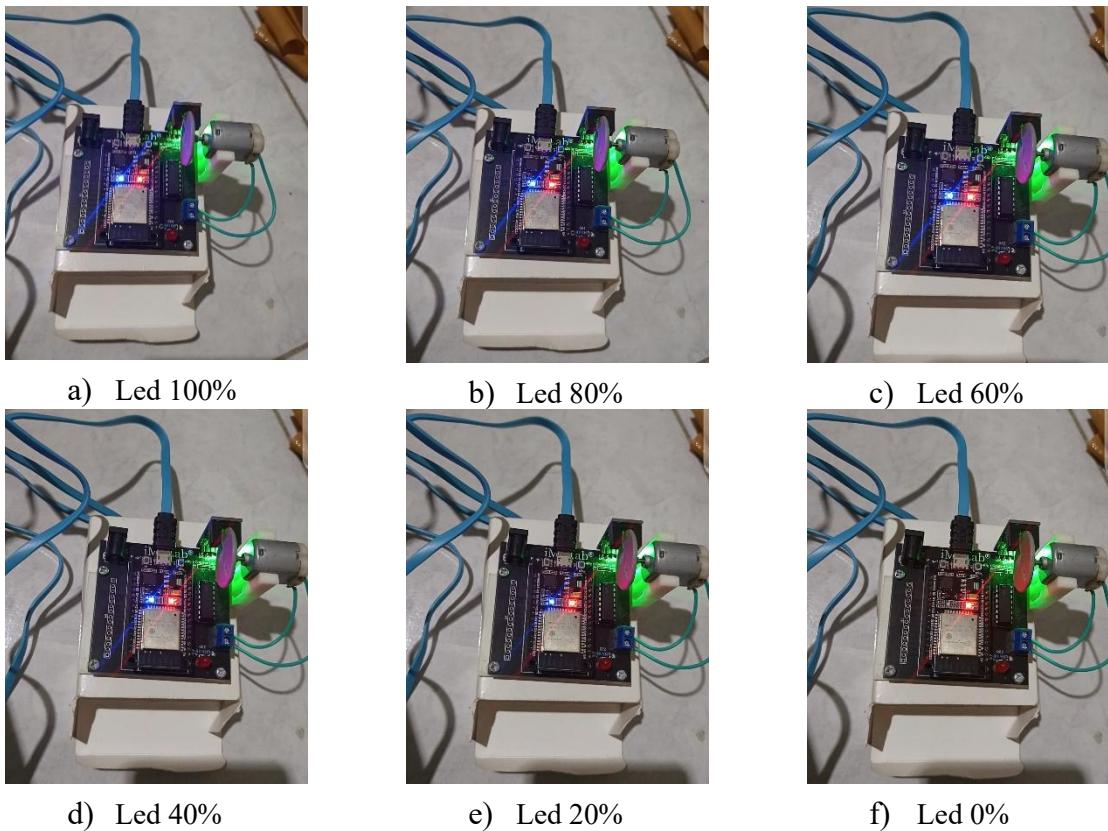
    # Jeda antar step
    if p > 0: # Tidak perlu jeda setelah step terakhir
        print(f" Waiting {DELAY_BETWEEN_STEPS} second(s)...")
        time.sleep(DELAY_BETWEEN_STEPS)

# LED OFF
print("\nTurning LED OFF...")
send("LED 0")

ser.close()
print("Disconnected")
```

3. Hasil Program

Berikut adalah hasil pengujian program yang menunjukkan perubahan kecerahan LED secara visual pada setiap tahap:



Gambar 1. Visualisasi Kecerahan LED pada Berbagai Tingkat Daya

Gambar 1 menunjukkan kondisi nyata LED pada berbagai tingkat kecerahan yang diatur oleh program. Pada setiap sub-gambar dapat diamati perbedaan intensitas cahaya LED mulai dari 100% (cahaya paling terang) hingga 0% (LED mati). Perubahan kecerahan ini sesuai dengan perintah yang dikirim melalui komunikasi serial, di mana nilai PWM (Pulse Width Modulation) diatur secara bertahap untuk menciptakan efek fade yang halus. Visual ini membuktikan bahwa program berhasil mengontrol perangkat hardware secara akurat.

Berikut adalah output teks yang dihasilkan oleh program saat dijalankan:

```

LED ON
Starting fade from 100% to 0%...

Setting power to 100%
Waiting 1.0 second(s)...

Setting power to 80%
Waiting 1.0 second(s)...

Setting power to 60%
Waiting 1.0 second(s)...

Setting power to 40%
Waiting 1.0 second(s)...

Setting power to 20%
Waiting 1.0 second(s)...

Setting power to 0%
Waiting 1.0 second(s)...

Turning LED OFF...
Disconnected

```

Gambar 2. Output Program pada Jupyter Notebook

Gambar 2 menampilkan hasil eksekusi program pada command prompt yang menunjukkan urutan perintah yang dikirim ke mikrokontroler. Output ini merekam setiap langkah proses fade, mulai dari menyalakan LED, mengatur kecerahan dari 100% hingga 0% dengan penurunan 20% per langkah, jeda waktu 1 detik antar perubahan, dan akhirnya mematikan LED. Output teks ini sesuai dengan logika program dan membuktikan bahwa komunikasi serial berjalan dengan baik tanpa adanya error atau gangguan koneksi.

4. Analisa Program

Program ini berhasil mengimplementasikan sistem kontrol kecerahan LED dengan efek fade melalui komunikasi serial antara komputer dan perangkat mikrokontroler. Secara struktur, program memiliki organisasi yang baik dengan pemisahan fungsi untuk pengiriman perintah dan pembacaan respons, memudahkan pemeliharaan dan pengembangan lebih lanjut. Implementasi fade dilakukan dengan pendekatan bertahap menggunakan loop yang mengurangi nilai kecerahan 20% setiap iterasi dengan jeda waktu 1 detik antar step, menciptakan transisi visual yang halus dan terkontrol.

Dari segi keandalan, program menunjukkan praktik pemrograman yang solid dengan penanganan koneksi serial yang tepat, termasuk inisialisasi port, penundaan untuk sinkronisasi, dan penutupan koneksi setelah selesai. Namun, program masih memiliki ruang pengembangan seperti penambahan exception handling untuk skenario koneksi gagal, konfigurasi parameter yang lebih fleksibel melalui input pengguna atau file eksternal, serta implementasi pola fade yang lebih variatif seperti fade in atau pattern tertentu.

Secara keseluruhan, program ini merupakan contoh efektif dari integrasi antara software dan hardware, mendemonstrasikan prinsip dasar Internet of Things (IoT) di mana komputer dapat mengendalikan perangkat fisik secara real-time. Aplikasi potensial mencakup sistem pencahayaan otomatis, instalasi seni digital, prototyping sistem kontrol industri, maupun sebagai alat pendidikan untuk pembelajaran komunikasi serial dan embedded systems. Keberhasilan program ini membuka peluang untuk pengembangan sistem kontrol yang lebih kompleks dengan sensor dan aktuator tambahan.

PROGRAM 2

1. Nama Program

Program Kontrol Motor DC via MQTT Menggunakan ESP32

2. Kode Program

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "SSID-WIFI";
const char* password = "PASSWORD-WIFI";
const char* mqtt_server = "broker.emqx.io";
const int mqtt_port = 1883;
const char* topic_control = "iot/motor/control";
const char* topic_speed = "iot/motor/speed";

int motor1Pin1 = 27;
int motor1Pin2 = 26;
int enable1Pin = 12;

const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;
int dutyCycle = 0;
bool motorState = false; // Status motor (ON/OFF)

WiFiClient espClient;
PubSubClient client(espClient);

void setup_wifi() {
    delay(10);
    Serial.begin(115200);
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("\nWiFi connected");
    Serial.print("IP: ");
    Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
    String message = "";
    for (int i = 0; i < length; i++) {
        message += (char)payload[i];
    }

    Serial.print("Topic: ");
    Serial.print(topic);
    Serial.print(" | Message: ");
    Serial.println(message);
```

```

// Kontrol ON/OFF motor
if (String(topic) == topic_control) {
    if (message == "1") {
        // MOTOR ON dengan kecepatan terakhir (dutyCycle)
        digitalWrite(motor1Pin1, HIGH);
        digitalWrite(motor1Pin2, LOW);
        ledcWrite(pwmChannel, dutyCycle);
        motorState = true;
        Serial.println("🔴 Motor ON");
        Serial.print("Kecepatan: ");
        Serial.print(dutyCycle);
        Serial.print(" (%)");
        Serial.print(map(dutyCycle, 0, 255, 0, 100));
        Serial.println("%)");
    }
    else if (message == "0") {
        // MOTOR OFF
        digitalWrite(motor1Pin1, LOW);
        digitalWrite(motor1Pin2, LOW);
        ledcWrite(pwmChannel, 0);
        motorState = false;
        Serial.println("🔴 Motor OFF");
    }
}

// Kontrol kecepatan
if (String(topic) == topic_speed) {
    int speedVal = message.toInt();

    // Cek apakah nilai valid (0-255)
    if (speedVal >= 0 && speedVal <= 255) {
        dutyCycle = speedVal;
        Serial.print("Kecepatan diatur ke: ");
        Serial.print(dutyCycle);
        Serial.print(" (%)");
        Serial.print(map(dutyCycle, 0, 255, 0, 100));
        Serial.println("%)");

        // Jika motor sedang ON, langsung terapkan kecepatan
        if (motorState) {
            ledcWrite(pwmChannel, dutyCycle);
            Serial.println("Motor sedang ON - kecepatan diterapkan");
        } else {
            Serial.println("Motor OFF - kecepatan disimpan untuk next ON");
        }
    } else {
        Serial.print("⚠️ Nilai kecepatan invalid: ");
        Serial.println(speedVal);
    }
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Connecting to MQTT...");
        if (client.connect("ESP32MotorClient")) {
            Serial.println("connected");
            client.subscribe(topic_control);
            client.subscribe(topic_speed);
        } else {

```

```

        Serial.print("Failed, rc=");
        Serial.print(client.state());
        Serial.println(" retrying in 5s...");
        delay(5000);
    }
}
}

void setup() {
    Serial.begin(115200);

    pinMode(motor1Pin1, OUTPUT);
    pinMode(motor1Pin2, OUTPUT);

    ledcSetup(pwmChannel, freq, resolution);
    ledcAttachPin(enable1Pin, pwmChannel);

    // Pastikan motor mati saat startup
    digitalWrite(motor1Pin1, LOW);
    digitalWrite(motor1Pin2, LOW);
    ledcWrite(pwmChannel, 0);

    setup_wifi();
    client.setServer(mqtt_server, mqtt_port);
    client.setCallback(callback);
}

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

```

3. Hasil Program

Berikut adalah hasil running program yang ditampilkan pada Serial Monitor:

```

Topic: iot/motor/control | Message: 1
💡 Motor ON
Kecepatan: 0 (0%)
Topic: iot/motor/speed | Message: 255
Kecepatan diatur ke: 255 (100%)
Motor sedang ON - kecepatan diterapkan

```

Gambar 3. Tampilan Motor ON dan Status Kecepatan pada Serial Monitor Arduino IDE

Gambar 3 menunjukkan dua pesan yang diterima. Pertama, pesan pada topik iot/motor/control dengan nilai 1 untuk menyalakan motor. Saat dinyalakan, motor berputar dengan kecepatan terakhir yang disimpan (awalnya 0). Kemudian, pesan kedua pada topik iot/motor/speed dengan nilai 255 (100%) diterima. Karena motor sudah ON, maka kecepatan 100% langsung diterapkan.

```
Topic: iot/motor/speed | Message: 232  
Kecepatan diatur ke: 232 (90%)  
Motor sedang ON - kecepatan diterapkan
```

Gambar 4. Tampilan Pengaturan Kecepatan Motor

Gambar 4 menunjukkan program menerima pesan pada topik iot/motor/speed dengan nilai 232. Nilai ini kemudian diubah menjadi 90% dari kecepatan maksimal (karena rentang 0-255). Karena motor dalam keadaan ON, maka kecepatan langsung diterapkan.

```
Topic: iot/motor/control | Message: 0  
● Motor OFF
```

Gambar 5. Tampilan Motor OFF pada Serial Monitor Arduino IDE

Gambar 5 menunjukkan program menerima pesan pada topik iot/motor/control dengan nilai 0. Ini adalah perintah untuk mematikan motor. Program kemudian mematikan motor dengan mengatur semua pin motor ke LOW dan menonaktifkan sinyal PWM.

4. Analisa Program

Program ini berhasil mengimplementasikan sistem kontrol motor DC berbasis IoT menggunakan protokol MQTT pada platform ESP32. Secara arsitektural, program mengadopsi pola publish-subscribe yang memungkinkan komunikasi asinkron antara pengendali dan perangkat, di mana ESP32 bertindak sebagai subscriber yang terus-menerus memantau dua topik utama: iot/motor/control untuk perintah ON/OFF dan iot/motor/speed untuk pengaturan kecepatan. Implementasi PWM dengan resolusi 8-bit (rentang 0-255) memberikan granularitas kontrol yang cukup halus untuk mengatur kecepatan motor, sementara mekanisme state management menjaga status motor (ON/OFF) secara internal sehingga kecepatan yang diatur saat motor mati akan diterapkan saat motor dinyalakan kembali.

Dari aspek keandalan, program menunjukkan struktur yang solid dengan pemisahan fungsi yang jelas: setup WiFi, callback MQTT, dan mekanisme reconnection. Program juga mengimplementasikan validasi input untuk nilai kecepatan (0-255) serta memberikan feedback informatif melalui Serial Monitor. Namun, terdapat beberapa area yang dapat dikembangkan lebih lanjut seperti penambahan autentikasi MQTT untuk keamanan, penyimpanan kredensial WiFi yang lebih aman (bukan hardcode), serta penambahan fitur kontrol arah putaran motor (maju/mundur) yang saat ini hanya mendukung satu arah.

Secara keseluruhan, program ini merupakan contoh nyata penerapan IoT dalam kontrol perangkat fisik, yang dapat diaplikasikan dalam berbagai skenario seperti otomasi rumah, kontrol industri, atau sistem robotika. Fleksibilitas protokol MQTT memungkinkan integrasi dengan berbagai platform dan antarmuka pengguna, sementara penggunaan broker publik EMQX (dengan potensi pergantian ke broker privat untuk keandalan produksi) menunjukkan kemudahan dalam prototyping sistem IoT. Program ini memberikan fondasi yang kuat untuk pengembangan sistem kontrol yang lebih kompleks dengan penambahan sensor, aktuator tambahan, atau integrasi dengan cloud platform.