

# **Laporan Final Project Mikrokontroller**

Disusun guna memenuhi Evaluasi Akhir Semester mata kuliah Mikrokontroller

Dosen Pengampu:

Dr. Basuki Rahmat, S.Si. MT



Disusun oleh:

Aaqilal Hafaafi (23081010269)

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

## Code Program File .INO

```
int ledPin = 2;
String inputString = "";
int motorSpeed = 0; // nilai PWM dari 0-255

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

const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;

void setup() {
    pinMode(ledPin, OUTPUT);
    Serial.begin(115200);

    pinMode(motor1Pin1, OUTPUT);
    pinMode(motor1Pin2, OUTPUT);
    pinMode(enable1Pin, OUTPUT);

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

    Serial.println("Ready...");
}

void loop() {
    while (Serial.available() > 0) {
        char c = Serial.read();

        if (c != '\n' && c != '\r') {
            inputString += c;
        } else if (inputString.length() > 0) {
            // --- Proses perintah ---
            if (inputString == "1") {
                digitalWrite(ledPin, HIGH);
                digitalWrite(motor1Pin1, HIGH);
                digitalWrite(motor1Pin2, LOW);
                ledcWrite(pwmChannel, motorSpeed);
                Serial.println("Motor ON");
            }
            else if (inputString == "0") {
                digitalWrite(ledPin, LOW);
                digitalWrite(motor1Pin1, LOW);
                digitalWrite(motor1Pin2, HIGH);
                ledcWrite(pwmChannel, 0);
                Serial.println("Motor OFF");
            }
        }
    }
}
```

```

        digitalWrite(motor1Pin2, LOW);
        ledcWrite(pwmChannel, 0);
        Serial.println("Motor OFF");
    }
    else {
        int speedVal = inputString.toInt();
        if (speedVal >= 0 && speedVal <= 255) {
            motorSpeed = speedVal;
            ledcWrite(pwmChannel, motorSpeed);
            Serial.print("PWM Speed: ");
            Serial.println(motorSpeed);
        }
        inputString = ""; // reset buffer
    }
}
}

```

## Penjelasan Code Program

Program di atas merupakan program untuk mengendalikan sebuah LED dan motor DC (melalui driver motor) menggunakan perintah yang dikirim lewat Serial (misalnya dari Serial Monitor di Arduino IDE). Di bagian awal, dideklarasikan beberapa variabel global. `int ledPin = 2;` menentukan bahwa LED terhubung ke pin digital 2. Variabel `inputString` digunakan sebagai buffer untuk menampung karakter-karakter yang diterima dari Serial hingga membentuk satu perintah utuh. `motorSpeed` menyimpan nilai PWM antara 0 sampai 255 yang akan menentukan kecepatan motor. Selanjutnya, terdapat tiga pin untuk motor: `motor1Pin1` dan `motor1Pin2` sebagai pin arah putaran motor, serta `enable1Pin` sebagai pin PWM untuk mengatur kecepatan motor. Konstanta `freq`, `pwmChannel`, dan `resolution` digunakan untuk konfigurasi sinyal PWM menggunakan fungsi `ledcSetup` yang merupakan fitur PWM pada ESP32 (frekuensi 30 kHz, kanal 0, resolusi 8 bit).

Pada fungsi `setup()`, terlebih dahulu pin LED diset sebagai OUTPUT dengan `pinMode(ledPin, OUTPUT);`. Komunikasi serial diinisialisasi dengan `Serial.begin(115200);` sehingga board dapat menerima dan mengirim data pada baudrate 115200. Pin-pin motor juga diset sebagai OUTPUT, karena akan digunakan untuk mengatur arah putaran dan mengaktifkan motor. Kemudian dilakukan konfigurasi PWM: `ledcSetup(pwmChannel, freq, resolution);` mendefinisikan kanal PWM dengan frekuensi dan resolusi yang ditentukan, dan `ledcAttachPin(enable1Pin, pwmChannel);` menghubungkan kanal PWM tersebut ke pin `enable1Pin`. Di akhir `setup()`, program menampilkan teks “Ready...” ke Serial sebagai tanda bahwa sistem sudah siap menerima perintah.

Bagian utama program terdapat di fungsi `loop()`. Di dalamnya, ada sebuah `while (Serial.available() > 0)` yang berarti program akan memproses data selama masih ada karakter yang masuk dari Serial. Setiap karakter yang dibaca dengan `Serial.read()` disimpan dalam variabel `c`. Jika karakter tersebut bukan `\n` (newline) dan bukan `\r` (carriage return), maka karakter tersebut

ditambahkan ke `inputString`. Tujuannya adalah mengumpulkan karakter menjadi satu perintah utuh sampai pengguna menekan Enter (yang biasanya mengirim `\n\r`).

Ketika karakter newline atau carriage return diterima dan `inputString` tidak kosong, program mulai memproses isi perintah. Jika `inputString` bernilai "1", maka program akan menyalakan LED (`digitalWrite(ledPin, HIGH);`), mengatur arah motor dengan `motor1Pin1 HIGH` dan `motor1Pin2 LOW` (artinya motor berputar ke satu arah tertentu), dan menuliskan nilai PWM saat ini (`motorSpeed`) ke kanal PWM dengan `ledcWrite(pwmChannel, motorSpeed);`. Lalu program mencetak teks "Motor ON" ke Serial sebagai feedback. Sebaliknya, jika `inputString` bernilai "0", program mematikan LED, mematikan kedua pin arah motor (LOW, sehingga motor berhenti), dan mengatur PWM menjadi 0 sehingga motor benar-benar tidak mendapat daya; kemudian mencetak "Motor OFF".

Jika perintah bukan "1" maupun "0", program menganggap bahwa perintah tersebut adalah nilai kecepatan PWM. `inputString.toInt()` mengubah string menjadi bilangan bulat dan disimpan di `speedVal`. Kemudian dicek apakah nilainya berada dalam rentang 0–255. Jika valid, nilai ini disimpan ke `motorSpeed` dan langsung dikirim ke `ledcWrite(pwmChannel, motorSpeed);` sehingga kecepatan motor berubah sesuai nilai yang dimasukkan. Program juga mencetak "PWM Speed: <nilai>" ke Serial untuk memberi tahu nilai PWM yang sedang digunakan. Pada akhir pemrosesan perintah (apapun perintahnya), `inputString` direset menjadi string kosong agar siap menampung perintah berikutnya. Dengan alur ini, pengguna bisa mengontrol motor dan LED hanya dengan mengirim "1" untuk menyalakan motor, "0" untuk mematikan motor, serta angka 0–255 untuk mengatur kecepatan motor melalui Serial.

## Code Program Python

```
import tkinter as tk
from tkinter import ttk, messagebox
import serial
import time
import threading

# Ganti COM7 sesuai port ESP32 kamu
PORT = 'COM7'
BAUD = 115200

try:
    esp32 = serial.Serial(PORT, BAUD, timeout=1)
    time.sleep(2)
except Exception as e:
    messagebox.showerror("Error", f"Gagal membuka port {PORT}\n{e}")
    esp32 = None

def kirim_data(data):
    """Kirim data ke ESP32 via serial"""
    if esp32 and esp32.is_open:
        esp32.write((str(data) + '\n').encode()) # kirim dengan newline
        print(f"Dikirim: {data}")
    else:
        messagebox.showerror("Error", "ESP32 tidak terhubung!")
```

```
def motor_on():
    kirim_data('1')
    status_label.config(text="Motor ON ✓", foreground="green")

def motor_off():
    kirim_data('0')
    status_label.config(text="Motor OFF ✘", foreground="red")

def ubah_kecepatan(val):
    nilai = int(float(val))
    kirim_data(nilai)
    speed_value_label.config(text=f"{nilai}")

def baca_serial():
    """Baca data dari ESP32"""
    while True:
        if esp32 and esp32.in_waiting > 0:
            data = esp32.readline().decode().strip()
            if data:
                print(f"[ESP32] {data}")

# Jalankan thread pembacaan serial
if esp32:
    threading.Thread(target=baca_serial, daemon=True).start()

# ----- GUI -----
root = tk.Tk()
root.title("Kontrol Motor ESP32")
root.geometry("350x300")
root.resizable(False, False)

frame = ttk.Frame(root, padding=20)
frame.pack(fill="both", expand=True)

# Judul
title_label = ttk.Label(frame, text="Kontrol Motor ESP32", font=("Segoe UI", 14, "bold"))
title_label.pack(pady=10)

# Tombol ON/OFF
btn_frame = ttk.Frame(frame)
btn_frame.pack(pady=10)

btn_on = ttk.Button(btn_frame, text="Motor ON", command=motor_on)
btn_on.grid(row=0, column=0, padx=10)

btn_off = ttk.Button(btn_frame, text="Motor OFF", command=motor_off)
btn_off.grid(row=0, column=1, padx=10)
```

```

# Slider Kecepatan
slider_label = ttk.Label(frame, text="Atur Kecepatan Motor (PWM 0-255):")
slider_label.pack(pady=(20, 5))

speed_slider = ttk.Scale(frame, from_=0, to=255, orient="horizontal",
command=ubah_kecepatan)
speed_slider.pack(fill="x", padx=10)

speed_value_label = ttk.Label(frame, text="0", font=("Segoe UI", 12))
speed_value_label.pack()

# Status label
status_label = ttk.Label(frame, text="Motor OFF 🔍", font=("Segoe UI", 12),
foreground="red")
status_label.pack(pady=15)

# Tombol keluar
btn_exit = ttk.Button(frame, text="Keluar", command=root.destroy)
btn_exit.pack(pady=10)

root.mainloop()

# Tutup serial saat keluar
if esp32 and esp32.is_open:
    esp32.close()

```

## Penjelasan Code Program

Program ini dibuat untuk membangun antarmuka grafis (GUI) di komputer menggunakan Tkinter yang berfungsi mengontrol motor yang terhubung ke ESP32 melalui komunikasi serial. Di bagian atas, program mengimpor beberapa modul penting: tkinter dan ttk untuk membuat tampilan GUI, messagebox untuk menampilkan pesan error, serial untuk komunikasi dengan ESP32, time untuk delay, serta threading untuk menjalankan pembacaan data serial di thread terpisah. Variabel PORT dan BAUD menentukan port serial yang digunakan (di sini COM7) dan baudrate 115200, yang harus sama dengan pengaturan Serial.begin(115200); di program ESP32. Blok try-except mencoba membuka koneksi serial ke ESP32, dan jika gagal akan menampilkan pesan error serta mengatur esp32 = None agar program tahu bahwa koneksi tidak tersedia.

Fungsi kirim\_data(data) berfungsi sebagai utilitas untuk mengirim perintah ke ESP32. Fungsi ini mengecek terlebih dahulu apakah objek esp32 ada dan portnya sedang terbuka (esp32.is\_open). Jika ya, maka data dikonversi menjadi string, ditambahkan karakter newline \n, lalu di-encode ke bentuk bytes dan dikirim melalui esp32.write(). Data yang dikirim juga dicetak ke console (terminal) sebagai log. Jika koneksi serial tidak tersedia, program menampilkan messagebox.showerror yang menginformasikan bahwa ESP32 tidak terhubung. Di atas fungsi ini bergantung beberapa fungsi lain: motor\_on() mengirimkan karakter '1' ke ESP32 untuk menyalakan motor dan memperbarui label status menjadi "Motor ON ✅" dengan warna hijau; sedangkan motor\_off() mengirim '0' untuk mematikan motor dan mengubah status menjadi merah "Motor OFF 🔍".

Fungsi ubah\_kecepatan(val) dipanggil setiap kali posisi slider di GUI berubah. Parameter

val dikirim dalam bentuk string float oleh Tkinter, sehingga dikonversi dulu menjadi int(float(val)). Nilai tersebut dikirim ke ESP32 sebagai perintah PWM (0–255) menggunakan kirim\_data(nilai). Label speed\_value\_label juga diperbarui untuk menampilkan angka terbaru sehingga pengguna tahu nilai PWM yang sedang dikirim. Dengan demikian, GUI dapat mengatur kecepatan motor secara real-time hanya dengan menggeser slider.

Selanjutnya ada fungsi baca\_serial() yang bertugas membaca data balikan dari ESP32. Fungsi ini berjalan dalam loop while True, sehingga terus memantau port serial. Di dalamnya, dicek apakah ada data yang menunggu (esp32.in\_waiting > 0). Jika ada, satu baris data dibaca menggunakan esp32.readline(), di-decode dari bytes ke string, kemudian di-strip() untuk menghilangkan karakter newline. Bila string tidak kosong, data tersebut dicetak ke console dengan prefix [ESP32]. Fungsi ini kemudian dijalankan di thread terpisah menggunakan threading.Thread(target=baca\_serial, daemon=True).start(). Dengan menjalankan pembacaan serial di thread lain, GUI tetap responsif dan tidak “freeze” saat menunggu data dari ESP32.

Bagian berikutnya adalah pembuatan antarmuka Tkinter. root = tk.Tk() membuat jendela utama dengan judul “Kontrol Motor ESP32” dan ukuran 350x300 piksel. root.resizable(False, False) membuat ukuran jendela tidak bisa diubah. Kemudian dibuat sebuah Frame sebagai container utama dengan padding 20 piksel. Di dalam frame ini disusun berbagai widget GUI. Label judul title\_label ditampilkan di bagian atas dengan font lebih besar dan tebal. Di bawahnya, dibuat frame btn\_frame yang berisi dua tombol: btn\_on untuk menyalakan motor dan btn\_off untuk mematikan motor, masing-masing terhubung ke fungsi motor\_on dan motor\_off.

Untuk pengaturan kecepatan motor, program membuat label keterangan slider, kemudian ttk.Scale sebagai slider dengan rentang 0 hingga 255 dan orientasi horizontal. Setiap perubahan posisi slider akan memanggil fungsi ubah\_kecepatan. Nilai PWM yang sedang digunakan ditampilkan dalam speed\_value\_label. Di bawahnya, terdapat status\_label yang menampilkan status motor (ON/OFF) dengan warna teks yang berbeda (merah untuk OFF, hijau untuk ON). Akhirnya, ada tombol Keluar yang memanggil root.destroy untuk menutup jendela aplikasi.

Pemanggilan root.mainloop() akan menjalankan loop utama Tkinter sehingga GUI aktif dan dapat merespons interaksi pengguna. Setelah jendela ditutup, program mengecek kembali objek esp32; jika masih terbuka, port serial akan ditutup dengan esp32.close() untuk mengakhiri komunikasi dengan rapi. Secara keseluruhan, program ini membentuk sistem kontrol motor ESP32 berbasis GUI: pengguna dapat menyalakan/mematikan motor dan mengatur kecepatannya dari komputer, sementara program juga membaca dan menampilkan feedback dari ESP32 melalui terminal.

## Lampiran Foto Hasil Percobaan

