LAPORAN EKSPERIMEN KIT MIKROKONTROLER IMCLAB

OLEH :

Lukman Hakim (23081010094)

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN" JAWA TIMUR

2025

1. ON OFF MIKROKONTROLLER MENGGUNAKAN ANDROID

Eksperimen pertama dilakukan dengan menyalakan dan mematikan kit IMCLab menggunakan tombol ON/OFF. Pada eksperimen ini, kecepatan motor tidak dapat diatur secara manual, melainkan bekerja secara otomatis sesuai dengan ketentuan yang telah ditetapkan dalam kode program.

a. Kode Program

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

/* ================ WIFI ================ */
const char* ssid = "realme 10";
const char* password = "bdaazqqae";

/* ================ MQTT ================ */
const char* mqtt_server = "broker.hivemq.com";
const char* mqtt_topic  = "esp32/motor/control";

WiFiClient espClient;
PubSubClient client(espClient);

/* ================ PIN MOTOR ================ */
#define MOTOR_IN1 27
#define MOTOR_IN2 26
#define MOTOR_EN  12

/* ================ PWM ================ */
#define PWM_CHANNEL 0
#define PWM_FREQ    30000
#define PWM_RES     8
#define TARGET_SPEED 200   // 🔥 KECEPATAN OTOMATIS (0-255)

bool motorOn = false;
int currentSpeed = 0;

/* ================ WIFI ================ */
void setup_wifi() {
  Serial.print("Connecting to WiFi");
  WiFi.begin(ssid, password);

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

  Serial.println("\nWiFi Connected");
  Serial.println(WiFi.localIP());
}

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

  msg.trim();
  Serial.println("MQTT: " + msg);

  if (msg == "on") {
    motorOn = true;
    digitalWrite(MOTOR_IN1, LOW);
    digitalWrite(MOTOR_IN2, HIGH);
    Serial.println("Motor ON");
  }
  else if (msg == "off") {
    motorOn = false;
    currentSpeed = 0;
    digitalWrite(MOTOR_IN1, LOW);
    digitalWrite(MOTOR_IN2, LOW);
    ledcWrite(PWM_CHANNEL, 0);
    Serial.println("Motor OFF");
  }
}

/* ================ MQTT RECONNECT ================ */
void reconnect() {
  while (!client.connected()) {
    if (client.connect("ESP32MotorClient")) {
      client.subscribe(mqtt_topic);
      Serial.println("MQTT Connected");
    } else {
      delay(3000);
    }
  }
}

/* ================ SETUP ================ */
void setup() {
  Serial.begin(115200);

  pinMode(MOTOR_IN1, OUTPUT);
  pinMode(MOTOR_IN2, OUTPUT);
  pinMode(MOTOR_EN, OUTPUT);

  ledcSetup(PWM_CHANNEL, PWM_FREQ, PWM_RES);
  ledcAttachPin(MOTOR_EN, PWM_CHANNEL);

  setup_wifi();

  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

/* ================ LOOP ================ */
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  /* ======== LOGIKA OTOMATIS SPEED ======== */
  if (motorOn && currentSpeed < TARGET_SPEED) {
    currentSpeed++;
    ledcWrite(PWM_CHANNEL, currentSpeed);
    delay(10);  // soft start
  }
}
```

b. Tujuan Program

1. Menghubungkan ESP32 ke jaringan WiFi.
2. Mengintegrasikan ESP32 dengan broker MQTT untuk menerima perintah kontrol.
3. Mengendalikan motor DC menggunakan driver motor melalui sinyal PWM.
4. Mengimplementasikan sistem kecepatan otomatis tanpa pengaturan manual.
5. Menerapkan metode soft start untuk mengurangi hentakan awal motor.

c. Perangkat dan Teknologi yang Digunakan
    1. Perangkat Keras
        - ESP32
        - Driver motor DC
        - Motor DC
        - Sumber daya eksternal untuk motor
    2. Perangkat Lunak
        - Arduino IDE
        - Library WiFi.h
        - Library PubSubClient.h
        - Broker MQTT: broker.hivemq.com
d. Struktur Program
    1. Konfigurasi WiFi
    2. Konfigurasi MQTT
    3. Pengaturan pin motor
    4. Pengaturan PWM
    5. Fungsi koneksi dan callback
    6. Logika utama pada fungsi loop()
e. Implementasi

f. Analisis Kinerja Sistem
    1. Sistem hanya mengenal dua kondisi: ON dan OFF
    2. Tidak terdapat kontrol kecepatan manual
    3. Kecepatan motor sepenuhnya dikendalikan oleh program
    4. MQTT memungkinkan pengendalian jarak jauh secara real-time

2. PENGATURAN KECEPATAN MOTOR MENGGUNAKAN ANDROID BERBASIS ESP32

Eksperimen ini bertujuan untuk mengendalikan kecepatan motor DC pada kit IMCLab menggunakan mikrokontroler ESP32 yang terhubung ke jaringan WiFi dan dikontrol melalui aplikasi Android berbasis protokol MQTT. Pengendalian dilakukan melalui slider kecepatan dengan rentang nilai 0–255, tanpa menggunakan tombol ON/OFF. Nilai 0 menyebabkan motor berhenti, sedangkan nilai di atas 0 membuat motor berputar. Hasil pengujian menunjukkan bahwa motor mulai berputar stabil pada nilai kecepatan ≥ 140.

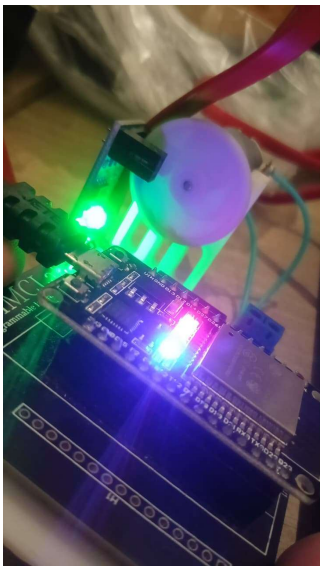
a. Kode Program

ESP32_MQTT_Motor_Control.ino.ino

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

/* ================ PIN ================ */
#define LED_WIFI 2        // LED onboard ESP32
#define MOTOR_IN1 27
#define MOTOR_IN2 26
#define MOTOR_EN  12

/* ================ WIFI ================ */
const char* ssid     = "realme 10";
const char* password = "bdaazqqae";

/* ================ MQTT ================ */
const char* mqtt_server = "broker.hivemq.com";
const char* mqtt_topic  = "iot/esp32/motor";

/* ================ PWM ================ */
const int pwmChannel = 0;
const int pwmFreq    = 30000;
const int pwmRes     = 8;

WiFiClient espClient;
PubSubClient client(espClient);

/* ================ WIFI SETUP ================ */
void setupWiFi() {
  WiFi.begin(ssid, password);
  Serial.print("Connecting WiFi");

  while (WiFi.status() != WL_CONNECTED) {
    digitalWrite(LED_WIFI, HIGH);
    delay(300);
    digitalWrite(LED_WIFI, LOW);
    delay(300);
    Serial.print(".");
  }

  digitalWrite(LED_WIFI, HIGH);
  Serial.println("\nWiFi Connected");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
}

/* ================ MQTT CALLBACK ================ */
void callback(char* topic, byte* payload, unsigned int length) {
  String msg = "";

  for (int i = 0; i < length; i++) {
    msg += (char)payload[i];
  }

  int speed = constrain(msg.toInt(), 0, 255);

  if (speed > 0) {
    digitalWrite(MOTOR_IN1, LOW);
    digitalWrite(MOTOR_IN2, HIGH);
    ledcWrite(pwmChannel, speed);

    Serial.print("Motor ON | Speed: ");
    Serial.println(speed);
  } else {
    digitalWrite(MOTOR_IN1, LOW);
    digitalWrite(MOTOR_IN2, LOW);
    ledcWrite(pwmChannel, 0);

    Serial.println("Motor OFF");
  }
}

/* ================ MQTT RECONNECT ================ */
void reconnectMQTT() {
  while (!client.connected()) {
    Serial.print("Connecting MQTT... ");

    String clientId = "ESP32-" + String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("Connected");
      client.subscribe(mqtt_topic);
      Serial.print("Subscribe: ");
      Serial.println(mqtt_topic);
    } else {
      Serial.print("Failed, retry...");
      delay(2000);
    }
  }
}

/* ================ SETUP ================ */
void setup() {
  Serial.begin(115200);

  pinMode(LED_WIFI, OUTPUT);
  pinMode(MOTOR_IN1, OUTPUT);
  pinMode(MOTOR_IN2, OUTPUT);
  pinMode(MOTOR_EN, OUTPUT);

  ledcSetup(pwmChannel, pwmFreq, pwmRes);
  ledcAttachPin(MOTOR_EN, pwmChannel);

  setupWiFi();

  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

/* ================ LOOP ================ */
void loop() {
  if (!client.connected()) {
    reconnectMQTT();
  }
  client.loop();
}
```

b. Tujuan Program
    1. Mengendalikan motor DC berbasis IoT menggunakan ESP32 dan Android.
    2. Mengatur kecepatan motor menggunakan nilai numerik 0–255.
    3. Menjadikan nilai kecepatan sebagai penentu kondisi ON dan OFF motor.
    4. Mengidentifikasi batas minimum kecepatan motor agar dapat berputar.
    5. Mengimplementasikan sistem kontrol yang realistis sesuai karakteristik motor DC.
c. Perangkat yang Digunakan
    1. Perangkat Keras
       ESP32
        - Driver motor DC
        - Motor DC
        - LED onboard ESP32 sebagai indikator WiFi
        - Catu daya motor
        - Smartphone Android
    2. Perangkat Lunak
        - Arduino IDE
        - Library WiFi.h
        - Library PubSubClient.h
        - Broker MQTT broker.hivemq.com
        - Aplikasi Android berbasis MQTT
d. Struktur Program
    1. Deklarasi library dan pin
    2. Konfigurasi WiFi
    3. Konfigurasi MQTT
    4. Konfigurasi PWM
    5. Fungsi koneksi WiFi
    6. Fungsi callback MQTT
    7. Fungsi reconnect MQTT
    8. Fungsi setup()
    9. Fungsi loop()

e. Implementasi



f. Analisis Kinerja Sistem

1. Motor DC memerlukan nilai PWM minimum sekitar 140 untuk mengatasi torsi awal.
2. Nilai PWM di bawah 140 tidak menghasilkan putaran yang stabil.
3. Sistem kontrol berbasis nilai kecepatan lebih fleksibel dibandingkan sistem ON/OFF.
4. Respons sistem terhadap perubahan nilai kecepatan bersifat real-time.
5. Faktor mekanik dan beban motor sangat mempengaruhi nilai minimum kecepatan.

3. KESIMPULAN

Berdasarkan hasil pengujian terhadap kedua program yang telah diimplementasikan, dapat disimpulkan bahwa masing-masing program memiliki karakteristik dan keunggulan yang berbeda dalam pengendalian motor DC berbasis ESP32 dan Android.

Pada program pertama (ON/OFF Mikrokontroler Menggunakan Android), sistem pengendalian motor masih bersifat sederhana karena hanya mengenal dua kondisi utama, yaitu menyala dan mati. Kecepatan motor ditentukan secara otomatis di dalam kode program dan tidak dapat diubah secara langsung oleh pengguna. Pendekatan ini memiliki kelebihan dalam hal kemudahan implementasi dan kestabilan sistem, namun kurang fleksibel karena tidak memberikan kontrol kecepatan yang dinamis sesuai kebutuhan pengguna. Program ini lebih sesuai untuk aplikasi yang hanya membutuhkan fungsi dasar ON dan OFF motor.

Sementara itu, program kedua (Pengaturan Kecepatan Motor Menggunakan Android Berbasis ESP32) menunjukkan peningkatan signifikan dalam aspek fleksibilitas dan realisme sistem. Pengendalian motor dilakukan melalui slider kecepatan dengan rentang nilai 0–255, sehingga pengguna dapat mengatur kecepatan motor secara langsung dan real-time. Hasil pengujian menunjukkan bahwa motor DC memerlukan nilai PWM minimum sekitar 140 agar dapat berputar secara stabil, yang mencerminkan karakteristik torsi awal motor DC pada kondisi nyata. Dengan demikian, program ini tidak hanya memberikan kontrol yang lebih presisi, tetapi juga lebih sesuai untuk aplikasi IoT yang membutuhkan pengaturan kecepatan motor secara bertahap dan adaptif.

Secara keseluruhan, dapat disimpulkan bahwa program pertama lebih cocok untuk sistem kontrol sederhana berbasis ON/OFF, sedangkan program kedua lebih unggul untuk sistem kontrol motor yang membutuhkan pengaturan kecepatan variabel dan interaksi pengguna yang lebih tinggi. Program kedua juga memberikan pemahaman yang lebih baik mengenai karakteristik kerja motor DC, khususnya terkait kebutuhan nilai PWM minimum untuk menghasilkan putaran yang stabil.