Logbook 2 : Penerapan Kontrol Motor DC pada iMCLab

Nama : Albi Akhsanul hakim
NPM : 22081010194

Pada logbook kedua ini, saya mempelajari penerapan kontrol Motor DC menggunakan ESP32 dalam lingkungan iMCLab. Fokus utama dari praktik ini adalah bagaimana mengatur arah dan kecepatan motor DC melalui pengendalian sinyal digital dan PWM (Pulse Width Modulation).

Berikut adalah contoh kode yang digunakan:

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

// Pengaturan properti PWM
const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;
int dutyCycle = 200;

void setup() {
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);

  // Konfigurasi fungsi PWM
  ledcSetup(pwmChannel, freq, resolution);
  ledcAttachPin(enable1Pin, pwmChannel);

  Serial.begin(115200);
  Serial.print("Testing DC Motor...");
}

void loop() {
  // Gerak maju dengan kecepatan maksimum
  Serial.println("Moving Forward");
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, HIGH);
  delay(2000);

  // Berhenti
  Serial.println("Motor stopped");
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, LOW);
  delay(1000);

  // Gerak mundur dengan kecepatan maksimum
  Serial.println("Moving Backwards");
  digitalWrite(motor1Pin1, HIGH);
```

```
  digitalWrite(motor1Pin2, LOW);
  delay(2000);

  // Berhenti
  Serial.println("Motor stopped");
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, LOW);
  delay(1000);

  // Gerak maju dengan kecepatan meningkat secara bertahap
  digitalWrite(motor1Pin1, HIGH);
  digitalWrite(motor1Pin2, LOW);
  while (dutyCycle <= 255) {
    ledcWrite(pwmChannel, dutyCycle);
    Serial.print("Forward with duty cycle: ");
    Serial.println(dutyCycle);
    dutyCycle += 5;
    delay(500);
  }

  dutyCycle = 200;
}}
```

Analisis:
Kode di atas mengatur tiga pin utama: dua pin digital (motor1Pin1 dan motor1Pin2) untuk mengendalikan arah putaran motor, serta satu pin PWM (enable1Pin) untuk mengatur kecepatan motor. Pada bagian setup(), ketiga pin ini diinisialisasi sebagai output. Kemudian, konfigurasi PWM dilakukan dengan frekuensi 30 kHz dan resolusi 8-bit, serta dihubungkan ke enable1Pin menggunakan ledcAttachPin().

Dalam fungsi loop(), motor dijalankan dalam beberapa mode: pertama bergerak maju dengan kecepatan maksimum, kemudian berhenti, lalu bergerak mundur dengan kecepatan maksimum, berhenti lagi, dan terakhir bergerak maju dengan kecepatan yang meningkat secara bertahap menggunakan penyesuaian nilai duty cycle melalui fungsi ledcWrite().

Sepanjang proses, komunikasi serial melalui Serial.println() digunakan untuk mencetak status motor ke Serial Monitor. Ini sangat membantu untuk memantau dan melakukan debugging selama pengujian. Implementasi ini menunjukkan cara yang efektif dalam mengontrol arah dan kecepatan motor DC menggunakan kemampuan PWM pada ESP32.