Materi Dasar Pemrograman ESP32

Pelatihan IEEE 2025

25 Oktober 2025

Daftar Isi

1 Dasar-Dasar Pemrograman ESP32 **3**
- 1.1 Contoh Program 1: Blink LED Internal . . . 3
  - 1.1.1 Kode Program . . . 3
  - 1.1.2 Penjelasan Kode . . . 3

2 Pertemuan 2: Input & Output Digital **4**
- 2.1 Tujuan . . . 4
- 2.2 Komponen di Wokwi . . . 4
- 2.3 Contoh Program 2: LED ON saat Tombol Ditekan . . . 4
  - 2.3.1 Kode Program . . . 4
  - 2.3.2 Penjelasan Kode . . . 4

3 Analog Input (Sensor Potensiometer) **6**
- 3.1 Tujuan . . . 6
- 3.2 Komponen . . . 6
- 3.3 Kode: Membaca Potensiometer . . . 6
- 3.4 Penjelasan Kode . . . 6

4 PWM (Pulse Width Modulation) **7**
- 4.1 Mengatur Kecerahan LED . . . 7
  - 4.1.1 Kode Program . . . 7
  - 4.1.2 Penjelasan Kode . . . 7

5 Kontrol Servo dengan Potensiometer **8**
- 5.1 Kode Program . . . 8
- 5.2 Penjelasan Kode . . . 8

6 Komunikasi Serial **9**
- 6.1 Tujuan . . . 9
- 6.2 Contoh Kode: Perintah ON/OFF LED . . . 9
- 6.3 Penjelasan Kode . . . 9

7 PROJECT AKHIR: Kontrol Servo via Perintah Serial **11**
- 7.1 Tujuan . . . 11
- 7.2 Komponen Wokwi . . . 11
- 7.3 Kode Lengkap: Kontrol Servo via Serial . . . 11
- 7.4 Penjelasan Kode . . . 12

8 BONUS: DHT22 + OLED Display **13**
8.1 Kode Program 13
8.2 Penjelasan Kode 13

9 Rangkuman Konsep Penting **15**
9.1 Fungsi Dasar Arduino/ESP32 15
9.2 Komunikasi Serial 15
9.3 Tipe Data 15
9.4 Struktur Kontrol 15

1 Dasar-Dasar Pemrograman ESP32

1.1 Contoh Program 1: Blink LED Internal

1.1.1 Kode Program

```
void setup() {
  pinMode(2, OUTPUT);  // GPIO2 = LED internal di board ESP32
}

void loop() {
  digitalWrite(2, HIGH);  // Nyalakan LED
  delay(1000);            // Tunggu 1 detik
  digitalWrite(2, LOW);   // Matikan LED
  delay(1000);            // Tunggu 1 detik
}
```

Listing 1: Program Blink LED

1.1.2 Penjelasan Kode

- `void setup()`: Fungsi yang dijalankan sekali saat ESP32 pertama kali dinyalakan atau di-reset
- `pinMode(2, OUTPUT)`: Mengonfigurasi GPIO pin 2 sebagai output. Parameter pertama adalah nomor pin, parameter kedua adalah mode (OUTPUT/INPUT)
- `void loop()`: Fungsi yang dijalankan berulang-ulang setelah `setup()` selesai
- `digitalWrite(2, HIGH)`: Memberikan tegangan HIGH (3.3V) pada pin 2, membuat LED menyala
- `delay(1000)`: Menghentikan eksekusi program selama 1000 milidetik (1 detik)
- `digitalWrite(2, LOW)`: Memberikan tegangan LOW (0V) pada pin 2, membuat LED mati

Hasil

LED internal berkedip setiap 1 detik (nyala 1 detik, mati 1 detik).

2 Pertemuan 2: Input & Output Digital

2.1 Tujuan

- Menggunakan tombol (push button) sebagai input
- Mengendalikan LED dengan tombol
- Memahami konsep debouncing sederhana

2.2 Komponen di Wokwi

- ESP32
- 1 Push Button
- 1 LED
- 1 Resistor (220 Ω)

Catatan: Aktifkan internal pull-up agar tidak perlu resistor tambahan pada tombol.

2.3 Contoh Program 2: LED ON saat Tombol Ditekan

2.3.1 Kode Program

```
const int ledPin = 2;
const int buttonPin = 4;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP); // tombol aktif LOW
}

void loop() {
  int tombol = digitalRead(buttonPin); // baca status tombol
  if (tombol == LOW) { // ditekan
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Listing 2: Kontrol LED dengan Tombol

2.3.2 Penjelasan Kode

- `const int ledPin = 2`: Mendefinisikan konstanta untuk pin LED (tidak dapat diubah)
- `const int buttonPin = 4`: Mendefinisikan konstanta untuk pin tombol
- `pinMode(buttonPin, INPUT_PULLUP)`: Mengaktifkan resistor pull-up internal ESP32 (sekitar 45kΩ). Ini membuat pin dalam kondisi HIGH secara default

- `digitalRead(buttonPin)`: Membaca status digital pin tombol (HIGH atau LOW)
- `if (tombol == LOW)`: Karena menggunakan INPUT_PULLUP, tombol yang ditekan akan memberikan nilai LOW (ground)
- Logika: Jika tombol ditekan (LOW), LED menyala. Jika tidak ditekan (HIGH), LED mati

3 Analog Input (Sensor Potensiometer)

3.1 Tujuan

- Membaca nilai analog menggunakan `analogRead()`
- Menampilkan hasil ke Serial Monitor

3.2 Komponen

- ESP32
- Potensiometer (hubungkan ke pin 34 atau 35)
- LED (opsional)

3.3 Kode: Membaca Potensiometer

```
1  const int potPin = 34;
2
3  void setup() {
4    Serial.begin(115200); // buka komunikasi serial
5  }
6
7  void loop() {
8    int nilai = analogRead(potPin);
9    Serial.println(nilai);
10   delay(200);
11 }
```

Listing 3: Membaca Nilai Analog

3.4 Penjelasan Kode

- `Serial.begin(115200)`: Menginisialisasi komunikasi serial dengan baud rate 115200 bps (bit per second)
- `analogRead(potPin)`: Membaca nilai analog dari pin 34. ESP32 memiliki ADC (Analog to Digital Converter) 12-bit
- **Rentang nilai**: 0 - 4095 ($2^{12} = 4096$ level)
  - 0 = 0V
  - 4095 = 3.3V
- `Serial.println(nilai)`: Mengirim nilai ke Serial Monitor dengan baris baru
- `delay(200)`: Menunda 200ms agar output tidak terlalu cepat

Catatan

Nilai ADC ESP32: **0–4095** (0–3.3V). Dapat digunakan untuk mengatur kecepatan motor, kecerahan LED, dll.

4 PWM (Pulse Width Modulation)

4.1 Mengatur Kecerahan LED

4.1.1 Kode Program

```
int ledPin = 2; //deklarasi pin led1

void setup() {
  pinMode(ledPin, OUTPUT); //pin led dijadikan sebagai pin output
  Serial.begin(9600); //memulai komunikasi ke serial monitor
}

void loop() {
  analogWrite(ledPin, 0); //led mati
  delay(2000);
  analogWrite(ledPin, 50); //led nyala redup
  delay(2000);
  analogWrite(ledPin, 255); //led nyala terang
  delay(2000);
  analogWrite(ledPin, 50); //led nyala redup
  delay(2000); //PWM dari mati, nyala redup, terang, redup
}
```

Listing 4: Kontrol Kecerahan LED dengan PWM

4.1.2 Penjelasan Kode

- **PWM (Pulse Width Modulation)**: Teknik untuk mengontrol daya dengan mengubah duty cycle sinyal digital
- `analogWrite(pin, value)`: Menghasilkan sinyal PWM pada pin
  - **value = 0**: LED mati (0% duty cycle)
  - **value = 50**: LED redup (≈20% duty cycle)
  - **value = 255**: LED terang penuh (100% duty cycle)
- Nilai PWM: **0-255** (8-bit)
- ESP32 memiliki 16 channel PWM dengan resolusi hingga 16-bit
- Duty cycle = $\frac{\text{value}}{255} \times 100\%$

5 Kontrol Servo dengan Potensiometer

5.1 Kode Program

```
#include <ESP32Servo.h>

Servo myservo;
const int potPin = 34;

void setup() {
  myservo.attach(15); // pin servo di GPIO15
}

void loop() {
  int potValue = analogRead(potPin);
  int angle = map(potValue, 0, 4095, 0, 180);
  myservo.write(angle);
  delay(20);
}
```

Listing 5: Kontrol Servo Berdasarkan Potensiometer

5.2 Penjelasan Kode

- `#include <ESP32Servo.h>`: Mengimpor library untuk mengontrol servo motor
- `Servo myservo`: Membuat objek servo dengan nama myservo
- `myservo.attach(15)`: Menghubungkan objek servo ke pin GPIO 15
- `map(potValue, 0, 4095, 0, 180)`: Fungsi mapping untuk mengkonversi nilai
  - Input: 0-4095 (nilai ADC)
  - Output: 0-180 (sudut servo dalam derajat)
  - Formula:
    $\text{output} = \frac{(\text{input} - \text{inMin}) \times (\text{outMax} - \text{outMin})}{\text{inMax} - \text{inMin}} + \text{outMin}$
- `myservo.write(angle)`: Menggerakkan servo ke sudut tertentu
- `delay(20)`: Delay singkat untuk stabilisasi servo

Catatan

Servo di Wokwi membutuhkan supply 5V dan sinyal pada pin yang mendukung PWM.

6 Komunikasi Serial

6.1 Tujuan

- Memahami cara berkomunikasi dengan PC via Serial Monitor
- Dapat menerima perintah teks dan mengontrol perangkat (LED/servo)

6.2 Contoh Kode: Perintah ON/OFF LED

```
const int ledPin = 2;
String input;

void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
  Serial.println("Ketik ON atau OFF:");
}

void loop() {
  if (Serial.available()) {
    input = Serial.readStringUntil('\n');
    input.trim();

    if (input == "ON") {
      digitalWrite(ledPin, HIGH);
      Serial.println("LED ON");
    } else if (input == "OFF") {
      digitalWrite(ledPin, LOW);
      Serial.println("LED OFF");
    } else {
      Serial.println("Perintah tidak dikenal");
    }
  }
}
```

Listing 6: Kontrol LED via Serial

6.3 Penjelasan Kode

- `String input`: Variabel untuk menyimpan string input dari serial
- `Serial.available()`: Mengecek apakah ada data yang tersedia di buffer serial (return true jika ada)
- `Serial.readStringUntil('\n')`: Membaca karakter sampai menemukan newline
- `input.trim()`: Menghapus whitespace (spasi, tab, newline) di awal dan akhir string
- `if (input == "ON")`: Membandingkan string input dengan "ON"
- **Flow**:
  1. Cek apakah ada data serial
  2. Baca string sampai enter

3. Bersihkan spasi
4. Eksekusi perintah sesuai input

7 PROJECT AKHIR: Kontrol Servo via Perintah Serial

7.1 Tujuan

Membuat sistem yang dapat menerima perintah dari Serial Monitor untuk mengontrol posisi servo (simulasi motor DC).

7.2 Komponen Wokwi

- ESP32 DevKit v1
- Servo Motor
- LED indikator (opsional)

Hubungan pin:

- Servo → pin 15
- LED → pin 2 (opsional)

7.3 Kode Lengkap: Kontrol Servo via Serial

```
#include <Servo.h>

Servo motor;
String command;  // perintah dari serial
int angle = 90;  // posisi awal

void setup() {
  Serial.begin(115200);
  motor.attach(15);
  motor.write(angle);
  Serial.println("Ketik perintah: LEFT, RIGHT, CENTER, atau ANGLE <nilai
   >");
}

void loop() {
  if (Serial.available()) {
    command = Serial.readStringUntil('\n');
    command.trim();

    if (command == "LEFT") {
      angle = 0;
    } else if (command == "RIGHT") {
      angle = 180;
    } else if (command == "CENTER") {
      angle = 90;
    } else if (command.startsWith("ANGLE")) {
      int value = command.substring(6).toInt();
      if (value >= 0 && value <= 180) angle = value;
    } else {
      Serial.println("Perintah tidak dikenal");
      return;
```

```
    }

    motor.write(angle);
    Serial.print("Servo di posisi: ");
    Serial.println(angle);
  }
}
```

Listing 7: Project: Kontrol Servo dengan Serial

7.4 Penjelasan Kode

- `motor.write(angle)`: Menggerakkan servo ke posisi awal (90°)
- `command.startsWith("ANGLE")`: Mengecek apakah string dimulai dengan "ANGLE"
- `command.substring(6)`: Mengambil substring mulai dari karakter ke-6 (setelah "ANGLE ")
  - Contoh: "ANGLE 45" → substring(6) = "45"
- `.toInt()`: Mengkonversi string menjadi integer
- **Validasi**: `if (value >= 0 && value <= 180)` memastikan nilai sudut valid
- `return`: Keluar dari fungsi loop() jika perintah tidak dikenal

Perintah yang didukung:

- `LEFT` → Servo ke 0°
- `RIGHT` → Servo ke 180°
- `CENTER` → Servo ke 90°
- `ANGLE 45` → Servo ke 45° (atau nilai 0-180)

8 BONUS: DHT22 + OLED Display

8.1 Kode Program

```
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include "DHT.h"

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#define DHTPIN 15
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  dht.begin();
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
}

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("Sensor DHT22");
  display.print("Suhu: ");
  display.print(t);
  display.println(" C");
  display.print("Lembab: ");
  display.print(h);
  display.println(" %");
  display.display();

  delay(2000);
}
```

Listing 8: Sensor DHT22 dengan OLED Display

8.2 Penjelasan Kode

- `#include <Wire.h>`: Library untuk komunikasi I^2C
- `#include <Adafruit_SSD1306.h>`: Library untuk OLED display SSD1306
- `#include "DHT.h"`: Library untuk sensor DHT22
- `#define SCREEN_WIDTH 128`: Mendefinisikan lebar layar OLED (128 pixel)
- `#define SCREEN_HEIGHT 64`: Mendefinisikan tinggi layar OLED (64 pixel)

- `Adafruit_SSD1306 display(...)`: Membuat objek display
  - `&Wire`: Pointer ke objek I^2C
  - `-1`: Reset pin (tidak digunakan)
- `#define DHTTYPE DHT22`: Menentukan tipe sensor (DHT22/DHT11)
- `DHT dht(DHTPIN, DHTTYPE)`: Membuat objek sensor DHT
- `display.begin(SSD1306_SWITCHCAPVCC, 0x3C)`:
  - `SSD1306_SWITCHCAPVCC`: Mode power internal
  - `0x3C`: Alamat I^2C OLED (biasanya 0x3C atau 0x3D)
- `dht.readHumidity()`: Membaca kelembaban (float, satuan %)
- `dht.readTemperature()`: Membaca suhu (float, satuan °C)
- `display.clearDisplay()`: Membersihkan buffer display
- `display.setCursor(0, 0)`: Set posisi kursor (x=0, y=0)
- `display.println()`: Menampilkan teks dengan pindah baris
- `display.display()`: Menampilkan buffer ke layar OLED (wajib dipanggil!)

Catatan Penting

Wokwi otomatis mengenali alamat I^2C OLED (0x3C).

9 Rangkuman Konsep Penting

9.1 Fungsi Dasar Arduino/ESP32

- `pinMode()` - Set mode pin
- `digitalWrite()` - Tulis digital
- `digitalRead()` - Baca digital
- `analogWrite()` - Tulis PWM
- `analogRead()` - Baca analog
- `delay()` - Tunda waktu

9.2 Komunikasi Serial

- `Serial.begin()` - Inisialisasi
- `Serial.print()` - Kirim data
- `Serial.available()` - Cek data tersedia
- `Serial.readStringUntil()` - Baca string

9.3 Tipe Data

- `int` - Integer (-32768 to 32767)
- `float` - Bilangan desimal
- `String` - Teks
- `const` - Konstanta

9.4 Struktur Kontrol

- `if-else` - Percabangan
- `loop()` - Perulangan otomatis
- `for/while` - Perulangan manual

Dibuat untuk pembelajaran ESP32 di Wokwi Simulator
Pelatihan IEEE 2025