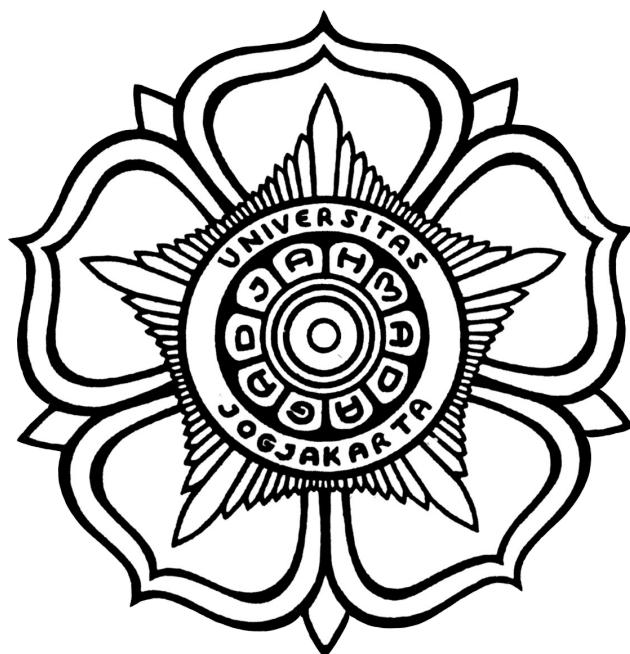


**LAPORAN PRAKTIKUM ELEKTRONIKA MESIN LISTRIK
DAN TEKNIK KENDALI**
“Rangkuman Mengenai ESP 32”

Dosen Pengampu: Irfan Bahiuddin, ST, M.Phil., Ph.D.



Disusun Oleh:

Kelompok 1

Catur Wardana (19/441194/SV/16546)

Santi Rahayu (19/441209/SV/16561)

Kelas: ARM 2

TEKNOLOGI REKAYASA MESIN

DEPARTEMEN TEKNIK MESIN SEKOLAH VOKASI

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2022

```
Latihan 1. LED_PUSH_Button
// set pin numbers
// the number of the pushbutton pin
const int buttonPin1 = 13;
const int buttonPin2 = 12;
const int buttonPin3 = 14;
const int buttonPin4 = 27;

// the number of the LED pin
const int ledPin1 = 26;
const int ledPin2 = 25;
const int ledPin3 = 33;
const int ledPin4 = 32;

// variable for storing the pushbutton status
int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
int buttonState4 = 0;

void setup() {
    Serial.begin(115200);
    // initialize the pushbutton pin as an input
    pinMode(buttonPin1, INPUT);
    pinMode(buttonPin2, INPUT);
    pinMode(buttonPin3, INPUT);
    pinMode(buttonPin4, INPUT);

    // initialize the LED pin as an output
    pinMode(ledPin1, OUTPUT);
    pinMode(ledPin2, OUTPUT);
    pinMode(ledPin3, OUTPUT);
    pinMode(ledPin4, OUTPUT);
}

void loop() {
    // read the state of the pushbutton value
    buttonState1 = digitalRead(buttonPin1);
    Serial.println(buttonState1);
    // check if the pushbutton is pressed.
    // if it is, the buttonState is HIGH
    if (buttonState1 == HIGH) {
        // turn LED on
        digitalWrite(ledPin1, HIGH);
    } else {
        // turn LED off
    }
}
```

```

    digitalWrite(ledPin1, LOW);
}

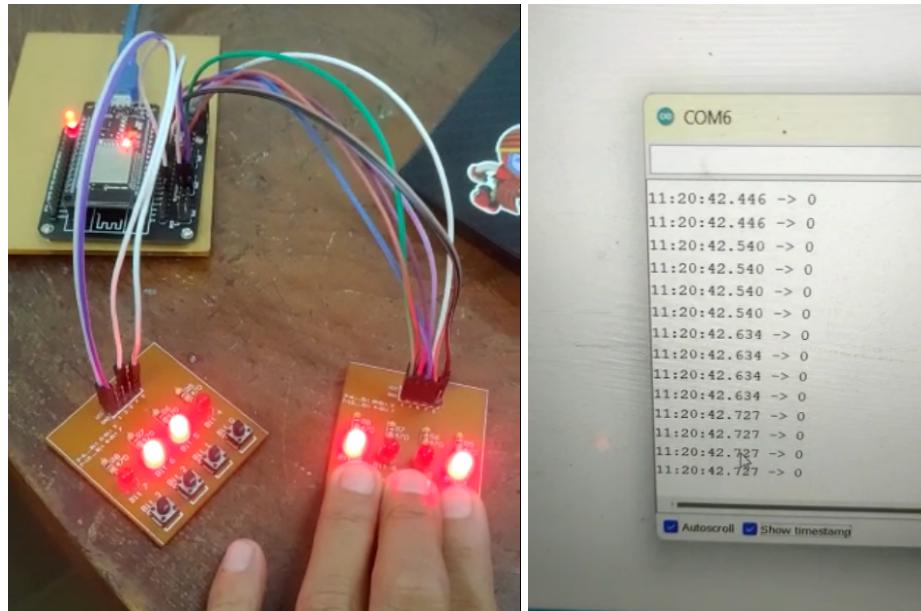
// read the state of the pushbutton value
buttonState2 = digitalRead(buttonPin2);
Serial.println(buttonState2);
// check if the pushbutton is pressed.
// if it is, the buttonState is HIGH
if (buttonState2 == HIGH) {
    // turn LED on
    digitalWrite(ledPin2, HIGH);
} else {
    // turn LED off
    digitalWrite(ledPin2, LOW);
}

// read the state of the pushbutton value
buttonState3 = digitalRead(buttonPin3);
Serial.println(buttonState3);
// check if the pushbutton is pressed.
// if it is, the buttonState is HIGH
if (buttonState3 == HIGH) {
    // turn LED on
    digitalWrite(ledPin3, HIGH);
} else {
    // turn LED off
    digitalWrite(ledPin3, LOW);
}

// read the state of the pushbutton value
buttonState4 = digitalRead(buttonPin4);
Serial.println(buttonState4);
// check if the pushbutton is pressed.
// if it is, the buttonState is HIGH
if (buttonState4 == HIGH) {
    // turn LED on
    digitalWrite(ledPin4, HIGH);
} else {
    // turn LED off
    digitalWrite(ledPin4, LOW);
}
delay(100);
}

```

Hasil :



Latihan 2. Blinking LED

```
/*
  CATUR - SANTI
  Blink LED DELAY 1 Second
*/

// ledPin refers to ESP32 GPIO 23
const int ledPin1 = 27;
const int ledPin2 = 26;
const int ledPin3 = 25;
const int ledPin4 = 33;

// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin ledPin as an output.
    Serial.begin(115200);
    pinMode(ledPin1, OUTPUT);
    pinMode(ledPin2, OUTPUT);
    pinMode(ledPin3, OUTPUT);
    pinMode(ledPin4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    //Turn On
    digitalWrite(ledPin1, HIGH); // turn the LED on (HIGH is the voltage level)
    Serial.println("LED1 is on");
    delay(1000); // wait for a second
    digitalWrite(ledPin2, HIGH); // turn the LED on (HIGH is the voltage level)
}
```

```

Serial.println("LED2 is on");
delay(1000);           // wait for a second
digitalWrite(ledPin3, HIGH); // turn the LED on (HIGH is the voltage level)
Serial.println("LED3 is on");
delay(1000);           // wait for a second
digitalWrite(ledPin4, HIGH); // turn the LED on (HIGH is the voltage level)
Serial.println("LED4 is on");
delay(1000);           // wait for a second

//Turn Off
digitalWrite(ledPin1, LOW); // turn the LED off by making the voltage LOW
Serial.println("LED1 is off");
delay(1000);           // wait for a second
digitalWrite(ledPin2, LOW); // turn the LED off by making the voltage LOW
Serial.println("LED2 is off");
delay(1000);           // wait for a second
digitalWrite(ledPin3, LOW); // turn the LED off by making the voltage LOW
Serial.println("LED3 is off");
delay(1000);           // wait for a second
digitalWrite(ledPin4, LOW); // turn the LED off by making the voltage LOW
Serial.println("LED4 is off");
delay(1000);           // wait for a second
}

```

Latihan 3. PWM_2LED

```

// the number of the LED pin
const int ledPin = 26; // 26 corresponds to GPIO16
const int ledPin2 = 25; // 15 corresponds to GPIO17

```

```

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

```

```

void setup(){
    // configure LED PWM functionalitites
    ledcSetup(ledChannel, freq, resolution);
}

```

```

    // attach the channel to the GPIO to be controlled
    ledcAttachPin(ledPin, ledChannel);
    ledcAttachPin(ledPin2, ledChannel);
}

```

```

void loop(){
    // increase the LED brightness
    for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
        // changing the LED brightness with PWM
}

```

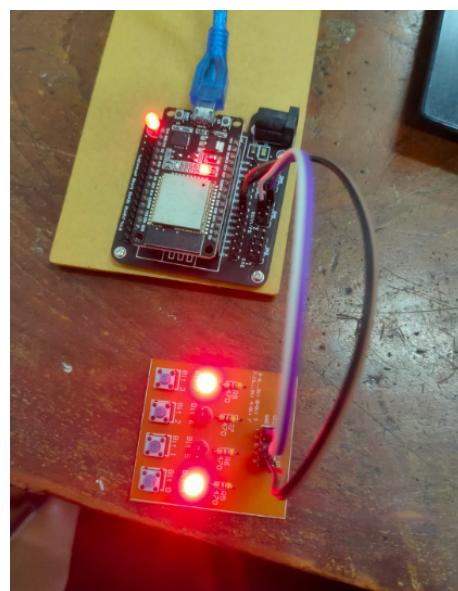
```

    ledcWrite(ledChannel, dutyCycle);
    delay(15);
}

// decrease the LED brightness
for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
}
}

```

Hasil :



Kedua LED akan redup dan terang dengan delay 15 milisecond

Latihan 4. Button_to_PWM

```

//Latihan Punya Catur, Santi
// the number of the LED pin
const int ledPin = 26; // 26 corresponds to GPIO26
const int ledPin2 = 25; // 15 corresponds to GPIO15

// the number of the Button
const int buttonPin1 = 13; //Button On
const int buttonPin2 = 12; //Butoon OFF

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

```

```

int buttonState1 = 0;
int buttonState2 = 0;

void setup(){

    Serial.begin(115200);
    // initialize the pushbutton pin as an input
    pinMode(buttonPin1, INPUT);
    pinMode(buttonPin2, INPUT);

    // initialize the LED pin as an output
    pinMode(ledPin, OUTPUT);
    pinMode(ledPin2, OUTPUT);

    digitalWrite(ledPin, HIGH);
    digitalWrite(ledPin2, HIGH);

    // configure LED PWM functionalitites
    ledcSetup(ledChannel, freq, resolution);

    // attach the channel to the GPIO to be controlled
    ledcAttachPin(ledPin, ledChannel);
    ledcAttachPin(ledPin2, ledChannel);
}

void loop(){
    buttonState1 = digitalRead(buttonPin1);
    Serial.println(buttonState1);
    // check if the pushbutton is pressed.
    // if it is, the buttonState is HIGH
    if (buttonState1 == LOW) {
        for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
            // changing the LED brightness with PWM
            ledcWrite(ledChannel, dutyCycle);\n
            delay(15);
        }
    }

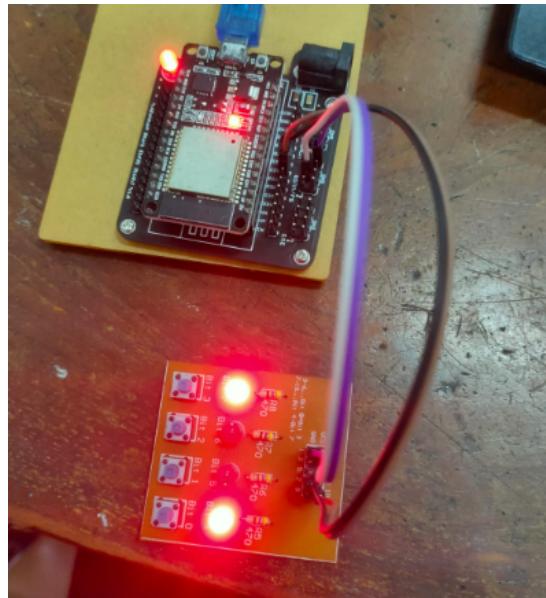
    buttonState2 = digitalRead(buttonPin2);
    Serial.println(buttonState2);
    // check if the pushbutton is pressed.
    // if it is, the buttonState is HIGH
    if (buttonState2 == LOW) {
        for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
            // changing the LED brightness with PWM
            ledcWrite(ledChannel, dutyCycle);
    }
}

```

```

    delay(15);
}
}
}

```



Kedua LED akan redup dan terang dengan delay 15 milisecond dengan menekan tombol dengan fungsi button

Latihan 5. Blinking_an_LED_with_millis

```

// Catur - Santi
const int ledPin = 26;      // the number of the LED pin

// Variables will change :
int ledState = LOW;        // ledState used to set the LED

unsigned long previousMillis = 0;    // will store last time LED was updated

// constants won't change :
const long interval = 1000;        // interval at which to blink (milliseconds)

void setup() {
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

```

```

// if the LED is off turn it on and vice-versa:
if (ledState == LOW) {
    ledState = HIGH;
} else {
    ledState = LOW;
}

// set the LED with the ledState of the variable:
digitalWrite(ledPin, ledState);
}
}

```

Latihan 6. Potensio Reader (ADC)

```

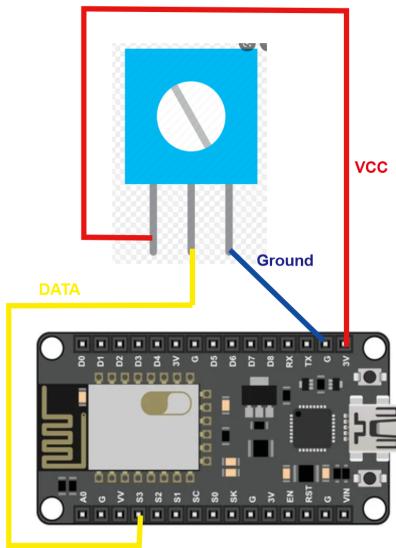
//Catur Santi
// Potentiometer is connected to GPIO 34 (Analog ADC1_CH6)
const int potPin = 34;

// variable for storing the potentiometer value
int potValue = 0;

void setup() {
    Serial.begin(115200);
    delay(1000);
}

void loop() {
    // Reading potentiometer value
    potValue = analogRead(potPin);
    Serial.println(potValue);
    delay(500);
}

```



Potensiometer Reader | Arduino 1.8.19 (Windows Store 1.8.57.0)

```

File Edit Sketch Tools Help
Potensiometer Reader
// Potentiometer is connected to GPIO 34 (Analog ADC1_CH6)
const int potPin = 34;

// variable for storing the potentiometer value
int potValue = 0;

void setup() {
  Serial.begin(115200);
  delay(1000);
}

void loop() {
  // Reading potentiometer value
  potValue = analogRead(potPin);
  Serial.println(potValue);
  delay(500);
}

```

COM6

```

08:11:11.174 -> 0
08:11:11.632 -> 0
08:11:12.109 -> 0
08:11:12.577 -> 0
08:11:13.102 -> 107
08:11:13.623 -> 0
08:11:14.143 -> 0
08:11:14.605 -> 618
08:11:15.114 -> 632
08:11:15.622 -> 608
08:11:16.131 -> 602
08:11:16.634 -> 602
08:11:17.140 -> 609
08:11:17.602 -> 619

```

BAB 1. Spesifikasi

Spesifikasi chip ESP32 :

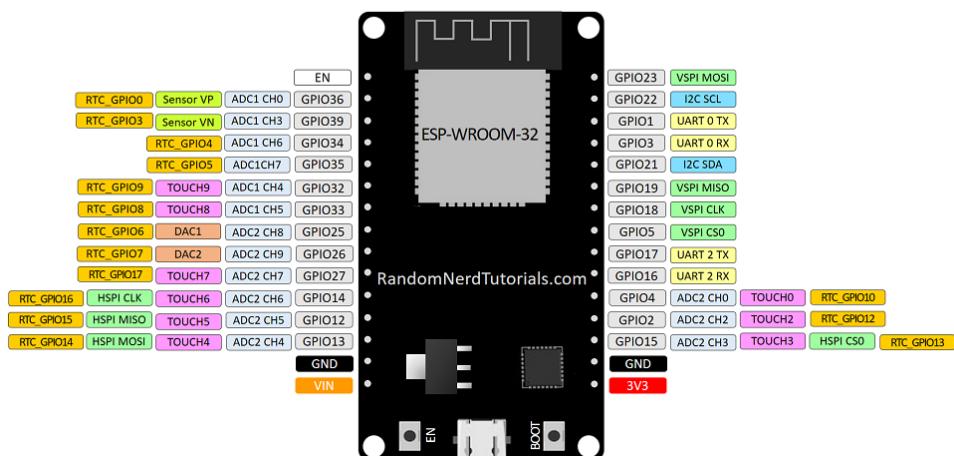
- ESP32 memiliki dual core processor, ini berarti memiliki 2 prosesor.
- Memiliki Wifi dan bluetooth built-in, yang berarti kita dapat melakukan pekerjaan kontrol yang membutuhkan bluetooth dan Wifi secara langsung tanpa perlu menambah modul lain.
- ESP32 dapat menjalankan program 32 bit.
- Frekuensi clock bisa mencapai 240 MHz dan memiliki RAM 512 kB.
- Produk modul yang menggunakan chip ESP32 ini memiliki 30 atau 36 pin.
- Memiliki berbagai macam fungsi yang tersedia, seperti: tombol tekan, ADC, DAC, UART, SPI, I2C dan banyak lagi.
- Terdapat sensor efek hall built-in dan sensor suhu built-in.

BAB 2. PinOut Beserta Fungsinya

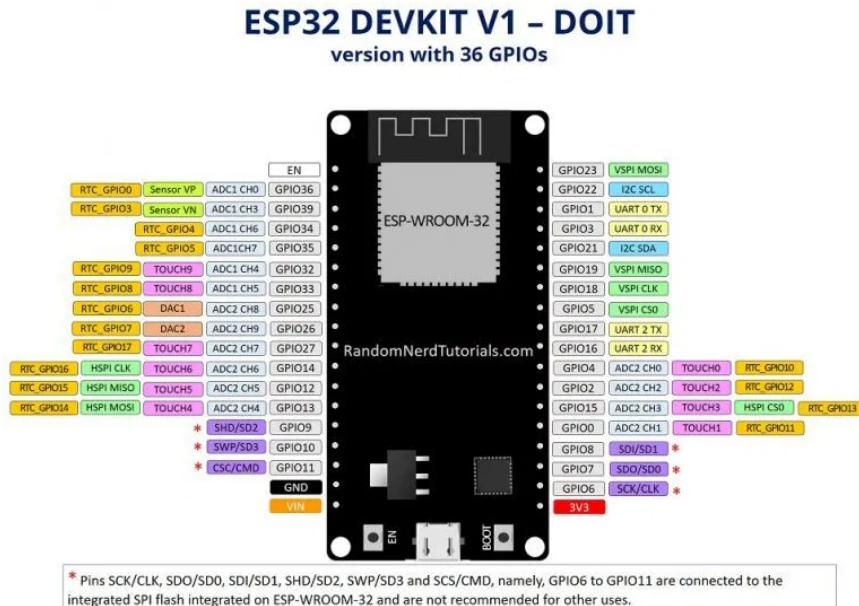
ESP32 memiliki lebih banyak GPIO dengan lebih banyak fungsi dibandingkan dengan ESP826. GPIO (*General Purpose Input-Output*) merupakan sebuah pin yang terdapat pada sebuah kontroler yang berfungsi untuk menerima sebuah input atau Output sesuai apa yang kita kehendaki pada program yang dibuat dan pengaplikasiannya di lapangan. Dengan ESP32 kita dapat memutuskan pin mana yang akan digunakan UART, I2C, dan SPI beberapa fungsi tersebut dapat diatur di dalam pendefinisian pada Kode nantinya.

2.1 Versi dengan 30 GPIOs

ESP32 DEVKIT V1 – DOIT version with 30 GPIOs



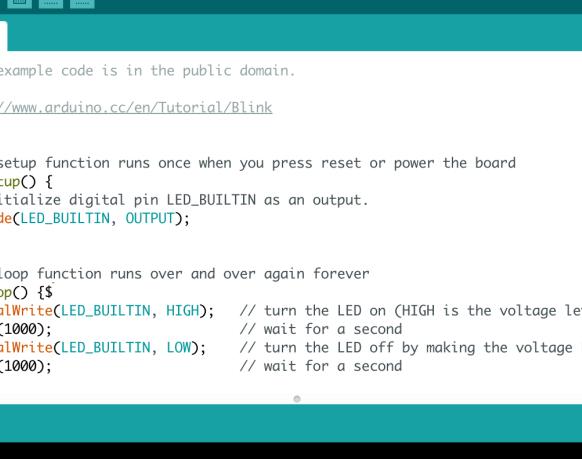
2.2 Version with 36 GPIOs



Dapat dilihat bahwa di kedua versi esp32 diatas memiliki beberapa fungsi dalam 1 pin. Hal ini merupakan sebuah *fitur multiplexing chip esp32* yang memungkinkan dalam 1 pin bisa didefinisikan menjadi beberapa fungsi. Namun jika pengguna tidak melakukan pendefinisian pin pada saat coding maka pin akan digunakan secara default.

BAB 3. Penggunaan Arduino IDE (*setting/konfigurasi*)

Arduino IDE merupakan sebuah Software Compiler dari ARDUINO yang dapat digunakan untuk melakukan programming logic microcontroller salah satunya ESP32.



The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.8.5". The top menu bar includes File, Edit, Tools, Sketch, Help, and a Language dropdown. Below the menu is a toolbar with icons for Open, Save, Upload, Download, and Print. The main workspace displays the "Blink" sketch. The code is as follows:

```
This example code is in the public domain.  
http://www.arduino.cc/en/Tutorial/Blink  
*/  
  
// the setup function runs once when you press reset or power the board  
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)  
  delay(1000);                      // wait for a second  
  digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making the voltage LOW  
  delay(1000);                      // wait for a second  
}
```

BAB 4. Esp32 Input/Output

4.1 ESP32 Input Digital

Input digital dapat dilakukan di ESP32 dengan melakukan konfigurasi pin seperti langkah berikut :

Pertama, atur GPIO yang ingin di baca sebagai INPUT, menggunakan fungsi `pinMode()` sebagai berikut:

```
pinMode(GPIO, INPUT);
```

Untuk membaca input digital, seperti tombol, Kita bisa menggunakan fungsi `digitalRead()`, yang menerima sebagai argumen, GPIO (nomor int) yang dimaksud.

```
digitalRead(GPIO);
```

Semua GPIO ESP32 dapat digunakan sebagai input, kecuali GPIO 6 hingga 11 (karena sudah terhubung ke flash SPI terintegrasi).

4.2 Kontrol Output Digital ESP32

Output digital dapat dilakukan di ESP32 dengan melakukan konfigurasi pin seperti langkah berikut :

Pertama, Kita perlu mengatur GPIO yang ingin di kontrol sebagai OUTPUT. Menggunakan fungsi `pinMode()` sebagai berikut:

```
pinMode(GPIO, OUTPUT);
```

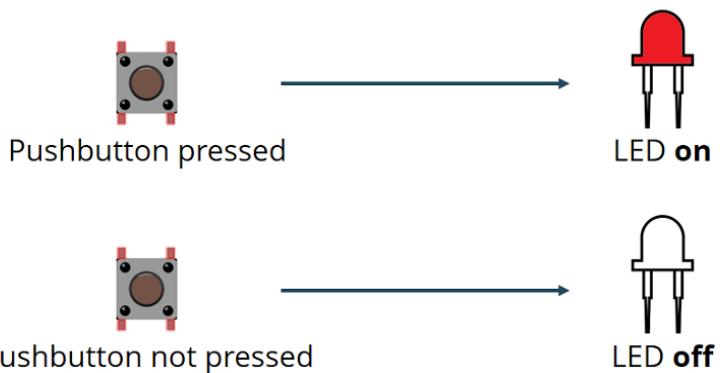
Untuk mengontrol output digital, Kami hanya perlu menggunakan `digitalWrite()` fungsi, yang menerima sebagai argument, GPIO (nomor int), dan statusnya HIGH atau LOW.

```
digitalWrite(GPIO, STATE);
```

Semua GPIO dapat digunakan sebagai output kecuali GPIO 6 hingga 11 (terhubung ke flash SPI terintegrasi) dan GPIO 34, 35, 36 dan 39 (hanya input GPIO);

4.3 Contoh Pengaplikasian Sederhana

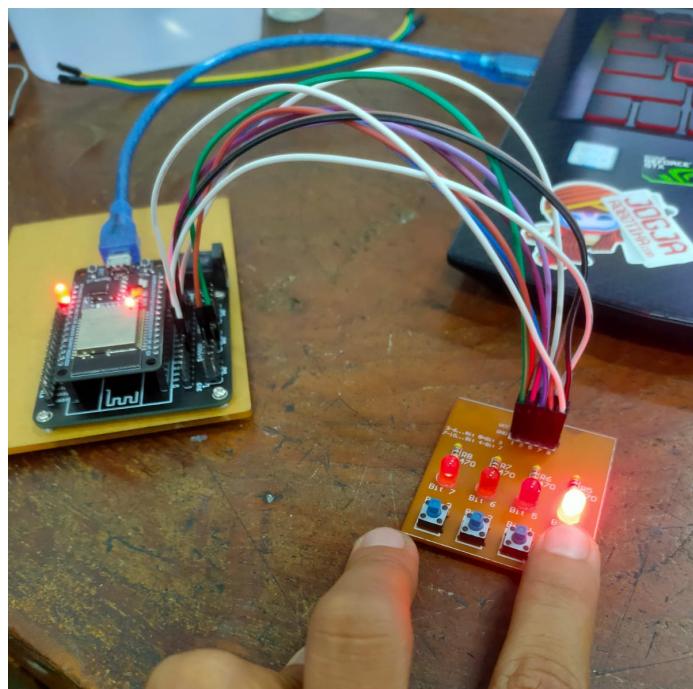
Membaca status tombol dan menyalakan LED sesuai dengan yang diilustrasikan pada gambar berikut.



```
//Kode
// set pin numbers
const int buttonPin = 4; // the number of the
pushbutton pin
const int ledPin = 5; // the number of the
LED pin
// variable for storing the pushbutton status
int buttonState = 0;
void setup() {
    Serial.begin(115200);
    // initialize the pushbutton pin as an input
    pinMode(buttonPin, INPUT);
    // initialize the LED pin as an output
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // read the state of the pushbutton value
    buttonState = digitalRead(buttonPin);
    Serial.println(buttonState);
    // check if the pushbutton is pressed.
    // if it is, the buttonState is HIGH
    if (buttonState == HIGH) {
        // turn LED on
        digitalWrite(ledPin, HIGH);
    } else {
        // turn LED off
        digitalWrite(ledPin, LOW);
    }
}
```

Contoh Praktik :



4.4 Cara Kerja Kode

Dalam dua baris berikut, kami membuat variabel untuk menentukan pin:

```
const int buttonPin = 4;  
const int ledPin = 5;
```

Tombol dihubungkan ke GPIO 4 dan LED terhubung ke GPIO 5. Saat menggunakan Arduino IDE dengan ESP32, Pin yang dituliskan 4 sesuai dengan GPIO 4 dan 5 sesuai dengan GPIO 5. Selanjutnya, Kita membuat variabel untuk menahan status tombol. Secara default, itu 0 (tidak ditekan) atau mati.

```
int buttonState = 0;
```

Dalam pendefinisian `setup()`, Kita menginisialisasi tombol sebagai `INPUT`, dan LED sebagai `OUTPUT`. Untuk itu, digunakan fungsi `pinMode()` yang menerima pin yang di maksud, dan modenya: `INPUT` atau `OUTPUT`.

```
pinMode(buttonPin, INPUT);  
pinMode(ledPin, OUTPUT);
```

- Dalam `loop()` adalah tempat membaca status tombol dan mengatur LED yang sesuai.

- Di baris berikutnya digunakan untuk membaca status tombol dan menyimpannya di button state variable. Seperti yang telah kita lihat sebelumnya dengan menggunakan digitalRead() function.

```
buttonState = digitalRead(buttonPin);
```

Pernyataan if berikut, memeriksa apakah status tombol adalah HIGH. Jika ya, itu menyalakan LED menggunakan fungsi digitalRead() yang menerima sebagai argumen ledpin, dan HIGH state.

```
if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
}
```

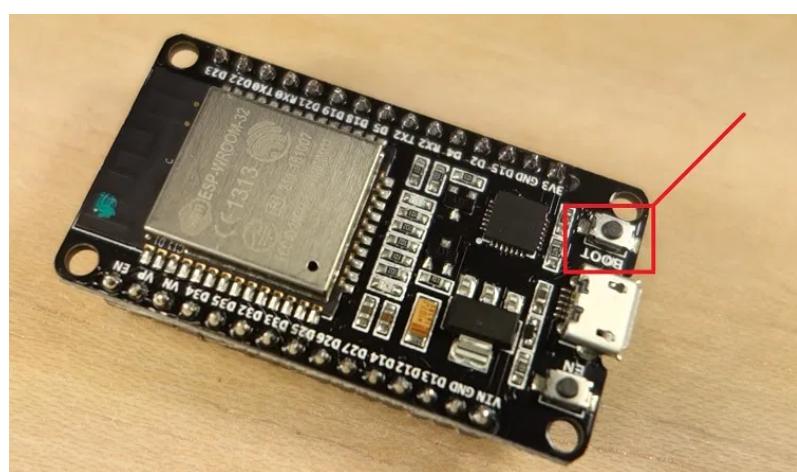
Jika status tombol tidak HIGH, maka LED dihidupkan. bisa diberikan pengecualian LED LOW sebagai argumen kedua dalam digitalWrite() function.

```
else {
    digitalWrite(ledPin, LOW);
}
```

4.5 Mengunggah Kode

Sebelum mengklik tombol unggah, buka **Alat > Papan**, dan pilih papan yang digunakan. Dalam kasus ini adalah **papan DOIT ESP32 DEVKIT V1**. Buka **Alat > Port** dan pilih port COM yang terhubung dengan ESP32. Kemudian, tekan tombol unggah dan tunggu pesan “**Done Upload**”.

Jika terlihat banyak titik (..._____) pada jendela debugging dan pesan “Gagal terhubung ke ESP32: Waktu habis menunggu header paket”, itu berarti perlu menekan tombol BOOT on-board ESP32 setelah titik mulai muncul.



BAB 5. Esp32 PWM

5.1 Pengontrol PWM LED ESP32

ESP32 memiliki pengontrol PWM LED dengan 16 saluran independen yang dapat dikonfigurasi untuk menghasilkan sinyal PWM dengan properti yang berbeda. Berikut langkah-langkah untuk meredupkan LED dengan PWM menggunakan Arduino IDE:

1. Pilih saluran PWM. Ada 16 saluran dari 0 hingga 15.
2. Kemudian, atur frekuensi sinyal PWM. Untuk LED, frekuensi 5000 Hz baik-baik saja untuk digunakan.
3. Atur resolusi siklus tugas sinyal: ESP32 memiliki resolusi dari 1 hingga 16 bit. Kami akan menggunakan resolusi 8-bit, yang berarti dapat mengontrol kecerahan LED menggunakan nilai dari 0 hingga 255.
4. Selanjutnya, tentukan ke GPIO atau GPIO mana sinyal akan muncul. Dengan menggunakan fungsi berikut:

`ledcAttachPin(GPIO, channel)`

Fungsi ini menerima dua argumen. Yang pertama adalah GPIO yang akan mengeluarkan sinyal, dan yang kedua adalah saluran yang akan menghasilkan sinyal.

5. Terakhir, untuk mengontrol kecerahan LED menggunakan PWM, digunakan fungsi berikut:

`ledcWrite(channel, dutycycle)`

Fungsi ini menerima sebagai argumen saluran yang menghasilkan sinyal PWM, dan siklus kerja.

Contoh Praktik PWM:

```
// the number of the LED pin
const int ledPin = 26; // 26 corresponds to GPIO16
const int ledPin2 = 25; // 15 corresponds to GPIO17

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

void setup(){
    // configure LED PWM functionalitites
    ledcSetup(ledChannel, freq, resolution);
```

```

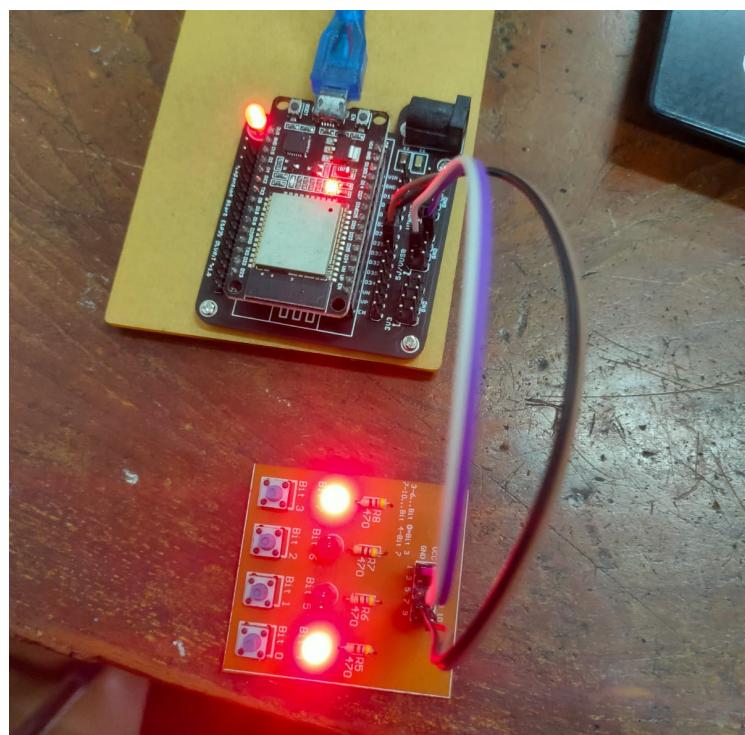
// attach the channel to the GPIO to be controlled
ledcAttachPin(ledPin, ledChannel);
ledcAttachPin(ledPin2, ledChannel);
}

void loop(){
    // increase the LED brightness
    for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
        // changing the LED brightness with PWM
        ledcWrite(ledChannel, dutyCycle);
        delay(15);
    }

    // decrease the LED brightness
    for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
        // changing the LED brightness with PWM
        ledcWrite(ledChannel, dutyCycle);
        delay(15);
    }
}

```

Lampu akan menyala secara terang redup dan redup ke terang secara otomatis

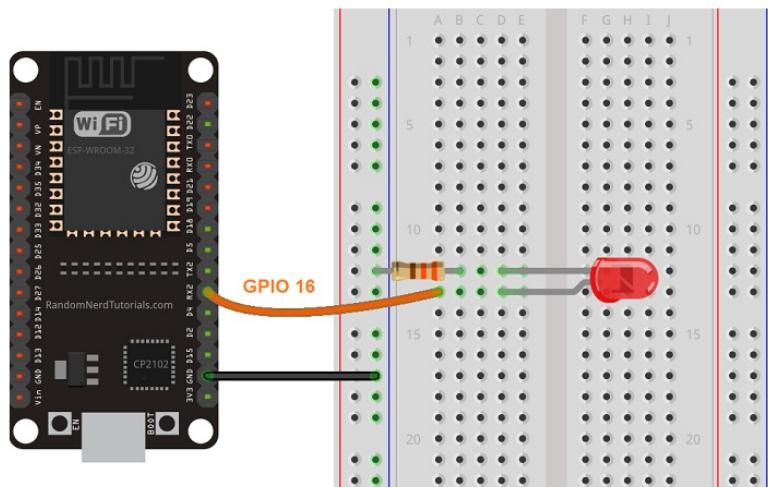


5.1.1 Meredupkan LED

Berikut merupakan contoh sederhana untuk melihat cara menggunakan kontroler PWM LED ESP32 menggunakan Arduino IDE.

- **Skema 2 LED**

Hubungkan LED ke ESP32 seperti pada diagram skematik berikut. LED harus terhubung ke **GPIO 16**.



(Skema ini menggunakan versi modul ESP32 DEVKIT V1 dengan 30 GPIO – jika menggunakan model lain, periksa pinout untuk papan yang digunakan.)

- **KODE**

Mulai dengan menentukan pin yang dipasangi LED. Dalam hal ini LED terpasang ke **GPIO 16**

```
const int ledPin = 16; // 16 corresponds to GPIO16
```

Kemudian, atur properti sinyal PWM. misalnya frekuensi 5000 Hz, memilih saluran 0 untuk menghasilkan sinyal, dan menetapkan resolusi 8 bit.

```
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;
```

Dalam `setup()` perlu dikonfigurasi PWM LED dengan properti yang telah ditentukan sebelumnya dengan menggunakan `ledcSetup()` fungsi yang menerima sebagai argumen, `ledChannel`, frekuensi, dan resolusi, sebagai berikut:

```
ledcSetup(ledChannel, freq, resolution);
```

Selanjutnya pilih GPIO yang akan menerima sinyal. Untuk itu gunakan fungsi ledcAttachPin() yang menerima sebagai argumen GPIO yang ingin mendapatkan sinyal, dan saluran GPIO ini yang menghasilkan sinyal. Dalam contoh ini, kita akan mendapatkan sinyal di led pin GPIO, yang sesuai dengan **GPIO 16**. Saluran yang menghasilkan sinyal adalah saluran yang dipimpin, yang sesuai dengan saluran 0.

```
ledcAttachPin(ledPin, ledChannel);
```

Loop akan memvariasikan siklus tugas antara 0 dan 255 untuk meningkatkan kecerahan LED.

```
for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
}
```

Nilai PWM, antara 255 dan 0 untuk mengurangi/menambah kecerahan.

```
for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
}
```

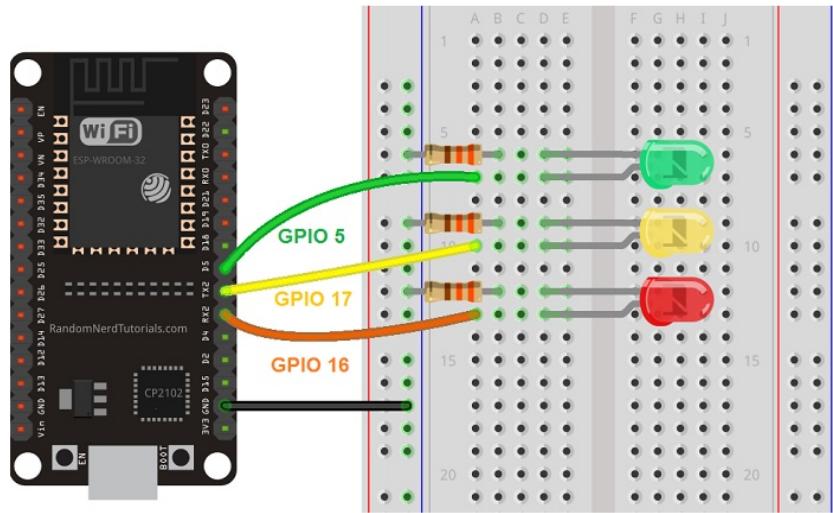
Untuk mengatur kecerahan LED, hanya perlu menggunakan ledcWrite() fungsi yang menerima sebagai argumen saluran yang menghasilkan sinyal, dan siklus kerja.

```
ledcWrite(ledChannel, dutyCycle);
```

Karena kita menggunakan resolusi 8-bit, siklus tugas akan dikontrol menggunakan nilai dari 0 hingga 255. Perhatikan bahwa di ledcWrite() fungsinya kita menggunakan channel yang menghasilkan sinyal, dan bukan GPIO.

- **Skema 3 LED**

Tambahkan dua LED lagi ke sirkuit Anda dengan mengikuti diagram skema berikut:



(Skema ini menggunakan versi modul ESP32 DEVKIT V1 dengan 30 GPIO – jika menggunakan model lain, periksa pinout untuk papan yang digunakan.)

Ini adalah kode yang sama dengan yang sebelumnya tetapi dengan beberapa modifikasi. Telah didefinisikan dua variabel lagi untuk dua LED baru, yang mengacu pada **GPIO 17** dan **GPIO 5**.

```
const int ledPin2 = 17; // 17 corresponds to GPIO17
```

```
const int ledPin3 = 5; // 5 corresponds to GPIO5
```

Kemudian, di `setup()`, telah ditambahkan baris berikut untuk menetapkan kedua GPIO ke saluran 0. Ini berarti bahwa akan mendapatkan sinyal yang sama, yang dihasilkan pada saluran 0, di kedua GPIO.

```
ledcAttachPin(ledPin2, ledChannel);  
ledcAttachPin(ledPin3, ledChannel);
```

BAB 6. Esp32 Interrupt

6.1 Memperkenalkan Interupsi

Untuk memicu suatu peristiwa dengan sensor gerak PIR, digunakan interupsi. Interupsi berguna untuk membuat sesuatu terjadi secara otomatis dalam program mikrokontroler, dan dapat membantu memecahkan masalah waktu. Dengan interupsi, tidak perlu terus-menerus memeriksa nilai pin saat ini. Dengan interupsi, ketika perubahan terdeteksi, suatu peristiwa dipicu (fungsi dipanggil).

Untuk mengatur interupsi di Arduino IDE, kami menggunakan **AttachInterrupt()** fungsi, yang menerima sebagai argumen: pin GPIO, nama fungsi yang akan dieksekusi, dan mode:

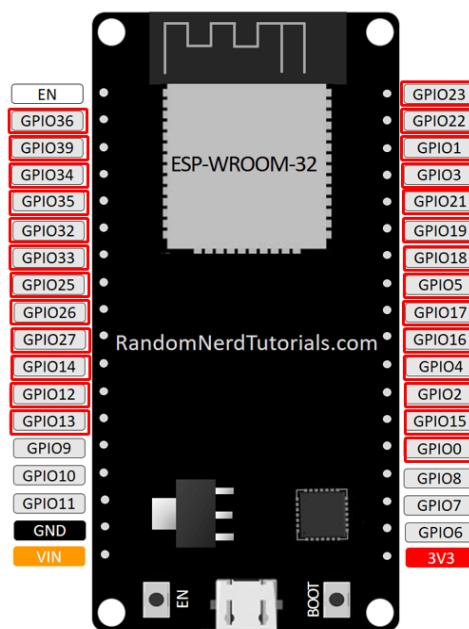
```
attachInterrupt(digitalPinToInterrupt(GPIO), function, mode);
```

6.2 Gangguan GPIO

Argumen pertama adalah nomor GPIO. Biasanya, kami harus menggunakan **digitalPinToInterrupt** (GPIO) untuk mengatur GPIO aktual sebagai pin interupsi. Misalnya, jika ingin menggunakan **GPIO 27** sebagai interupsi, gunakan:

```
digitalPinToInterrupt(27)
```

Dengan papan ESP32, semua pin yang disorot dengan persegi panjang merah pada gambar berikut dapat dikonfigurasi sebagai pin interupsi. Dalam contoh ini kita akan menggunakan **GPIO 27** sebagai interupsi yang terhubung ke sensor PIR Motion.



6.3 Fungsi yang akan dipicu

Argumen kedua dari fungsi **AttachInterrupt()** adalah nama fungsi yang akan dipanggil setiap kali interupsi dipicu.

6.4 Mode

Argumen ketiga adalah modus. Ada 5 mode berbeda:

- **LOW**: untuk memicu interupsi setiap kali pin **LOW**;
- **HIGH**: untuk memicu interupsi setiap kali pin **HIGH**;
- **CHANGE**: untuk memicu interupsi setiap kali pin berubah nilai – misalnya dari **HIGH** ke **LOW** atau **LOW** ke **HIGH**;

- FALLING: ketika pin berubah dari HIGH ke LOW;
- RISING: untuk memicu ketika pin berubah dari LOW ke HIGH.

Dalam contoh ini akan menggunakan mode RISING, karena ketika sensor gerak PIR detect Movement, maka GPIO yang terhubung akan berubah dari LOW ke HIGH.

6.5 Memperkenalkan Timer

Dalam contoh ini juga akan dikenalkan timer. jika ingin LED tetap menyala selama beberapa detik yang telah ditentukan setelah gerakan terdeteksi. Alih-alih menggunakan `delay()` function yang memblokir kode lain dan tidak memungkinkan untuk melakukan hal lain selama beberapa detik yang ditentukan, kita harus menggunakan pengatur waktu.

- **Fungsi delay ()**

`delay()` berfungsi untuk menerima nomor int tunggal sebagai argumen. Angka ini menunjukkan waktu dalam milidetik program harus menunggu sampai pindah ke baris kode berikutnya.

`delay(time in milliseconds)`

Saat telah dilakukan `delay(1000)` program, berarti berhenti pada baris itu selama 1 detik. `delay()` adalah fungsi pemblokiran. Fungsi pemblokiran mencegah program melakukan hal lain sampai tugas tertentu selesai. Jika pekerjaan membutuhkan banyak tugas untuk dilakukan secara bersamaan, hal itu tidak dapat menggunakan `delay()`. Untuk sebagian besar proyek, harus menghindari penggunaan `delay` dan menggunakan pengatur waktu sebagai gantinya.

- **Fungsi millis()**

Menggunakan fungsi yang disebut `millis()` dengan tujuan dapat mengembalikan jumlah milidetik yang telah berlalu sejak program pertama kali dimulai.

`millis()`

Fungsi ini dapat dengan mudah memverifikasi berapa banyak waktu yang telah berlalu tanpa memblokir kode yang lainnya.

6.6 Mengedipkan LED dengan Milis

Kode berikut menunjukkan bagaimana dapat digunakan `millis ()` berfungsi untuk membuat proyek LED berkedip. Itu menyalakan LED selama 1000 milidetik, dan kemudian mematikannya.

```
const int ledPin = 26;
int ledState = LOW;
unsigned long previousMillis = 0;
```

```

// constants won't change :
const long interval = 1000; // interval at which to blink (milliseconds)
void setup() {
    // set the digital pin as output:
    pinMode(ledPin, OUTPUT);
}
void loop() {
    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= interval) {
        // save the last time you blinked the LED
        previousMillis = currentMillis;
        // if the LED is off turn it on and vice-versa:
        if (ledState == LOW) {
            ledState = HIGH;
        } else {
            ledState = LOW;
        }
        // set the LED with the ledState of the variable:
        digitalWrite(ledPin, ledState);
    }
}

```

• Cara kerja kode

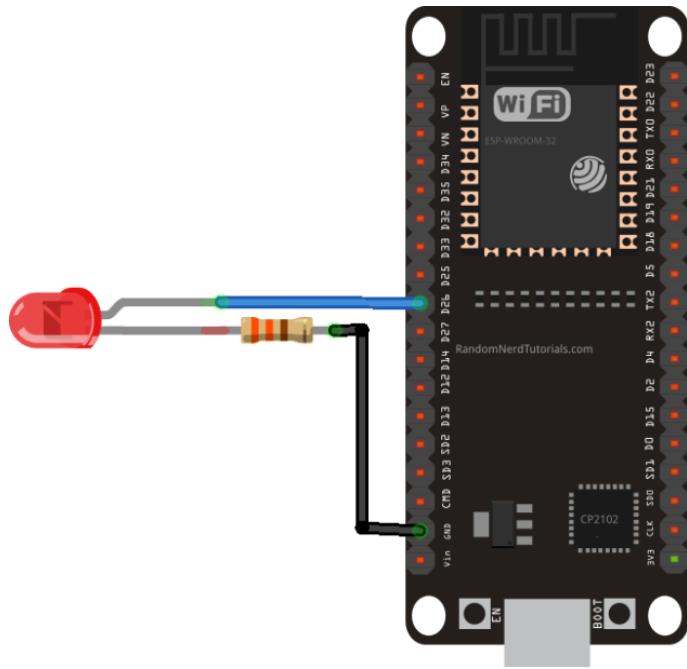
Pada dasarnya, kode ini mengurangi waktu yang direkam sebelumnya (`previousMillis`) dari waktu sekarang (`current millis`). Jika sisa lebih besar dari interval (dalam hal ini, 1000 milidetik), program memperbarui variabel `previousMillis` ke waktu saat ini, dan menyalakan atau mematikan LED.

```

if(currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;
    ...
}

```

Kita dapat menambahkan tugas lain ke `loop()` fungsi dan kode akan tetap mengedipkan LED setiap satu detik. Dengan cara mengunggah kode ini ke ESP32 dan merakit diagram skematik berikut untuk mengujinya dan mengubah jumlah milidetik untuk melihat cara kerjanya.

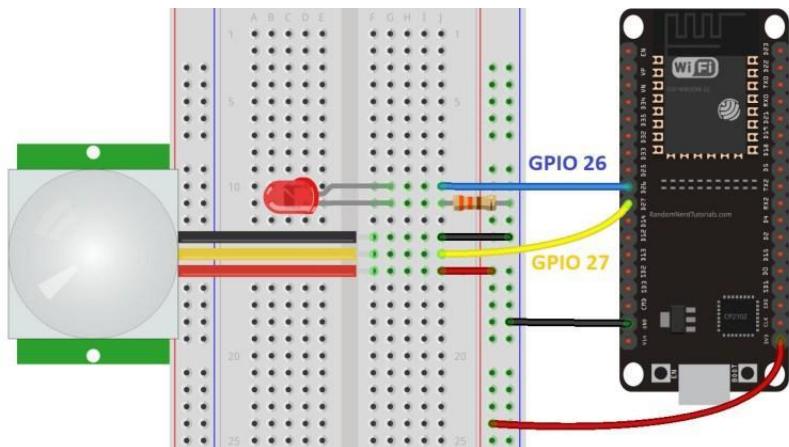


6.7 ESP32 dengan Sensor Gerak PIR

Setelah memahami konsep interupsi dan setting timer, maka pembahasan selanjutnya adalah mengenai ESP32 dengan sensor gerak PIR.

6.7.1 Skema

Rangkaian yang akan kita buat mudah untuk dirakit, kita akan menggunakan LED dengan resistor. LED terhubung ke **GPIO 26**. Akan digunakan **Sensor Gerak Mini AM312 PIR** yang beroperasi pada 3.3V. Ini akan terhubung ke **GPIO 27**. Cukup ikuti diagram skema berikutnya.



Penting : Sensor Gerak Mini AM312 PIR yang digunakan dalam proyek ini beroperasi pada 3.3V. Namun, jika menggunakan

sensor gerak PIR lain seperti HC-SR501 , ia beroperasi pada 5V yang dapat dimodifikasi untuk beroperasi pada 3.3V atau cukup menyalakannya menggunakan pin Vin.

6.7.2 Cara Kerja Kode

Kode dapat diunggah apa adanya, atau dapat mengubah jumlah detik LED menyala setelah mendeteksi gerakan. Cukup ubah variabel timeSeconds dengan jumlah detik yang diinginkan.

```
#define timeSeconds 10

// Set GPIOs for LED and PIR Motion Sensor
const int led = 26;
const int motionSensor = 27;

// Timer: Auxiliary variables
unsigned long now = millis();
unsigned long lastTrigger = 0;
boolean startTimer = false;

// Checks if motion was detected, sets LED
HIGH and starts a timer
void IRAM_ATTR detectsMovement() {
    Serial.println("MOTION DETECTED!!!!");
    digitalWrite(led, HIGH);
    startTimer = true;
    lastTrigger = millis();
}

void setup() {
    // Serial port for debugging purposes
    Serial.begin(115200);

    // PIR Motion Sensor mode INPUT_PULLUP
    pinMode(motionSensor, INPUT_PULLUP);
    // Set motionSensor pin as interrupt,
    assign interrupt function and set RISING mode

    attachInterrupt(digitalPinToInterrupt(motionSens
or), detectsMovement, RISING);

    // Set LED to LOW
    pinMode(led, OUTPUT);
```

```

    digitalWrite(led, LOW);
}

void loop() {
    // Current time
    now = millis();
    // Turn off the LED after the number of
    // seconds defined in the timeSeconds variable
    if(startTimer && (now - lastTrigger >
    (timeSeconds*1000))) {
        Serial.println("Motion stopped...");
        digitalWrite(led, LOW);
        startTimer = false;
    }
}

```

Mulailah dengan menetapkan dua pin GPIO ke led dan variabel motionSensor.

```
// Set GPIOs for LED and PIR Motion Sensor
const int led = 26;
const int motionSensor = 27;
```

Kemudian, buat variabel yang memungkinkan untuk mengatur timer untuk mematikan LED setelah gerakan terdeteksi.

```
// Timer: Auxiliar variables
long now = millis();
long lastTrigger = 0;
boolean startTimer = false;
```

Itu now variable memegang waktu saat ini. Dimana variabel LastTrigger menahan waktu ketika sensor PIR mendeteksi pergerakan. startTimer adalah variabel boolean yang memulai timer ketika gerakan terdeteksi.

- **Setup()**

Pada tahap setup(), mulai dengan menginisialisasi port Serial pada 115200 baud rate.

```
Serial.begin(115200);
```

Atur sensor Gerak PIR sebagai INPUT PULLUP.

```
pinMode(motionSensor, INPUT_PULLUP);
```

Untuk mengatur pin sensor PIR sebagai interupsi, gunakan fungsi `attachInterrupt()` seperti yang dijelaskan sebelumnya.

```
attachInterrupt(digitalPinToInterrupt(motionSensor),  
detectsMovement, RISING);
```

Pin yang akan mendeteksi pergerakan adalah **GPIO 27** dan itu akan memanggil fungsi `detectsMovement()` pada mode RISING dan LED adalah OUTPUT yang statusnya dimulai dari LOW.

```
pinMode(led, OUTPUT);  
digitalWrite(led, LOW);
```

- **Loop()**

`loop()` adalah fungsi yang terus berjalan berulang-ulang. Dalam setiap loop, variabel `now` diperbarui dengan waktu saat ini.

```
now = millis();
```

Tidak ada lagi yang dilakukan di `loop()`. Tapi, ketika gerakan terdeteksi, fungsi `detectsMovement()` dipanggil karena telah mengatur interupsi sebelumnya pada `setup()`. fungsi `detectsMovement()` adalah mencetak pesan di Serial Monitor, menyalakan LED, menyetel `startTimer` variabel boolean ke `true` dan memperbarui `LastTrigger` variabel dengan waktu sekarang.

```
void IRAM_ATTR detectsMovement() {  
    Serial.println("MOTION DETECTED!!!");  
    digitalWrite(led, HIGH);  
    startTimer = true;  
    lastTrigger = millis();  
}
```

Catatan: `IRAM_ATTR` digunakan untuk menjalankan kode interupsi di RAM, jika tidak, kode disimpan dalam flash dan akan berjalan lebih lambat. Setelah langkah ini, kode kembali ke `loop()`.

Kali ini, `startTimer` variabel adalah benar. Jadi, ketika waktu yang ditentukan dalam detik telah berlalu (sejak gerakan terdeteksi), berikut ini:if pernyataan akan benar.

```
if(startTimer && (now - lastTrigger > (timeSeconds*1000))) {  
    Serial.println("Motion stopped...");  
    digitalWrite(led, LOW);
```

```
startTimer = false;  
}
```

Pesan “Gerakan berhenti...” akan tercetak di Serial Monitor, LED dimatikan, dan variabel startTimer disetel ke false.

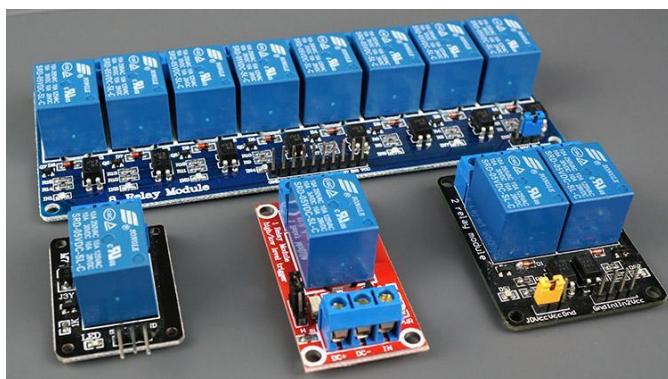
BAB 7. Esp32 Relay

7.1 Memperkenalkan Relay

Relay adalah saklar yang dioperasikan secara listrik dan seperti saklar lainnya, relay dapat dihidupkan atau dimatikan, membiarkan arus mengalir atau tidak. Relay dapat dikontrol dengan tegangan rendah, seperti 3.3V yang disediakan oleh GPIO ESP32 dan memungkinkan kita untuk mengontrol tegangan tinggi seperti 12V, 24V atau tegangan listrik (230V di Eropa dan 120V di AS).

7.2 Modul relay 1, 2, 4, 8, 16 Saluran

Ada modul relay yang berbeda dengan jumlah saluran yang berbeda. dapat ditemukan modul relay dengan satu, dua, empat, delapan, dan bahkan enam belas saluran. Jumlah saluran menentukan jumlah keluaran yang dapat dikendalikan.



Ada modul relay yang elektromagnetnya dapat ditenagai oleh 5V dan dengan 3.3V. Keduanya dapat digunakan dengan ESP32 – atau dapat menggunakan pin VIN (yang menyediakan 5V) atau pin 3.3V. Selain itu, beberapa dilengkapi dengan optocoupler internal yang menambahkan "lapisan" perlindungan ekstra, yang secara optik mengisolasi ESP32 dari sirkuit relay.

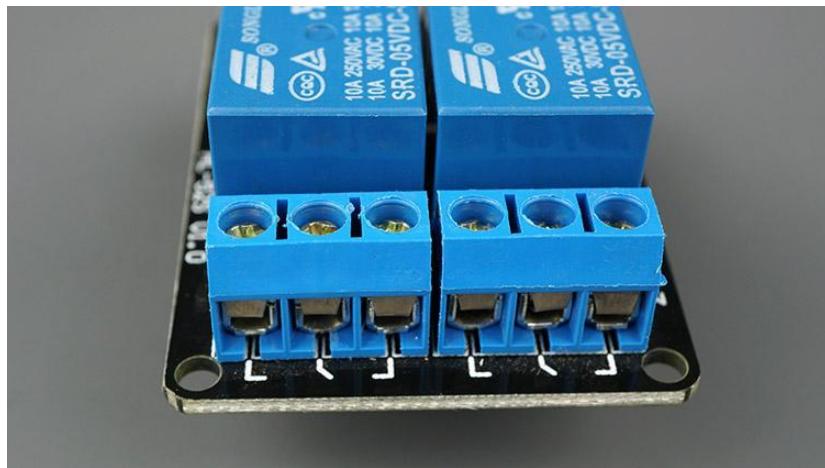
7.3 Pinout Relay

Untuk tujuan demonstrasi, mari kita lihat pin out modul relay 2 saluran. Menggunakan modul relay dengan jumlah saluran yang berbeda serupa.



Di sisi kiri, ada dua set tiga soket untuk menghubungkan tegangan tinggi, dan pin di sisi kanan (tegangan rendah) terhubung ke GPIO ESP32.

7.4 Koneksi Tegangan Listrik



Modul relay yang ditunjukkan pada foto sebelumnya memiliki dua konektor, masing-masing dengan tiga soket: common (COM), Normally Closed (NC), and Normally Open (NO).

- **COM:** hubungkan arus yang ingin dikendalikan (tegangan listrik).
- **NC (Normally Closed):** konfigurasi yang biasanya tertutup digunakan bila ingin relay ditutup secara default. NC adalah pin COM yang terhubung, artinya arus mengalir kecuali jika ingin mengirim sinyal dari ESP32 ke modul relay untuk membuka rangkaian dan menghentikan aliran arus.
- **NO (Normally Open):** konfigurasi yang biasanya terbuka bekerja sebaliknya: tidak ada koneksi antara pin NO dan COM, sehingga sirkuit terputus kecuali jika ingin mengirim sinyal dari ESP32 untuk menutup sirkuit.

7.5 Pin Kontrol



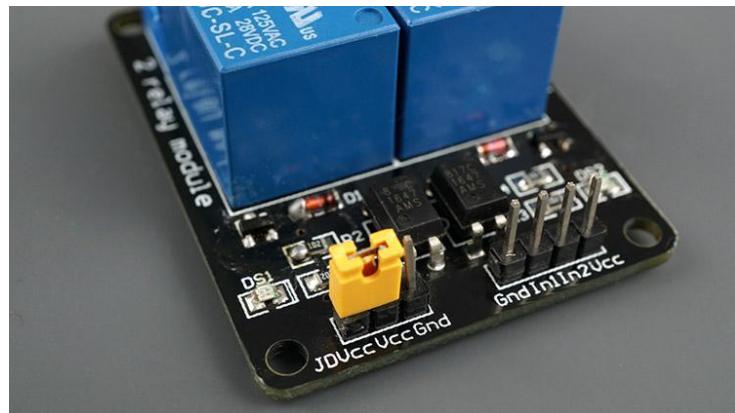
Sisi tegangan rendah memiliki satu set empat pin dan satu set tiga pin. Set pertama terdiri dari **VCC** dan **GND** untuk menyalakan modul, dan masukan 1 (**IN1**) dan masukan 2 (**IN2**) untuk mengontrol relay bawah dan atas, masing-masing. Jika modul relay hanya memiliki satu saluran, maka hanya akan memiliki satu pin IN. Namun jika memiliki empat saluran, maka akan memiliki empat pin IN, dan seterusnya.

Sinyal yang dikirim ke pin IN, menentukan apakah relay aktif atau tidak. relay dipicu ketika input berjalan di bawah sekitar 2V.

- **Normally Closed configuration (NC) :**
 - Sinyal HIGH – arus mengalir
 - Sinyal LOW – arus **tidak** mengalir
- **Normally Open configuration (NO) :**
 - Sinyal HIGH – arus **tidak** mengalir
 - Sinyal LOW – arus mengalir

Dalam program ini, digunakan konfigurasi yang biasanya tertutup ketika arus harus mengalir sebagian besar waktu, dan hanya ingin menghentikannya sesekali. Namun, gunakan konfigurasi yang biasanya terbuka ketika ingin arus mengalir sesekali (misalnya, menyalakan lampu sesekali).

7.6 Pilihan Power Supply



Set kedua pin terdiri dari: **GND**, **VCC**, dan **JD-VCC** pin. **JD-VCC** pin memberi daya pada elektromagnet relay. Perhatikan bahwa modul memiliki tutup jumper yang menghubungkan pin VCC dan JD-VCC; yang ditampilkan di sini berwarna kuning, tetapi warna mungkin berbeda.

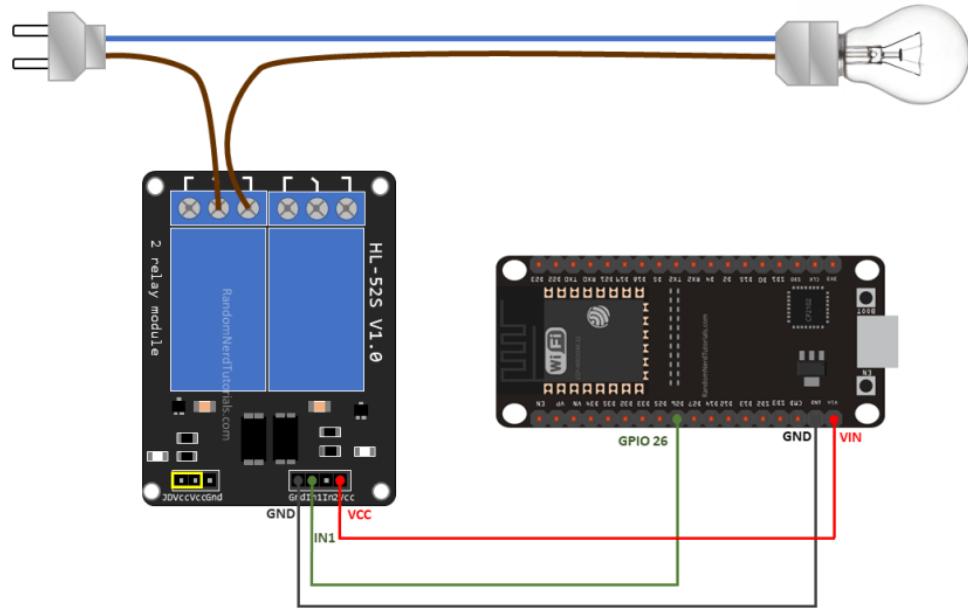
Dengan tutup jumper terpasang, **VCC** dan **JD-VCC** pin terhubung. Itu berarti elektromagnet relay dialiri langsung dari pin power ESP32, sehingga modul relay dan rangkaian ESP32 tidak terisolasi secara fisik satu sama lain.

Tanpa tutup jumper, diperlukan menyediakan sumber daya independen untuk menyalakan elektromagnet relay melalui **JD-VCC** pin. Konfigurasi itu secara fisik mengisolasi relay dari ESP32 dengan optocoupler bawaan modul, yang mencegah kerusakan pada ESP32 jika terjadi lonjakan listrik.

7.7 Menghubungkan Modul Relay ke ESP32

Hubungkan modul relay ke ESP32 seperti yang ditunjukkan pada diagram berikut. Diagram menunjukkan pengkabelan untuk modul relay 2 saluran, pengkabelan dengan jumlah saluran yang berbeda serupa.

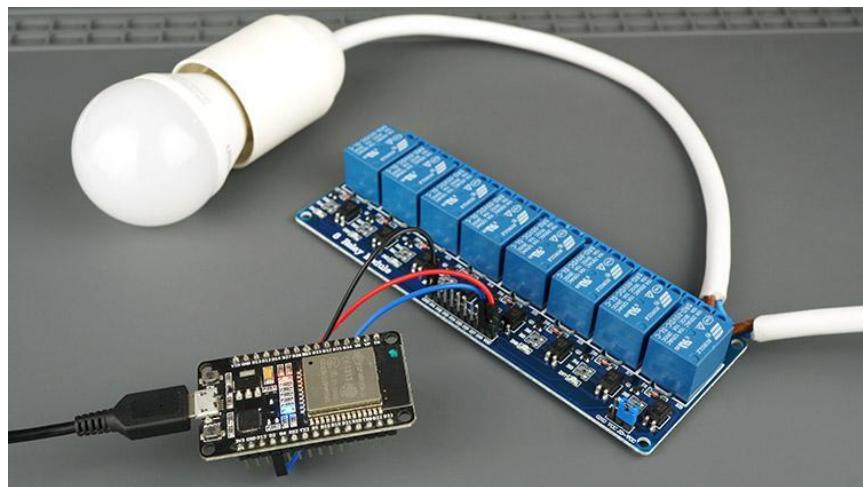
Peringatan: dalam contoh ini, kita berurusan dengan tegangan listrik. Penyalahgunaan dapat mengakibatkan cedera serius. Jika Kami tidak terbiasa dengan tegangan listrik, mintalah seseorang yang akan membantu Kami. Saat memprogram ESP atau memasang kabel sirkuit Kami, pastikan semuanya terputus dari tegangan listrik. Atau, Kami dapat menggunakan sumber daya 12V untuk mengontrol peralatan 12V.



Dalam contoh ini, kita sedang mengontrol sebuah lampu. Kami hanya ingin menyalakan lampu sesekali, jadi lebih baik menggunakan konfigurasi yang biasanya terbuka. Kami menghubungkan pin IN1 ke **GPIO 26**, Kami dapat menggunakan GPIO lain yang sesuai.

7.8 Mengontrol Modul Relay dengan ESP32 – Arduino Sketch

Kode untuk mengontrol relay dengan ESP32 semudah mengontrol LED atau output lainnya. Dalam contoh ini, karena kita menggunakan konfigurasi yang biasanya terbuka, kita perlu mengirim sinyal LOW untuk membiarkan arus mengalir, dan sinyal HIGH untuk menghentikan aliran arus.



Kode berikut akan menyalakan lampu Kami selama 10 detik dan mematikannya selama 10 detik lagi.

```

*****
const int relay = 26;
void setup() {
    Serial.begin(115200);
    pinMode(relay, OUTPUT);
}
void loop() {
    // Normally Open configuration, send LOW
    signal to let current flow
    // (if you're usong Normally Closed
    configuration send HIGH signal)
    digitalWrite(relay, LOW);
    Serial.println("Current Flowing");
    delay(5000);
    // Normally Open configuration, send HIGH
    signal stop current flow
    // (if you're usong Normally Closed
    configuration send LOW signal)
    digitalWrite(relay, HIGH);
    Serial.println("Current not Flowing");
    delay(5000);
}

```

7.9 Cara Kerja Kode

Tentukan pin yang terhubung dengan pin IN relay.

```
const int relay = 26;
```

Dalam `setup()`, tentukan relay sebagai keluaran.

```
pinMode(relay, OUTPUT);
```

Dalam `loop()`, Kirim `LOW` signal untuk membiarkan arus mengalir dan menyalaikan lampu.

```
digitalWrite(relay, LOW);
```

Jika Kami menggunakan konfigurasi yang biasanya tertutup, kirim `HIGH` signal untuk menyalaikan lampu. Kemudian, tunggu 5 detik.

```
delay(5000);
```

Hentikan aliran arus dengan mengirimkan `HIGH` signal ke pin relay. Jika Kami menggunakan konfigurasi yang biasanya tertutup, kirim `LOW` signal untuk menghentikan aliran arus.

```
digitalWrite(relay, HIGH);
```

7.10 Kontrol Beberapa Relay dengan Server Web ESP32



Di bagian ini, kami telah membuat contoh server web yang memungkinkan fungsi untuk mengontrol relay sebanyak yang diinginkan melalui server web apakah mereka dikonfigurasi sebagai biasanya dibuka atau ditutup secara normal. Kita hanya perlu mengubah beberapa baris kode untuk menentukan jumlah relay yang ingin Kami kendalikan dan penetapan pin. Untuk referensi dalam pembuatan server web ini, kami menggunakan [ESPAsyncWebServer library](#).

7.11 Memasang ESPAsyncWebServer library

Ikuti langkah-langkah selanjutnya untuk menginstal `ESPAsyncWebServer` library :

1. Klik di sini untuk mengunduh `ESPAsyncWebServer` library. Kami harus memiliki folder `.zip` di folder Unduhan Kami
2. Buka zip folder `.zip` dan Kami akan mendapatkan folder `ESPAsyncWebServer-master`
3. Ganti nama folder Kami dari `ESPAsyncWebServer-master` menjadi `ESPAsyncWebServer`

4. Pindahkan folder *ESPAsyncWebServer* ke folder Arduino IDE installation libraries

Atau, di Arduino IDE, bisa di cek ke **Sketch > Include Library > Add ZIP library...** dan pilih library yang baru saja diunduh.

Setelah menginstal perlibraryan yang diperlukan, salin kode berikut ke Arduino IDE yang digunakan.

```
// Import required libraries
#include "WiFi.h"
#include "ESPAsyncWebServer.h"
// Set to true to define Relay as Normally Open (NO)
#define RELAY_NO    true
// Set number of relays
#define NUM_RELAYS  5
// Assign each GPIO to a relay
int relayGPIOs[NUM_RELAYS] = {2, 26, 27, 25, 33};
// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
const char* PARAM_INPUT_1 = "relay";
const char* PARAM_INPUT_2 = "state";
// Create AsyncWebServer object on port 80
AsyncWebServer server(80);
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
    <meta name="viewport" content="width=device-width,
initial-scale=1">

<style>
    html {font-family: Arial; display: inline-block;
text-align: center;}
    h2 {font-size: 3.0rem; }
    p {font-size: 3.0rem; }
        body {max-width: 600px; margin:0px auto;
padding-bottom: 25px; }
```

```

.switch {position: relative; display: inline-block;
width: 120px; height: 68px}

.switch input {display: none}

.slider {position: absolute; top: 0; left: 0;
right: 0; bottom: 0; background-color: #ccc;
border-radius: 34px}

.slider:before {position: absolute; content: "";
height: 52px; width: 52px; left: 8px; bottom: 8px;
background-color: #fff; -webkit-transition: .4s;
transition: .4s; border-radius: 68px}

input:checked+.slider {background-color: #2196F3}

input:checked+.slider:before {-webkit-transform:
translateX(52px); -ms-transform: translateX(52px);
transform: translateX(52px)}

</style>
</head>
<body>
<h2>ESP Web Server</h2>
%BUTTONPLACEHOLDER%
<script>function toggleCheckbox(element) {
    var xhr = new XMLHttpRequest();
        if(element.checked){ xhr.open("GET",
"/update?relay="+element.id+"&state=1", true); }
        else { xhr.open("GET",
"/update?relay="+element.id+"&state=0", true); }
    xhr.send();
}</script>
</body>
</html>
) rawliteral";
// Replaces placeholder with button section in your web
page
String processor(const String& var){
//Serial.println(var);
if(var == "BUTTONPLACEHOLDER"){
    String buttons ="";
    for(int i=1; i<=NUM_RELAYS; i++){
        String relayStateValue = relayState(i);

```

```

        buttons+= "<h4>Relay #"+ String(i) + " - GPIO "
+         relayGPIOs[i-1]           +       "</h4><label"
class=\"switch\">><input type=\"checkbox\""
onchange=\"toggleCheckbox(this)\" id=\"" + String(i) +
"\">"           "+"          relayStateValue           +"><span
class=\"slider\">></span></label>";
    }
    return buttons;
}
return String();
String relayState(int numRelay){
if(RELAY_NO) {
    if(digitalRead(relayGPIOs[numRelay-1])) {
        return "";
    }
    else {
        return "checked";
    }
}
else {
    if(digitalRead(relayGPIOs[numRelay-1])) {
        return "checked";
    }
    else {
        return "";
    }
}
return "";
}

void setup() {
// Serial port for debugging purposes
Serial.begin(115200);

// Set all relays to off when the program starts - if
set to Normally Open (NO), the relay is off when you
set the relay to HIGH

for(int i=1; i<=NUM_RELAYS; i++) {
pinMode(relayGPIOs[i-1], OUTPUT);

if(RELAY_NO) {
digitalWrite(relayGPIOs[i-1], HIGH);
}
}
}

```

```

        }

    else{
        digitalWrite(relayGPIOs[i-1], LOW);
    }

}

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
}

// Print ESP32 Local IP Address

Serial.println(WiFi.localIP());

// Route for root / web page

server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request) {
    request->send_P(200, "text/html", index_html,
processor);
} );

// Send a GET request to
<ESP_IP>/update?relay=<inputMessage>&state=<inputMessage2>

server.on("/update", HTTP_GET, []
(AsyncWebServerRequest *request) {

    String inputMessage;
    String inputParam;
String inputMessage2;
    String inputParam2;
    // GET input1 value on
<ESP_IP>/update?relay=<inputMessage>

    if (request->hasParam(PARAM_INPUT_1) &
request->hasParam(PARAM_INPUT_2)) {

```

```

        inputMessage      =
request->getParam(PARAM_INPUT_1)->value();

        inputParam = PARAM_INPUT_1;

        inputMessage2      =
request->getParam(PARAM_INPUT_2)->value();
        inputParam2 = PARAM_INPUT_2;
        if(RELAY_NO) {
            Serial.print("NO ");
digitalWrite(relayGPIOs[inputMessage.toInt() -1],
!inputMessage2.toInt());
        }
        else{
            Serial.print("NC ");
}

digitalWrite(relayGPIOs[inputMessage.toInt() -1],
inputMessage2.toInt());
        }
    }
    else {
        inputMessage = "No message sent";
        inputParam = "none";
    }
    Serial.println(inputMessage + inputMessage2);
    request->send(200, "text/plain", "OK");
} );
// Start server
server.begin();
}
void loop() {
}

```

7.12 Tentukan Konfigurasi Relay

Ubah variabel berikut untuk menunjukkan apakah Kami menggunakan relay dalam konfigurasi normally open (NO) atau normally closed (NC). Mengatur RELAY_NO variabel ke TRUE untuk os yang normally open (NO) disetel ke FALSE untuk normally closed (NC).

```
#define RELAY_NO true
```

7.13 Tentukan Jumlah Relay (Channel)

Kita dapat menentukan jumlah relay yang ingin dikendalikan pada variabel NUM_RELAYS. Misalnya pada kasus ini, kami menyetelnya ke 5.

```
#define NUM_RELAYS 5
```

7.14 Tentukan Tugas Pin Relay

Dalam variabel array berikut, Kami dapat menentukan GPIO ESP32 yang akan mengontrol relay:

```
int relayGPIOs[NUM_RELAYS] = {2, 26, 27, 25,  
33};
```

Jumlah relay yang disetel pada `NUM_RELAYS` variabel harus sesuai dengan jumlah GPIO yang ditetapkan dalam `relayGPIO` array.

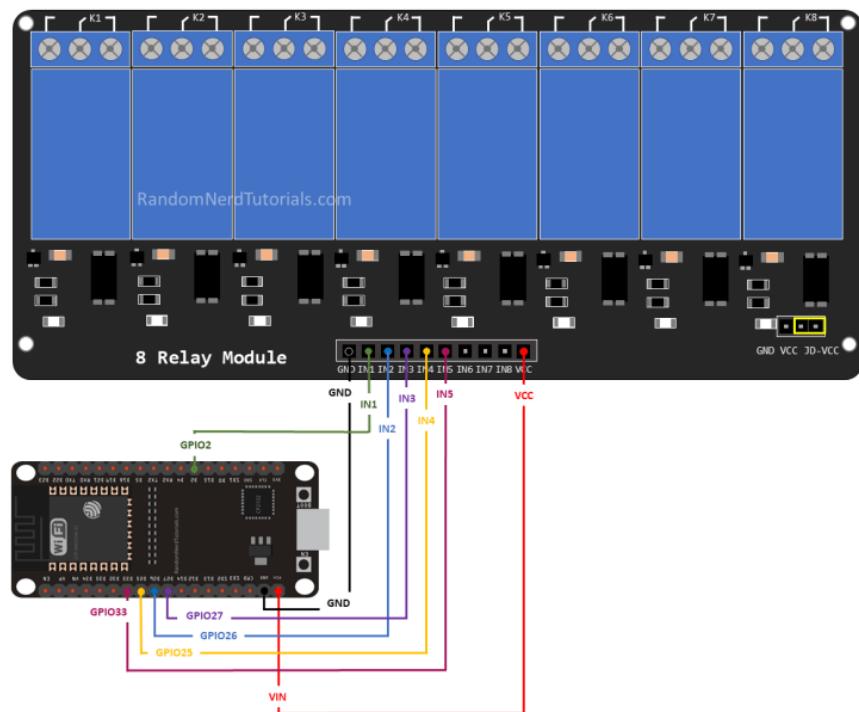
7.15 Kredensial Jaringan

Masukkan kredensial jaringan kedalam variabel berikut.

```
const char* ssid      = "REPLACE_WITH_YOUR_SSID";  
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

7.16 Pengkabelan 8 Saluran Relay ke ESP32

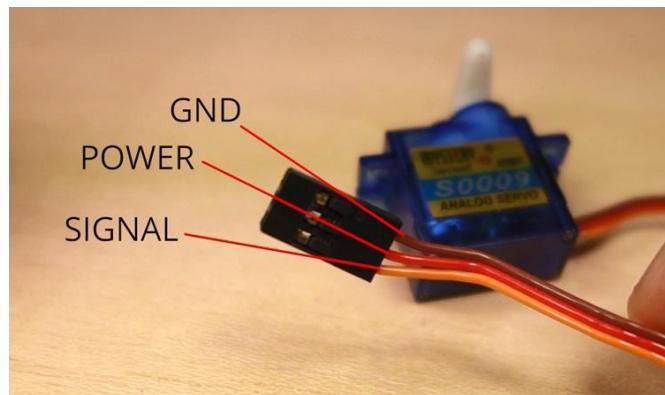
Untuk tujuan demonstrasi, dikontrol 5 saluran relay. Sambungkan ESP32 ke modul relay seperti yang ditunjukkan pada diagram skema berikut.



BAB 8. Esp32 Servo

8.1 Menghubungkan Motor Servo ke ESP32

Motor servo memiliki tiga kabel: daya, ground, dan sinyal. Daya biasanya berwarna merah, GND berwarna hitam atau coklat, dan kabel sinyal biasanya berwarna kuning, oranye, atau putih.

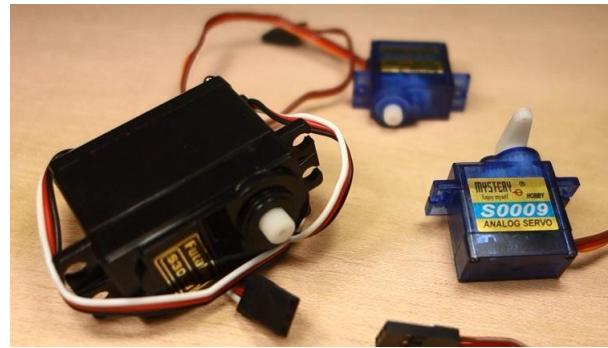


Kabel	Warna
Kekuatan	Merah
GND	Hitam, atau coklat
Sinyal	Kuning, orange, atau putih

Saat menggunakan servo kecil seperti S0009 seperti yang ditunjukkan pada gambar di bawah, Kami dapat menyalakannya langsung dari ESP32.



Tetapi jika Kami menggunakan lebih dari satu servo atau jenis lainnya, Kami mungkin perlu menyalakan servo Kami menggunakan power supply eksternal.



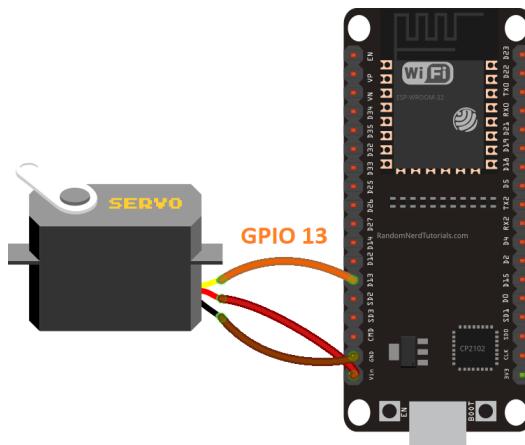
Jika Kami menggunakan servo kecil seperti S0009, Kami harus menghubungkan:

- GND -> ESP32 GND pin;
- Daya -> ESP32 VIN pin;
- Sinyal -> **GPIO 13** (atau pin PWM apa pun).

Catatan: dalam hal ini, Kami dapat menggunakan GPIO ESP32 apa pun, karena GPIO apa pun dapat menghasilkan sinyal PWM. Namun, kami tidak menyarankan penggunaan GPIO 9, 10, dan 11 yang terhubung ke flash SPI terintegrasi dan tidak disarankan untuk penggunaan lain.

● Skema

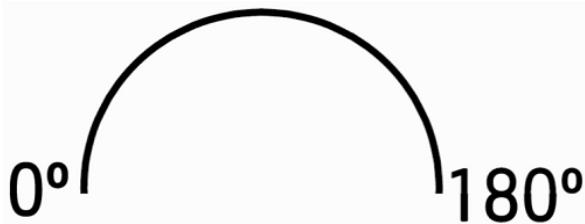
Dalam contoh ini, kami akan menghubungkan kabel sinyal ke **GPIO 13**.



(Skema ini menggunakan versi modul **ESP32 DEVKIT V1** dengan 36 GPIO – jika yang digunakan model lain, periksa pinout untuk papan yang Kami gunakan.)

8.2 Bagaimana Cara Mengontrol Motor Servo?

Kita dapat memposisikan poros servo dalam berbagai sudut dari 0 hingga 180°. Servo dikendalikan menggunakan sinyal modulasi lebar pulsa (PWM). Artinya sinyal PWM yang dikirim ke motor akan menentukan posisi poros.



Untuk mengendalikan motor cukup menggunakan kemampuan PWM dari ESP32 dengan mengirimkan sinyal 50Hz dengan lebar pulsa yang sesuai. Atau Kami dapat menggunakan perpustakaan untuk membuat tugas ini lebih sederhana.

8.3 Mempersiapkan Arduino IDE

Add-on untuk Arduino IDE memungkinkan untuk memprogram ESP32 menggunakan Arduino IDE dan bahasa pemrogramannya.

8.4 Menginstal ESP32 Arduino Servo Library

ESP32 Arduino Servo Library memudahkan untuk mengontrol motor servo. Dengan ESP32 yang menggunakan Arduino IDE. Ikuti langkah-langkah berikut untuk menginstal library di Arduino IDE:

1. Klik di sini untuk mengunduh [ESP32_Arduino_Servo_Library](#). Kami harus memiliki folder .zip di folder Unduhan.
2. Buka zip folder .zip dan Kami akan mendapatkan folder *ESP32-Arduino-Servo-Library-Master*.
3. Ganti nama folder Kami dari *ESP32-Arduino-Servo-Library-Master* ke *ESP32_Arduino_Servo_Library*
4. Pindahkan folder *ESP32_Arduino_Servo_Library* ke folder library instalasi Arduino IDE.
5. Terakhir, buka kembali Arduino IDE.

8.5 Menguji Contoh

Setelah menginstal perpustakaan, buka Arduino IDE Kami. Pastikan Kami memilih papan ESP32, lalu, buka **File > Contoh > ServoESP32 > Simple Servo**.

```
#include <Servo.h>
Servo myservo; // create servo object to
control a servo
// twelve servo objects can be created on most
boards
int pos = 0; // variable to store the servo
position
void setup() {
```

```

        myservo.attach(13); // attaches the servo
on pin 13 to the servo object
    }
void loop() {
    for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
        // in steps of 1 degree
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15); // waits 15ms for the servo to reach the position
    }
    for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15); // waits 15ms for the servo to reach the position
    }
}

```

8.6 Memahami kode

Sketsa ini memutar servo 180 derajat ke satu sisi, dan 180 derajat ke sisi lainnya. Mari kita lihat cara kerjanya.

Pertama, sertakan library Servo:

```
#include <Servo.h>
```

Kemudian, buat objek servo. Dalam hal ini disebut `myservo`.

```
Servo myservo;
```

- **Setup ()**

Dalam `setup()`, Kami menginisialisasi komunikasi serial untuk keperluan debugging, dan melampirkan **GPIO 13** ke objek servo.

```
void setup() {
    myservo.attach(13);
}
```

- **Loop ()**

Dalam `loop()`, kami mengubah posisi poros motor dari 0 menjadi 180 derajat, dan kemudian dari 180 menjadi 0 derajat. Untuk mengatur poros ke

posisi tertentu, hanya perlu menggunakan metode `write()` dalam `servo` objek.

```
myservo.write(pos);
```

8.7 Menguji Sketsa

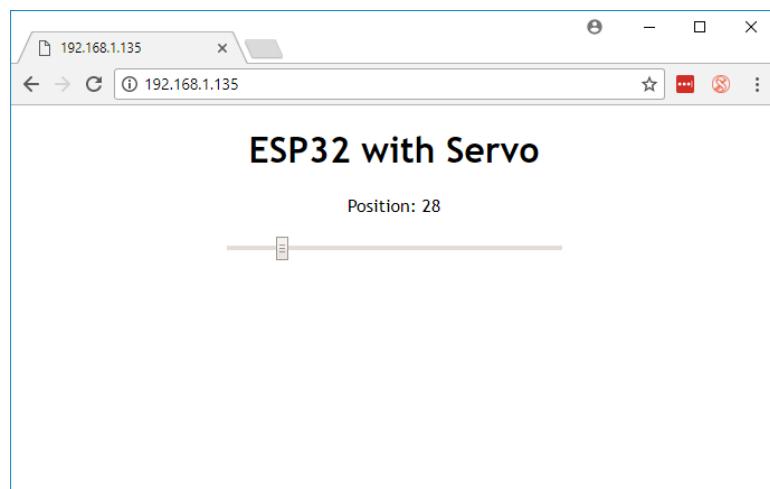
Unggah kode ke ESP32 yang telah dibuat. Setelah mengupload kode, akan terlihat poros motor berputar ke satu sisi dan kemudian ke sisi lainnya.



8.8 Membuat Server Web ESP32

Setelah mengetahui cara mengontrol servo dengan ESP32, selanjutnya membuat server web untuk mengontrolnya. Spesifikasi server web yang akan kami buat adalah sebagai berikut:

- Berisi slider dari 0 hingga 180, yang dapat Kami sesuaikan untuk mengontrol posisi poros servo;
- Nilai sliding saat ini diperbarui secara otomatis di halaman web, serta posisi poros, tanpa perlu menyegarkan halaman web. Untuk ini, kami menggunakan AJAX untuk mengirim permintaan HTTP ke ESP32 di latar belakang;
- Menyegarkan halaman web tidak mengubah nilai sliding, begitu pula posisi poros.



8.9 Membuat Halaman HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <link rel="icon" href="data:, ">
    <style>
        body {
            text-align: center;
            font-family: "Trebuchet MS", Arial;
            margin-left:auto;
            margin-right:auto;
        }
        .slider {
            width: 300px;
        }
    </style>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.
3.1/jquery.min.js"></script>
</head>
<body>
    <h1>ESP32 with Servo</h1>
    <p>Position: <span id="servoPos"></span></p>
        <input type="range" min="0" max="180"
class="slider" id="servoSlider"
onchange="servo(this.value)"/>
    <script>
        var slider =
document.getElementById("servoSlider");
        var servoP =
document.getElementById("servoPos");
        servoP.innerHTML = slider.value;
        slider.oninput = function() {
            slider.value = this.value;
            servoP.innerHTML = this.value;
        }
        $.ajaxSetup({timeout:1000});
        function servo(pos) {
            $.get("/?value=" + pos + "&");
            {Connection: close};
        }
    </script>
</body>
```

```
</script>
</body>
</html>
```

8.10 Membuat Slider

Halaman HTML untuk proyek ini melibatkan pembuatan slider. Untuk membuat slider di HTML Kami menggunakan tag <input>. Tag <input> menentukan bidang tempat pengguna dapat memasukkan data. Ada berbagai macam jenis input. Untuk menentukan sliding, gunakan atribut “**type**” dengan “**range**” value. Di sliding, Kami juga perlu menentukan rentang minimum dan maksimum menggunakan atribut “**min**” dan “**max**”.

```
<input type="range" min="0" max="180"
class="slider" id="servoSlider"
onchange="servo(this.value)"/>
```

Selain itu juga perlu mendefinisikan atribut lain seperti:

- kelas untuk **menata** bilah geser
- **id** untuk memperbarui posisi saat ini yang ditampilkan di halaman web
- Dan terakhir, atribut **onchange** untuk memanggil fungsi servo untuk mengirim permintaan HTTP ke ESP32 saat slider bergerak.

8.11 Menambahkan JavaScript ke File HTML

Selanjutnya, Kami perlu menambahkan beberapa kode JavaScript ke file HTML Kami menggunakan tag <script> dan </script>. Cuplikan kode ini memperbarui halaman web dengan posisi sliding saat ini:

```
var slider = document.getElementById("servoSlider");
var servoP = document.getElementById("servoPos");
servoP.innerHTML = slider.value;    slider.oninput
= function() {           slider.value = this.value;
servoP.innerHTML = this.value;     }
```

Dan baris berikutnya membuat permintaan HTTP GET pada alamat IP ESP di jalur URL khusus ini /?value=[SLIDER_POSITION]&.

```
$ . ajaxSetup ({ batas waktu : 1000 });
```

```
fungsi servo ( pos ) {
```

```
    $ . dapatkan ( "/?nilai=" + pos + "&" );
```

```
}
```

Misalnya, saat bilah geser berada di **0**, maka membuat permintaan HTTP GET di URL berikut:

```
http://192.168.1.135/?nilai= 0&
```

Dan ketika sliding berada pada **180** derajat, Kami akan memiliki sesuatu sebagai berikut:

```
http://192.168.1.135/?nilai= 180&
```

Dengan cara ini, ketika ESP32 menerima permintaan GET, ia dapat mengambil parameter nilai di URL dan memindahkan motor servo ke posisi yang tepat.

8.12 Kode

Sekarang, kita perlu memasukkan teks HTML sebelumnya ke dalam sketsa dan memutar servo sesuai dengannya. Sketsa berikutnya melakukan hal itu dengan tepat.

Salin kode berikut ke Arduino IDE Kami, tetapi jangan mengunggahnya dulu. Pertama, kita akan melihat sekilas cara kerjanya.

```
*****
Rui Santos
Complete project details at https://randomnerdtutorials.com
*****/
#include <WiFi.h>
#include <Servo.h>
Servo myservo; // create servo object to control a servo
// twelve servo objects can be created on most boards

// GPIO the servo is attached to
static const int servoPin = 13;
// Replace with your network credentials
const char* ssid   = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
// Set web server port number to 80
WiFiServer server(80);
// Variable to store the HTTP request
String header;
// Decode HTTP GET value
String valueString = String(5);
```

```

int pos1 = 0;
int pos2 = 0;
// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;
void setup() {
    Serial.begin(115200);
        myservo.attach(servoPin); // attaches the servo on the servoPin
to the servo object
    // Connect to Wi-Fi network with SSID and password
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    // Print local IP address and start web server
    Serial.println("");
    Serial.println("WiFi connected.");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    server.begin();
}
void loop(){
    WiFiClient client = server.available(); // Listen for incoming clients
    if (client) { // If a new client connects,
        currentTime = millis();
        previousTime = currentTime;
            Serial.println("New Client."); // print a message out in
the serial port
        String currentLine = ""; // make a String to hold
incoming data from the client
            while (client.connected() && currentTime - previousTime <=
timeoutTime) { // loop while the client's connected
                currentTime = millis();
                if (client.available()) { // if there's bytes to read from
the client,
                    char c = client.read(); // read a byte, then

```

```

Serial.write(c);           // print it out the serial monitor
header += c;
if (c == '\n') {          // if the byte is a newline character
    // if the current line is blank, you got two newline
    // characters in a row.
    // that's the end of the client HTTP request, so send a response:
    if (currentLine.length() == 0) {
        // HTTP headers always start with a response code (e.g.
        // HTTP/1.1 200 OK)
        // and a content-type so the client knows what's coming,
        then a blank line:
        client.println("HTTP/1.1 200 OK");
        client.println("Content-type:text/html");
        client.println("Connection: close");
        client.println();
        // Display the HTML web page
        client.println("<!DOCTYPE html><html>");
        client.println("<head><meta name=\"viewport\""
content="width=device-width, initial-scale=1\"");
        client.println("<link rel=\"icon\" href=\"data:,\"");
        // CSS to style the on/off buttons
        // Feel free to change the background-color and font-size
        attributes to fit your preferences
        client.println("<style>body { text-align: center;
font-family: \"Trebuchet MS\", Arial; margin-left:auto;
margin-right:auto; }");
        client.println(".slider { width: 300px; }</style>");
        client.println("<script"
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>");

        // Web Page
        client.println("</head><body><h1>ESP32 with Servo</h1>");
        client.println("<p>Position: <span"
id="servoPos"></span></p>");
        client.println("<input type=\"range\" min=\"0\""
max="180" class="slider" id="servoSlider"
onchange="servo(this.value)" value="" + valueString + "/>");
        client.println("<script>var slider ="
document.getElementById("servoSlider");
        client.println("var servoP ="
document.getElementById("servoPos"); servoP.innerHTML =
slider.value;" );

```

```

        client.println("slider.oninput = function() { slider.value =
this.value; servoP.innerHTML = this.value; }");
        client.println("$.ajaxSetup({timeout:1000}); function
servo(pos) { ");
        client.println("$.get('/?value=" + pos + "&');
{Connection: close};}</script>");
        client.println("</body></html>");
//GET /?value=180& HTTP/1.1
        if(header.indexOf("GET /?value=")>=0) {
            pos1 = header.indexOf('=');
            pos2 = header.indexOf('&');
            valueString = header.substring(pos1+1, pos2);
//Rotate the servo
            myservo.write(valueString.toInt());
            Serial.println(valueString);
        }
// The HTTP response ends with another blank line
        client.println();
// Break out of the while loop
        break;
} else { // if you got a newline, then clear currentLine
    currentLine = "";
}
} else if (c != '\r') { // if you got anything else but a
carriage return character,
    currentLine += c; // add it to the end of the currentLine
}
}
}
// Clear the header variable
header = "";
// Close the connection
client.stop();
Serial.println("Client disconnected.");
Serial.println("");
}
}

```

8.13 Cara Kerja Kode

Pertama, kami menyertakan perpustakaan Servo, dan membuat objek servo yang disebut `myservo`.

```
#include <Servo.h>
```

```
Servo myservo; // create servo object to control a servo
```

Kami juga membuat variabel untuk menyimpan nomor GPIO yang terhubung dengan servo. Pada kasus ini, **GPIO 13**.

```
const int servoPin = 13;
```

Jangan lupa bahwa untuk memodifikasi dua baris berikut untuk menyertakan kredensial jaringan Kami.

```
// Replace with your network credentials const char* ssid = ""; const char* password = "";
```

Kemudian, buat beberapa variabel yang akan digunakan untuk mengekstrak posisi slider dari permintaan HTTP.

```
// Decode HTTP GET value String valueString = String(5); int pos1 = 0; int pos2 = 0;
```

- **setup()**

Dalam `setup()`, Kami harus memasang servo ke GPIO yang terhubung dengannya, dengan `myservo.attach()`.

```
myservo.attach(servoPin); // attaches the servo on the servoPin to the servo object
```

- **loop()**

Bagian pertama dari `loop()` membuat server web dan mengirimkan teks HTML untuk menampilkan halaman web. Bagian kode dibawah ini mengambil nilai slider dari permintaan HTTP.

```
//GET /?value=180& HTTP/1.1 if ( header . indexOf ( "GET /?value=" ) >= 0 ) { pos1 = header . indeksDari ( '=' ); pos2 = judul . indeksDari ( '&' ); nilaiString = header . substring ( pos1 + 1 , pos2 );
```

Saat Kami memindahkan sliding, Kami membuat permintaan HTTP pada URL berikut, yang berisi posisi sliding antara tKami = dan &.

```
http://your-esp-ip-address/?value=[SLIDER_POSITION] &
```

Nilai posisi sliding disimpan di variabel `valueString`. Kemudian, atur servo ke posisi tertentu menggunakan `myservo.write()` dengan variabel `valueString` sebagai argumen. Variabel `valueString` adalah string, jadi kita perlu menggunakan metode `ToInt()` untuk mengubahnya menjadi bilangan bulat – tipe data yang diterima oleh metode `write()`.

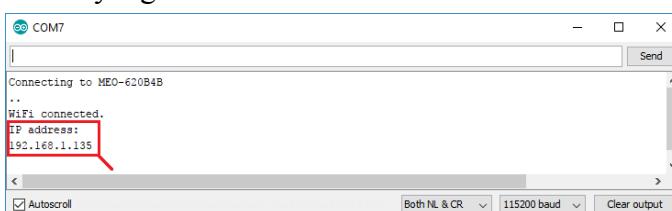
```
myservo.write(valueString.toInt());
```

8.14 Menguji Server Web

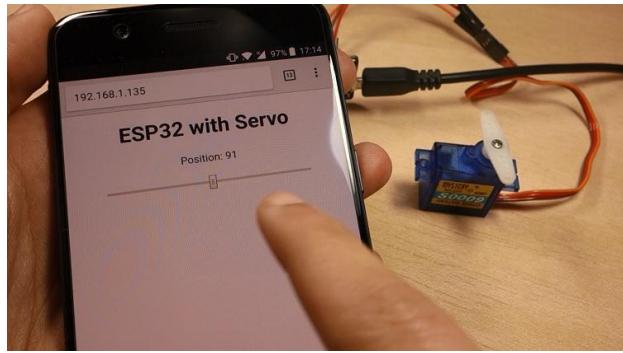
Sekarang kita dapat mengunggah kode ke ESP32, pastikan telah memilih papan dan port COM yang tepat. Juga jangan lupa untuk mengubah kode untuk memasukkan kredensial jaringan. Setelah mengupload kode, buka Serial Monitor dengan baud rate 115200.



Tekan tombol "Enable" ESP32 untuk memulai ulang board, dan salin alamat IP ESP32 yang muncul di Serial Monitor.



Buka browser, rekatkan alamat IP ESP, kemudian akan terlihat halaman web yang telah dibuat sebelumnya. Gerakkan slider untuk mengontrol motor servo.



Di Serial Monitor, Kami juga dapat melihat permintaan HTTP yang Kami kirim ke ESP32 saat Kami menggerakkan slider.

```
COM7
|
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/5
Referer: http://192.168.1.135/
Accept-Encoding: gzip, deflate
Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7
137
Client disconnected.

New Client.
HTTP/1.1 200 OK
Host: 192.168.1.135
Connection: keep-alive
Accept: */*
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0: Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/5
Referer: http://192.168.1.135/
Accept-Encoding: gzip, deflate
Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7
28
Client disconnected.
<
 Autoscroll
Both NL & CR. 115200 baud  Clear output
```

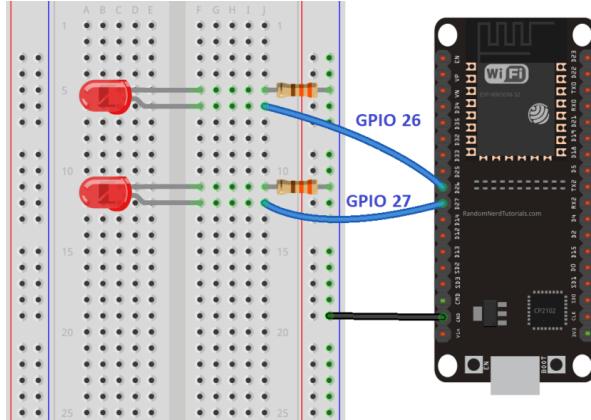
Percobaan dengan server web bertujuan untuk melihat apakah program tersebut telah bekerja dengan benar atau belum.

BAB 9. Esp32 Output WebServer

9.1 Skema

Program dimulai dengan membangun sirkuit. Hubungkan dua LED ke ESP32 seperti yang ditunjukkan pada diagram skema berikut – satu LED terhubung ke **GPIO 26**, dan yang lainnya ke **GPIO 27**.

Catatan : Kami menggunakan board ESP32 DEVKIT DOIT dengan 36 pin. Sebelum merakit sirkuit, pastikan Kami memeriksa pinout untuk papan yang Kami gunakan.



9.2 Kode Server Web ESP32

Berikut merupakan kode untuk membuat web server ESP32. Salin kode berikut ke Arduino IDE, tetapi jangan mengunggahnya dulu karena diperlukan beberapa perubahan sebelum diunggah.

```

/*****
 * Rui Santos
 * Complete project details at
 * https://randomnerdtutorials.com
 ****/
// Load Wi-Fi library
#include <WiFi.h>
// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password =
"REPLACE_WITH_YOUR_PASSWORD";
// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Auxiliar variables to store the current
output state
String output26State = "off";
String output27State = "off";

// Assign output variables to GPIO pins
const int output26 = 26;
const int output27 = 27;

```

```

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds
(example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
    Serial.begin(115200);
    // Initialize the output variables as
outputs
    pinMode(output26, OUTPUT);
    pinMode(output27, OUTPUT);
    // Set outputs to LOW
    digitalWrite(output26, LOW);
    digitalWrite(output27, LOW);

    // Connect to Wi-Fi network with SSID and
password
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    // Print local IP address and start web
server
    Serial.println("'");
    Serial.println("WiFi connected.");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    server.begin();
}

void loop() {
    WiFiClient client = server.available(); // 
Listen for incoming clients

    if (client) { // 
If a new client connects,
        currentTime = millis();

```

```

        previousTime = currentTime;
        Serial.println("New Client."); // print a message out in the serial port
        String currentLine = ""; // make a String to hold incoming data from the client
        while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while the client's connected
            currentTime = millis();
            if (client.available()) { // if there's bytes to read from the client,
                char c = client.read(); // read a byte, then
                Serial.write(c); // print it out the serial monitor
                header += c;
                if (c == '\n') { // if the byte is a newline character
                    // if the current line is blank, you got two newline characters in a row.
                    // that's the end of the client HTTP request, so send a response:
                    if (currentLine.length() == 0) {
                        // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
                        // and a content-type so the client knows what's coming, then a blank line:
                        client.println("HTTP/1.1 200 OK");
                        client.println("Content-type:text/html");
                        client.println("Connection: close");
                        client.println();
                    }
                    // turns the GPIOs on and off
                    if (header.indexOf("GET /26/on") >= 0) {
                        Serial.println("GPIO 26 on");
                        output26State = "on";
                        digitalWrite(output26, HIGH);
                    } else if (header.indexOf("GET /26/off") >= 0) {
                        Serial.println("GPIO 26 off");
                    }
                }
            }
        }
    }
}

```

```

        output26State = "off";
        digitalWrite(output26, LOW);
    } else if (header.indexOf("GET
/27/on") >= 0) {
    Serial.println("GPIO 27 on");
    output27State = "on";
    digitalWrite(output27, HIGH);
} else if (header.indexOf("GET
/27/off") >= 0) {
    Serial.println("GPIO 27 off");
    output27State = "off";
    digitalWrite(output27, LOW);
}

// Display the HTML web page
client.println("<!DOCTYPE
html><html>");
client.println("<head><meta
name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");
client.println("<link rel=\"icon\""
href="data:,>");
// CSS to style the on/off buttons
// Feel free to change the
background-color and font-size attributes to fit
your preferences
client.println("<style>html {
font-family: Helvetica; display: inline-block;
margin: 0px auto; text-align: center;}");
client.println(".button {
background-color: #4CAF50; border: none; color:
white; padding: 16px 40px;}");
client.println("text-decoration:
none; font-size: 30px; margin: 2px; cursor:
pointer;}");
client.println(".button2
{background-color: #555555;}</style></head>");

// Web Page Heading
client.println("<body><h1>ESP32
Web Server</h1>" );

```

```

                                // Display current state, and
ON/OFF buttons for GPIO 26
        client.println("<p>GPIO 26 - State
" + output26State + "</p>");
                                // If the output26State is off, it
displays the ON button
        if (output26State=="off") {
            client.println("<p><a
href=\"/26/on\"><button
class=\"button\">ON</button></a></p>");

        } else {
            client.println("<p><a
href=\"/26/off\"><button
class=\"button
button2\">OFF</button></a></p>");

        }
    }

                                // Display current state, and
ON/OFF buttons for GPIO 27
        client.println("<p>GPIO 27 - State
" + output27State + "</p>");
                                // If the output27State is off, it
displays the ON button
        if (output27State=="off") {
            client.println("<p><a
href=\"/27/on\"><button
class=\"button\">ON</button></a></p>");

        } else {
            client.println("<p><a
href=\"/27/off\"><button
class=\"button
button2\">OFF</button></a></p>");

        }
        client.println("</body></html>");

    }

                                // The HTTP response ends with
another blank line
        client.println();
        // Break out of the while loop
        break;
    } else { // if you got a newline,
then clear currentLine
        currentLine = "";
    }
}

```

```

        } else if (c != '\r') { // if you got
anything else but a carriage return character,
            currentLine += c; // add it to
the end of the currentLine
        }
    }
}

// Clear the header variable
header = "";
// Close the connection
client.stop();
Serial.println("Client disconnected.");
Serial.println("");
}

}

```

9.3 Mengatur Kredensial Jaringan

Kita perlu mengubah baris berikut dengan kredensial jaringan Kami: SSID dan kata sandi. Kode dikomentari dengan baik di mana Kami harus membuat perubahan.

```

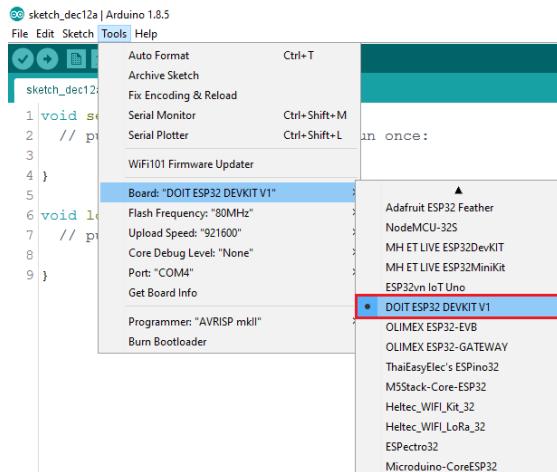
// Replace with your network credentials
const char* ssid =
"REPLACE_WITH_YOUR_SSID";
const char* password =
"REPLACE_WITH_YOUR_PASSWORD";

```

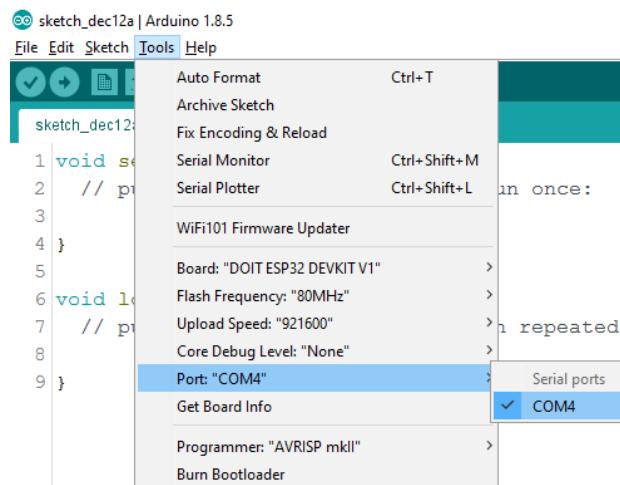
9.4 Mengunggah Kode

Sekarang, Kami dapat mengunggah kode dan server web akan langsung bekerja. Ikuti langkah selanjutnya untuk mengunggah kode ke ESP32:

1. Pasang papan ESP32 di komputer;
2. Di Arduino IDE pilih board Kami di **Tools > Board** (dalam kasus kami, kami menggunakan board ESP32 DEVKIT DOIT);



3. Pilih port COM di Tools > Port .



4. Tekan tombol **Upload** di Arduino IDE dan tunggu beberapa detik sementara kode dikompilasi dan diunggah.



5. Tunggu pesan “ Done uploading ”.

```

Done uploading
Writing at 0x0004c000... (84 %)
Writing at 0x00050000... (89 %)
Writing at 0x00054000... (94 %)
Writing at 0x00058000... (100 %)
Wrote 481440 bytes (299651 compressed) at 0x00010000 in 4.7 seconds
Hash of data verified.
Compressed 3072 bytes to 122...

Writing at 0x00008000... (100 %)
Wrote 3072 bytes (122 compressed) at 0x00008000 in 0.0 seconds (e
Hash of data verified.

Leaving...
Hard resetting...

```

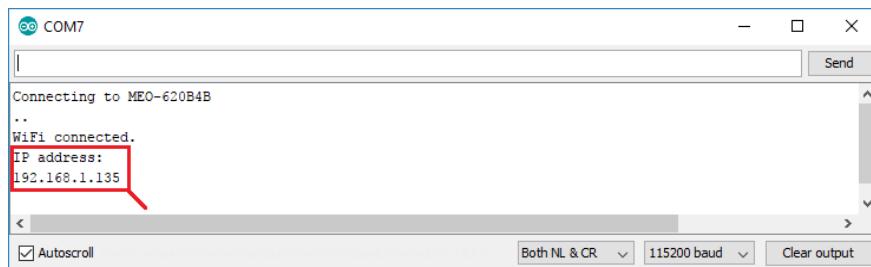
DOIT ESP32 DEVKIT V1, 80MHz, 921600, None on COM4

9.5 Menemukan Alamat IP ESP

Setelah mengupload kode, buka Serial Monitor dengan baud rate 115200.



Tekan tombol ESP32 EN (reset). ESP32 terhubung ke Wi-Fi, dan menampilkan alamat IP ESP di Serial Monitor. Salin alamat IP itu, karena akan diperlukan untuk mengakses server web ESP32.



9.6 Mengakses Server Web

Untuk mengakses server web, buka browser Kami, tempel alamat IP ESP32, dan Kami akan melihat halaman berikut. Misal pada contoh dibawah ini adalah **192.168.1.135**.

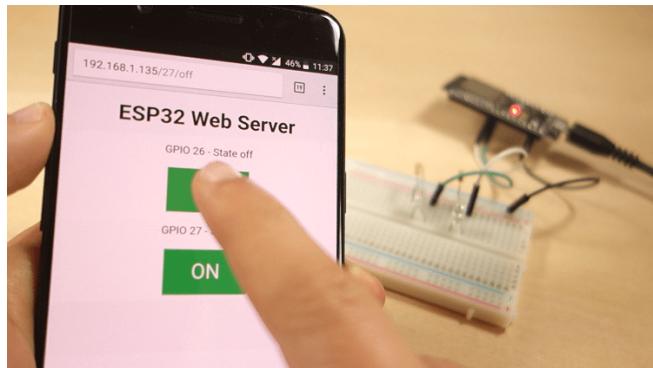


Jika terdapat Serial Monitor, maka dapat melihat apa yang terjadi di latar belakang. ESP menerima permintaan HTTP dari klien baru (browser yang digunakan).



9.7 Menguji Server Web

Sekarang kita dapat menguji apakah server web berfungsi dengan baik dengan cara klik tombol untuk mengontrol LED.



Pada saat yang sama, kita dapat melihat Serial Monitor untuk melihat apa yang terjadi di latar belakang. Misalnya, ketika mengklik tombol untuk memutar **GPIO 26 ON**, ESP32 menerima permintaan pada URL **/26/on**.

```
COM7
|
New Client
GET /26/on HTTP/1.1
Host: 192.168.1.135
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://192.168.1.135/
Accept-Encoding: gzip, deflate
Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7
GPIO 26 on
Client disconnected.

< >
Autoscroll Both NL & CR 115200 baud Clear output
```

Ketika ESP32 menerima permintaan itu, ternyata LED terpasang ke **GPIO 26 ON** dan memperbarui statusnya di halaman web.



Tombol untuk **GPIO 27** bekerja dengan cara yang serupa.

9.8 Cara Kerja Kode

Pada bagian ini akan melihat lebih dekat pada kode untuk melihat cara kerjanya.

Hal pertama yang perlu Kami lakukan adalah memasukkan WiFi library.

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

Seperti disebutkan sebelumnya, Kami perlu memasukkan ssid dan kata sandi Kami di baris berikut di dalam tKami kutip gKami.

```
const char* ssid = " ";
const char* password = " ";
```

Kemudian, Kami mengatur server web Kami ke port 80.

```
WiFiServer server(80);
```

Baris berikut membuat variabel untuk menyimpan header permintaan HTTP:

```
String header;
```

Selanjutnya, Kami membuat variabel tambahan untuk menyimpan status keluaran Kami saat ini. Jika Kami ingin menambahkan lebih banyak output dan menyimpan statusnya, Kami perlu membuat lebih banyak variabel.

```
String output26State = "off";
String output27State = "off";
```

Kita juga perlu menetapkan GPIO untuk setiap output Kami. Di sini kami menggunakan **GPIO 26** dan **GPIO 27**. Kami dapat menggunakan GPIOs lain yang sesuai.

```
const int output26 = 26;
const int output27 = 27;
```

- **Setup()**

Sekarang, mari kita masuk ke `setup()`. Pertama, kita memulai komunikasi serial pada baud rate 115200 untuk keperluan debugging.

```
Serial.begin(115200);
```

Kita juga menentukan GPIO sebagai OUTPUT dan menyetelnya ke LOW.

```

// Initialize the output variables as outputs
pinMode(output26, OUTPUT);
pinMode(output27, OUTPUT);

// Set outputs to LOW
digitalWrite(output26, LOW);
digitalWrite(output27, LOW);

Baris berikut memulai koneksi Wi-Fi dengan
WiFi.begin(ssid, kata sandi), tunggu koneksi berhasil dan
cetak alamat IP ESP di Serial Monitor.

```

```

// Connect to Wi-Fi network with SSID and
password
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
// Print local IP address and start web server
Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
server.begin();

```

- **loop()**

Dalam `loop()` kami memprogram apa yang terjadi ketika klien baru membuat koneksi dengan server web.

ESP32 selalu merespon klien yang masuk dengan baris berikut:

```

WiFiClient client = server.available(); // 
Listen for incoming clients

```

Ketika permintaan diterima, selanjutnya menyimpan data yang masuk. Kemudian salin kode berikut ini.

```

if (client) { // If a new client connects,

```

```

        Serial.println("New Client."); // print a
message out in the serial port
        String currentLine = ""; // make a String to
hold incoming data from the client
        while (client.connected()) { // loop while
the client's connected
            if (client.available()) { // if there's
bytes to read from the client,
                char c = client.read(); // read a byte,
then
                    Serial.write(c); // print it out the
serial monitor
                header += c;
                if (c == '\n') { // if the byte is a
newline character
                    // if the current line is blank, you got
two newline characters in a row.
                    / that's the end of the client HTTP
request, so send a response:
                    if (currentLine.length() == 0) {
                        // HTTP headers always start with a
response code (e.g. HTTP/1.1 200 OK)
                        // and a content-type so the client
knows what's coming, then a blank line:
                        client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");
client.println("Connection: close");
client.println();

```

Bagian selanjutnya dari pernyataan if dan else memeriksa tombol mana yang ditekan di halaman web Kami, dan mengontrol output yang sesuai. Seperti yang telah kita lihat sebelumnya, kita membuat permintaan pada URL yang berbeda tergantung pada tombol yang ditekan.

```

// turns the GPIOs on and off
if (header.indexOf("GET /26/on") >= 0) {
    Serial.println("GPIO 26 on");
    output26State = "on";
    digitalWrite(output26, HIGH);
} else if (header.indexOf("GET /26/off") >= 0)
{
    Serial.println("GPIO 26 off");
}

```

```

        output26State = "off";
        digitalWrite(output26, LOW);
    } else if (header.indexOf("GET /26/on") >= 0)
    {
        Serial.println("GPIO 26 on");
        output26State = "on";
        digitalWrite(output26, HIGH);
    } else if (header.indexOf("GET /26/off") >= 0)
    {
        Serial.println("GPIO 26 off");
        output26State = "off";
        digitalWrite(output26, LOW);
    }
}

```

Misalnya, jika telah menekan tombol GPIO 26 ON, dan ESP32 menerima permintaan pada **URL /26/ON** (kita dapat melihat informasi tersebut pada header HTTP di Serial Monitor). Jadi, kita dapat memeriksa apakah header berisi ekspresi **GET /26/on**. Jika berisi, kami mengubah variabel `output26state` ke ON, dan ESP32 menyalaikan LED.

9.9 Menampilkan halaman web HTML

Hal berikutnya yang perlu dilakukan adalah membuat halaman web. ESP32 akan mengirimkan respons ke browser dengan beberapa kode HTML untuk membangun halaman web. Halaman web dikirim ke klien menggunakan ekspresi `klien.println()`.

Hal pertama yang harus kita kirim selalu baris berikut, yang menunjukkan bahwa kita mengirim HTML.

```
<!DOCTYPE HTML><html>
```

Kemudian, baris berikut membuat halaman web responsive di browser web apa pun.

```
client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
```

Dan berikut ini digunakan untuk mencegah request pada favicon. - Kami tidak perlu khawatir tentang baris ini.

```
client.println("<link rel=\"icon\" href=\"data:, \">");
```

9.10 Menata Halaman Web

Selanjutnya, kita memiliki beberapa teks CSS untuk menata tombol dan tampilan halaman web. Kami memilih font Helvetica, menentukan konten yang akan ditampilkan sebagai blok dan disejajarkan di tengah.

```
    client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center; }");
```

Kami menata tombol kami dengan warna #4CAF50, tanpa batas, teks dalam warna putih, dan dengan padding ini: 16px 40px. Kami juga mengatur dekorasi teks ke none, menentukan ukuran font, margin, dan kurSOR ke pointer.

```
    client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px; }");

    client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer; }");
```

Kami juga mendefinisikan gaya untuk tombol kedua, dengan semua properti tombol yang telah kami definisikan sebelumnya, tetapi dengan warna yang berbeda. Ini akan menjadi gaya untuk tombol off.

```
    client.println(".button2 {background-color: #555555;}</style></head>");
```

9.11 Mengatur Judul Pertama Halaman Web

Di baris berikutnya Kami dapat mengatur judul pertama halaman web Kami. Di sini kami memiliki "**ESP32 Web Server**", tetapi Kami dapat mengubah teks ini menjadi apa pun yang Kami suka.

```
// Web Page Heading
client.println("<h1>ESP32 Web Server</h1>");
```

9.12 Menampilkan Tombol dan Status yang Sesuai

Kemudian, Kami menulis paragraf untuk menampilkan **GPIO 26** kondisi saat ini. Seperti yang Kami lihat, kami menggunakan `output26State` variabel, sehingga status diperbarui secara instan ketika variabel ini berubah.

```
    client.println("<p>GPIO 26 - State " +
output26State + "</p>");
```

Kemudian, kami menampilkan tombol on atau off, tergantung pada kondisi GPIO saat ini. Jika keadaan GPIO saat ini mati, kami menunjukkan tombol ON, jika tidak, kami menampilkan tombol OFF.

```
if (output26State=="off") {
```

```

        client.println("<p><a href=\"/26/on\"><button class=\"button\">ON</button></a></p>") ;
    } else {
        client.println("<p><a href=\"/26/off\"><button button2\">OFF</button></a></p>") ;
    }
}

```

Gunakan prosedur yang sama untuk **GPIO 27**.

9.13 Menutup Koneksi

Akhirnya, ketika respons berakhir, kami menghapus header variabel, dan hentikan koneksi dengan klien dengan `client.stop()`.

```

// Clear the header variable
header = "";
// Close the connection

client.stop();

```

BAB 10. Esp32 WebServer with Slider



- ESP32 host server web yang menampilkan halaman web dengan slider;
- Saat memindahkan slider, hal itu berarti membuat permintaan HTTP ke ESP32 dengan nilai slider baru;
- Permintaan HTTP datang dalam format berikut:
GET/slайдер?значение=SLIDERVALUE, где SLIDERVALUE
является числом между 0 и 255. Мы можем изменить диапазон
перемены для включения других;
- Из HTTP-запроса ESP32 получает значение слайдера в этот момент;
- ESP32 корректирует цикл работы таймера PWM в соответствии с значением слайдера;

- Ini berguna untuk mengontrol kecerahan LED (seperti yang akan kita lakukan dalam contoh ini), motor servo, pengaturan nilai ambang batas, atau aplikasi lain.

10.1 Arduino IDE

Kami akan memprogram papan ESP32 menggunakan Arduino IDE, jadi sebelum melanjutkan dengan tutorial ini, pastikan telah menginstal papan ESP32 di Arduino.

10.2 Async Web Server Libraries

Kami akan membangun server web menggunakan libraries berikut:

- [ESPAsyncWebServer](#)
- [AsyncTCP](#)

Library tersebut tidak tersedia untuk diinstall melalui Arduino Library Manager, kita perlu menyalin file library ke folder Library Instalasi Arduino. Atau di Arduino IDE dengan cara **Sketch > Include Library > Add .zip Library** dan pilih library yang baru saja Kami unduh.

10.3 Kode

Kode berikut mengontrol kecerahan LED bawaan ESP32 menggunakan slider di server web. Dengan kata lain, Kami dapat mengubah siklus tugas PWM dengan slider. Ini dapat berguna untuk mengontrol kecerahan LED atau mengontrol motor servo, misalnya salin kode ke Arduino IDE Kami. Masukkan kredensial jaringan Kami dan kode akan langsung berfungsi.

```

/ *****
Rui Santos
Complete project details at
https://RandomNerdTutorials.com/esp32-web-server-
slider-pwm/

Permission is hereby granted, free of
charge, to any person obtaining a copy
of this software and associated
documentation files.

The above copyright notice and this
permission notice shall be included in all
copies or substantial portions of the
Software.

*****/

```

```

// Import required libraries
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>

// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password =
"REPLACE_WITH_YOUR_PASSWORD";

const int output = 2;

String sliderValue = "0";

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

const char* PARAM_INPUT = "value";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

const char index_html[] PROGMEM =
R"rawliteral(
<!DOCTYPE HTML><html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>ESP Web Server</title>
        <style>
            html {font-family: Arial; display: inline-block; text-align: center;}
            h2 {font-size: 2.3rem; margin: 0px auto; padding-bottom: 25px; }
            .slider { -webkit-appearance: none; margin: 14px; width: 360px; height: 25px; background: #FFD65C; outline: none; -webkit-transition: .2s; transition: opacity .2s; }
            p {font-size: 1.9rem; }
            body {max-width: 400px; margin: 0px auto; padding-bottom: 25px; }
        </style>
)</head>
<body>
    <h2>ESP Web Server</h2>
    <p>Control the LED</p>
    <input type="range" value="0" min="0" max="100" step="1" class="slider" oninput="updateSliderValue()"/>
    <script>
        function updateSliderValue() {
            const sliderValue = document.querySelector('.slider').value;
            document.getElementById('sliderValue').innerHTML = sliderValue;
        }
    </script>
</body>
)
```

```

        .slider::-webkit-slider-thumb
{-webkit-appearance: none; appearance: none;
width: 35px; height: 35px; background: #003249;
cursor: pointer;}
        .slider::-moz-range-thumb { width: 35px;
height: 35px; background: #003249; cursor:
pointer; }
    </style>
</head>
<body>
    <h2>ESP Web Server</h2>
        <p><span
id="textSliderValue">%SLIDERVALUE%</span></p>
        <p><input type="range"
onchange="updateSliderPWM(this)" id="pwmSlider"
min="0" max="255" value="%SLIDERVALUE%" step="1"
class="slider"></p>
    <script>
        function updateSliderPWM(element) {
            var sliderValue =
document.getElementById("pwmSlider").value;

document.getElementById("textSliderValue").innerHTML =
sliderValue;
            console.log(sliderValue);
            var xhr = new XMLHttpRequest();
            xhr.open("GET",
"/slider?value="+sliderValue, true);
            xhr.send();
        }
    </script>
</body>
</html>
) rawliteral";

// Replaces placeholder with button section
in your web page
String processor(const String& var) {
//Serial.println(var);
if (var == "SLIDERVALUE") {
    return sliderValue;
}
return String();
}

```

```

    }

void setup() {
    // Serial port for debugging purposes
    Serial.begin(115200);

    // configure LED PWM functionalitites
    ledcSetup(ledChannel, freq, resolution);

    // attach the channel to the GPIO to be
controlled
    ledcAttachPin(output, ledChannel);

    ledcWrite(ledChannel,
sliderValue.toInt());

    // Connect to Wi-Fi
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi..");
    }

    // Print ESP Local IP Address
    Serial.println(WiFi.localIP());

    // Route for root / web page
    server.on("/", HTTP_GET,
[])(AsyncWebRequest *request) {
    request->send_P(200, "text/html",
index_html, processor);
}

        // Send a GET request to
<ESP_IP>/slider?value=<inputMessage>
server.on("/slider", HTTP_GET, []
(AsyncWebRequest *request) {
    String inputMessage;
    // GET input1 value on
<ESP_IP>/slider?value=<inputMessage>
    if (request->hasParam(PARAM_INPUT)) {
        inputMessage = request->getParam(PARAM_INPUT)->value();
    }
}
}

```

```

        sliderValue = inputMessage;
        ledcWrite(ledChannel,
sliderValue.toInt());
    }
    else {
        inputMessage = "No message sent";
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
} );

// Start server
server.begin();
}

void loop() {
}

```

10.4 Cara Kerja Kode

- **Mengimpor perpustakaan**

Pertama, impor perpustakaan yang diperlukan. `WiFi`, `ESPAsyncWebServer` dan `ESPAsyncTCP` diperlukan untuk membangun server web.

```
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
```

- **Mengatur kredensial jaringan Kami**

Masukkan kredensial jaringan Kami di variabel berikut, sehingga ESP32 dapat terhubung ke jaringan lokal Kami.

```
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

- **Definisi variabel**

Digunakan untuk mengontrol kecerahan LED bawaan ESP32. LED bawaan sesuai dengan [GPIO 2](#). Simpan GPIO yang ingin dikontrol output variabel. Variabel `sliderValue` akan menahan nilai slider. Pada awalnya, itu diatur ke nol.

```
String sliderValue = "0";
```

- **Setel Properti PWM**

Baris berikut mendefinisikan properti PWM untuk mengontrol LED.

```
// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;
```

Pada kasus kali ini akan digunakan resolusi 8-bit, yang berarti dapat mengontrol kecerahan LED menggunakan nilai dari 0 hingga 255.

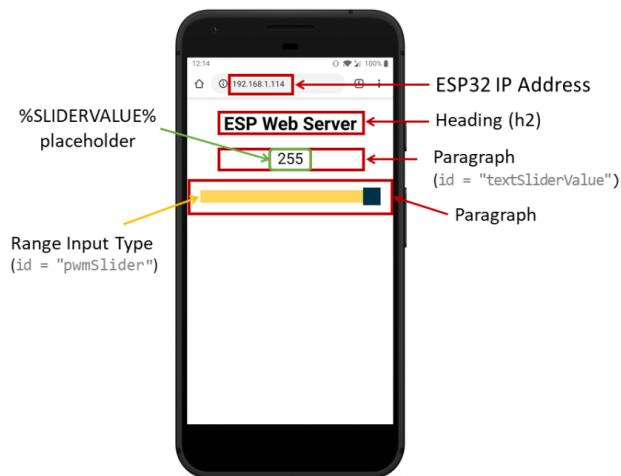
- **Input Parameters**

Variabel `PARAM_INPUT` akan digunakan untuk "mencari" nilai slider pada permintaan yang diterima oleh ESP32 saat slider dipindahkan. (Ingat: ESP32 akan menerima permintaan seperti ini: `GET/slider?value=SLIDERVALUE`)

```
const char* PARAM_INPUT = "value";
```

Kode tersebut akan mencari `value` pada URL dan mendapatkan nilai yang diberikan padanya.

- **Membuat Halaman Web**



Halaman web untuk proyek ini cukup sederhana yaitu berisi satu judul, satu paragraf dan satu input dari rentang jenis. Semua teks HTML dengan gaya yang disertakan disimpan di variabel `index_html`.

Tag `<meta>` berikut membuat halaman web Kami responsif di browser apapun.

```
<meta name="viewport" content="width=device-width,  
initial-scale=1">
```

Di antara tag `<title></title>` ada judul server web yang dibuat. Judul adalah teks yang muncul di tab browser web.

- **Styles**

Di antara tag `<style></style>`, kita menambahkan beberapa CSS untuk menata halaman web.

```
<style>  
    html {font-family: Arial; display: inline-block;  
text-align: center;}  
    h2 {font-size: 2.3rem;}  
    p {font-size: 1.9rem;}  
    body {max-width: 400px; margin:0px auto;  
padding-bottom: 25px;}  
    .slider { -webkit-appearance: none; margin: 14px;  
width: 360px; height: 25px; background: #FFD65C;  
outline: none; -webkit-transition: .2s;  
transition: opacity .2s;}  
    .slider::-webkit-slider-thumb  
{-webkit-appearance: none; appearance: none; width:  
35px; height: 35px; background: #003249; cursor:  
pointer;}  
    .slider::-moz-range-thumb { width: 35px; height:  
35px; background: #003249; cursor: pointer; }  
</style>
```

Pada dasarnya, kami mengatur halaman HTML untuk menampilkan teks dengan font Arial di blok tanpa margin, dan rata di tengah.

```
html {font-family: Arial; display: inline-block;  
text-align: center;}
```

Baris berikut mengatur ukuran font untuk heading (h2) dan paragraf (p).

```
h2 {font-size: 2.3rem;}  
p {font-size: 1.9rem;}
```

Setel properti body HTML.

```
body {max-width: 400px; margin:0px auto;  
padding-bottom: 25px;}
```

Baris berikut menyesuaikan slider:

```
.slider { -webkit-appearance: none; margin: 14px;  
width: 360px; height: 25px; background: #FFD65C;  
outline: none; -webkit-transition: .2s;  
transition: opacity .2s;}  
.slider::-webkit-slider-thumb { -webkit-appearance:  
none; appearance: none; width: 35px; height: 35px;  
background: #003249; cursor: pointer; }  
.slider::-moz-range-thumb { width: 35px; height:  
35px; background: #003249; cursor: pointer; }
```

- **HTML Body**

Di dalam tag `<body></body>` adalah tempat kita menambahkan konten halaman web.

Tag `<h2></h2>` menambahkan heading ke halaman web. Dalam hal ini, teks "ESP Web Server", tetapi Kami dapat menambahkan teks lainnya.

```
<h2>ESP Web Server</h2>
```

Paragraf pertama akan berisi nilai slider saat ini. Tag HTML tertentu memiliki id `textSliderValue` yang ditetapkan padanya, sehingga kita dapat merujuknya nanti.

```
<p><span  
id="textSliderValue">%SLIDERVALUE%</span></p>
```

`%SLIDERVALUE%` adalah pengganti untuk nilai slider. Ini akan digantikan oleh ESP32 dengan nilai aktual saat mengirimkannya ke browser. Ini berguna untuk menunjukkan nilai saat ini saat Kami mengakses browser untuk pertama kalinya.

- **Membuat Slider**

Untuk membuat slider di HTML digunakan tag `<input>`. Tag `<input>` menentukan bidang tempat pengguna dapat memasukkan data. Ada berbagai macam jenis input. Untuk menentukan slider, gunakan atribut "`type`" dengan nilai "`range`". Dalam slider, Kami juga perlu menentukan rentang

minimum dan maksimum menggunakan atribut “**min**” dan “**max**” (**dalam hal ini**, 0 dan 255, masing-masing).

```
<p><input type="range"
onchange="updateSliderPWM(this)" id="pwmSlider"
min="0" max="255" value="%SLIDERVALUE%" step="1"
class="slider"></p>
```

Kasus ini juga perlu mendefinisikan atribut lain seperti:

1. Atribut **step** menentukan interval antara angka yang valid . Dalam kasus kami, ini diatur ke 1;
 2. **Class** untuk mengatur gaya slider (`class="slider"`);
 3. **id** untuk memperbarui posisi saat ini yang ditampilkan di halaman web;
 4. Atribut **onchange** untuk memanggil fungsi `(updateSliderPWM(this))` untuk mengirim permintaan HTTP ke ESP32 saat slider bergerak.
- **Menambahkan JavaScript ke File HTML**

Selanjutnya, perlu menambahkan beberapa kode JavaScript ke file HTML dengan menggunakan tag `<script>` dan `</script>`. Kita perlu menambahkan fungsi `updateSliderPWM()` yang akan membuat permintaan ke ESP32 dengan nilai slider saat ini.

```
<script>
function updateSliderPWM(element) {
    var sliderValue =
document.getElementById("pwmSlider").value;

document.getElementById("textSliderValue").innerHTML =
sliderValue;
console.log(sliderValue);
var xhr = new XMLHttpRequest();
xhr.open("GET", "/slider?value=" + sliderValue,
true);
xhr.send();
}

</script>
```

Baris berikutnya ini mendapatkan nilai slider saat ini dengan id-nya dan menyimpannya di variabel `sliderValue` JavaScript. Sebelumnya, kami telah menetapkan id slider ke `pwmSlider`. Jadi, kita mendapatkannya sebagai berikut:

```
var sliderValue =  
document.getElementById("pwmSlider").value;
```

Setelah itu, kita atur label slider (yang id-nya adalah `teksSliderValue`) ke nilai yang disimpan pada variabel `sliderValue`.

Terakhir, buat [HTTP GET request](#).

```
var xhr = new XMLHttpRequest();  
xhr.open("GET", "/slider?value="+sliderValue, true);  
xhr.send();
```

Misalnya, saat slider berada di **0**, kita membuat request HTTP GET di URL berikut:

```
http://ESP-IP-ADDRESS/slider?value=0
```

Dan ketika nilai slider adalah 200, kita akan memiliki request di URL berikut:

```
http://ESP-IP-ADDRESS/slider?value=200
```

Dengan cara ini, ketika ESP32 menerima permintaan GET, ESP32 dapat mengambil parameter nilai di URL dan mengontrol sinyal PWM yang sesuai seperti yang akan kita lihat di bagian selanjutnya.

● Prosesor

Sekarang, kita perlu membuat `prosesor()` function, yang akan menggantikan placeholder dalam teks HTML kita dengan nilai slider saat ini saat Kami mengaksesnya untuk pertama kali di browser.

```
// Replaces placeholder with button section in your  
web page  
String processor(const String& var) {  
    //Serial.println(var);  
    if (var == "SLIDERVALUE") {  
        return sliderValue;  
    }  
    return String();  
}
```

Saat halaman web diminta, kami memeriksa apakah HTML memiliki placeholder. Jika menemukan **%SLIDERVALUE%** placeholder, kami mengembalikan nilai yang disimpan di variabel `sliderValue`.

- **Setup()**

Dalam `setup()`, inisialisasi Serial Monitor untuk keperluan debugging.

```
Serial.begin(115200);
```

Konfigurasikan properti PWM LED yang ditentukan sebelumnya.

```
ledcSetup(ledChannel, freq, resolution);
```

Lampirkan saluran ke GPIO yang ingin Kami kontrol.

```
ledcAttachPin(output, ledChannel);
```

Atur siklus tugas sinyal PWM ke nilai yang disimpan di `sliderValue` (ketika ESP32 dimulai, itu diatur ke 0).

```
ledcWrite(ledChannel, sliderValue.toInt());
```

Hubungkan ke jaringan lokal Kami dan cetak alamat IP ESP32.

```
// Connect to Wi-Fi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
}

// Print ESP Local IP Address
Serial.println(WiFi.localIP());
```

- **Handle Requests**

Terakhir, tambahkan baris kode berikutnya untuk menangani server web.

```
// Route for root / web page
server.on("/", HTTP_GET, [] (AsyncWebServerRequest
*request) {
```

```

        request->send_P(200, "text/html", index_html,
processor);
} );

// Send a GET request to
<ESP_IP>/slider?value=<inputMessage>
server.on("/slider", HTTP_GET, [] (AsyncWebServerRequest *request) {
    String inputMessage;
    // GET input1 value on
<ESP_IP>/slider?value=<inputMessage>
    if (request->hasParam(PARAM_INPUT)) {
        inputMessage =
request->getParam(PARAM_INPUT)->value();
        sliderValue = inputMessage;
        ledcWrite(ledChannel, sliderValue.toInt());
    }
    else {
        inputMessage = "No message sent";
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
} );

```

Saat membuat permintaan pada URL root, harus mengirim teks HTML yang disimpan di variabel `index_html`. Kita juga harus melewati `prosesor` function, yang akan menggantikan semua placeholder dengan nilai yang tepat.

```

// Route for root / web page
server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request) {
    request->send_P(200, "text/html", index_html,
processor);
} );

```

Kami membutuhkan penangan lain yang akan menyimpan nilai slider saat ini dan mengatur kecerahan LED yang sesuai.

```

server.on("/slider", HTTP_GET, [] (AsyncWebServerRequest *request) {

```

```

String inputMessage;
    //      GET      input1      value      on
<ESP_IP>/slider?value=<inputMessage>
if (request->hasParam(PARAM_INPUT)) {
    inputMessage =
request->getParam(PARAM_INPUT)->value();
    sliderValue = inputMessage;
    ledcWrite(ledChannel, sliderValue.toInt());
}
else {
    inputMessage = "No message sent";
}
Serial.println(inputMessage);
request->send(200, "text/plain", "OK");
} );

```

Pada dasarnya, kami mendapatkan nilai slider pada baris berikut:

```

if (request->hasParam(PARAM_INPUT)) {
    inputMessage =
request->getParam(PARAM_INPUT)->value();
    sliderValue = inputMessage;

```

Kemudian, perbarui kecerahan LED (siklus tugas PWM) menggunakan `ledcWrite()` fungsi yang menerima sebagai argumen saluran yang ingin Kami kontrol dan nilainya.

```
ledcWrite(ledChannel, sliderValue.toInt());
```

Terakhir, mulai server.

```
server.begin();
```

Karena ini adalah server web asinkron, kami tidak perlu menulis apa pun `dilingkaran()`.

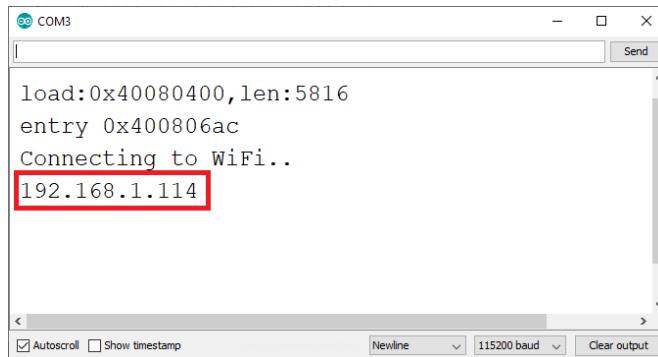
```

void loop() {
}
```

Kurang lebih begitulah cara kerja kode.

- **Unggah Kode**

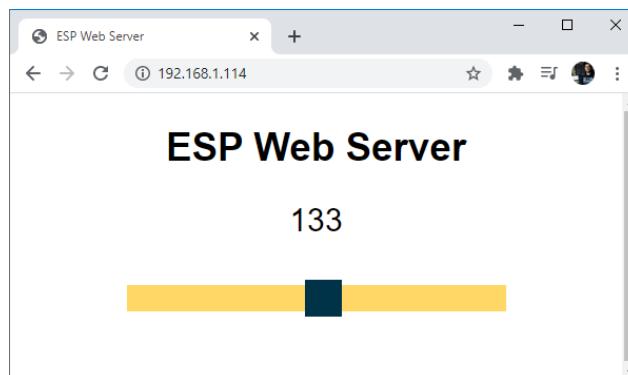
Setelah mengunggah, buka Serial Monitor dengan baud rate 115200. Tekan tombol reset ESP32. Alamat IP ESP32 harus dicetak di monitor serial.



```
load:0x40080400, len:5816
entry 0x400806ac
Connecting to WiFi..
192.168.1.114
```

- **Demonstrasi Server Web**

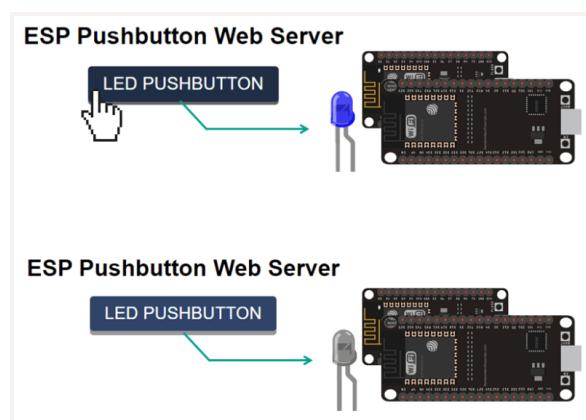
Buka browser dan ketik alamat IP ESP32. Server web Kami harus menampilkan slider dan nilainya saat ini.



Gerak-kan slider dan lihat LED bawaan ESP32 meningkat dan menurun kecerahannya.

BAB 11. Momentary Switch WebServer

Gambar dibawah merupakan gambaran project Esp32 Momentary Switch Webserver



- ESP32 atau ESP8266 host server web yang dapat Kami akses untuk mengontrol output;
- Status default output adalah LOW, tetapi Kami dapat mengubahnya tergantung pada aplikasi proyek Kami;
- Ada tombol yang berfungsi seperti saklar sesaat:
 - jika Kami menekan tombol, output berubah statusnya menjadi TINGGI selama Kami terus menahan tombol;
 - setelah tombol dilepaskan, status output kembali ke LOW.

11.1 Kode

Salin kode berikut ke Arduino IDE

```
/*
  Rui Santos
  Complete project details at
  https://RandomNerdTutorials.com/esp32-esp8266-web-server-outputs-momentary-switch/

  Permission is hereby granted, free of charge, to
  any person obtaining a copy
  of this software and associated documentation
  files.

  The above copyright notice and this permission
  notice shall be included in all
  copies or substantial portions of the Software.
  *****/
#ifndef ESP32
  #include <WiFi.h>
  #include <AsyncTCP.h>
#else
  #include <ESP8266WiFi.h>
  #include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>

// REPLACE WITH YOUR NETWORK CREDENTIALS
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const          char*           password        =
"REPLACE_WITH_YOUR_PASSWORD";

const int output = 2;
```

```
// HTML web page
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
  <head>
    <title>ESP Pushbutton Web Server</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>
      body { font-family: Arial; text-align: center; margin:0px auto; padding-top: 30px; }
      .button {
        padding: 10px 20px;
        font-size: 24px;
        text-align: center;
        outline: none;
        color: #fff;
        background-color: #2f4468;
        border: none;
        border-radius: 5px;
        box-shadow: 0 6px #999;
        cursor: pointer;
        -webkit-touch-callout: none;
        -webkit-user-select: none;
        -khtml-user-select: none;
        -moz-user-select: none;
        -ms-user-select: none;
        user-select: none;
        -webkit-tap-highlight-color: rgba(0,0,0,0);
      }
      .button:hover {background-color: #1f2e45}
      .button:active {
        background-color: #1f2e45;
        box-shadow: 0 4px #666;
        transform: translateY(2px);
      }
    </style>
  </head>
  <body>
    <h1>ESP Pushbutton Web Server</h1>
    <button class="button" onmousedown="toggleCheckbox('on');"
    ontouchstart="toggleCheckbox('on');"
```

```

onmouseup="toggleCheckbox('off');"
ontouchend="toggleCheckbox('off');">LED
PUSHBUTTON</button>
<script>
function toggleCheckbox(x) {
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "/" + x, true);
    xhr.send();
}
</script>
</body>
</html>) rawliteral";

void notFound(AsyncWebServerRequest *request) {
    request->send(404, "text/plain", "Not found");
}

AsyncWebServer server(80);

void setup() {
    Serial.begin(115200);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    if (WiFi.waitForConnectResult() != WL_CONNECTED)
    {
        Serial.println("WiFi Failed!");
        return;
    }
    Serial.println();
    Serial.print("ESP IP Address: http://");
    Serial.println(WiFi.localIP());

    pinMode(output, OUTPUT);
    digitalWrite(output, LOW);

    // Send web page to client
    server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request) {
        request->send_P(200, "text/html", index_html);
    });
}

// Receive an HTTP GET request

```

```

        server.on("/on",      HTTP_GET,      [] )
(AsyncWebRequest *request) {
    digitalWrite(output, HIGH);
    request->send(200, "text/plain", "ok");
} );

// Receive an HTTP GET request
server.on("/off",      HTTP_GET,      [] )
(AsyncWebRequest *request) {
    digitalWrite(output, LOW);
    request->send(200, "text/plain", "ok");
} );

server.onNotFound(notFound);
server.begin();
}

void loop() {
}

```

11.2 Kredensial Jaringan

Masukkan kredensial jaringan pada baris berikut:

```

const char* ssid = "REPLACE_WITH_YOUR_SSID";
const           char*           password           =
"REPLACE_WITH_YOUR_PASSWORD";

```

11.3 Tombol Pengalihan Sesaat

Baris berikut membuat tombol saklar sesaat.

```

<button class="button"
onmousedown="toggleCheckbox('on');" ontouc

```

11.4 HTTP GET Permintaan untuk Mengubah Status Tombol (JavaScript)

Fungsi `toggleCheckbox()`, membuat permintaan HTTP ke ESP32 baik di `/on` atau `/off` URL:

```

function toggleCheckbox(x) {
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "/" + x, true);
    xhr.send();
}

```

11.5 Handle Request

Ketika permintaan diterima di /on URL, aktifkan GPIO (HIGH) seperti yang ditunjukkan di bawah ini:

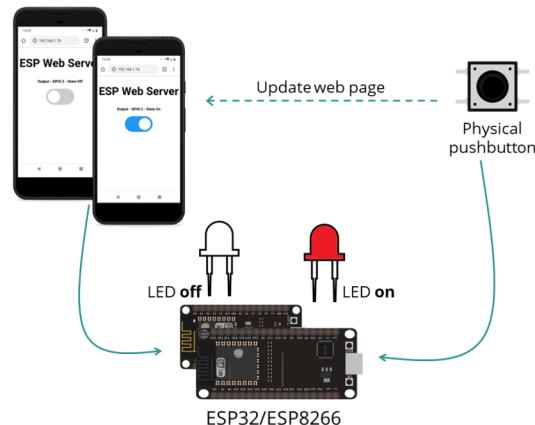
```
server.on("/on", HTTP_GET, [] (AsyncWebServerRequest
*request) {
    digitalWrite(output, HIGH);
    request->send(200, "text/plain", "ok");
});
```

Saat permintaan diterima di /off URL, kami mematikan GPIO (LOW):

```
server.on("/off", HTTP_GET, [] (AsyncWebServerRequest
*request) {
    digitalWrite(output, LOW);
    request->send(200, "text/plain", "ok");
});
```

BAB 12. ESP32 Physical Button Webserver

12.1 Project Overview



- ESP32 atau ESP8266 merupakan hosting server web yang memungkinkan Kami mengontrol status keluaran;
- Status keluaran saat ini ditampilkan di server web;
- ESP juga terhubung ke tombol fisik yang mengontrol output yang sama;
- Jika Kami mengubah status keluaran menggunakan tombol puchs fisik, status saat ini juga diperbarui di server web.

Singkatnya, proyek ini memungkinkan Kami untuk mengontrol output yang sama menggunakan server web dan tombol tekan secara bersamaan. Setiap kali status keluaran berubah, server web diperbarui.

12.2 Schematic Diagram

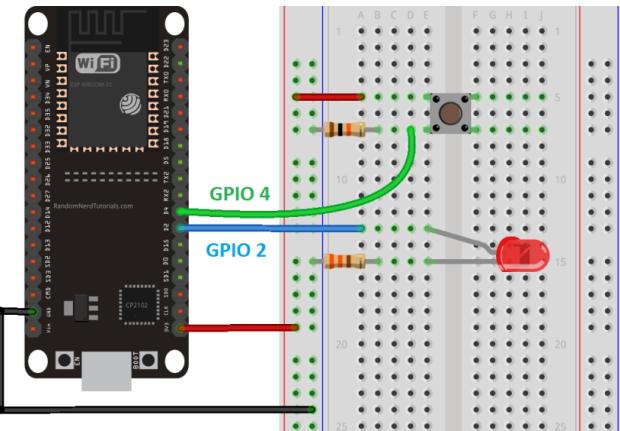
Sebelum melanjutkan, Kami perlu merakit sirkuit dengan LED dan tombol tekan. Kami akan menghubungkan LED ke **GPIO 2** dan tombol tekan ke **GPIO 4**.

12.2.1 Parts Required

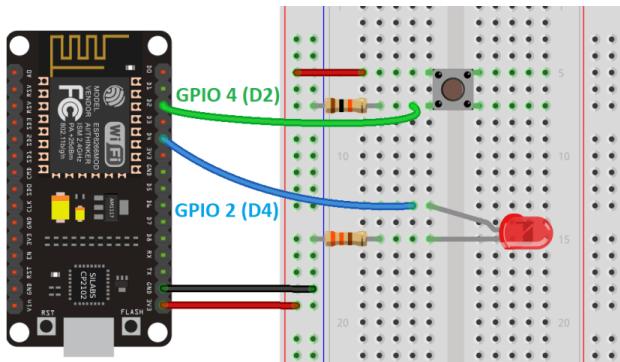
Berikut daftar bagian yang Kami perlukan untuk membangun sirkuit:

- 1) ESP32 (read Best ESP32 Dev Boards) or ESP8266
- 2) 5 mm LED
- 3) 330 Ohm resistor
- 4) Pushbutton
- 5) 10k Ohm resistor
- 6) Breadboard
- 7) Jumper wires

12.2.2 ESP32 Schematic



a. ESP8266 NodeMCU Schematic



12.3 ESP Web Server Code

```
*****
Rui Santos
Complete project details at
https://RandomNerdTutorials.com/esp32-esp8266-web-server-physical-button/
```

The above copyright notice and this permission notice
shall be included in all
copies or substantial portions of the Software.

```
*****/
```

```
// Import required libraries
#ifndef ESP32
    #include <WiFi.h>
    #include <AsyncTCP.h>
#else
    #include <ESP8266WiFi.h>
    #include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>
```

```
// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

const char* PARAM_INPUT_1 = "state";

const int output = 2;
const int buttonPin = 4;

// Variables will change:
int ledState = LOW; // the current state of the
output pin
int buttonState; // the current reading from
the input pin
int lastButtonState = LOW; // the previous reading
from the input pin

// the following variables are unsigned longs because
the time, measured in
// milliseconds, will quickly become a bigger number
than can be stored in an int.
unsigned long lastDebounceTime = 0; // the last time
the output pin was toggled
unsigned long debounceDelay = 50; // the debounce
time; increase if the output flickers

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
    <title>ESP Web Server</title>
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <style>
        html {font-family: Arial; display: inline-block;
text-align: center;}
        h2 {font-size: 3.0rem;}
        p {font-size: 3.0rem;}
        body {max-width: 600px; margin: 0px auto;
padding-bottom: 25px;}
        .switch {position: relative; display: inline-block;
width: 120px; height: 68px}
        .switch input {display: none}
)rawliteral";
```

```

        .slider {position: absolute; top: 0; left: 0; right
0; bottom: 0; background-color: #ccc; border-radius:
34px}
        .slider:before {position: absolute; content: "";
height: 52px; width: 52px; left: 8px; bottom: 8px;
background-color: #fff; -webkit-transition: .4s;
transition: .4s; border-radius: 68px}
            input:checked+.slider {background-color: #2196F3}
            input:checked+.slider:before {-webkit-transform:
translateX(52px); -ms-transform: translateX(52px);
transform: translateX(52px)}
        </style>
</head>
<body>
    <h2>ESP Web Server</h2>
    %BUTTONPLACEHOLDER%
<script>function toggleCheckbox(element) {
    var xhr = new XMLHttpRequest();
    if(element.checked){ xhr.open("GET",
"/update?state=1", true); }
    else { xhr.open("GET", "/update?state=0", true); }
    xhr.send();
}

setInterval(function () {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            var inputChecked;
            var outputStateM;
            if( this.responseText == 1){
                inputChecked = true;
                outputStateM = "On";
            }
            else {
                inputChecked = false;
                outputStateM = "Off";
            }
            document.getElementById("output").checked =
inputChecked;
            document.getElementById("outputState").innerHTML =
outputStateM;
        }
    };
    xhttp.open("GET", "/state", true);
    xhttp.send();
}, 1000 ) ;

```

```
</script>
</body>
</html>
)rawliteral";

// Replaces placeholder with button section in your web
page
String processor(const String& var){
    //Serial.println(var);
    if(var == "BUTTONPLACEHOLDER"){
        String buttons = "";
        String outputStateValue = outputState();
        buttons+= "<h4>Output - GPIO 2 - State <span"
        id=\"outputState\"></span></h4><label"
        class=\"switch\"><input type=\"checkbox\""
        onchange=\"toggleCheckbox(this)\" id=\"output\" " +
        outputStateValue + "><span"
        class=\"slider\"></span></label>";
        return buttons;
    }
    return String();
}

String outputState(){
    if(digitalRead(output)){
        return "checked";
    }
    else {
        return "";
    }
    return "";
}

void setup(){
    // Serial port for debugging purposes
    Serial.begin(115200);

    pinMode(output, OUTPUT);
    digitalWrite(output, LOW);
    pinMode(buttonPin, INPUT);

    // Connect to Wi-Fi
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi..");
    }
}
```

```

// Print ESP Local IP Address
Serial.println(WiFi.localIP());

// Route for root / web page
server.on("/", HTTP_GET, [](AsyncWebServerRequest
*request){
    request->send_P(200, "text/html", index_html,
processor);
});

// Send a GET request to <ESP_IP>/update?state=<inputMessage>
server.on("/update", HTTP_GET, []
(AsyncWebServerRequest *request) {
    String inputMessage;
    String inputParam;
    // GET input1 value on
<ESP_IP>/update?state=<inputMessage>
    if (request->hasParam(PARAM_INPUT_1)) {
        inputMessage =
request->getParam(PARAM_INPUT_1)->value();
        inputParam = PARAM_INPUT_1;
        digitalWrite(output, inputMessage.toInt());
        ledState = !ledState;
    }
    else {
        inputMessage = "No message sent";
        inputParam = "none";
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
});

// Send a GET request to <ESP_IP>/state
server.on("/state", HTTP_GET, []
(AsyncWebServerRequest *request) {
    request->send(200, "text/plain",
String(digitalRead(output)).c_str());
});
// Start server
server.begin();
}

void loop() {
// read the state of the switch into a local variable
int reading = digitalRead(buttonPin);

```

```

// check to see if you just pressed the button
// (i.e. the input went from LOW to HIGH), and you've
waited long enough
// since the last press to ignore any noise:

// If the switch changed, due to noise or pressing:
if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
}

if ((millis() - lastDebounceTime) > debounceDelay) {
    // whatever the reading is at, it's been there for
longer than the debounce
    // delay, so take it as the actual current state:

    // if the button state has changed:
    if (reading != buttonState) {
        buttonState = reading;

        // only toggle the LED if the new button state is
HIGH
        if (buttonState == HIGH) {
            ledState = !ledState;
        }
    }
}

// set the LED:
digitalWrite(output, ledState);

// save the reading. Next time through the loop, it'll
be the lastButtonState:
lastButtonState = reading;
}

```

12.4 How the Code Works

a. Network Credentials

```

const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

```

b. Button State and Output State

Variabel `ledState` menyimpan status keluaran LED. Untuk default, ketika server web dimulai, itu `LOW`.

```
int ledState = LOW; // the current state of the output pin
```

buttonState dan lastButtonState digunakan untuk mendeteksi apakah tombol ditekan atau tidak.

```
int buttonState; // the current reading from the input pin
int lastButtonState = LOW; // the previous reading from the input pin
```

c. Button (web server)

Kami tidak menyertakan HTML untuk membuat tombol pada variabel index_html. Itu karena kami ingin dapat mengubahnya tergantung pada status LED saat ini yang juga dapat diubah dengan tombol. Jadi, kami telah membuat placeholder untuk tombol %BUTTONPLACEHOLDER% yang akan diganti dengan teks HTML untuk membuat tombol nanti pada kode (ini dilakukan di fungsi prosesor()).

```
<h2>ESP Web Server</h2>
%BUTTONPLACEHOLDER%
```

d. Processor()

Fungsi processor() menggantikan placeholder pada teks HTML dengan nilai sebenarnya. Pertama, ia memeriksa apakah teks HTML berisi placeholder %BUTTONPLACEHOLDER%.

```
if(var == "BUTTONPLACEHOLDER") {
```

Kemudian, panggil fungsi outputState() yang mengembalikan status output saat ini. Kami menyimpannya di variabel outputStateValue.

```
String outputStateValue = outputState();
```

Setelah itu, gunakan nilai tersebut untuk membuat teks HTML untuk menampilkan tombol dengan status yang benar:

```
buttons+= "<h4>Output - GPIO 2 - State <span id=\"outputState\">"
```

e. HTTP GET Request to Change Output State (JavaScript)

Saat menekan tombol, fungsi toggleCheckbox() dipanggil. Fungsi ini akan membuat permintaan pada URL yang berbeda untuk menyalakan atau mematikan LED.

```

function toggleCheckbox(element) {
    var xhr = new XMLHttpRequest();
    if(element.checked){ xhr.open("GET", "/update?state=1", true)
    else { xhr.open("GET", "/update?state=0", true); }
    xhr.send();
}

```

Untuk menyalakan LED, itu membuat permintaan pada /update?state=1 URL:

```

if(element.checked){ xhr.open("GET", "/update?state=1", true); }

```

Jika tidak, itu membuat permintaan pada /update?state=0 URL.

f. HTTP GET Request to Update State (JavaScript)

Untuk menjaga status keluaran diperbarui di server web, kami memanggil fungsi berikut yang membuat permintaan baru di /status URL setiap detik.

```

setInterval(function () {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            var inputChecked;
            var outputStateM;
            if( this.responseText == 1){
                inputChecked = true;
                outputStateM = "On";
            }
            else {
                inputChecked = false;
                outputStateM = "Off";
            }
            document.getElementById("output").checked =
inputChecked;
            document.getElementById("outputState").innerHTML =
outputStateM;
        }
    };
    xhttp.open("GET", "/state", true);
    xhttp.send();
}, 1000 ) ;

```

g. Handle Request

Kemudian, kita perlu menangani apa yang terjadi ketika ESP32 atau ESP8266 menerima permintaan pada URL tersebut. Ketika permintaan

diterima di root / URL, terjadi proses mengirim halaman HTML serta prosesor.

```
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){  
    request->send_P(200, "text/html", index_html, processor);  
});
```

Baris berikut memeriksa apakah telah menerima permintaan pada /update?state=1 atau /update?state=0 URL dan mengubah ledState yang sesuai.

```
server.on("/update", HTTP_GET, [] (AsyncWebServerRequest  
*request) {  
    String inputMessage;  
    String inputParam;  
    // GET input1 value on  
<ESP_IP>/update?state=<inputMessage>  
    if (request->hasParam(PARAM_INPUT_1)) {  
        inputMessage =  
request->getParam(PARAM_INPUT_1)->value();  
        inputParam = PARAM_INPUT_1;  
        digitalWrite(output, inputMessage.toInt());  
        ledState = !ledState;  
    }  
    else {  
        inputMessage = "No message sent";  
        inputParam = "none";  
    }  
    Serial.println(inputMessage);  
    request->send(200, "text/plain", "OK");  
});
```

Saat permintaan diterima di /state URL, kami mengirim status keluaran saat ini:

```
server.on("/state", HTTP_GET, []  
(AsyncWebServerRequest *request) {  
    request->send(200, "text/plain",  
String(digitalRead(output)).c_str());  
});
```

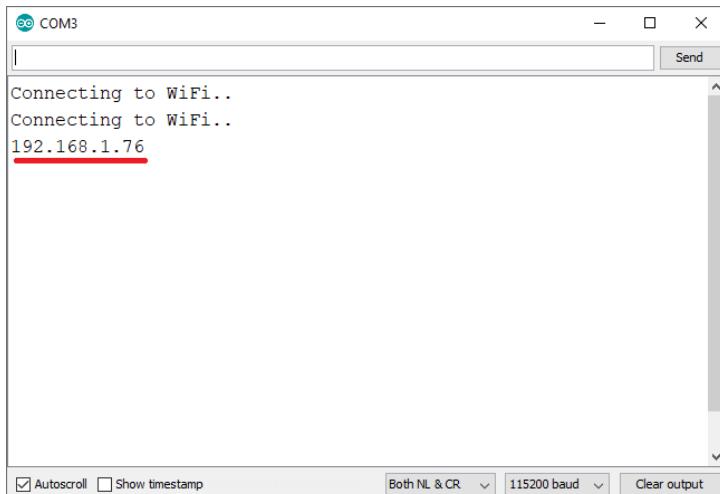
h. Loop()

Dalam loop(), kami memberikan fungsi bounce pada tombol tekan dan menyalakan atau mematikan LED tergantung pada nilai variabel ledState.

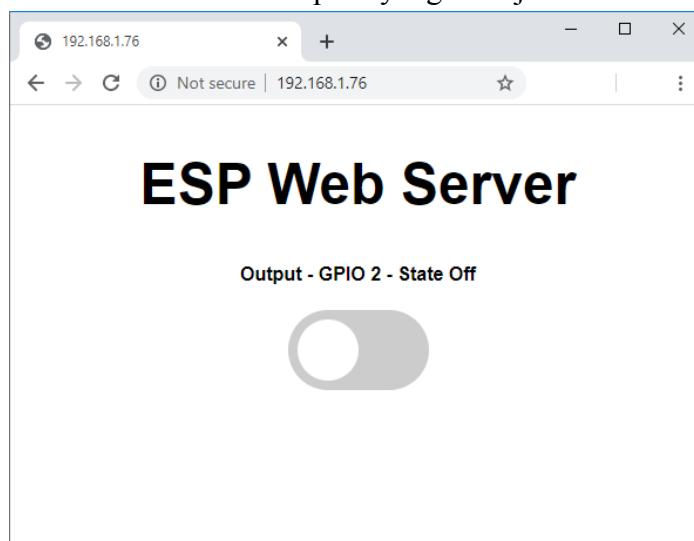
```
digitalWrite(output, ledState);
```

12.5 Demonstration

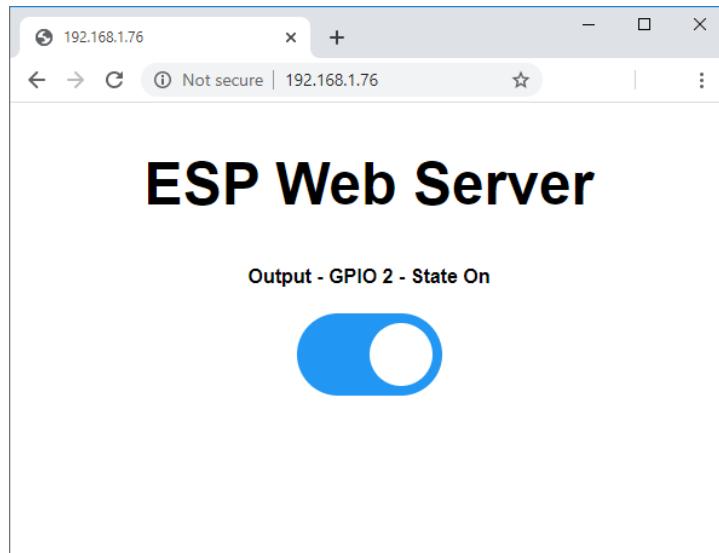
Menungguh kode ke papan ESP32 atau ESP8266. Kemudian, buka Serial Monitor pada baud rate 115200. Tekan tombol EN/RST on-board untuk mendapatkan alamat IP.



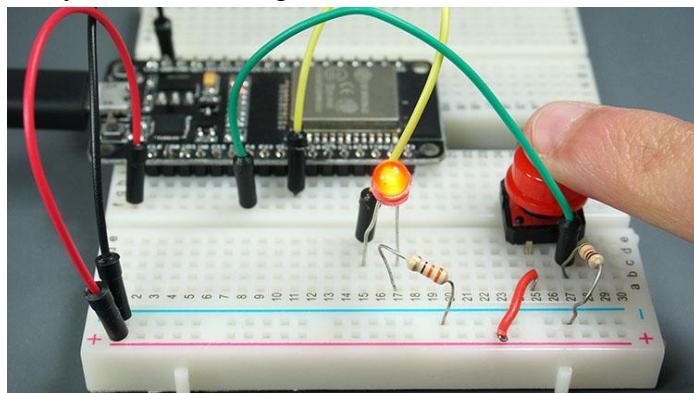
Buka browser di jaringan lokal, dan ketik alamat IP ESP. Kami harus memiliki akses ke server web seperti yang ditunjukkan di bawah ini.



Kami dapat mengaktifkan tombol di server web untuk menyalakan LED.



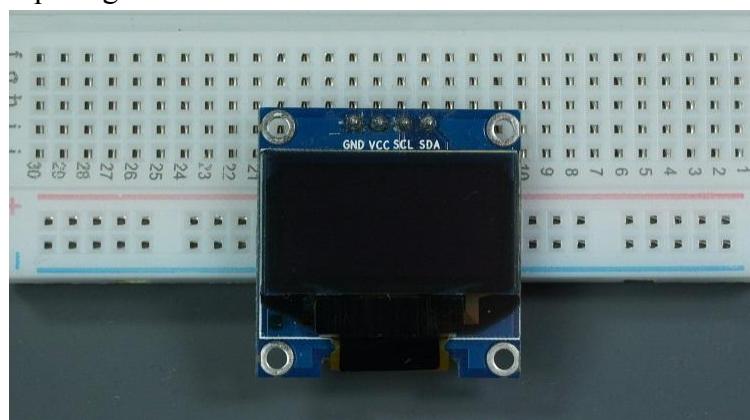
Kami juga dapat mengontrol LED yang sama dengan tombol tekan fisik. Statusnya akan selalu diperbarui secara otomatis di server web.



BAB 13. ESP32 Oled Display

13.1 Pengenalan 0.96 inch OLED Display

Layar OLED yang akan kita gunakan dalam tutorial ini adalah model SSD1306: layar 0,96 inci monocolor dengan 128x64 piksel seperti yang ditunjukkan pada gambar berikut.



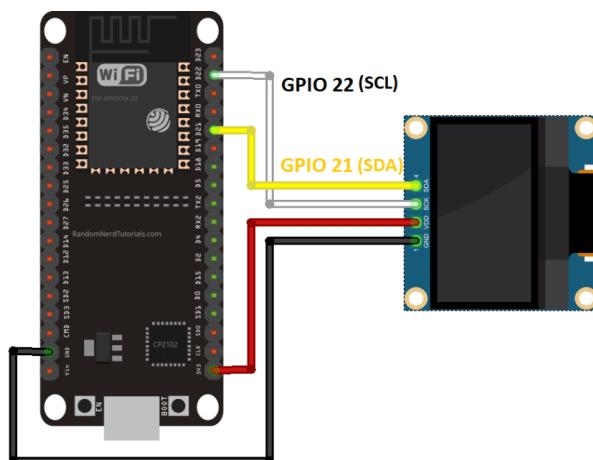
Layar OLED tidak memerlukan lampu latar, yang menghasilkan kontras yang sangat bagus di lingkungan yang gelap. Selain itu, pikselnya hanya

mengkonsumsi energi saat menyala, sehingga layar OLED mengonsumsi lebih sedikit daya jika dibandingkan dengan layar lainnya. Model yang kami gunakan memiliki empat pin dan berkomunikasi dengan mikrokontroler apapun menggunakan protokol komunikasi I2C. Ada model yang datang dengan pin RESET ekstra atau yang berkomunikasi menggunakan protokol komunikasi SPI.

13.2 OLED Display SSD1306 Pin Wiring

Pin	ESP32
Vin	3.3V
GND	GND
SCL	GPIO 22
SDA	GPIO 21

Atau ikuti diagram skema berikutnya untuk menyambungkan ESP32 ke layar OLED.

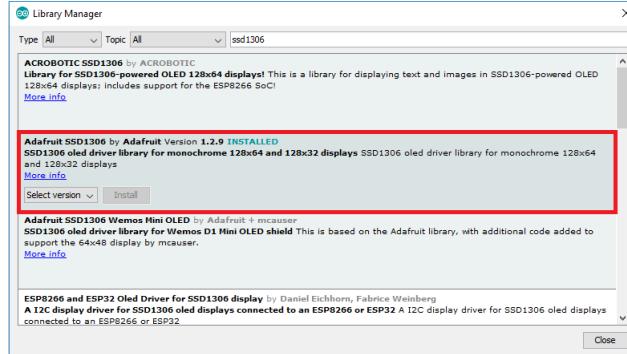


Dalam contoh ini, kami menggunakan protokol komunikasi I2C. Pin yang paling cocok untuk komunikasi I2C di ESP32 adalah GPIO 22 (SCL) dan GPIO 21 (SDA). Jika Kami menggunakan layar OLED dengan protokol komunikasi SPI, gunakan GPIO berikut.

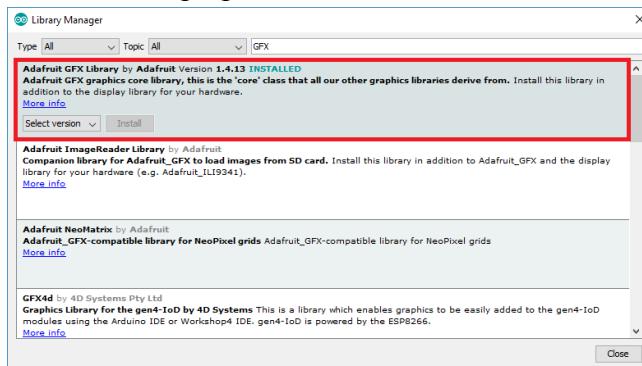
- 1) GPIO 18: CLK
- 2) GPIO 19: MISO
- 3) GPIO 23: MOSI
- 4) GPIO 5: CS

13.3 Installing SSD1306 OLED Library – ESP32

1. Tulis IDE Arduino Kami dan buka Sketch > Include Library > Manage Libraries. Manajer Perpustakaan harus terbuka.
2. Ketik "SSD1306" di kotak pencarian dan instal perpustakaan SSD1306 dari Adafruit.



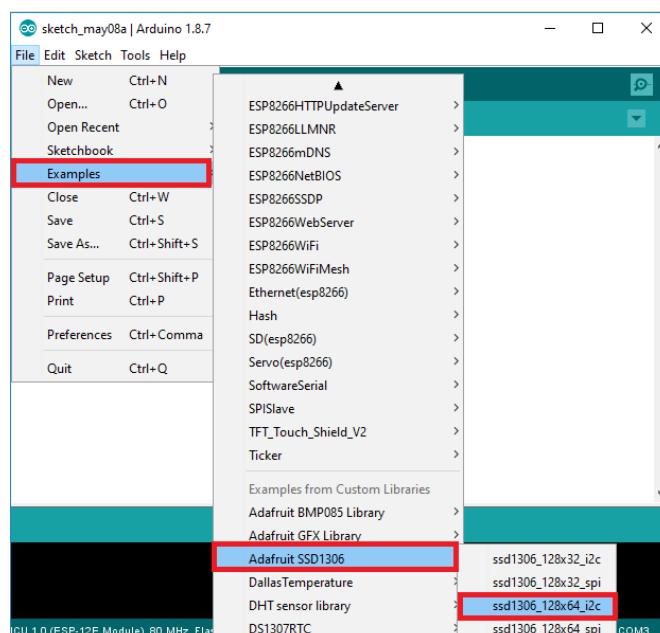
- Setelah menginstal perpustakaan SSD1306 dari Adafruit, ketik "GFX" di kotak pencarian dan instal perpustakaan.



- Setelah menginstal perpustakaan, restart Arduino IDE.

13.4 Testing OLED Display with ESP32

Setelah memasang kabel layar OLED ke ESP32 dan menginstal semua pustaka yang diperlukan, Kami dapat menggunakan satu contoh dari pustaka untuk melihat apakah semuanya berfungsi dengan baik. Di Arduino IDE Kami, buka File > Contoh > Adafruit SSD1306 dan pilih contoh untuk tampilan yang Kami gunakan.



Kode berikut harus dimuat:

```
*****
 Complete project details at
https://randomnerdtutorials.com

 This is an example for our Monochrome OLEDs based on
SSD1306 drivers. Pick one up today in the adafruit shop!
-----> http://www.adafruit.com/category/63_98
 This example is for a 128x32 pixel display using I2C
to communicate 3 pins are required to interface (two I2C
and one reset).
 Adafruit invests time and resources providing this
open source code, please support Adafruit and
open-source hardware by purchasing products from
Adafruit!
 Written by Limor Fried/Ladyada for Adafruit
Industries, with contributions from the open source
community. BSD license, check license.txt for more
information All text above, and the splash screen below
must be included in any redistribution.
*****/

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in
pixels
#define SCREEN_HEIGHT 64 // OLED display height, in
pixels

// Declaration for an SSD1306 display connected to I2C
(SDA, SCL pins)
#define OLED_RESET -1 // Reset pin # (or -1 if
sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, OLED_RESET);

#define NUMFLAKES 10 // Number of snowflakes in the
animation example

#define LOGO_HEIGHT 16
#define LOGO_WIDTH 16
static const unsigned char PROGMEM logo_bmp[] =
{ B00000000, B11000000,
B00000001, B11000000,
```

```
B00000001, B11000000,
B00000011, B11100000,
B11110011, B11100000,
B11111110, B11111000,
B01111110, B11111111,
B00110011, B10011111,
B00011111, B11111100,
B00001101, B01110000,
B00011011, B10100000,
B00111111, B11100000,
B00111111, B11110000,
B01111100, B11110000,
B01110000, B01110000,
B00000000, B00110000 };
```

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

  // SSD1306_SWITCHCAPVCC = generate display voltage
  // from 3.3V internally
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  // Show initial display buffer contents on the screen
  --
  // the library initializes this with an Adafruit
  // splash screen.
  display.display();
  delay(2000); // Pause for 2 seconds

  // Clear the buffer
  display.clearDisplay();

  // Draw a single pixel in white
  display.drawPixel(10, 10, WHITE);

  // Show the display buffer on the screen. You MUST
  // call display() after
  // drawing commands to make them visible on screen!
  display.display();
  delay(2000);
  // display.display() is NOT necessary after every
  // single drawing command,
  // unless that's what you want...rather, you can batch
  // up a bunch of
```

```
// drawing operations and then update the screen all
at once by calling
// display.display(). These examples demonstrate both
approaches...

testdrawline();           // Draw many lines

testdrawrect();           // Draw rectangles (outlines)

testfillrect();           // Draw rectangles (filled)

testdrawcircle();          // Draw circles (outlines)

testfillcircle();          // Draw circles (filled)

testdrawroundrect(); // Draw rounded rectangles
(outlines)

testfillroundrect(); // Draw rounded rectangles
(filled)

testdrawtriangle(); // Draw triangles (outlines)

testfilltriangle(); // Draw triangles (filled)

testdrawchar();           // Draw characters of the default
font

testdrawstyles();          // Draw 'stylized' characters

testscrolltext();          // Draw scrolling text

testdrawbitmap();          // Draw a small bitmap image

// Invert and restore display, pausing in-between
display.invertDisplay(true);
delay(1000);
display.invertDisplay(false);
delay(1000);

testanimate(logo_bmp, LOGO_WIDTH, LOGO_HEIGHT); // 
Animate bitmaps
}

void loop() {
}
```

```
void testdrawline() {
    int16_t i;

    display.clearDisplay(); // Clear display buffer

    for(i=0; i<display.width(); i+=4) {
        display.drawLine(0, 0, i, display.height()-1,
WHITE);
        display.display(); // Update screen with each
newly-drawn line
        delay(1);
    }
    for(i=0; i<display.height(); i+=4) {
        display.drawLine(0, 0, display.width()-1, i, WHITE);
        display.display();
        delay(1);
    }
    delay(250);

    display.clearDisplay();

    for(i=0; i<display.width(); i+=4) {
        display.drawLine(0, display.height()-1, i, 0,
WHITE);
        display.display();
        delay(1);
    }
    for(i=display.height()-1; i>=0; i-=4) {
        display.drawLine(0, display.height()-1,
display.width()-1, i, WHITE);
        display.display();
        delay(1);
    }
    delay(250);

    display.clearDisplay();

    for(i=display.width()-1; i>=0; i-=4) {
        display.drawLine(display.width()-1,
display.height()-1, i, 0, WHITE);
        display.display();
        delay(1);
    }
    for(i=display.height()-1; i>=0; i-=4) {
        display.drawLine(display.width()-1,
display.height()-1, 0, i, WHITE);
        display.display();
    }
}
```

```

        delay(1);
    }
delay(250);

display.clearDisplay();

for(i=0; i<display.height(); i+=4) {
    display.drawLine(display.width()-1, 0, 0, i, WHITE);
    display.display();
    delay(1);
}
for(i=0; i<display.width(); i+=4) {
    display.drawLine(display.width()-1, 0, i,
display.height()-1, WHITE);
    display.display();
    delay(1);
}

delay(2000); // Pause for 2 seconds
}

void testdrawrect(void) {
display.clearDisplay();

for(int16_t i=0; i<display.height()/2; i+=2) {
    display.drawRect(i, i, display.width()-2*i,
display.height()-2*i, WHITE);
    display.display(); // Update screen with each
newly-drawn rectangle
    delay(1);
}

delay(2000);
}

void testfillrect(void) {
display.clearDisplay();

for(int16_t i=0; i<display.height()/2; i+=3) {
    // The INVERSE color is used so rectangles alternate
white/black
    display.fillRect(i, i, display.width()-i*2,
display.height()-i*2, INVERSE);
    display.display(); // Update screen with each
newly-drawn rectangle
    delay(1);
}

```

```

        delay(2000);
    }

void testdrawcircle(void) {
    display.clearDisplay();

    for(int16_t i=0;
i<max(display.width(),display.height())/2; i+=2) {
        display.drawCircle(display.width()/2,
display.height()/2, i, WHITE);
        display.display();
        delay(1);
    }

    delay(2000);
}

void testfillcircle(void) {
    display.clearDisplay();

    for(int16_t i=max(display.width(),display.height())/2;
i>0; i-=3) {
        // The INVERSE color is used so circles alternate
white/black
        display.fillCircle(display.width() / 2,
display.height() / 2, i, INVERSE);
        display.display(); // Update screen with each
newly-drawn circle
        delay(1);
    }

    delay(2000);
}

void testdrawroundrect(void) {
    display.clearDisplay();

    for(int16_t i=0; i<display.height()/2-2; i+=2) {
        display.drawRoundRect(i, i, display.width()-2*i,
display.height()-2*i,
        display.height()/4, WHITE);
        display.display();
        delay(1);
    }

    delay(2000);
}

```

```

}

void testfillroundrect(void) {
    display.clearDisplay();

    for(int16_t i=0; i<display.height()/2-2; i+=2) {
        // The INVERSE color is used so round-rects
        alternate white/black
        display.fillRoundRect(i, i, display.width()-2*i,
        display.height()-2*i,
        display.height()/4, INVERSE);
        display.display();
        delay(1);
    }

    delay(2000);
}

void testdrawtriangle(void) {
    display.clearDisplay();

    for(int16_t i=0;
i<max(display.width(),display.height())/2; i+=5) {
        display.drawTriangle(
            display.width()/2 , display.height()/2-i,
            display.width()/2-i, display.height()/2+i,
            display.width()/2+i, display.height()/2+i, WHITE);
        display.display();
        delay(1);
    }

    delay(2000);
}

void testfilltriangle(void) {
    display.clearDisplay();

    for(int16_t i=max(display.width(),display.height())/2;
i>0; i-=5) {
        // The INVERSE color is used so triangles alternate
        white/black
        display.fillTriangle(
            display.width()/2 , display.height()/2-i,
            display.width()/2-i, display.height()/2+i,
            display.width()/2+i, display.height()/2+i,
            INVERSE);
        display.display();
    }
}

```

```
    delay(1);
}

delay(2000);
}

void testdrawchar(void) {
    display.clearDisplay();

    display.setTextSize(1);      // Normal 1:1 pixel scale
    display.setTextColor(WHITE); // Draw white text
    display.setCursor(0, 0);     // Start at top-left
corner
    display.cp437(true);        // Use full 256 char
'Code Page 437' font

    // Not all the characters will fit on the display.
This is normal.
    // Library will draw what it can and the rest will be
clipped.
    for(int16_t i=0; i<256; i++) {
        if(i == '\n') display.write(' ');
        else           display.write(i);
    }

    display.display();
    delay(2000);
}

void testdrawstyles(void) {
    display.clearDisplay();

    display.setTextSize(1);      // Normal 1:1
pixel scale
    display.setTextColor(WHITE); // Draw white text
    display.setCursor(0,0);     // Start at
top-left corner
    display.println(F("Hello, world!"));

    display.setTextColor(BLACK, WHITE); // Draw 'inverse'
text
    display.println(3.141592);

    display.setTextSize(2);      // Draw 2X-scale
text
    display.setTextColor(WHITE);
```

```
    display.print(F("0x")); display.println(0xDEADBEEF,
HEX);

    display.display();
    delay(2000);
}

void testscrolltext(void) {
    display.clearDisplay();

    display.setTextSize(2); // Draw 2X-scale text
    display.setTextColor(WHITE);
    display.setCursor(10, 0);
    display.println(F("scroll"));
    display.display();      // Show initial text
    delay(100);

    // Scroll in various directions, pausing in-between:
    display.startscrollright(0x00, 0x0F);
    delay(2000);
    display.stopscroll();
    delay(1000);
    display.startscrollleft(0x00, 0x0F);
    delay(2000);
    display.stopscroll();
    delay(1000);
    display.startscrolldiagright(0x00, 0x07);
    delay(2000);
    display.startscrolldiagleft(0x00, 0x07);
    delay(2000);
    display.stopscroll();
    delay(1000);
}

void testdrawbitmap(void) {
    display.clearDisplay();

    display.drawBitmap(
        (display.width() - LOGO_WIDTH) / 2,
        (display.height() - LOGO_HEIGHT) / 2,
        logo_bmp, LOGO_WIDTH, LOGO_HEIGHT, 1);
    display.display();
    delay(1000);
}

#define XPOS 0 // Indexes into the 'icons' array in
function below
```

```

#define YPOS 1
#define DELTAY 2

void testanimate(const uint8_t *bitmap, uint8_t w,
uint8_t h) {
    int8_t f, icons[NUMFLAKES][3];

    // Initialize 'snowflake' positions
    for(f=0; f< NUMFLAKES; f++) {
        icons[f][XPOS] = random(1 - LOGO_WIDTH,
display.width());
        icons[f][YPOS] = -LOGO_HEIGHT;
        icons[f][DELTAY] = random(1, 6);
        Serial.print(F("x: "));
        Serial.print(icons[f][XPOS], DEC);
        Serial.print(F(" y: "));
        Serial.print(icons[f][YPOS], DEC);
        Serial.print(F(" dy: "));
        Serial.println(icons[f][DELTAY], DEC);
    }

    for(;;) { // Loop forever...
        display.clearDisplay(); // Clear the display buffer

        // Draw each snowflake:
        for(f=0; f< NUMFLAKES; f++) {
            display.drawBitmap(icons[f][XPOS], icons[f][YPOS],
bitmap, w, h, WHITE);
        }

        display.display(); // Show the display buffer on the
screen
        delay(200); // Pause for 1/10 second

        // Then update coordinates of each flake...
        for(f=0; f< NUMFLAKES; f++) {
            icons[f][YPOS] += icons[f][DELTAY];
            // If snowflake is off the bottom of the screen...
            if (icons[f][YPOS] >= display.height()) {
                // Reinitialize to a random position, just off
the top
                icons[f][XPOS] = random(1 - LOGO_WIDTH,
display.width());
                icons[f][YPOS] = -LOGO_HEIGHT;
                icons[f][DELTAY] = random(1, 6);
            }
        }
    }
}

```

```
    }  
}
```

Jika OLED tidak memiliki pin RESET, maka harus mengatur variabel OLED_RESET ke -1 seperti yang ditunjukkan di bawah ini:

```
#define OLED_RESET -1 // Reset pin # (or -1 if sharing  
Arduino reset pin)  
  
#define SCREEN_WIDTH 128 // OLED display width, in pixels  
#define SCREEN_HEIGHT 64 // OLED display height, in pixels  
  
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)  
#define OLED_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)  
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

13.5 Write Text – OLED Display

a. “Hello, world!” OLED Display

Sketsa berikut menampilkan Hello, world! pesan di layar OLED.

```
*****  
Rui Santos  
Complete project details at https://randomnerdtutorials.com  
*****  
  
#include <Wire.h>  
#include <Adafruit_GFX.h>  
#include <Adafruit_SSD1306.h>  
  
#define SCREEN_WIDTH 128 // OLED display width, in pixels  
#define SCREEN_HEIGHT 64 // OLED display height, in pixels  
  
// Declaration for an SSD1306 display connected to I2C  
// pins)  
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &  
  
void setup() {  
  Serial.begin(115200);  
  
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Attempt to  
for 128x64  
    Serial.println(F("SSD1306 allocation failed"));  
    for(;;);  
  }  
  delay(2000);  
  display.clearDisplay();  
  
  display.setTextSize(1);
```

```
    display.setTextColor(WHITE);
    display.setCursor(0, 10);
    // Display static text
    display.println("Hello, world!");
    display.display();
}

void loop() {
}
```

Setelah mengunggah kode, inilah yang akan didapatkan di OLED:



b. Importing libraries

Pertama, impor perpustakaan yang diperlukan. Pustaka Wire untuk menggunakan I2C dan pustaka Adafruit untuk menulis ke layar: Adafruit_GFX dan Adafruit_SSD1306.

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

c. Initialize the OLED display

Kemudian, tentukan lebar dan tinggi OLED. Dalam contoh ini, kami menggunakan layar OLED 128x64. Jika ingin menggunakan ukuran lain, dapat mengubahnya di variabel SCREEN_WIDTH, dan SCREEN_HEIGHT.

```
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
```

Kemudian, inisialisasi objek tampilan dengan lebar dan tinggi yang ditentukan sebelumnya dengan protokol komunikasi I2C (&Wire).

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

Parameter (-1) berarti layar OLED tidak memiliki pin RESET. Jika layar OLED memang memiliki pin RESET, itu harus terhubung ke GPIO. Dalam hal ini, maka harus memberikan nomor GPIO sebagai parameter. Dalam setup(), inisialisasi Serial Monitor dengan baud rate 115200 untuk keperluan debugging.

```
Serial.begin(115200);
```

Inisialisasi tampilan OLED dengan metode begin() sebagai berikut:

```
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {  
    Serial.println("SSD1306 allocation failed");  
    for(;;) // Don't proceed, loop forever  
}
```

Cuplikan ini juga mencetak pesan di Serial Monitor, jika tidak dapat terhubung ke layar.

```
Serial.println("SSD1306 allocation failed");
```

Jika menggunakan layar OLED yang berbeda, maka perlu mengubah alamat OLED. Dalam kasus ini, alamatnya adalah 0x3C.

```
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
```

Setelah menginisialisasi tampilan, tambahkan penundaan dua detik, sehingga OLED memiliki cukup waktu untuk menginisialisasi sebelum menulis teks:

```
delay(2000);
```

d. Clear display, set font size, color and write text

Setelah menginisialisasi tampilan, bersihkan buffer tampilan dengan metode clearDisplay():

```
display.clearDisplay();
```

Sebelum menulis teks, atur ukuran teks, warna dan dimana teks akan ditampilkan di OLED. Atur ukuran font menggunakan metode setTextSize():

```
display.setTextSize(1);
```

Atur warna font dengan metode setTextColor():

```
display.setTextColor(WHITE);
```

WHITE mengatur font putih dan latar belakang hitam. Tentukan posisi awal teks menggunakan metode setCursor(x,y). Dalam hal ini, kami mengatur teks untuk memulai pada koordinat (0,0) – di sudut kiri atas.

```
display.setCursor(0,0);
```

Terakhir, Kami dapat mengirim teks ke tampilan menggunakan metode println(), sebagai berikut:

```
display.println("Hello, world!");
```

Kemudian, panggil metode display() untuk benar-benar menampilkan teks di layar.

```
display.display();
```

e. Scrolling Text

Pustaka OLED Adafruit menyediakan metode yang berguna untuk menggulir teks dengan mudah.

- 1) startscrollright (0x00, 0x0F): gulir teks dari kiri ke kanan
- 2) startscrollleft(0x00, 0x0F): gulir teks dari kanan ke kiri
- 3) startcerolldiagright (0x00, 0x07): gulir teks dari sudut kiri bawah ke sudut kanan atas
- 4) startcerolldiagleft (0x00, 0x07): gulir teks dari sudut kanan bawah ke sudut kiri atas

Sketsa berikut mengimplementasikan metode tersebut.

```
*****
Rui Santos
Complete project details at
https://randomnerdtutorials.com
*****
```

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C
// (SDA, SCL pins)
```

```

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, -1);

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

    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // 
Address 0x3D for 128x64
        Serial.println(F("SSD1306 allocation failed"));
        for(;;);
    }
    delay(2000);
    display.clearDisplay();

    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0, 0);
    // Display static text
    display.println("Scrolling Hello");
    display.display();
    delay(100);

}

void loop() {
    // Scroll in various directions, pausing in-between:
    display.startscrollright(0x00, 0x0F);
    delay(2000);
    display.stopscroll();
    delay(1000);
    display.startscrollleft(0x00, 0x0F);
    delay(2000);
    display.stopscroll();
    delay(1000);
    display.startscrolldiagright(0x00, 0x07);
    delay(2000);
    display.startscrolldiagleft(0x00, 0x07);
    delay(2000);
    display.stopscroll();
    delay(1000);
}

```

13.6 Draw Shapes in the OLED Display

Pustaka OLED Adafruit menyediakan metode yang berguna untuk menggambar piksel, garis, dan bentuk. Mari kita lihat sekilas metode-metode itu.

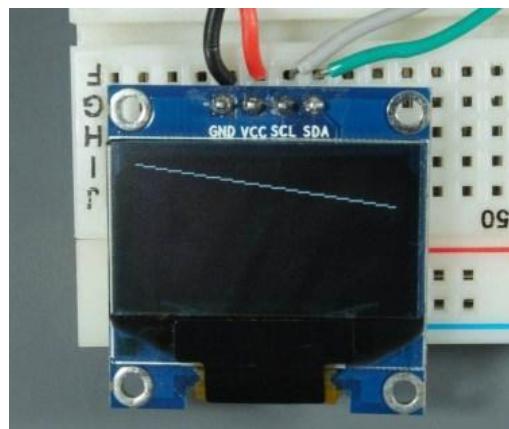
a. Draw a pixel



Untuk menggambar piksel dalam tampilan OLED, gunakan metode `drawPixel(x, y, color)` yang menerima sebagai argumen koordinat x dan y tempat piksel muncul, dan warna. Sebagai contoh:

```
display.drawPixel(64, 32, WHITE);
```

b. Draw a line



Gunakan metode `drawLine(x1, y1, x2, y2, color)` untuk membuat garis. Koordinat (x1, y1) menunjukkan awal garis, dan koordinat (x2, y2) menunjukkan di mana garis berakhir. Sebagai contoh:

```
display.drawLine(0, 0, 127, 20, WHITE);
```

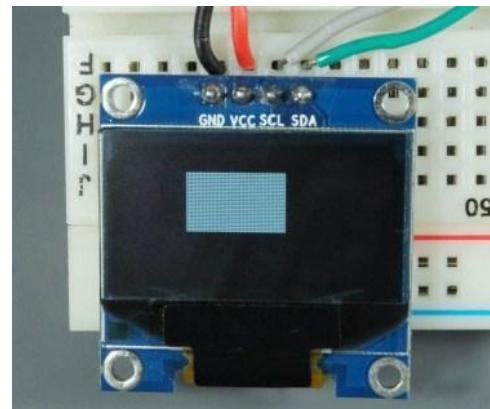
c. Draw a rectangle



`drawRect(x, y, width, height, color)` menyediakan cara mudah untuk menggambar persegi panjang. Koordinat (x,y) menunjukkan sudut kiri atas persegi panjang. Kemudian, tentukan lebar, tinggi dan warna:

```
display.drawRect(10, 10, 50, 30, WHITE);
```

Gunakan `fillRect(x, y, width, height, color)` untuk menggambar persegi panjang yang terisi. Metode ini menerima argumen yang sama dengan `drawRect()`.



d. Draw a circle

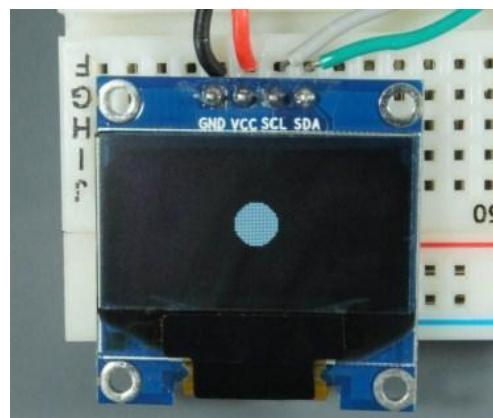


Untuk menggambar lingkaran gunakan metode `drawCircle(x, y, radius, color)`. Koordinat (x,y) menunjukkan pusat lingkaran, juga harus melewatkkan radius sebagai argumen. Sebagai contoh:

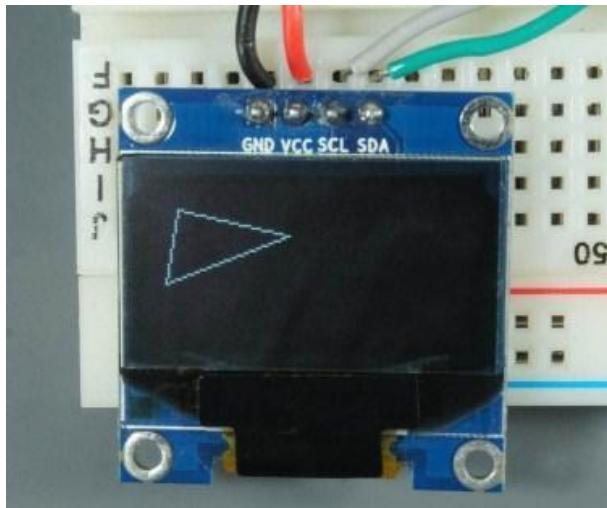
```
display.drawCircle(64, 32, 10, WHITE);
```

Dengan cara yang sama, untuk membuat lingkaran penuh, gunakan metode `fillCircle()` dengan argumen yang sama:

```
display.fillCircle(64, 32, 10, WHITE);
```



e. **Draw a triangle**



Gunakan metode `drawTriangle(x1, y1, x2, y2, x3, y3, warna)` untuk membangun sebuah segitiga. Metode ini menerima sebagai argumen koordinat setiap sudut dan warna.

```
display.drawTriangle(10, 10, 55, 20, 5, 40, WHITE);
```

Gunakan metode `fillTriangle()` untuk menggambar segitiga yang terisi.

```
display.fillTriangle(10, 10, 55, 20, 5, 40, WHITE);
```

f. **Invert**

Pustaka menyediakan metode tambahan yang bisa digunakan dengan bentuk atau teks: metode `invertDisplay()`. Berikan `true` sebagai argumen untuk membalikkan warna layar atau `false` untuk kembali ke warna aslinya. Jika memanggil perintah berikut setelah mendefinisikan segitiga:

```
display.invertDisplay(true);
```

g. **Code – Draw Shapes**

Unggah sketsa berikut yang mengimplementasikan setiap potongan kode yang telah kita bahas sebelumnya dan melewati semua bentuk.

```
*****
Rui Santos
Complete project details at
https://randomnerdtutorials.com
*****/

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

```
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// Declaration for an SSD1306 display connected to I2C
// (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, -1);

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

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000); // Pause for 2 seconds

  // Clear the buffer
  display.clearDisplay();

  // Draw a single pixel in white
  display.drawPixel(64, 32, WHITE);
  display.display();
  delay(3000);

  // Draw line
  display.clearDisplay();
  display.drawLine(0, 0, 127, 20, WHITE);
  display.display();
  delay(3000);

  // Draw rectangle
  display.clearDisplay();
  display.drawRect(30, 10, 50, 30, WHITE);
  display.display();
  delay(3000);
  // Fill rectangle
  display.fillRect(30, 10, 50, 30, WHITE);
  display.display();
  delay(3000);

  // Draw round rectangle
  display.clearDisplay();
  display.drawRoundRect(10, 10, 30, 50, 2, WHITE);
  display.display();
  delay(3000);
```

```

// Fill round rectangle
display.clearDisplay();
display.fillRoundRect(10, 10, 30, 50, 2, WHITE);
display.display();
delay(3000);

// Draw circle
display.clearDisplay();
display.drawCircle(64, 32, 10, WHITE);
display.display();
delay(3000);
// Fill circle
display.fillCircle(64, 32, 10, WHITE);
display.display();
delay(3000);

// Draw triangle
display.clearDisplay();
display.drawTriangle(10, 10, 55, 20, 5, 40, WHITE);
display.display();
delay(3000);
// Fill triangle
display.fillTriangle(10, 10, 55, 20, 5, 40, WHITE);
display.display();
delay(3000);

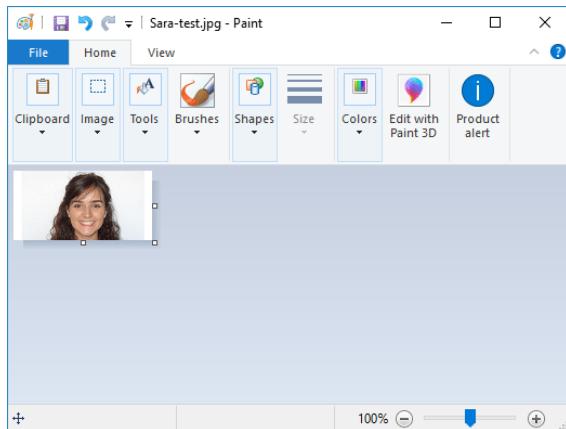
// Invert and restore display, pausing in-between
display.invertDisplay(true);
delay(3000);
display.invertDisplay(false);
delay(3000);
}

void loop() {
}

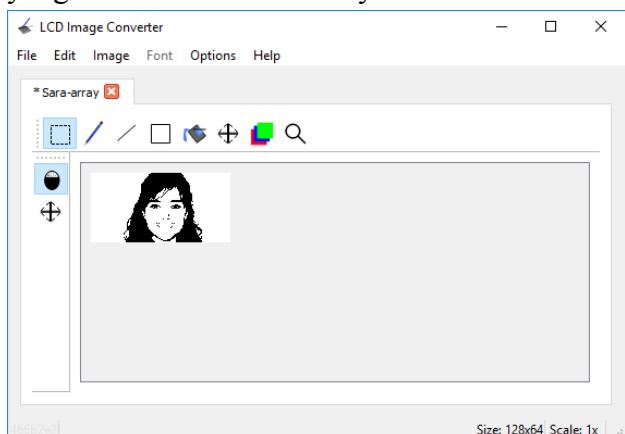
```

13.7 Display Bitmap Images in the OLED

Pertama, gunakan program pencitraan untuk mengubah ukuran foto atau gambar dan menyimpannya sebagai bitmap monokrom. Jika yang digunakan PC Windows, maka dapat menggunakan Paint.

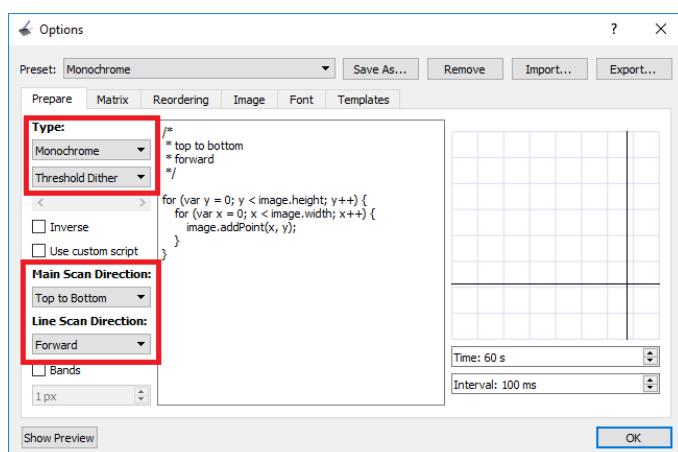


Kemudian, gunakan konverter Array Gambar ke C untuk mengubah gambar menjadi array. Saya telah menggunakan Pengonversi Gambar LCD. Jalankan program dan mulai dengan gambar baru. Buka Gambar > Impor dan pilih gambar bitmap yang telah dibuat sebelumnya.



Buka Opsi > Konversi dan di tab Siapkan, pilih opsi berikut:

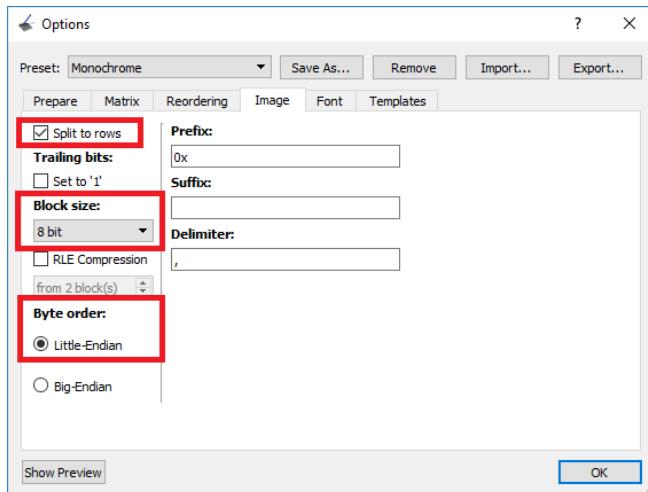
- 1) **Jenis:** Monokrom, Threshold Dither
- 2) **Arah Pemindaian Utama:** Atas ke Bawah
- 3) **Arah Pemindaian Garis:** Maju



Buka tab Gambar dan pilih opsi berikut:

- 1) Pisahkan menjadi baris

- 2) **Ukuran blok:** 8 bit
- 3) **Urutan byte:** Little-Endian



Salin array ke sketsa. Kemudian, untuk menampilkan larik, gunakan metode drawBitmap() yang menerima argumen berikut (x, y, larik gambar, lebar gambar, tinggi gambar, rotasi). Koordinat (x, y) menentukan di mana gambar mulai ditampilkan. Salin kode di bawah ini untuk menampilkan gambar bitmap di OLED.

```
*****
  Rui Santos
  Complete project details at
https://randomnerdtutorials.com
*****


#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, -1);

static const uint8_t image_data_Saraarray[1024] = {
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00,
    0x00, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00,
    0x00, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00,
    0x00, 0x00, 0x07, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00,
    0x00, 0x00, 0x07, 0xff, 0xff, 0xff, 0xff, 0xff,
```

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00,
0x00, 0x00, 0x03, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00,
0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00,
0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00,
0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00,
0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00,
0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x00, 0xa0, 0x00, 0x1f, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x14, 0x9e, 0x00, 0x1f, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x36, 0x3f, 0x00, 0x1f, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x6d, 0xff, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0xfb, 0xff, 0x80, 0x1f, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x03,
0xd7, 0xff, 0x80, 0x0f, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x07,
0xef, 0xff, 0x80, 0x07, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x0f,
0xdf, 0xff, 0x90, 0x07, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x0f,
0xbf, 0xff, 0xd0, 0x07, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x1d,
0x7f, 0xff, 0xd0, 0x07, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xf0, 0x01, 0x1b,
0xff, 0xff, 0xc0, 0x07, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xf0, 0x02, 0xa7,
0xff, 0xff, 0xc0, 0x07, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x03,
0xff, 0xc0, 0x00, 0x07, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00,
0xff, 0x80, 0x00, 0xb, 0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x03, 0xff,
0xff, 0xff, 0xf0, 0x0f, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x07, 0xff,
0xff, 0xff, 0xf0, 0x0f, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x0f, 0x07,
0xff, 0xf8, 0xf8, 0x03, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x0e, 0x01,
0xff, 0xc0, 0x38, 0x07, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x1c, 0x46,
0xff, 0xb1, 0x18, 0x07, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xfe, 0x00, 0x3f, 0x97,
0xff, 0xc0, 0x7a, 0x07, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xfe, 0x00, 0x3f, 0xff,
0xff, 0xff, 0xfe, 0x03, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xfe, 0x01, 0x3f, 0xff,
0xff, 0xff, 0xfe, 0x01, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xfe, 0x01, 0xbff, 0xff,
0xff, 0xff, 0xfe, 0x81, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xfc, 0x00, 0xbff, 0xff,
0xff, 0xff, 0xfc, 0x81, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xfc, 0x00, 0xff, 0xff,
0xfe, 0xff, 0xfd, 0x83, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0xbff, 0xff,
0xfe, 0xff, 0xfd, 0x01, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x7f, 0xff,
0xff, 0xff, 0xff, 0x01, 0xff, 0xff, 0x0ff, 0xff,
 0xff, 0xff, 0xff, 0xf0, 0x00, 0x7f, 0xff,
0xff, 0xff, 0xfb, 0x03, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xe0, 0x00, 0x3f, 0xff,
0xdc, 0xff, 0xfa, 0x03, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xd8, 0x00, 0x1f, 0xff,
0xff, 0xff, 0xf8, 0x03, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xd0, 0x00, 0x1f, 0xff,
0xff, 0xff, 0xf8, 0x01, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0x90, 0x00, 0x1f, 0xff,
0xff, 0xff, 0xf8, 0x02, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xb0, 0x00, 0x0f, 0xf5,
0xff, 0xd7, 0xf8, 0x01, 0xff, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xb0, 0x00, 0x0f, 0xff,
0xff, 0xff, 0xf8, 0x00, 0x5f, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0xa0, 0x00, 0x0f, 0xfb,
0xff, 0xff, 0xf0, 0x00, 0x3f, 0xff, 0xff, 0xff,
 0xff, 0xff, 0xff, 0x80, 0x00, 0x0f, 0xfd,
0xff, 0xdf, 0xf0, 0x00, 0x3f, 0xff, 0xff, 0xff,

```

        0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x07, 0xff,
0xff, 0xbff, 0xf0, 0x00, 0x0f, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x07, 0xff,
0xff, 0xff, 0xe0, 0x00, 0x87, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x03, 0xff,
0xff, 0xff, 0xc0, 0x00, 0x43, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0x60, 0x00, 0x01, 0xff,
0xff, 0xff, 0xc0, 0x00, 0x73, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xfe, 0xe0, 0x00, 0x00, 0xff,
0xff, 0xff, 0x80, 0x00, 0x7b, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xfd, 0xe0, 0x00, 0x00, 0x7f,
0xff, 0xfe, 0x00, 0x00, 0x33, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xfd, 0xe0, 0x00, 0x00, 0x3f,
0xff, 0xf8, 0x00, 0x00, 0x27, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x0f,
0xff, 0xf0, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0x60, 0x00, 0x00, 0x67,
0xff, 0xe0, 0x00, 0x00, 0x1b, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xfd, 0x40, 0x00, 0x00, 0xf3,
0xff, 0xc4, 0x00, 0x00, 0xb, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xfe, 0x80, 0x00, 0x00, 0xfc,
0xff, 0x8c, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x7f,
0x3c, 0x3c, 0x00, 0x00, 0x07, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xfd, 0x00, 0x00, 0x00, 0x3f,
0xc0, 0x7c, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xfd, 0x00, 0x00, 0x00, 0x1f,
0xff, 0xfc, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff
};

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

    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        Serial.println(F("SSD1306 allocation failed"));
        for(;;);
    }
    delay(2000); // Pause for 2 seconds

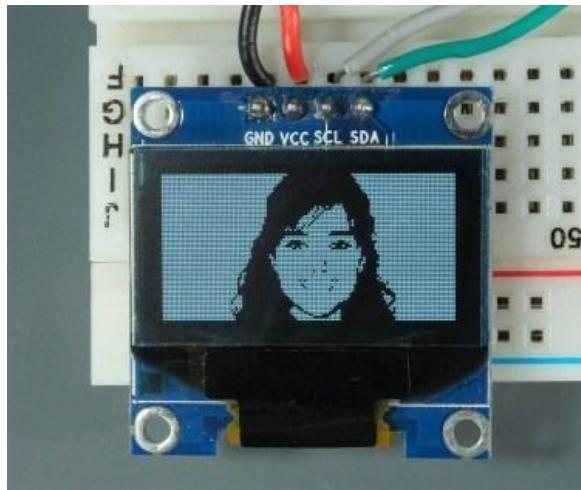
    // Clear the buffer.
    display.clearDisplay();

    // Draw bitmap on the screen
    display.drawBitmap(0, 0, image_data_Saraarray, 128,
64, 1);
    display.display();
}

```

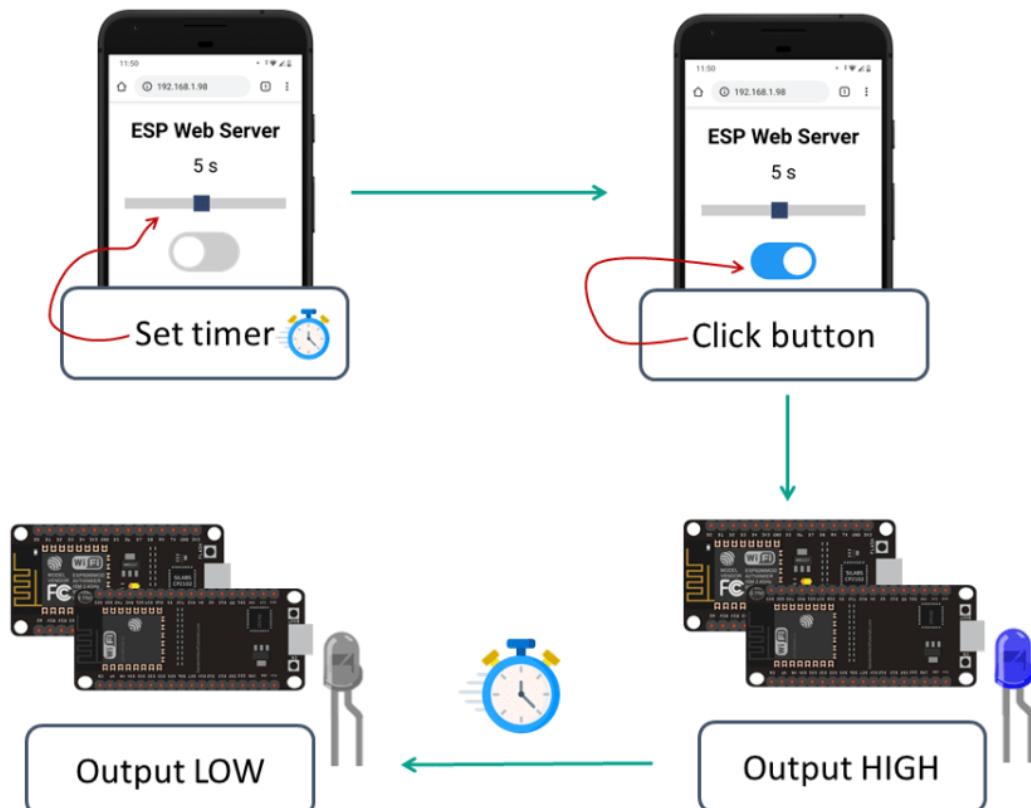
```
void loop() {  
}
```

Setelah mengupload kode, inilah yang kita dapatkan di tampilan.



Bab 14. ESP32 Timer Or Pulse Webserver

14.1 Project Overview



1. ESP32/ESP8266 merupakan hosting server web yang memungkinkan untuk mengontrol output dengan pulsa;

2. Server web berisi slider yang memungkinkan untuk menentukan lebar pulsa (berapa detik output harus HIGH);
3. Ada tombol ON/OFF. Atur ke ON untuk mengirim pulsa. Setelah itu, akan terlihat timer berkurang selama durasi lebar pulsa;
4. Ketika pengatur waktu berakhir, output diatur ke LOW, dan tombol server web kembali ke status OFF;
5. Server web ini dapat berguna untuk mengontrol perangkat yang membutuhkan pulsa untuk diaktifkan seperti pembuka pintu garasi, misalnya.

14.2 Installing Libraries – Async Web Server

Untuk membangun server web, kami perlu menginstal pustaka berikut:

1) ESP32: instal pustaka ESPAsyncWebServer dan AsyncTCP.

2) ESP8266: instal pustaka ESPAsyncWebServer dan ESPAsyncTCP.

Pustaka ini tidak tersedia untuk diinstall melalui Manajer Perpustakaan Arduino, jadi salin file perpustakaan ke folder Perpustakaan Instalasi Arduino. Atau, di Arduino ID. Pergi ke Sketch > Include Library > Add .zip Library dan pilih library yang baru saja diunduh.

14.3 Code

```
*****
Rui Santos
Complete project details at
https://RandomNerdTutorials.com/esp32-esp8266-web-server-timer-pulse/

The above copyright notice and this permission notice
shall be included in all
copies or substantial portions of the Software.
*****


// Import required libraries
#ifndef ESP32
  #include <WiFi.h>
  #include <AsyncTCP.h>
#else
  #include <ESP8266WiFi.h>
  #include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>

// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

const char* PARAM_INPUT_1 = "state";
const char* PARAM_INPUT_2 = "value";
```

```

const int output = 2;

String timerSliderValue = "10";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
    <meta name="viewport" content="width=device-width,
initial-scale=1">
        <title>ESP Web Server</title>
        <style>
            html {font-family: Arial; display: inline-block;
text-align: center;}
            h2 {font-size: 2.4rem;}
            p {font-size: 2.2rem;}
            body {max-width: 600px; margin:0px auto;
padding-bottom: 25px;}
            .switch {position: relative; display: inline-block;
width: 120px; height: 68px}
            .switch input {display: none}
            .slider {position: absolute; top: 0; left: 0; right: 0;
bottom: 0; background-color: #ccc; border-radius: 34px}
            .slider:before {position: absolute; content: "";
height: 52px; width: 52px; left: 8px; bottom: 8px;
background-color: #fff; -webkit-transition: .4s;
transition: .4s; border-radius: 68px}
            input:checked+.slider {background-color: #2196F3}
            input:checked+.slider:before {-webkit-transform:
translateX(52px); -ms-transform: translateX(52px);
transform: translateX(52px)}
            .slider2 { -webkit-appearance: none; margin: 14px;
width: 300px; height: 20px; background: #ccc;
outline: none; -webkit-transition: .2s; transition:
opacity .2s;}
            .slider2::-webkit-slider-thumb {-webkit-appearance:
none; appearance: none; width: 30px; height: 30px;
background: #2f4468; cursor: pointer;}
            .slider2::-moz-range-thumb { width: 30px; height: 30px;
background: #2f4468; cursor: pointer; }
        </style>
</head>
<body>
    <h2>ESP Web Server</h2>

```

```

<p><span id="timerValue">%TIMERVALUE%</span> s</p>
<p><input type="range" onchange="updateSliderTimer(this)" id="timerSlider" min="1" max="20" value="%TIMERVALUE%" step="1" class="slider2"></p>
    %BUTTONPLACEHOLDER%
<script>
function toggleCheckbox(element) {
    var sliderValue =
document.getElementById("timerSlider").value;
    var xhr = new XMLHttpRequest();
    if(element.checked){ xhr.open("GET", "/update?state=1",
true); xhr.send();
        var count = sliderValue, timer = setInterval(function()
{
            count--;
        document.getElementById("timerValue").innerHTML = count;
            if(count == 0){ clearInterval(timer);
        document.getElementById("timerValue").innerHTML =
document.getElementById("timerSlider").value; }
            }, 1000);
        sliderValue = sliderValue*1000;
        setTimeout(function(){ xhr.open("GET",
"/update?state=0", true);
            document.getElementById(element.id).checked = false;
xhr.send(); }, sliderValue);
    }
}
function updateSliderTimer(element) {
    var sliderValue =
document.getElementById("timerSlider").value;
    document.getElementById("timerValue").innerHTML =
sliderValue;
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "/slider?value="+sliderValue, true);
    xhr.send();
}
</script>
</body>
</html>
)rawliteral";

// Replaces placeholder with button section in your web
page
String processor(const String& var){
    //Serial.println(var);
    if(var == "BUTTONPLACEHOLDER"){
        String buttons = "";

```

```

        String outputStateValue = outputState();
        buttons+= "<p><label class=\"switch\"><input
type=\"checkbox\" onchange=\"toggleCheckbox(this)\""
id=\"output\" " + outputStateValue + "><span
class=\"slider\"></span></label></p>";
        return buttons;
    }
    else if(var == "TIMERVALUE"){
        return timerSliderValue;
    }
    return String();
}

String outputState(){
    if(digitalRead(output)){
        return "checked";
    }
    else {
        return "";
    }
    return "";
}

void setup(){
    // Serial port for debugging purposes
    Serial.begin(115200);

    pinMode(output, OUTPUT);
    digitalWrite(output, LOW);

    // Connect to Wi-Fi
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi..");
    }

    // Print ESP Local IP Address
    Serial.println(WiFi.localIP());

    // Route for root / web page
    server.on("/", HTTP_GET, [] (AsyncWebRequest
*request){
        request->send_P(200, "text/html", index_html,
processor);
    });
}

```

```

    // Send a GET request to
<ESP_IP>/update?state=<inputMessage>
    server.on("/update", HTTP_GET, [] (AsyncWebServerRequest
*request) {
        String inputMessage;
        // GET input1 value on
<ESP_IP>/update?state=<inputMessage>
        if (request->hasParam(PARAM_INPUT_1)) {
            inputMessage =
request->getParam(PARAM_INPUT_1)->value();
            digitalWrite(output, inputMessage.toInt());
        }
        else {
            inputMessage = "No message sent";
        }
        Serial.println(inputMessage);
        request->send(200, "text/plain", "OK");
    });

    // Send a GET request to
<ESP_IP>/slider?value=<inputMessage>
    server.on("/slider", HTTP_GET, [] (AsyncWebServerRequest
*request) {
        String inputMessage;
        // GET input1 value on
<ESP_IP>/slider?value=<inputMessage>
        if (request->hasParam(PARAM_INPUT_2)) {
            inputMessage =
request->getParam(PARAM_INPUT_2)->value();
            timerSliderValue = inputMessage;
        }
        else {
            inputMessage = "No message sent";
        }
        Serial.println(inputMessage);
        request->send(200, "text/plain", "OK");
    });

    // Start server
    server.begin();
}

void loop() {
}

```

Kami hanya perlu memasukkan kredensial jaringan (SSID dan kata sandi) dan server web akan langsung bekerja. Kode ini kompatibel dengan papan

ESP32 dan ESP8266 dan mengontrol LED GPIO 2 on-board – kami dapat mengubah kode untuk mengontrol GPIO lainnya.

14.4 How the Code Works

Kami telah menjelaskan dengan sangat rinci bagaimana server web seperti ini bekerja di tutorial sebelumnya (Server Web Suhu DHT atau Server Web Relay), jadi kami hanya akan melihat bagian yang relevan untuk proyek ini.

a. Network Credentials

```
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

b. Slider Label

```
<p><span id="timerValue">%TIMERVALUE%</span> s</p>
```

Secara default, nilai slider diatur ke %TIMERVALUE% placeholder. %TIMERVALUE% adalah placeholder yang akan diganti dengan nilai yang disimpan dalam variabel timerSliderValue yang diatur ke 10 secara default. Tetapi Kami dapat mengubahnya di baris berikut:

```
String timerSliderValue = "10";
```

c. Slider

```
<input type="range" onchange="updateSliderTimer(this)"
id="timerSlider" min="1" max="20" value="%TIMERVALUE%"
step="1" class="slider2">
```

Dalam HTML, slider adalah tipe input. Tag <input> menentukan bidang input tempat pengguna dapat memasukkan data. Slider adalah bidang input dari rentang tipe. Ada banyak jenis bidang input lainnya.

```
<input type="range">
```

Rentang default slider adalah 0 hingga 100. Kami dapat menggunakan atribut berikut untuk menyesuaikan pengaturan slider:

- 1) max: menentukan nilai maksimum yang diizinkan. Dalam contoh, kami menyetelnya ke 20, tetapi Kami dapat mengubah nilai itu.
- 2) min: menentukan nilai minimum. Dalam hal ini, kami menyetelnya ke 1.
- 3) langkah: menentukan interval nomor. Ini disetel ke 1.
- 4) value: menentukan nilai default slider. Dalam hal ini, sama dengan %TIMERVALUE%.

```
<input type="range" onchange="updateSliderTimer(this)"  
id="timerSlider" min="1" max="20" value="%TIMERVALUE%"  
step="1" class="slider2">
```

%TIMERVALUE% adalah pengganti yang akan diganti dengan nilai sebenarnya. Dalam kode, itu akan diganti dengan nilai variabel timerSliderValue yang diatur ke 10 secara default. Tetapi Kami dapat mengubahnya di baris berikut:

```
String timerSliderValue = "10";
```

Slider memiliki dua atribut lagi: id dan onchange.

- 1) **id**: menentukan id unik untuk elemen HTML (slider). Id memungkinkan kita untuk memanipulasi elemen menggunakan CSS atau JavaScript.
- 2) **onchange**: adalah atribut event yang terjadi ketika kita mengubah nilai elemen (slider). Saat memindahkan slider, itu akan memanggil fungsi updateSliderTimer().

d. Update Slider Value (JavaScript)

Saat memindahkan slider, fungsi updateSliderTimer() dijalankan. Itu mendapat nilai slider saat ini dengan mengacu pada id timerSlider-nya:

```
var sliderValue = document.getElementById("timerSlider").value;
```

Updates the slider label to the current slider value by referring to its id timerValue:

```
document.getElementById("timerValue").innerHTML = sliderValue;
```

Then, it makes a request on the `/slider?value=sliderValue` URL. Where the sliderValue is equal to the current slider value. Then, the ESP32/ESP8266 handles what happens when it receives a request on that URL.

e. Control the Output with Timer (JavaScript)

Saat mengklik tombol ON/OFF untuk mengontrol keluaran, itu akan memanggil fungsi JavaScript toogleCheckbox(). Fungsi ini mendapatkan nilai saat ini dari label slider:

```
var sliderValue = document.getElementById("timerSlider").value;
```

Membuat permintaan pada URL `/update?state=1` sehingga ESP mengetahui bahwa ia perlu menyetel output ke TINGGI.

```
if(element.checked){ xhr.open("GET", "/update?state=1", true);
```

Baris berikut mengurangi nilai label slider setiap detik membuat penghitung waktu mundur.

```
var count = sliderValue, timer = setInterval(function() {
```

```

        count--;
document.getElementById("timerValue").innerHTML = count;
if(count == 0){ clearInterval(timer);
document.getElementById("timerValue").innerHTML =
document.getElementById("timerSlider").value; }
}, 1000);

```

Ketika pengatur waktu mencapai nol, nilai label akan kembali ke nilai aslinya dan permintaan dibuat pada URL `/update?state=0`, sehingga ESP tahu sudah waktunya untuk mengatur output ke LOW. Tombol pada server web akan kembali ke keadaan mati.

```

setTimeout(function(){ xhr.open("GET",
"/update?state=0", true);
document.getElementById(element.id).checked =
false; xhr.send(); }, sliderValue);

```

f. Handle Request

ESP32/ESP8266 perlu menangani apa yang terjadi saat menerima permintaan pada URL tertentu.

1) Root URL

Saat mengakses URL root /, kirim teks HTML yang disimpan pada variabel `index_html`. Semua placeholder diganti dengan nilai aktual oleh fungsi `processor()`.

```

server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
});

```

2) Control Output State

Baris berikut menangani apa yang terjadi saat menerima permintaan pada `/update?state=1` dan `/update?state=0` URL. Ini mengatur status output ke HIGH atau LOW yang sesuai.

```

server.on("/update", HTTP_GET, [] (AsyncWebServerRequest
*request) {
    String inputMessage;
    // GET input1 value on
<ESP_IP>/update?state=<inputMessage>
    if (request->hasParam(PARAM_INPUT_1)) {
        inputMessage =
request->getParam(PARAM_INPUT_1)->value();
        digitalWrite(output, inputMessage.toInt());
    }
    else {
        inputMessage = "No message sent";
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
}

```

```
});
```

g. Update Slider Value

Setiap kali menyeret slider, ESP menerima permintaan dengan nilai baru. Kami menyimpan nilai slider baru dan mencetaknya di Serial Monitor.

```
// Send a GET request to
<ESP_IP>/slider?value=<inputMessage>
server.on("/slider", HTTP_GET, [] (AsyncWebServerRequest
*request) {
    String inputMessage;
    // GET input1 value on
<ESP_IP>/slider?value=<inputMessage>
    if (request->hasParam(PARAM_INPUT_2)) {
        inputMessage =
request->getParam(PARAM_INPUT_2)->value();
        timerSliderValue = inputMessage;
    }
    else {
        inputMessage = "No message sent";
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
});
```

14.5 Demonstration

Mengunggah kode ke papan NodeMCU ESP32 atau ESP8266. Kemudian, buka Serial Monitor dan tekan tombol RST/EN on-board untuk mendapatkan alamat IP. Buka browser di jaringan lokal dan ketik alamat IP ESP. Halaman berikut harus dimuat.



Seret slider untuk menyesuaikan lebar pulsa, lalu klik tombol ON/OFF. Output (dalam hal ini GPIO 2 – built-in LED) akan tetap menyala selama jangka waktu yang telah di atur pada slider.

