LAPORAN PRAKTIKUM ELEKTRONIKA MESIN LISTRIK DAN TEKNIK KENDALI

“Raingkaian ESP 32”

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

Disusun Oleh:

Kelompok 8

Raihan Nur Fadhila (19/447124/SV/16843)

Bagus Sadewa (19/447112/SV/16831)

Kelas: ARM 2

TEKNOLOGI REKAYASA MESIN

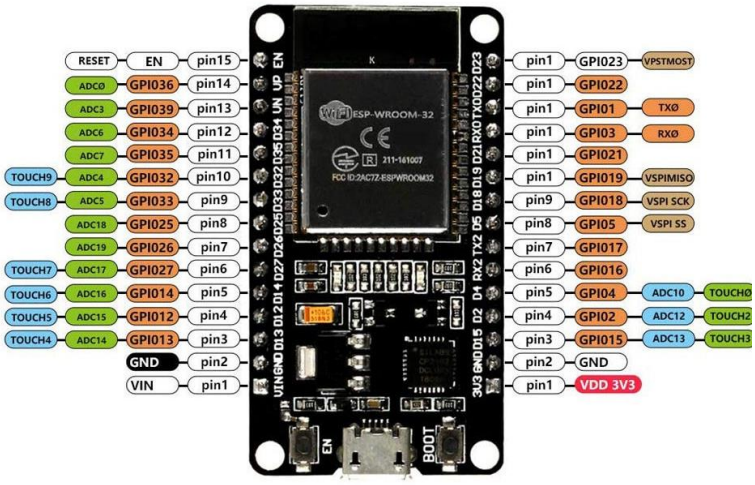
DEPARTEMEN TEKNIK MESIN SEKOLAH VOKASI

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2022

Bab 1 – Spesifikasi Esp32



Esp32 merupakan salah satu board mikrokontroler yang banyak digunakan, programnya pun dapat dikombinasikan dengan berbagai bahasa pemrograman seperti Adruino IDE, Espressif IDF, Microphyton, Javascript dan lainnya. Esp32 memiliki berbagai varian, diantaranya :

1. DOIT DEVKIT V1

2. **ESP32 DEVKIT**

3. **ESP-32S NodeMCU**

4. **ESP Thing**

5. **WEMOS LOLIN32**

6. **“WeMos” OLED**

7. HUZZAH32

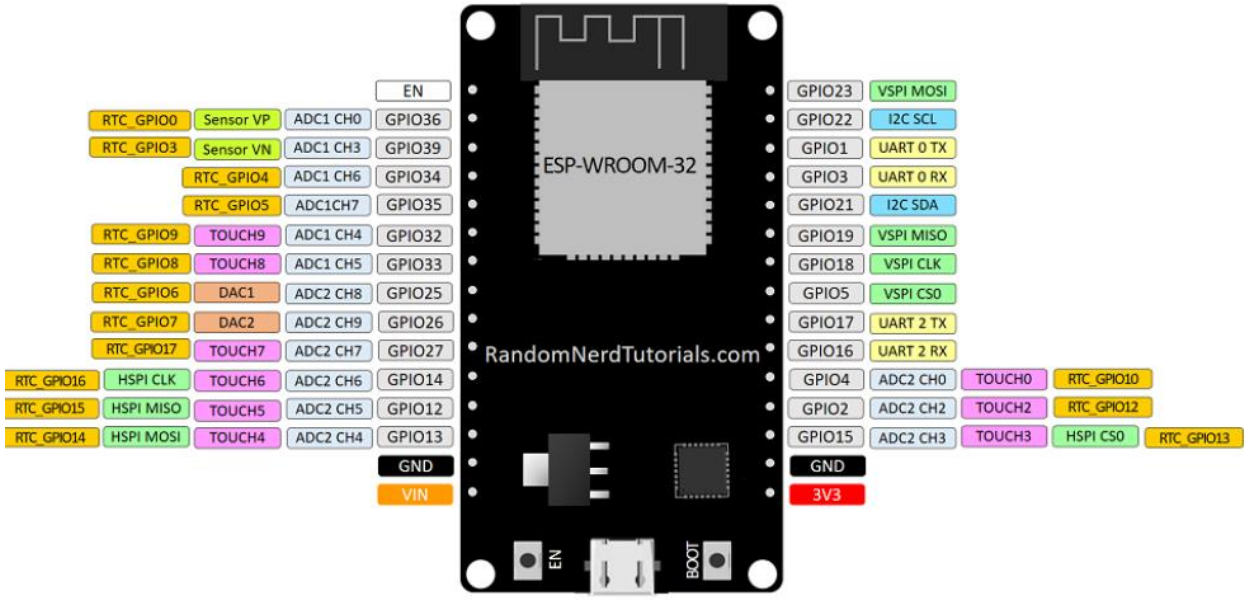
Spesifikasi Esp32 varian - DOIT DEVKIT V1

Beberapa fitur Esp32 DOIT DEVKIT V1 diantaranya sebagai berikut :

- Dapat terkoneksi dengan Wifi dan Bluetooth secara built sehingga dapat dikontrol menggunakan perangkat mobile tanpa modul yang lain.
- ESP32 memiliki 2 prosesor (dual core)
- ESP32 dapat menjalankan program 32 bit.
- Produk modul yang menggunakan chip ESP32 ini memiliki 30 atau 36 pin.
- Frekuensi clock bisa mencapai 240 MHz dan memiliki RAM 512 kB.
- Memiliki sensor efek hall built-in dan suhu built-in
- Memiliki berbagai macam fungsi yang tersedia, seperti: tombol tekan, ADC, DAC, UART, SPI, I2C dan banyak lagi.

Selama praktikum kami menggunakan Esp32 dengan varian DOIT DEVKIT V1 dengan bahasa pemrogaman menggunakan Arduino IDE yang dihubungkan dengan laptop.

Bab 2 – Kaki-kaki/PinOut Esp32

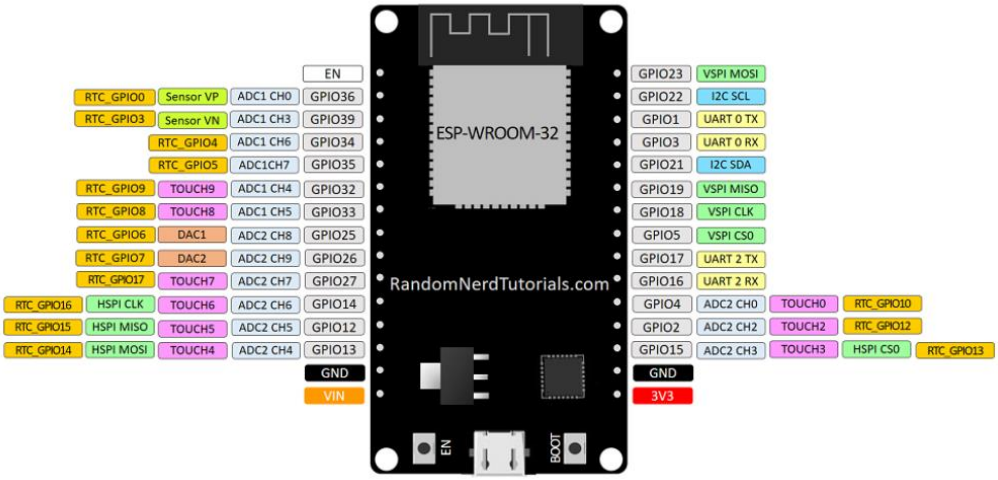


PinOut yang ada di Esp32 DOIT DEVKIT V1 meliputi :

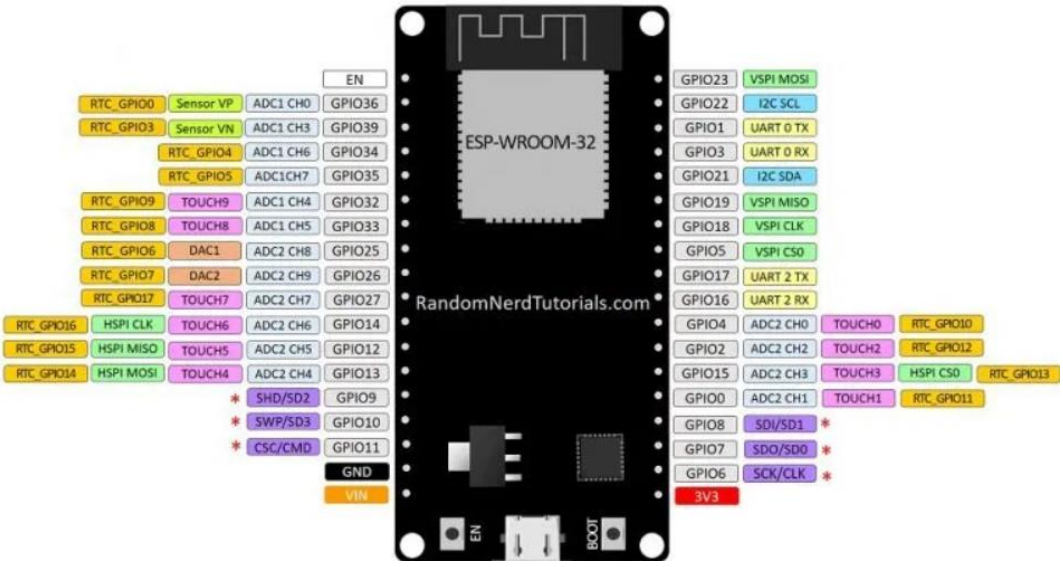
- Analog-to-Digital Converter (ADC) channels
- SPI interfaces
- UART interfaces
- I2C interfaces
- PWM output channels
- Digital-to-Analog Converters (DAC)
- I2S interfaces
- Capacitive sensing GPIOs

PinOut pada Esp32 memiliki istilah GPIO (General Purpose Input-Output), merupakan pin dalam mikrokontroler yang berfungsi untuk menerima sinyal baik input maupun output sesuai pemrograman yang kita tanamkan. Dengan hal tersebut kita juga dapat memerintahkan penggunaan pin dengan penggunaan sebagai SPI, I2C dan UART. Fitur multiplexing chip esp32 dapat memungkinkan adanya sebuah pin yang dapat didefinisikan menjadi beberapa fungsi. Beberapa versi PinOut pada Esp32 DOIT DEVKIT V1 yaitu :

1. **Version with 30 GPIOs**



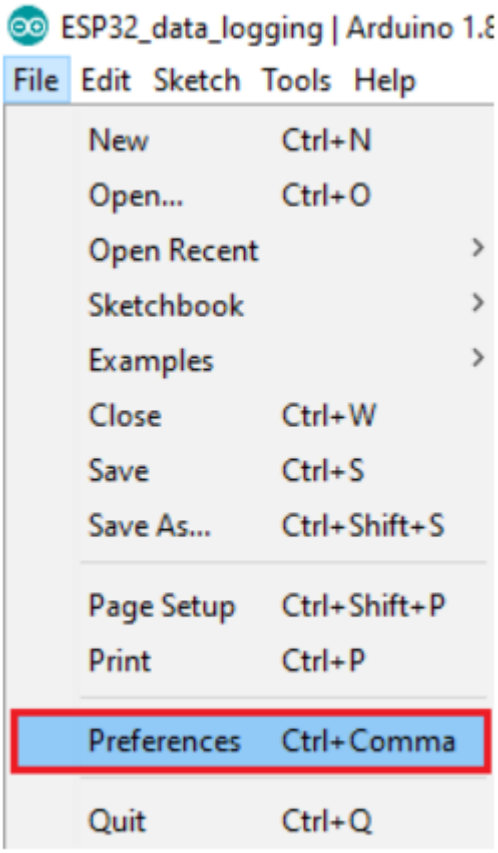
2. **Version with 30 GPIOs**



Bab 3 – Konfigurasi Esp32 pada Arduino

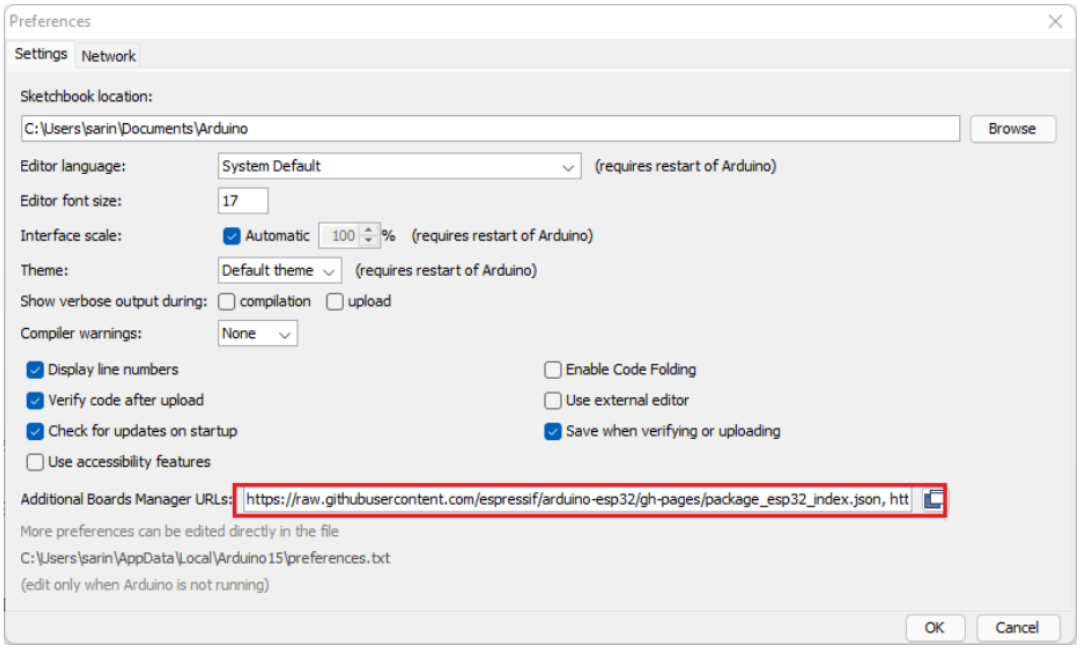
Sebagai *disclaimer* kami menggunakan Esp32 dengan versi DOIT DEVKIT V1 yang memiliki beberapa konfigurasi pada Arduino IDE agar kode yang ada dapat berjalan lancar dan terkoneksi antara Arduino IDE dengan hardware Esp32. Adapun tahap untuk menginstal Add-on di Arduino IDE sebagai konfigurasi, yaitu :

1. Buka aplikasi Arduino IDE, lalu pilih **File**> **Preferences**



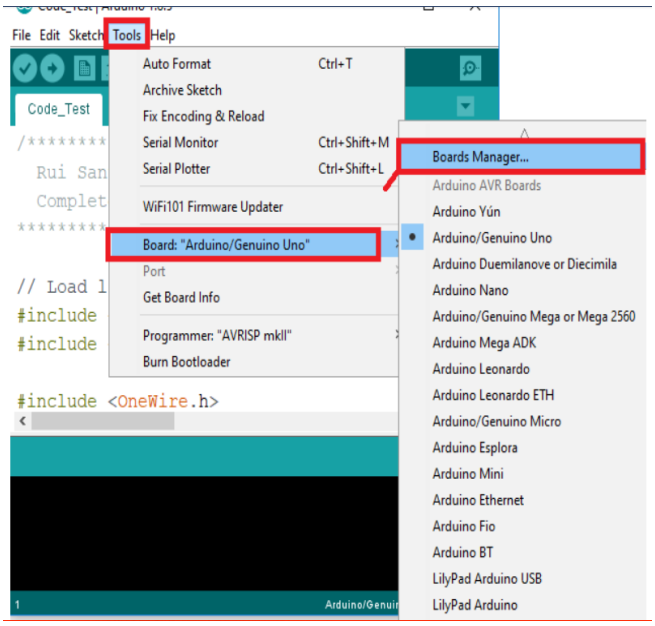
2. Masukan tautan berikut ke dalam pilihan “Additional Board Manager URLs”

```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
```

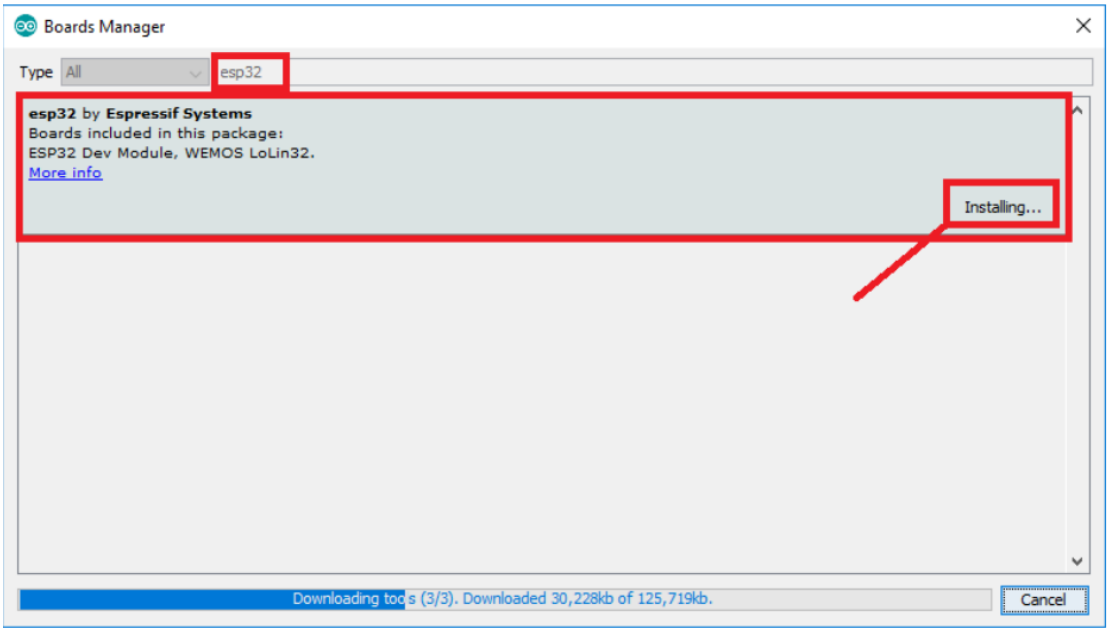


Lalu klik “OK”

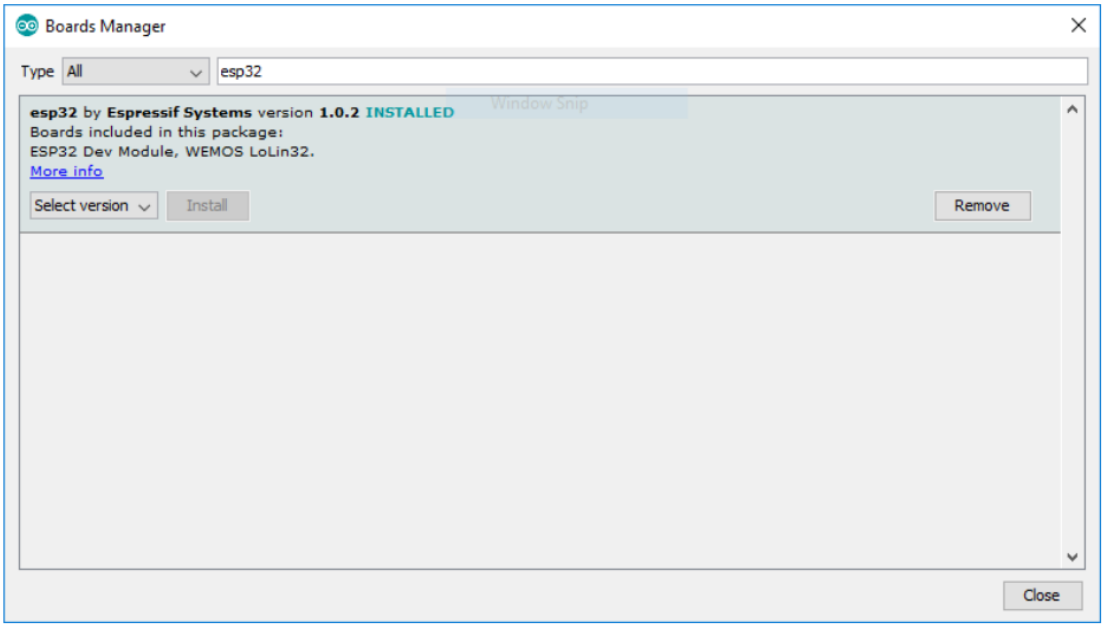
3. Buka menu Boards Manager. Go to **Tools** > **Board** > **Boards Manager**…



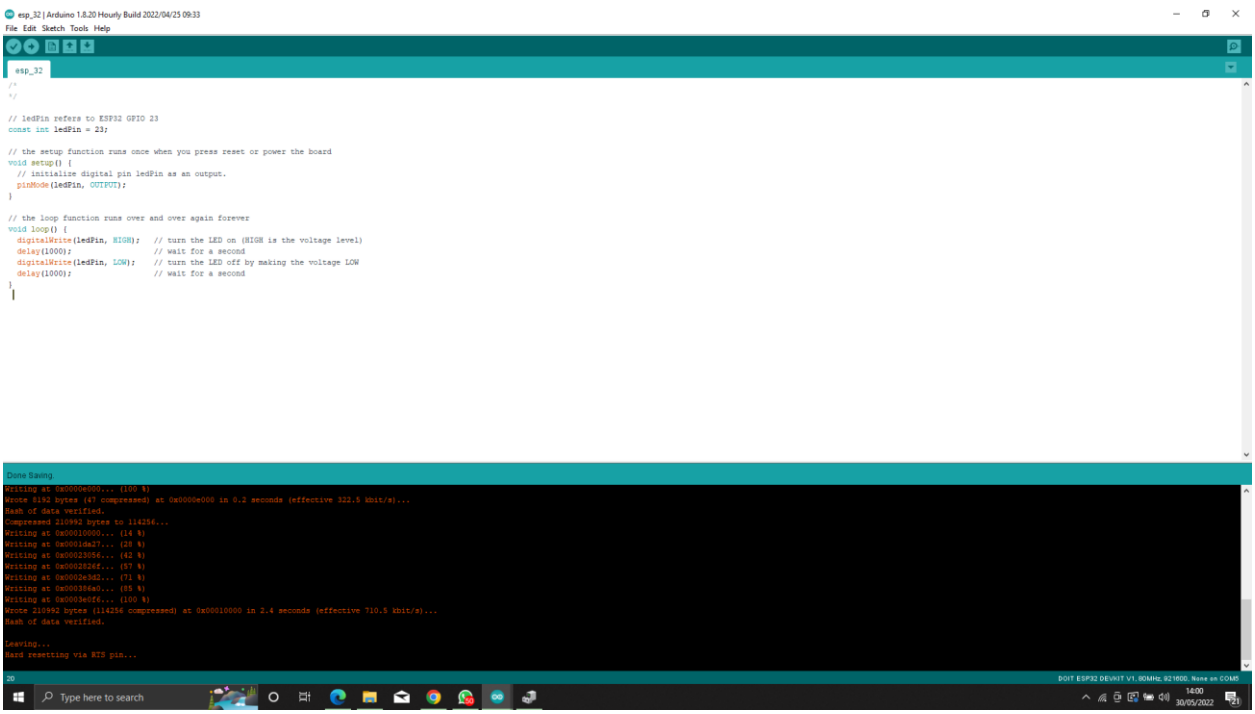
4. Cari ESP32 dan klik tombol opsi yang “**ESP32 by Espressif Systems**“



5. Tunggu hasilnya hingga muncul tanda “**Installed**”



Kami juga melakukan test pada konfigurasi yang telah kami lakukan dengan program lampu “blink”, dengan pemrogaman :

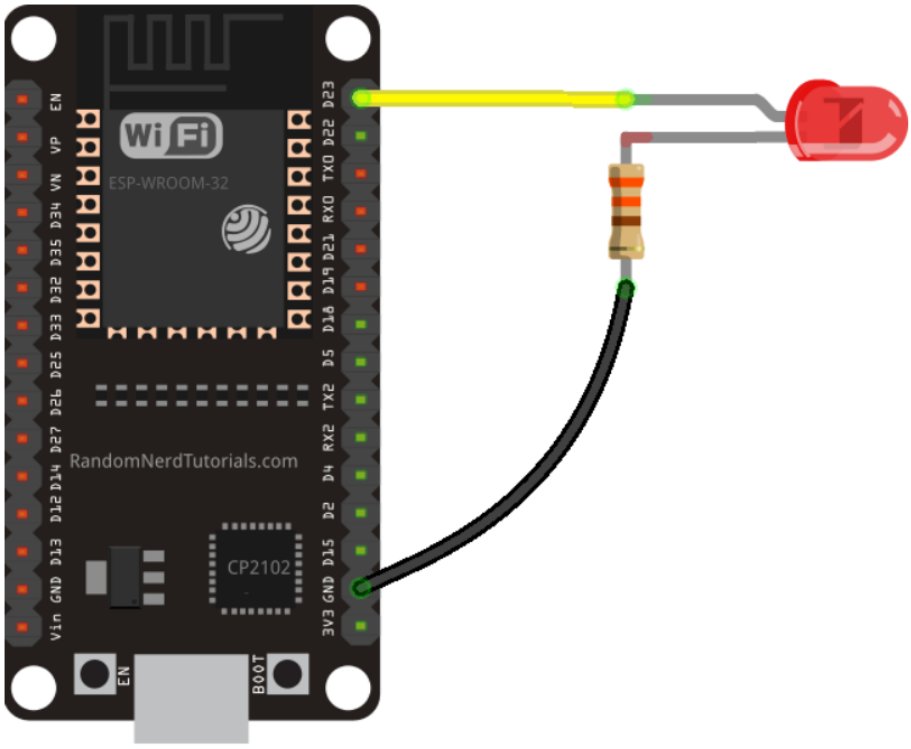


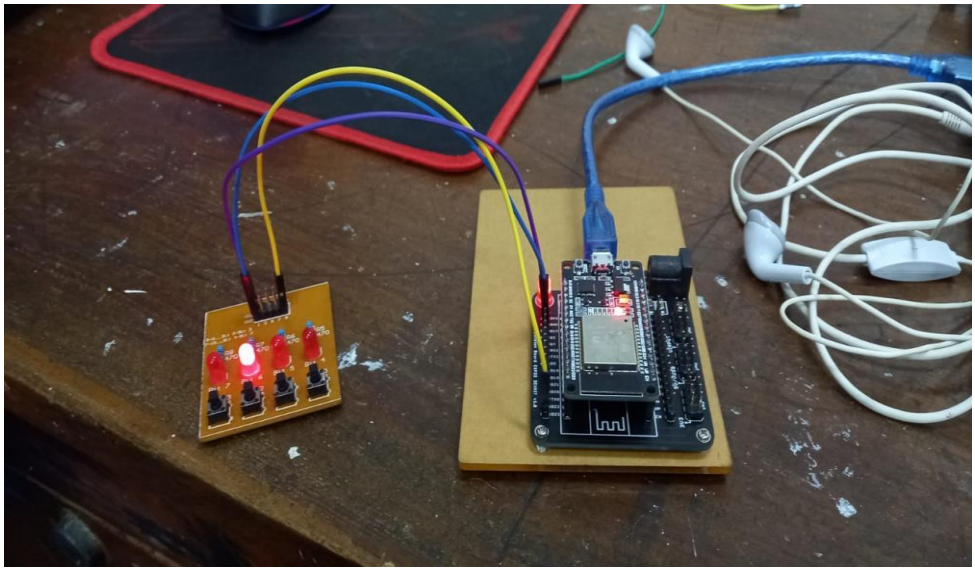
Kode :

```
/*
  Blink
*/


// ledPin refers to ESP32 GPIO 23
const int ledPin = 23;
```

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin ledPin as an output.
  pinMode(ledPin, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(ledPin, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                  // wait for a second
}
```

Wiring pada Esp32 :





Cara kerja :

Mengontrol lampu LED yang dikoneksikan ke pin GPIO 23. Lalu tekan tombol  untuk mengupload ke hardware Esp32

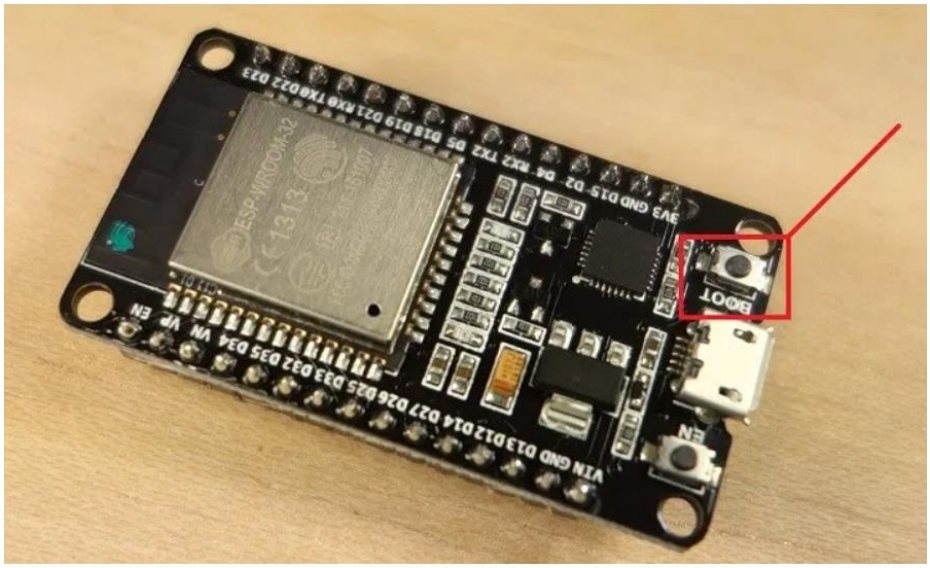
Permasalahan :

1. **Serial Port antara Esp32 dan Laptop tidak terhubung**
   Solusinya ialah melakukan update pada serial port

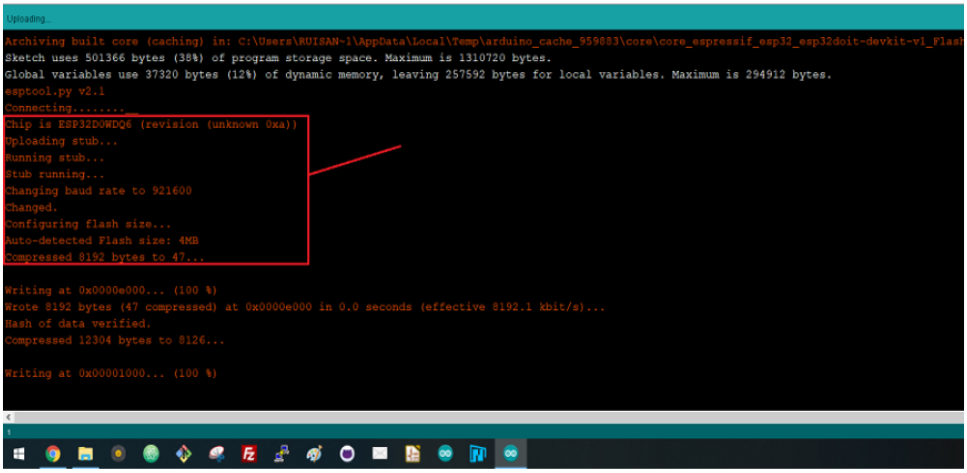
2. **Adanya masalah board dan comm yang tidak mucul pada mennu pilihan “Tools”**
   Solusinya dengan melakukan penginstalan ulang perangkat Esp32 pada Arduino

3. **Adanya “Fatal Eror/Failed to connect Esp32” yang mengakibatkan program Arduino IDE tidak dapat terhubung ke Laptop**
   Solusinya dengan melakukan adanya tindakan berikut :

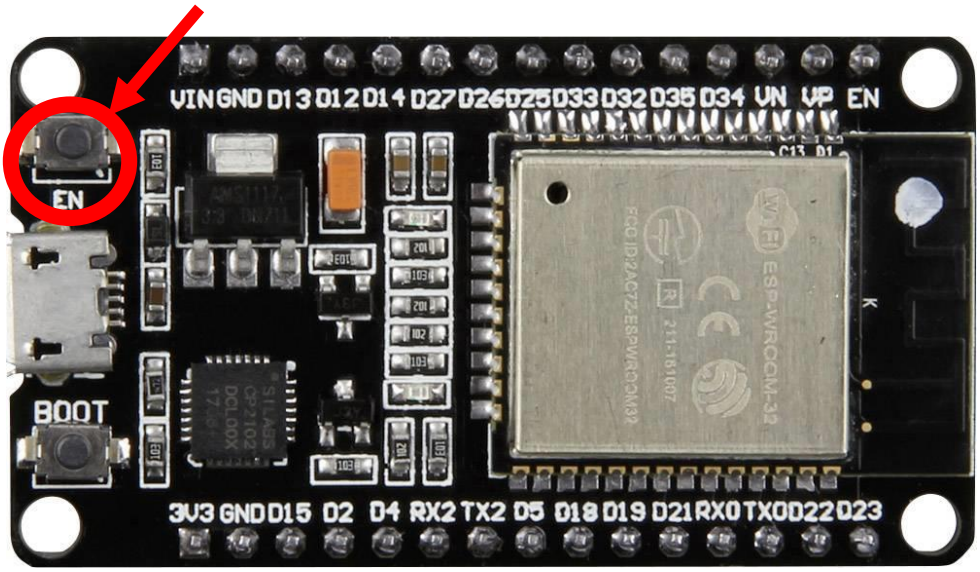
   1) Tekan tombol “Boot” dan tahan, lalu klik upload



2) Tunggu hingga “Done Uploading”, baru lepas tombol “Boot”

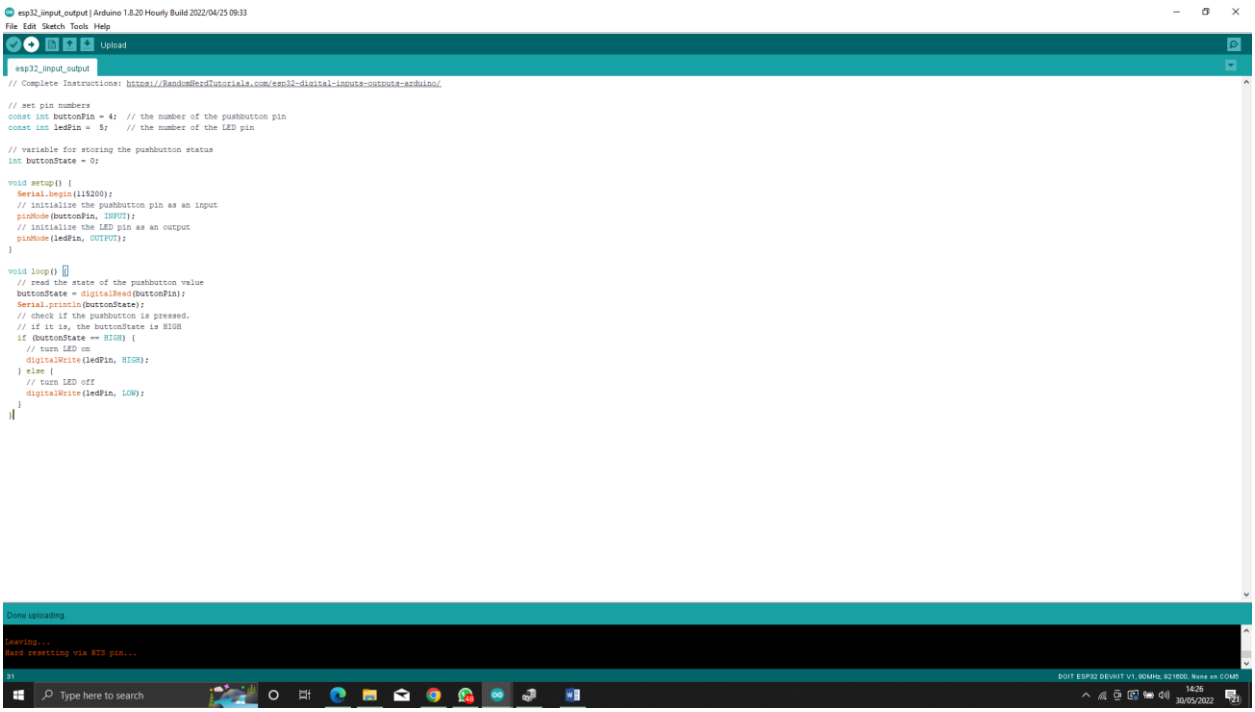


3) Tekan tombol “EN” untuk meng-*enable* program dari Arduino IDE ke Esp32



Bab 4 – Esp32 Input/Output

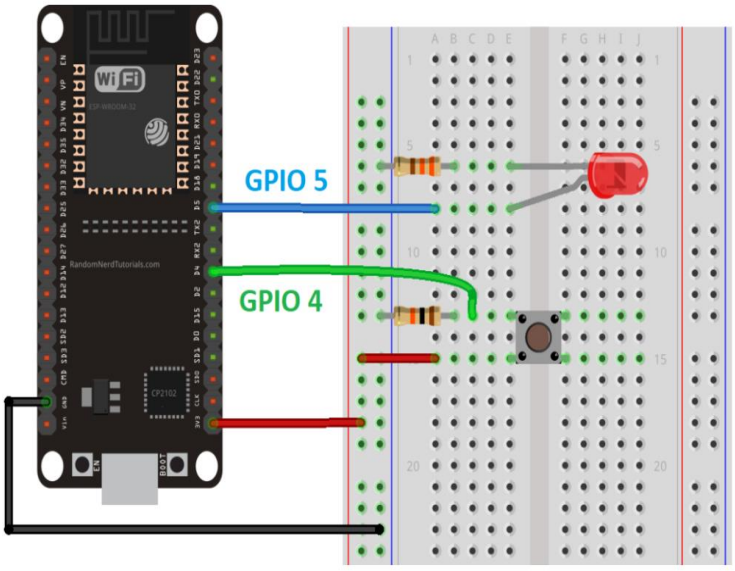
4.1 Kode untuk Arduino IDE

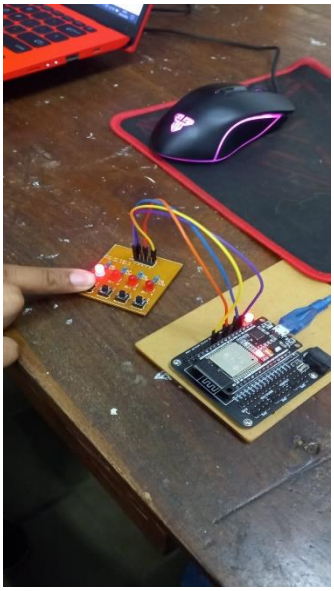



```
// Complete Instructions:
https://RandomNerdTutorials.com/esp32-digital-inputs-outputs-
arduino/

// 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;
```

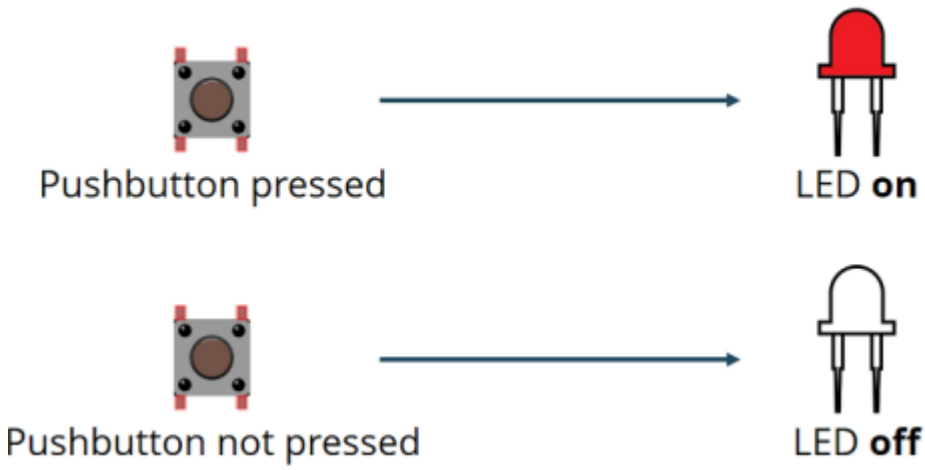
```
void setup() {
  Serial.begin(115200);
  // initialize the pushbutton pin as an input
  pinMode(buttonPin, INPUT);
  // initialize the LED pin as an output
  pinMode(ledPin, OUTPUT);
}

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);
  }
}
```

4.2 Wiring dari Arduino IDE ke Esp32





4.3 Cara Kerja :



Konfigurasi Input digital:

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).

Konfigurasi Output:

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

pinMode(GPIO, OUTPUT);

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

digitalRead(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).

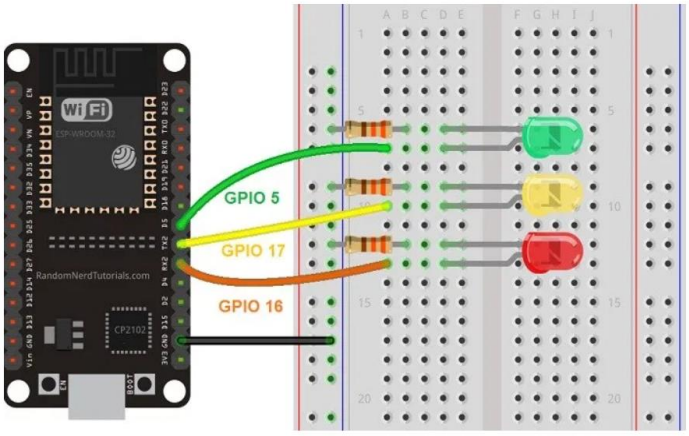
Bab 5 – Esp32 PWM

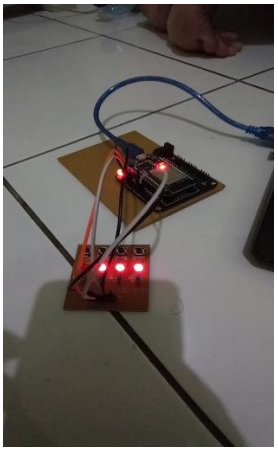
5.1 Kode untuk Arduino IDE


```
// the number of the LED pin
const int ledPin = 15;  // 16 corresponds to GPIO16
const int ledPin2 = 18; // 17 corresponds to GPIO17
const int ledPin3 = 5;  // 5 corresponds to GPIO5

// 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);
  ledcAttachPin(ledPin3, 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);
  }
}
```

5.2 Wiring dari Arduino IDE ke Esp32





5.3 Cara Kerja :

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 aman 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.

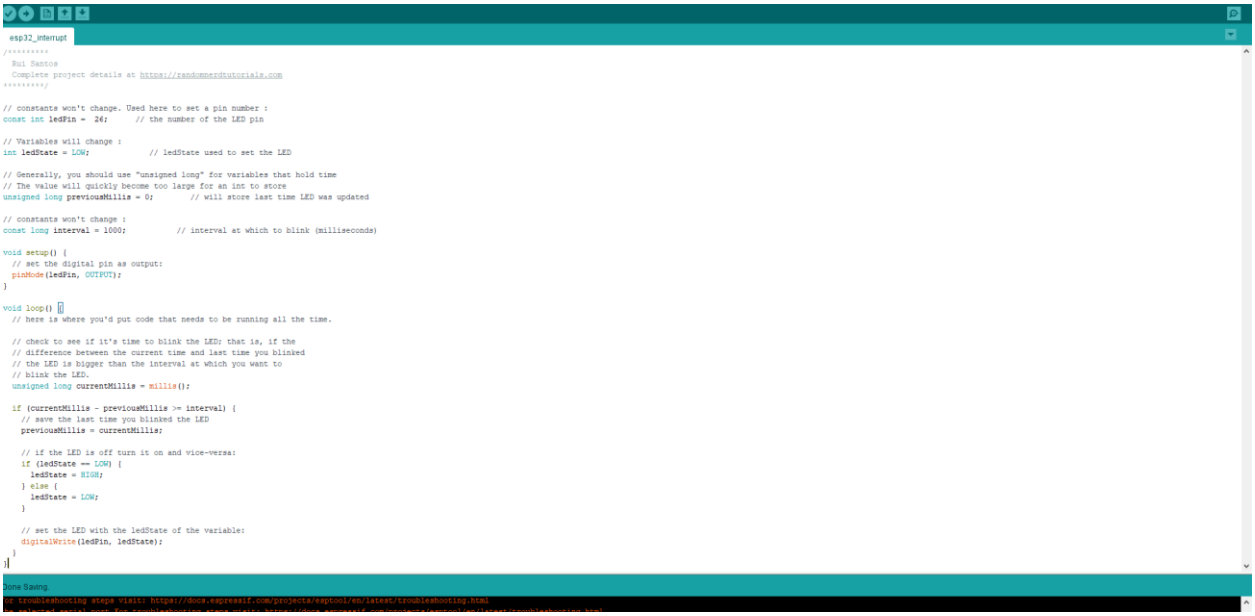
5. Terakhir, untuk mengontrol kecerahan LED menggunakan PWM, digunakan fungsi berikut:

   ledcWrite(channel, dutycycle)

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

Bab 6 – Esp32 Interrupt

6.1 Kode untuk Arduino IDE

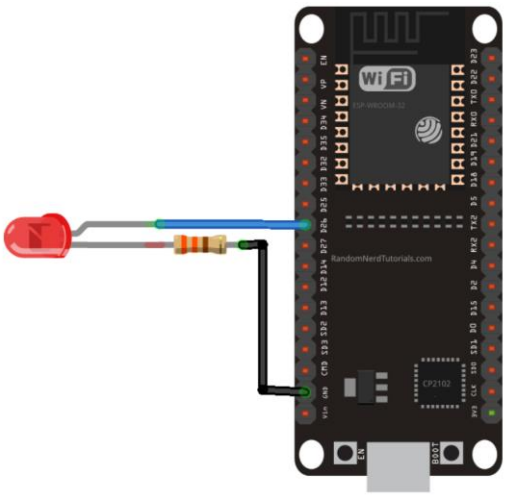


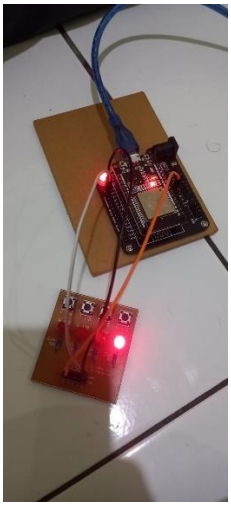

```
/*********
  Rui Santos
  Complete project details at https://randomnerdtutorials.com
*********/

// constants won't change. Used here to set a pin number :
```

```
const int ledPin =  26;      // the number of the LED pin

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

// Generally, you should use "unsigned long" for variables that
hold time
// The value will quickly become too large for an int to store
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() {
  // here is where you'd put code that needs to be running all
the time.

  // check to see if it's time to blink the LED; that is, if
the
```

```
  // difference between the current time and last time you
blinked
  // the LED is bigger than the interval at which you want to
  // blink the LED.
  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);
  }
}
```

6.2 Wiring dari Arduino IDE ke Esp32





6.3 Cara Kerja :

Untuk men-trigger suatu kerja dari sensor gerak PIR. Dapat membuat sesuatu menjadi terjadi dengan sendirinya dalam suatu program mikrokontroller dan juga dapat memecahkan masalah waktu.

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);

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 HI

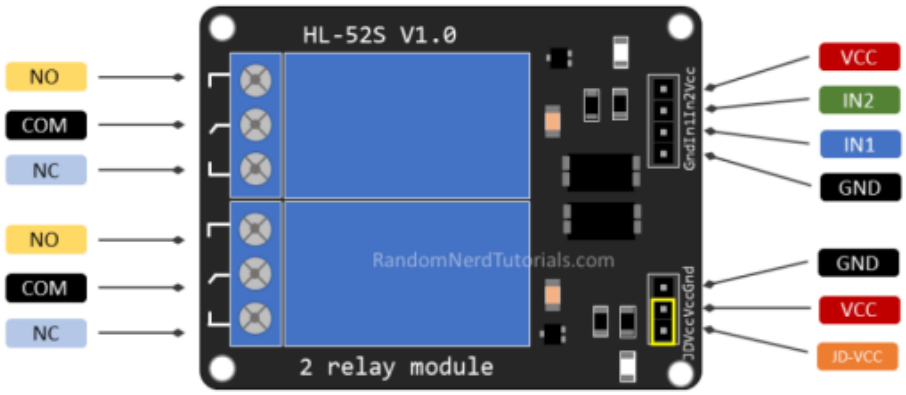
Adapun fungsi delay dan milis, delay berfungsi untuk menunda program dalam waktu tertentu . Sedangkan milis berfungsi untuk mengembalikan jumlah detik yang ada tanpa harus memblokir porgram.

Bab 7 – Esp32 Relay

7.1 Pengertian Relay

Relay ialah sejenis saklar yang dapat dioperasikan menggunakan program dan berfungsi sebagai penyambung maupun pemutus arus.

1. **Pinout Relay**



Ini adalah modul relay dengan 2 saluran. Pada sisi kiri ada 2 dua set tiga soket untuk menghubungkan tegangan tinggi dan pin di sisi kanan (tegangan rendah) terhubung ke GPIO ESP32

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) :

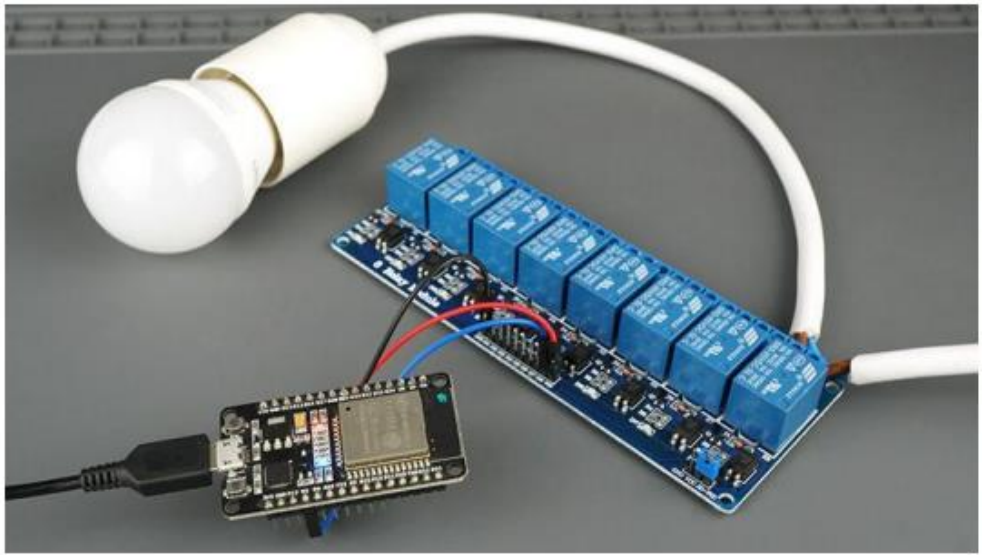
- Sinyal HIGH – arus tidak mengalir
- Sinyal LOW – arus mengalir

7.3 Kode untuk mengontrol Relay dengan Esp32

```
/*********
  Rui Santos
  Complete project details at
https://RandomNerdTutorials.com/esp32-relay-module-ac-web-
server/

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

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);
}
```

7.4 Wiring untuk mengontrol Relay dengan Esp32



7.5 Cara kerja mengontrol Relay dengan Esp32

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 menyalakan lampu.

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

Jika Kami menggunakan konfigurasi yang biasanya tertutup, kirim HIGH signal untuk menyalakan 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);
```

7.5 Kode untuk mengontrol Relay dengan Esp32 WebServer

Sebelum pengkodean, lakukan langkah berikut :

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

Kode :

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-relay-module-ac-web-server/

  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 "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>
```

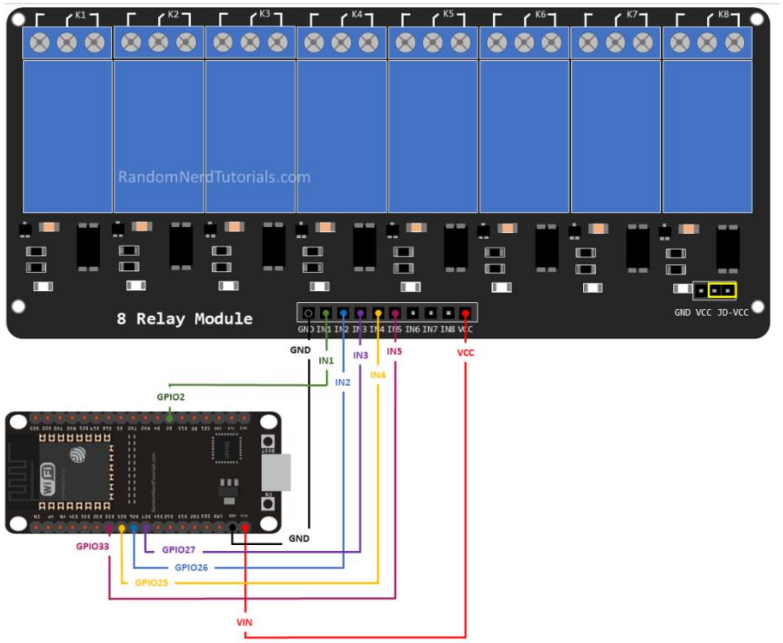
```
<head>
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <style>
    html {font-family: Arial; display: inline-block; text-
align: center;}
    h2 {font-size: 3.0rem;}
    p {font-size: 3.0rem;}
    body {max-width: 600px; margin:0px auto; padding-
bottom: 25px;}
    .switch {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() {

}
```

7.6 Wiring untuk mengontrol Relay dengan Esp32 WebServer



7.7 Cara Kerja mengontrol Relay dengan Esp32 WebServer

Sama seperti cara kerja mengontrol Relay ke Esp32, hanya saja memerlukan input SSID dan Password ke wifi agar bisa deikendalikan melalui browser.

const char ssid = "REPLACE_WITH_YOUR_SSID";*

const char password = "REPLACE_WITH_YOUR_PASSWORD";*

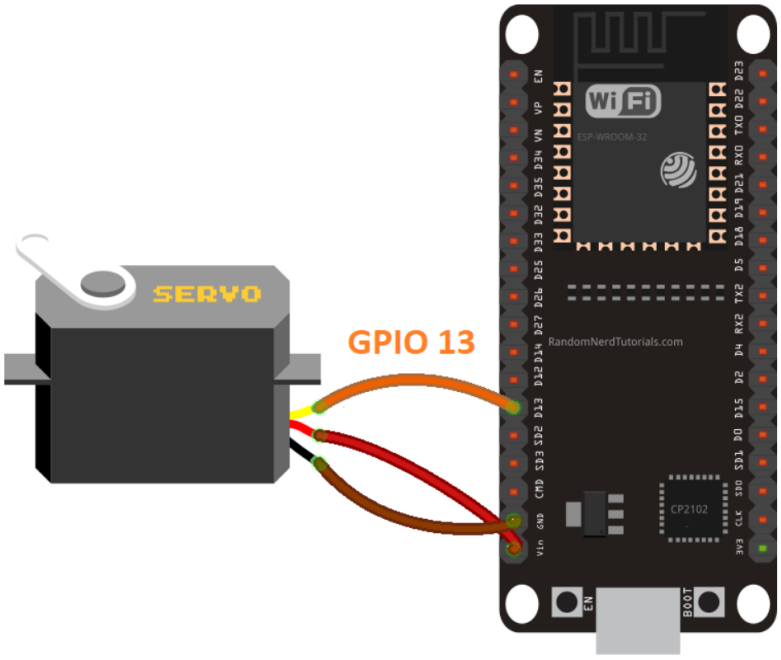
Bab 8 – Esp32 Servo

8.1 Kode untuk Arduino IDE


```
/*********
  Rui Santos
  Complete project details at
https://randomnerdtutorials.com
  Written by BARRAGAN and modified by Scott Fitzgerald
*********/

#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
  }
}
```

8.2 Wiring dari Arduino IDE ke Esp32



8.3 Cara Kerja :

Sketsa ini memutar servo 180 derajat ke satu sisi, dan 180 derajat ke sisi lainnya. 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 ()**

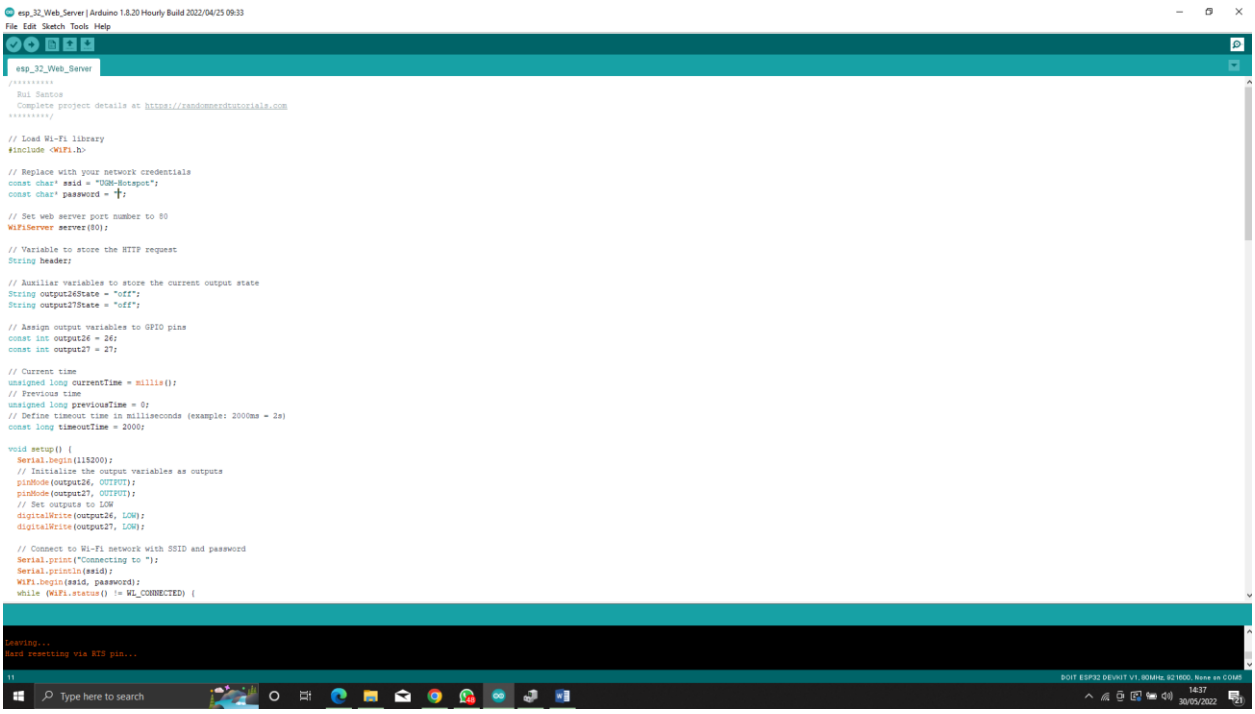
Dalam loop(), kami mengubah posisi poros motor dari 0 menjadi 180 derajat, dan kemudian dari 180 menjadi 0 derajat. Untuk mengatur poros ke posisi tertentu, hanya perlu menggunakan metode write() dalam servo objek.

myservo.write(pos);

Kontrol pada WebServer untuk Servo

Bab 9 – Esp32 Output WebServer

9.1 Kode untuk Arduino IDE



```
/*********
  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: ");
```

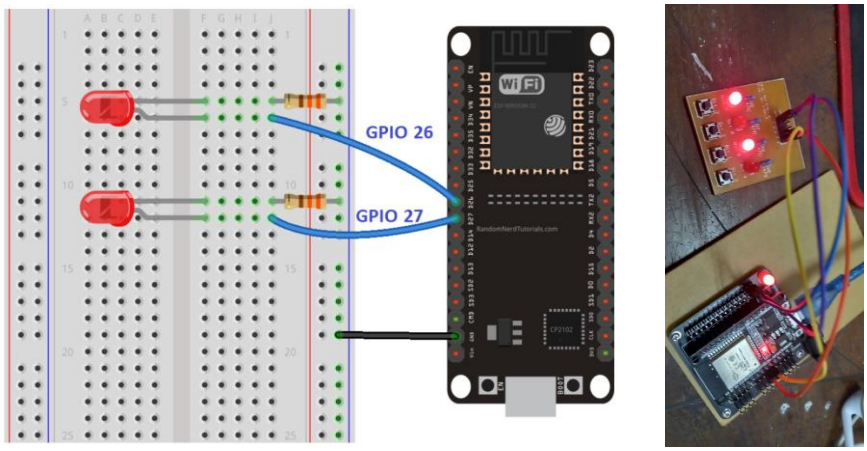
```
  Serial.println(WiFi.localIP());
  server.begin();
}

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("");
  }
}
```

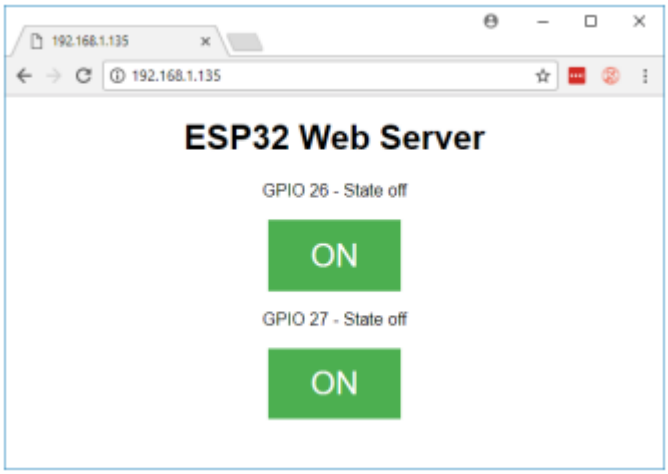
9.2 Wiring dari Arduino IDE ke Esp32



9.3 Cara Kerja :

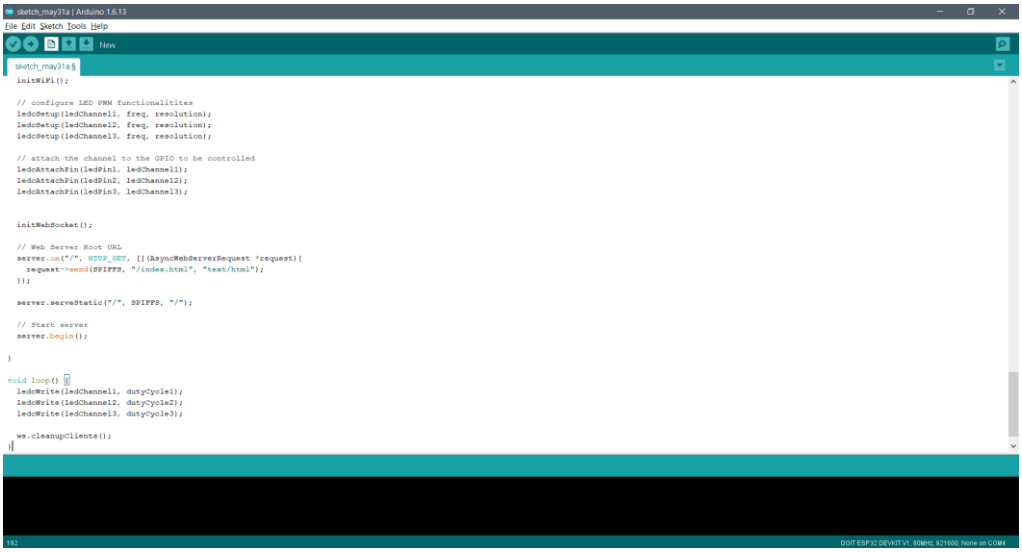
Dengan mengatur kredensial jaringan, kita dapat mengontrol Esp32 lewat browser

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



Bab 10 – Esp32 WebServer with Slider

10.1 Kode untuk Arduino IDE



Kode untuk Arduino ke Esp32

```
/*
  Rui Santos
  Complete project details at
https://RandomNerdTutorials.com/esp32-web-server-websocket-
sliders/

  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.
*/

#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "SPIFFS.h"
#include <Arduino_JSON.h>

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

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

AsyncWebSocket ws("/ws");
// Set LED GPIO
const int ledPin1 = 12;
const int ledPin2 = 13;
const int ledPin3 = 14;

String message = "";
String sliderValue1 = "0";
String sliderValue2 = "0";
```

```
String sliderValue3 = "0";

int dutyCycle1;
int dutyCycle2;
int dutyCycle3;

// setting PWM properties
const int freq = 5000;
const int ledChannel1 = 0;
const int ledChannel2 = 1;
const int ledChannel3 = 2;

const int resolution = 8;

//Json Variable to Hold Slider Values
JSONVar sliderValues;

//Get Slider Values
String getSliderValues(){
  sliderValues["sliderValue1"] = String(sliderValue1);
  sliderValues["sliderValue2"] = String(sliderValue2);
  sliderValues["sliderValue3"] = String(sliderValue3);

  String jsonString = JSON.stringify(sliderValues);
  return jsonString;
}

// Initialize SPIFFS
void initFS() {
  if (!SPIFFS.begin()) {
    Serial.println("An error has occurred while mounting
SPIFFS");
  }
  else{
   Serial.println("SPIFFS mounted successfully");
  }
}

// Initialize WiFi
void initWiFi() {
```

```
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}

void notifyClients(String sliderValues) {
  ws.textAll(sliderValues);
}

void handleWebSocketMessage(void *arg, uint8_t *data,
size_t len) {
  AwsFrameInfo *info = (AwsFrameInfo*)arg;
  if (info->final && info->index == 0 && info->len == len
&& info->opcode == WS_TEXT) {
    data[len] = 0;
    message = (char*)data;
    if (message.indexOf("1s") >= 0) {
      sliderValue1 = message.substring(2);
      dutyCycle1 = map(sliderValue1.toInt(), 0, 100, 0,
255);
      Serial.println(dutyCycle1);
      Serial.print(getSliderValues());
      notifyClients(getSliderValues());
    }
    if (message.indexOf("2s") >= 0) {
      sliderValue2 = message.substring(2);
      dutyCycle2 = map(sliderValue2.toInt(), 0, 100, 0,
255);
      Serial.println(dutyCycle2);
      Serial.print(getSliderValues());
      notifyClients(getSliderValues());
    }
    if (message.indexOf("3s") >= 0) {
      sliderValue3 = message.substring(2);
```

```
      dutyCycle3 = map(sliderValue3.toInt(), 0, 100, 0,
255);
      Serial.println(dutyCycle3);
      Serial.print(getSliderValues());
      notifyClients(getSliderValues());
    }
    if (strcmp((char*)data, "getValues") == 0) {
      notifyClients(getSliderValues());
    }
  }
}
void onEvent(AsyncWebSocket *server, AsyncWebSocketClient
*client, AwsEventType type, void *arg, uint8_t *data,
size_t len) {
  switch (type) {
    case WS_EVT_CONNECT:
      Serial.printf("WebSocket client #%u connected from
%s\n", client->id(), client-
>remoteIP().toString().c_str());
      break;
    case WS_EVT_DISCONNECT:
      Serial.printf("WebSocket client #%u disconnected\n",
client->id());
      break;
    case WS_EVT_DATA:
      handleWebSocketMessage(arg, data, len);
      break;
    case WS_EVT_PONG:
    case WS_EVT_ERROR:
      break;
  }
}

void initWebSocket() {
  ws.onEvent(onEvent);
  server.addHandler(&ws);
}


void setup() {
```

```
  Serial.begin(115200);
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);
  initFS();
  initWiFi();

  // configure LED PWM functionalitites
  ledcSetup(ledChannel1, freq, resolution);
  ledcSetup(ledChannel2, freq, resolution);
  ledcSetup(ledChannel3, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(ledPin1, ledChannel1);
  ledcAttachPin(ledPin2, ledChannel2);
  ledcAttachPin(ledPin3, ledChannel3);


  initWebSocket();

  // Web Server Root URL
  server.on("/", HTTP_GET, [](AsyncWebServerRequest
*request){
    request->send(SPIFFS, "/index.html", "text/html");
  });

  server.serveStatic("/", SPIFFS, "/");

  // Start server
  server.begin();

}

void loop() {
  ledcWrite(ledChannel1, dutyCycle1);
  ledcWrite(ledChannel2, dutyCycle2);
  ledcWrite(ledChannel3, dutyCycle3);

  ws.cleanupClients();
}
```

Kode untuk HTML File

```
<!-- Complete project details:
https://randomnerdtutorials.com/esp32-web-server-websocket-
sliders/ -->

<!DOCTYPE html>
<html>
<head>
    <title>ESP IOT DASHBOARD</title>
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <link rel="icon" type="image/png" href="favicon.png">
    <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
    <div class="topnav">
        <h1>Multiple Sliders</h1>
    </div>
    <div class="content">
        <div class="card-grid">
            <div class="card">
                <p class="card-title">Fader 1</p>
                <p class="switch">
                    <input type="range"
onchange="updateSliderPWM(this)" id="slider1" min="0"
max="100" step="1" value ="0" class="slider">
                </p>
                <p class="state">Brightness: <span
id="sliderValue1"></span> &percnt;</p>
            </div>
            <div class="card">
                <p class="card-title"> Fader 2</p>
                <p class="switch">
                    <input type="range"
onchange="updateSliderPWM(this)" id="slider2" min="0"
max="100" step="1" value ="0" class="slider">
                </p>
                <p class="state">Brightness: <span
id="sliderValue2"></span> &percnt;</p>
```

```
            </div>
            <div class="card">
                <p class="card-title"> Fader 3</p>
                <p class="switch">
                    <input type="range"
onchange="updateSliderPWM(this)" id="slider3" min="0"
max="100" step="1" value ="0" class="slider">
                </p>
                <p class="state">Brightness: <span
id="sliderValue3"></span> &percnt;</p>
            </div>
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Kode untuk CSS File

```
/*  Complete project details:
https://randomnerdtutorials.com/esp32-web-server-websocket-
sliders/  */

html {
    font-family: Arial, Helvetica, sans-serif;
    display: inline-block;
    text-align: center;
  }
  h1 {
    font-size: 1.8rem;
    color: white;
  }
  p {
    font-size: 1.4rem;
  }
  .topnav {
    overflow: hidden;
    background-color: #0A1128;
  }
  body {
    margin: 0;
```

```
  }
  .content {
    padding: 30px;
  }
  .card-grid {
    max-width: 700px;
    margin: 0 auto;
    display: grid;
    grid-gap: 2rem;
    grid-template-columns: repeat(auto-fit, minmax(200px,
1fr));
  }
  .card {
    background-color: white;
    box-shadow: 2px 2px 12px 1px rgba(140,140,140,.5);
  }
  .card-title {
    font-size: 1.2rem;
    font-weight: bold;
    color: #034078
  }
  .state {
    font-size: 1.2rem;
    color:#1282A2;
  }
  .slider {
    -webkit-appearance: none;
    margin: 0 auto;
    width: 100%;
    height: 15px;
    border-radius: 10px;
    background: #FFD65C;
    outline: none;
  }
  .slider::-webkit-slider-thumb {
    -webkit-appearance: none;
    appearance: none;
    width: 30px;
    height: 30px;
    border-radius: 50%;
```

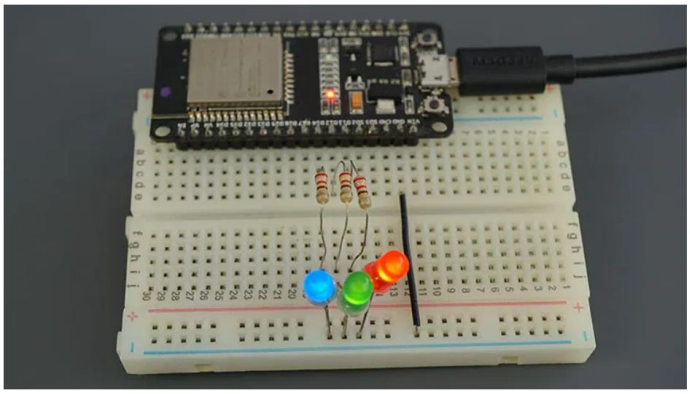
```
    background: #034078;
    cursor: pointer;
  }
  .slider::-moz-range-thumb {
    width: 30px;
    height: 30px;
    border-radius: 50% ;
    background: #034078;
    cursor: pointer;
  }
  .switch {
    padding-left: 5%;
    padding-right: 5%;
  }
```

Kode untuk Javascript

```
// Complete project details:
https://randomnerdtutorials.com/esp32-web-server-websocket-
sliders/

var gateway = `ws://${window.location.hostname}/ws`;
var websocket;
window.addEventListener('load', onload);

function onload(event) {
    initWebSocket();
}

function getValues(){
    websocket.send("getValues");
}

function initWebSocket() {
    console.log('Trying to open a WebSocket connection…');
    websocket = new WebSocket(gateway);
    websocket.onopen = onOpen;
    websocket.onclose = onClose;
    websocket.onmessage = onMessage;
}
```

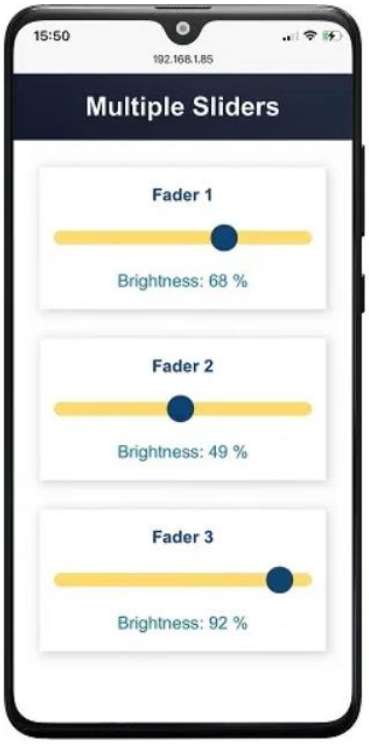
```
function onOpen(event) {
    console.log('Connection opened');
    getValues();
}

function onClose(event) {
    console.log('Connection closed');
    setTimeout(initWebSocket, 2000);
}

function updateSliderPWM(element) {
    var sliderNumber = element.id.charAt(element.id.length-
1);
    var sliderValue =
document.getElementById(element.id).value;

document.getElementById("sliderValue"+sliderNumber).innerHT
ML = sliderValue;
    console.log(sliderValue);

websocket.send(sliderNumber+"s"+sliderValue.toString());
}

function onMessage(event) {
    console.log(event.data);
    var myObj = JSON.parse(event.data);
    var keys = Object.keys(myObj);

    for (var i = 0; i < keys.length; i++){
        var key = keys[i];
        document.getElementById(key).innerHTML =
myObj[key];
        document.getElementById("slider"+
(i+1).toString()).value = myObj[key];
    }
}
```

10.2 Wiring dari Arduino IDE ke Esp32



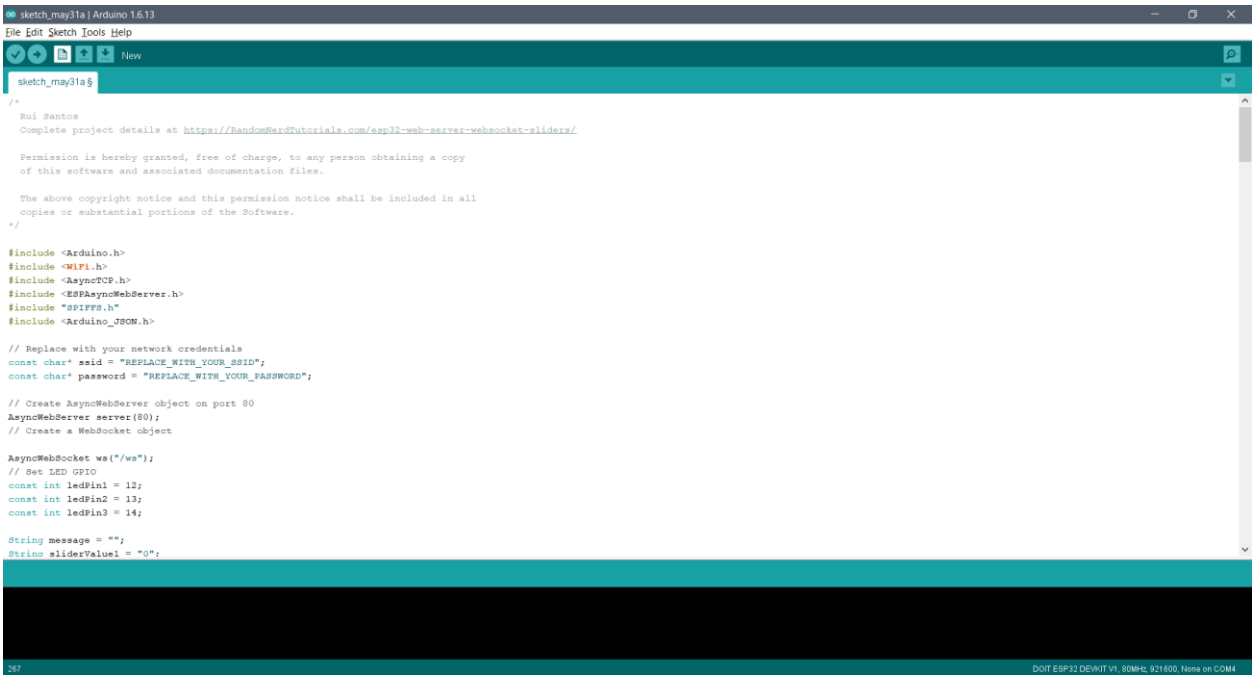
10.3 Cara Kerja :

Esp32 di sini dapat mengatur tingkat brightness pada 3 lampu sekaligus. Kode pemrograman HTTP, CSS, hingga Javascript untuk mengatur tampilan laman Web hingga muncul gambar seperti berikut



Bab 11 – Momentary Switch WebServer

11.1 Kode untuk Arduino IDE



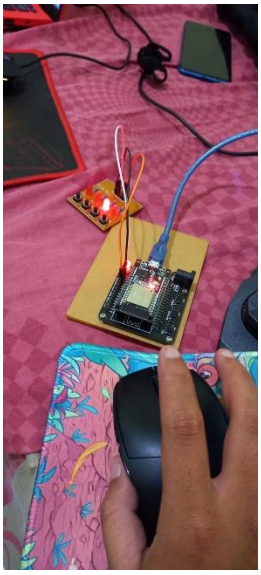

```
/*********
  Rui Santos
  Complete project details at
https://RandomNerdTutorials.com/esp32-esp8266-web-server-
outputs-momentary-switch/

  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.
*********/

#ifdef 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, [] (AsyncWebServerRequest
*request) {
    digitalWrite(output, HIGH);
    request->send(200, "text/plain", "ok");
  });

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

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

void loop() {

}
```

11.2 Wiring dari Arduino IDE ke Esp32

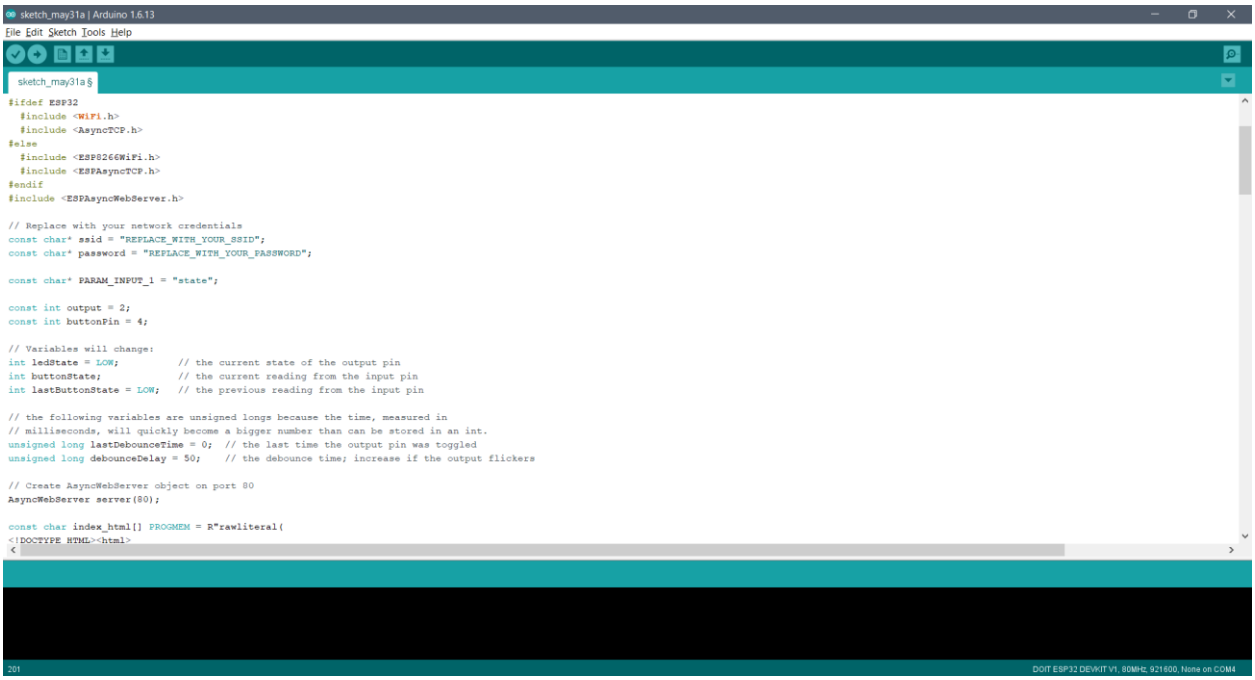
11.3 Cara Kerja :

Esp32 terkoneksi dengan WebServer yang memungkinkan kita mengontrol nyala lampu secara momentary (kali ini, ketika tombol on ditekan menerus, lampu akan mati. Jika tombol on dilepas, maka lampu akan menyala) dengan kredensial wifi yang terhubung.

```
// Replace with your network credentials

const char* ssid = "REPLACE_WITH_YOUR_SSID";

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

Bab 12 – Esp32 Physical Button WebServer

12.1 Kode untuk Arduino IDE




```
/*********
  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
#ifdef 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}
    .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";
      }
```

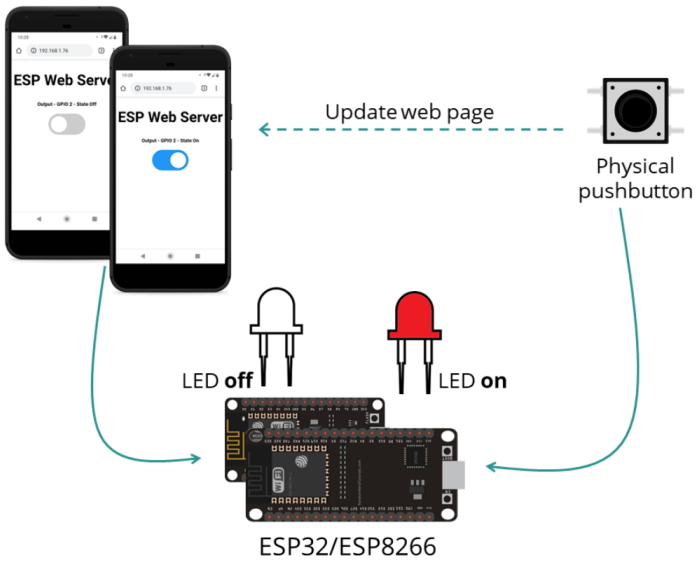
```
      else {
        inputChecked = false;
        outputStateM = "Off";
      }
      document.getElementById("output").checked =
inputChecked;
      document.getElementById("outputState").innerHTML =
outputStateM;
    }
  };
  xhttp.open("GET", "/state", true);
  xhttp.send();
}, 1000 ) ;
</script>
</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";
  }
```

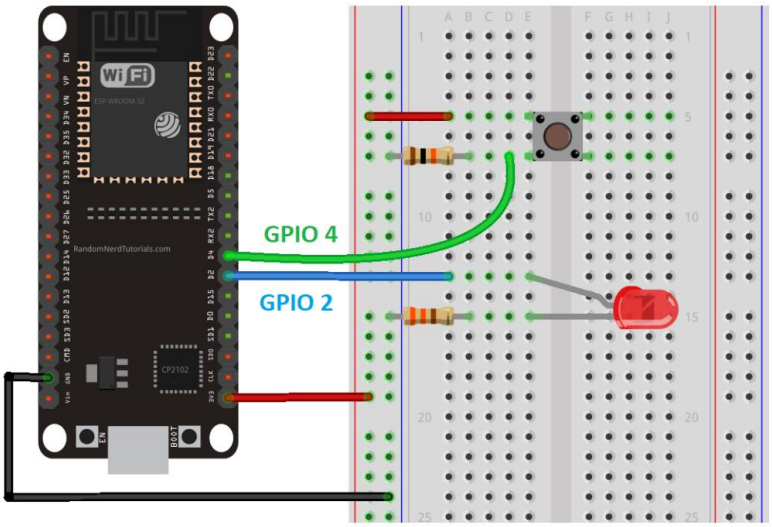
```
  else {
    return "";
  }
  return "";
}

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;
}
```

12.2 Wiring dari Arduino IDE ke Esp32





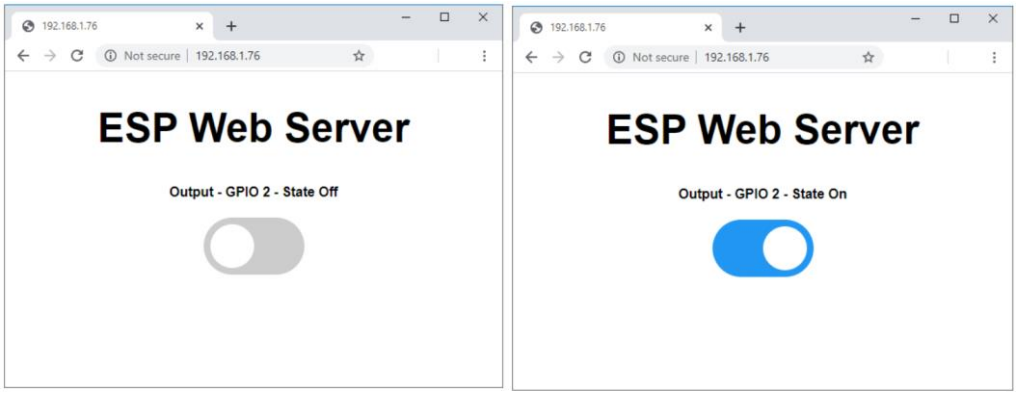
12.3 Cara Kerja :

Untuk kasus ini, lampu LED dengan Esp32 bisa dikendalikan dengan 2 cara, yaitu menggunakan tombol fisik maupun melalui WebServer.

Kalaupun tombol fisik ditekan, lampu otomatis menyala dan mengupdate data ke WebServer menjadikan tampilan button pada WebServer dalam kondisi menyala/ON juga.

Kredensial wifi diperlukan untuk menghubungkan Esp32 dengan WebServer

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



Saat tombol fisik tidak ditekan

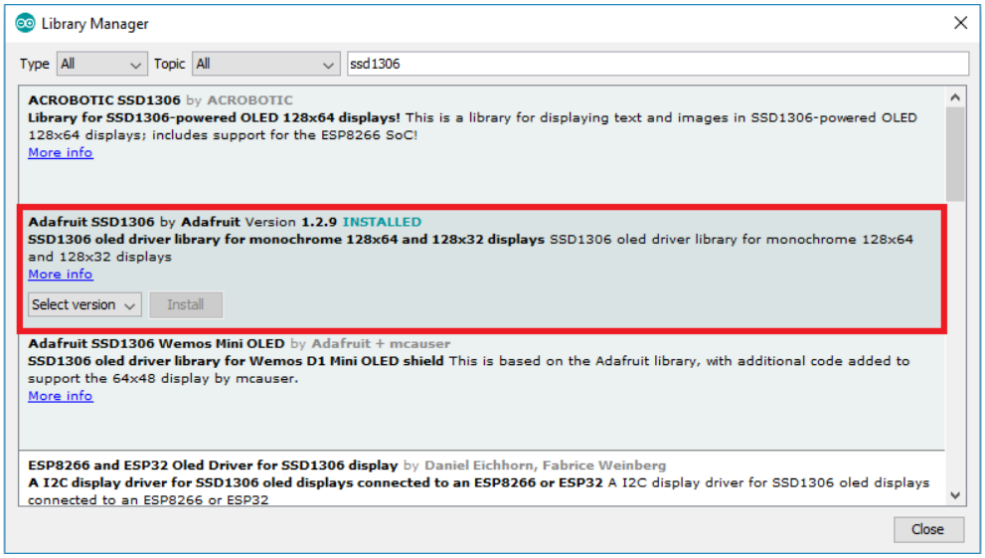
Saat tombol fisik ditekan

Bab 13 – Oled Display

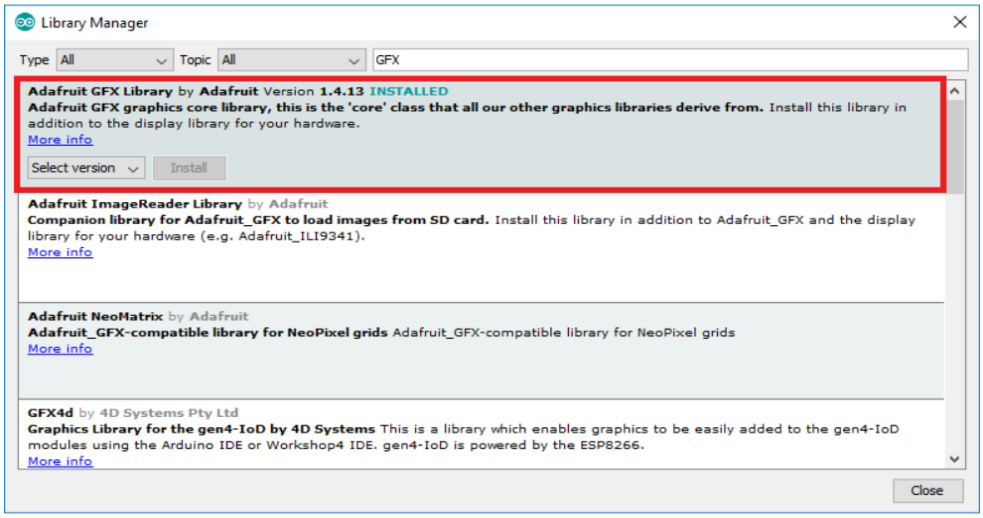
13.1 Kode untuk Arduino IDE

Sebelum pengkodean pastikan OLED Display library terpasang pada Arduino IDE. Caranya ialah :

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.



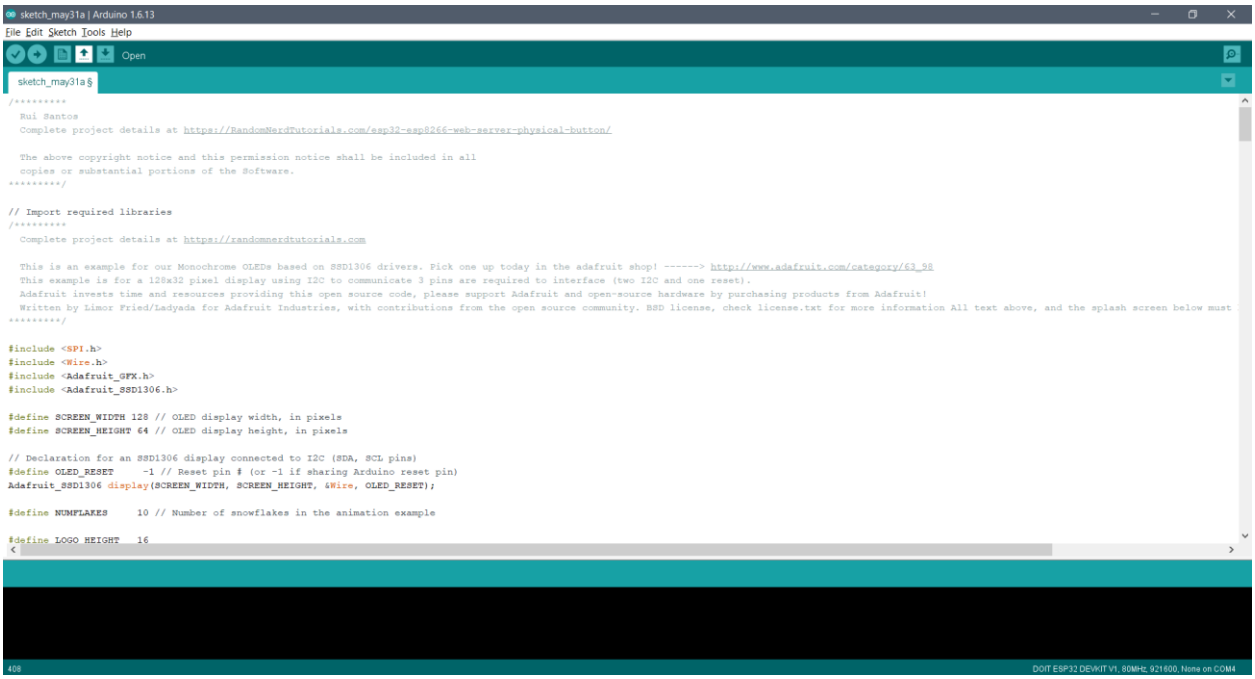
3. Setelah menginstal perpustakaan SSD1306 dari Adafruit, ketik "GFX" di kotak pencarian dan instal perpustakaan.



4. Setelah menginstal perpustakaan, restart Arduino IDE

Adapun kode untuk OLED Display pada Arduino IDE agar terhubung ke Esp32

Kode OLED Display untuk berbagai bentuk gambar




```
/*********
  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);
      }
    }
```

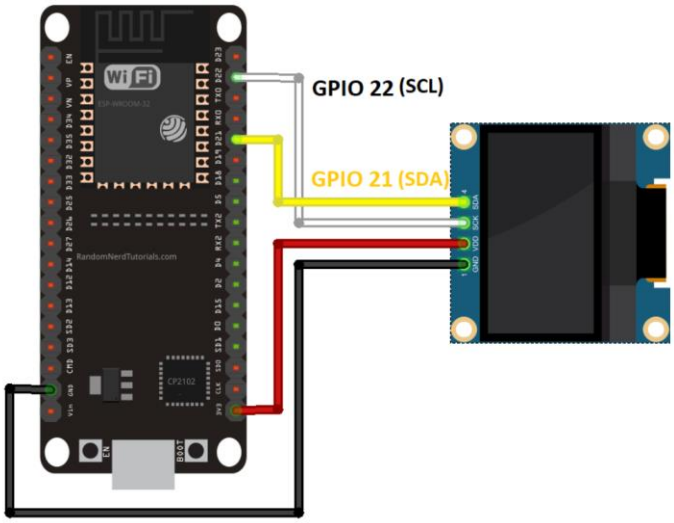
```
  }
}
```

Kode OLED Display untuk menampilkan Text

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

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Fonts/FreeSerif9pt7b.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)) {
    Serial.println("SSD1306 allocation failed");
    for(;;);
  }
  delay(2000);

  display.setFont(&FreeSerif9pt7b);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,20);
  display.println("PLC AsiX :D");
  display.display();
  delay(2000);
```

```
}
void loop() {

}
```

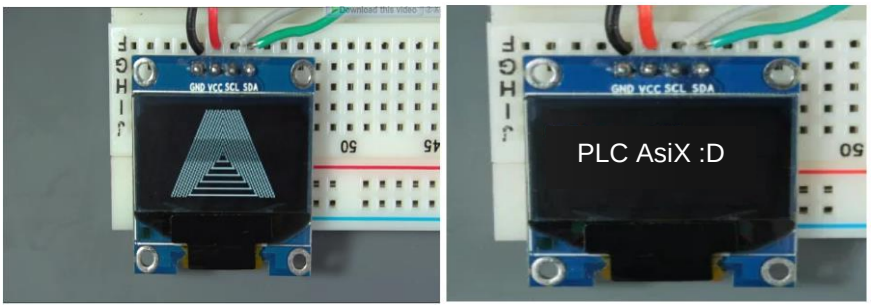
13.2 Wiring dari Arduino IDE ke Esp32



13.3 Cara Kerja :

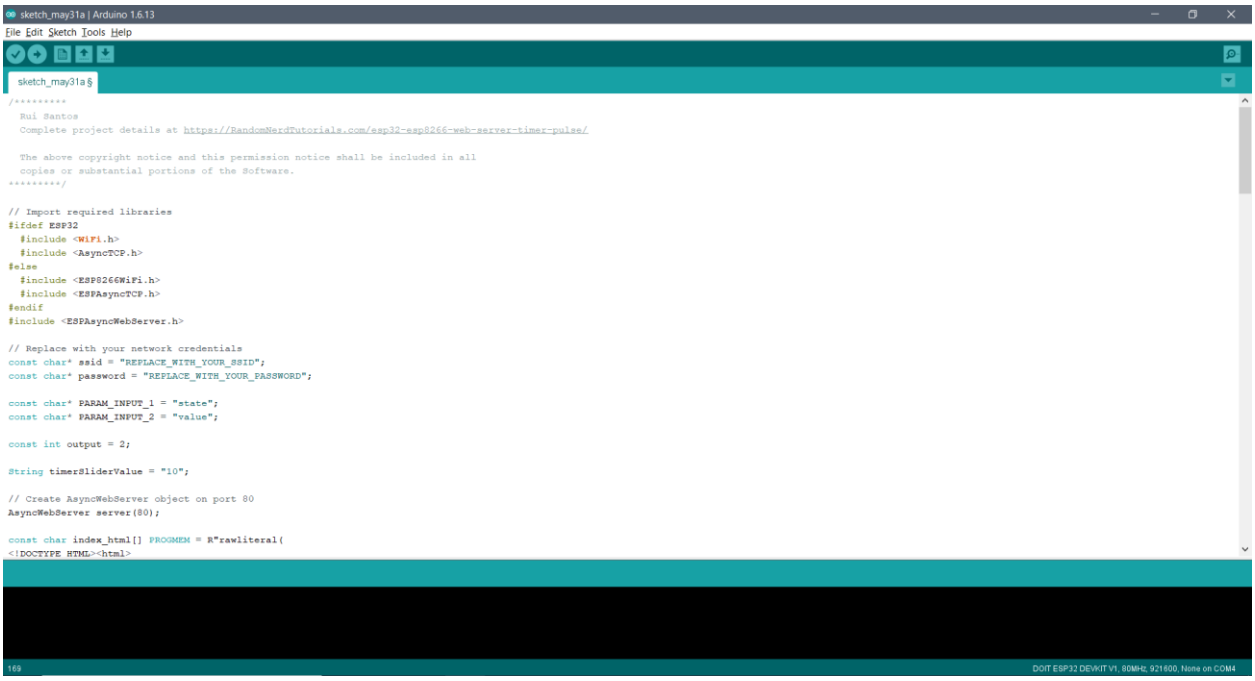
Program yang ada Arduino IDE dapat mengontrol modul OLED Display melalui Esp32 yang dimana terdapat variasi bentuk visual yang akan ditampilkan pada modul OLED Display.

Beberapa bentuk yang dapat ditampilkan pada OLED Display yaitu :



Bab 14 – Esp32 Timer/Pulse WebServer

5.1 Kode untuk Arduino IDE




```
/*********
  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
#ifdef 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}
```

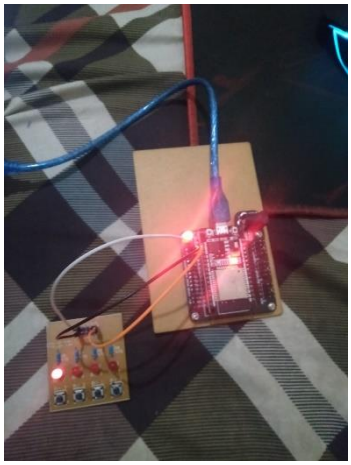
```
    input:checked+.slider {background-color: #2196F3}
    input:checked+.slider:before {-webkit-transform:
translateX(52px); -ms-transform: translateX(52px);
transform: translateX(52px)}
    .slider2 { -webkit-appearance: none; margin: 14px;
width: 300px; height: 20px; background: #ccc;
      outline: 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, [](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;
    // 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() {

}
```

14.2 Wiring dari Arduino IDE ke Esp32



5.3 Cara Kerja :

Pada program Arduino IDE, terdapat konfigurasi untuk Esp32 yang bisa terkoneksi dengan WebServer untuk dapat mengatur timer untuk menyalakan lampu dengan slider. Kita bisa mengatur timer nya sesuai keinginan.