

**LAPORAN PRAKTIKUM ELEKTRONIKA
MESIN LISTRIK DAN TEKNIK KENDALI**



Disusun Oleh:

Akbar Hendrawan Pratama (19/441190/SV/16542)

Dika Puspito (19/441196/SV/16548)

TEKNOLOGI REKAYASA MESIN
DEPARTEMEN TEKNIK MESIN SEKOLAH VOKASI
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2022

BAB 1. SPESIFIKASI

ESP32 merupakan mikrokontroler yang dikenalkan oleh *Espressif System* merupakan penerus dari mikrokontroler ESP8266. Pada mikrokontroler saat ini sudah tersedia teknologi yang dapat mencakup *wifi* dalam *chip* sehingga sangat mendukung untuk membuat sistem aplikasi pada internet. Pada ESP32 terdapat pin yang dapat digunakan sebagai input atau output untuk menyalakan LCD, lampu, bahkan untuk menggerakan motor DC yang tersedia dengan melakukan pemrograman pada sistem aplikasi Arduino IDE atau dapat dilakukan dengan micropython.

Berdasarkan spesifikasi lebih mendalam mengenai ESP32 sebagai berikut:

Specifications – ESP32 DEVKIT V1 DOIT

Number of cores	2 (dual core)
Wi-Fi	2.4 GHz up to 150 Mbits/s
Bluetooth	BLE (Bluetooth Low Energy) and legacy Bluetooth
Architecture	32 bits
Clock frequency	Up to 240 MHz
RAM	512 KB
Pins	30 or 36 (depends on the model)
Peripherals	Capacitive touch, ADC (analog to digital converter), DAC (digital to analog converter), I2C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I2S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation), and more.

Pada chip ESP32 dapat ditemukan beberapa spesifikasi lainnya yang ditemukan dalam sumber lainnya seperti:

- ESP32 merupakan dual core yang dapat diartikan memiliki 2 prosesor.
- Chip pada ESP32 sudah dilengkapi dengan bluetooth dan wifi didalamnya.
- Pada ESP32 sudah dapat dijalankan pada program 32 bit.
- Frekuensi clock pada ESP32 dapat menjangkau hingga 240MHz serta memiliki 512kB RAM.
- Pada satu bagian board pada ESP32 terdapat 30 hingga 36 pin, yaitu 15 di setiap barisnya.
- ESP32 dapat menjangkau banyak variasi yang tersedia seperti : capacitive touch, ADCs, DACs, UART, SPI, I2C dan lainnya.
- Dapat muncul dengan efek dari sensor built-in dan sensor temperature built-in.

Pada wifi key feature:

- 802.11 b/g/n
- 802.11 n (2.4 GHz), up to 150 Mbps

- WMM
- TX/RX A-MPDU, RX A-MSDU
- Immediate Block ACK
- Defragmentation
- Automatic beacon monitoring (hardware TSF)
- 4 x virtual Wi-Fi interface
- Simultaneous support pada infrastructure Station, SoftAP, serta Promiscuous modes note pada ESP32 mode stasiun, performing a scan, channel pada AP nantinya akan berubah.
- Antenna diversity

Pada bluetooth key feature :

- Compliant dengan Bluetooth v4.2 BR/EDR dan spesifikasi bluetooth LE
- Class-1, class-2 and class-3 transmitter without external power amplifier
- Enhanced Power Control
- +9 dBm transmitting power
- NZIF receiver with -94 dBm Bluetooth LE sensitivity
- Adaptive Frequency Hopping (AFH)
- Standard HCI based on SDIO/SPI/UART
- High-speed UART HCI, up to 4 Mbps
- Bluetooth 4.2 BR/EDR Bluetooth LE dual mode controller
- Synchronous Connection-Oriented/Extended (SCO/eSCO)
- CVSD and SBC for audio codec
- Bluetooth Piconet and Scatternet
- Multi-connections in Classic Bluetooth and Bluetooth LE
- Simultaneous advertising and scanning

Pada MCU and Advanced feature :

A. CPU and Memory

- Xtensa® single-/dual-core 32-bit LX6 microprocessor(s)
- CoreMark® score:
- 1 core at 240 MHz: 504.85 CoreMark; 2.10 CoreMark/MHz
 - 2 cores at 240 MHz: 994.26 CoreMark; 4.14 CoreMark/MHz
 - 448 KB ROM
 - 520 KB SRAM
 - 16 KB SRAM in RTC
 - QSPI supports multiple flash/SRAM chips

B. Clocks and Timers

- Internal 8 MHz oscillator with calibration
- Internal RC oscillator with calibration

- External 2 MHz ~ 60 MHz crystal oscillator (40 MHz only for Wi-Fi/Bluetooth functionality)
- External 32 kHz crystal oscillator for RTC with calibration
- Two timer groups, including $2 \times$ 64-bit timers and $1 \times$ main watchdog in each group
 - One RTC timer
 - RTC watchdog

C. Advanced Peripheral Interfaces

- $34 \times$ programmable GPIOs
- 12-bit SAR ADC up to 18 channels
- $2 \times$ 8-bit DAC
- $10 \times$ touch sensors
- $4 \times$ SPI
- $2 \times$ I₂S
- $2 \times$ I₂C
- $3 \times$ UART
- 1 host (SD/eMMC/SDIO)
- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- TWAI®, compatible with ISO 11898-1 (CAN Specification 2.0)
- RMT (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor

D. Security

- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration
- AES
- Hash (SHA-2)
- RSA
- ECC

E. Applications (A Non-exhaustive List)

- Generic Low-power IoT Sensor Hub
- Generic Low-power IoT Data Loggers
- Cameras for Video Streaming
- Over-the-top (OTT) Devices
- Speech Recognition

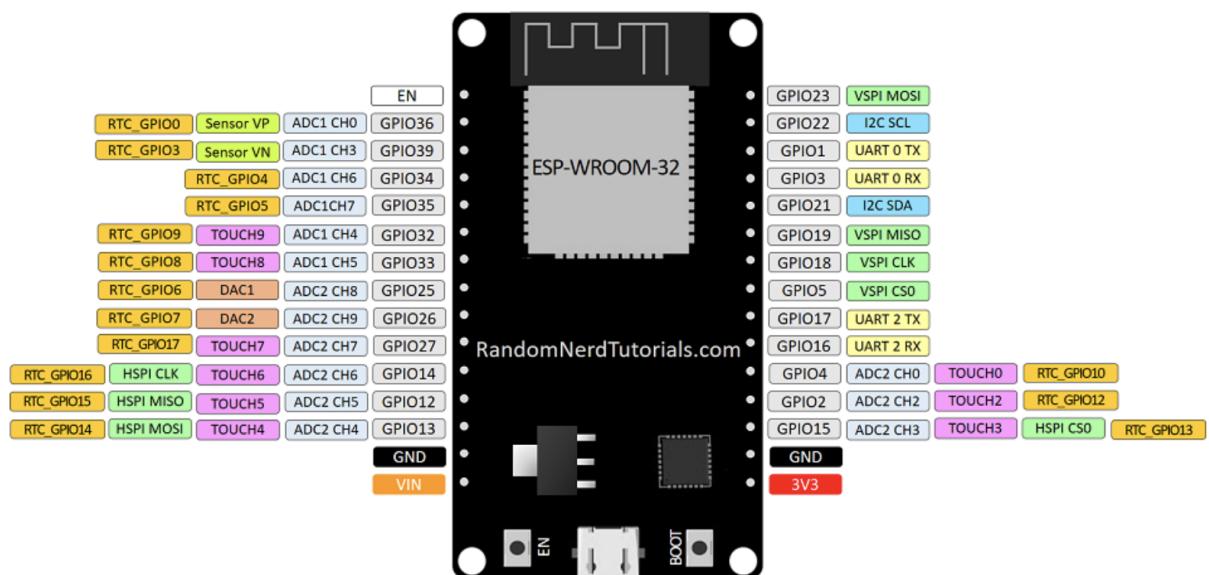
- Image Recognition
- Mesh Network
- Home Automation
- Light control
- Smart plugs
- Smart door locks
- Smart Building
- Smart lighting
- Energy monitoring
- Industrial Automation
- Industrial wireless control
- Industrial robotics
- Smart Agriculture
- Smart greenhouses
- Smart irrigation
- Agriculture robotics
- Audio Applications
- Internet music players
- Live streaming devices

BAB 2. KAKI-KAKI ATAU PINOUT

ESP32 merupakan GPIOs yang memiliki fungsi lebih beragam serta fungsional jika dibandingkan dengan terdahulunya yaitu ESP826, berdasarkan ESP32 terdapat beragam pin yaitu UART,I2C, dan SPI, pada pinout memungkinkan untuk chip dari ESP32 menggunakan serta digunakan pada beragam future serta beragam fungsi pada pin yang sama

Version with 30 GPIOs

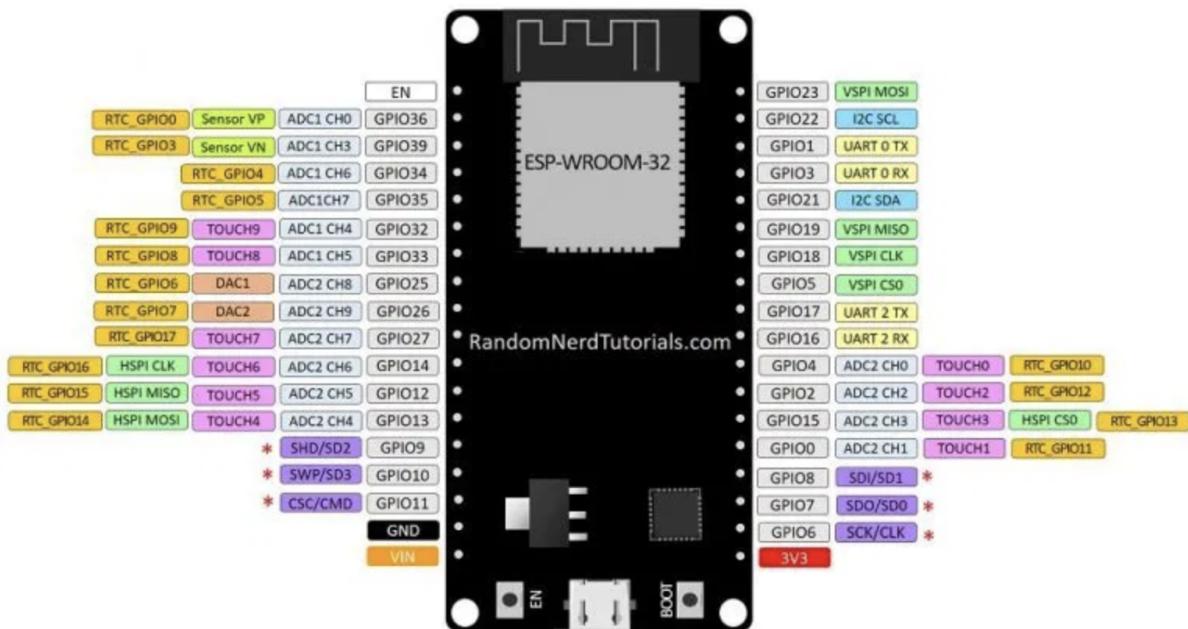
ESP32 DEVKIT V1 – DOIT version with 30 GPIOs



Version with 36 GPIOs

ESP32 DEVKIT V1 – DOIT

version with 36 GPIOs



Dapat diaplikasikan dengan menggunakan bahasa pemrograman, sehingga pinout yang dikolaborasikan verse.

```
1 // ledPin refers to ESP32 GPIO 23
2 const int ledPin = 23;
3 const int ledPin2 = 22;
4 const int ledPin3 = 5;
5 // the setup function runs once when you press reset or power the board
6 void setup() {
7 // initialize digital pin ledPin as an output.
8 pinMode(ledPin, OUTPUT);
9 pinMode(ledPin2, OUTPUT);
10 pinMode(ledPin3, OUTPUT);
11
12 }
13
14 // the loop function runs over and over again forever
15 void loop() {
16 digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
17 digitalWrite(ledPin2, LOW); // turn the LED off by making the voltage LOW
18 digitalWrite(ledPin3, HIGH); // turn the LED on (HIGH is the voltage level)
19 delay(1000); // wait for a second
20 digitalWrite(ledPin2, HIGH); // turn the LED on (HIGH is the voltage level)
21 digitalWrite(ledPin, LOW); // turn the LED off by making the voltage LOW
22 digitalWrite(ledPin3, HIGH); // turn the LED on (HIGH is the voltage level)
23 delay(1000); // wait for a second
24 digitalWrite(ledPin3, HIGH); // turn the LED on (HIGH is the voltage level)
25 digitalWrite(ledPin, LOW); // turn the LED on (HIGH is the voltage level)
26 digitalWrite(ledPin2, LOW); // turn the LED off by making the voltage LOW
27 delay(1000); // wait for a second
28 digitalWrite(ledPin2, HIGH); // turn the LED on (HIGH is the voltage level)
29 digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
30 digitalWrite(ledPin3, LOW); // turn the LED off by making the voltage LOW
31 delay(1000); // wait for a second
32 }
```

Dapat dilakukan penyimpulan bahwa jika berwarna hijau yaitu OK untuk digunakan begitu juga dengan penggunaan warna kuning, namun jika terdeteksi warna kuning harus

mendapat perhatian lebih dikarenakan dapat terdapat kegagalan terutama pada saat boot. Disamping itu jika terdapat warna merah maka sangat untuk tidak direkomendasikan untuk digunakan pada input serta output .

GPIO	Input	Output	Notes
0	pulled up	OK	outputs PWM signal at boot
1	TX pin	OK	debug output at boot
2	OK	OK	connected to on-board LED
3	OK	RX pin	HIGH at boot
4	OK	OK	
5	OK	OK	outputs PWM signal at boot
6	x	x	connected to the integrated SPI flash
7	x	x	connected to the integrated SPI flash
8	x	x	connected to the integrated SPI flash
9	x	x	connected to the integrated SPI flash
10	x	x	connected to the integrated SPI flash
11	x	x	connected to the integrated SPI flash
12	OK	OK	boot fail if pulled high
13	OK	OK	
14	OK	OK	outputs PWM signal at boot
15	OK	OK	outputs PWM signal at boot

16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		input only
35	OK		input only
36	OK		input only
39	OK		input only

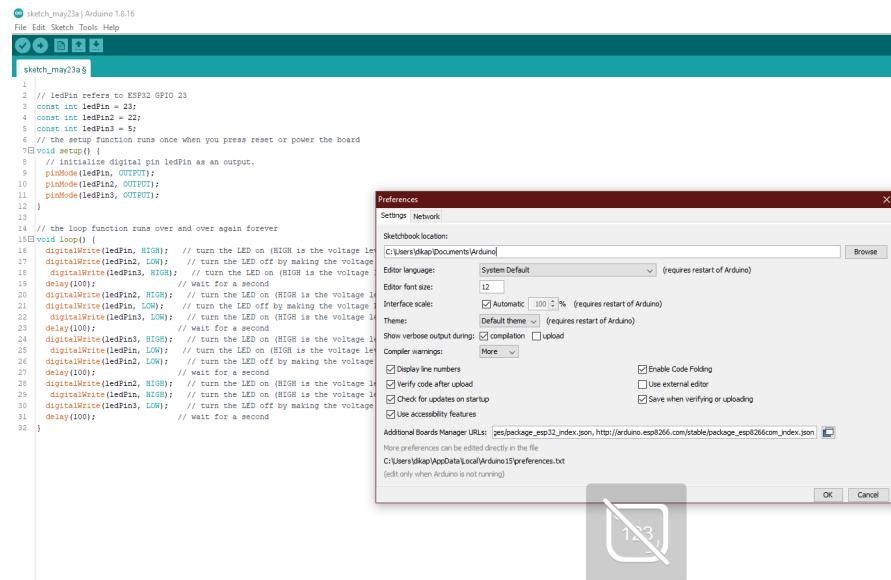
Arus yang dapat ditarik pada GPIO

Arus maksimal yang dapat ditarik oleh periferal di tiap GPIO adalah 12mA, sebagaimana tertulis pada tabel

Parameter	Symbol	Min	Max	Unit
Input low voltage	V_{IL}	-0.3	$0.25 \times V_{IO}$	V
Input high voltage	V_{IH}	$0.75 \times V_{IO}$	3.3	V
Input leakage current	I_{IL}	-	50	nA
Output low voltage	V_{OL}	-	$0.1 \times V_{IO}$	V
Output high voltage	V_{OH}	$0.8 \times V_{IO}$	-	V
Input pin capacitance	C_{pad}	-	2	pF
VDDIO	V_{IO}	1.8	3.3	V
Maximum drive capability	I_{MAX}	-	12	mA
Storage temperature range	T_{STR}	-40	150	°C

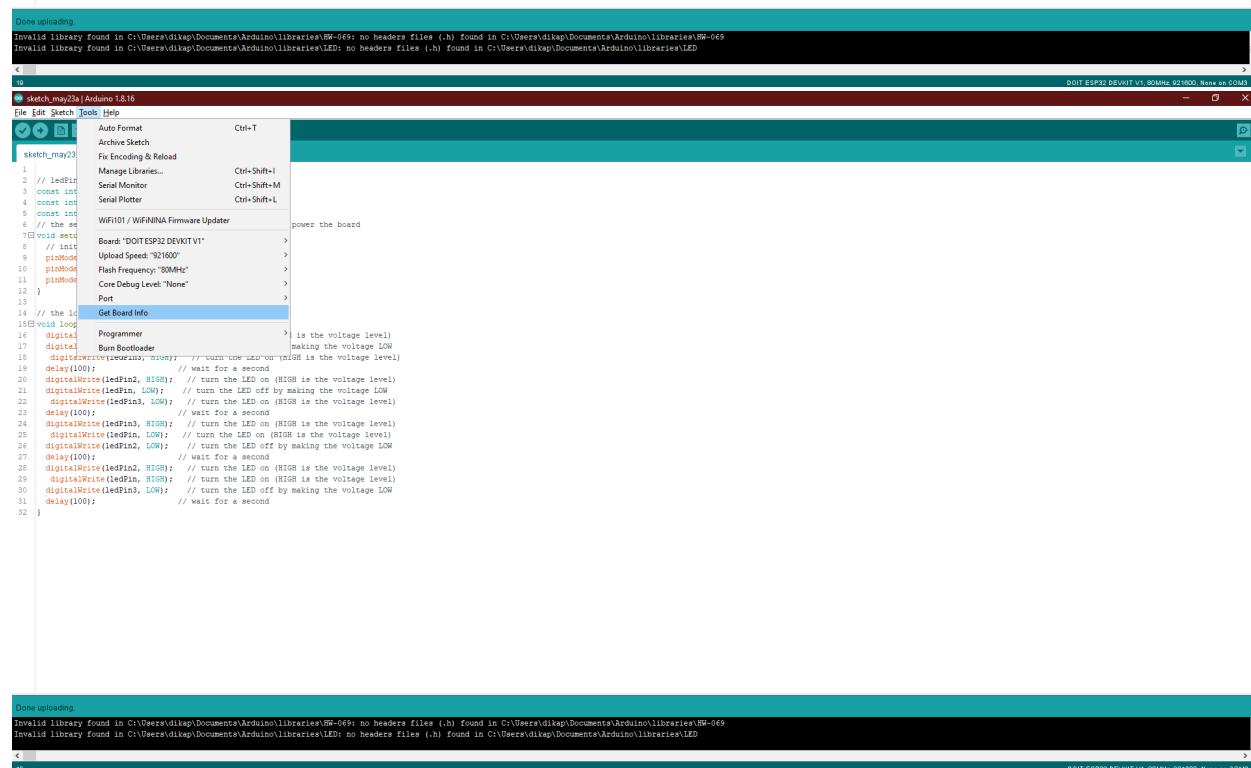
BAB 3. PENGGUNAAN ARDUINO IDE

Penggunaan arduino dengan cara mengaplikasikannya pada chip ESP32 dilakukan dengan cara dibawah ini agar dapat memunculkan library / modul untuk menunjang pengoprasian ESP32.



```
sketch_may23a | Arduino 1.8.16
File Edit Sketch Tools Help
sketch_may23a.g

1 // ledPin refers to ESP32 GPIO 23
2 const int ledPin = 23;
3 const int ledPin2 = 22;
4 const int ledPin3 = 5;
5 const int ledPin4 = 6;
6 // the setup() function runs once when you press reset or power the board
7 void setup()
8 {
9     // initializes digital pin ledPin as an output.
10    pinMode(ledPin, OUTPUT);
11    pinMode(ledPin2, OUTPUT);
12    pinMode(ledPin3, OUTPUT);
13 }
14 // the loop function runs over and over again forever
15 void loop()
16 {
17     digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
18     digitalWrite(ledPin2, LOW); // turn the LED off by making the voltage
19     delay(1000); // wait for a second
20     digitalWrite(ledPin2, HIGH); // turn the LED on (HIGH is the voltage level)
21     digitalWrite(ledPin3, LOW); // turn the LED on (HIGH is the voltage level)
22     delay(1000); // wait for a second
23     digitalWrite(ledPin3, HIGH); // turn the LED on (HIGH is the voltage level)
24     digitalWrite(ledPin4, LOW); // turn the LED off by making the voltage
25     delay(1000); // wait for a second
26     digitalWrite(ledPin4, HIGH); // turn the LED on (HIGH is the voltage level)
27     delay(1000); // wait for a second
28     digitalWrite(ledPin2, HIGH); // turn the LED on (HIGH is the voltage level)
29     digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
30     digitalWrite(ledPin, LOW); // turn the LED off by making the voltage
31     delay(1000); // wait for a second
32 }
```



```
Done uploading.
Invalid library found in C:\Users\dikap\Documents\Arduino\libraries\HW-069: no headers files (.h) found in C:\Users\dikap\Documents\Arduino\libraries\HW-069
Invalid library found in C:\Users\dikap\Documents\Arduino\libraries\LED: no headers files (.h) found in C:\Users\dikap\Documents\Arduino\libraries\LED
< > 19
sketch_may23a | Arduino 1.8.16
File Edit Sketch Tools Help
Auto Format Ctrl-T
Archive Sketch
Fix Encoding & Reload
Manage Libraries...
Serial Monitor Ctrl+Shift-M
Serial Plotter Ctrl+Shift-P
WiFi101 / WIFININA Firmware Updater
Board: "DOIT ESP32 DEVKIT V1"
Upload Speed: 921600
Flash Frequency: 80MHz
Core Debug Level: None
Port: >
Get Board Info
```

```
power the board
is the voltage level
making the voltage LOW
is the voltage level
wait for a second
turn the LED on (HIGH is the voltage level)
turn the LED off by making the voltage LOW
turn the LED on (HIGH is the voltage level)
delay(1000)
turn the LED on (HIGH is the voltage level)
turn the LED off by making the voltage LOW
turn the LED off by making the voltage LOW
delay(1000)
turn the LED on (HIGH is the voltage level)
turn the LED on (HIGH is the voltage level)
turn the LED off by making the voltage LOW
delay(1000)
turn the LED on (HIGH is the voltage level)
turn the LED on (HIGH is the voltage level)
turn the LED off by making the voltage LOW
delay(1000)
turn the LED on (HIGH is the voltage level)
turn the LED on (HIGH is the voltage level)
turn the LED off by making the voltage LOW
delay(1000)
```

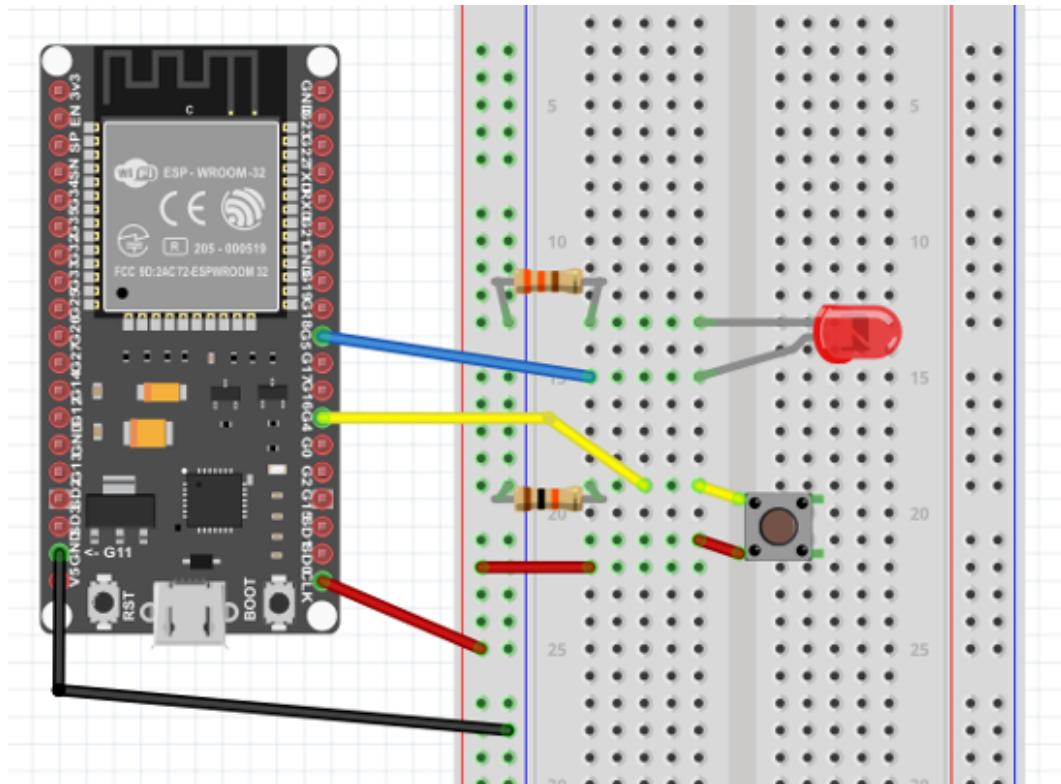
```
Done uploading.
Invalid library found in C:\Users\dikap\Documents\Arduino\libraries\HW-069: no headers files (.h) found in C:\Users\dikap\Documents\Arduino\libraries\HW-069
Invalid library found in C:\Users\dikap\Documents\Arduino\libraries\LED: no headers files (.h) found in C:\Users\dikap\Documents\Arduino\libraries\LED
< > 19
DOIT ESP32 DEVKIT V1, 80MHz, 921600, None on COM3
```

BAB 4. INPUT ATAU OUTPUT

Pada chip ESP32 memiliki pin Input dan Output sebagai antarmuka untuk berkomunikasi dengan user lainnya. Pin dinamani dengan General Purpose Input Output (GPIO).

Input Output pada ESP32, terdiri dari:

- 18 buah kanal Analog-to-Digital Converter (ADC)
- 3 buah antarmuka Serial-Parallel Interface (SPI)
- 3 buah antarmuka UART
- 2 buah antarmuka I2C
- 16 kanal output PWM
- 2 Digital-to-Analog Converter (DAC)
- 2 buah antarmuka I2S
- 10 buah GPIO Capacitive Sensing



Berikut merupakan code yang diaplikasikan pada pemrograman.



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_may23b | Arduino 1.8.16
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Undo, Redo, Cut, Copy, Paste, Select All, Find, and Open.
- Code Editor:** Displays the C++ code for the sketch_may23b file. The code reads three pushbutton states and prints them to the Serial Monitor, turning an LED on or off based on the first pushbutton's state.

```
// sketch_may23b | Arduino 1.8.16
File Edit Sketch Tools Help
sketch_may23b
1 // Complete Instructions: https://RandomNerdTutorials.com/esp32-digital-inputs-outputs-arduino/
2
3 // set pin numbers
4 const int buttonPin1 = 4; // the number of the pushbutton pin
5 const int ledPin1 = 5; // the number of the LED pin
6
7
8
9 // variable for storing the pushbutton status
10 int buttonState1 = 0;
11
12 void setup() {
13   Serial.begin(115200);
14   // initialize the pushbutton pin as an input
15   pinMode(buttonPin1, INPUT);
16   // initialize the LED pin as an output
17 }
18 void loop() {
19   // read the state of the pushbutton value
20   buttonState1 = digitalRead(buttonPin1);
21   buttonState2 = digitalRead(buttonPin2);
22   buttonState3 = digitalRead(buttonPin3);
23   Serial.println(buttonState1);
24   Serial.println(buttonState2);
25   Serial.println(buttonState3);
26   // check if the pushbutton is pressed.
27   // if it is, the buttonState is HIGH
28   if (buttonState1 == HIGH) {
29     // turn LED on
30     digitalWrite(ledPin1, HIGH);
31
32
33 } else {
34   // turn LED off
35   digitalWrite(ledPin1, LOW);
36 }
37 }
```

BAB 5. PWM

Pada suatu chip yang bermodelkan ESP32 memiliki pengontrol PWM LED dengan 16 saluran independen yang 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. Selanjutnya , melakukan setting pada frekuensi sinyal PWM. Untuk LED, frekuensi 5000 Hz baik-baik saja untuk digunakan.
3. Atur resolusi siklus tugas sinyal: anda memiliki resolusi dari 1 hingga 16 bit. Kami akan menggunakan resolusi 8-bit, yang berarti anda 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:

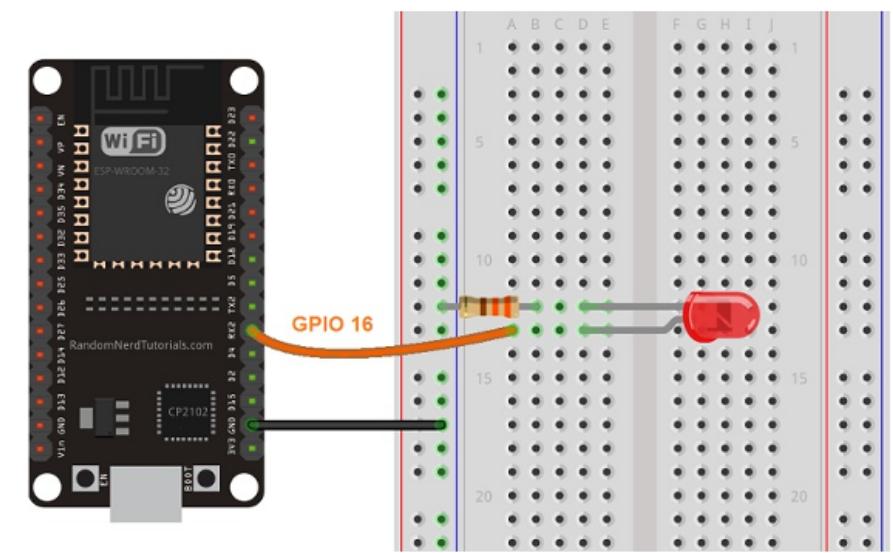
`ledcAttachPin(GPIO, channel)`

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

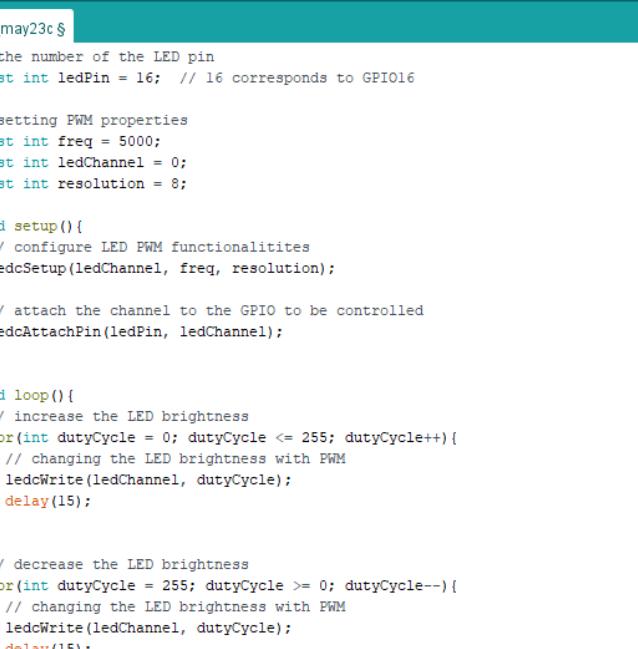
5. Terakhir, untuk mengontrol kecerahan LED menggunakan PWM, digunakan fungsi berikut:

`ledcWrite(channel, dutycycle)`

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



Berikut merupakan code pemrograman yang diaplikasikan pada software arduino.



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_may23c | Arduino 1.8.16
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Open, Print, and Upload.
- Code Editor:** The code for sketch_may23c is displayed. It uses the LEDC library to control an LED via PWM. The code includes setup() and loop() functions to set up the LED pin, configure PWM properties, attach the channel to the GPIO, and then loop to increase and decrease the LED brightness using PWM duty cycles.

```
// the number of the LED pin
const int ledPin = 16; // 16 corresponds to GPIO16

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

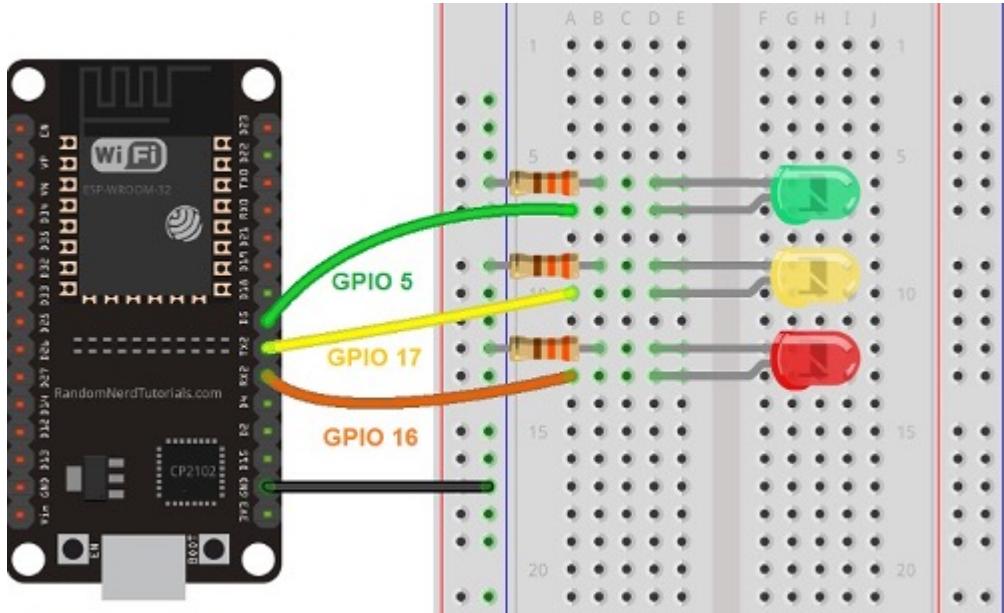
void setup(){
    // configure LED PWM functionalities
    ledcSetup(ledChannel, freq, resolution);

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

Skema



BAB 6. INTERRUPT

Seperti sistem yang menginterupsi suatu hal mirip fungsi detektor gerakan. Hal ini membuat hal-hal yang sekiranya dapat menjadi terkendala akan menjadi suatu hal yang memudahkan, dapat kita implementasikan seperti timer, buzzer, dsb. Terdapat berbagai komponen yang mendukung seperti diantaranya :

A. GPIO Interrupt

GPIO memiliki sebuah serial nomor pada umumnya harus menggunakan digitalPinToInterrupt(GPIO) untuk mengatur GPIO yang seharusnya sebagai pin interupsi. Dengan chip ESP32.

B. 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 mengubah nilai – misalnya dari HIGH ke LOW atau LOW ke HIGH.
- FALLING: ketika pin beralih dari TINGGI ke RENDAH.
- RISING: untuk memicu ketika pin beralih dari LOW ke HIGH.

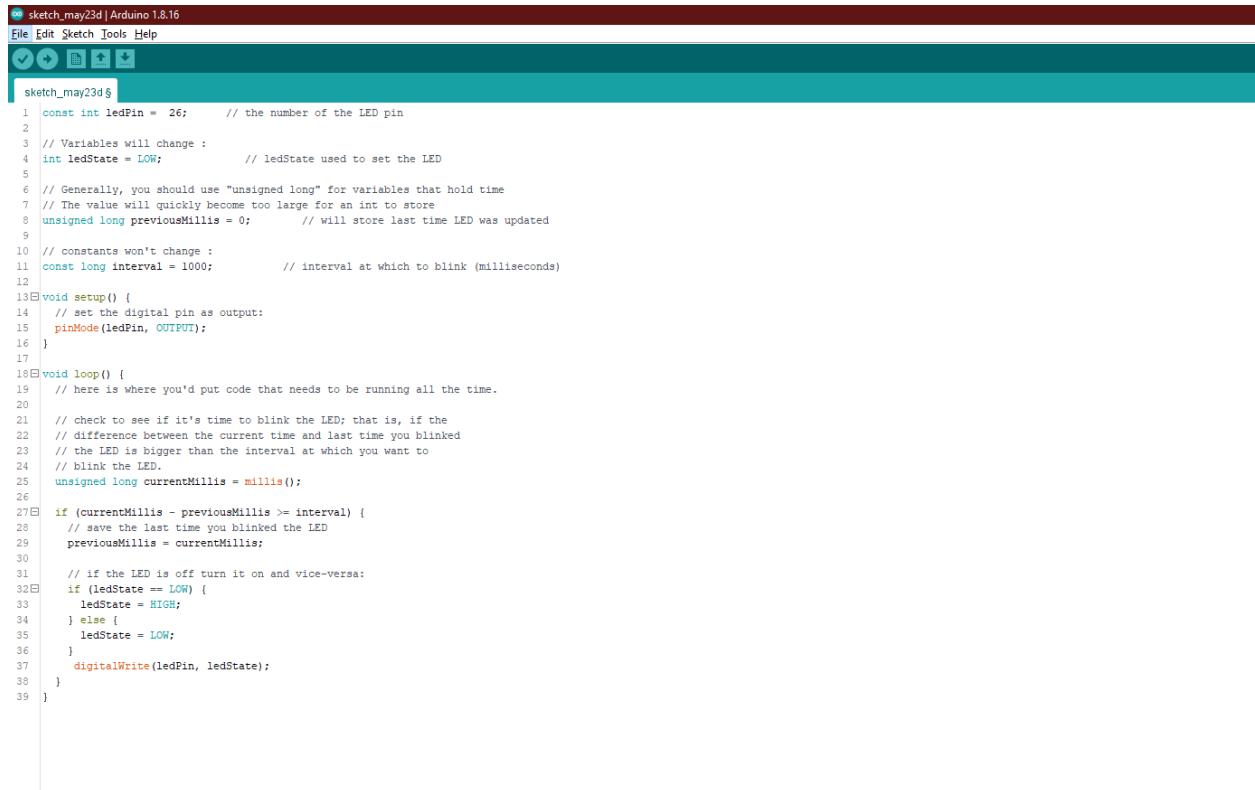
C. Timer

Dalam penjelasan kaliini akan menyertakan timer untuk di bahas dalam bab interrupt. Lampu LED nantinya akan diprogram untuk tetap menyala selama beberapa detik yang telah ditentukan setelah gerakan terdeteksi. Pada kesempatan kaliini akan memanfaatkan untuk menggunakan fungsi delay yang memblokir kode serta tidak memungkinkan untuk melakukan hal lain selama beberapa detik yang ditentukan, diharuskan menggunakan timer.

D. The Delay Function

Fungsi delay yang kerap digunakan membuat program ini kerap dipelajari serta diaplikasikan pada kehidupan. Fungsi yang cukup mudah untuk digunakan dengan cara menerima nomor int tunggal sebagai argumen nantinya terdapat angka yang akan menunjukkan waktu dalam milidetik program harus menunggu sampai pindah ke baris kode berikutnya.

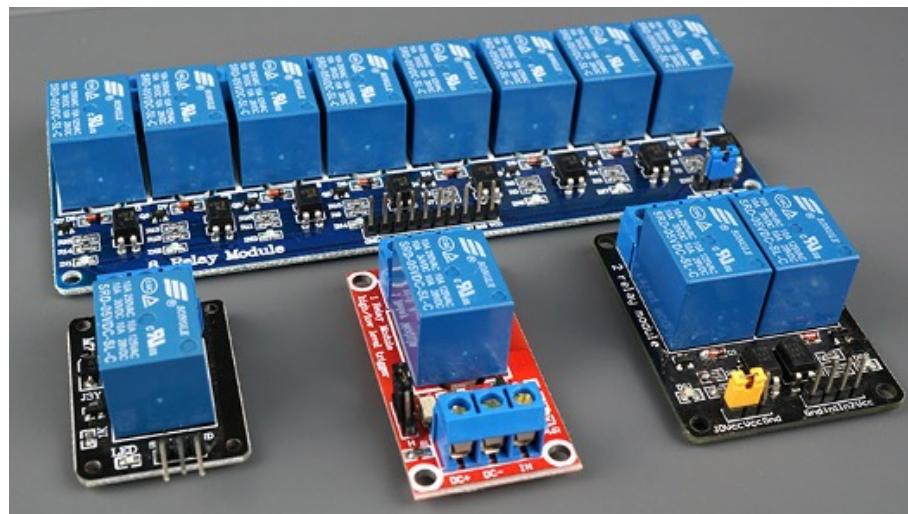
Berikut merupakan code pada pengaplikasian interrupt pada software arduino.



```
sketch_may23d | Arduino 1.8.16
File Edit Sketch Tools Help
sketch_may23d.ino
1 const int ledPin = 26;      // the number of the LED pin
2
3 // Variables will change :
4 int ledState = LOW;        // ledState used to set the LED
5
6 // Generally, you should use "unsigned long" for variables that hold time
7 // The value will quickly become too large for an int to store
8 unsigned long previousMillis = 0;      // will store last time LED was updated
9
10 // constants won't change :
11 const long interval = 1000;           // interval at which to blink (milliseconds)
12
13 void setup() {
14   // set the digital pin as output:
15   pinMode(ledPin, OUTPUT);
16 }
17
18 void loop() {
19   // here is where you'd put code that needs to be running all the time.
20
21   // check to see if it's time to blink the LED; that is, if the
22   // difference between the current time and last time you blinked
23   // the LED is bigger than the interval at which you want to
24   // blink the LED.
25   unsigned long currentMillis = millis();
26
27   if (currentMillis - previousMillis >= interval) {
28     // save the last time you blinked the LED
29     previousMillis = currentMillis;
30
31     // if the LED is off turn it on and vice-versa:
32     if (ledState == LOW) {
33       ledState = HIGH;
34     } else {
35       ledState = LOW;
36     }
37     digitalWrite(ledPin, ledState);
38   }
39 }
```

BAB 7. RELAY

Relay adalah output yang dapat digunakan sebagai switch atau saklar untuk perangkat lain. Relay dikontrol dengan tegangan dari pin Arduino sehingga dapat melakukan switch. Terdapat 3 koneksi utama yaitu COM untuk input dari perangkat lain. NC(Normally Close) pada keadaan biasa com akan terhubung ke pin NC. NO(Normally Open) pada keadaan biasa tidak terhubung, namun saat relay mendapat tegangan dari Arduino maka COM akan berpindah dari NC dan terhubung dengan NO.



Terdapat modul relay yang elektromagnetnya dapat ditenagai oleh 5V dan dengan 3.3V. Dengan asumsi dapat digunakan pada chip ESP32 sehingga dapat menggunakan pin VIN 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.

Relay pinout yaitu ditujukan untuk demonstrasi, sehingga dapat dilihat pada pinout modul relai 2 saluran. Menggunakan modul relai dengan jumlah saluran yang berbeda. Di sisi kiri, terdapat dua set tiga soket untuk menghubungkan tegangan tinggi, dan pin di sisi kanan yang memiliki tegangan rendang terhubung ke GPIO ESP32.



Mains Voltages Connections yaitu relay yang ditunjukkan pada gambar diatas yang memiliki dua konektor, masing-masing dengan tiga soket: umum (COM), Biasanya Tertutup (NC), dan Biasanya Terbuka (NO).

- COM: hubungkan arus yang ingin Anda kendalikan (tegangan listrik).
- NC (Normally Closed): konfigurasi yang biasanya tertutup digunakan bila Anda ingin relai ditutup secara default. NC adalah pin COM yang terhubung, artinya arus mengalir kecuali jika Anda mengirim sinyal dari ESP32 ke modul relai 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 Anda mengirim sinyal dari ESP32 untuk menutup sirkuit.

Control Pins merupakan, sisi tegangan rendah memiliki satu set empat pin dan satu set tiga pin. Set pertama terdiri dari VCC dan GND untuk menyalaikan modul, dan input 1 (IN1) dan input 2 (IN2) untuk mengontrol relai bawah dan atas, masing-masing. Jika modul relai Anda hanya memiliki satu saluran, Anda hanya akan memiliki satu pin IN. Jika Anda memiliki empat saluran, Anda akan memiliki empat pin IN, dan seterusnya. Sinyal yang Anda kirim ke pin IN, menentukan apakah relai aktif atau tidak. Relai dipicu ketika input berjalan di bawah sekitar 2V. Ini berarti Anda akan memiliki skenario berikut:

- Konfigurasi Biasanya Tertutup (NC):
 - Sinyal TINGGI – arus mengalir
 - Sinyal RENDAH – arus tidak mengalir

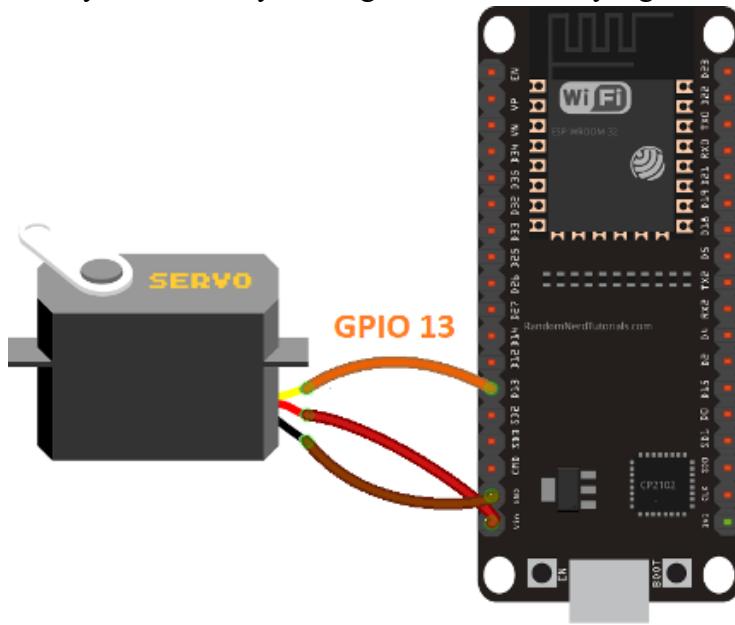
- Konfigurasi Biasanya Terbuka (TIDAK):
 - Sinyal TINGGI – arus tidak mengalir
 - Sinyal RENDAH – arus mengalir

Nantinya harus menggunakan konfigurasi yang biasanya tertutup ketika arus harus mengalir sebagian besar waktu, dan Anda hanya ingin menghentikannya sesekali. Gunakan konfigurasi yang biasanya terbuka ketika Anda ingin arus mengalir sesekali (misalnya, menyalakan lampu sesekali).

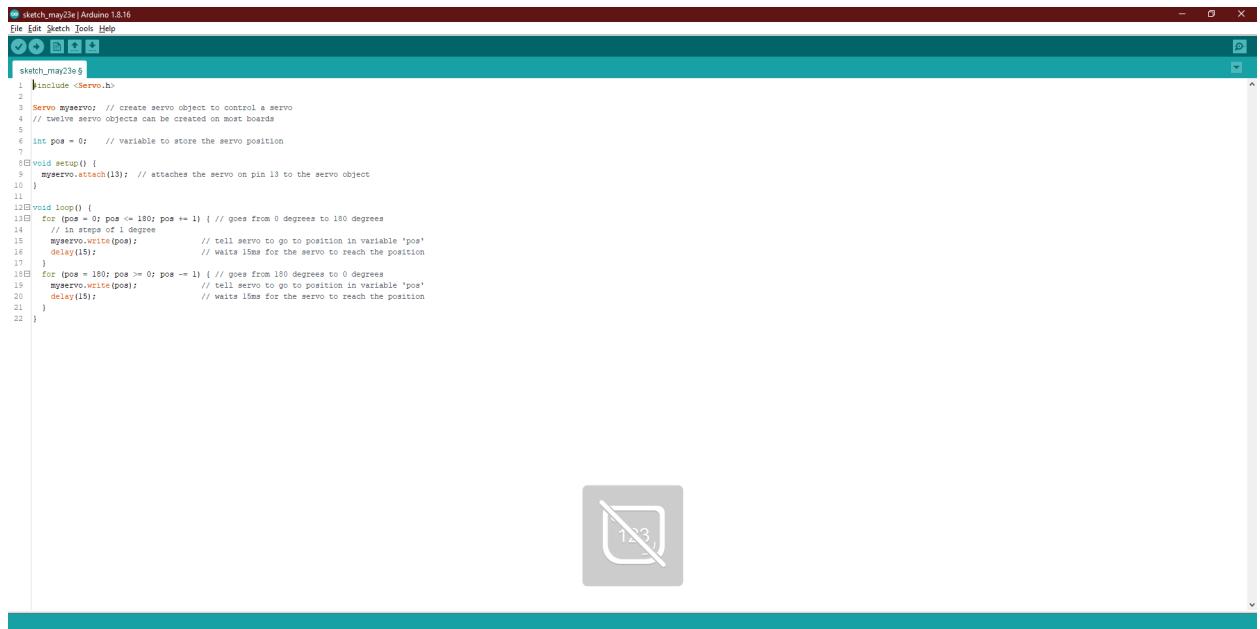
BAB 8. SERVO

Servo adalah salah satu motor DC yang dimana memiliki kekurangan dan kelebihan dibanding motor DC yang lain. Untuk kekurangan motor dc servo yaitu hanya bisa bergerak 0° - 180° yang dimana hal ini sangat minim untuk kebutuhan yang relatif luas. Untuk keuntungan motor servo sendiri yaitu dia hanya menggunakan 3 point input yang dimana berisi, ground, power, dan input setting ke arduino/esp32.

Dalam suatu kasus, akan menghubungkan kabel sinyal ke GPIO 13. Sehingga, dapat mengikuti diagram skema berikutnya untuk menyambungkan motor servo yang akan digunakan.



Untuk program dasar motor servo seperti yang dituangkan pada program arduino berikut.



The image shows a screenshot of the Arduino IDE version 1.8.16. The window title is "sketch_may29e | Arduino 1.8.16". The code editor contains the following C++ code:

```
sketch_may29e.ino
1 #include <Servo.h>
2
3 Servo myservo; // create servo object to control a servo
4 // twelve servo objects can be created on most boards
5
6 int pos = 0; // variable to store the servo position
7
8 void setup() {
9   myservo.attach(13); // attaches the servo on pin 13 to the servo object
10 }
11
12 void loop() {
13   for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
14     // in steps of 1 degree
15     myservo.write(pos); // tell servo to go to position in variable 'pos'
16     delay(15); // waits 15ms for the servo to reach the position
17   }
18   for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
19     myservo.write(pos); // tell servo to go to position in variable 'pos'
20     delay(15); // waits 15ms for the servo to reach the position
21 }
22 }
```

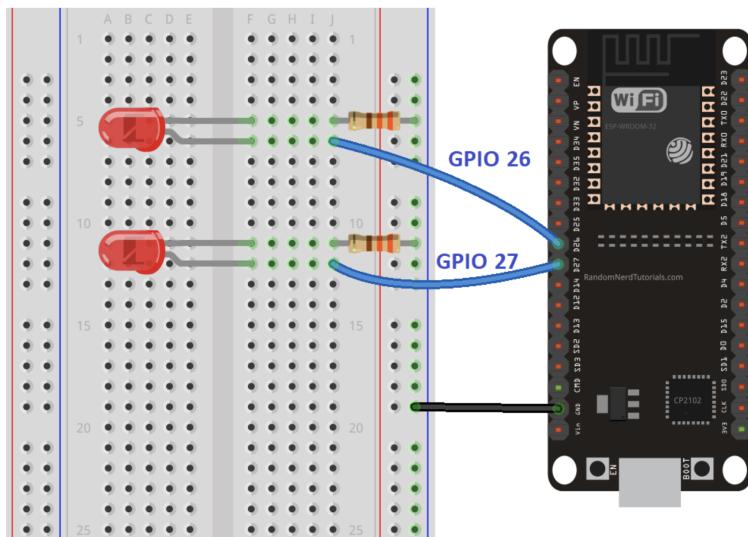
In the center of the screen, there is a small gray square icon containing a white icon of a servo motor with the number "123" below it, indicating the current sketch number.

BAB 9. OUTPUT WEB SERVER

Pada output web server ditujukan untuk menciptakan server secara fungsional dengan menggunakan ESP32 dengan mengontrol output yaitu sebuah lampu LED menggunakan sistem pemrograman arduino IDE. Dengan webserver sendiri yang memiliki sifat mobile responsive serta dapat diakses pada perangkat apapun yang terhubung dengan jaringan wifi atau jaringan lokal yang pada browser.

Dapat dijelaskan secara ringkas tahapannya seperti berikut:

- Pada web server yang dirancang dapat digunakan untuk melakukan kontrol pada lampu LED yang terkoneksi dengan ESP32 GPIO26 serta GPIO27.
- ESP32 dapat diakses dengan melakukan pencarian ESP32 pada IP address pada browser di local network yang sama.
- Dengan melakukan klik pada button yang dirancang pada web server, nantinya dapat dengan mudah untuk mengganti status output setiap lampu LED yang sudah di setting pada program.



```

sketch_may23e|Arduino 1.8.16
File Edit Sketch Tools Help
sketch_may23e
1 #include <WiFi.h>
2
3 // Replace with your network credentials
4 const char* ssid = "REPLACE_WITH_YOUR_SSID";
5 const char* password = "REPLACE_WITH_YOUR_PASSWORD";
6
7 // Set web server port number to 80
8 WiFiServer server(80);
9
10 // Variable to store the HTTP request
11 String header;
12
13 // Auxiliar variables to store the current output state
14 String output26State = "off";
15 String output27State = "off";
16
17 // Assign output variables to GPIO pins
18 const int output26 = 26;
19 const int output27 = 27;
20
21 // Current time
22 unsigned long currentTime = millis();
23 // Previous time
24 unsigned long previousTime = 0;
25 // Define timeout time in milliseconds (example: 2000ms = 2s)
26 const long timeoutTime = 2000;
27
28 void setup() {
29   Serial.begin(115200);
30   // Initialize the output variables as outputs
31   pinMode(output26, OUTPUT);
32   pinMode(output27, OUTPUT);
33   // Set outputs to LOW
34   digitalWrite(output26, LOW);
35   digitalWrite(output27, LOW);
36
37   // Connect to Wi-Fi network with SSID and password
38   Serial.print("Connecting to ");
39   Serial.println(ssid);
40   WiFi.begin(ssid, password);
41   while (WiFi.status() != WL_CONNECTED) {
42     delay(500);
43     Serial.print(".");
44   }
45   // Print local IP address and start web server
46   Serial.println("");
47   Serial.println("WiFi connected.");
48   Serial.println("IP address: ");
49   Serial.println(WiFi.localIP());
50
51 }
52
53 void loop()
54 {
55   WiFiClient client = server.available(); // Listen for incoming clients
56
57   if (client) { // If a new client connects,
58     currentTime = millis();
59     previousTime = currentTime;
60     Serial.println("Client connected.");
61     String currentLine = "";
62     while (client.connected() && currentTime - previousTime < timeoutTime) { // loop while the client's connected
63       currentTime = millis();
64       if (client.available()) { // if there's bytes to read from the client,
65         char byte = client.read(); // read a byte, then
66         Serial.write(byte); // print it out the serial monitor
67         header += byte;
68         if (byte == '\n') { // if the byte is a newline character
69           // if previous byte was a carriage return, you got two newline characters in a row.
70           // that's the end of the client's HTTP request, so send a response:
71           if (currentLine.length() == 0) {
72             // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
73             client.println("HTTP/1.1 200 OK");
74             client.println("Content-type:text/html");
75             client.println("Connection: close");
76             client.println();
77
78             // turns the GPIOs on and off
79             if (header.indexOf("GET /26/on") > 0) {
80               Serial.println("GPIO 26 on");
81               output26State = "on";
82               digitalWrite(output26, HIGH);
83             } else if (header.indexOf("GET /26/off") > 0) {
84               Serial.println("GPIO 26 off");
85               output26State = "off";
86               digitalWrite(output26, LOW);
87             } else if (header.indexOf("GET /27/on") > 0) {
88               Serial.println("GPIO 27 on");
89               output27State = "on";
90               digitalWrite(output27, HIGH);
91             } else if (header.indexOf("GET /27/off") > 0) {
92               Serial.println("GPIO 27 off");
93             }
94           }
95         }
96       }
97     }
98   }
99 }

```

```

sketch_may23e | Arduino 1.8.16
File Edit Sketch Tools Help
sketch_may23e
90     digitalWrite(output27, HIGH);
91 } else if (header.indexOf("GET /27/of") >= 0) {
92     Serial.println("GET 27/of");
93     output27State = "off";
94     digitalWrite(output27, LOW);
95 }
96
97 // Display the HTML web page
98 client.println("<!DOCTYPE html>");
99 client.println("<head><meta name="viewport" content="width=device-width, initial-scale=1">");
100 client.println("<style>body { background-color: #00FFFF; margin: 0px auto; text-align: center; }");
101 client.println("button { border: 1px solid black; border-radius: 5px; color: white; padding: 16px 40px; }");
102 client.println("button2 { background-color: #00FFFF; border: none; font-size: 30px; margin: 10px; cursor: pointer; }");
103 client.println("</style></head>");
104
105 // CSS to style the on/off buttons
106 // Feel free to change the background-color and font-size attributes to fit your preferences
107 client.println("<style>h1 { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center; }");
108 client.println("h1 button { border: 1px solid black; border-radius: 5px; color: white; padding: 16px 40px; }");
109 client.println("h1 button2 { background-color: #00FFFF; border: none; font-size: 30px; margin: 10px; cursor: pointer; }");
110 client.println("</style></head>\"");
111
112 // Web Page Heading
113 client.println("<body><h1>ESP32 Web Server</h1>\"");
114
115 // Display current state, and ON/OFF buttons for GPIO 26
116 client.println("<p>GPIO 26 - State " + output26State + "</p>");
117 // If the output26state is off, it displays the ON button
118 if (output26State == "off") {
119     client.println("<p><a href=\"/26/on\">button class=\"button\">ON</button></a></p>\"");
120 } else {
121     client.println("<p><a href=\"/26/off\">button class=\"button button2\">OFF</button></a></p>\"");
122 }
123
124 // Display current state, and ON/OFF buttons for GPIO 27
125 client.println("<p>GPIO 27 - State " + output27State + "</p>");
126 // If the output27state is off, it displays the ON button
127 if (output27State == "off") {
128     client.println("<p><a href=\"/27/on\">button class=\"button\">ON</button></a></p>\"");
129 } else {
130     client.println("<p><a href=\"/27/off\">button class=\"button button2\">OFF</button></a></p>\"");
131 }
132
133 client.println("</body></html>\"");
134
135 // The HTTP response ends with another blank line
136 client.println();
137 // Break out of the while loop
138 break;
139 } else { // if you got a newline, then clear currentLine
140     currentLine = "";
141 }
142
143 // Clear the header variable
144 header = "";
145 // Close the connection
146 client.stop();
147 Serial.println("Client disconnected.");
148 Serial.println("");
149 }

```

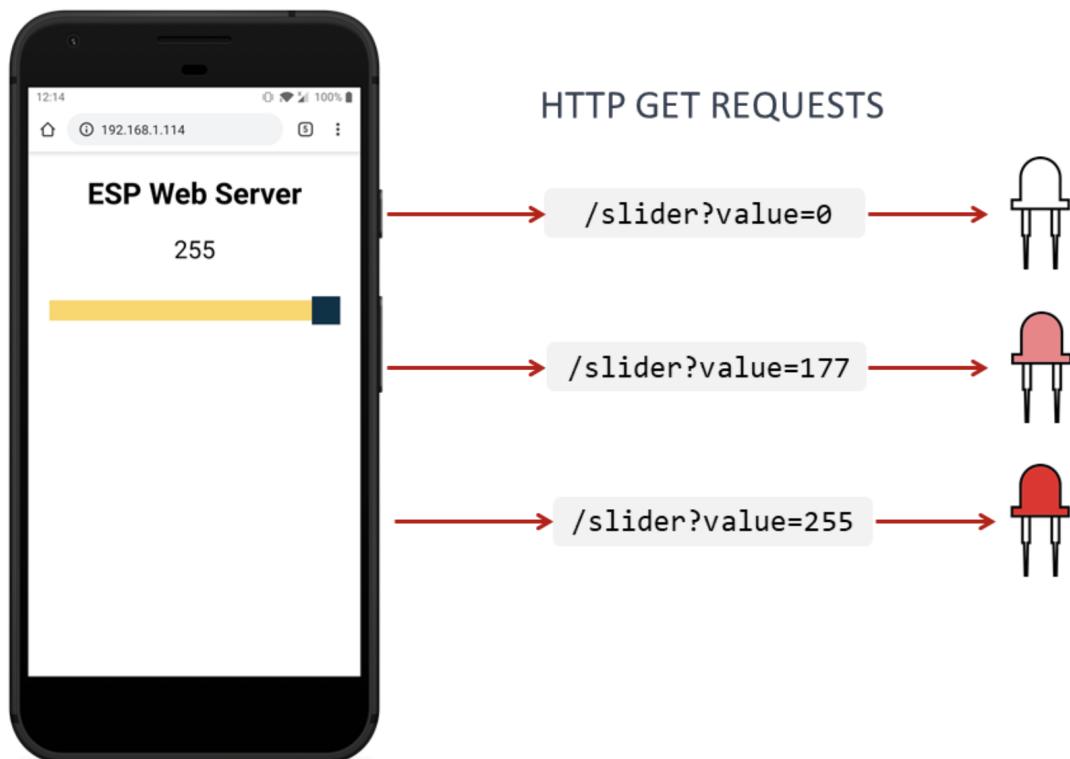
Setelah dilakukan upload pada code pemrograman yang telah dilakukan, nantinya akan ditujukan pada tahapan mencari ESP IP Address yang bertujuan untuk menghubungkan dengan ESP webserver tentunya pada saat melakukan pencarian pada IP Address dilakukan dengan melakukan connect pada wifi atau local network yang sama. Testing web server yang dilakukan pada akhir setelah web server terhubung bertujuan agar dilakukannya monitor pada program yang telah dirancang.

BAB 10. WEB SERVER WITH SLIDER

Webserver with Slider merupakan sistem pemrograman web server ESP32 menggunakan Arduino IDE, bertujuan untuk melakukan kontrol kecerahan pada lampu LED dengan cara menggeser. Variabel yang digunakan merupakan ESP32 yang nantinya terdapat suatu program yang dapat mengontrol sinyal, program tersebut dapat digunakan untuk melakukan kontrol pada sinyal PWM dan sistem lainnya yaitu mengubah kecerahan pada lampu LED. Selain penggunaannya pada mengubah kecerahan lampu LED, dapat juga digunakan untuk mengontrol motor servo dan lainnya.

Dapat dijelaskan secara lebih ringkas tahapannya seperti :

- ESP32 melakukan hosting web server yang menampilkan halaman web dengan penggeser yang nantinya berguna untuk mengatur tingkat kecerahan cahaya.
- Dengan melakukan perpindahan terhadap penggeser, permintaan HTTP diajukan kepada ESP32 dengan menggunakan nilai penggeser yang baru.
- Permintaan pada HTTP nantinya akan berbentuk format sebagai berikut: “GET/slider?value=SLIDERVALUE”, super value merupakan rentang angka antara 0 dan 255.
- Sehingga setelah melakukan permintaan pada HTTP, ESP32 sudah mendapatkan nilai.
- ESP32 menyesuaikan siklus tugas PWM sesuai dengan nilai slider.
- Ini berguna untuk mengontrol kecerahan LED (seperti yang akan kita lakukan dalam contoh ini), motor servo, pengaturan nilai ambang batas, atau aplikasi lain.
- Nantinya dapat digunakan untuk melakukan kontrol kecerahan pada lampu LED, motor servo, pengaturan nilai batas, serta aplikasi lainnya.

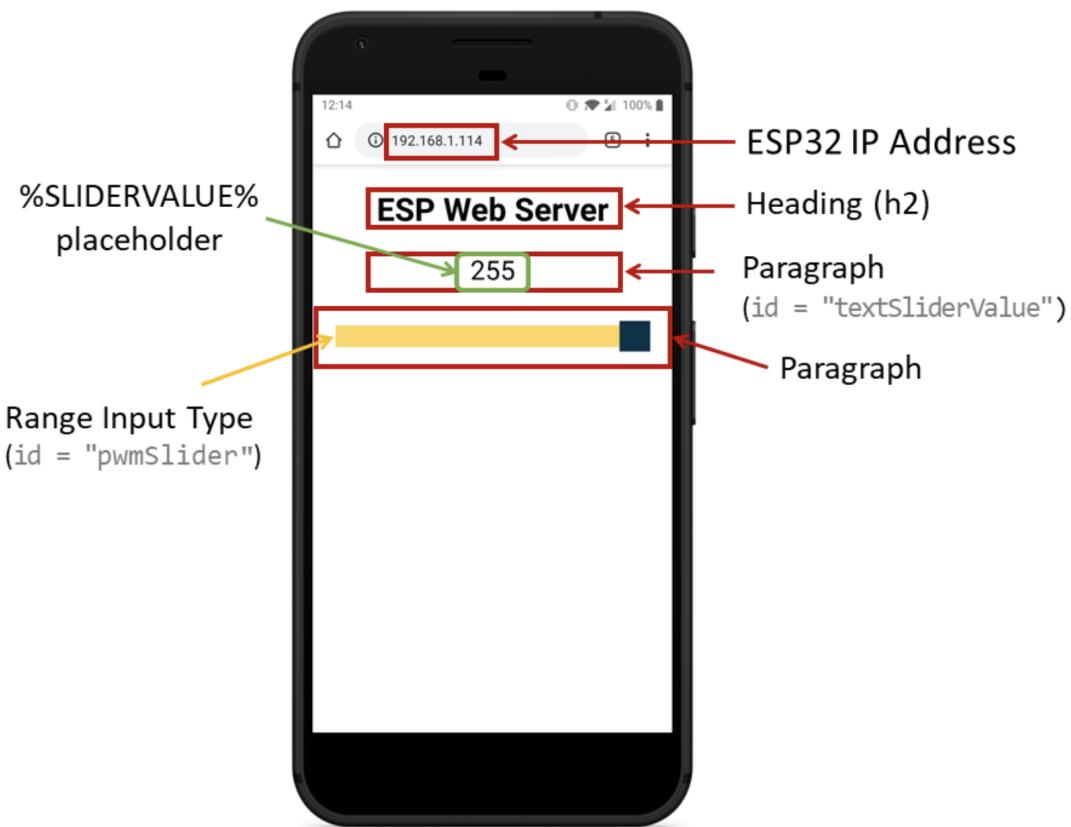


```

sketch_may24b | Arduino 1.8.16
File Edit Sketch Tools Help
sketch_may24b
1 // Import required libraries
2 #include <WiFi.h>
3 #include <AsyncTCP.h>
4 #include <ESPAsyncWebServer.h>
5
6 // Replace with your network credentials
7 const char* ssid = "REPLACE_WITH_YOUR_SSID";
8 const char* password = "REPLACE_WITH_YOUR_PASSWORD";
9
10 const int output = 2;
11
12 String sliderValue = "0";
13
14 // setting PWM properties
15 const int freq = 5000;
16 const int ledChannel = 0;
17 const int resolution = 8;
18
19 const char* PARAM_INPUT = "value";
20
21 // Create AsyncWebServer object on port 80
22 AsyncWebServer server(80);
23
24 const char index_html[] PROGMEM = R"rawliteral(
25 <!DOCTYPE HTML><html>
26 <head>
27   <meta name="viewport" content="width=device-width, initial-scale=1">
28   <title>ESP Web Server</title>
29   <style>
30     h1 {font-family: Arial; display: inline-block; text-align: center; margin: 10px; font-size: 1.5em; border: 1px solid black; padding: 5px; width: fit-content; margin-left: auto; margin-right: auto; background-color: #f0f0f0; border-radius: 10px; }
31     h2 {font-family: Arial; display: inline-block; text-align: center; margin: 10px; font-size: 1.2em; border: 1px solid black; padding: 5px; width: fit-content; margin-left: auto; margin-right: auto; background-color: #f0f0f0; border-radius: 10px; }
32     body {margin: 0; padding: 0; margin-top: 20px; font-family: Arial; font-size: 1em; color: black; }
33     .slider { -webkit-appearance: none; margin: 14px; width: 360px; height: 25px; background: #F0F0F0; }
34     .slider::-webkit-slider-thumb { -webkit-appearance: none; appearance: none; width: 35px; height: 35px; background: #003249; cursor: pointer; }
35     .slider::-moz-range-thumb { width: 35px; height: 35px; background: #003249; cursor: pointer; }
36   </style>
37 </head>
38 <body>
39   <h1>ESP Web Server</h1>
40   <p><input id="textSlider" type="range" value="0" max="255" step="1" class="slider"></p>
41   <script>
42     function updateSliderPWM(element) {
43       var sliderValue = document.getElementById("textSlider").value;
44       document.getElementById("textSlider").innerHTML = sliderValue;
45       console.log(sliderValue);
46     }
47   </script>
48 )</raw>
49
50 void setup() {
51   // Set the slider value
52   var xhttp = new XMLHttpRequest();
53   xhttp.open("GET", "/slider?value=" + sliderValue, true);
54   xhttp.send();
55 }
56
57 // Replaces placeholder with button section in your web page
58 String processor(const String var) {
59   //Serial.println(var);
60   if (var == "SLIDERVALUE") {
61     return sliderValue;
62   }
63   return String();
64 }
65
66 void loop() {
67   // Serial port for debugging purposes
68   Serial.begin(115200);
69
70   // configure LED PWM functionalitites
71   ledcSetup(ledChannel, freq, resolution);
72
73   // attach the channel to the GPIO to be controlled
74   ledcAttachPin(output, ledChannel);
75
76   ledcWrite(ledChannel, sliderValue.toInt());
77
78   // Connect to WiFi
79   WiFi.begin(ssid, password);
80   while (WiFi.status() != WL_CONNECTED) {
81     delay(1000);
82     Serial.println("Connecting to WiFi..");
83   }
84
85   // Print ESP Local IP Address
86   Serial.println(WiFi.localIP());
87
88   // Route for root / web page
89   server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request) {
90     request->send_P(200, "text/html", index_html, processor);
91   });
92
93   // Send a GET request to <ESP_IP>/slider?value=<inputMessage>
94   server.on("/slide", HTTP_GET, [] (AsyncWebServerRequest *request) {
95     //-----*
96     //-----*
97     Serial.println("Connecting to WiFi..");
98   });
99
100 // Print ESP Local IP Address
101 Serial.println(WiFi.localIP());
102
103 // Route for root / web page
104 server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request) {
105   request->send_P(200, "text/html", index_html, processor);
106 });
107
108 // Send a GET request to <ESP_IP>/slider?value=<inputMessage>
109 server.on("/slide", HTTP_GET, [] (AsyncWebServerRequest *request) {
110   String inputMessage;
111   // GET input value on <ESP_IP>/slider?value=<inputMessage>
112   if (request->hasParam(PARAM_INPUT)) {
113     inputMessage = request->getParam(PARAM_INPUT)->value();
114     sliderValue = inputMessage;
115     ledcWrite(ledChannel, sliderValue.toInt());
116   }
117   else {
118     inputMessage = "No message sent";
119   }
120   Serial.println(inputMessage);
121   request->send(200, "text/plain", "OK");
122 });
123
124 void loop() {
125 }

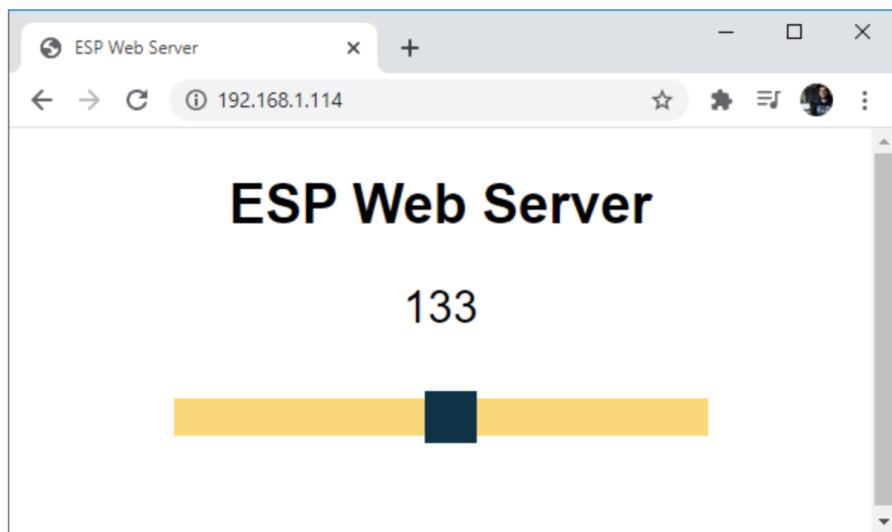
```

Setelah tahapan coding dilakukan, selanjutnya merupakan membuat tampilan pada web page sebagai berikut dengan menginput HTML :



Tampilan pada web untuk project yang dibuat hanya sederhana. Berisikan judul, deskripsi singkat serta input dari rentang jenis. Tampilan web dirancang untuk mudah dilihat dan digunakan sehingga seluruh teks HTML dengan style yang disertakan disimpan dalam variabel index_html. Sehingga dapat dilihat pada teks HTML dan melihat apa yang dilakukan setiap bagian.

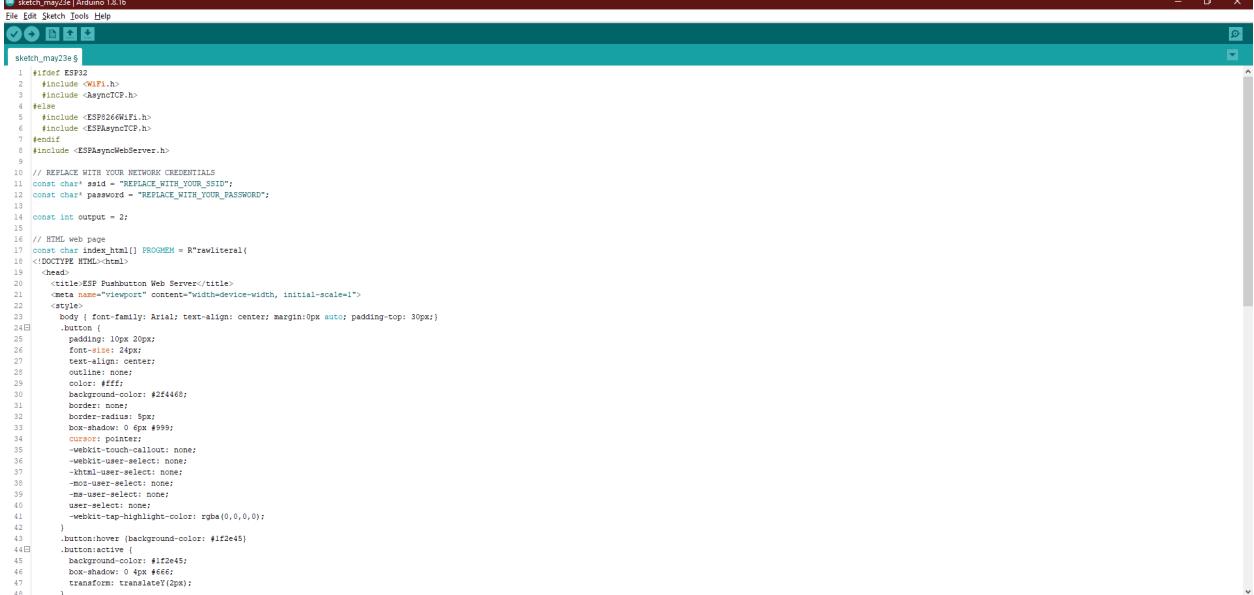
Demonstrasi web server guna melihat tampilan pada website yang telah dirancang :



BAB 11. MOMENTARY SWITCH WEB SERVER

Momentary Switch Web Server dapat digunakan untuk membuat server dengan button sehingga momentary dapat melakukan kontrol terhadap output ESP32 atau ESP8266. Jika diambil dengan asumsi sebuah lampu LED yang ditekan lama pada jarak kendali jauh maka lampu tersebut akan menyala terang, namun jika tombol kendali di lepaskan maka akan kembali redup. Namun nantinya momentary dapat mengontrol aspek lainnya.

Pada bagian ini menyimpulkan bahwa penggunaan web server sebagai kendali jarak jauh untuk menampilkan button yang dibutuhkan serta menggunakan program esp seperti berikut:



```
sketch_may23a.ino
1 //for ESP32
2 #include <WiFi.h>
3 #include <AsyncTCP.h>
4 #include <ESP8266WiFi.h>
5 #include <ESPAsyncTCP.h>
6 #endif
7 #include <ESPAsyncWebServer.h>
8
9 // REPLACE WITH YOUR NETWORK CREDENTIALS
10 const char* ssid = "REPLACE_WITH_YOUR_SSID";
11 const char* password = "REPLACE_WITH_YOUR_PASSWORD";
12
13 const int output = 2;
14
15 // HTML web page
16 const char index_html[] PROGMEM = R"rawliteral(
17 <!DOCTYPE HTML><html>
18 <head>
19   <title>ESP Pushbutton Web Server</title>
20   <meta name="viewport" content="width=device-width, initial-scale=1">
21   <style>
22     body { font-family: Arial; text-align: center; margin:0px auto; padding-top: 30px; }
23     .button {
24       padding: 10px 20px;
25       font-size: 24px;
26       text-align: center;
27       outline: none;
28       border: none;
29       color: #fff;
30       background-color: #2f4468;
31       border-radius: 5px;
32       box-shadow: 0 4px #999;
33       cursor: pointer;
34       -webkit-touch-callout: none;
35       -webkit-user-select: none;
36       -khtml-user-select: none;
37       -ms-user-select: none;
38       -moz-user-select: none;
39       -ms-user-select: none;
40       user-select: none;
41       -webkit-tap-highlight-color: rgba(0,0,0,0);
42     }
43     .button:hover {background-color: #1f2e45;}
44     .button:active {background-color: #1f2e45;
45       box-shadow: 0 4px #666;
46       transform: translateZ(2px);
47     }
48   }
49   <style>
50 </head>
51 <body>
52   <h1>ESP Pushbutton Web Server</h1>
53   <button class="button" onmousedown="toggleCheckbox('on');" ontouchstart="toggleCheckbox('on');" onmouseup="toggleCheckbox('off');" ontouchend="toggleCheckbox('off');">LED PUSHBUTTON</button>
54 </script>
55 </body>
56 </html>)<rawliteral>
57
58 voidNotFound(AsyncWebServerRequest *request) {
59   request->send(404, "text/plain", "Not found");
60 }
61
62 AsyncWebServer server(80);
63
64 void setup() {
65   Serial.begin(115200);
66   WiFi.mode(WIFI_STA);
67   WiFi.begin(ssid, password);
68   if (WiFi.waitForConnectResult() != WL_CONNECTED) {
69     Serial.println("WiFi Failed!");
70     return;
71   }
72   Serial.println();
73   Serial.print("ESP IP Address: http://");
74   Serial.println(WiFi.localIP());
75
76   pinMode(output, OUTPUT);
77   digitalWrite(output, LOW);
78
79   // Send web page to client
80   server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request){
81     request->send_P(200, "text/html", index_html);
82   });
83
84   // Receive an HTTP GET request
85   server.on("/on", HTTP_GET, [] (AsyncWebServerRequest *request) {
86     digitalWrite(output, HIGH);
87     request->send(200, "text/plain", "ok");
88   });
89 }
```

```
96 // Receive an HTTP GET request
97 #include <ESP8266WebServer.h>
98
99 AsyncWebServer server(80);
100
101 void setup() {
102     // Connect to WiFi
103     WiFi.begin("SSID", "PASSWORD");
104     // Start the web server
105     server.on("/", HTTP_GET, [] () {
106         // Set the content type to HTML
107         server.setContentTypeName("text/html");
108         // Set the content
109         server.send(200, "text/html", "



ESP8266 Web Server

# Welcome to the ESP8266 Web Server!



Control the LED:



Turn On



Turn Off

");
110     });
111     // Handle 404 errors
112     server.onNotFound([] () {
113         server.send(404, "text/plain", "Page not found");
114     });
115     // Start the server
116     server.begin();
117 }
118
119 void loop() {
120     // Check if there's a new connection
121     WiFiClient client = WiFi.available();
122     if (client) {
123         // Handle the connection
124         server.handleClient();
125     }
126 }
```

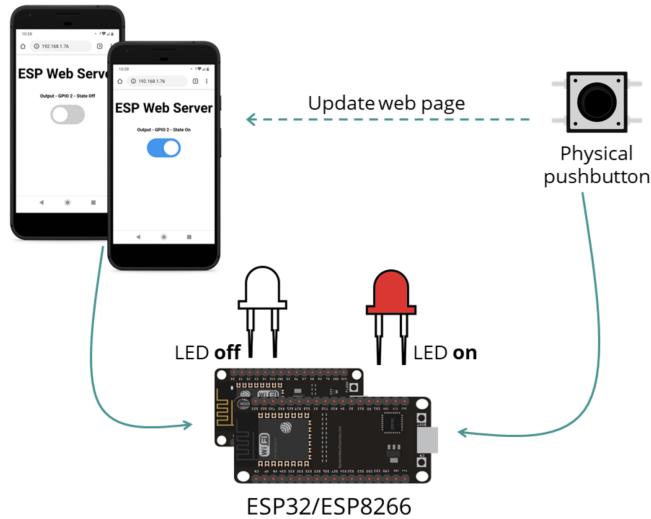
Pada tampilan button dapat menggunakan program HTML yang dimana dapat disambungkan dengan metode yang digunakan.

BAB 12. PHYSICAL BUTTON WEB SERVER

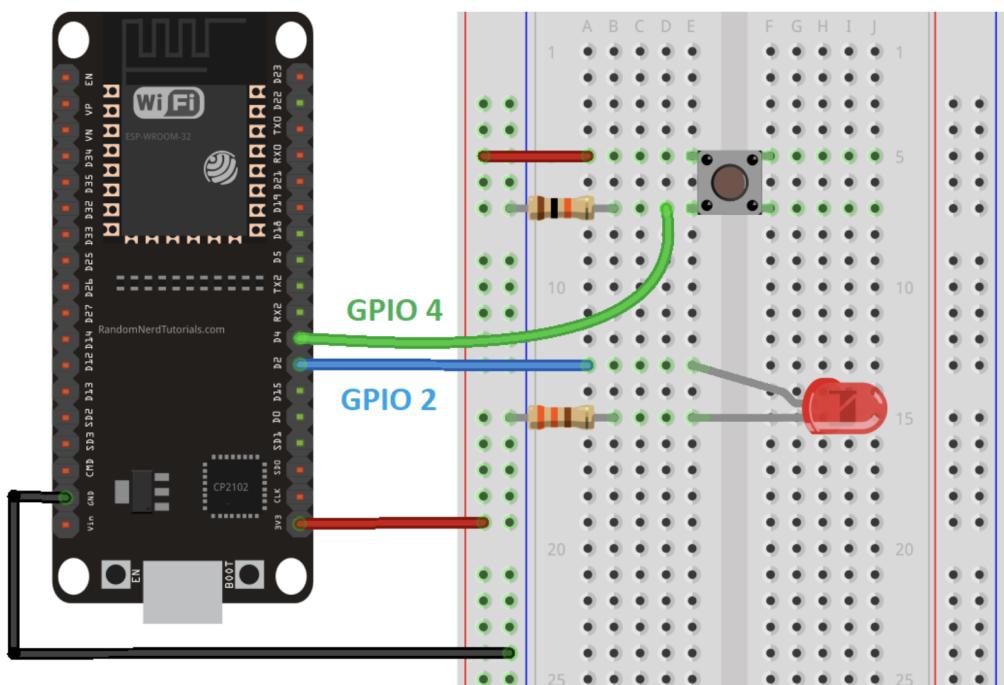
Physical Button Webserver yaitu merupakan mengontrol output pada ESP32 atau ESP8266, dengan menggunakan web server dan tombol secara langsung. Sehingga output yang dihasilkan pada laman web yang terbaru dapat terdeteksi nantinya apakah terdapat perubahan yang dilakukan pada web server atau tombol secara langsung.

Dapat dijelaskan secara lebih ringkas tahapannya seperti :

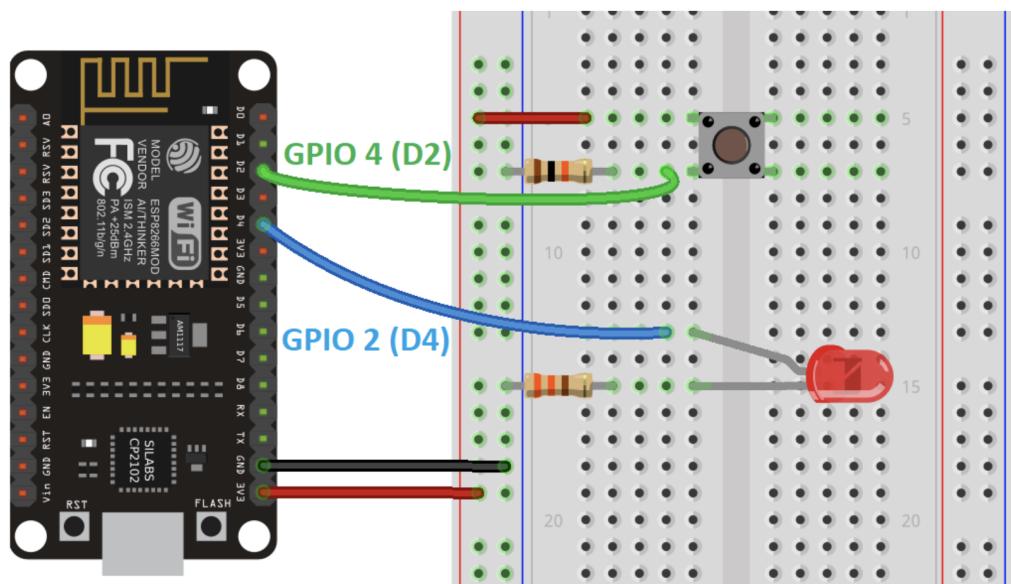
- ESP32 atau ESP8266 melakukan hosting web server yang memungkinkan untuk user untuk mengontrol status secara keluaran.
- Status yang sudah didapatkan saat ini ditampilkan pada web server.
- ESP nantinya juga terhubung ke tombol secara langsung yang dapat melakukan kontrol pada output yang sama.
- Sehingga nantinya jika akan mengubah status output menggunakan tombol secara langsung , status saat ini juga diperbarui pada web server.



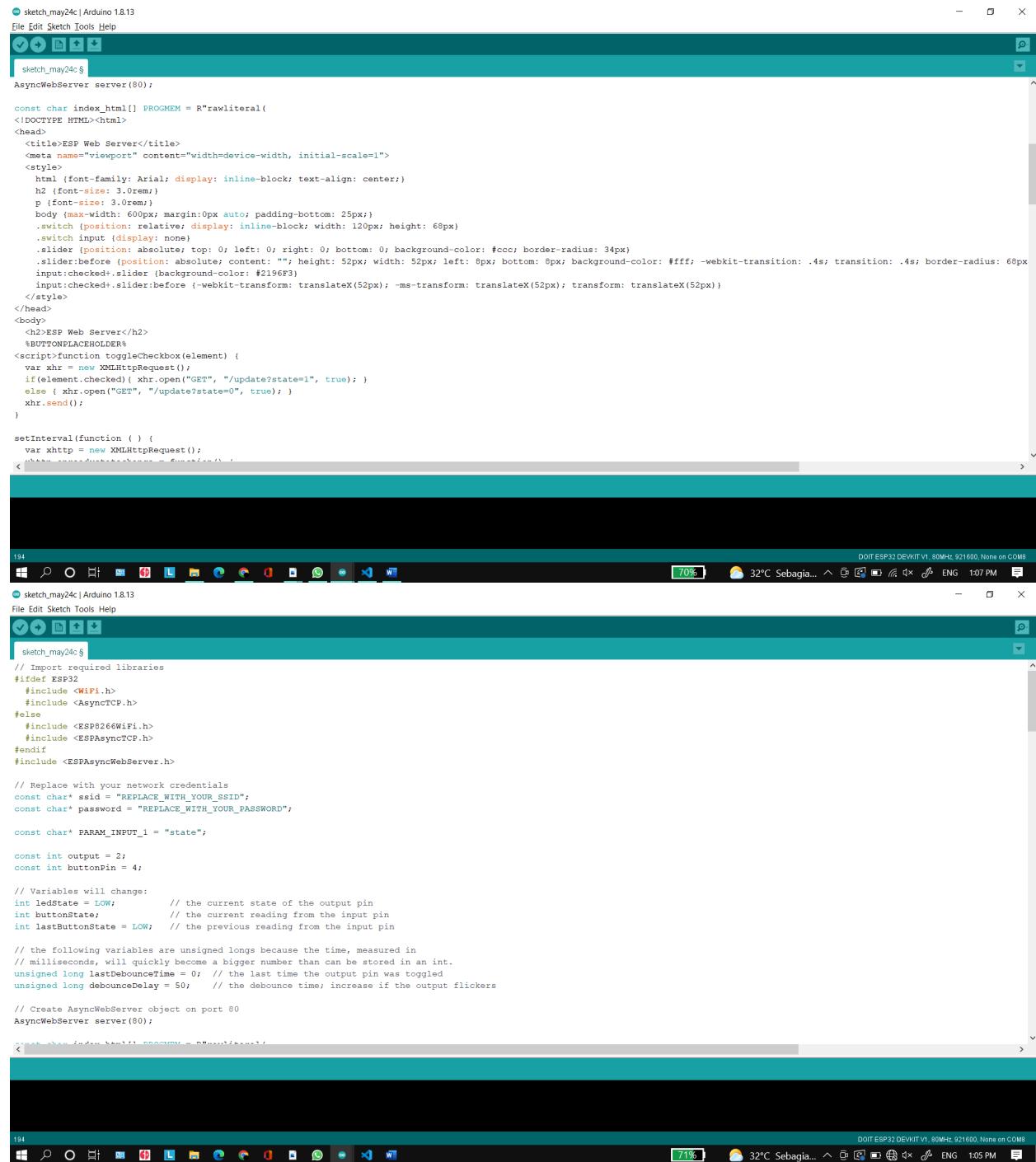
ESP32 Schematic



ESP8266 NodeMCU Schematic



Berikut merupakan code pemrograman pada Physical Button Webserver.



The screenshot shows two instances of the Arduino IDE. The top instance displays the following code:

```
sketch_may24c | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_may24c $ AsyncWebServer server(80);

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE 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; background-color: transparent; border: 2px solid #000; transform: translateX(52px); -ms-transform: translateX(52px); transform: translateX(52px);}
input:checked+.slider::before {background-color: #2196F3; border: 2px solid #000; transform: translateX(0px); -ms-transform: translateX(0px); transform: translateX(0px);}
</style>
</head>
<body>
<h2>ESP Web Server</h2>
<#BUTTONPLACESHOLDERS>
<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.open("GET", "/index.html");
xhttp.onreadystatechange = function() {
if(xhttp.readyState == 4 && xhttp.status == 200) {
document.getElementById("index").innerHTML = xhttp.responseText;
}
}
xhttp.send();
}, 1000);
</script>
<div style="text-align: center; margin-top: 20px;">
<input type="checkbox" checked="" onclick="toggleCheckbox(this)"> State: <span id="index">ON</span>

```

The bottom instance shows the code with comments removed:

```
sketch_may24c | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_may24c $ // 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";

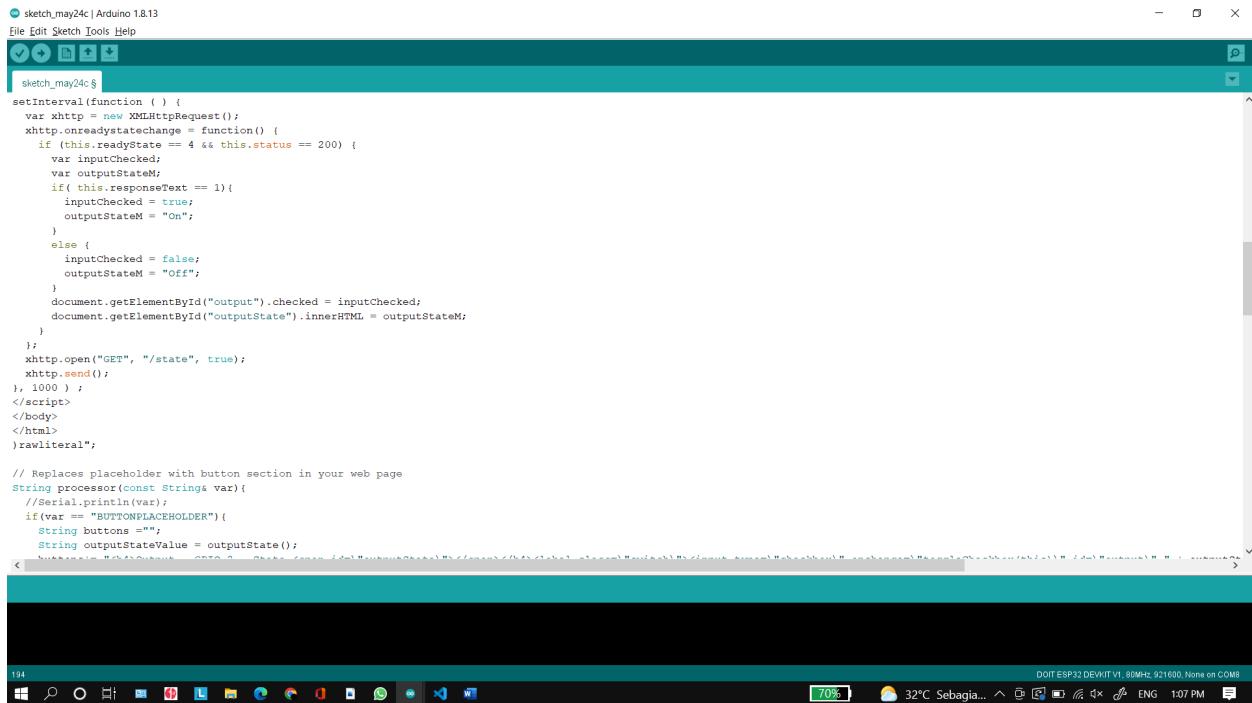
const int output = 2;
const int buttonpin = 4;

// 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);

<#BUTTONPLACESHOLDERS>
```

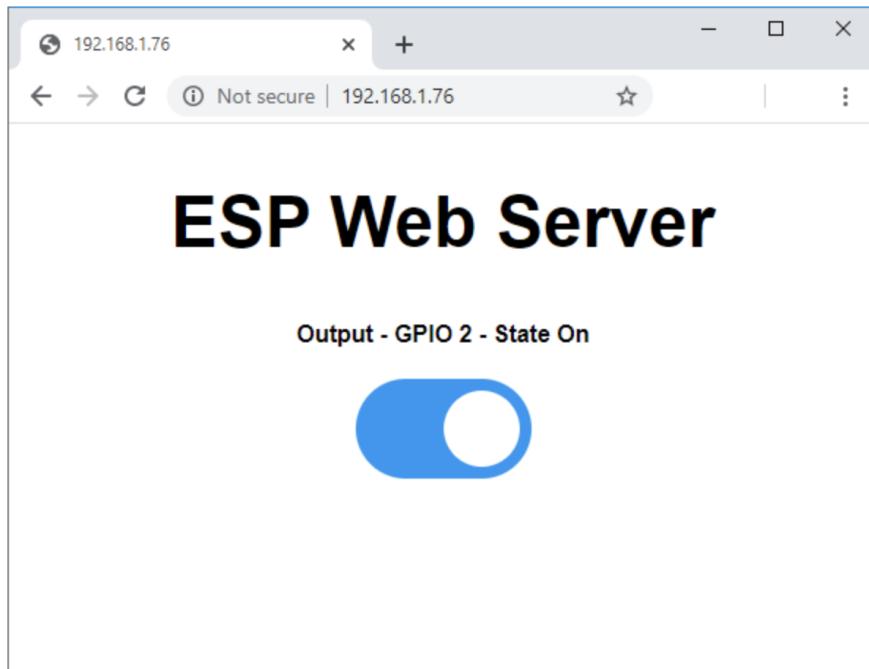


```
sketch_may24c | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_may24c.js
setInterval(function () {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var inputChecked;
      var outputStateM;
      if( this.responseText == 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>
)rawiteral";

// 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 = " <input type='button' value='";
    buttons += outputStateValue;
    buttons += "' > ";
    return buttons;
  }
  return var;
}

```

Jika pemrograman telah selesai dilakukan serta pembuatan website telah dilakukan tahapan selanjutnya merupakan demonstrasi pada website seperti berikut :

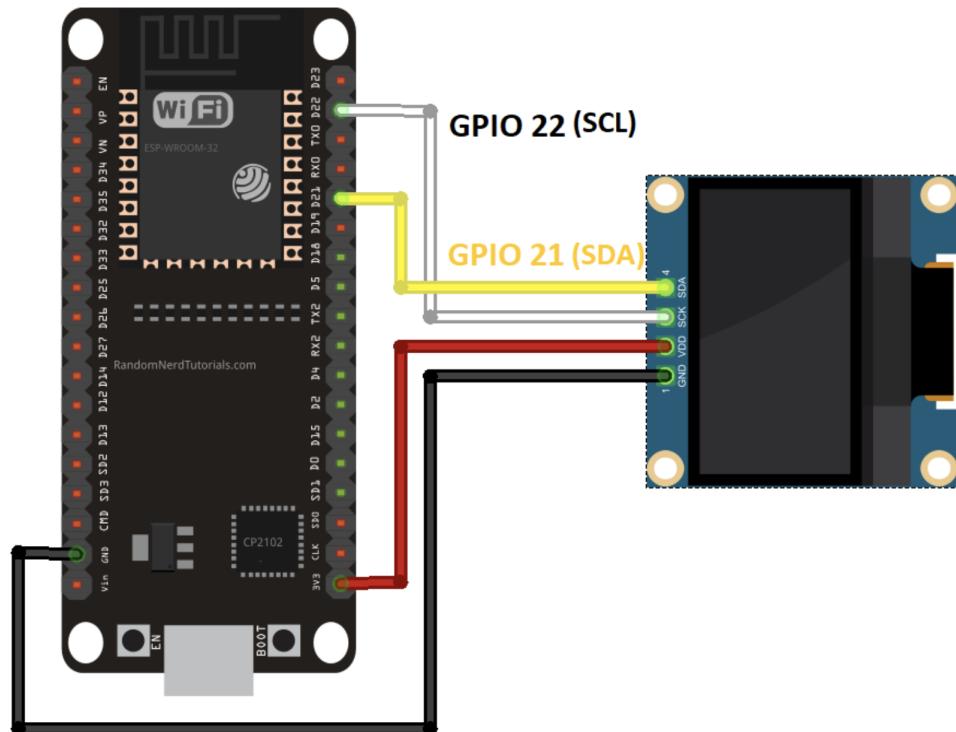


BAB 13. OLED DISPLAY

Oled Display merupakan bentuk suatu pemrograman dengan cara menunjukkan teks, mengatur font yang berbeda, menggambar suatu bentuk, serta menampilkan gambar bitmap. Oled Display nantinya menggunakan layar OLED SSD1306 0.96 inci dengan ESP32 menggunakan arduino IDE.

Layar Oled nantinya tidak memerlukan lampu sebagai latarnya untuk menghasilkan kontras di lingkungan yang gelap, selain itu piksel yang dimiliki hanya menggunakan energi pada saat cahaya menyala saja sehingga layar OLED menggunakan sedikit energi jika dibandingkan dengan layar lainnya. Model yang digunakan memiliki empat pin dan berkomunikasi dengan mikrokontroler sejenis dengan menggunakan protokol komunikasi I2C. Terdapat model yang dihadirkan dengan pin RESET ekstra atau yang berkomunikasi menggunakan protokol komunikasi SPI.

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



Berikut merupakan code pemrograman pada Oled Display.

The screenshot shows the Arduino IDE interface with the code for a snowflake animation. The code includes definitions for screen dimensions, a logo bitmap, and a main loop with a delay. The IDE window has a teal header bar with tabs for sketch, may24c, and Arduino 1.8.13. Below the header is a toolbar with icons for file operations, sketch tools, and help. The main area contains the C++ code. At the bottom of the IDE window, there is a status bar showing the board as D0IT ESP32 DEVKIT V1, 80MHz, 921600, None on COM8, and system information like battery level (70%), temperature (32°C), and time (10:09 PM). The background of the IDE window is black, and the code is white with syntax highlighting.

```
#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, B10000000,
B00000001, B10000000,
B00000001, B10000000,
B00000011, B11100000,
B11110011, B11100000,
B11111100, B11110000,
B01111110, B11111111,
B00110012, B10011111,
B00011111, B11111100,
B00001101, B01110000,
B00011011, B10100000,
B00111111, B11100000,
B01111100, B11110000,
B01110000, B01110000,
B00000000, B00110000 },
```

The screenshot shows the Arduino IDE interface with the code for a basic display setup and drawing operations. The code includes a setup function with serial.begin(115200) and a loop with a delay of 2000ms. It then performs various drawing commands like drawPixel, clearDisplay, and drawRect. The IDE window has a teal header bar with tabs for sketch, may24c, and Arduino 1.8.13. Below the header is a toolbar with icons for file operations, sketch tools, and help. The main area contains the C++ code. At the bottom of the IDE window, there is a status bar showing the board as D0IT ESP32 DEVKIT V1, 80MHz, 921600, None on COM8, and system information like battery level (70%), temperature (32°C), and time (10:09 PM). The background of the IDE window is black, and the code is white with syntax highlighting.

```
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();

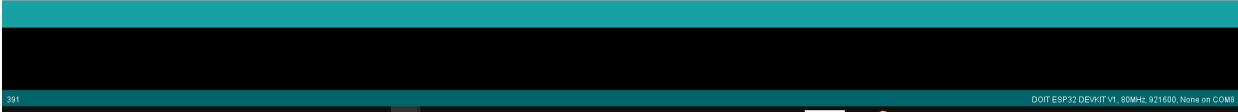
// Draw a single pixel in white
display.drawPixel(10, 10, WHITE);

// 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)
```

```
sketch_may24c | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_may24c $ 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(logobmp, LOGO_WIDTH, LOGO_HEIGHT); // Animate bitmaps
```



391 D01T ESP32 DEVKIT V1, 80MHz, 921600, None on COM8
70% 32°C Sebagia... ENG 1:09 PM

```
sketch_may24c | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_may24c $ }

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);
}
```



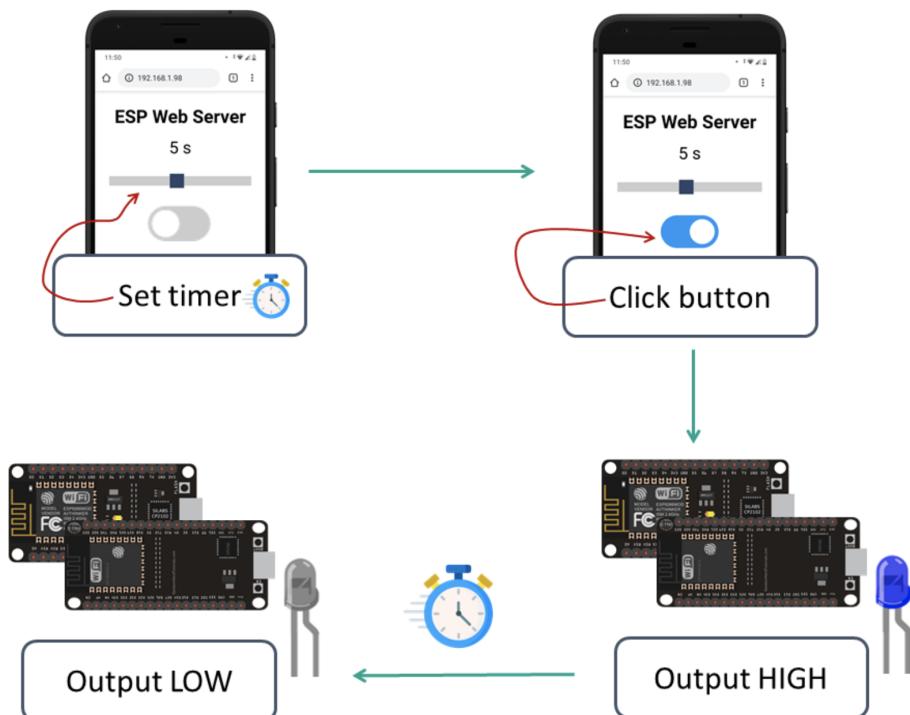
391 D01T ESP32 DEVKIT V1, 80MHz, 921600, None on COM8
70% 32°C Sebagia... ENG 1:09 PM

BAB 14. TIMER ATAU PULSE WEB SERVER

Timer atau Pulse Web Server digunakan untuk menetapkan jumlah detik pada slider, yaitu memiliki cara kerja untuk menggunakan web server untuk melakukan kontrol output NodeMCU ESP32 atau ESP8266 dengan menggunakan Arduino IDE. Sehingga penentuan timer dapat diatur menggunakan slider pada halaman website, tentunya sistem pemrograman sejenis ini sangat berguna untuk melakukan kontrol terhadap yang membutuhkan sinyal tinggi selama beberapa detik yang telah ditentukan untuk saatnya digerakkan.

Dapat dijelaskan secara lebih ringkas tahapannya seperti :

- ESP32 atau ESP8266 menghosting server web yang memungkinkan untuk mengontrol output dengan sinyal
- Server web berisi penggeser yang memungkinkan untuk menentukan lebar pulsa (berapa detik output harus tinggi)
- Ada tombol ON/OFF. Dapat diatur ke ON untuk mengirim pulsa. Setelah itu, Anda akan melihat timer berkurang selama durasi lebar pulsa;
- Ketika pengatur waktu berakhir, output diatur ke rendah, dan tombol server web kembali ke status OFF;
- Server web ini dapat berguna untuk mengontrol perangkat yang membutuhkan pulsa untuk diaktifkan seperti pembuka pintu garasi, misalnya.



Berikut merupakan code pemrograman pada Timer atau Pulse Webserver.

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_may24c | Arduino 1.8.13
- Menu Bar:** File Edit Sketch Tools Help
- Sketch Name:** sketch_may24c
- Code Content:**

```
// 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;}
  .switch {font-size: 2.0rem;}
```
- Bottom Status Bar:** 182 70% 32°C Sebagai... ENG 1:10 PM
- Bottom Taskbar:** Shows icons for various applications like File Explorer, Command Prompt, and browser.

```
sketch_may24c | Arduino 1.8.13
File Edit Sketch Tools Help

sketch_may24c.js
<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: 60px}
  .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>
  %BUTTONONPLACEHOLDER%
<script>
function togglecheckbox(element) {
  var sliderValue = document.getElementById("timerSlider").value;
  var xhr = new XMLHttpRequest();
  if(element.checked){ xhr.open("GET", "/updatestate=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);
}
</script>
```

```
sketch_may24c | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_may24c
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 == "IMERVALUE"){
return timerSliderValue;
}
return String();
}

162
OOIESP32_DEVKITV1 80MHz: 921600, None on COM8
70% 32°C Sebagia... 1:10 PM
```

Output yang didapatkan pada Timer dan Pulse Web Server yaitu berupa :

