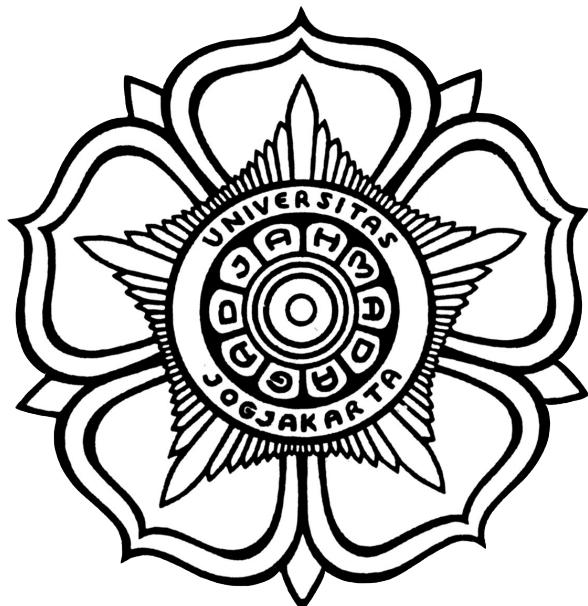


LAPORAN PRAKTIKUM TEKNIK KENDALI DAN MESIN LISTRIK

“Modul Pengenalan 32 bit Microcontroller”

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



Disusun Oleh :

Kelompok 10

Ridwan Ferdian (19/447126/SV/16845)

Rafif Rafikhansa (19/447315/SV/17009)

Kelas : ARM 2

TEKNOLOGI REKAYASA MESIN

DEPARTEMEN TEKNIK MESIN SEKOLAH VOKASI

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2022

1. Spesifikasi

ESP32 adalah mikrokontroller berharga rendah dan hemat energi dengan wifi dan dual-mode bluetooth terintegrasi. Generasi ESP32 menggunakan mikroprosesor Tensilica Xtensa LX6 sebagai inti. Baik dalam mode single-core maupun dual-core. Pada ESP32 terdapat pin yang dapat digunakan sebagai input atau output untuk menyalakan LCD, lampu, bahkan untuk menggerakkan motor DC yang tersedia dengan melakukan pemrograman pada sistem aplikasi Arduino IDE atau dapat dilakukan dengan micropython.

ESP32 menggunakan CMOS untuk radio dan pita dasar terintegrasi penuh chip tunggal, sementara juga mengintegrasikan sirkuit kalibrasi canggih yang memungkinkan solusi menghilangkan ketidaksempurnaan sirkuit eksternal atau menyesuaikan dengan perubahan kondisi eksternal. Dengan demikian, produksi massal solusi ESP32 tidak memerlukan peralatan pengujian Wi-Fi yang mahal dan khusus.

ESP32 adalah solusi yang sangat terintegrasi untuk aplikasi IoT Wi-Fi-dan-Bluetooth, dengan sekitar 20 komponen eksternal. ESP32 mengintegrasikan saklar antena, balun RF, penguat daya, penguat penerima kebisingan rendah, filter, dan modul manajemen daya. Dengan demikian, seluruh solusi minimal menempati area Printed Circuit Board (PCB).

Berikut merupakan spesifikasi - ESP32 DEVKIT V1 DOIT

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.

Spesifikasi chip ESP32 :

- ESP32 memiliki dual core processor, ini berarti memiliki 2 prosesor.
- Memiliki Wifi dan bluetooth built-in, yang berarti kita dapat melakukan pekerjaan kontrol yang membutuhkan bluetooth dan Wifi secara langsung tanpa perlu menambah modul lain.
- ESP32 dapat menjalankan program 32 bit.
- Frekuensi clock bisa mencapai 240 MHz dan memiliki RAM 512 kB.

- Produk modul yang menggunakan chip ESP32 ini memiliki 30 atau 36 pin.
- Memiliki berbagai macam fungsi yang tersedia, seperti: tombol tekan, ADC, DAC, UART, SPI, I2C dan banyak lagi.
- Terdapat sensor efek hall built-in dan sensor suhu built-in.

A. 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

B. 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

C. MCU and Advanced Features

1. 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

2. Clock and Timer

- 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

D. 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

E. Security

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

F. 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

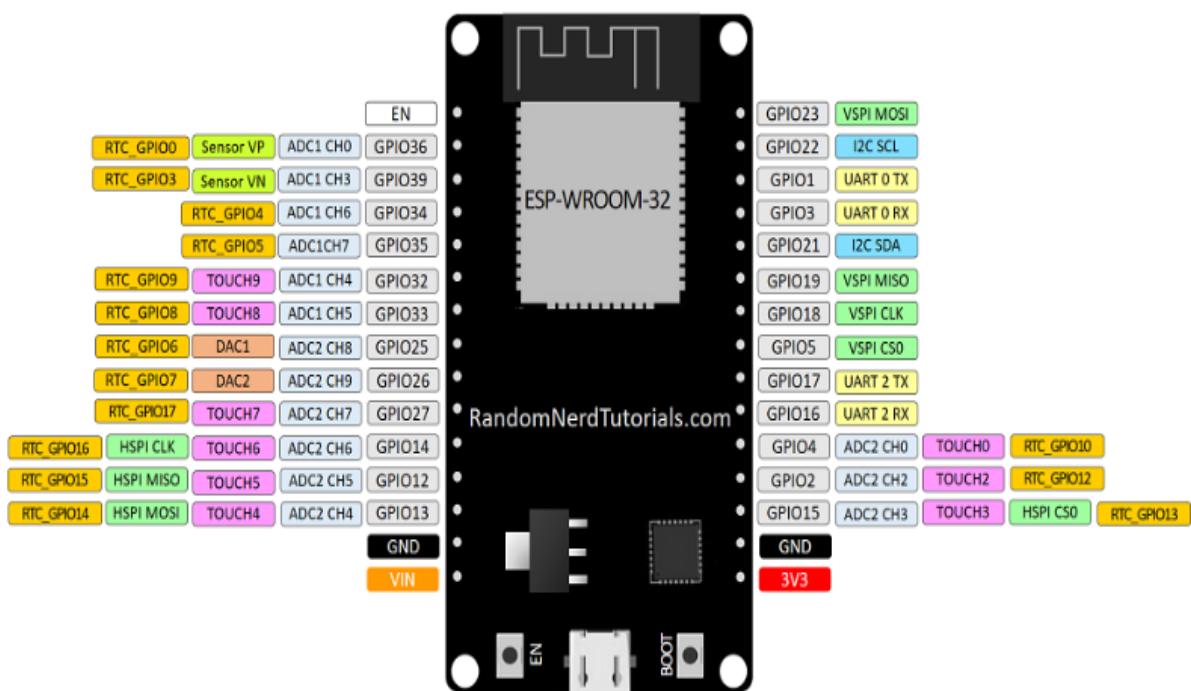
- Smart Agriculture
- Smart greenhouses
- Smart irrigation
- Agriculture robotics
- Audio Applications
- Internet music players
- Live streaming devices

2. Kaki-kaki atau Pinout Beserta Fungsinya

ESP32 memiliki lebih banyak GPIO dengan lebih banyak fungsi dibandingkan dengan ESP826. Dengan ESP32 Anda dapat memutuskan pin mana yang UART, I2C, atau SPI - Anda hanya perlu mengaturnya pada kode. Ini dimungkinkan karena fitur multiplexing chip ESP32 yang memungkinkan untuk menetapkan beberapa fungsi ke pin yang sama. Jika Anda tidak mengaturnya pada kode, pin akan digunakan sebagai default – seperti yang ditunjukkan pada gambar di bawah ini (lokasi pin dapat berubah tergantung pada produsennya).

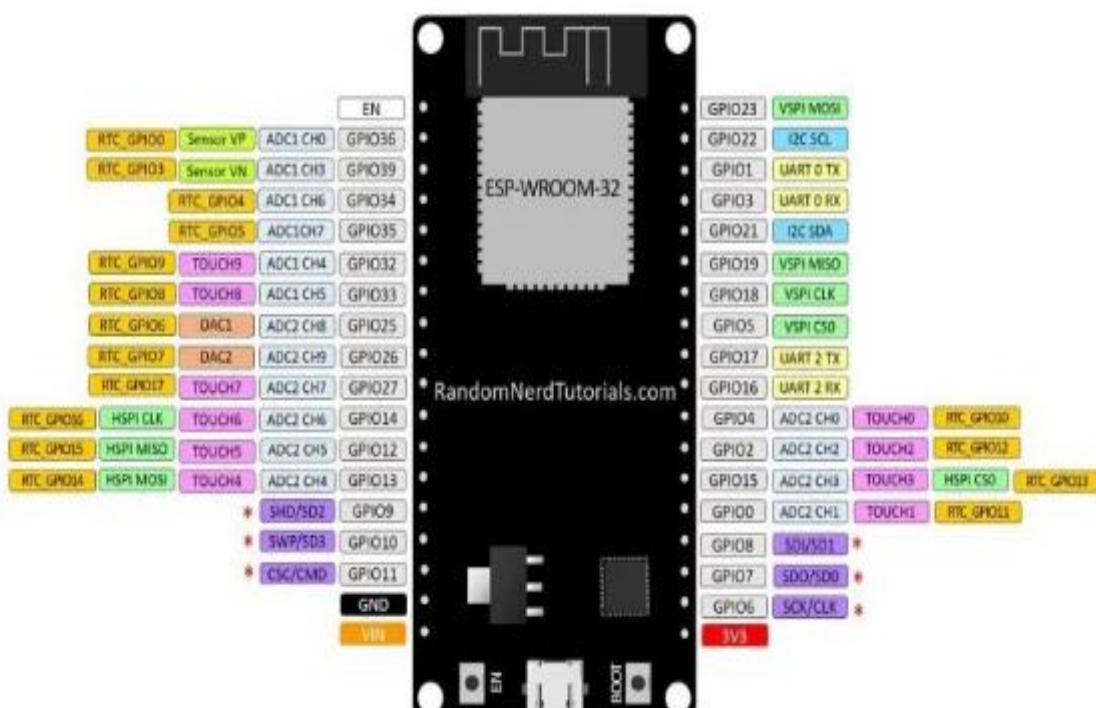
ESP32 DEVKIT V1 – DOIT

version with 30 GPIOs



Version with 36 GPIOs

ESP32 DEVKIT V1 – DOIT
version with 36 GPIOs



Pada warna hijau dapat disimpulkan 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

- **SPI (Serial Peripheral Interface)**

Secara default, mapping pin untuk SPI di ESP 32 adalah sebagai berikut :

SPI	MOSI	MISO	CLK	CS
VSPI	GPIO 23	GPIO 19	GPIO 18	GPIO 5
HSPI	GPIO 13	GPIO 12	GPIO 14	GPIO 15

- **Arus yang Dapat Ditarik Pada GPIO**

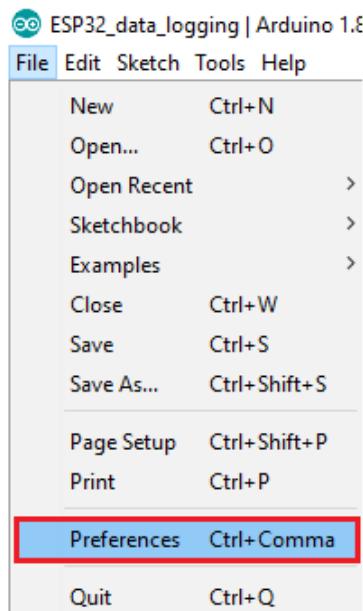
Arus maksimal yang dapat ditarik oleh periferal di tiap GPIO adalah 12mA, sebagaimana tertulis pada tabel.

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

3. PENGGUNAAN ARDUINO IDE

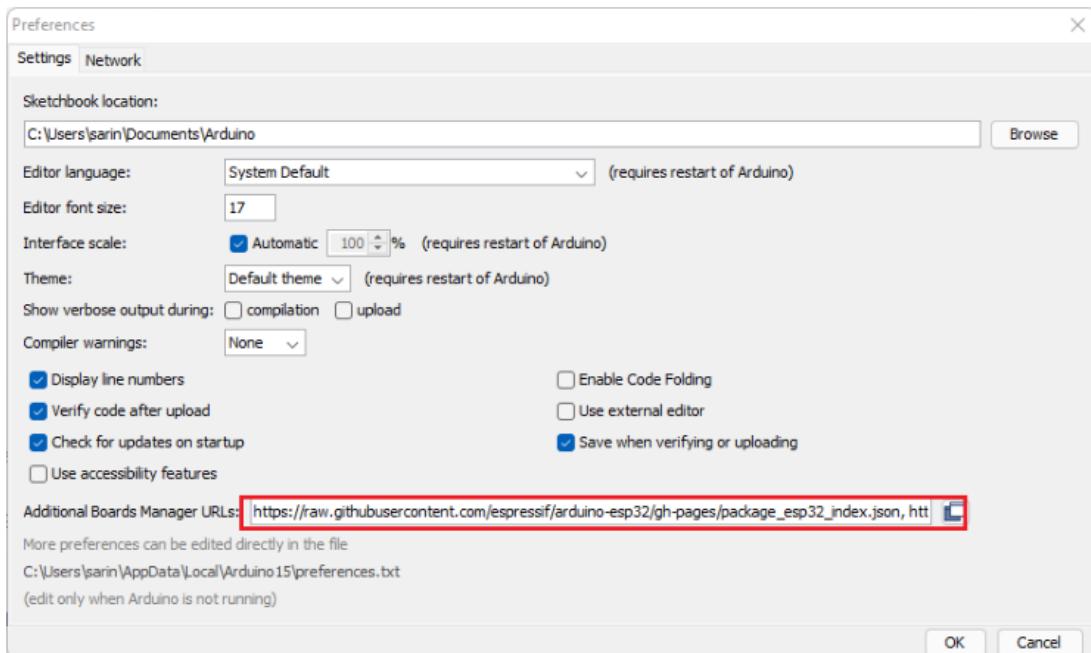
Arduino IDE merupakan sebuah Software Compiler dari ARDUINO yang dapat digunakan untuk melakukan programming logic microcontroller salah satunya ESP32.

1. In your Arduino IDE, go to **File> Preferences**

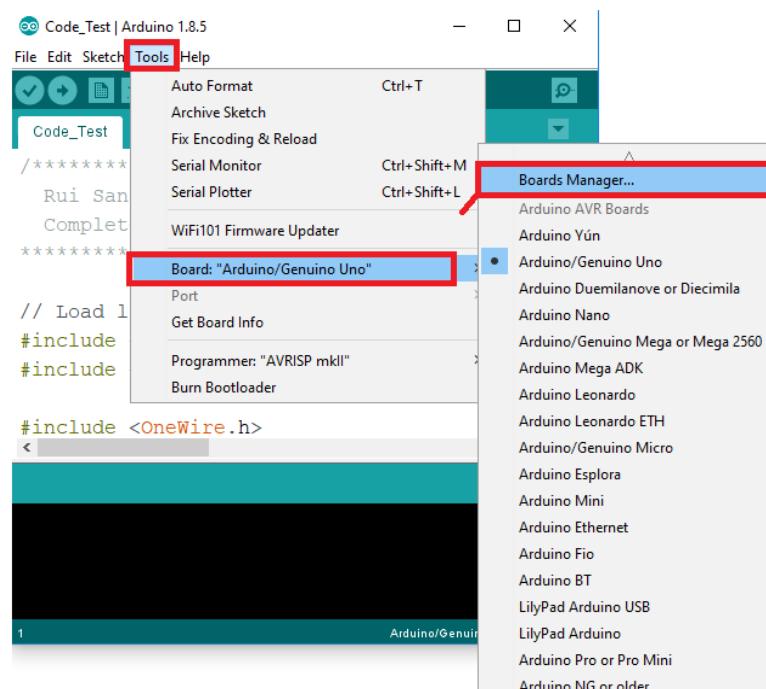


2. Enter the following into the “Additional Board Manager URLs” field:

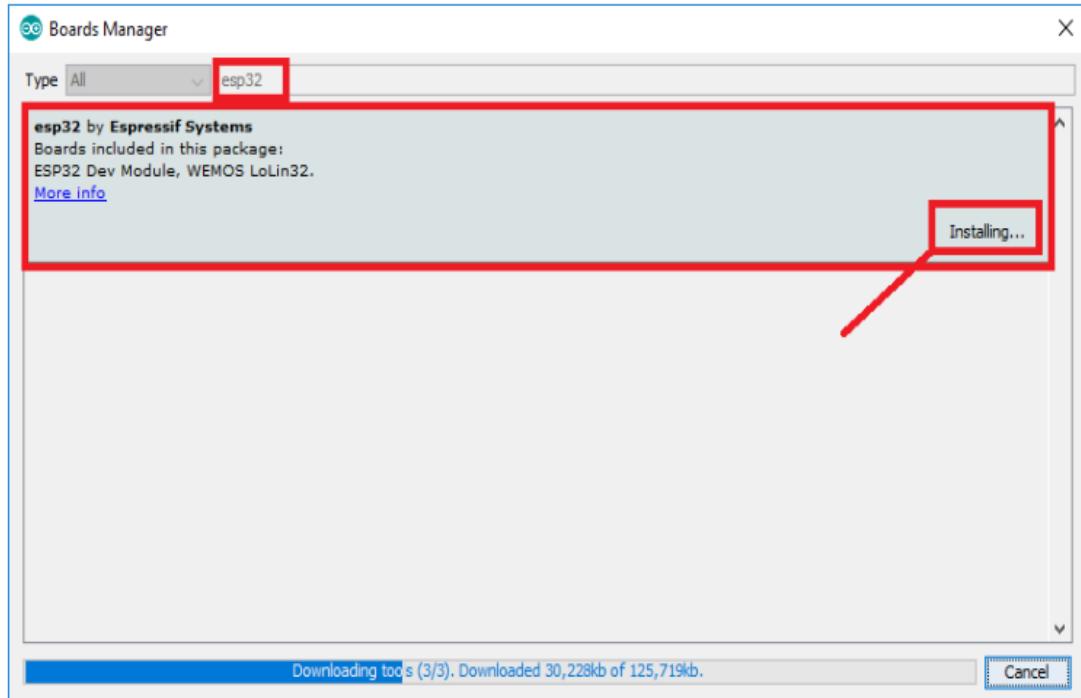
Then, click the “OK” button:



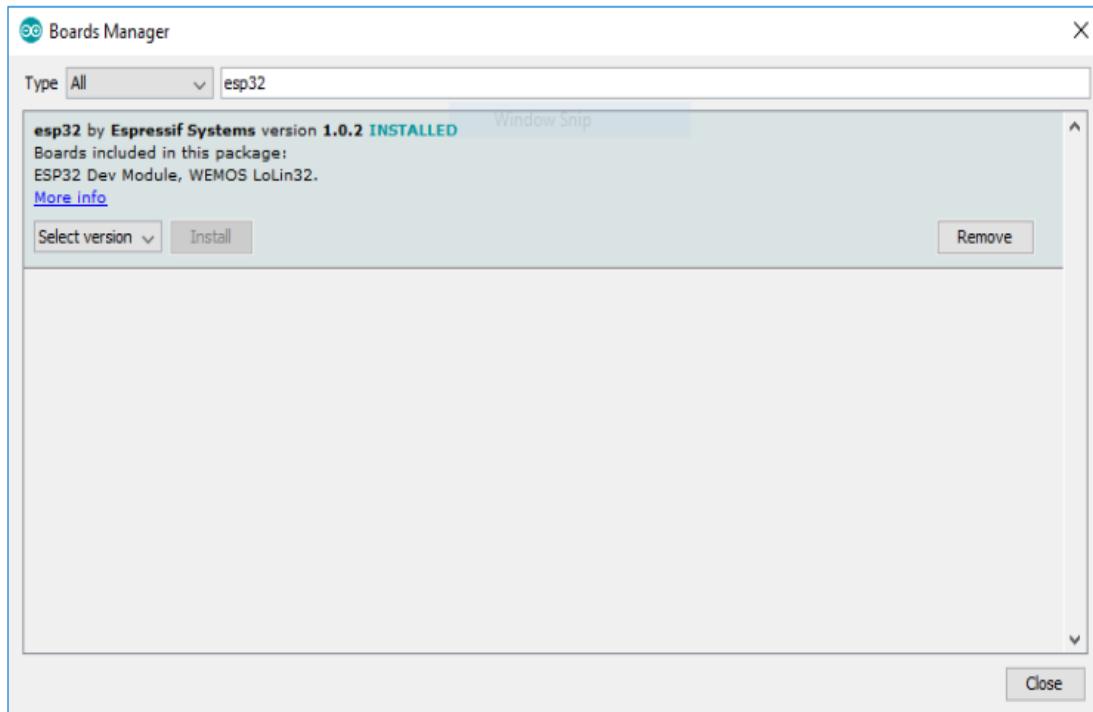
3. Open the Boards Manager. Go to **Tools > Board > Boards Manager...**



4. Search for **ESP32** and press install button for the “**ESP32 by Espressif Systems**“:



5. That's it. It should be installed after a few seconds.



4. ESP32 Input/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

A. ESP32 Input Digital

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

Pertama, atur GPIO yang ingin dibaca 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).

B. Kontrol ESP32 Output Digital

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

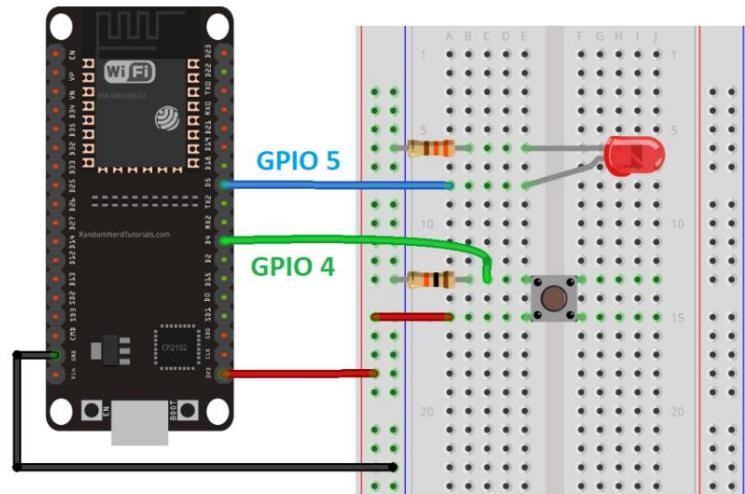
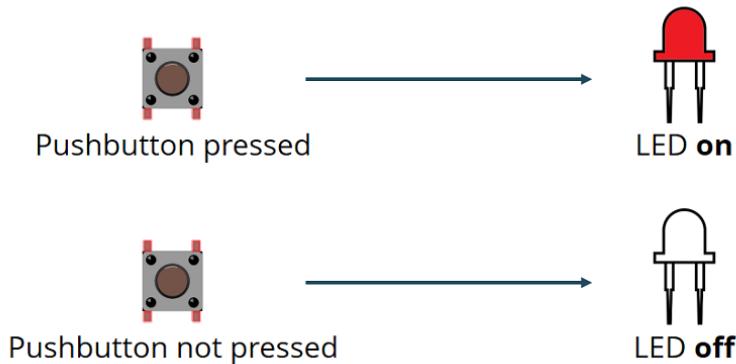
Pertama, Kita perlu mengatur GPIO yang ingin dikontrol sebagai OUTPUT. Menggunakan fungsi pinMode() sebagai berikut :

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

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

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

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



C. Kode Pemrograman

```

1_LedPush4.ino
/*
  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;

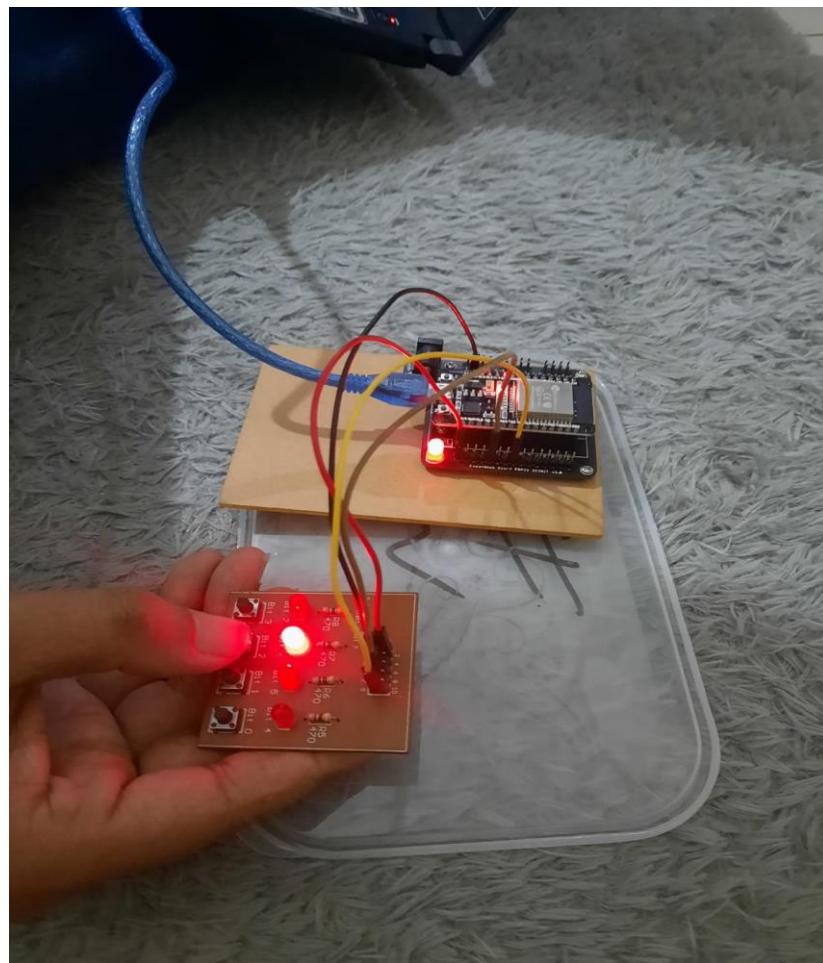
void setup() {
  Serial.begin(115200);
  // initialize the pushbutton pin as an input
  pinMode(buttonPin, INPUT);
  // initialize the LED pin as an output
  pinMode(ledPin, OUTPUT);
}

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);
  }
}

Data uploading
Changed.
Configuring flash size...
Flash will be erased from 0x00001000 to 0x00005fff...

```

D. Hasil Percobaan



5. ESP32 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:

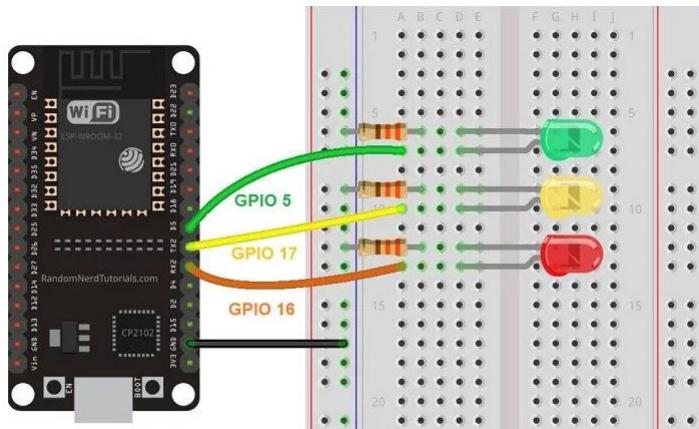
```
ledcAttachPin(GPIO, channel)
```

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

6. 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.



- Menentukan tempat pin LED terpasang dengan GPIO 16

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

Kemudian, mengatur properti sinyal PWM. Menentukan frekuensi 5000 Hz, memilih saluran 0 untuk menghasilkan sinyal, dan mengatur resolusi 8 bit. Kita dapat memilih properti lain, berbeda dari ini, untuk menghasilkan sinyal PWM yang berbeda.

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

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

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

Dalam loop, akan memvariasikan siklus tugas antara 0 dan 255 untuk meningkatkan kecerahan LED.

```

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

```

Dan kemudian, antara 255 dan 0 untuk mengurangi kecerahan.

```

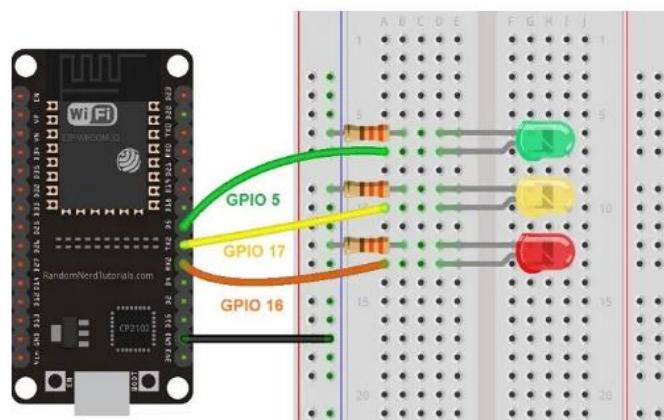
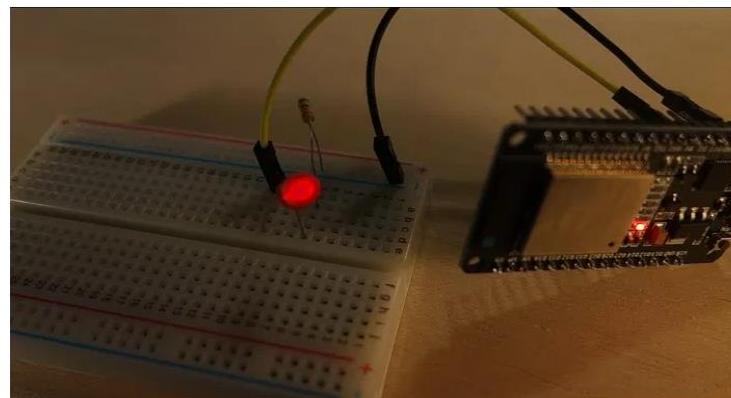
for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
}

```

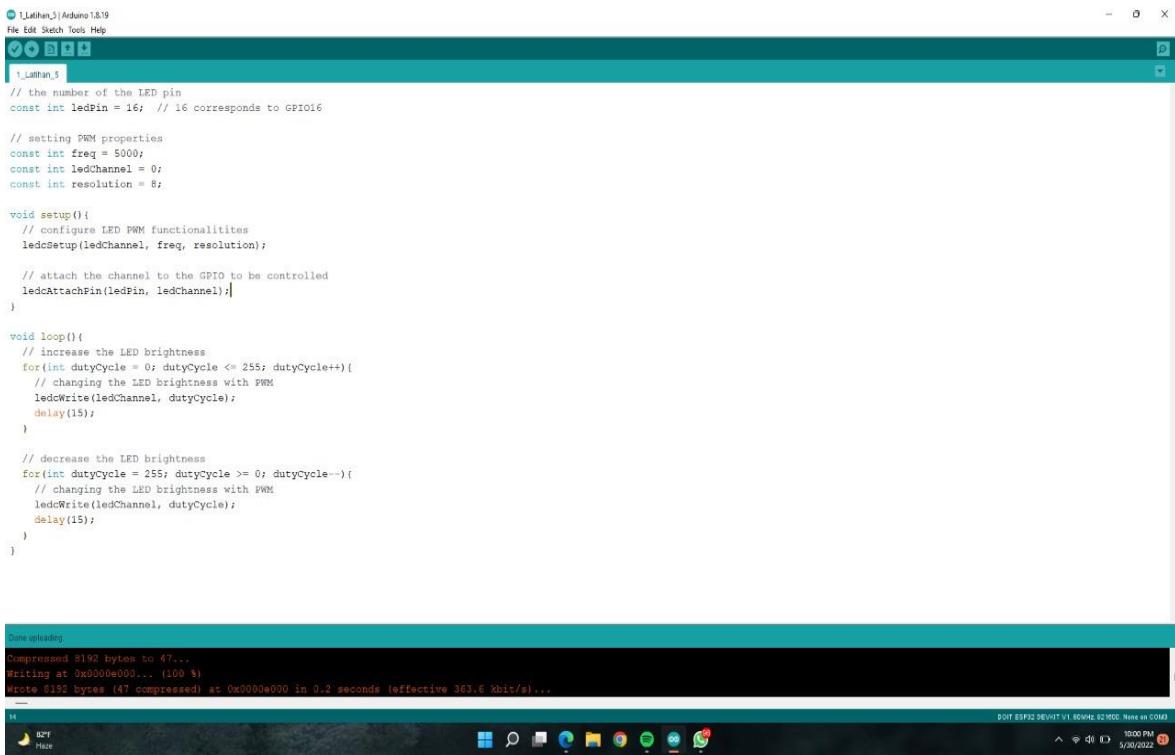
Untuk mengatur kecerahan LED, hanya perlu menggunakan fungsi `ledcWrite()` yang menerima sebagai argumen saluran yang menghasilkan sinyal, dan siklus tugas.

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

Saat kami menggunakan resolusi 8-bit, siklus tugas akan dikontrol menggunakan nilai dari 0 hingga 255. Perhatikan bahwa dalam fungsi `ledcWrite()` kita menggunakan saluran yang menghasilkan sinyal, dan bukan GPIO.



- Kode Pemrograman



```
1_Lamhan_5
// the number of the LED pin
const int ledPin = 16; // 16 corresponds to GPIO16

// setting PWM properties
const int freq = 5000;
const int ledChannel1 = 0;
const int resolution = 8;

void setup(){
  // configure LED PWM functionalities
  ledcSetup(ledChannel1, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(ledPin, ledChannel1);
}

void loop(){
  // increase the LED brightness
  for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel1, dutyCycle);
    delay(15);
  }

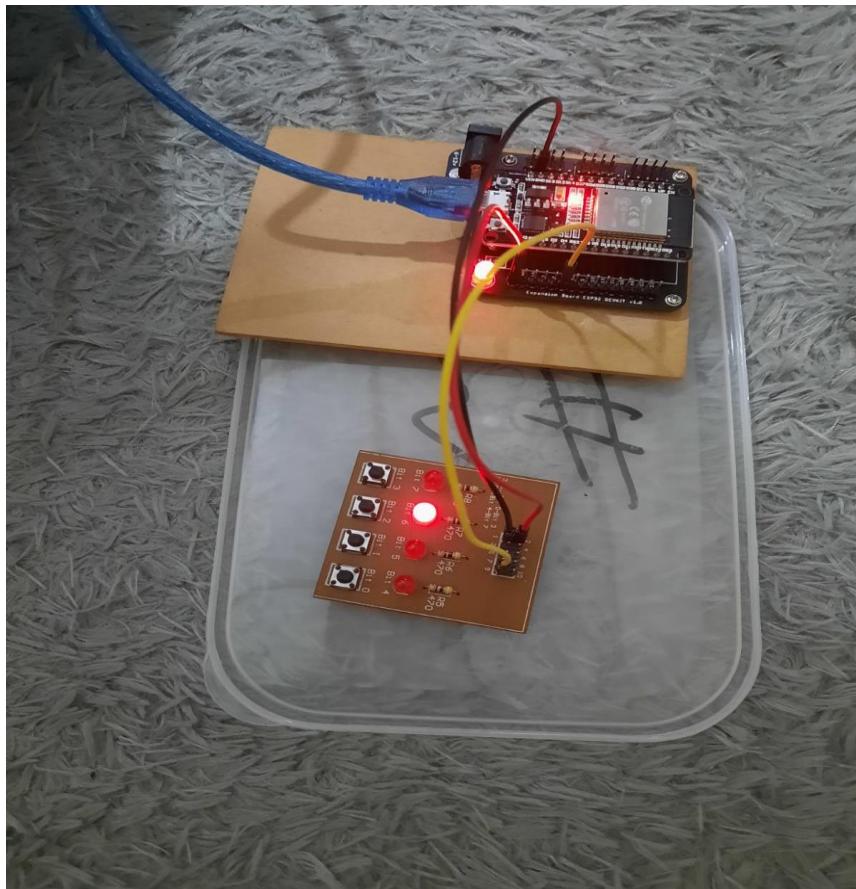
  // decrease the LED brightness
  for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel1, dutyCycle);
    delay(15);
  }
}
```

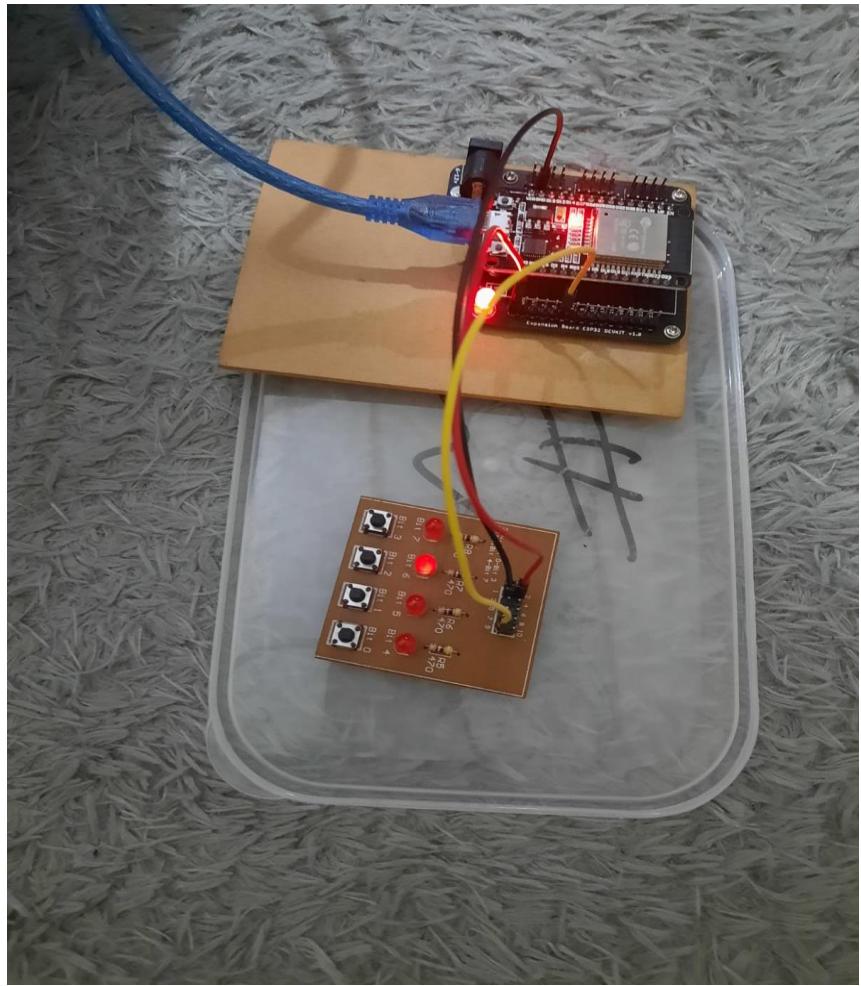
Done uploading
Compressed 5192 bytes to 47...
Writing at 0x0000e000... (100 %)
Wrote 5192 bytes (47 compressed) at 0x0000e000 in 0.3 seconds (effective 363.6 kbit/s)...

DOIT ESP32 DevKit v1.80MHz 02 MODE Read in COM3
10:00 PM 5/30/2022

- Hasil Percobaan

Pada ESP32 PMW lampu LED berkedip





6. ESP32 Interrupt

Untuk memicu suatu peristiwa dengan sensor gerak PIR, menggunakan interupsi. Interupsi berguna untuk membuat sesuatu terjadi secara otomatis dalam program mikrokontroler, dan dapat membantu memecahkan masalah waktu.

Dengan interupsi, kita tidak perlu terus-menerus memeriksa nilai pin saat ini. Dengan interupsi, ketika perubahan terdeteksi, suatu peristiwa dipicu (fungsi dipanggil).

Untuk menyetel interupsi di Arduino IDE, menggunakan fungsi `attachInterrupt()`, yang menerima sebagai argumen: pin GPIO, nama fungsi yang akan dieksekusi, dan mode:

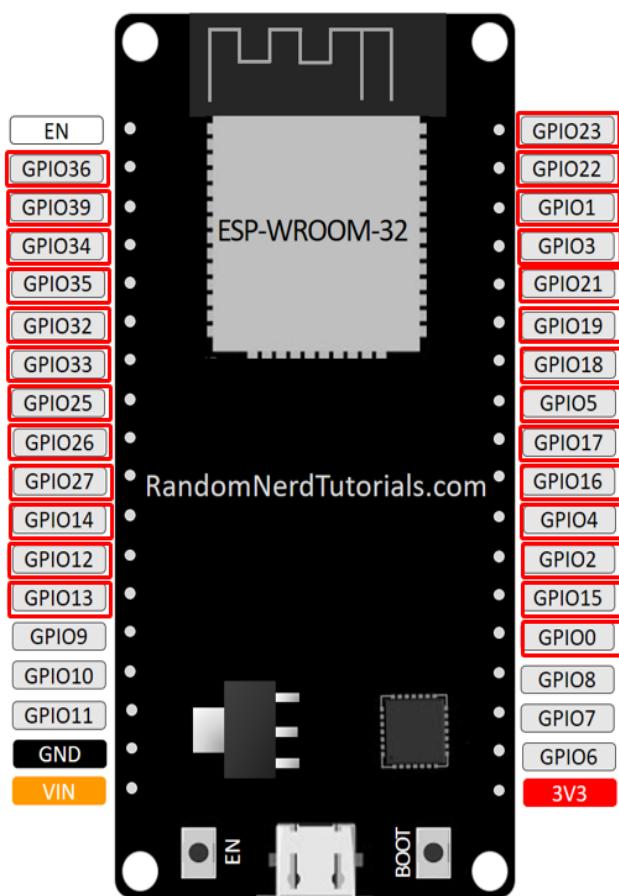
```
attachInterrupt(digitalPinToInterruption(GPIO), function,  
mode);
```

- **GPIO Interrupt**

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

```
digitalPinToInterrupt(27)
```

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



- **Fungsi yang akan dipicu**

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

- **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.

- **Timer**

Dalam contoh ini juga akan memperkenalkan timer. Maka dari itu ingin LED tetap menyala selama beberapa detik yang telah ditentukan setelah gerakan terdeteksi. Alih-alih menggunakan fungsi delay() yang memblokir kode yang telah diinput dan tidak memungkinkan untuk melakukan hal lain selama beberapa detik yang ditentukan, kita harus menggunakan timer.

- **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.

- **The Millis Function**

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

`millis()`

Mengapa fungsi itu berguna? Karena dengan menggunakan beberapa matematika, Anda dapat dengan mudah memverifikasi berapa banyak waktu yang telah berlalu tanpa memblokir kode kita.

- **Blinking an LED with millis()**

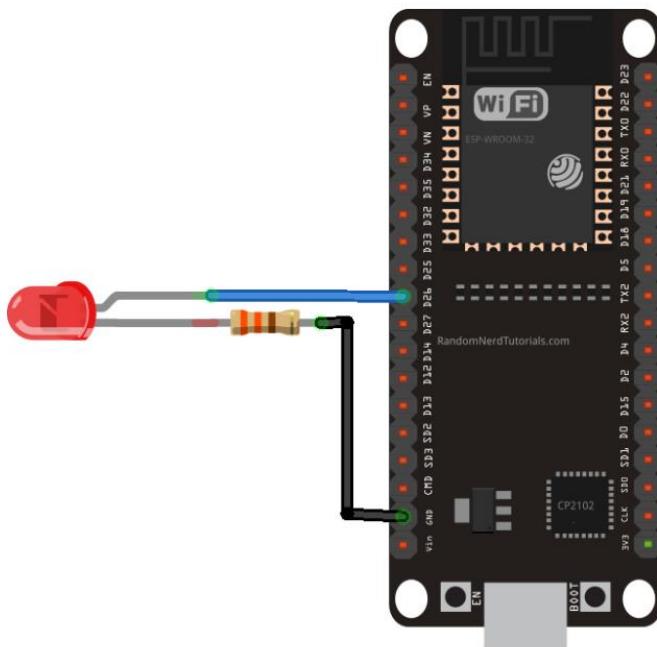
Cuplikan kode berikut menunjukkan bagaimana Anda dapat menggunakan fungsi millis() untuk membuat proyek LED berkedip. Itu menyalakan LED selama 1000 milidetik, dan kemudian mematikannya.

- **Kode Pemograman**

```
*****  
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 upda  
  
// constants won't change :  
const long interval = 1000; // interval at which to blink (millisec  
  
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);  
    }  
}
```

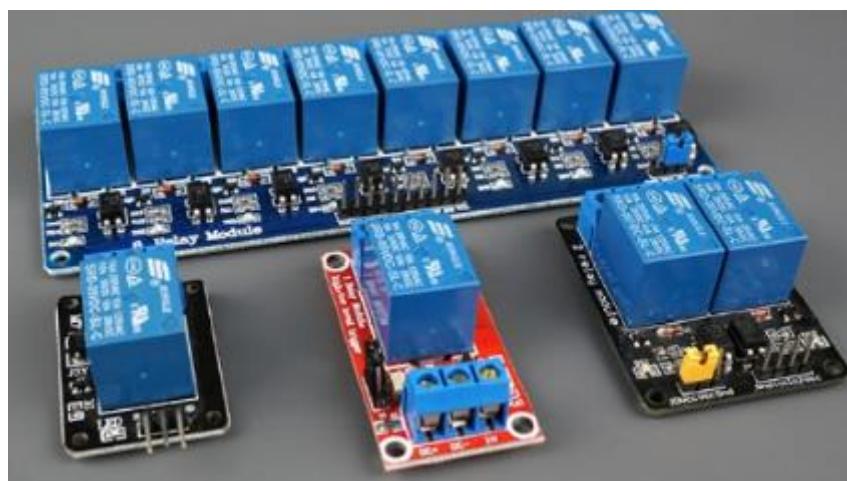
Kita sekarang seharusnya dapat memahami bahwa menambahkan tugas lain ke fungsi loop() dan kode Anda akan tetap mengedipkan LED setiap satu detik.

Kita dapat mengunggah kode ini ke ESP32 kita dan merakit diagram skematik berikut untuk mengujinya dan mengubah jumlah milidetik untuk melihat cara kerjanya.



7. ESP32 Relay

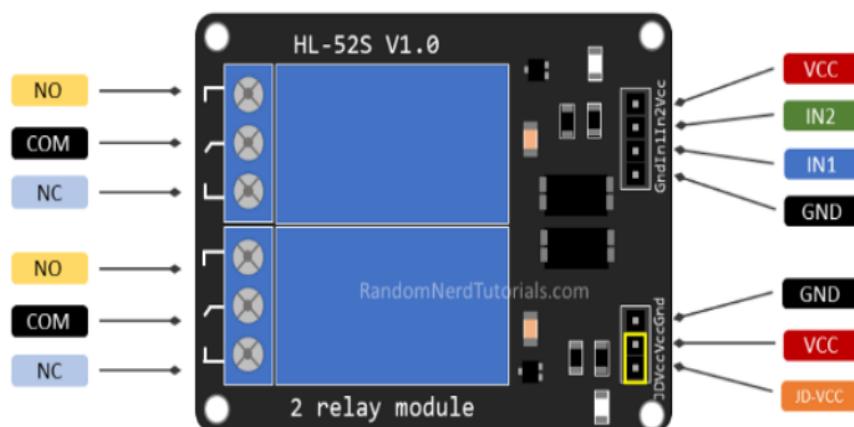
Relay adalah output yang dapat digunakan sebagai switch atau saklar untuk perangkat lain. Relay dikontrol dengan tegangan dari pin Arduino sehingga dapat melakukan switch. Terdapat 3 koneksi utama yaitu COM untuk input dari perangkat lain. NC(Normally Close) pada keadaan biasa com akan terhubung ke pin 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.



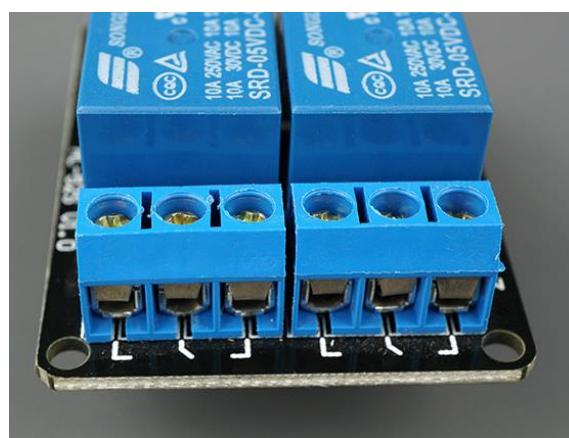
Ada modul relai yang berbeda dengan jumlah saluran yang berbeda. Anda dapat menemukan modul relai dengan satu, dua, empat, delapan, dan bahkan enam belas saluran. Jumlah saluran menentukan jumlah keluaran yang dapat kami kendalikan. Ada modul relai yang elektromagnetnya dapat ditenagai oleh 5V dan dengan 3.3V. Keduanya dapat digunakan dengan ESP32 – Anda dapat menggunakan pin VIN (yang menyediakan 5V) atau pin 3.3V.

- **Relay Pinout**

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**



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**

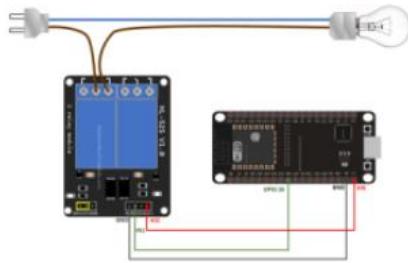


Sisi tegangan rendah memiliki satu set empat pin dan satu set tiga pin. Set pertama terdiri dari VCC dan GND untuk menyalakan modul, dan input 1 (IN1) dan input 2 (IN2) untuk mengontrol relai bawah dan atas, masing-masing. Jika modul relai Anda hanya memiliki satu saluran, Anda hanya akan memiliki satu 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

- **Wiring a Relay Modul to the ESP32**

Hubungkan modul relai ke ESP32 seperti yang ditunjukkan pada diagram berikut. Diagram menunjukkan pengkabelan untuk modul relai 2 saluran, pengkabelan dengan jumlah saluran yang berbeda serupa. Atau, Anda dapat menggunakan sumber daya 12V untuk mengontrol peralatan 12V.



Dalam contoh ini, kami mengendalikan lampu. Kami hanya ingin menyalakan lampu sesekali, jadi lebih baik menggunakan konfigurasi yang biasanya terbuka. Kami menghubungkan pin IN1 ke GPIO 26, Anda dapat menggunakan GPIO lain yang sesuai. Lihat Panduan Referensi GPIO ESP32.

- **Kode Pemograman**

```
const int relay = 26;

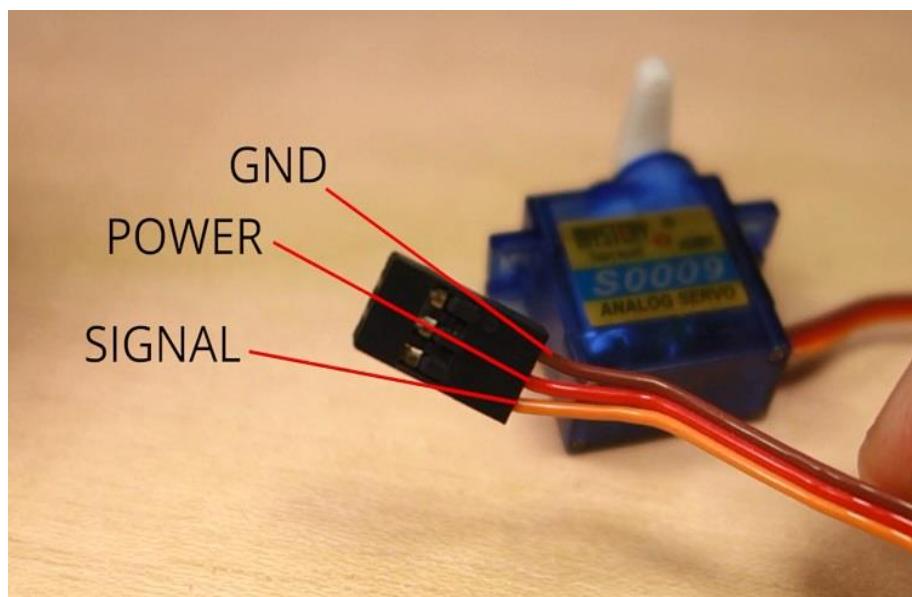
void setup() {
  Serial.begin(115200);
  pinMode(relay, OUTPUT);
}

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);
}
```

8. ESP32 SERVO

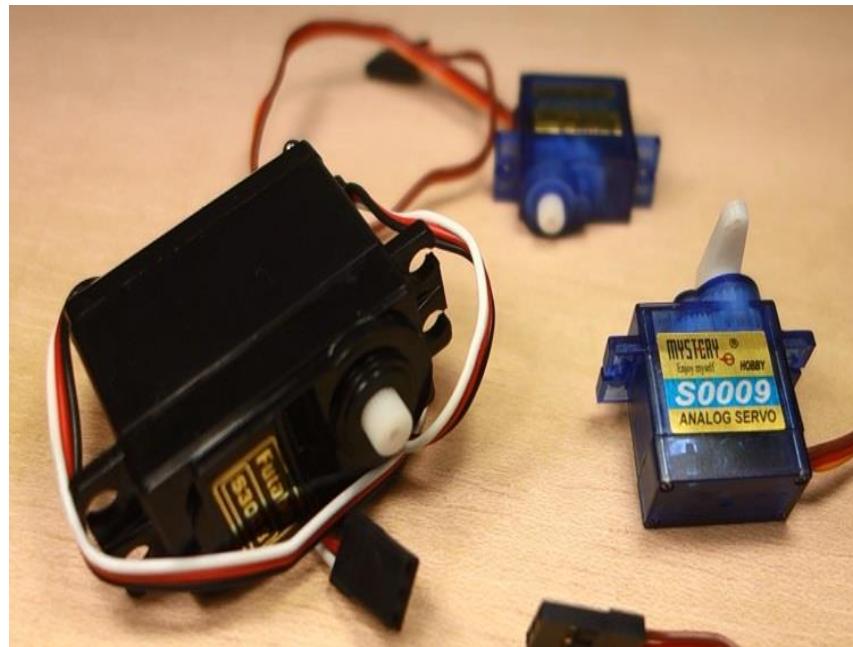
Servo adalah salah satu motor DC yang dimana memiliki kekurangan dan kelebihan dibanding motor DC yang lain. Untuk kekurangan motor dc servo yaitu hanya bisa bergerak $0^\circ - 180^\circ$ yang dimana hal ini sangat minim untuk kebutuhan yang relatif luas. Untuk keuntungan motor servo sendiri yaitu dia 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.



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



Tetapi jika kita menggunakan lebih dari satu servo atau jenis lainnya, mungkin perlu menyalakan servo kita menggunakan catu daya eksternal.



Saat menggunakan servo kecil seperti S0009 seperti yang ditunjukkan pada gambar di bawah, Anda dapat menyalakannya langsung dari ESP32. Jika Anda menggunakan servo kecil seperti S0009, Anda perlu menghubungkan:

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

- **Installing the ESP32 Arduino Servo Library**

- Klik di sini untuk mengunduh [ESP32_Arduino_Servo_Library](#). Anda harus memiliki folder .zip di folder Unduhan Anda
- Buka zip folder .zip dan Anda akan mendapatkan folder ESP32-Arduino-Servo-Library-Master
- Ganti nama folder Anda dari ESP32-Arduino-Servo-Library-Master menjadi ESP32_Arduino_Servo_Library
- Pindahkan folder ESP32_Arduino_Servo_Library ke folder library instalasi Arduino IDE anda
- Terakhir, buka kembali IDE Arduino Anda

- **Kode Pemrograman**

```
#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
}

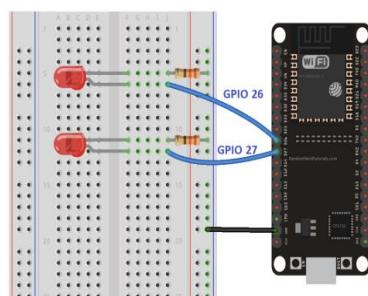
void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos); // tell servo to go to position in variable
    'pos'
    delay(15); // waits 15ms for the servo to reach the
    position
  }
  for (pos = 180; pos >= 0; pos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

9. ESP32 Output WebServer

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.

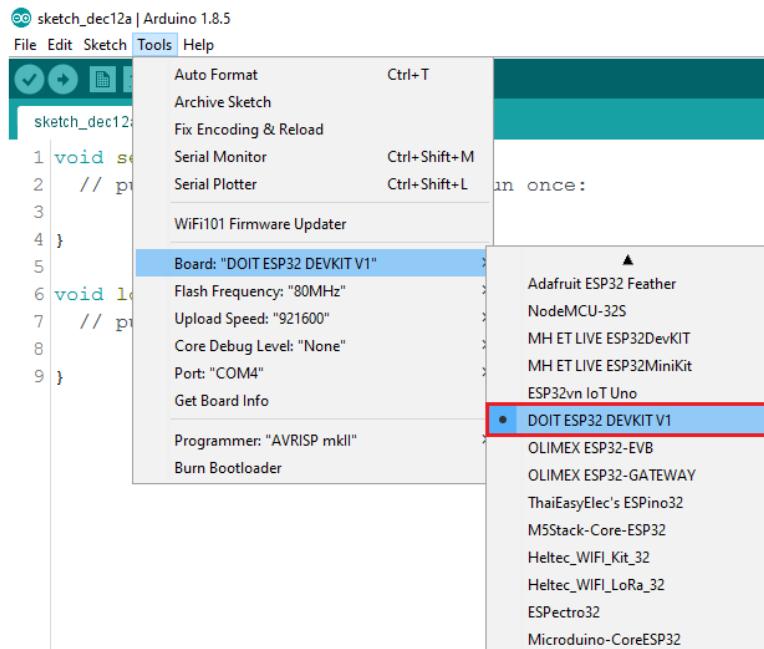
Dapat dijelaskan secara ringkas tahapannya seperti berikut:

- 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.

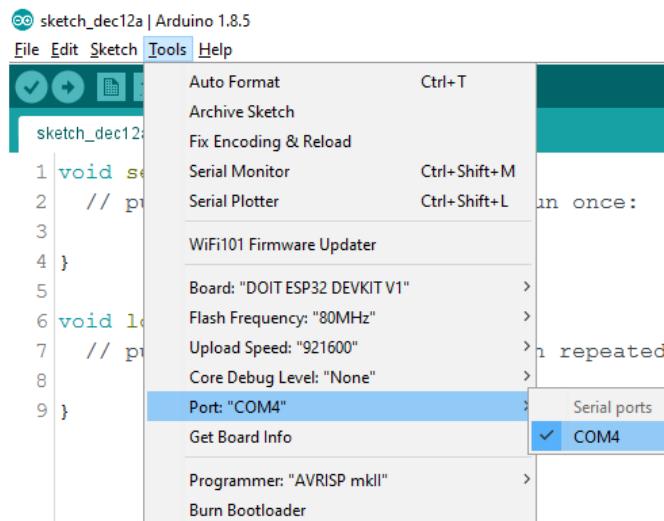


- **Mengunggah Kode**

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



3. Pilih port COM di Tools > Port.



4. Tekan tombol Unggah di Arduino IDE dan tunggu beberapa detik saat kode dikompilasi dan diunggah ke papan Anda.
5. Tunggu pesan "Selesai mengunggah".

```

D:\Lathan_9
Writing at 0x00001000... (84 %)
Writing at 0x00050000... (89 %)
Writing at 0x00054000... (94 %)
Writing at 0x00058000... (100 %)
Wrote 481440 bytes (299651 compressed) at 0x00010000 in 4.7 seconds
Hash of data verified.
Compressed 3072 bytes to 122...

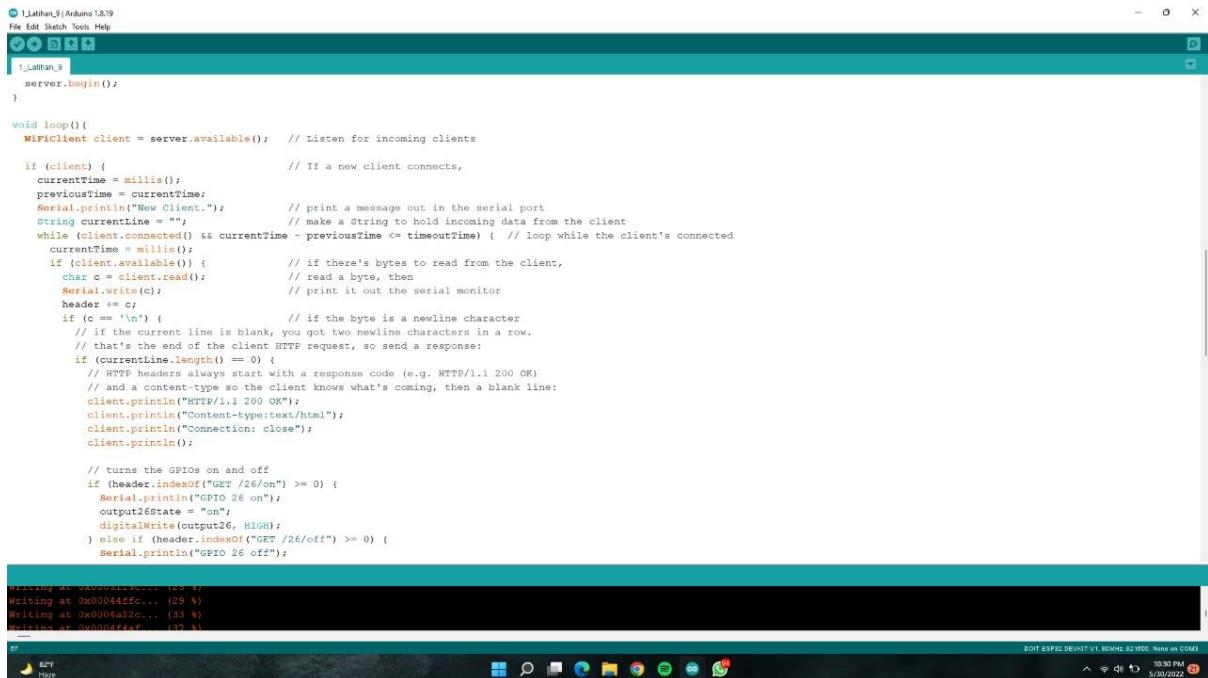
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (122 compressed) at 0x00008000 in 0.0 seconds
Hash of data verified.

Leaving...
Hard resetting...

```

DOIT ESP32 DEVKIT V1, 80MHz, 021600, None on COM4

• Kode Pemrograman



```

1_Lathan_9 | Arduino 1.8.19
File Edit Sketch Tools Help
1_Lathan_9
server.begin();
}

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

    if (client) {
        currentTime = millis();
        previousTime = currentTime;
        Serial.println("New Client.");
        String currentLine = "";
        while (client.connected() && currentTime - previousTime < timeoutTime) { // loop while the client's connected
            currentTime = millis();
            if (client.available()) {
                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 newlines 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 GPIO 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");
                    }
                    // turns the GPIO 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");
                    }
                }
                previousTime = currentTime;
            }
        }
        client.println("<p><a href=\"/26/on\"><button class=\"button\">ON</button></a></p>");
        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>");

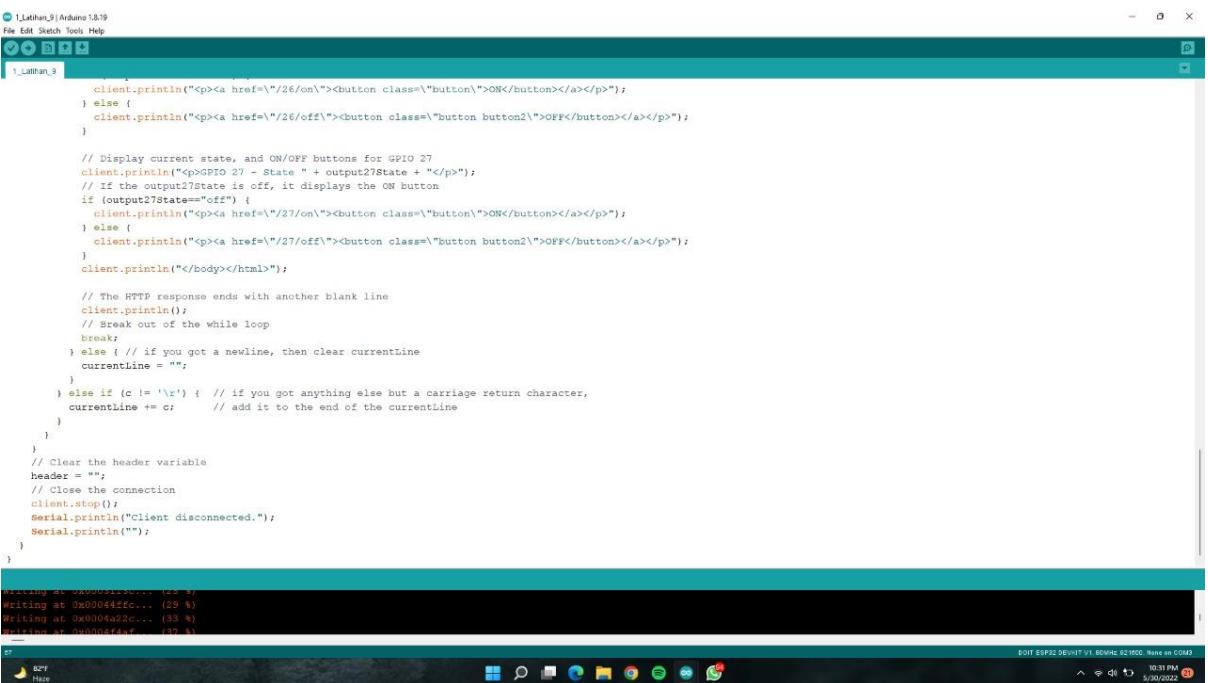
    // The HTTP response ends with another blank line
    client.println();
    // Break out of the while loop
    break;
} else { // If you got a newline, then clear currentLine
    currentLine = "";
}
} else if (c != '\r') { // If you got anything else but a carriage return character,
    currentLine += 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("");
}

Writing at 0x00001000... (85 %)
Writing at 0x00044ffcc... (29 %)
Writing at 0x00004a22c... (33 %)
Writing at 0x00004afaf... (37 %)

```

DOIT ESP32 DEVKIT V1, 80MHz, 021600, None on COM4



```

1_Lathan_9 | Arduino 1.8.19
File Edit Sketch Tools Help
1_Lathan_9
    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("");
}

Writing at 0x00001000... (85 %)
Writing at 0x00044ffcc... (29 %)
Writing at 0x00004a22c... (33 %)
Writing at 0x00004afaf... (37 %)

```

DOIT ESP32 DEVKIT V1, 80MHz, 021600, None on COM4

```

1_Lathan_3
    output26State = "off";
    digitalWrite(output26, LOW);
    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>");
    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("body{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>");
    }
}

```

```

Writing at 0x000422cc... (29 %)
Writing at 0x00044ffcc... (29 %)
Writing at 0x0004a22cc... (33 %)
Writing at 0x0004df4af... (37 %)
—
ef B2F Help
1_Lathan_3 | Arduino 1.8.19
File Edit Sketch Tools Help
1_Lathan_3
    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>"); —
    }
}

// The HTTP response ends with another carriage return
client.println();
// Break out of the while loop
break;
} else { // If you get 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("");
}

Writing at 0x000422cc... (29 %)
Writing at 0x00044ffcc... (29 %)
Writing at 0x0004a22cc... (33 %)
Writing at 0x0004df4af... (37 %)
—
ef B2F Help
1_Lathan_3 | Arduino 1.8.19
File Edit Sketch Tools Help
1_Lathan_3

```

ESP32 Web Server

GPIO 26 - State off

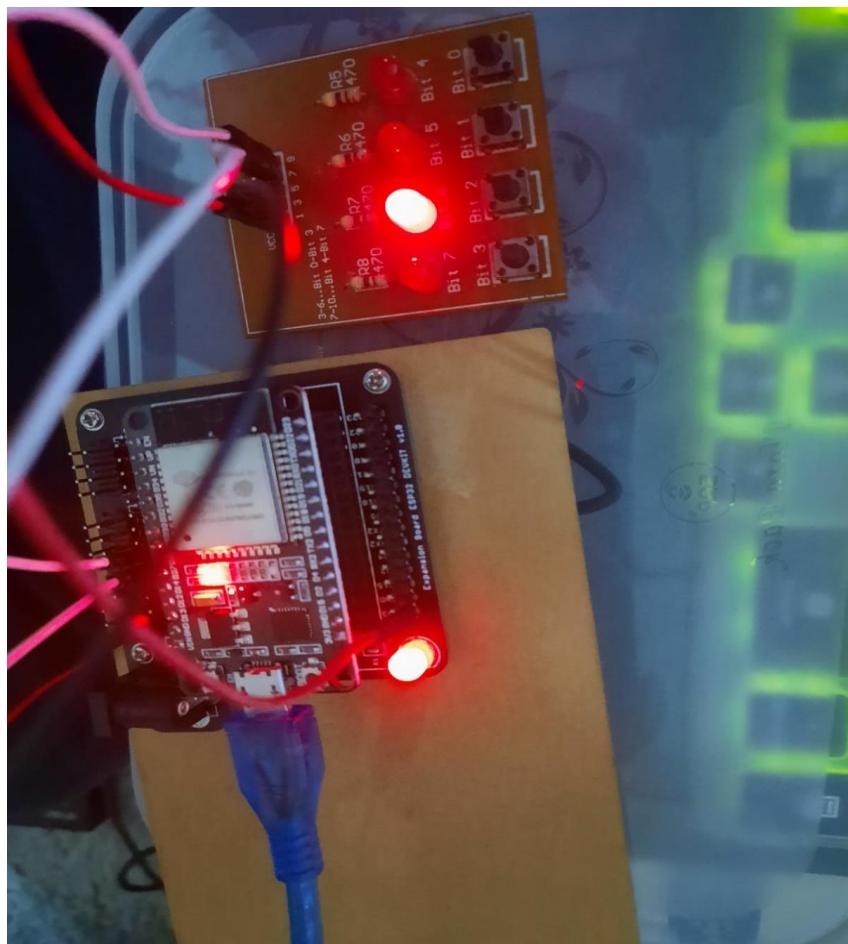
ON

GPIO 27 - State on

OFF



- Hasil Percobaan



10. ESP32 WebServer With Slider

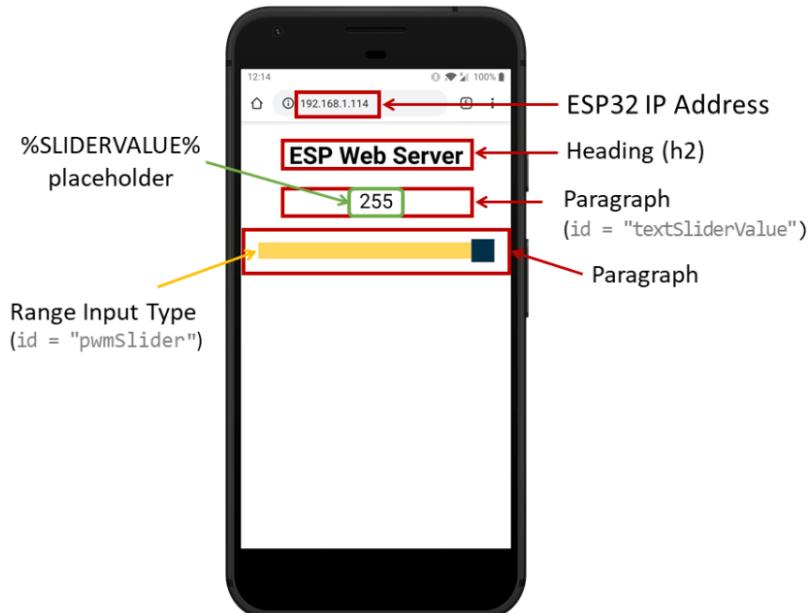
Webserver with Slider merupakan sistem pemrograman web server ESP32 menggunakan Arduino IDE, bertujuan untuk melakukan kontrol kecerahan pada lampu LED dengan cara menggeser. Variabel yang digunakan merupakan ESP32 yang nantinya terdapat suatu program yang dapat mengontrol sinyal, program tersebut dapat 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.



Penjelasan singkat :

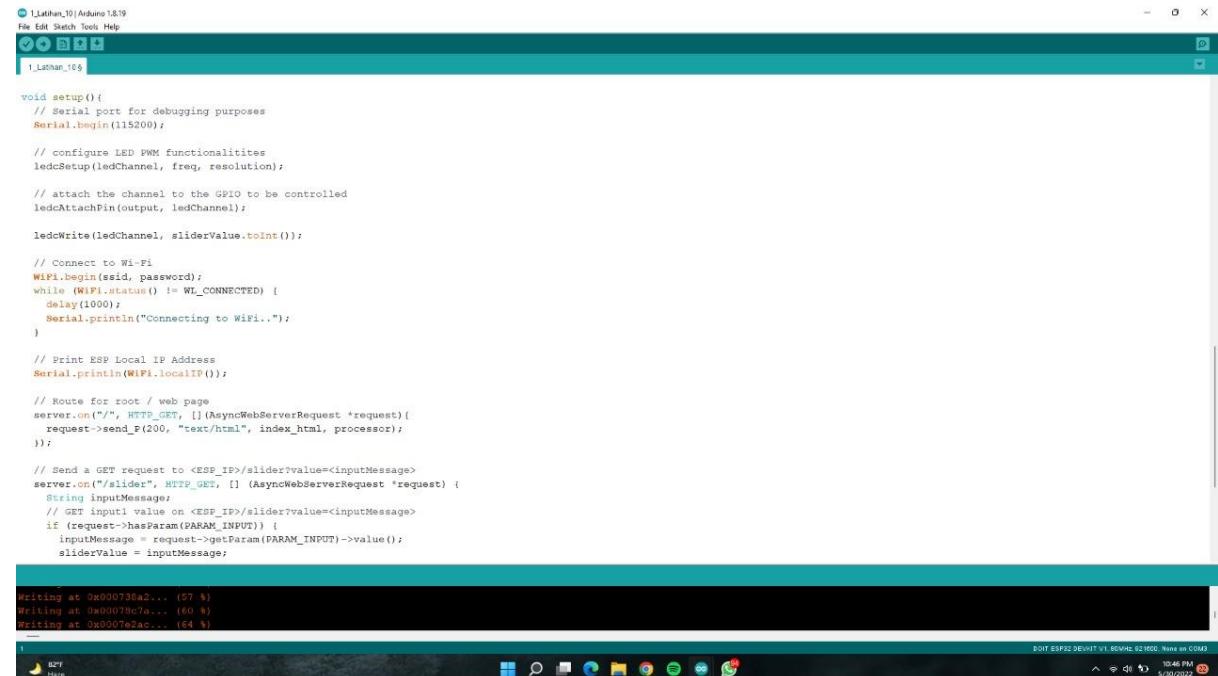
- ESP32 melakukan hosting web server yang menampilkan halaman web dengan penggeser yang nantinya berguna untuk mengatur tingkat kecerahan cahaya.
- Dengan melakukan perpindahan terhadap penggeser, permintaan HTTP diajukan kepada ESP32 dengan menggunakan nilai penggeser yang baru.
- Permintaan pada HTTP 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.

- **Building the Web Page**



Halaman web untuk proyek ini cukup sederhana. Ini berisi satu judul, satu paragraf dan satu input dari rentang jenis.

• Kode Pemrograman



```
1_Latihan_10|Arduino 1.8.19
File Edit Sketch Tools Help
1_Latihan_10
void setup() {
    // Serial port for debugging purposes
    Serial.begin(115200);

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

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

    ledcWrite(ledChannel, sliderValue.toInt());

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

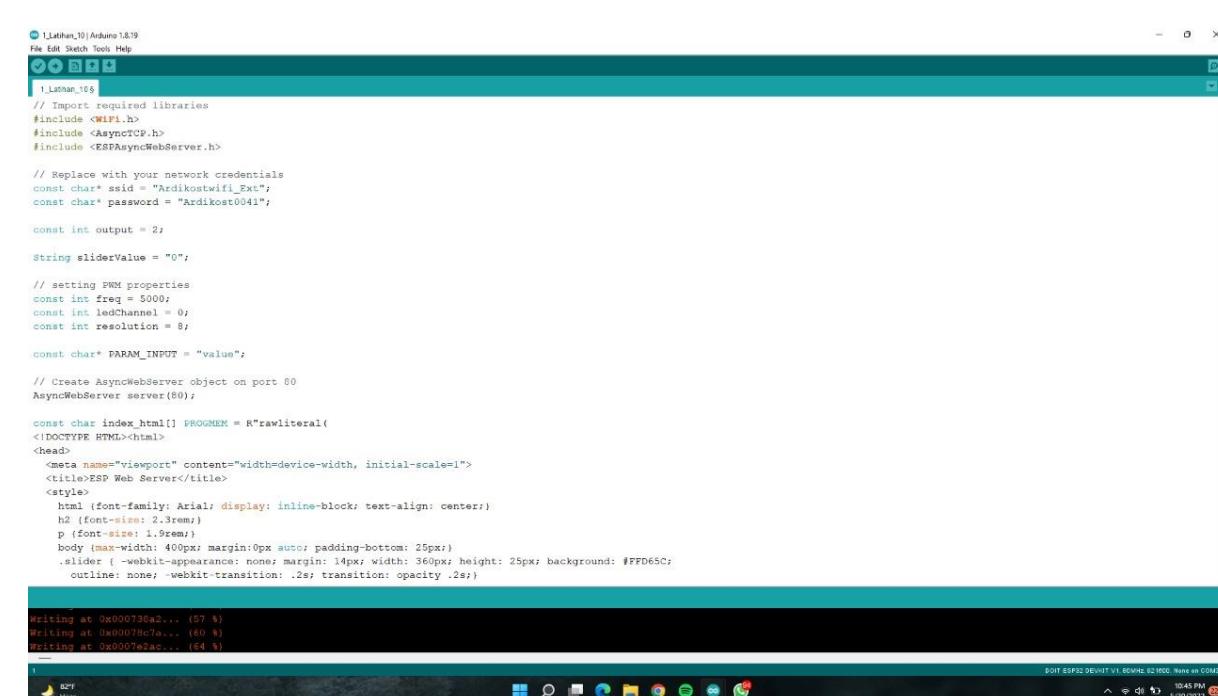
    // Print ESP Local IP Address
    Serial.println(WiFi.localIP());

    // 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>/slider?value=<inputMessage>
    server.on("/slider", HTTP_GET, [] (AsyncWebServerRequest *request) {
        String inputMessage;
        // GET input value on <ESP_IP>/slider?value=<inputMessage>
        if (request->hasParam(PARAM_INPUT)) {
            inputMessage = request->getParam(PARAM_INPUT)->value();
            sliderValue = inputMessage;
        }
    });
}

Writing at 0x0000730a2... (57 %)
Writing at 0x000078e7a... (60 %)
Writing at 0x00007e2ac... (64 %)

1
ESP32 DevKit V1.8MHz 82MHz Nano on COM3
^ % # 10:46 PM 5/30/2022
```



```
1_Latihan_10|Arduino 1.8.19
File Edit Sketch Tools Help
1_Latihan_10
// Import required libraries
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>

// Replace with your network credentials
const char* ssid = "Ardikostwifif_Ext";
const char* password = "Ardikost0041";

const int output = 2;

String sliderValue = "0";

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

const char* PARAM_INPUT = "value";

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

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>ESP Web Server</title>
    <style>
        html {font-family: Arial; display: inline-block; text-align: center;}
        h2 {font-size: 2.3rem;}
        p {font-size: 1.9rem;}
        body {max-width: 400px; margin:0px auto; padding-bottom: 25px;}
        .slider { -webkit-appearance: none; margin: 14px; width: 360px; height: 25px; background: #FFD65C;
        outline: none; -webkit-transition: .2s; transition: opacity .2s; }
</style>
)</head>
<body>
    <h1>ESP Web Server</h1>
    <p>Move the slider below and see the value here</p>
    <input type="range" min="0" max="100" value="0" oninput="this.value = Math.floor(this.value)" />
    <p>Value: <span id="value">0</span></p>
</body>
)rawliteral";

server.serveStatic("/", "/index.html");

server.on("/slider", HTTP_POST, [] (AsyncWebServerRequest *request) {
    String inputMessage;
    // POST input value on /slider
    if (request->hasParam(PARAM_INPUT)) {
        inputMessage = request->getParam(PARAM_INPUT)->value();
        sliderValue = inputMessage;
    }
});

server.onNotFound([] (AsyncWebServerRequest *request) {
    request->send(index_html);
});

server.begin();

Writing at 0x0000730a2... (57 %)
Writing at 0x000078e7a... (60 %)
Writing at 0x00007e2ac... (64 %)

1
ESP32 DevKit V1.8MHz 82MHz Nano on COM3
^ % # 10:46 PM 5/30/2022
```

```
1_Lathan_10 | Arduino 1.8.19
File Edit Sketch Tools Help
1_Lathan_10
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>/slider?value<inputMessage>
server.on("/slider", HTTP_GET, [] (AsyncWebServerRequest *request) {
    String inputMessage;
    // GET input value on <ESP_IP>/slider?value<inputMessage>
    if (request->hasParam(PARAM_INPUT)) {
        inputMessage = request->getParam(PARAM_INPUT)->value();
        sliderValue = inputMessage;
        ledcWrite(channel, sliderValue.toInt());
    }
    else {
        inputMessage = "No message sent";
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
});

// Start server
server.begin();

void loop() {
}

Writing at 0x0000738a2... (57 %)
Writing at 0x000078c7a... (60 %)
Writing at 0x00007e2ac... (64 %)
—
1 82% Haze 10:46 PM 5/30/2022
```

```
1_Lathan_10 | Arduino 1.8.19
File Edit Sketch Tools Help
1_Lathan_10
h2 {font-size: 2.3rem;}
p {font-size: 1.9rem;}
body {max-width: 400px; margin:0px auto; padding-bottom: 25px;}
.slider { -webkit-appearance: none; margin: 14px; width: 360px; height: 25px; background: #FFD65C;
outline: none; -webkit-transition: .2s; transition: opacity .2s;}
.slider::-webkit-slider-thumb { -webkit-appearance: none; appearance: none; width: 35px; height: 35px; background: #003249; cursor: pointer;}
.slider::-moz-range-thumb { width: 35px; height: 35px; background: #003249; cursor: pointer; }
</style>
</head>
<body>
<h2>ESP Web Server</h2>
<p><span id="textSliderValue">%SLIDERVALUE%</span></p>
<p><input type="range" onchange="updateSliderPWM(this)" id="pwmSlider" min="0" max="255" value="%SLIDERVALUE%" step="1" class="slider"></p>
<script>
function updateSliderPWM(element) {
    var sliderValue = document.getElementById("pwmSlider").value;
    document.getElementById("textSliderValue").innerHTML = sliderValue;
    console.log(sliderValue);
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "/slider?value="+sliderValue, true);
    xhr.send();
}
</script>
</body>
</html>
)rawliteral";

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

Writing at 0x0000738a2... (57 %)
Writing at 0x000078c7a... (60 %)
Writing at 0x00007e2ac... (64 %)
—
1 82% Haze 10:46 PM 5/30/2022
```

1_Latihan_10 | Arduino 1.8.19

File Edit Sketch Tools Help

1_Latihan_10

```

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=>send_P(200, "text/html", index_html);
});

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

// Start server
server.begin();
}

void loop() {
}

Writing at 0x000730a2... (57 %)
Writing at 0x00079c7a... (60 %)
Writing at 0x0007e2ac... (64 %)

```

COMB

Connecting to WiFi..
ets Jun 8 2016 00:22:57
rst:0xl (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x40078000, len:12812
load:0x40080400, len:3032
entry 0x400805e4
Connecting to WiFi..
192.168.0.174

Newline 115200 baud Clear output

ARDUINO IDE

10:47 PM 5/30/2022

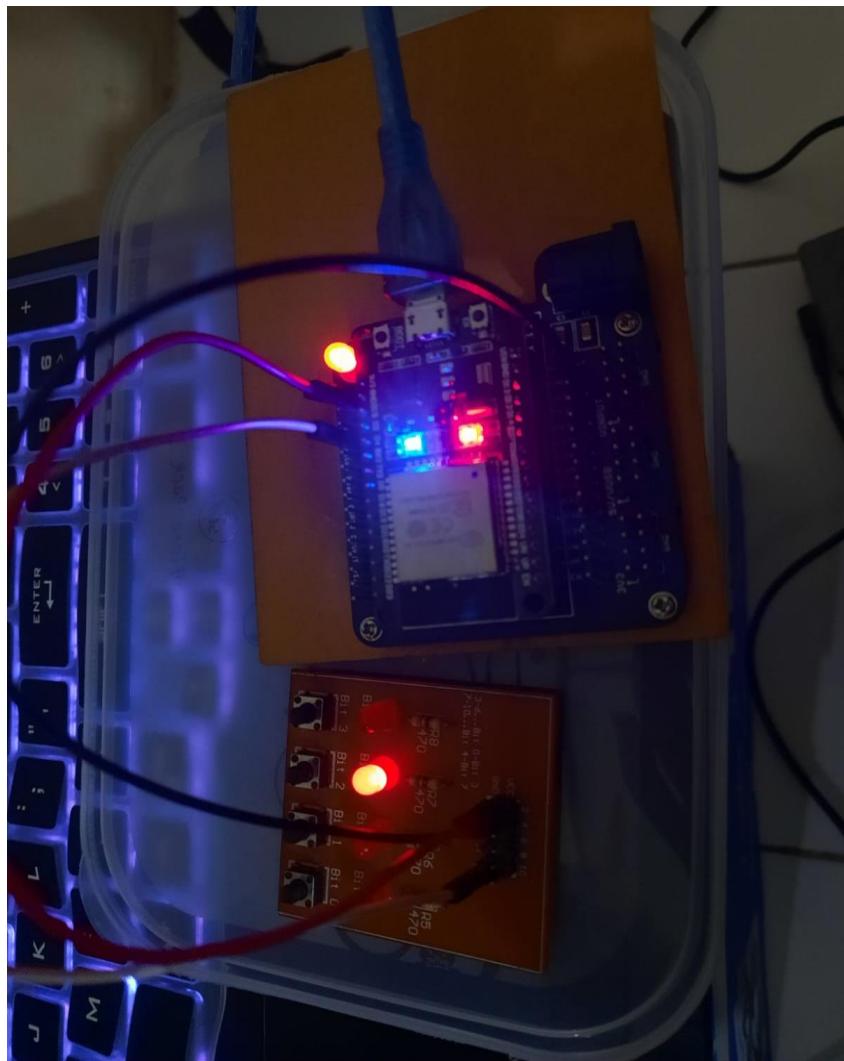


ESP Web Server

199



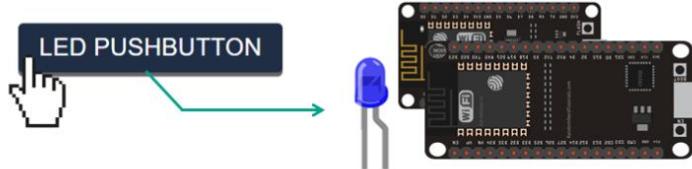
- **Hasil Percobaan**



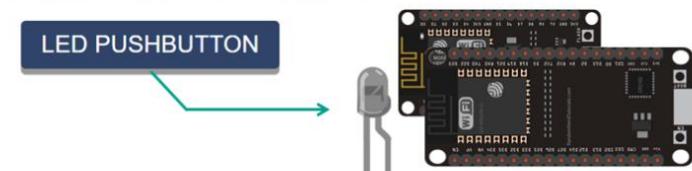
11. ESP32 Momentary Switch WebServer

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.

ESP Pushbutton Web Server



ESP Pushbutton Web Server



- ESP32 atau ESP8266 menghosting server web yang dapat Anda akses untuk mengontrol output;
- Status default output adalah LOW, tetapi Anda dapat mengubahnya tergantung pada aplikasi proyek Anda;
- Ada tombol yang bertindak seperti sakelar sesaat;
- Jika Anda menekan tombol, output mengubah statusnya menjadi TINGGI selama Anda terus menahan tombol;
- Setelah tombol dilepaskan, status output kembali ke LOW.

• Kode Pemrograman

The screenshot shows the Arduino IDE interface with the code for the ESP Pushbutton Web Server. The code includes CSS for a button style, an HTML structure with a button, and JavaScript logic to handle the button's state via XMLHttpRequest. The Arduino board is set to AsyncWebServer at port 80.

```
1_Latihan_11 | Arduino 1.8.19
File Edit Sketch Tools Help
1_Latihan_11


```

<head>
 <style>
 -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);
 </style>
</head>
<body>
 <h1>ESP Pushbutton Web Server</h1>
 <button class="button" onclick="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;

```


voidNotFound(AsyncWebserverRequest request) {
  request->send(404, "text/plain", "Not found");
}

AsyncWebServer server(80);


```

Writing at 0x0006e0... (53 %)
Writing at 0x0007397e... (57 %)
Writing at 0x00078c37... (60 %)
...

EDIT ESP32 DEBUTT V1.60MHz 821600 Nano on COM3
10:51 PM
5/30/2022

```
1.Lathan_11|Arduino 1.8.19
File Edit Sketch Tools Help
1.Lathan_11
1

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");
  });
}

// Writing at 0x000065f4... (53 %)
// Writing at 0x0007397e... (57 %)
// Writing at 0x00070c97... (60 %)

10:53 PM 5/30/2022
```

```
1.Lathan_11|Arduino 1.8.19
File Edit Sketch Tools Help
1.Lathan_11
1

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

// REPLACE WITH YOUR NETWORK CREDENTIALS
const char* ssid = "Ardikostwifi_Enc";
const char* password = "Ardikost0041";

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;
    }
  </style>
</head>
<body>
  <h1>ESP Pushbutton Web Server</h1>
  <button class="button" onclick="buttonClicked()>Push Me</button>
</body>
</html>
)rawliteral";

// Writing at 0x000065f6... (53 %)
// Writing at 0x0007397e... (57 %)
// Writing at 0x00070c97... (60 %)

10:53 PM 5/30/2022
```

```
1.Lathan_11| Arduino 1.8.19
File Edit Sketch Tools Help

1.Lathan_11
  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(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() {
}


```

Writing at 0x0006e5f6... (53 %)
Writing at 0x0007397e... (57 %)
Writing at 0x00078c37... (60 %)

DOIT ESP32 DEVKIT V1. 80MHz 82MHz. None on COM3
10:53 PM 5/30/2022

```
1.Lathan_11| Arduino 1.8.19
File Edit Sketch Tools Help

1.Lathan_11
  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(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() {
}


```

ets Jun 8 2016 00:22:57
rst:0xl (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
config:ip: 0, SWIP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x40078000, len:12812
load:0x40080400, len:3032
entry 0x400005e4
ESP IP Address: http://192.168.0.174

COM3

Autoscroll Show timestamp
Newline 115200 baud Clear output

Writing at 0x0006e5f6... (53 %)
Writing at 0x0007397e... (57 %)
Writing at 0x00078c37... (60 %)

DOIT ESP32 DEVKIT V1. 80MHz 82MHz. None on COM3
10:54 PM 5/30/2022

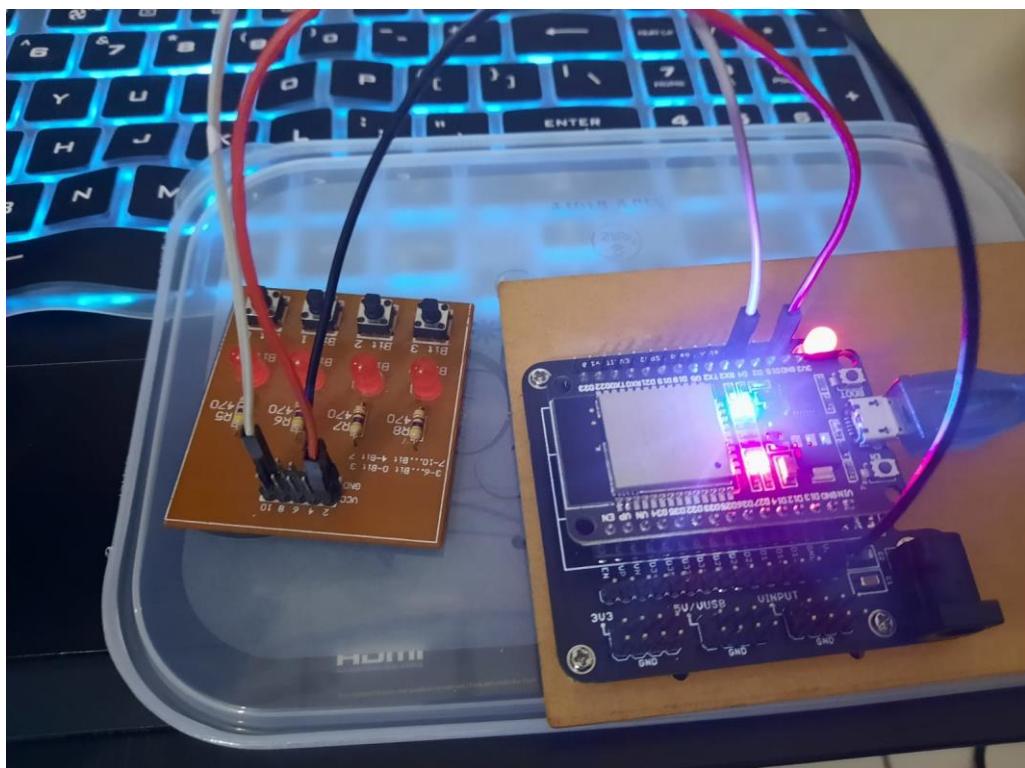


ESP Pushbutton Web Server

LED PUSHBUTTON

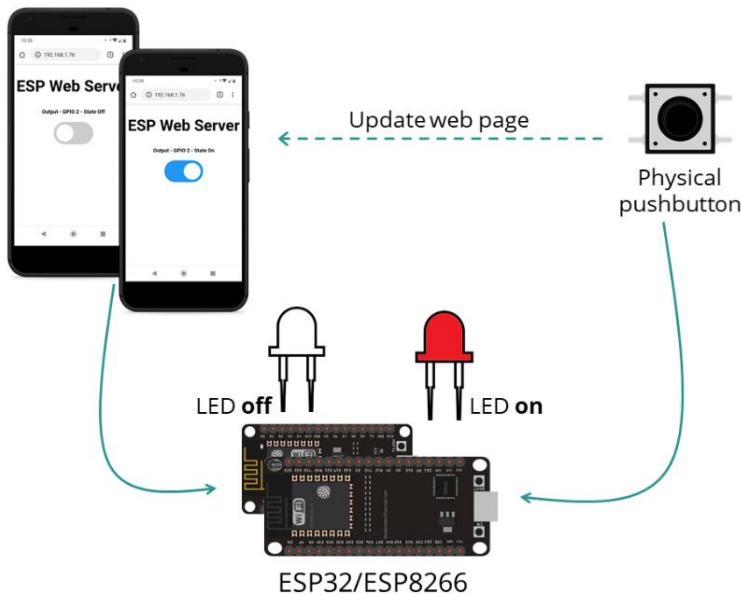


- **Hasil Percobaan**



12. ESP32 Physical Button WebServer

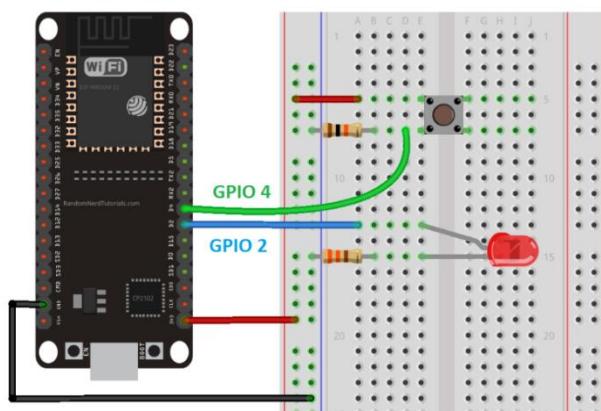
Physical Button Webserver yaitu merupakan mengontrol output pada ESP32 atau ESP8266, dengan menggunakan web server dan tombol secara langsung. Sehingga output yang dihasilkan pada laman web yang terbaru dapat terdeteksi nantinya apakah terdapat perubahan yang dilakukan pada web server atau tombol secara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



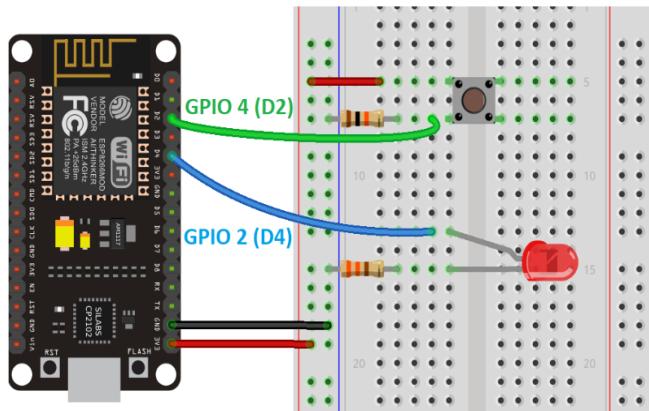
- ESP32 atau ESP8266 melakukan hosting web server yang memungkinkan untuk user untuk mengontrol status secara keluaran.
- Status yang sudah didapatkan saat ini ditampilkan pada web server.
- ESP nantinya juga terhubung ke tombol secara langsung yang dapat melakukan kontrol pada output yang sama.
- Sehingga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

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

ESP32 Schematic



ESP8266 NodeMCU Schematic



- **Schematic Diagram**

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

Berikut daftar bagian yang Kami perlukan untuk membangun sirkuit:

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

- **Kode Pemrograman**

```

1.Lahan_12 | Arduino 1.8.19
File Edit Sketch Tools Help
1.Lahan_12
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= "<h4>Output - State <span id="outputState">"</span></h4><label class="switch"><input type="checkbox" onchange="toggleCheckbox(this)" id="output" " + outputSta
  return buttons;
}

Writing at 0x000063c07... (46 %)
Writing at 0x00006927... (50 %)
Writing at 0x0000658d... (53 %)

```

```
1_Lathan_12|Arduino 1.8.19
File Edit Sketch Tools Help
1_Lathan_12
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;
return buttons;
}

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());
}

Writing at 0x00063c87... (46 %)
Writing at 0x0006927c... (50 %)
Writing at 0x0006e58d... (53 %)

7
80% Here
10:02 PM 5/30/2022
```

```
1_Lathan_12|Arduino 1.8.19
File Edit Sketch Tools Help
1_Lathan_12
}

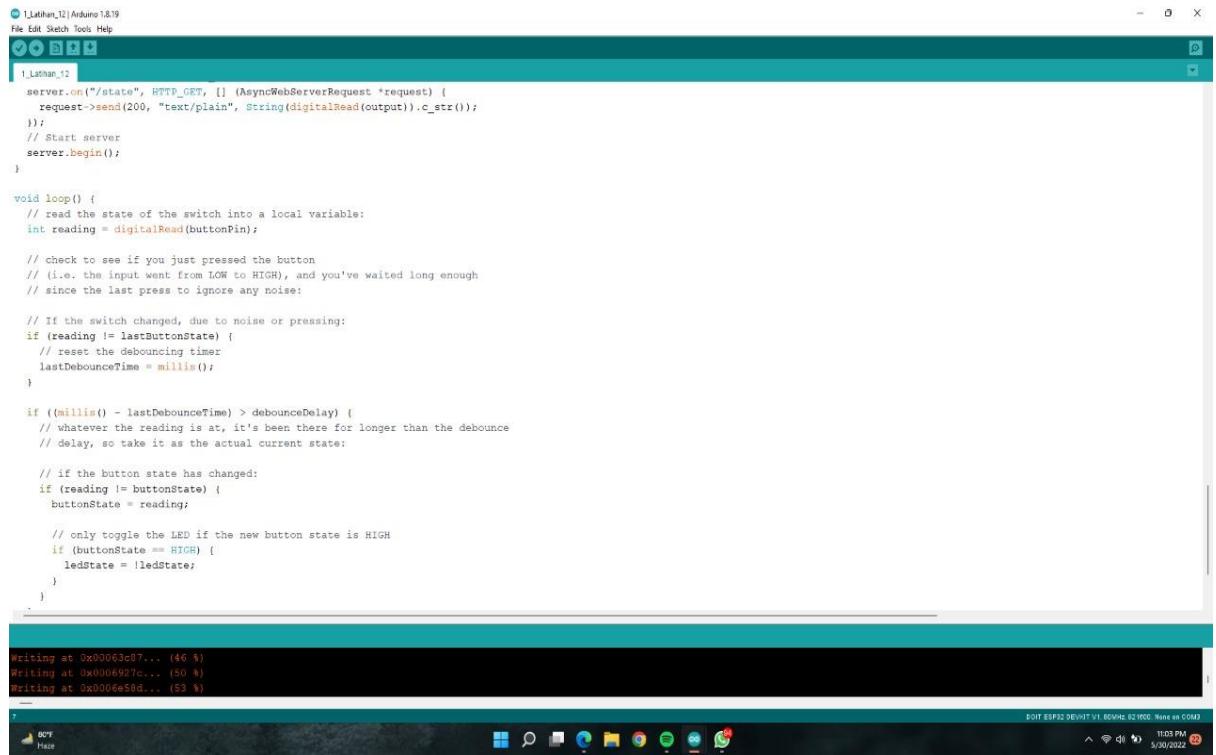
// Print ESP Local IP Address
Serial.println(WiFi.localIP());

// 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 input 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
.

Writing at 0x00063c07... (46 %)
Writing at 0x0006927c... (50 %)
Writing at 0x0006e58d... (53 %)

7
80% Here
10:02 PM 5/30/2022
```



```
1.Lathan_12 | Arduino 1.8.19
File Edit Sketch Tools Help
1.Lathan_12
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:

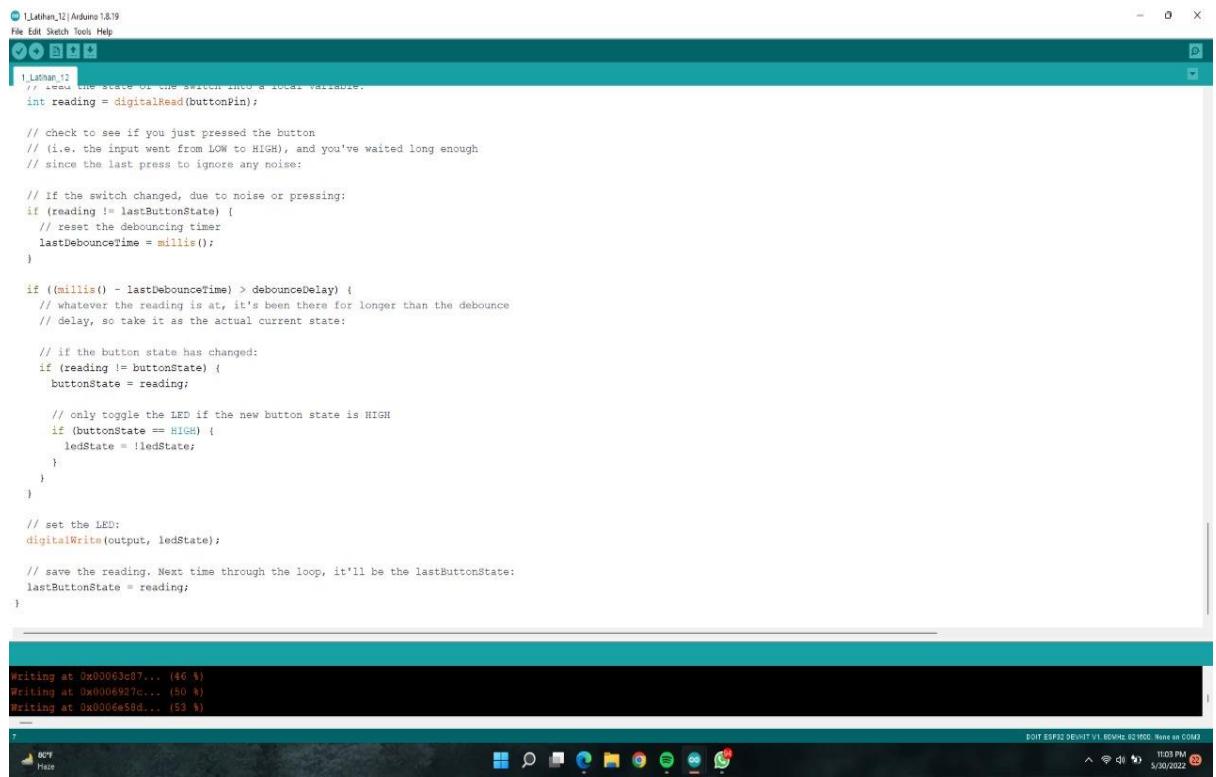
        // if the button state has changed:
        if (reading != buttonState) {
            buttonState = reading;

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

    #writing at 0x00063c07... (46 %)
#writing at 0x0006927c... (50 %)
#writing at 0x0006e58d... (53 %)

```

Writing at 0x00063c07... (46 %)
Writing at 0x0006927c... (50 %)
Writing at 0x0006e58d... (53 %)



```
1.Lathan_12 | Arduino 1.8.19
File Edit Sketch Tools Help
1.Lathan_12
// 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;

```

Writing at 0x00063c07... (46 %)
Writing at 0x0006927c... (50 %)
Writing at 0x0006e58d... (53 %)

```

1_Lathan_12 | Arduino 1.8.19
File Edit Sketch Tools Help
[ ] [ ] [ ] [ ] [ ] [ ]
1_Lathan_12

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: 3.0mm;}
    body {max-width: 600px; margin:0px auto; padding-bottom: 25px;}
    .switch {position: relative; display: inline-block; width: 120px; height: 60px;}
    .switch input {position: absolute; top: 0; left: 0; right: 0; bottom: 0; background-color: #ccc; border-radius: 34px;}
    .switch::before {position: absolute; content: ""; height: 52px; width: 52px; left: 0px; bottom: 0px; background-color: #fff; -webkit-transition: .4s; transition: .4s; border-radius: 60px; border: 2px solid #000; box-sizing: border-box; position: absolute; top: 0; left: 0; right: 0; bottom: 0; margin: auto; width: 100%; height: 100%; border-radius: 50%; transform: translateX(52px); -ms-transform: translateX(52px); transform: translateX(52px); z-index: 1; }
    input:checked+.switch::before {background-color: #2196F3; border: 2px solid #000; border-radius: 60px; width: 52px; height: 52px; left: 0px; bottom: 0px; margin: auto; position: absolute; top: 0; left: 0; right: 0; bottom: 0; width: 100%; height: 100%; border-radius: 50%; transform: translateX(52px); -ms-transform: translateX(52px); transform: translateX(52px); z-index: 1; }
    .switch::after {position: absolute; content: " "; width: 100%; height: 100%; background-color: #fff; border-radius: 34px; border: 2px solid #000; border-radius: 60px; border: 2px solid #000; border-radius: 50%; width: 52px; height: 52px; left: 0px; bottom: 0px; margin: auto; position: absolute; top: 0; left: 0; right: 0; bottom: 0; width: 100%; height: 100%; border-radius: 50%; transform: translateX(52px); -ms-transform: translateX(52px); transform: translateX(52px); z-index: 1; }
    .switch input:checked::before {background-color: #2196F3; border: 2px solid #000; border-radius: 60px; width: 52px; height: 52px; left: 0px; bottom: 0px; margin: auto; position: absolute; top: 0; left: 0; right: 0; bottom: 0; width: 100%; height: 100%; border-radius: 50%; transform: translateX(52px); -ms-transform: translateX(52px); transform: translateX(52px); z-index: 1; }
    .switch input:checked::after {background-color: #fff; border: 2px solid #000; border-radius: 60px; width: 52px; height: 52px; left: 0px; bottom: 0px; margin: auto; position: absolute; top: 0; left: 0; right: 0; bottom: 0; width: 100%; height: 100%; border-radius: 50%; transform: translateX(52px); -ms-transform: translateX(52px); transform: translateX(52px); z-index: 1; }
</style>
<body>
<h2>ESP Web Server</h2>
<#BUTTONHOLDER>
<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;
            ...
        }
    }
}

Writing at 0x00063c07... (46 $)
Writing at 0x0006927c... (50 $)
Writing at 0x0006@58d... (53 $)

7
BCF
Haze
[ ] [ ] [ ] [ ] [ ] [ ]
DHT ESP8266 DEVKIT V1.80MHz 821600. None on COM3
^ % & 11:01 PM
5/30/2022

```

```

1_Lathan_12 | Arduino 1.8.19
File Edit Sketch Tools Help
[ ] [ ] [ ] [ ] [ ] [ ]
1_Lathan_12

// 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
// since the last press to ignore any noise
if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
}

if ((millis() - lastDebounceTime) > debounceTime) {
    // whatever the reading is at, it's been there long enough to be real
    // delay, so take it as the actual current state
    if (reading != buttonState) {
        buttonState = reading;
    }
}

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

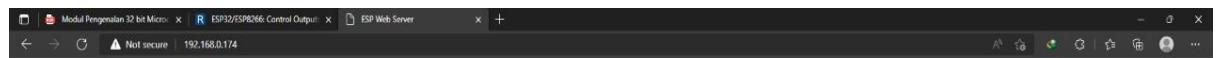
// set the LED:
digitalWrite(output, ledState);

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

Writing at 0x00063c07... (46 $)
Writing at 0x0006927c... (50 $)
Writing at 0x0006@58d... (53 $)

7
BCF
Haze
[ ] [ ] [ ] [ ] [ ] [ ]
DHT ESP8266 DEVKIT V1.80MHz 821600. None on COM3
^ % & 11:03 PM
5/30/2022

```

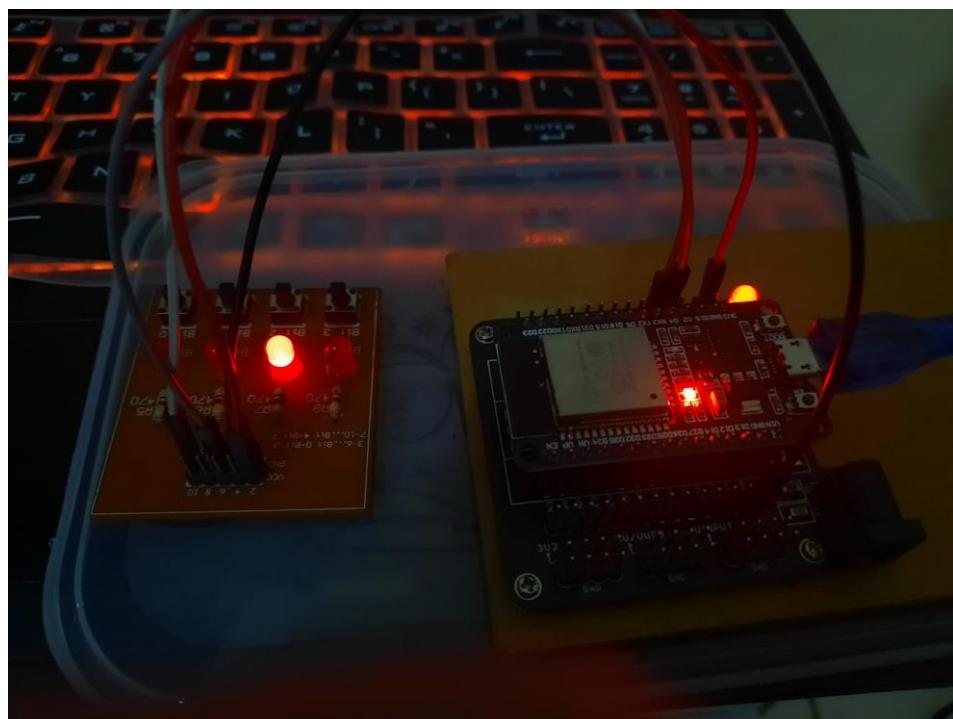


ESP Web Server

Output - GPIO 2 - State On

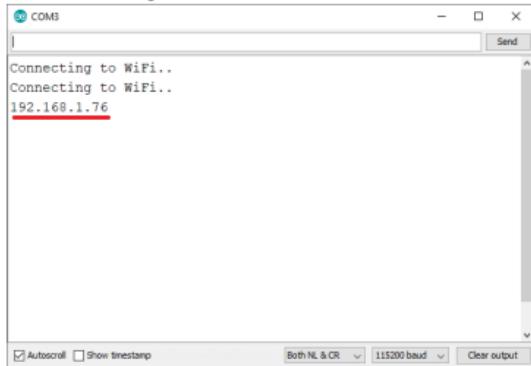


- **Hasil Percobaan**



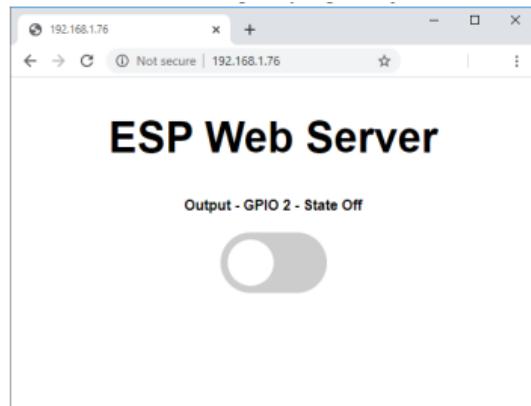
- **Demonstration**

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

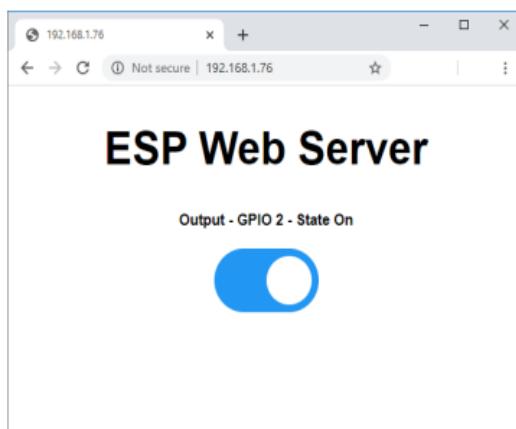


The screenshot shows a Windows-style COM port monitor window titled "COM3". The text area displays the following logs:
Connecting to WiFi..
Connecting to WiFi..
192.168.1.76

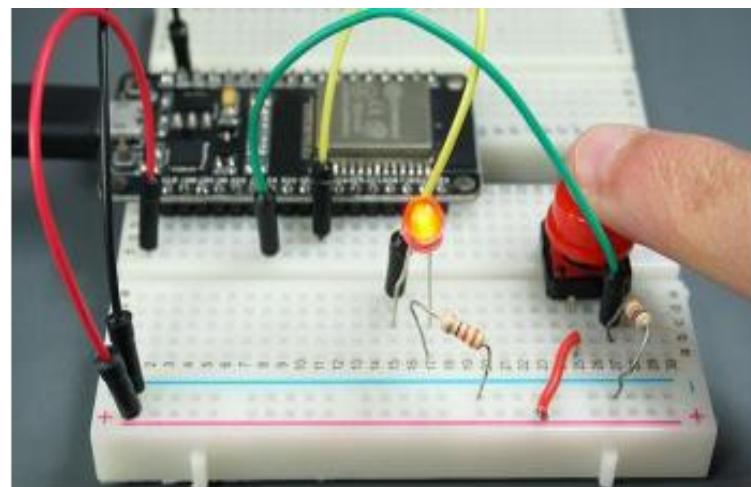
Buka browser di jaringan lokal, dan ketik alamat IP ESP. Kami harus memiliki akses ke server web seperti yang ditunjukkan di bawah ini.



Kami dapat mengaktifkan tombol di server web untuk menyalakan LED



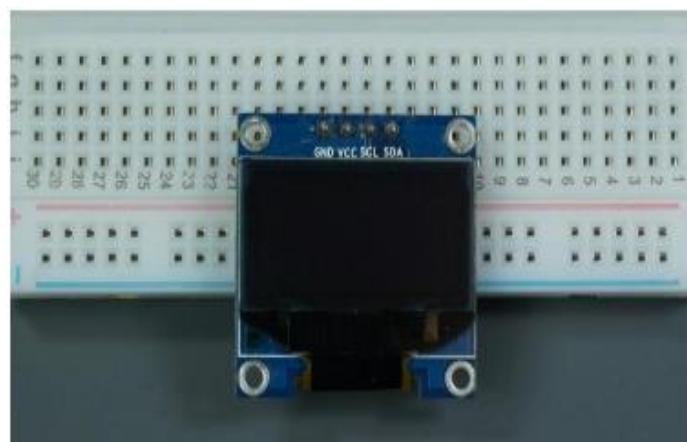
Kami juga dapat mengontrol LED yang sama dengan tombol tekan fisik. Statusnya akan selalu diperbarui secara otomatis di server web.



13.ESP32 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.

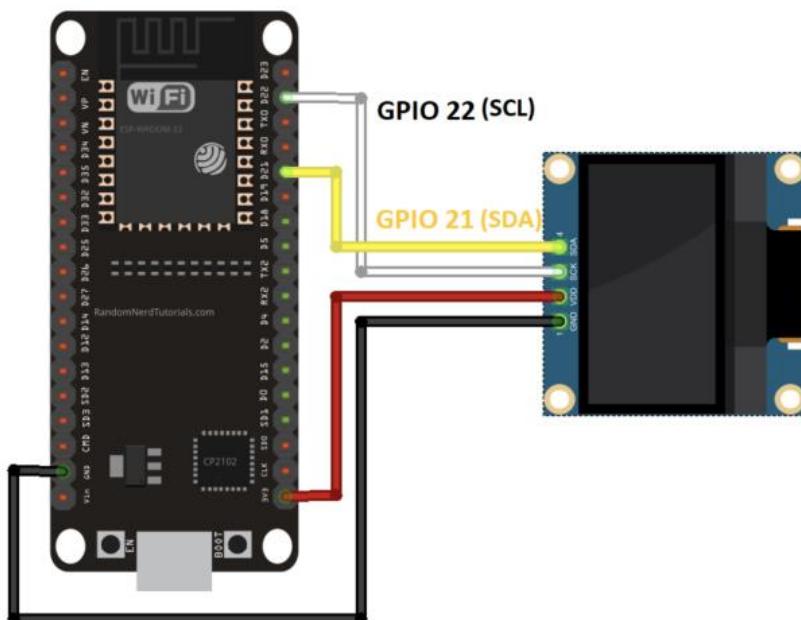


- **OLED Display SSD1306 Pin Wiring**

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

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

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

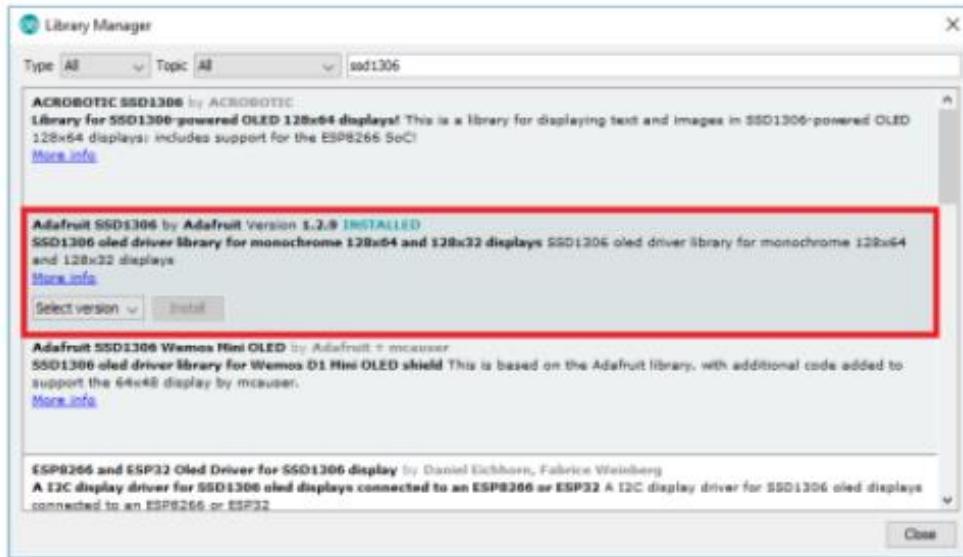


Dalam contoh ini, kami menggunakan protokol komunikasi I2C. Pin yang paling cocok untuk komunikasi I2C di ESP32 adalah GPIO 22 (SCL) dan GPIO 21 (SDA). Jika Kami menggunakan layar OLED dengan protokol komunikasi SPI, gunakan GPIO berikut.

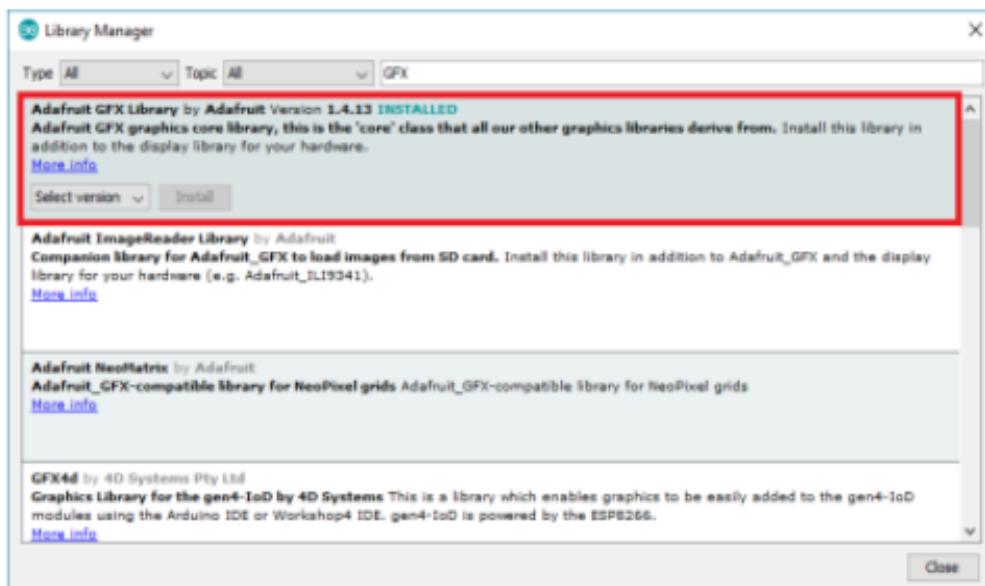
- 1) GPIO 18: CLK
- 2) GPIO 19: MISO
- 3) GPIO 23: MOSI
- 4) GPIO
- 5: CS

- **Installing SSD1306 OLED Library – ESP32**

1. Buka Arduino IDE Anda dan buka Sketsa > Sertakan Perpustakaan > Kelola Perpustakaan. Manajer Perpustakaan harus terbuka.
2. Ketik “SSD1306” di kotak pencarian dan instal perpustakaan SSD1306 dari Adafruit.



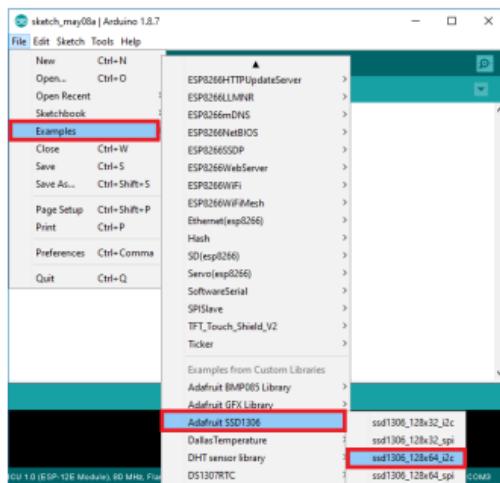
3. Setelah menginstal perpustakaan SSD1306 dari Adafruit, ketik "GFX" di kotak pencarian dan instal perpustakaan.



4. Setelah menginstal perpustakaan, restart Arduino IDE Anda.

- **Testing OLED Display with ESP32**

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



- **Draw Shapes in the OLED Display**

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

- a. **Draw a pixel**



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

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

b. Draw a line



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

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

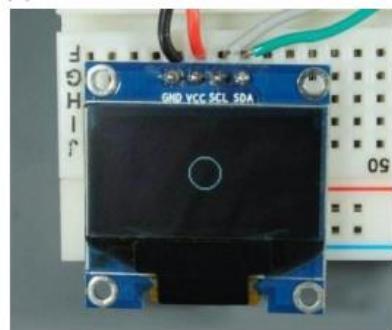
c. Draw a rectangle



`drawRect(x, y, width, height, color)` menyediakan cara mudah untuk menggambar persegi panjang. Koordinat (x,y) menunjukkan sudut kiri atas persegi panjang. Kemudian, tentukan lebar, tinggi dan warna: Gunakan `fillRect(x, y, width, height, color)` untuk menggambar persegi panjang yang terisi. Metode ini menerima argumen yang sama dengan `drawRect()`.

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

d. Draw a circle



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

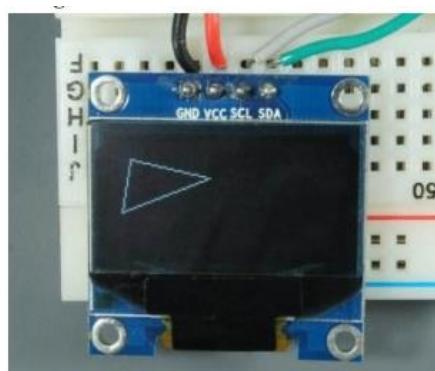
```
display.drawCircle(64, 32, 10, WHITE);
```

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

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



e. Draw a triangle



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

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

Gunakan metode fillTriangle() untuk menggambar segitiga yang terisi.

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

f. Invert Pustaka

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

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

g. Code – Draw Shapes

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

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

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

```
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

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

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

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

  // Clear the buffer
  display.clearDisplay();

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

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

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

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

```

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

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

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

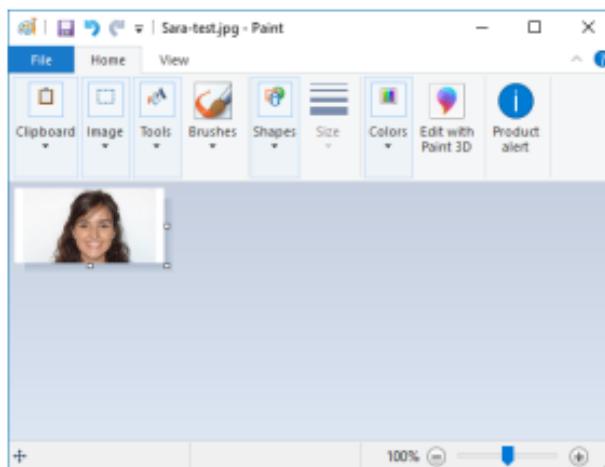
// Invert and restore display, pausing in-between
display.invertDisplay(true);
delay(3000);
display.invertDisplay(false);
delay(3000);

void loop() {

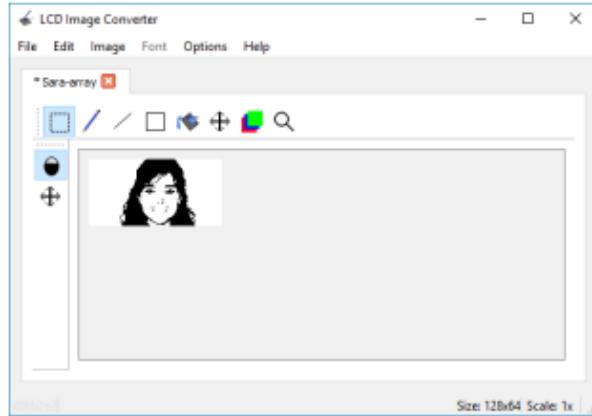
```

- **Display Bitmap Images in the OLED**

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

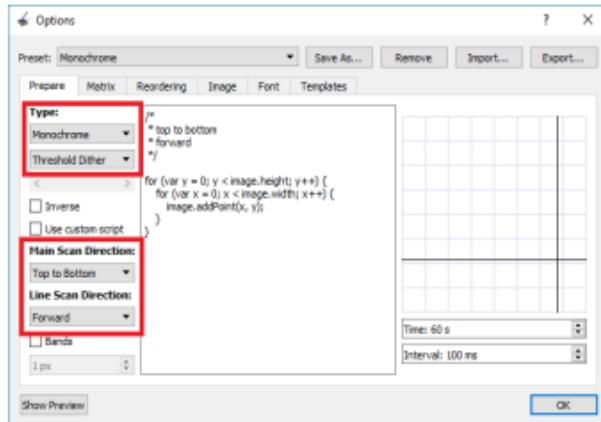


Kemudian, gunakan konverter Array Gambar ke C untuk mengubah gambar menjadi array. Saya telah menggunakan Pengonversi Gambar LCD. Jalankan program dan mulai dengan gambar baru. Buka Gambar > Impor dan pilih gambar bitmap yang telah dibuat sebelumnya.



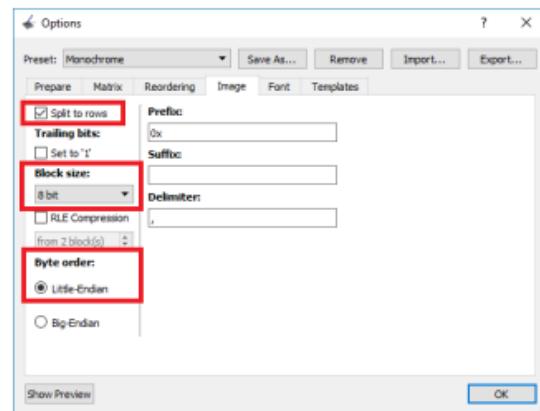
Buka Opsi > Konversi dan di tab Siapkan, pilih opsi berikut:

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



Buka tab Gambar dan pilih opsi berikut:

- 1) Pisahkan menjadi baris
- 2) Ukuran blok: 8 bit
- 3) Urutan byte: Little-Endian



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

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

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

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

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

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

```
    0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00,
0x00, 0x00, 0x03, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00,
0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00,
0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00,
0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00,
0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00,
0x00, 0x0a, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x14, 0x9e, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00,
0x36, 0x3f, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x6d, 0xff, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00,
0xfb, 0xff, 0x80, 0x1f, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x03,
0xd7, 0xff, 0x80, 0x0f, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x07,
0xef, 0xff, 0x80, 0x07, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x0f,
0xdf, 0xff, 0x90, 0x07, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x0f,
0xbf, 0xff, 0xd0, 0x07, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x1d,
0x7f, 0xff, 0xd0, 0x07, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xf0, 0x01, 0x1b,
0xff, 0xff, 0xc0, 0x07, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xf0, 0x02, 0xa7,
0xff, 0xff, 0xc0, 0x07, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x03,
0xff, 0xc0, 0x00, 0x07, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00,
0xff, 0x80, 0x00, 0x0b, 0xff, 0xff, 0xff, 0xff,
```

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

```

    0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x07, 0xff,
0xff, 0xbff, 0xf0, 0x00, 0x0f, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x07, 0xff,
0xff, 0xff, 0xe0, 0x00, 0x87, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x03, 0xff,
0xff, 0xff, 0xc0, 0x00, 0x43, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0x60, 0x00, 0x01, 0xff,
0xff, 0xff, 0xc0, 0x00, 0x73, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xfe, 0xe0, 0x00, 0x00, 0xff,
0xff, 0xff, 0x80, 0x00, 0x7b, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xfd, 0xe0, 0x00, 0x00, 0x7f,
0xff, 0xfe, 0x00, 0x00, 0x33, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xfd, 0xe0, 0x00, 0x00, 0x3f,
0xff, 0xf8, 0x00, 0x00, 0x27, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x0f,
0xff, 0xf0, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0x60, 0x00, 0x00, 0x67,
0xff, 0xe0, 0x00, 0x00, 0x1b, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xfd, 0x40, 0x00, 0x00, 0xf3,
0xff, 0xc4, 0x00, 0x00, 0x0b, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xfe, 0x80, 0x00, 0x00, 0xfc,
0xff, 0x8c, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x7f,
0x3c, 0x3c, 0x00, 0x00, 0x07, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x3f,
0xc0, 0x7c, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x1f,
0xff, 0xfc, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff
};

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

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

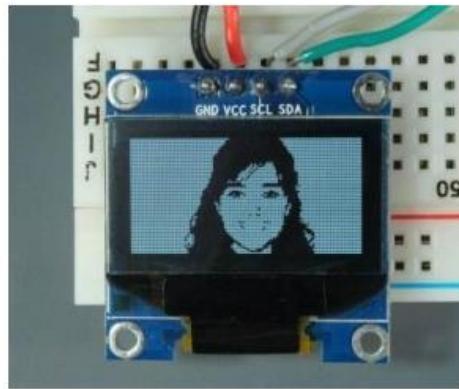
    // Clear the buffer.
    display.clearDisplay();

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

void loop() {
}

```

Setelah mengupload kode, inilah yang kita dapatkan di tampilan.



- **Kode Pemrograman**

```
sketch_may26 | Arduino IDE 1.8.13
File Edit Sketch Tools Help
src\main.cpp:26:
#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 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 SNOWFLAKES 10 // Number of snowflakes in the animation example

#define LOGO_WIDTH 16
#define LOGO_HEIGHT 16
static const unsigned char PROGMEM logo_tmpl = {
  $13300000, $11330000,
  $00000001, $01133000,
  $00000001, $11330000,
  $00000001, $01133000,
  $00000001, $21133000,
  $11110011, $21133000,
  $11111100, $21111000,
  $21111110, $21111111,
  $00110011, $00111111,
  $00011111, $01111100,
  $00011101, $01110000,
  $00111111, $01110000,
  $00111111, $02111000,
  $01111100, $21111000,
  $01110000, $02111000,
  $00000000, $00011000 };
```

The screenshot shows the Arduino IDE interface with the file `sketch_may24c.ino` open. The code is for the `SSD1306` library and includes examples for initializing the display, drawing pixels, lines, rectangles, circles, and triangles, as well as scroll text and bitmaps. It also demonstrates the `invertDisplay` function.

```
sketch_may24c | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_may24c
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
testdrawstylized();  // Draw 'stylized' characters
testscrolltext();    // Draw scrolling text
testdrawbitmap();    // Draw a small bitmap image

// Invert and restore display, pausing in-between
display.invertDisplay(true);
delay(10000);
display.invertDisplay(false);
delay(10000);

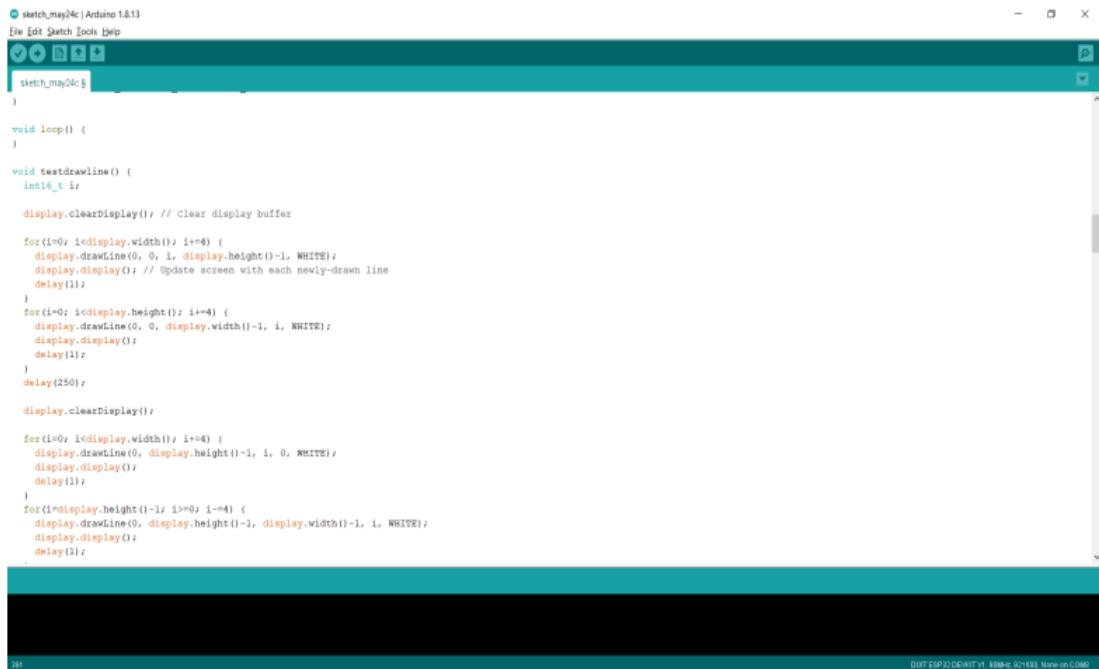
testanimate(logos_hmp, LOGO_WIDTH, LOGO_HEIGHT); // Animate bitmaps
```

The screenshot shows the Arduino IDE interface with the file `sketch_may24c.ino` open. The code is for the `SSD1306` library and includes examples for initializing the display, drawing pixels, lines, rectangles, circles, and triangles, as well as scroll text and bitmaps. It also demonstrates the `invertDisplay` function.

```
sketch_may24c | Arduino 1.8.1
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
testdrawstylized();  // Draw 'stylized' characters
testscrolltext();    // Draw scrolling text
testdrawbitmap();    // Draw a small bitmap image

// Invert and restore display, pausing in-between
display.invertDisplay(true);
delay(10000);
display.invertDisplay(false);
delay(10000);

testanimate(logos_hmp, LOGO_WIDTH, LOGO_HEIGHT); // Animate bitmaps
```



```

sketch_may24c | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_may24c.ino
}

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);

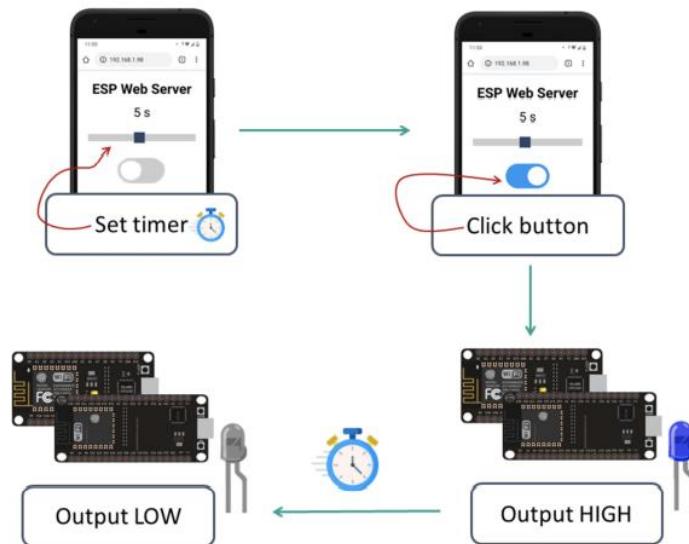
    display.clearDisplay();

    for(i=0; i<display.width(); i+=4) {
        display.drawLine(0, 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);
    }
}

```

14.ESP32 Timer/Pulse WebServer

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.



- 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.

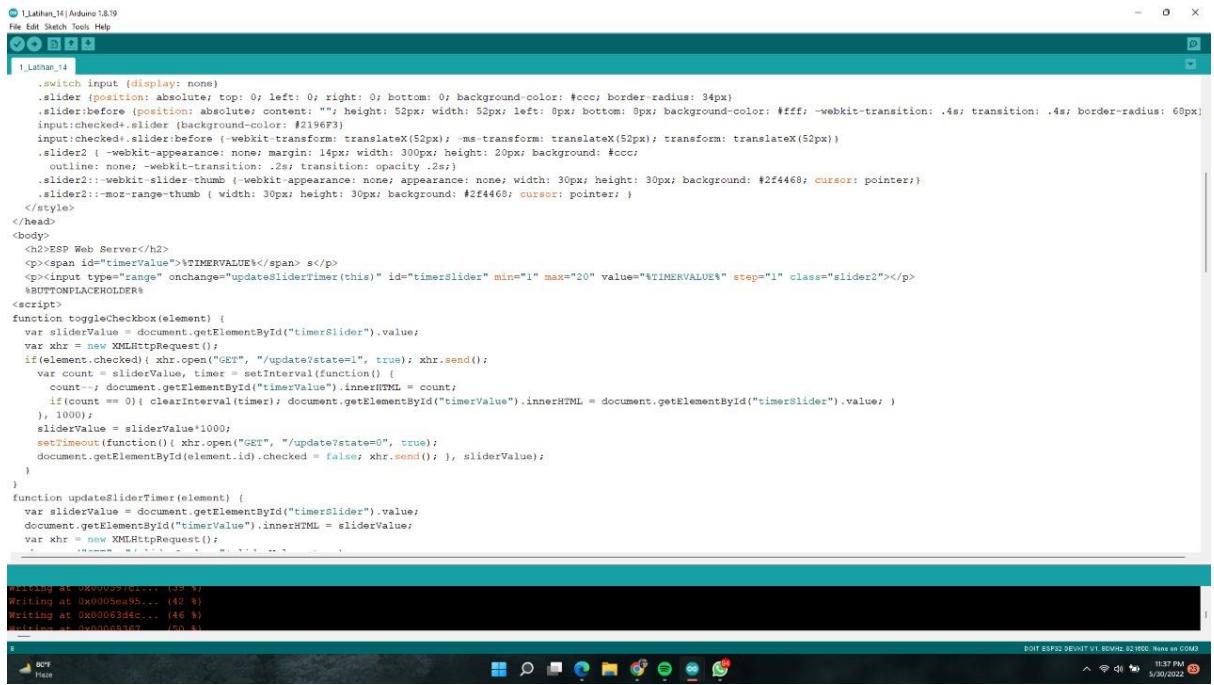
- **Installing Libraries – Async Web Server**

Untuk membangun server web, kami perlu menginstal pustaka berikut:

- 1) ESP32: instal pustaka ESPAsyncWebServer dan AsyncTCP.
- 2) ESP8266: instal pustaka ESPAsyncWebServer dan ESPAsyncTCP.

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

- **Kode Pemrograman**



```

1_Lahan_14 | Arduino 1.8.19
File Edit Sketch Tools Help
1_Lahan_14
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: 60px}
input:checked+slider::before {background-color: #2196F3}
input:checked+slider::before {transform: translateX(52px); -ms-transform: translateX(52px); transform: translateX(52px)}
.slider2 {background: none; margin: 14px; width: 300px; height: 20px; background: #ccc}
.outline: none; -webkit-transition: .2s; transition: opacity .2s}
.slider2::-webkit-slider-thumb {background: 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">%TIMEVALUE%</span> </p>
<p><input type="range" onchange="updateSliderTimer(this)" id="timerSlider" min="1" max="20" value="%TIMEVALUE%" step="1" class="slider2"></p>
<button>ON</button>
<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();
  ...
}

Writing at 0x00009780... (39 %)
Writing at 0x0000e95... (42 %)
Writing at 0x00063d4c... (46 %)
Writing at 0x0030e8362... (50 %)

```

```
1_Lathan_14 | Arduino 1.8.19
File Edit Sketch Tools Help

1_Lathan_14
function updateSliderTimer(element) {
    var sliderValue = document.getElementById("timerSlider").value;
    document.getElementById("timeValue").innerHTML = sliderValue;
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "/slider?value=" + sliderValue, true);
    xhr.send();
}
</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 += "<p><label class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"" + outputValue + "\" ><span class=\"slider\"></span></label></p>";
    return buttons;
}
else if(var == "TIMEVALUE"){
    return timerSliderValue;
}
return String();
}

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

void loop(){
}


```

Writing at 0x000057c1... (39 \$)
Writing at 0x0005e095... (42 \$)
Writing at 0x0006324c... (46 \$)
Writing at 0x00068367... (50 \$)

DUOT ESP8266 DEVIKIT V1.60MHz 82MHz. None on COM3
11:37 PM
5/30/2022

```
1_Lathan_14 | Arduino 1.8.19
File Edit Sketch Tools Help

1_Lathan_14
}

else if(var == "TIMEVALUE"){
    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..");
}

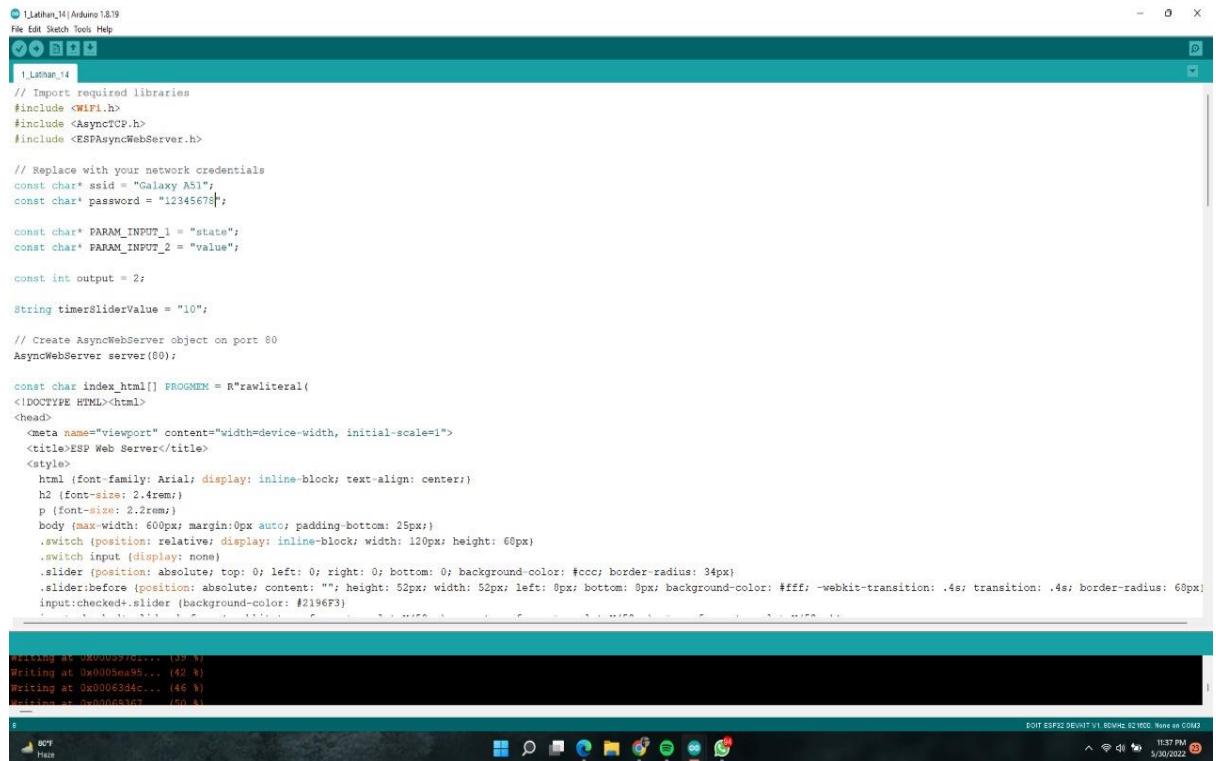
// Print ESP Local IP Address
Serial.println(WiFi.localIP());
}

void loop(){
}


```

Writing at 0x000057c1... (39 \$)
Writing at 0x0005e095... (42 \$)
Writing at 0x0006324c... (46 \$)
Writing at 0x00068367... (50 \$)

DUOT ESP8266 DEVIKIT V1.60MHz 82MHz. None on COM3
11:37 PM
5/30/2022



```
1.Lathan_14 | Arduino 1.8.19
File Edit Sketch Tools Help
1.Lathan_14
// Import required libraries
#include <WiFi.h>
#include <SyncTCP.h>
#include <ESPAsyncWebServer.h>

// Replace with your network credentials
const char* ssid = "Galaxy A51";
const char* password = "12345678";

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

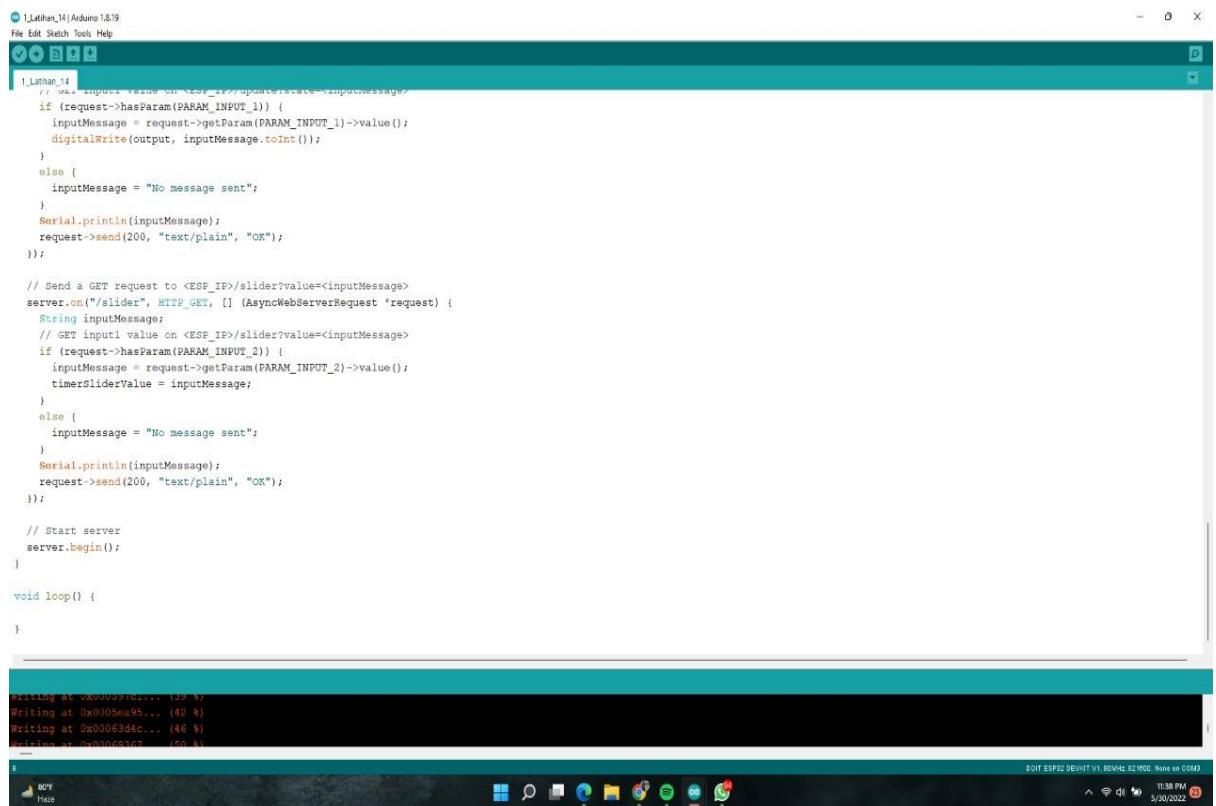
const int output = 2;

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>
  h1 {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 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}
</style>
<body>
  <h1>ESP Web Server</h1>
  <h2>Control</h2>
  <p>State: <input type="checkbox" id="checkbox1" checked="checked"/></p>
  <p>Value: <input type="text" id="text1" value="10"/></p>
  <h2>Slider</h2>
  <input type="range" id="range1" min="0" max="100" value="50" style="width: 100%;"/>
</body>
)
```

Writing at 0x00059761... (39 %)
Writing at 0x0005ea95... (42 %)
Writing at 0x0006334c... (46 %)
Writing at 0x00065307... (50 %)



```
1.Lathan_14 | Arduino 1.8.19
File Edit Sketch Tools Help
1.Lathan_14
void setup() {
    // Set input value on <ESP_IP>/update/state<br/>message
    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 input 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() {
}


```

Writing at 0x00059761... (39 %)
Writing at 0x0005ea95... (42 %)
Writing at 0x0006334c... (46 %)
Writing at 0x00065307... (50 %)

1_Lathan_14 | Arduino 1.8.19

```
// 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 input 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 input 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");
});

void loop() {
    // Start server
    server.begin();
}

Writing at 0x000097c1... (39 %)
Writing at 0x0005e95... (42 %)
Writing at 0x00063d4c... (46 %)
Writing at 0x00065307... (50 %)
```

BOI ESP32 DEVKIT V1.60MHz 82MHz None on COM3
11:38 PM 5/30/2022

1_Lathan_14 | Arduino 1.8.19

```
// Get input value on <ESP_IP>/update?state=<inputMessage>
if (request->hasParam(PARAM_INPUT_1)) {
    inputMessage = request->getParam(PARAM_INPUT_1);
    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 input value on <ESP_IP>/slider?value=<inputMessage>
    if (request->hasParam(PARAM_INPUT_2)) {
        inputMessage = request->getParam(PARAM_INPUT_2);
        timersliderValue = inputMessage;
    }
    else {
        inputMessage = "No message sent";
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
});

// Start server
server.begin();

void loop() {
}

Writing at 0x000097c1... (39 %)
Writing at 0x0005e95... (42 %)
Writing at 0x00063d4c... (46 %)
Writing at 0x00065307... (50 %)
```

BOI ESP32 DEVKIT V1.60MHz 82MHz None on COM3
11:38 PM 5/30/2022



ESP Web Server

9 s



- **Hasil Percobaan**



- **Demonstration**

Unggah kode ke papan NodeMCU ESP32 atau ESP8266 Anda. Kemudian, buka Serial Monitor dan tekan tombol RST/EN on-board untuk mendapatkan alamat IP. Buka browser di jaringan lokal Anda dan ketik alamat IP ESP. Halaman berikut harus dimuat. Seret penggeser untuk menyesuaikan lebar pulsa, lalu klik tombol ON/OFF. Output (dalam hal ini GPIO 2 – built-in LED) akan tetap menyala selama jangka waktu yang telah Anda atur pada slider.

