LAPORAN PRAKTIKUM ELEKTRONIKA
MESIN LISTRIK DAN TEKNIK KENDALI

Disusun Oleh:

Syarif M N Cahya (19/441211/SV/16563)

TEKNOLOGI REKAYASA MESIN

DEPARTEMEN TEKNIK MESIN SEKOLAH VOKASI

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2021

1. Spesifikasi

a. Feature Solution

ESP32 adalah chip kombo Wi-Fi-dan-Bluetooth 2,4 GHz tunggal yang dirancang dengan teknologi 40 nm ultra-low-power TSMC. Ini dirancang untuk mencapai kinerja daya dan RF terbaik, menunjukkan kekokohan, keserbagunaan, dan keandalan dalam berbagai aplikasi dan skenario daya.

ESP32 dirancang untuk aplikasi seluler, elektronik yang dapat dikenakan, dan Internet-of-Things (IoT). Ini menampilkan semua karakteristik canggih dari chip berdaya rendah, ESP32 dibangunkan secara berkala hanya ketika kondisi tertentu terdeteksi. Siklus tugas rendah digunakan untuk meminimalkan jumlah energi yang dikeluarkan chip. Output dari power amplifier juga dapat disesuaikan, sehingga berkontribusi pada trade-off yang optimal antara jangkauan komunikasi, kecepatan data, dan konsumsi daya.

ESP32 adalah solusi yang sangat terintegrasi untuk aplikasi IoT Wi-Fi-dan-Bluetooth, dengan sekitar 20 komponen eksternal. ESP32 mengintegrasikan sakelar 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).

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.

b. Wi-Fi 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 for Infrastructure Station, SoftAP, and Promiscouous modes Note that when ESP32 is in Station mode, performing a scan, the Soft AP channel will be changed.
- Antenna diversity

c. Bluetooth Key Features

- Compliant with Bluetooth v4.2 BR/EDR and Bluetooth LE specifications
- 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

d. MCU and Advanced Features

i. 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

ii. Clocks and Timers

- Internal 8 MHz oscillator with calibration
- Internal RC oscillator with calibration
- External 2 MHz ~ 60 MHz crystal oscillator (40 MHz only for Wi-Fi/Bluetooth functionality)
- External 32 kHz crystal oscillator for RTC with calibration
- Two timer groups, including 2 × 64-bit timers and 1 × main watchdog in each group
- One RTC timer
- RTC watchdog

iii. Advanced Peripheral Interfaces

- 34 × programmable GPIOs
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DAC
- 10 × touch sensors
- 4 × SPI
- 2 × I2S

- 2 × I2C
- 3 × 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

iv. Security

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

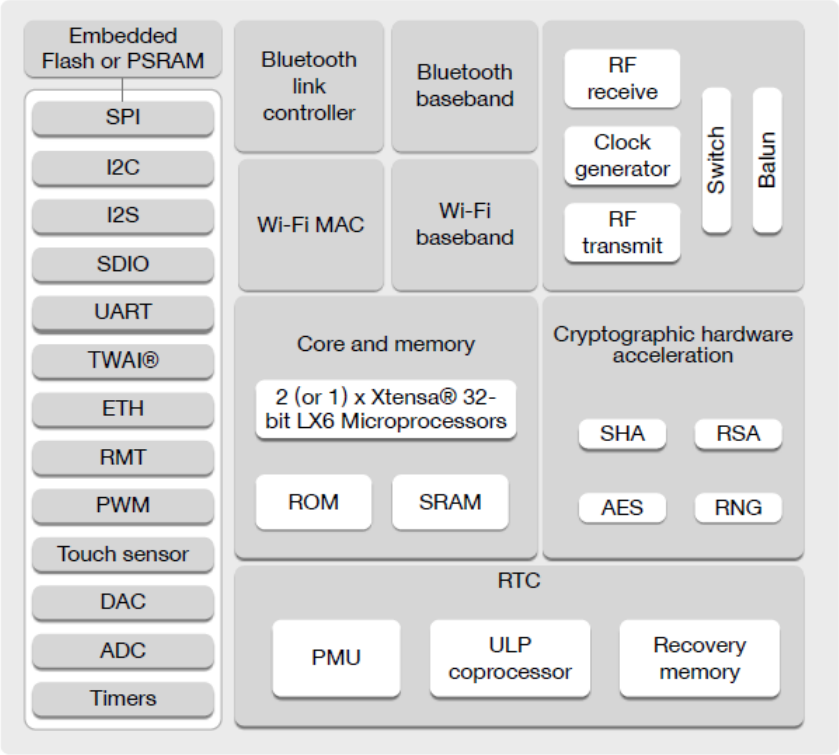
e. Applications (A Non-exhaustive List)

- Generic Low-power IoT Sensor Hub
- Generic Low-power IoT Data Loggers
- Cameras for Video Streaming
- Over-the-top (OTT) Devices
- Speech Recognition
- Image Recognition
- Mesh Network
- Home Automation
  - Light control
  - Smart plugs
  - Smart door locks
- Smart Building
  - Smart lighting
  - Energy monitoring
- Industrial Automation
  - Industrial wireless control
  - Industrial robotics
- Smart Agriculture
  - Smart greenhouses
  - Smart irrigation

- Agriculture robotics

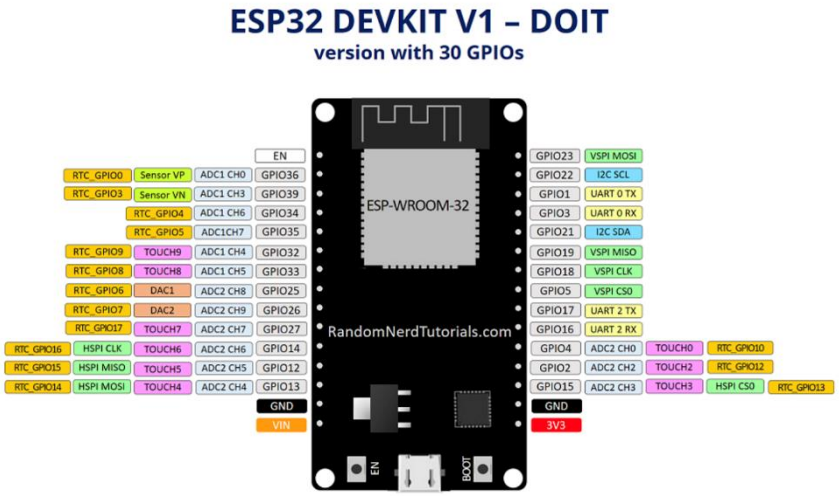
- Audio Applications
  - Internet music players
  - Live streaming devices
  - Internet radio players
  - Audio headsets
- Health Care Applications
  - Health monitoring
  - Baby monitors
- Wi-Fi-enabled Toys
  - Remote control toys
  - Proximity sensing toys
  - Educational toys
- Wearable Electronics
  - Smart watches
  - Smart bracelets
- Retail & Catering Applications
  - POS machines
  - Service robots

f. Block Diagram

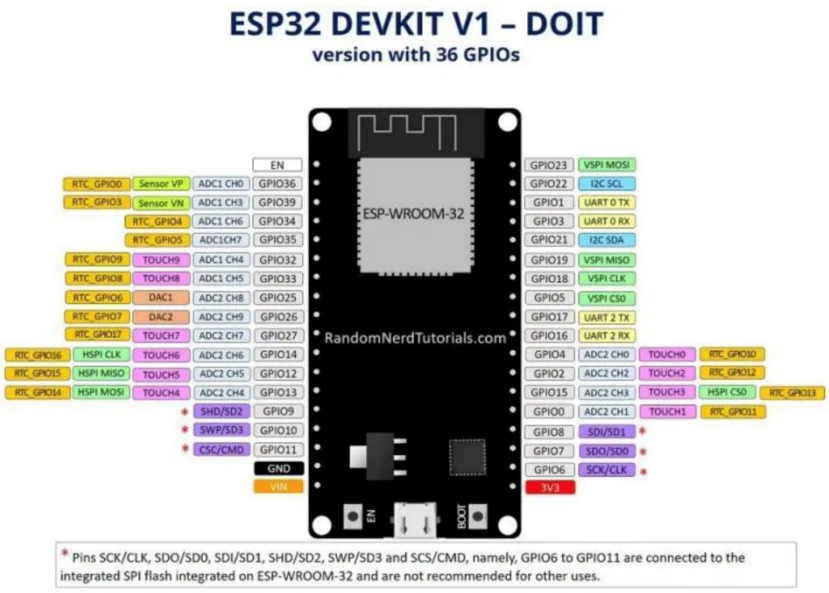


2. Kaki-kaki atau Pinout beserta fungsinya

Version with 30 GPIOs



Version with 36 GPIOs



GPIO	Input	Output	Catatan
0	pulled up	OK	output sinyal PWM saat **boot**
1	TX pin	OK	output debug saat **boot**
2	OK	OK	Terhubung ke LED on board
3	OK	TX pin	HIGH saat boot
4	OK	OK	

5	OK	OK	output sinyal PWM saat boot
6	x	x	terhubung dengan SPI Flash terintegrasi
7	x	x	terhubung dengan SPI Flash terintegrasi
8	x	x	terhubung dengan SPI Flash terintegrasi
9	x	x	terhubung dengan SPI Flash terintegrasi
10	x	x	terhubung dengan SPI Flash terintegrasi
11	x	x	terhubung dengan SPI Flash terintegrasi
12	OK	OK	boot gagal ketika mendapatkan input high
13	OK	OK	
14	OK	OK	output sinyal PWM saat boot
15	OK	OK	output sinyal PWM saat boot
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
20	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
24	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
28	OK	OK	
29	OK	OK	
30	OK	OK	
31	OK	OK	

32	OK	OK	
33	OK	OK	
34	OK	OK	
35	OK	OK	
36	OK		Hanya input
37	OK		Hanya input
38	OK		Hanya input
39	OK		Hanya input

- **Pin Hanya Untuk Input**

GPIO 34 hingga 39 hanyalah dipergunakan sebagai input. Pin – pin tersebut tidak memiliki *pull up internal* atau resistor pull down. Berikut adalah pin – pin tersebut

- GPIO 34
- GPIO 35
- GPIO 36
- GPIO 39

- **SPI flash terintegrsi dengan ESP-WROOM-32**

GPIO 6 hingga GPIO 11 dapat diakses oleh beberapa *development board* ESP32. Namun pin – pin tersebut terhubung kepada **SPI Flash** yang teritegrasi dengan ESP-WROOM-32 sehingga tidak direkomendasikan digunakan untuk keperluan lain. Jadi jangan gunakan pin – pin berikut

- GPIO 6 (SCK/CLK)
- GPIO 7 (SDO/SD0)
- GPIO 8 (SDI/SD1)
- GPIO 9 (SHD/SD2)
- GPIO 10 (SWP/SD3)
- GPIO 11 (CSC/CMD)

- **Capacitive touch GPIO**

ESP32 memiliki 10 sensor sentuh kapasitif yang dapat mengindera benda apapun yang menyimpan muatan listrik seperti kulit manusia. Sehingga pin – pin tersebut dapat mendeteksi variasi induksi ketika GPIO disentuh dengan jari. Pin ini dapat dengan mudah diintegrasikan dengan bantalan kapasitif dan menggantikan tombol mekanik.

Berikut adalah sensor internal sentuh yang terhubung dengan GPIO

- T0 (GPIO 4)
- T1 (GPIO 0)
- T2 (GPIO 2)
- T3 (GPIO 15)
- T4 (GPIO 13)
- T5 (GPIO 12)
- T6 (GPIO 14)
- T7 (GPIO 27)
- T8 (GPIO 33)
- T9 (GPIO 32)

- **Analog to Digital Converter (ADC)**

ESP32 memiliki 18 kanal masukan ADC 12 bit, sedangkan ESP8266 hanya 1 kanal ADC 10 bit. Berikut adalah GPIO yang dapat dipergunakan sebagai ADC berikut dengan kanalnya.

- ADC1_CH0 (GPIO 36)
- ADC1_CH1 (GPIO 37)
- ADC1_CH2 (GPIO 38)
- ADC1_CH3 (GPIO 39)
- ADC1_CH4 (GPIO 32)
- ADC1_CH5 (GPIO 33)
- ADC1_CH6 (GPIO 34)
- ADC1_CH7 (GPIO 35)
- ADC2_CH0 (GPIO 4)
- ADC2_CH1 (GPIO 0)
- ADC2_CH2 (GPIO 2)
- ADC2_CH3 (GPIO 15)
- ADC2_CH4 (GPIO 13)
- ADC2_CH5 (GPIO 12)
- ADC2_CH6 (GPIO 14)
- ADC2_CH7 (GPIO 27)
- ADC2_CH8 (GPIO 25)
- ADC2_CH9 (GPIO 26)

- **Digital to Analog Converter (DAC)**

Ada 2 kanal DAC 8 bit pada ESP32 yang berfungsi untuk mengubah sinyal digital ke keluaran tegangan analog. Berikut adalah GPIO dan kanal tersebut

- DAC1 (GPIO25)
- DAC2 (GPIO26)

- **GPIO Real Time Clock**

ESP32 juga dilengkapi dengan GPIO yang diarahkan ke RTC subsistem rendah daya yang dapat digunakan ketika ESP32 dalam kondisi *deep sleep*. GPIO RTC ini dapat digunakan untuk membangungkan ESP32 dari kondisi deep sleep ketika co-prosesor ULP (*Ultra Low Power*) sedang berjalan. Berikut adalah GPIO yang dapat digunakan sebagai *external wake up source*

- RTC_GPIO0 (GPIO36)
- RTC_GPIO3 (GPIO39)
- RTC_GPIO4 (GPIO34)
- RTC_GPIO5 (GPIO35)
- RTC_GPIO6 (GPIO25)
- RTC_GPIO7 (GPIO26)
- RTC_GPIO8 (GPIO33)
- RTC_GPIO9 (GPIO32)
- RTC_GPIO10 (GPIO4)
- RTC_GPIO11 (GPIO0)
- RTC_GPIO12 (GPIO2)
- RTC_GPIO13 (GPIO15)
- RTC_GPIO14 (GPIO13)
- RTC_GPIO15 (GPIO12)
- RTC_GPIO16 (GPIO14)
- RTC_GPIO17 (GPIO27)

- **PWM**

ESP32 memiliki 16 kanal PWM independen yang dapat dikonfigurasi untuk menghasilkan sinyal PWM dengan pengaturan yang berbeda – beda. Semua pin yang dapat menjadi keluaran dapat dipergunakan sebagai pin PWM (kecuali GPIO 34 hingga 39)

Untuk mengatur sinyal PWM, perlu ditentukan terlebih dahulu parameter – parameter berikut pada program

- Frekuensi gelombang;
- *Duty cycle*;
- Kanal PWM;
- GPIO mana yang dipergunakan sebagai keluaran gelombang.

- **I2C (Inter-Integrated Circuit)**

Jika pembaca memprogram ESP32 menggunakan Arduini IDE, ada dua pin *default* yang mendukung I2C dan didukung oleh pustaka Wire, yaitu

- GPIO 21 (SDA)

- GPIO 22 (SCL)

- **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

- **Interrupts**

  Semua GPIO pada ESP32 dapat diatur sebagai *Interrupt*

- **Strapping Pins**

  ESP32 memiliki 6 pin *strapping* sebagai berikut

  - GPIO 0
  - GPIO 2
  - GPIO 4
  - GPIO 5
  - GPIO 12
  - GPIO 15

  Pin – pin tersebut digunakan oleh ESP32 saat mode *flashing* atau *bootloader*. Pada sebagian besar board development yang memiliki USB/Serial built-in. kondisi state pin ini tidak perlu dikuatirkan. Karena board yang akan mengaturnya ke kondisi yang sesuai saat mode *flashing* atau *boot.*

  Meskipun begitu, jika terdapat periferal yang terhubung dengan ke-6 pin tersebut. Ada kemungkinan, proses *flashing* akan mengalami masalah. Penyebabnya bisa jadi karena periferal yang terhubung dengan pin – pin tersebut mencegah ESP32 masuk ke mode *flashing* atau *boot*. Baca tautan berikut tentang dokumentasi Boot Mode pada ESP32

- **Pin berada pada kondisi HIGH saat Boot**

  Beberapa GPIO berubah kondisi *state*nya menjadi HIGH atau keluaran sinyal PWM saat boot atau reset. Ini berarti, akan terdapat keluaran yang tak terduga saat pin – pin GPIO berikut berada pada kondisi **reset** atau **boot.**

  - GPIO 1
  - GPIO 3
  - GPIO 5

- GPIO 6 s/d GPIO 11 (terhubung dengan ESP32 yang terintegrasi dengan memori flash SPI – tidak direkomendasikan untuk digunakan).
- GPIO 14
- GPIO 15

- **Enable (EN)**

  Pin **Enable** (EN) adalah pin enable regulator 3.3V yang di-pullup dengan resistor sehingga mengetanahkan pin tersebut (menghubungkan ke ground), akan mendisable regulator 3.3V. Sehingga pin ini dapat dihubungkan dengan *push button* misalnya, guna merestart ESP32

- **Arus Yang Dapat Ditarik Pada GPIO**

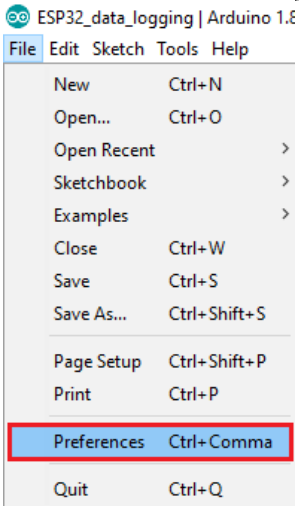
  Arus maksimal yang dapat ditarik oleh periferal di tiap GPIO adalah 12mA, sebagaimana tertulis pada *datasheet*

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

To install the ESP32 board in your Arduino IDE, follow these next instructions:

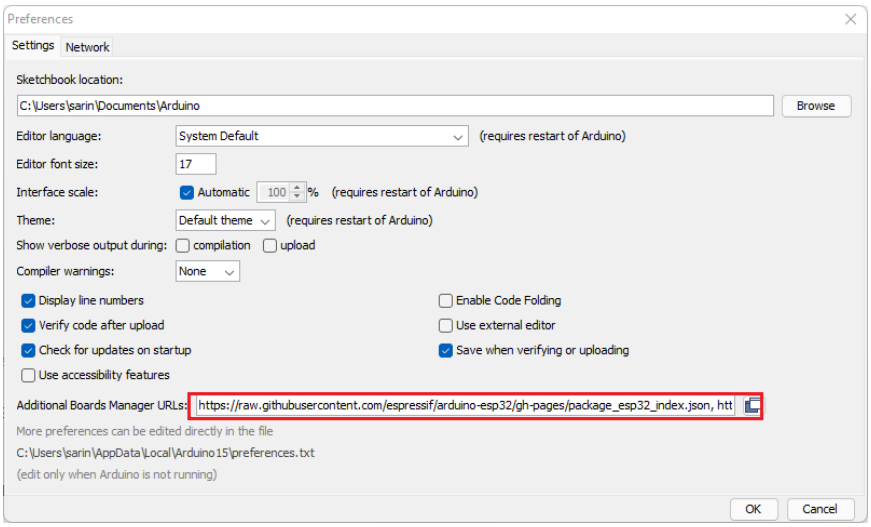
1. In your Arduino IDE, go to **File**> **Preferences**



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

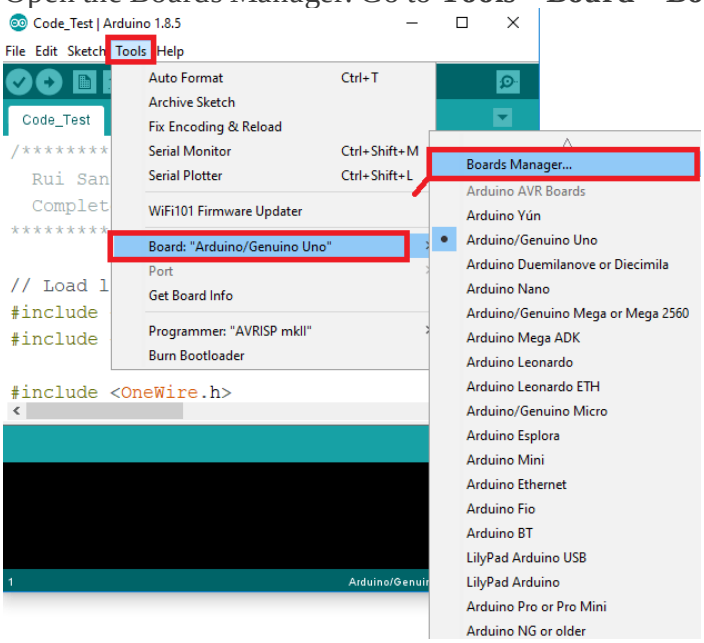
   https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

   Then, click the “OK” button:

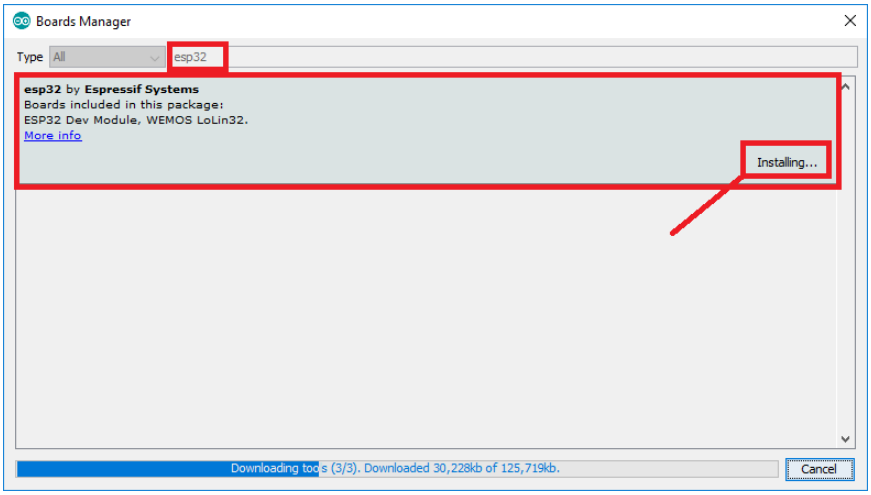


   Note: if you already have the ESP8266 boards URL, you can separate the URLs with a comma as follows: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json

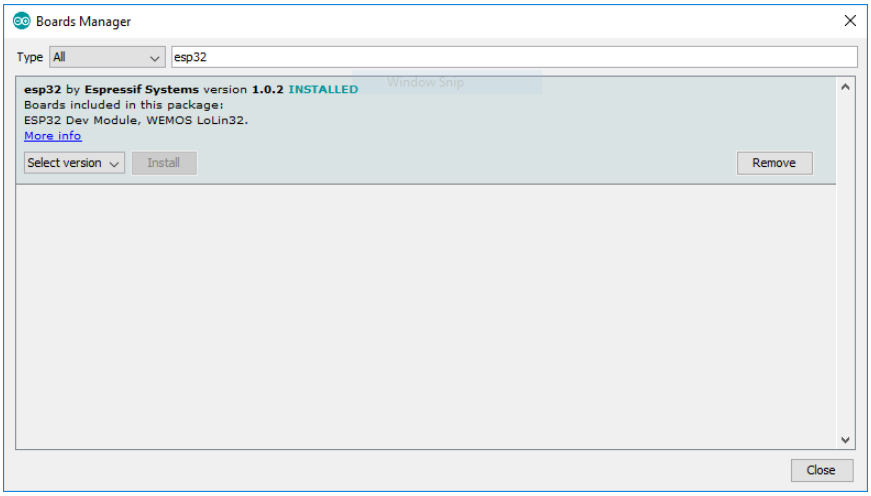
3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**



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.



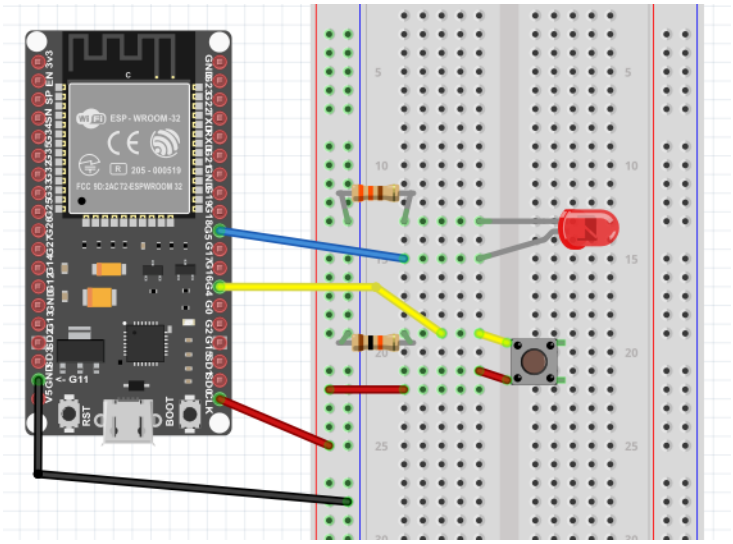
4. ESP32 Input / Output

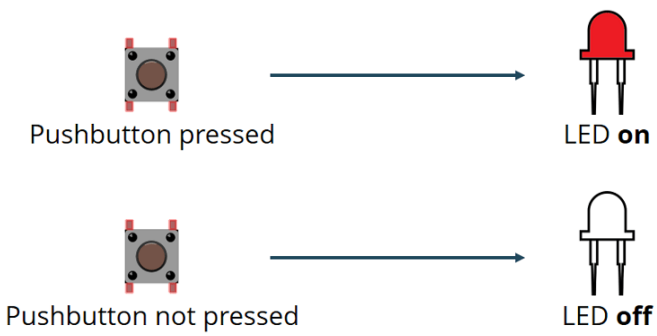
ESP32 memiliki pin Input dan Output sebagai antarmuka untuk berkomunikasi dengan dunia luar. Pin ini disebut sebagai General Purpose Input Output (GPIO). Antarmuka Input Output pada ESP32, terdiri dari:

- 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

GPIO bisa digunakan untuk menerima sinyal input digital maupun analog, dan juga mengeluarkan sinyal output digital maupun analog. Pada sinyal digital berarti hanya ada

dua kondisi, yaitu OFF (0 Volt) dan ON (3,3 Volt). Sinyal analog memiliki besaran yang bukan hanya ON dan OFF saja, tetapi ada nilai lain di antara kondisi ON dan OFF.





//Kode

```
// set pin numbers
const int buttonPin = 4;  // the number of the pushbutton pin
const int ledPin =  5;    // the number of the LED pin
// variable for storing the pushbutton status
int buttonState = 0;
void setup() {
  Serial.begin(115200);
  // initialize the pushbutton pin as an input
  pinMode(buttonPin, INPUT);
  // initialize the LED pin as an output
  pinMode(ledPin, OUTPUT);
}
void loop() {
  // read the state of the pushbutton value
  buttonState = digitalRead(buttonPin);
  Serial.println(buttonState);
  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH
  if (buttonState == HIGH) {
    // turn LED on
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off
    digitalWrite(ledPin, LOW);
  }

}
```

```
// set pin numbers
const int buttonPin = 4;
const int ledPin =  5;

int buttonState = 0;

void setup() {
  Serial.begin(115200);
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);
  Serial.println(buttonState);
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Set GPIO 4 untuk push button dan GPIO 5 untuk LED

Variabel untuk menahan state button di 0

Setup Push button sebagai INPUT dan LED sebagai Outpunya

Loop agar program terus berjalan

membaca status button dan menyimpannya di variabel buttonState, degnan digitalRead()

jika buttonstate HIGH (ditekan) maka LedPin akan diubah ke HIGH dengan digitalWrite()

Jika tida, maka ledPin dibuit pada keadaan LOW

5. ESP32 PWM

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

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

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

   Fungsi ini menerima dua argumen. Yang pertama adalah GPIO yang akan mengeluarkan sinyal, dan yang kedua adalah saluran yang akan menghasilkan sinyal.
5. Terakhir, untuk mengontrol kecerahan LED menggunakan PWM, digunakan fungsi berikut:

   ```
   ledcWrite(channel, dutycycle)
   ```

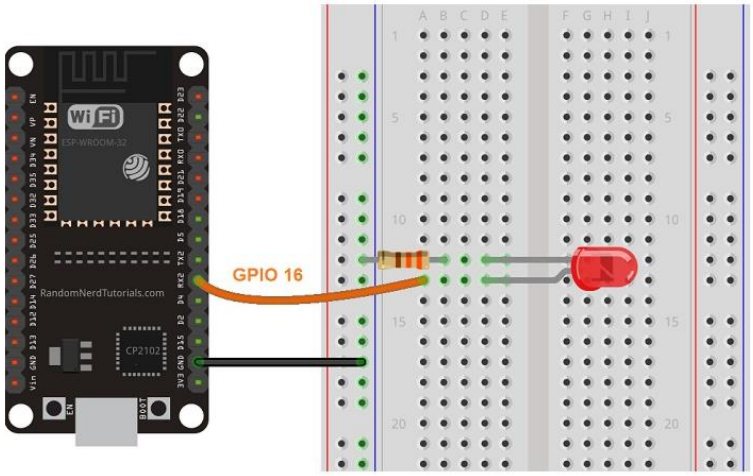
   Fungsi ini menerima sebagai argumen saluran yang menghasilkan sinyal PWM, dan siklus kerja.

Meredupkan LED

Mari kita lihat contoh sederhana untuk melihat cara menggunakan kontroler PWM LED ESP32 menggunakan Arduino IDE.

Skema

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



Code

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

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

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

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

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

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

Anda mulai dengan menentukan pin yang dipasangi LED. Dalam hal ini LED terpasang ke GPIO 16.

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

Kemudian, Anda mengatur properti sinyal PWM. Anda menentukan frekuensi 5000 Hz, memilih saluran 0 untuk menghasilkan sinyal, dan menetapkan resolusi 8 bit. Anda 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;
```

Di setup(), Anda perlu mengkonfigurasi LED PWM dengan properti yang telah Anda tentukan sebelumnya dengan menggunakan fungsi ledcSetup() yang menerima sebagai argumen, ledChannel, frekuensi, dan resolusi, sebagai berikut:

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

Selanjutnya, Anda harus memilih GPIO yang akan menerima sinyal. Untuk itu gunakan fungsi ledcAttachPin() yang menerima sebagai argumen GPIO tempat Anda ingin mendapatkan sinyal, dan saluran yang menghasilkan sinyal. Dalam contoh ini, kita akan mendapatkan sinyal di ledPin GPIO, yang sesuai dengan GPIO 16. Saluran yang menghasilkan sinyal adalah ledChannel, yang sesuai dengan saluran 0.

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

Dalam loop, Anda 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.

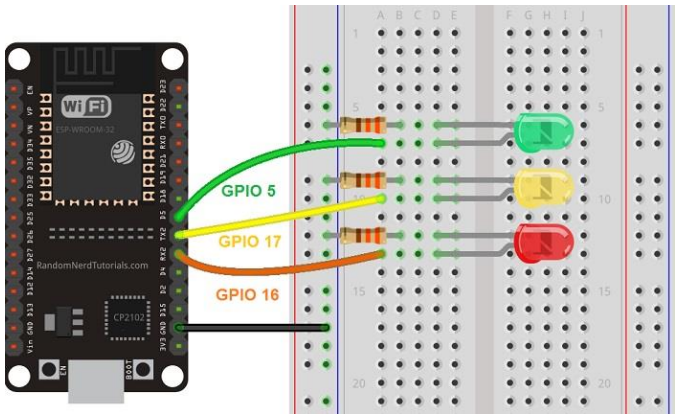
```
for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
 }
```

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

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

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

Skema



Code

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

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

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

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(ledPin, ledChannel);
  ledcAttachPin(ledPin2, ledChannel);
  ledcAttachPin(ledPin3, ledChannel);
}

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

  // decrease the LED brightness
```

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

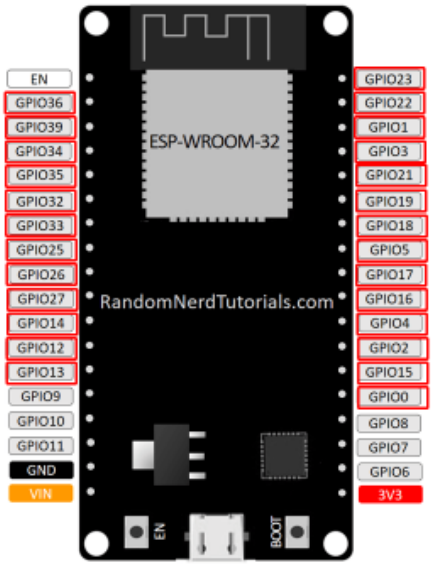
6. ESP32 Interrupt

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

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

GPIO Interrupt

Argumen pertama adalah nomor GPIO. Biasanya, Anda harus menggunakan digitalPinToInterrupt(GPIO) untuk mengatur GPIO yang sebenarnya sebagai pin interupsi. Dengan papan ESP32, semua pin yang disorot dengan persegi panjang merah pada gambar berikut dapat dikonfigurasi sebagai pin interupsi.



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.

Untuk contoh ini akan menggunakan mode RISING, karena ketika sensor gerak PIR mendeteksi gerakan, maka GPIO yang terhubung akan berubah dari LOW ke HIGH.

Timer

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

The Delay() Function

Anda harus terbiasa dengan fungsi delay() karena banyak digunakan. Fungsi ini cukup mudah digunakan. Ia menerima nomor int tunggal sebagai argumen. Angka ini menunjukkan waktu dalam milidetik program harus menunggu sampai pindah ke baris kode berikutnya.

```
delay(time in milliseconds)
```

Ketika Anda melakukan delay(1000) program Anda berhenti pada baris itu selama 1 detik. delay() adalah fungsi pemblokiran. Fungsi pemblokiran mencegah program melakukan hal lain sampai tugas tertentu selesai. Jika Anda memerlukan beberapa tugas untuk dilakukan pada saat yang sama, Anda tidak dapat menggunakan delay(). Untuk sebagian besar proyek, Anda harus menghindari penggunaan penundaan dan menggunakan pengatur waktu sebagai gantinya.

The Millis() Function

Menggunakan fungsi yang disebut milis() Anda dapat mengembalikan jumlah milidetik yang telah berlalu sejak program pertama kali dimulai. Mengapa fungsi itu berguna? Karena dengan menggunakan beberapa matematika, Anda dapat dengan mudah memverifikasi berapa banyak waktu yang telah berlalu tanpa memblokir kode Anda.

Blinking an LED with millis()

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

Code

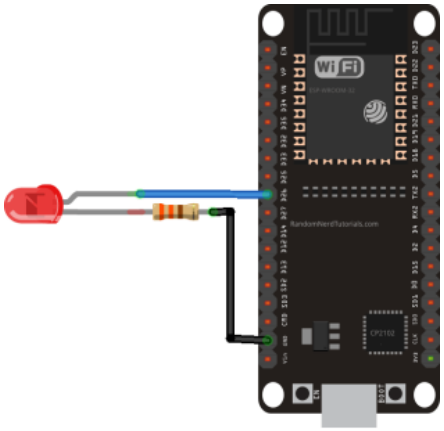
```
          // constants won't change. Used here to set a pin number :
       const int ledPin =  26;      // the number of the LED pin

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

       // Generally, you should use "unsigned long" for variables that hold time
       // The value will quickly become too large for an int to store
       unsigned long previousMillis = 0;        // will store last time LED was
updated

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

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

       void loop() {
         // here is where you'd put code that needs to be running all the time.

         // check to see if it's time to blink the LED; that is, if the
         // difference between the current time and last time you blinked
         // the LED is bigger than the interval at which you want to
         // blink the LED.
         unsigned long currentMillis = millis();

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

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

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

How The Code Works

Mari kita lihat lebih dekat sketsa kedipan ini yang bekerja tanpa fungsi delay() (sebagai gantinya menggunakan fungsi millis()).

Pada dasarnya, kode ini mengurangi waktu yang tercatat sebelumnya (sebelumnyaMillis) dari waktu saat ini (Millis saat ini). Jika sisa lebih besar dari interval (dalam hal ini, 1000 milidetik), program memperbarui variabel Millis sebelumnya ke waktu saat ini, dan menyalakan atau mematikan LED.

Karena cuplikan ini tidak memblokir, kode apa pun yang terletak di luar pernyataan if pertama akan berfungsi secara normal. Anda sekarang seharusnya dapat memahami bahwa Anda dapat menambahkan tugas lain ke fungsi loop() dan kode Anda akan tetap mengedipkan LED setiap satu detik. Anda dapat mengunggah kode ini ke ESP32 Anda dan merakit diagram skematik berikut untuk mengujinya dan mengubah jumlah milidetik untuk melihat cara kerjanya.



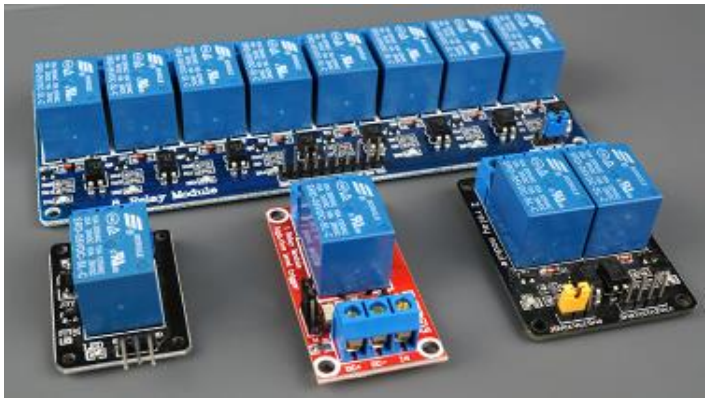
7. ESP32 Relay

Menggunakan relai dengan ESP32 adalah cara yang bagus untuk mengontrol peralatan rumah tangga AC dari jarak jauh. Tutorial ini menjelaskan cara mengontrol modul relai dengan ESP32. Kami akan melihat cara kerja modul relai, cara menghubungkan relai ke ESP32 dan membangun server web untuk mengontrol relai dari jarak jauh (atau relai sebanyak yang Anda inginkan).

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

1, 2, 4, 8, 16 Channels Relay Modules

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.

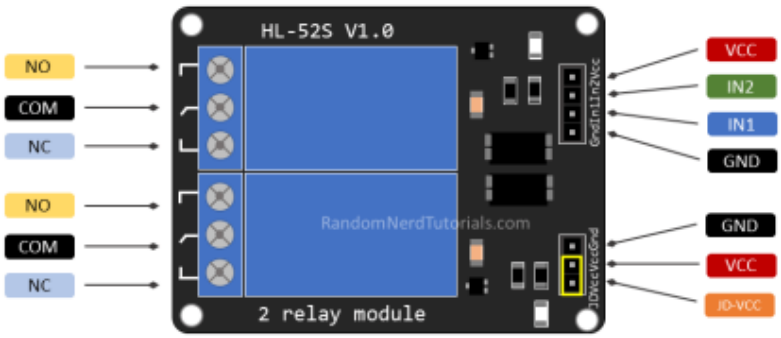


Ada modul relai yang elektromagnetnya dapat ditenagai oleh 5V dan dengan 3.3V. Keduanya dapat digunakan dengan ESP32 – Anda dapat menggunakan pin VIN (yang menyediakan 5V) atau pin 3.3V.

Selain itu, beberapa dilengkapi dengan optocoupler internal yang menambahkan "lapisan" perlindungan ekstra, yang secara optik mengisolasi ESP32 dari sirkuit relai.

Relay Pinout

Untuk tujuan demonstrasi, mari kita lihat pinout modul relai 2 saluran. Menggunakan modul relai dengan jumlah saluran yang berbeda serupa.



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

Mains Voltages Connections

Modul relai yang ditunjukkan pada foto sebelumnya 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

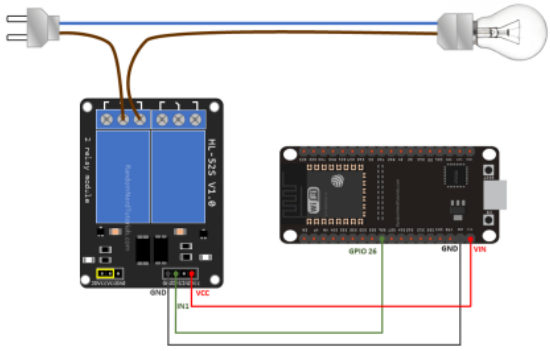
Sisi tegangan rendah memiliki satu set empat pin dan satu set tiga pin. Set pertama terdiri dari VCC dan GND untuk menyalakan modul, dan input 1 (IN1) dan input 2 (IN2) untuk mengontrol relai bawah dan atas, masing-masing. Jika modul relai Anda hanya memiliki satu saluran, Anda hanya akan memiliki satu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

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

Wiring a Relay Module 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.

Controlling a Relay Module with the ESP32

Kode untuk mengontrol relai dengan ESP32 semudah mengontrol LED atau output lainnya. Dalam contoh ini, karena kita menggunakan konfigurasi yang biasanya terbuka, kita perlu mengirim sinyal RENDAH untuk membiarkan arus mengalir, dan sinyal TINGGI untuk menghentikan aliran arus. Kode berikut akan menyalakan lampu Anda selama 10 detik dan mematikannya selama 10 detik lagi.

Code

```
 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);
}
```

How The Code Works

Tentukan pin yang terhubung dengan pin IN relay.

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

Dalam setup(), tentukan relai sebagai output.

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

Di loop(), kirim sinyal RENDAH untuk membiarkan arus mengalir dan menyalakan lampu.

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

Jika Anda menggunakan konfigurasi yang biasanya tertutup, kirim sinyal HIGH untuk menyalakan lampu. Kemudian, tunggu 5 detik.

```
delay(5000);
```

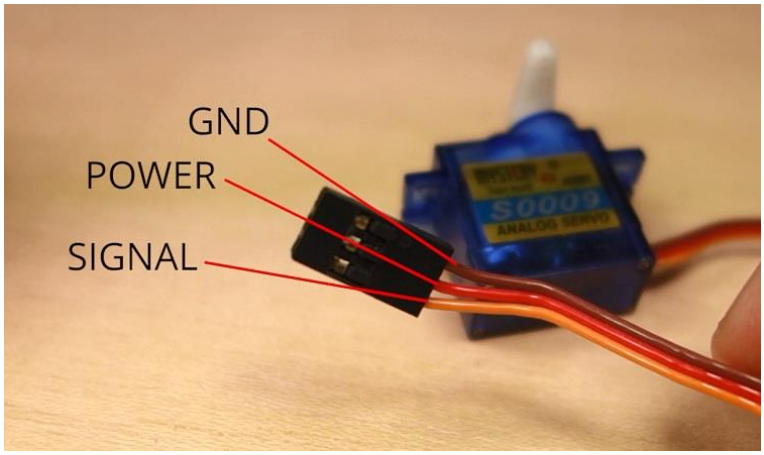
Hentikan aliran arus dengan mengirimkan sinyal TINGGI ke pin relai. Jika Anda menggunakan konfigurasi yang biasanya tertutup, kirim sinyal RENDAH untuk menghentikan aliran arus.

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

8. ESP32 Servo

Connecting the Servo Motor to the ESP32

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



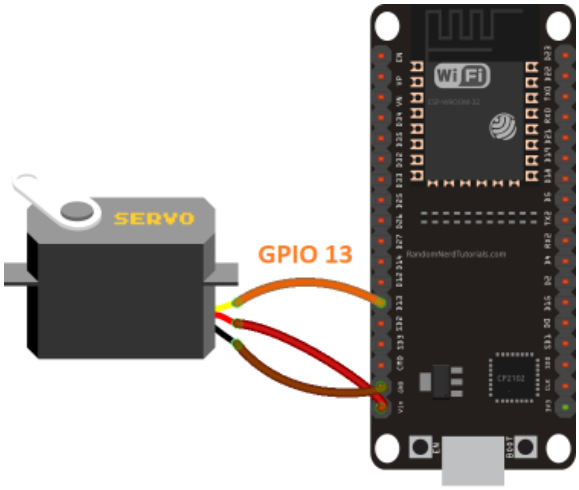
Wire	Color
Power	Red
GND	Black, or brown
Signal	Yellow, orange, or white

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).

Scematic

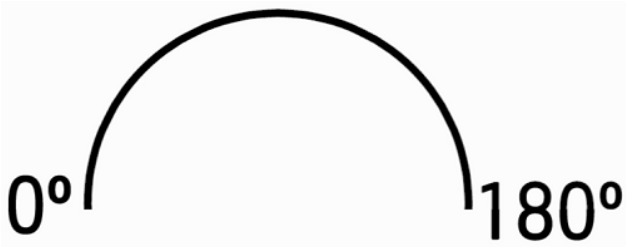
Dalam contoh kami, kami akan menghubungkan kabel sinyal ke GPIO 13. Jadi, Anda dapat mengikuti diagram skema berikutnya untuk menyambungkan motor servo Anda.



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

How to Control a Servo Motor

Anda dapat memposisikan poros servo dalam berbagai sudut dari 0 hingga 180º. Servo dikendalikan menggunakan sinyal modulasi lebar pulsa (PWM). Artinya sinyal PWM yang dikirim ke motor akan menentukan posisi poros. Untuk mengendalikan motor cukup menggunakan kemampuan PWM dari ESP32 dengan mengirimkan sinyal 50Hz dengan lebar pulsa yang sesuai. Atau Anda dapat menggunakan perpustakaan untuk membuat tugas ini lebih sederhana.



Preparing the Arduino IDE

Ada add-on untuk Arduino IDE yang memungkinkan Anda memprogram ESP32 menggunakan Arduino IDE dan bahasa pemrogramannya. Ikuti salah satu tutorial berikutnya untuk mempersiapkan Arduino IDE Anda untuk bekerja dengan ESP32, jika Anda belum melakukannya.

- Instruksi Windows – Papan ESP32 di Arduino IDE
- Instruksi Mac dan Linux – Papan ESP32 di Arduino IDE

Setelah memastikan Anda telah menginstal add-on ESP32, Anda dapat melanjutkan tutorial ini.

Installing the ESP32 Arduino Servo Library

Perpustakaan Servo Arduino ESP32 memudahkan untuk mengontrol motor servo dengan ESP32 Anda, menggunakan Arduino IDE. Ikuti langkah-langkah selanjutnya untuk menginstal perpustakaan di Arduino IDE Anda:

a. Klik di sini untuk mengunduh ESP32_Arduino_Servo_Library. Anda harus memiliki folder .zip di folder Unduhan Anda
b. Buka zip folder .zip dan Anda akan mendapatkan folder ESP32-Arduino-Servo-Library-Master
c. Ganti nama folder Anda dari ESP32-Arduino-Servo-Library-Master menjadi ESP32_Arduino_Servo_Library
d. Pindahkan folder ESP32_Arduino_Servo_Library ke folder library instalasi Arduino IDE anda
e. Terakhir, buka kembali IDE Arduino Anda

Testing

Setelah menginstal perpustakaan, buka Arduino IDE Anda. Pastikan Anda telah memilih papan ESP32, lalu, buka File > Contoh > ServoESP32 > Servo Sederhana.

Code

```
  #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
  }
}
```

Understanding the Code

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

Pertama, Anda perlu menyertakan perpustakaan Servo:

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

Kemudian, Anda perlu membuat objek servo. Dalam hal ini disebut myservo.

```
Servo myservo;
```

Setup()

Dalam setup(), Anda menginisialisasi komunikasi serial untuk tujuan debugging, dan melampirkan GPIO 13 ke objek servo.

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

Loop()

Pada loop(), kita mengubah posisi poros motor dari 0 menjadi 180 derajat, kemudian dari 180 menjadi 0 derajat. Untuk mengatur poros ke posisi tertentu, Anda hanya perlu menggunakan metode write() di objek servo. Anda lulus sebagai argumen, bilangan bulat dengan posisi dalam derajat.

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

Testing the Sketch

Unggah kode ke ESP32 Anda. Setelah mengunggah kode, Anda akan melihat poros motor berputar ke satu sisi dan kemudian, ke sisi lainnya.

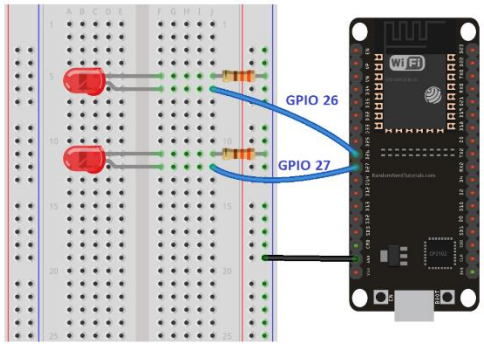
9. ESP32 Output WebServer

Pada output webserver 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 webserver 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 di rancang pada webserver, nantinya dapat dengan mudah untuk mengganti status output setiap lampu LED yang sudah di setting pada program.



Code

```
// Load Wi-Fi library
#include <WiFi.h>

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

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

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

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

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

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

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

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

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

            // turns the GPIOs on and off
            if (header.indexOf("GET /26/on") >= 0) {
              Serial.println("GPIO 26 on");
              output26State = "on";
              digitalWrite(output26, HIGH);
            } else if (header.indexOf("GET /26/off") >= 0) {
              Serial.println("GPIO 26 off");
              output26State = "off";
              digitalWrite(output26, LOW);
            } else if (header.indexOf("GET /27/on") >= 0) {
              Serial.println("GPIO 27 on");
              output27State = "on";
              digitalWrite(output27, HIGH);
            } else if (header.indexOf("GET /27/off") >= 0) {
              Serial.println("GPIO 27 off");
              output27State = "off";
              digitalWrite(output27, LOW);
            }

            // Display the HTML web page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=device-
width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the on/off buttons
            // Feel free to change the background-color and font-size attributes to
fit your preferences
```

```
            client.println("<style>html { font-family: Helvetica; display: inline-
block; margin: 0px auto; text-align: center;}");
            client.println(".button { background-color: #4CAF50; border: none;
color: white; padding: 16px 40px;");
            client.println("text-decoration: none; font-size: 30px; margin: 2px;
cursor: pointer;}");
            client.println(".button2 {background-color: #555555;}</style></head>");

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

            // Display current state, and ON/OFF buttons for GPIO 26
            client.println("<p>GPIO 26 - State " + output26State + "</p>");
            // If the output26State is off, it displays the ON button
            if (output26State=="off") {
              client.println("<p><a href=\"/26/on\"><button
class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/26/off\"><button class=\"button
button2\">OFF</button></a></p>");
            }

            // Display current state, and ON/OFF buttons for GPIO 27
            client.println("<p>GPIO 27 - State " + output27State + "</p>");
            // If the output27State is off, it displays the ON button
            if (output27State=="off") {
              client.println("<p><a href=\"/27/on\"><button
class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/27/off\"><button class=\"button
button2\">OFF</button></a></p>");
            }
            client.println("</body></html>");

            // The HTTP response ends with another blank line
            client.println();
            // Break out of the while loop
            break;
          } else { // if you got a newline, then clear currentLine
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a carriage return
character,
          currentLine += c;      // add it to the end of the currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```

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

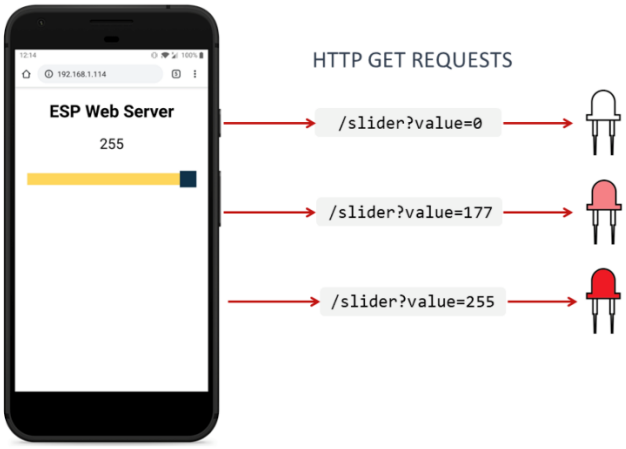
Setting Your Networks Credentials

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

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

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. Variable 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.



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

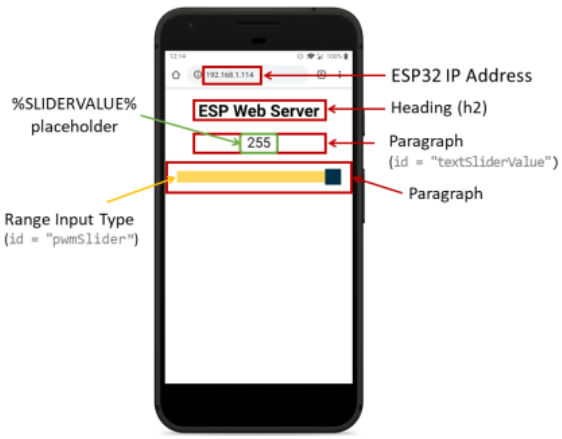
Code

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

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

```
const int output = 2;

String sliderValue = "0";

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

const char* PARAM_INPUT = "value";

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

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>ESP Web Server</title>
  <style>
    html {font-family: Arial; display: inline-block; text-align: center;}
    h2 {font-size: 2.3rem;}
    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();
}

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

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

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

  ledcWrite(ledChannel, sliderValue.toInt());

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

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

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

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

  // Start server
  server.begin();
}

void loop() {

}
```

Building the Web Page



Halaman web untuk proyek ini cukup sederhana. Ini berisi satu judul, satu paragraf dan satu input dari rentang jenis. Mari kita lihat bagaimana halaman web dibuat. Semua teks HTML dengan gaya yang disertakan disimpan dalam variabel index_html. Sekarang kita akan melihat teks HTML dan melihat apa yang dilakukan setiap bagian.

Code (style)

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

Adding JavaScript to the HTML File

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

Code

```
<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>
```

Processor

Sekarang, kita perlu membuat fungsi processor(), yang akan menggantikan placeholder dalam teks HTML kita dengan nilai slider saat ini ketika Anda mengaksesnya untuk pertama kali di browser.

Code

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

Handle Request

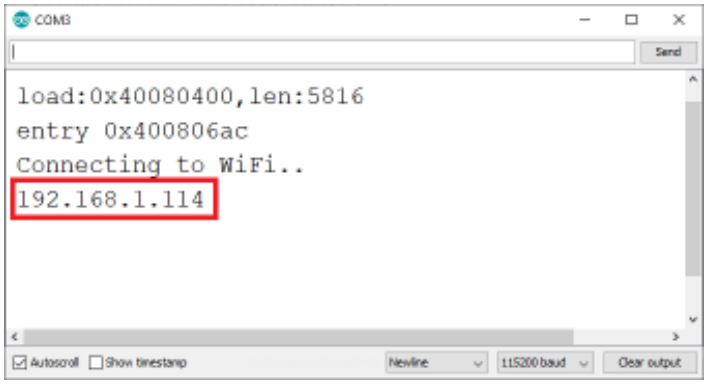
Terakhir, tambahkan baris kode berikutnya untuk menangani server web.

Code

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

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

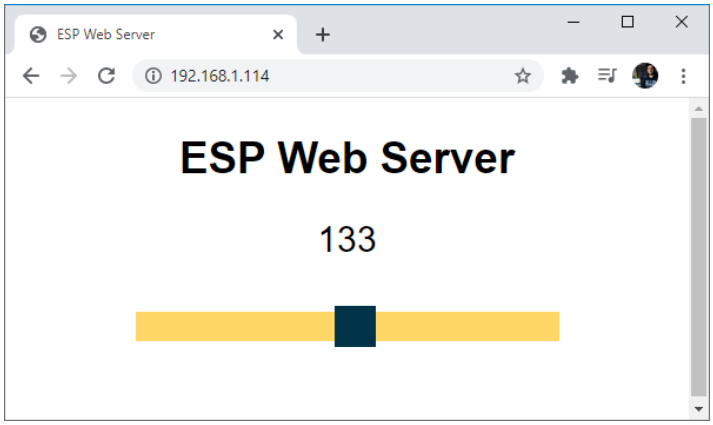
Upload The Code

Sekarang, unggah kode ke ESP32 Anda. Pastikan Anda memilih papan dan port COM yang tepat. Setelah mengunggah, buka Serial Monitor dengan baud rate 115200. Tekan tombol reset ESP32. Alamat IP ESP32 harus dicetak di monitor serial.

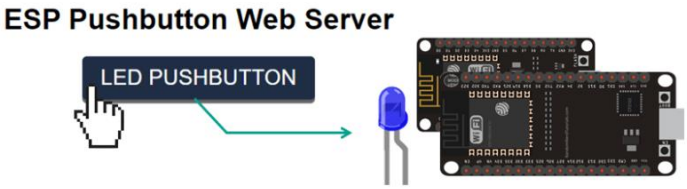


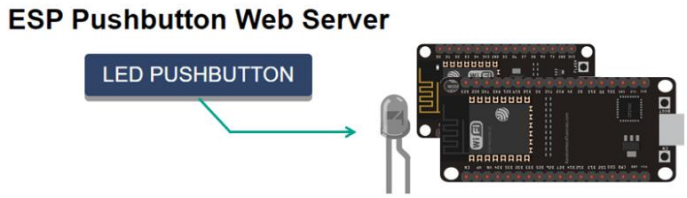
Web Server Demonstration

Buka browser dan ketik alamat IP ESP32. Server web Anda harus menampilkan bilah geser dan nilainya saat ini. Gerakkan penggeser dan lihat LED internal ESP32 meningkat dan menurun kecerahannya.



11.Esp32 Momentary Switch WebServer





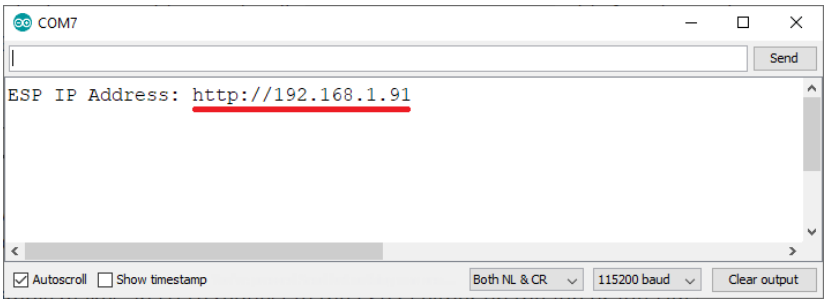
- ESP32 atau ESP8266 host server web yang dapat Anda akses untuk mengontrol output;
- Status default output adalah LOW, tetapi Anda dapat mengubahnya tergantung pada aplikasi proyek Anda;
- Ada tombol yang berfungsi seperti sakelar sesaat:
    - jika Anda menekan tombol, output berubah statusnya menjadi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

Code

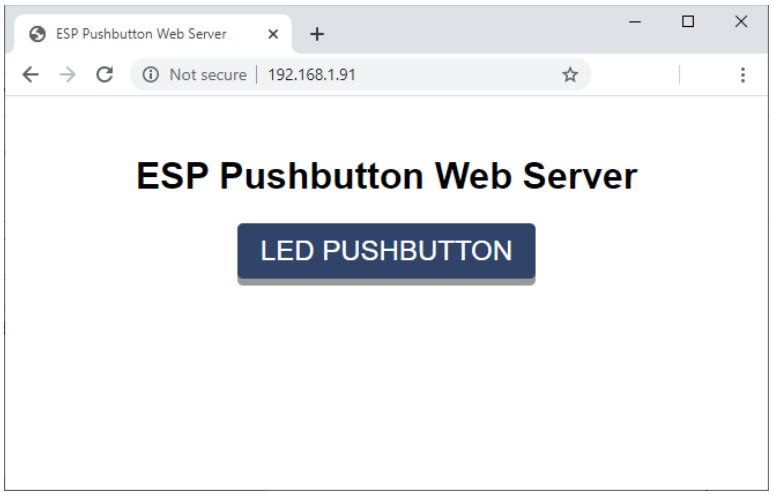
```
#ifdef ESP32
  #include <WiFi.h>
  #include <AsyncTCP.h>
#else
  #include <ESP8266WiFi.h>
  #include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>

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

const int output = 2;

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

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

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

AsyncWebServer server(80);

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

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

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

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

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

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

void loop() {

}
```

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



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



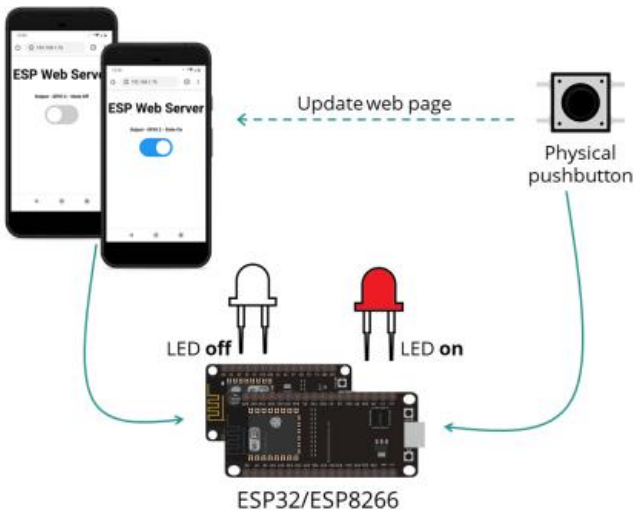
LED on-board tetap menyala selama Anda terus menekan tombol di halaman web.

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 perbuahan yang dilakukan pada web server atau tombol secara langsung.

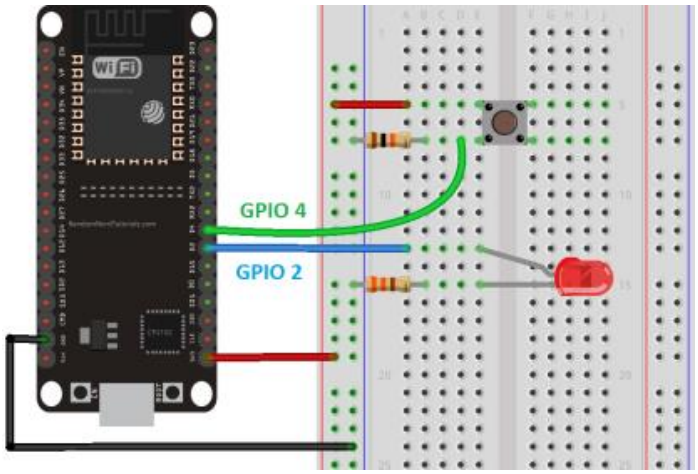
Dapat dijelaskan secara lebih ringkas tahapannya seperti :

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

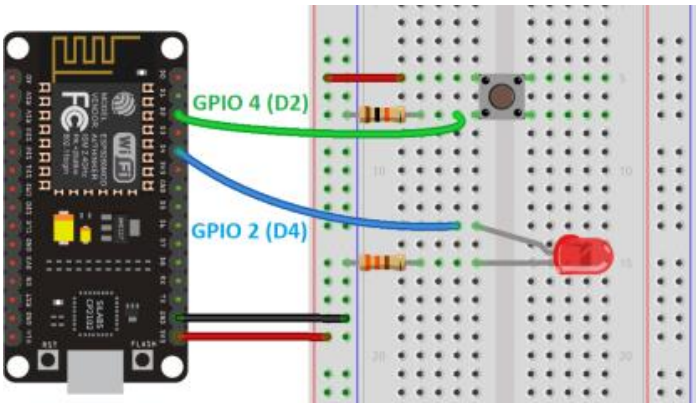


Schematic Diagram

ESP32 Schematic



ESP8266 NodeMCU Schematic



Code

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

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

const char* PARAM_INPUT_1 = "state";

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

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

// the following variables are unsigned longs because the time, measured in
// milliseconds, will quickly become a bigger number than can be stored in an int.
```

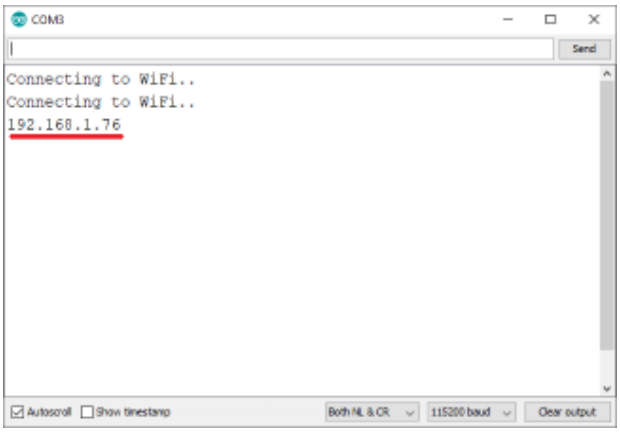
```
unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled
unsigned long debounceDelay = 50;    // the debounce time; increase if the output
flickers

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

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <title>ESP Web Server</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    html {font-family: Arial; display: inline-block; text-align: center;}
    h2 {font-size: 3.0rem;}
    p {font-size: 3.0rem;}
    body {max-width: 600px; margin:0px auto; padding-bottom: 25px;}
    .switch {position: relative; display: inline-block; width: 120px; height: 68px}
    .switch input {display: none}
    .slider {position: absolute; top: 0; left: 0; right: 0; bottom: 0; background-
color: #ccc; border-radius: 34px}
    .slider:before {position: absolute; content: ""; height: 52px; width: 52px;
left: 8px; bottom: 8px; background-color: #fff; -webkit-transition: .4s;
transition: .4s; border-radius: 68px}
    input:checked+.slider {background-color: #2196F3}
    input:checked+.slider:before {-webkit-transform: translateX(52px); -ms-
transform: translateX(52px); transform: translateX(52px)}
  </style>
</head>
<body>
  <h2>ESP Web Server</h2>
  %BUTTONPLACEHOLDER%
<script>function toggleCheckbox(element) {
  var xhr = new XMLHttpRequest();
  if(element.checked){ xhr.open("GET", "/update?state=1", true); }
  else { xhr.open("GET", "/update?state=0", true); }
  xhr.send();
}

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

// Replaces placeholder with button section in your web page
```

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

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

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

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

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

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

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

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

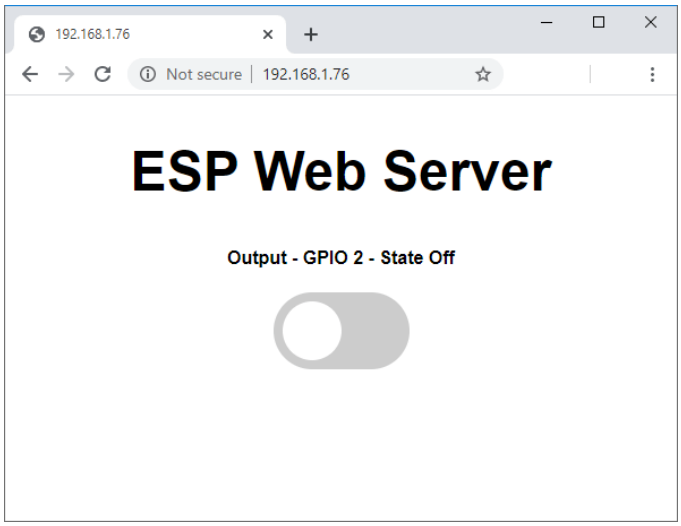
```
  // Send a GET request to <ESP_IP>/state
  server.on("/state", HTTP_GET, [] (AsyncWebServerRequest *request) {
    request->send(200, "text/plain", String(digitalRead(output)).c_str());
  });
  // Start server
  server.begin();
}

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

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

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

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

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

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

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

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

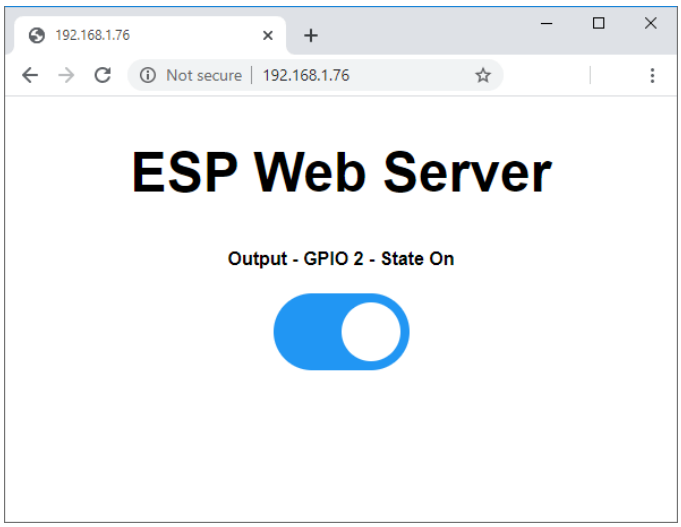
Demonstration

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

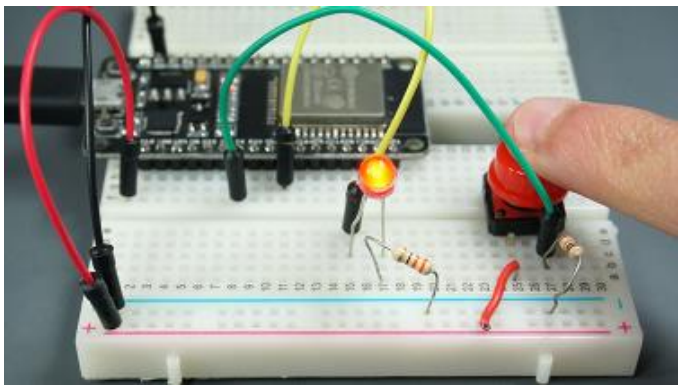


Buka browser di jaringan lokal Anda, dan ketik alamat IP ESP. Anda harus memiliki akses ke server web seperti yang ditunjukkan di bawah ini. Anda dapat mengaktifkan tombol di server web untuk menyalakan LED.





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



13. ESP32 OLED Display

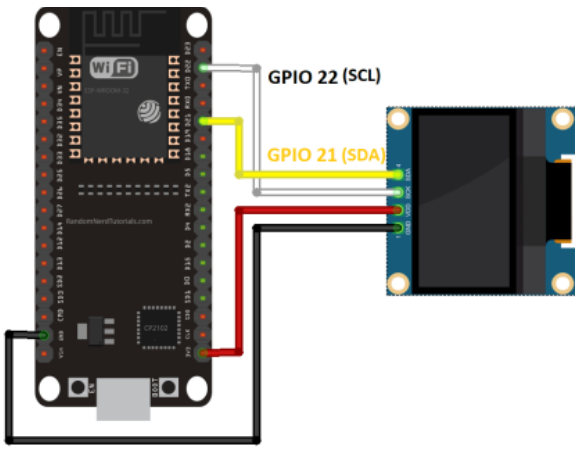
Oled Display merupakan bentuk suatu pemrograman dengan cara menunjukkan teks, mengatur font yang berbeda, menggambar suatu bentuk, serta menampilkan gambar bitmap. Oled Display nantinya menggunakan layar OLED SSD1306 0.96 inci dengan ESP32 menggunakan arduino IDE.

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.

Karena layar OLED menggunakan protokol komunikasi I2C, pengkabelan menjadi sangat sederhana. Anda dapat menggunakan tabel berikut sebagai referensi.

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

Atau, Anda dapat mengikuti diagram skema berikutnya untuk menyambungkan ESP32 ke layar OLED.



Dalam contoh ini, kami menggunakan protokol komunikasi I2C. Pin yang paling cocok untuk komunikasi I2C di ESP32 adalah GPIO 22 (SCL) dan GPIO 21 (SDA).

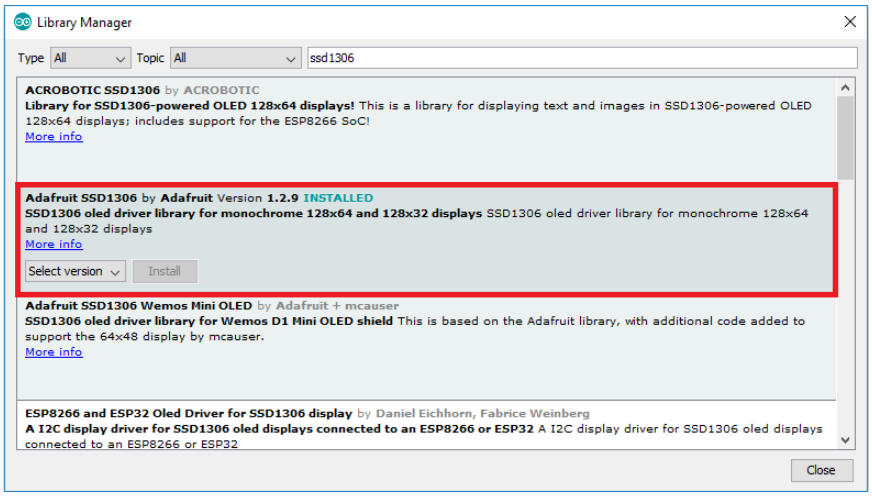
Jika Anda menggunakan layar OLED dengan protokol komunikasi SPI, gunakan GPIO berikut.

- GPIO 18 : CLK
- GPIO 19 : MISO
- GPIO 23 : MOSI
- GPIO 5 : CS

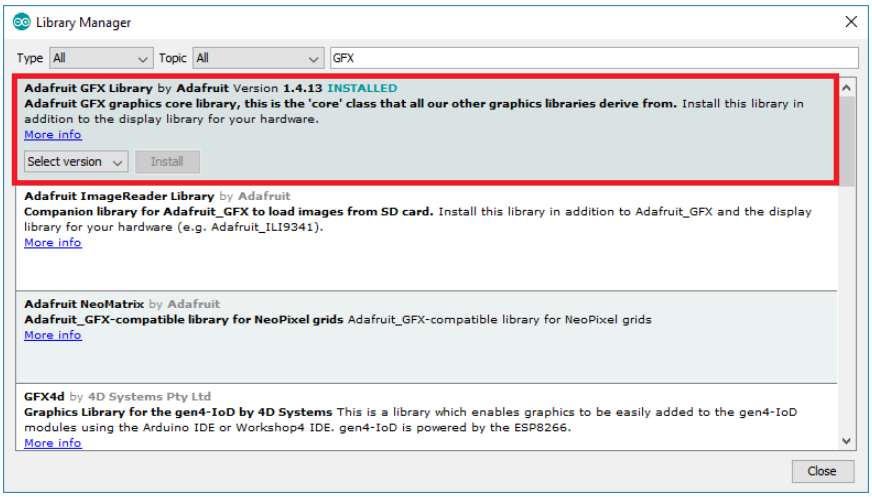
Installing SSD1306 OLED Library – ESP32

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.

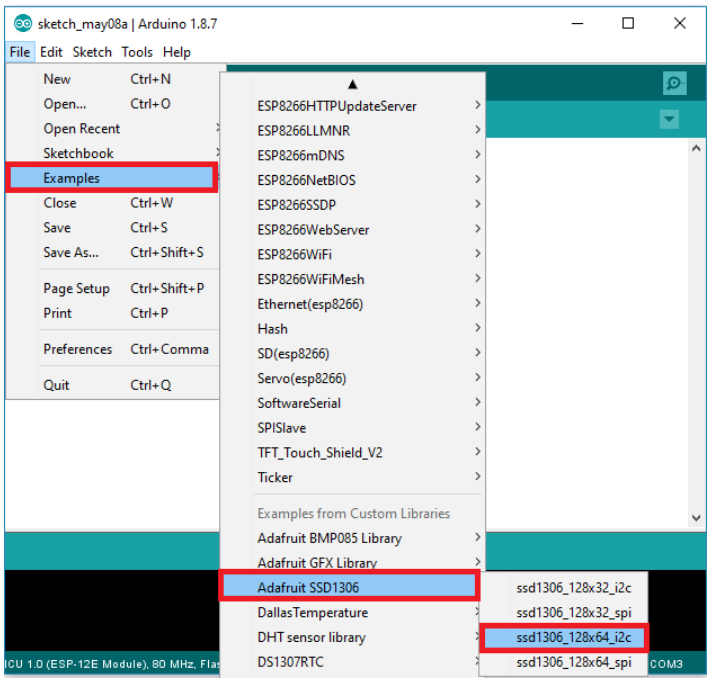


4. Setelah menginstal perpustakaan, restart Arduino IDE Anda.

Testing

Setelah memasang kabel layar OLED ke ESP32 dan menginstal semua pustaka yang diperlukan, Anda dapat menggunakan satu contoh dari pustaka untuk melihat apakah semuanya berfungsi dengan baik.

Di Arduino IDE Anda, buka File > Contoh > Adafruit SSD1306 dan pilih contoh untuk tampilan yang Anda gunakan.



Code

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

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

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

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

#define LOGO_HEIGHT   16
#define LOGO_WIDTH    16
static const unsigned char PROGMEM logo_bmp[] =
{ B00000000, B11000000,
  B00000001, B11000000,
  B00000001, B11000000,
  B00000011, B11100000,
  B11110011, B11100000,
  B11111110, B11111000,
  B01111110, B11111111,
  B00110011, B10011111,
  B00011111, B11111100,
```

```
  B00001101, B01110000,
  B00011011, B10100000,
  B00111111, B11100000,
  B00111111, B11110000,
  B01111100, B11110000,
  B01110000, B01110000,
  B00000000, B00110000 };

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

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

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

  // Clear the buffer
  display.clearDisplay();

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

  // Show the display buffer on the screen. You MUST call display() after
  // drawing commands to make them visible on screen!
  display.display();
  delay(2000);
  // display.display() is NOT necessary after every single drawing command,
  // unless that's what you want...rather, you can batch up a bunch of
  // drawing operations and then update the screen all at once by calling
  // display.display(). These examples demonstrate both approaches...

  testdrawline();      // Draw many lines

  testdrawrect();      // Draw rectangles (outlines)

  testfillrect();      // Draw rectangles (filled)

  testdrawcircle();    // Draw circles (outlines)

  testfillcircle();    // Draw circles (filled)

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

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

  testdrawtriangle();  // Draw triangles (outlines)

  testfilltriangle();  // Draw triangles (filled)

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

  testdrawstyles();    // Draw 'stylized' characters

  testscrolltext();    // Draw scrolling text

  testdrawbitmap();    // Draw a small bitmap image

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

```
  display.invertDisplay(false);
  delay(1000);

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

void loop() {
}

void testdrawline() {
  int16_t i;

  display.clearDisplay(); // Clear display buffer

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

  display.clearDisplay();

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

  display.clearDisplay();

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

  display.clearDisplay();

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

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

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

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

  delay(2000);
}

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

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

  delay(2000);
}

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

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

  delay(2000);
}

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

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

  delay(2000);
}

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

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

  delay(2000);
}
```

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

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

  delay(2000);
}

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

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

  delay(2000);
}

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

  for(int16_t i=max(display.width(),display.height())/2; i>0; i-=5) {
    // The INVERSE color is used so triangles alternate white/black
    display.fillTriangle(
      display.width()/2  , display.height()/2-i,
      display.width()/2-i, display.height()/2+i,
      display.width()/2+i, display.height()/2+i, INVERSE);
    display.display();
    delay(1);
  }

  delay(2000);
}

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

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

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

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

void testdrawstyles(void) {
```

```
  display.clearDisplay();

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

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

  display.setTextSize(2);             // Draw 2X-scale text
  display.setTextColor(WHITE);
  display.print(F("0x")); display.println(0xDEADBEEF, HEX);

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

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

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

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

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

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

#define XPOS   0 // Indexes into the 'icons' array in function below
#define YPOS   1
#define DELTAY 2

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

  // Initialize 'snowflake' positions
  for(f=0; f< NUMFLAKES; f++) {
    icons[f][XPOS]   = random(1 - LOGO_WIDTH, display.width());
```

```
    icons[f][YPOS]   = -LOGO_HEIGHT;
    icons[f][DELTAY] = random(1, 6);
    Serial.print(F("x: "));
    Serial.print(icons[f][XPOS], DEC);
    Serial.print(F(" y: "));
    Serial.print(icons[f][YPOS], DEC);
    Serial.print(F(" dy: "));
    Serial.println(icons[f][DELTAY], DEC);
  }

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

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

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

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

Write Text OLED – Display

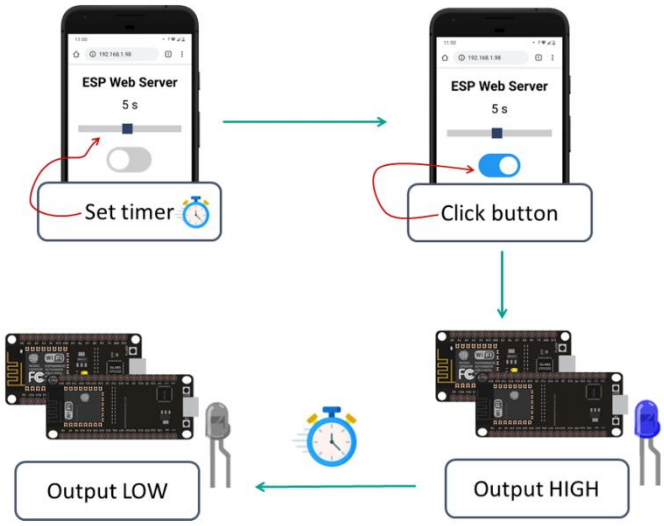
Code

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

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

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

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

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

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  // Display static text
  display.println("Hello, world!");
```

```
  display.display();
}

void loop() {

}
```

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.



Code

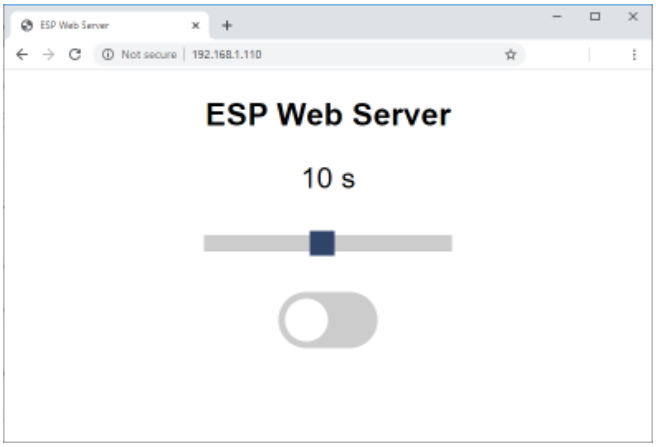
```
  // Import required libraries
#ifdef ESP32
  #include <WiFi.h>
  #include <AsyncTCP.h>
#else
  #include <ESP8266WiFi.h>
  #include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>

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

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

const int output = 2;

String timerSliderValue = "10";

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

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

```
  if(element.checked){ xhr.open("GET", "/update?state=1", true); xhr.send();
    var count = sliderValue, timer = setInterval(function() {
      count--; document.getElementById("timerValue").innerHTML = count;
      if(count == 0){ clearInterval(timer);
document.getElementById("timerValue").innerHTML =
document.getElementById("timerSlider").value; }
    }, 1000);
    sliderValue = sliderValue*1000;
    setTimeout(function(){ xhr.open("GET", "/update?state=0", true);
    document.getElementById(element.id).checked = false; xhr.send(); },
sliderValue);
  }
}
function updateSliderTimer(element) {
  var sliderValue = document.getElementById("timerSlider").value;
  document.getElementById("timerValue").innerHTML = sliderValue;
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/slider?value="+sliderValue, true);
  xhr.send();
}
</script>
</body>
</html>
)rawliteral";

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

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

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

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

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

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

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

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

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

  // Start server
  server.begin();
}

void loop() {

}
```

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.