MODUL 2
ESP32

Ahmad Aziz (13220034)
Asisten: Sidartha Prastya P (13219033)
Tanggal Percobaan: 24/2/2023
EL3215 - Praktikum Sistem Mikroprosesor
Laboratorium Sistem Kendali dan Komputer - Sekolah Teknik Elektro dan Informatika ITB

Abstrak

Abstrak *Praktikum sistem mikroprosesor modul ini adalah percobaan dengan ESP32. Percobaan dilakukan dengan dua compiler yaitu ESP-IDF dan juga Arduino IDE.*

Kata kunci: ESP32, ESP-IDF, Arduino IDE, Register, API.

1. PENDAHULUAN

ESP32 adalah sebuah sistem mikrokontroler (microcontroller system) yang dikembangkan oleh perusahaan asal Tiongkok bernama Espressif Systems. ESP32 memiliki fitur yang sangat lengkap, dengan kemampuan dual-core processor, Wi-Fi, Bluetooth, serta dukungan penggunaan jaringan LoRa, yang memungkinkan penggunaan perangkat ESP32 untuk berkomunikasi melalui jaringan nirkabel.

ESP32 banyak digunakan dalam proyek IoT (Internet of Things) karena kemampuannya untuk terhubung ke jaringan Wi-Fi dan Bluetooth, serta kemampuan untuk memproses data secara lokal di perangkat itu sendiri. Selain itu, ESP32 juga dilengkapi dengan berbagai sensor, antena, dan fitur lainnya yang memungkinkan penggunaan perangkat ini untuk berbagai macam aplikasi, seperti sensor suhu, sensor kelembaban, sensor gerakan, dan sebagainya.

Pada percobaan modul 2 ini dilakukan percobaan dengan ESP32 yang bertujuan sebagai berikut:

1. Praktikan memahami datasheet ESP32.
2. Praktikan mampu membuat aplikasi input dan output pada ESP32 dengan menggunakan bahasa pemrograman C pada ESP-IDF dan Arduino IDE.
3. Praktikan mampu membuat aplikasi timer dan interrupt pada ESP32 dengan menggunakan bahasa pemrograman C pada ESP-IDF dan Arduino IDE.
4. Praktikan mampu membuat web server sederhana pada ESP32 menggunakan ESP-IDF dan Arduino IDE.

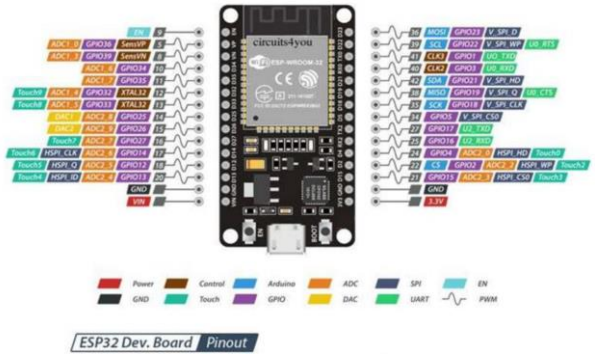
2. STUDI PUSTAKA

2.1 ESP32

Menurut Espressif Systems, ESP32 adalah chip kombo Wi-Fi dan Bluetooth 2,4 GHz yang dirancang dengan teknologi TSMC ultra-low-power 40 nm. ESP32 dirancang untuk mencapai kinerja daya dan RF terbaik, menunjukkan ketahanan, keserbagunaan, dan keandalan dalam berbagai aplikasi dan skenario daya. FItur utama yang ditawarkan ESP32 adalah sebagai berikut.

- Ultra-Low-Power sehingga cocok untuk aplikasi mobile, wearable electronics, dan Internet-of-Things (IoT).
- Complete Integration untuk aplikasi IoT dengan Wi-Fi dan Bluetooth. ESP32 mengintegrasikan antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, dan modul power management.

Berikut adalah tampilan dari ESP32 versi DOIT (yang akan digunakan pada praktikum) beserta pin-pinnya.



2.2 REGISTER

2.2.1 KONFIGURASI I/O ESP32 DAN INTERRUPT

Input dan output pada ESP32 diatur dengan GPIO. Ada 34 pad GPIO pada ESP32: 0-19, 21-23, 25-27, 32-39. Menurut datasheet, pad GPIO 0-19, 21-23, 25-27, 32-33 dapat berfungsi untuk input dan output. Pad GPIO 34-39 hanya untuk input. IO_MUX, RTC IO_MUX, dan matriks GPIO bertanggung jawab untuk merutekan sinyal dari periferal ke pad GPIO. Di bawah ini, akan diperlihatkan beberapa register GPIO matrix sebagai contoh.

2.2.2 DDR

Pada ATmega, mode input (PIN) atau output (PORT) diatur dengan DDR. Pada ESP32, mode input atau output diatur dengan GPIO_ENABLE_REG (jika 1 maka output).

Berikut adalah deskripsi tiap register GPIO matrix untuk Enable (Sumber: ESP32 Technical Reference Hal. 66):

Register 5.7: GPIO_ENABLE_REG (0x0020)

GPIO_ENABLE_REG GPIO0-31 output enable. (R/W)

2.2.3 OUTPUT

● GPIO_OUT_REG : Mengatur langsung output

● GPIO_OUT_W1TS_REG : Jika bit x diberikan 1, maka bit x pada GPIO_OUT_REG akan berubah menjadi 1

● GPIO_OUT_W1TC_REG : Jika bit x diberikan 1, maka bit x pada GPIO_OUT_REG akan berubah menjadi 0

Catatan: Untuk mengatur output, dapat digunakan 2 cara, yaitu dengan [menggunakan GPIO_OUT_REG langsung] atau [menggunakan GPIO_OUT_W1TS_REG dan GPIO_OUT_W1TC_REG]. Perbedaannya kedua cara ini akan terjadi jika ada multitasking. Penggunaan GPIO_OUT_REG pada multitasking dapat menimbulkan masalah. Berikut adalah deskripsi tiap register GPIO matrix untuk output (Sumber: ESP32 Technical Reference Hal. 65).

Register 5.1: GPIO_OUT_REG (0x0004)

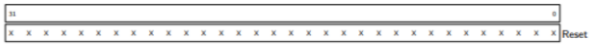
GPIO_OUT_REG GPIO0-31 output value. (R/W)

2.2.4 INPUT

● GPIO_OUT_REG: Membaca langsung input

Berikut adalah deskripsi tiap register GPIO matrix untuk input (Sumber: ESP32 Technical Reference Hal. 68).

Register 5.14: GPIO_IN_REG (0x003c)



GPIO_IN_REG GPIO0-31 input value. Each bit represents a pad input value, 1 for high level and 0 for low level. (RO)

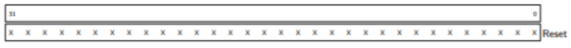
2.2.1 INTERUPT

● GPIO_STATUS_REG: Membaca status interrupt

● GPIO_PINx_REG: Bagian GPIO_PINx_INT_TYPE mengatur tipe interrupt (falling edge, rising edge, dll)

Berikut adalah deskripsi tiap register GPIO matrix untuk interrupt (Sumber: ESP32 Technical Reference Hal. 68).

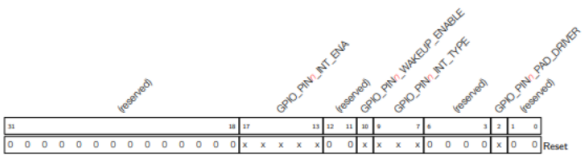
Register 5.16: GPIO_STATUS_REG (0x0044)



GPIO_STATUS_REG GPIO0-31 interrupt status register. Each bit can be either of the two interrupt sources for the two CPUs. The enable bits in GPIO_STATUS_INTERRUPT, corresponding to the 0-4 bits in GPIO_PIN*n*_REG should be set to 1. (R/W)

Berikut adalah deskripsi tiap register GPIO matrix untuk konfigurasi interrupt GPIO pin n (Sumber: ESP32 Technical Reference Hal. 72).

Register 5.30: GPIO_PIN*n*_REG (*n*: 0-39) (0x88+0x4**n*)



GPIO_PIN*n*_INT_ENA Interrupt enable bits for pin *n*: (R/W)
bit0: APP CPU interrupt enable;
bit1: APP CPU non-maskable interrupt enable;
bit3: PRO CPU interrupt enable;
bit4: PRO CPU non-maskable interrupt enable.

GPIO_PIN*n*_WAKEUP_ENABLE GPIO wake-up enable will only wake up the CPU from Light-sleep. (R/W)

GPIO_PIN*n*_INT_TYPE Interrupt type selection: (R/W)
0: GPIO interrupt disable;
1: rising edge trigger;
2: falling edge trigger;
3: any edge trigger;
4: low level trigger;
5: high level trigger.

GPIO_PIN*n*_PAD_DRIVER 0: normal output; 1: open drain output. (R/W)

2.2.2 TIMER DAN INTERRUPT TIMER

Pada ESP32, terdapat 2 jenis timer: 64-bit Timer dan Watchdog Timer. Watchdog Timer adalah timer yang digunakan untuk mereset ESP32, namun timer tersebut tidak akan dibahas pada modul ini. 64-bit Timer adalah timer yang dapat dipakai untuk penggunaan umum seperti yang telah dilakukan dengan ATmega pada Modul 1. Menurut datasheet, ESP32 mempunyai 2 modul Timer (ditandai dengan huruf n) yang mempunyai masing-masing 2 Timer (ditandai dengan huruf x) dengan nama TIMGn_Tx. Berikut adalah fitur 64-bit Timer (Datasheet ESP32 Hal. 25).

• Menggunakan APB clock (APB_CLK, normal 80 MHz) sebagai clock acuan

• 16-bit clock prescaler, dari 2 to 65536

• 64-bit time-base counter

• Configurable up/down time-base counter: incrementing atau decrementing

• Tersedia Interrupt (TIMGn_Tx_INT_T1_INT dan TIMGn_Tx_INT_T0_INT)

Berikut adalah deskripsi tiap register timer (Sumber: ESP32 Technical Reference Hal. 501).

Register 19.1: TIMGn_TxCONFIG_REG (x: 0-1) (0x0+0x24*x)

TIMGn_Tx_EN When set, the timer x time-base counter is enabled. (R/W)

TIMGn_Tx_INCREASE When set, the timer x time-base counter will increment every clock tick. When cleared, the timer x time-base counter will decrement. (R/W)

TIMGn_Tx_AUTORELOAD When set, timer x auto-reload at alarm is enabled. (R/W)

TIMGn_Tx_DIVIDER Timer x clock (Tx_clk) prescale value. (R/W)

TIMGn_Tx_EDGE_INT_EN When set, an alarm will generate an edge type interrupt. (R/W)

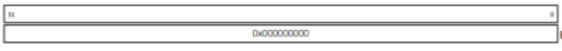
TIMGn_Tx_LEVEL_INT_EN When set, an alarm will generate a level type interrupt. (R/W)

TIMGn_Tx_ALARM_EN When set, the alarm is enabled. This bit is automatically cleared once an alarm occurs. (R/W)

Register 19.4: TIMGn_TxUPDATE_REG (x: 0-1) (0xC+0x24*x)

TIMGn_TxUPDATE_REG Write any value to trigger a timer x time-base counter value update (timer x current value will be stored in registers above). (WO)

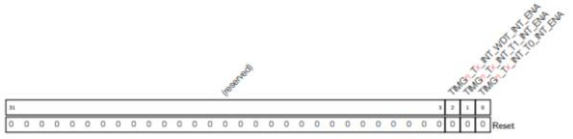
Register 19.9: TIMGn_TxLOAD_REG (x: 0-1) (0x20+0x24*x)



TIMGn_TxLOAD_REG Write any value to trigger a timer x time-base counter reload. (WO)

Berikut adalah deskripsi register timer interrupt (Sumber: ESP32 Technical Reference Hal. 505).

Register 19.18: TIMGn_Tx_INT_ENA_REG (0x0098)

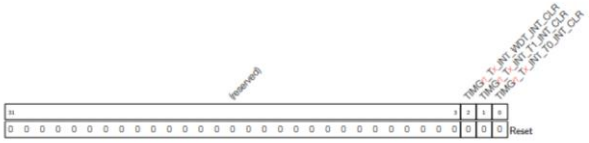


TIMGn_Tx_INT_WDT_INT_ENA The interrupt enable bit for the TIMGn_Tx_INT_WDT_INT interrupt. (H/W) (H/W)

TIMGn_Tx_INT_T1_INT_ENA The interrupt enable bit for the TIMGn_Tx_INT_T1_INT interrupt. (R/W) (R/W)

TIMGn_Tx_INT_T0_INT_ENA The interrupt enable bit for the TIMGn_Tx_INT_T0_INT interrupt. (R/W) (R/W)

Register 19.21: TIMGn_Tx_INT_CLR_REG (0x00a4)



TIMGn_Tx_INT_WDT_INT_CLR Set this bit to clear the TIMGn_Tx_INT_WDT_INT interrupt. (WO)

TIMGn_Tx_INT_T1_INT_CLR Set this bit to clear the TIMGn_Tx_INT_T1_INT interrupt. (WO)

TIMGn_Tx_INT_T0_INT_CLR Set this bit to clear the TIMGn_Tx_INT_T0_INT interrupt. (WO)

2.3 ESP-IDF

ESP-IDF adalah IoT Development Framework resmi Espressif untuk SoC seri ESP32 dan ESP32-S. ESP-IDF menyediakan SDK untuk pengembangan aplikasi umum pada platform tersebut menggunakan bahasa pemrograman seperti C dan C ++.

2.4 API

2.4.1 GPIO.H

Pada praktikum modul 1, telah dicontohkan cara pemrograman mikroprosesor dengan mengubah secara langsung register yang ada. Pada modul 2 ini, akan digunakan API dari gpio.h (library) yang akan melakukan pemrograman register 36 tersebut secara otomatis ketika kita memanggil fungsi yang ktia inginkan. Berikut adalah beberapa contoh nama fungsi yang akan digunakan dalam modul 2 ini beserta kegunaannya dari dokumentasi Espressif (Sumber: ESP-IDF API Reference).

2.4.2 TASK.H

task.h adalah API yang digunakan bersama dengan FreeRTOS.h untuk membuat Real-Time Operating System (RTOS). Penjelasan dan praktikum lebih lanjut mengenai FreeRTOS akan diberikan pada modul 4. Pada modul 2 ini, task.h akan digunakan untuk fungsi delay. Delay yang diapakai adalah vTaskDelay yang penjelasannya adalah sebagai berikut dari dokumentasi FreeRTOS (Sumber: FreeRTOS API Reference).

2.4.3 TIMER

timer.h adalah API yang digunakan untuk mengatur timer pada ESP32 yang akan digunakan. Berikut adalah beberapa contoh nama fungsi timer

yang akan digunakan dalam modul 2 ini beserta kegunaannya dari dokumentasi Espressif (Sumber: ESP-IDF API Reference).

2.4.4 ESP-WIFI.H

esp_wifi.h adalah API yang digunakan untuk mengatur dan memonitor fungsi jaringan WiFi pada ESP32. Pengaturan ini termasuk untuk keperluan berikut.

• Station mode (aka STA mode atau WiFi client mode). ESP32 terkoneksi ke access point.

• AP mode (aka Soft-AP mode atau Access Point mode). Station terkoneksi ke ESP32.

• Combined AP-STA mode (ESP32 berperan sebagai access point dan station yang terknoeksi ke access point lainnya).

• Beberapa security modes untuk mode-mdoe diatas (WPA, WPA2, WEP, etc.)

• Scanning untuk access points (active & passive scanning).

• Promiscuous mode untuk monitoring IEEE802.11 WiFi packets.

Berikut adalah beberapa contoh nama fungsi WiFi yang akan digunakan dalam modul 2 ini beserta kegunaannya dari dokumentasi Espressif (Sumber: ESP-IDF API Reference).

2.4.5 ESP_HTTP_SERVER.H

esp_wifi.h adalah API yang digunakan ESP32 dapat menjalankan fungsi web server ringan. Berikut adalah beberapa contoh nama fungsi HTTP yang akan digunakan dalam modul 2 ini beserta kegunaannya dari dokumentasi Espressif (Sumber: ESP-IDF API Reference).

3. METODOLOGI

Dalam percobaan pada modul ini, ada beberapa peralatan yang digunakan yaitu sebagai berikut:

1. ESP32
2. LED (8 buah)
3. Resistor 150 Ohm (8 buah)
4. Push Button
5. Kabel Jumper

Langkah umum penyettingan alat dalam melakukan percobaan pada modul ini adalah sebagai berikut:

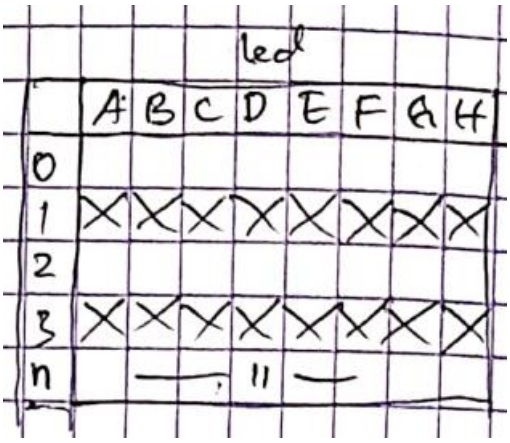
1. Buatlah rangkaian percobaan pada breadboard.
2. Buat kode yang akan digunakan pada percobaan.
3. Compile kode yang sudah dibuat.
4. Jika kode berhasil compile, hubungkan ESP32 ke komputer dengan kabel USB.
5. Pastikan ESP32 terhubung ke komputer dengan mengecek port pada pengelola perangkat.
6. Flash kode yang sudah berhasil dikompilasi ke ESP32.
7. Amati hasil percobaan dan catat pada buku catatan laboratorium.

4. HASIL DAN ANALISIS

Setelah dilakukan simulasi rangkaian percobaan pada praktikum ini, didapatkan data dan hasil analisis sebagai berikut:

4.1 251-OUTPUT DIGITAL

Pada percobaan ini dilakukan percobaan menggukan output GPIO dari ESP32 dengan compiler ESP-IDF. Setelah dilakukan pembuatan rangkaian pada breadboard, dan program diflash ke ESP32, didapatkan hasil pola nyala dan mati LED sebagai berikut:



Kedelapan LED nyala dan mati secara bersamaan dengan delay yang diberikan yaitu sebesar 500ms. Hasil percobaan ini sudah sesuai dengan pola yang diprogram pada ESP32.

Pada program tersebut untuk menggunakan sebuah pin sebagai output digital kita perlu mendefine pin tersebut sebagai output yaitu pada blok kode berikut:

```
#define GPIO_OUTPUT_A 2
...
#define GPIO_OUTPUT_H 23
#define GPIO_OUTPUT_PIN_SEL
((1ULL<<GPIO_OUTPUT_A) |
...
(1ULL<<GPIO_OUTPUT_H))
```

Selanjutnya endefinisikan variabel yang digunakan untuk menentukan waktu delay dalam satuan tick.

```
#define DELAY_MS 500 // waktu delay

const TickType_t xDelay = DELAY_MS /
portTICK_PERIOD_MS;
```

Dari baris program terasebut akan menghasilkan delay pada xDelay sebesar 500ms.

Selanjutnya didalam app main perlu dilakukan konfigurasi yang mana GPIO akan digunakan sebagai output digital

```
gpio_config_t io_conf;
      io_conf.intr_type =
GPIO_INTR_DISABLE;      // tidak
menggunakan interrupt
      io_conf.mode =
GPIO_MODE_OUTPUT;       // mode
output
      io_conf.pin_bit_mask =
GPIO_OUTPUT_PIN_SEL;
      io_conf.pull_down_en = 0;
                  // tidak
menggunakan pull down
      io_conf.pull_up_en = 0;
                  // tidak
menggunakan pull up

      gpio_config(&io_conf);
```

Untuk tiap konfigurasinya sesuai dengan referensi pada landasan teori.

Selanjutnya pada main superloop, diprogram untuk menjalankan output sesuai dengan pola yang diinginkan dengan cara mematikan dan juga menyalakan LED.

```
while (1) {
            // Buatlah kondisi dimana
8 buah LED tersebut (GPIO_OUTPUT_A
hingga GPIO_OUTPUT_H) menyala
bergantian (menggunakanan
gpio_set_level())setiap 0,5 detik
dengan menggunakan vTaskDelay dan
variabel xDelay di atas.


      gpio_set_level(GPIO_OUTPUT_A,
1);
...
      gpio_set_level(GPIO_OUTPUT_H,
1);

            vTaskDelay(xDelay);


      gpio_set_level(GPIO_OUTPUT_A,
0);
...
      gpio_set_level(GPIO_OUTPUT_H,
0);

            vTaskDelay(xDelay);
      }
```

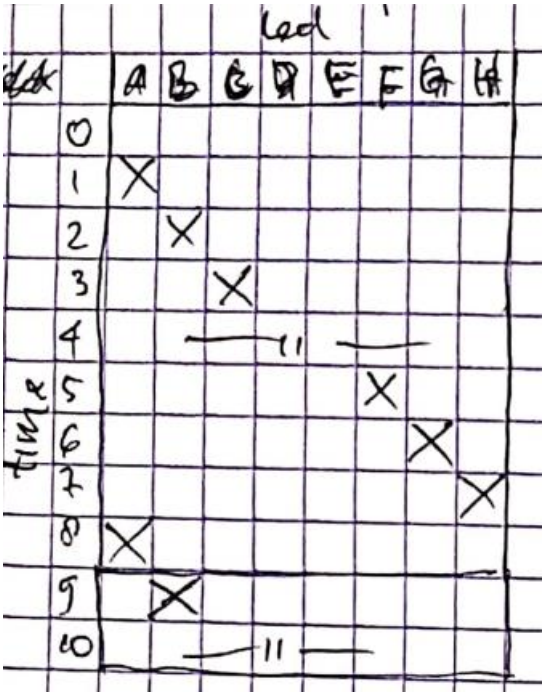
Untuk mengatur output dari GPIO, digunakan fungsi

```
gpio_set_level(pin, state);
```

Dimana pin merupakan pin yang akan diatur outpunya dan state adalah kondisi pin yang diinginkan yang bernilai true/false atau 0/1 dimana 0 untuk kondisi mati dan 1 untuk kondisi aktif.

Dengan program sederhana tersebut dan pemberian delay tiap pergantian statenya dalam main superloop kita akan mendapatkan pola berkedip.

Selanjutnya dilakukan modifikasi program untuk menghasilkan pola berjalan. Berikut ini adalah pola LED yang diamati:

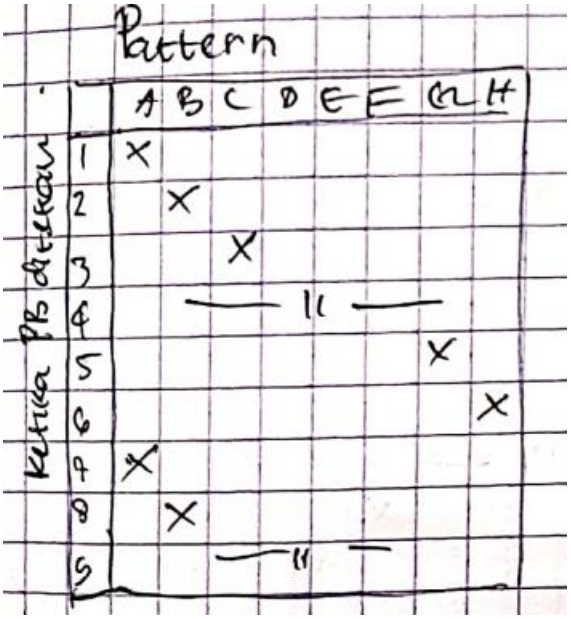


Untuk menghasilkan pola tersebut hanya perlu mengubah kode pada main superloop untuk dengan menambahkan incremental variable sehingga dalam setiap loopnya variable tersebut akan bertambah dan menyalakan led yang berpindah sedangkan LED lainnya dimatikan.

4.2 252-INPUT DIGITAL

Pada percobaan ini dilakukan percobaan untuk menggunakan input digital pada ESP32 dengan membaca input dari sebuah push button.

Pada program percobaan ini akan dibuat LED berjalan ketika led ditekan. Dengan rangkaian dan program yang sudah diflash ke ESP32 didapatkan pola sebagai berikut yang mana pola berubah setiap kali led ditekan.



Pola ini sama dengan pola pada percobaan sebelumya, sehingga untuk setup pin output digital juga sama dengan percobaan sebelumnya. Untuk pengesetan pin input digital untuk membaca tombol adalah sebagai berikut:

```
#define GPIO_INPUT_PB 15
```

Baris kode tersebut kita mendefinisikan penggunaan pin 15 yang dihubungkan ke push button.

```
gpio_config_t io_conf;
      io_conf.intr_type =
GPIO_INTR_DISABLE;
      io_conf.mode =
GPIO_MODE_OUTPUT;
      io_conf.pin_bit_mask =
GPIO_OUTPUT_PIN_SEL;
      io_conf.pull_down_en = 0;
      io_conf.pull_up_en = 0;
      gpio_config(&io_conf);
      io_conf.pin_bit_mask =
GPIO_INPUT_PIN_SEL;
      io_conf.mode = GPIO_MODE_INPUT;
// mode input
      io_conf.pull_up_en = 0; //
menggunakan pull up
      io_conf.pull_down_en = 1; //
menggunakan pull down

      gpio_config(&io_conf);
```

dan pada bagian konfigurasi diatur penggunaan input pull down dengan memberikan nilai 1.

Selanjutnya dilakukan pengecekan apakah tombol ditekan, kemudian menambah nilai pada counter

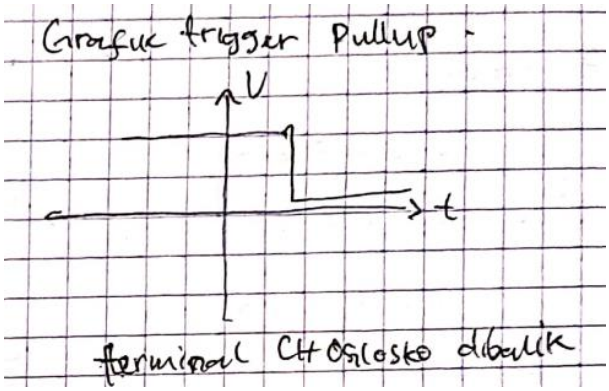
```
if (gpio_get_level(GPIO_INPUT_PB) ==
0) {
      counter++;
      if (counter > 7) {
            counter = 0;
      }
      vTaskDelay(xDelay);
}
```

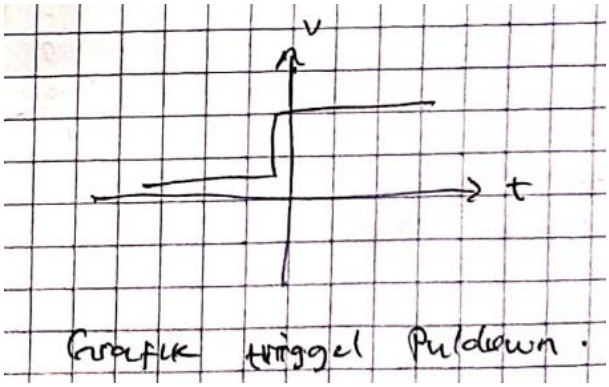
Dan apabila sudah sampai ke led terakhir, counter akan direset. Diberikan juga delay untuk menghindari debouncing

Kemudian setelah mendapatkan nilai couter, cukup menyalakan led dan mematikan led untuk mendapatkan pola led.

```
if (counter == 0) {

      gpio_set_level(GPIO_OUTPUT_A,
1);
} else {

      gpio_set_level(GPIO_OUTPUT_A,
0);

}
```

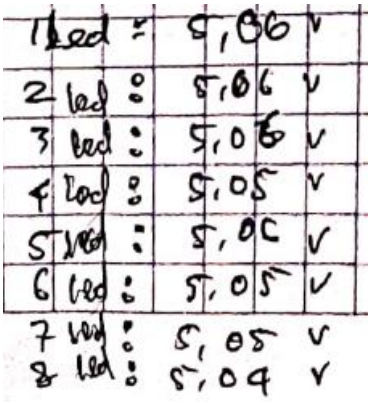
Selanjutnya juga dilakukan pengamatan untuk tegangan push button dengan osiloskop





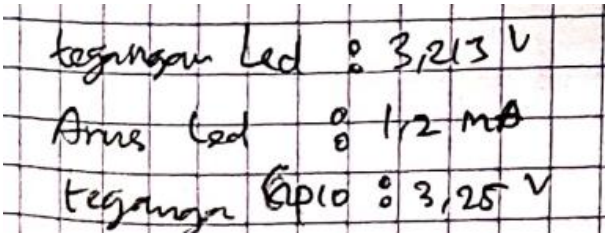
Osiloskop ditrigger ketika push button ditekan. Untuk pin dengan pull-up, terminal osiloskop dibalik untuk dapat mentrigger osiloaskop karena tegangannya terbalik. Grafik output tidak menunjukkan perbedaan yang signifika dari bentuk sinyalnya, namun perbedaan yang paling jelas adalah mode aktif dari kedua konfigurasi tersebut yang berbeda, dimana dengan pull-up push button akan bersifat active low dan pull-down bersifat active high.

Selanjutnya diamati juga tegangan output 5 volt dari ESP32 ketika n buah led dinyalakan.



Tegangan 5V pada ESP32 tidak berubah signifikan, karna tegangan tersebut merupakan tegangan yang sama dari USB yang digunakan untuk menyuplai dan memprogram ESP yang dalam hal ini menggunakan USB port laptop, sehingga 8 buah led tidak terlalu berpengaruh.

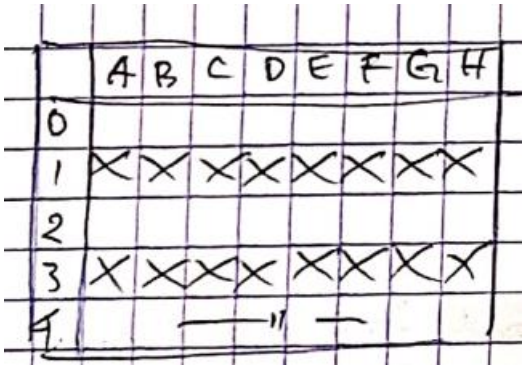
Dari hasil pengukuran, untuk tegangan dan arus pada led adalah sebagai berikut:



Arus pada led dan drop tegangan karena adanya resistor yang digunakan seri dengan led untuk membatasi arus led.

4.3 261-INPUT DIGITAL

Perobaan ini sama dengan percobaan 251 menggunakan Arduino IDE. Pola led pada percobaan ini adalah sebagai berikut:



Kemudian dilakukan modifikasi untuk pola berikutnya:

Pada arduino IDE, setup pin GPIO lebih sederhana yaitu sebagai berikut:

```
pinMode(A, OUTPUT);
pinMode(B, OUTPUT);
pinMode(C, OUTPUT);
pinMode(D, OUTPUT);
pinMode(E, OUTPUT);
pinMode(F, OUTPUT);
pinMode(G, OUTPUT);

pinMode(H, OUTPUT);
```

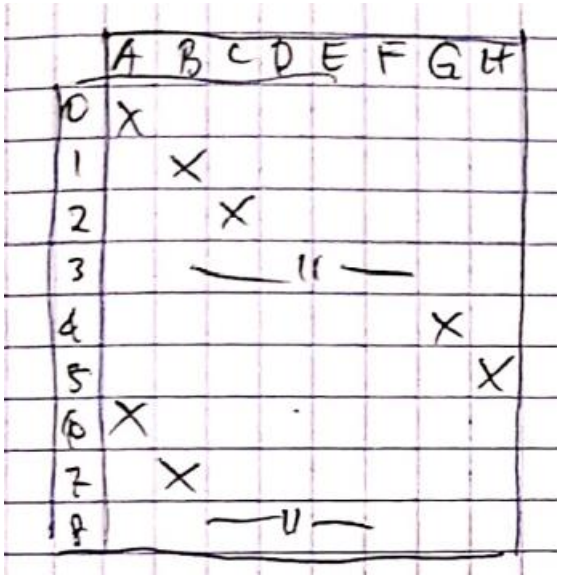
Selanjutnya menyalakan dan mematikan LED dengan digital output set kode sebagai berikut:

```
digitalWrite(A, 1);
delay(500);

digitalWrite(A, 0);
```

Untuk pola berikutnya menggunakan prinsip increment variable, sama dengan percobaan 251 sebelumnya.

4.4 262-OUTPUT DIGITAL

Percobaan ini serupa dengan percobaan 252 namun menggunakan arduino IDE. Pola hasil percobaan ini adalah sebagai berikut:



Pengesetan pin GPIO sebagai output pada arduino ada pada baris kode berikut:

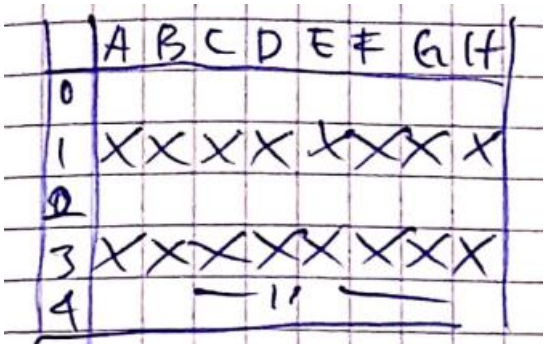
```
pinMode(BUTTON, INPUT_PULLDOWN);
```

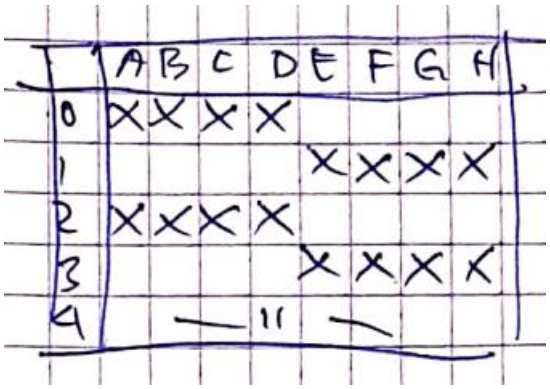
Kita juga dapat mengatur pin tanpa pullup atau pulldown,

4.5 264-TIMER INTERUPT

Percobaan ini adalah menggunakan interupt timer untuk memberikan interupsi pada program dalam durasi waktu tertentu

Pola yang didapatkan adalah sebagai berikut:





Untuk pola kedua merupakan hasil modifikasi pola dari pola defaultnya.

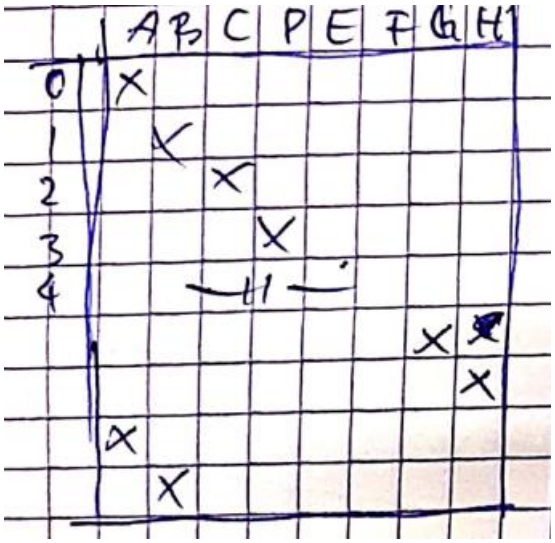
Untuk melakukan setup pada timer interupt ada pada blok kode berikut:

```
hw_timer_t * My_timer = NULL;

bool counter = 0;

void IRAM_ATTR onTimer(){
    if (counter == 0) {
      digitalWrite(A,
!digitalRead(A));
      digitalWrite(B,
!digitalRead(B));
      digitalWrite(C,
!digitalRead(C));
      digitalWrite(D,
!digitalRead(D));
    } else {
      digitalWrite(E,
!digitalRead(E));
      digitalWrite(F,
!digitalRead(F));
      digitalWrite(G,
!digitalRead(G));
      digitalWrite(H,
!digitalRead(H));
    }

    counter = !counter;
}
```

Dibuat sebubah fungsi yang akan dipanggil ketika adanya interupt. Dalam kasus ini tidak perlu ada program pada mainloop.

4.6 265-BUTTON INTERUPT

Percobaan ini menggunakan interupt pada button, pola yang dihasilkan sebagai berikut:



Pengesetan pin sebagai interupt ada pada blok kode berikut:

```
pinMode(15, INPUT_PULLUP);

    attachInterrupt(15,     ledShift,
RISING);
```

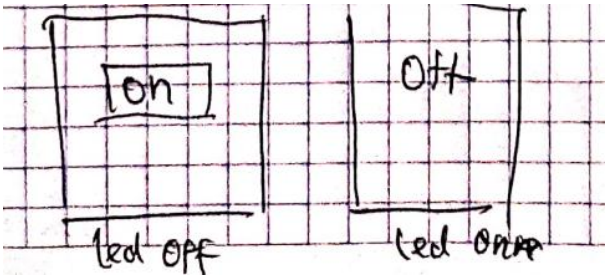
dan untuk fungsi yang dipanggil adalah sebagai berikut:

```
void ledShift(){
    delay(200);
    digitalWrite(ledList[i],
!digitalRead(ledList[i]));
    i++;
    if(i == 8){
        i = 0;
    }
}
```

4.7 266-IoT

Selanjutnya pada percobaan ini adalah IoT dimana penggunaan wifi untuk menyalakan dab mematikan led juga mengirim data state led tersebut ke website

Berikut adalah hasil percobaan IoT:



Berhasil menampilkan button pada web dan mengubah statenya.

5. KESIMPULAN

a. Interuput dapat ditrigger pada rising atau falling edge.

b. ESP32 memiliki internal resistor pullup dan pulldown yang dapat digunakan.

c. Penggunaan output digital dengan arus yang tidak melebihi batas secara bersamaan tidak menurunkan tegangan pada 5V esp secara signifikan

DAFTAR PUSTAKA

[1] Adijarto, Waskita dkk , *Petunjuk Praktikum Sistem Mikropeosesor*, Institut Teknologi Bandung, Bandung, 2023.

LAMPIRAN

1. Source code 251

```
# include <stdio.h> // Jika membutuhkan serial.print, cukup printf
seperti pada program C
#include "driver/gpio.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#define GPIO_OUTPUT_A 2
#define GPIO_OUTPUT_B 4
#define GPIO_OUTPUT_C 5
#define GPIO_OUTPUT_D 18
#define GPIO_OUTPUT_E 19
#define GPIO_OUTPUT_F 21
#define GPIO_OUTPUT_G 22
#define GPIO_OUTPUT_H 23
#define GPIO_OUTPUT_PIN_SEL ((1ULL<<GPIO_OUTPUT_A) |
(1ULL<<GPIO_OUTPUT_B) | (1ULL<<GPIO_OUTPUT_C)| (1ULL<<GPIO_OUTPUT_D)|
(1ULL<<GPIO_OUTPUT_E)| (1ULL<<GPIO_OUTPUT_F)| (1ULL<<GPIO_OUTPUT_G)|
(1ULL<<GPIO_OUTPUT_H))

#define DELAY_MS 500 // isi waktu delay

const TickType_t xDelay = DELAY_MS / portTICK_PERIOD_MS;
void app_main() {

   gpio_config_t io_conf;
   io_conf.intr_type = GPIO_INTR_DISABLE;    // tidak menggunakan
interrupt
   io_conf.mode = GPIO_MODE_OUTPUT;          // mode output
   io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SEL;
   io_conf.pull_down_en = 0;                        // tidak
menggunakan pull down
   io_conf.pull_up_en = 0;                          // tidak
menggunakan pull up
   gpio_config(&io_conf);

   while (1) {
         // Buatlah kondisi dimana 8 buah LED tersebut (GPIO_OUTPUT_A
hingga GPIO_OUTPUT_H) menyala bergantian (menggunakanan
gpio_set_level())setiap 0,5 detik dengan menggunakan vTaskDelay dan
variabel xDelay di atas.

         gpio_set_level(GPIO_OUTPUT_A, 1);
         gpio_set_level(GPIO_OUTPUT_B, 1);
         gpio_set_level(GPIO_OUTPUT_C, 1);
         gpio_set_level(GPIO_OUTPUT_D, 1);
         gpio_set_level(GPIO_OUTPUT_E, 1);
         gpio_set_level(GPIO_OUTPUT_F, 1);
         gpio_set_level(GPIO_OUTPUT_G, 1);
         gpio_set_level(GPIO_OUTPUT_H, 1);

         vTaskDelay(xDelay);

         gpio_set_level(GPIO_OUTPUT_A, 0);
         gpio_set_level(GPIO_OUTPUT_B, 0);
         gpio_set_level(GPIO_OUTPUT_C, 0);
         gpio_set_level(GPIO_OUTPUT_D, 0);
         gpio_set_level(GPIO_OUTPUT_E, 0);
         gpio_set_level(GPIO_OUTPUT_F, 0);
         gpio_set_level(GPIO_OUTPUT_G, 0);
         gpio_set_level(GPIO_OUTPUT_H, 0);
```

```
            vTaskDelay(xDelay);
        }
    }
```

2. Source code 252

```
#include <stdio.h>
#include "driver/gpio.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

#define GPIO_OUTPUT_A 2
#define GPIO_OUTPUT_B 4
#define GPIO_OUTPUT_C 5
#define GPIO_OUTPUT_D 18
#define GPIO_OUTPUT_E 19
#define GPIO_OUTPUT_F 21
#define GPIO_OUTPUT_G 22
#define GPIO_OUTPUT_H 23
#define GPIO_OUTPUT_PIN_SEL ((1ULL<<GPIO_OUTPUT_A) |
(1ULL<<GPIO_OUTPUT_B) | (1ULL<<GPIO_OUTPUT_C)| (1ULL<<GPIO_OUTPUT_D)|
(1ULL<<GPIO_OUTPUT_E)| (1ULL<<GPIO_OUTPUT_F)| (1ULL<<GPIO_OUTPUT_G)|
(1ULL<<GPIO_OUTPUT_H))

#define GPIO_INPUT_PB 15
#define GPIO_INPUT_PIN_SEL (1ULL<<GPIO_INPUT_PB)

#define DELAY_MS 200

const TickType_t xDelay = DELAY_MS / portTICK_PERIOD_MS;

void app_main() {
   gpio_config_t io_conf;
   io_conf.intr_type = GPIO_INTR_DISABLE;
   io_conf.mode = GPIO_MODE_OUTPUT;
   io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SEL;
   io_conf.pull_down_en = 0;
   io_conf.pull_up_en = 0;
   gpio_config(&io_conf);
   io_conf.pin_bit_mask = GPIO_INPUT_PIN_SEL;
   io_conf.mode = GPIO_MODE_INPUT; // mode input
   io_conf.pull_up_en = 0; // menggunakan pull up
   io_conf.pull_down_en = 1; // menggunakan pull down
   gpio_config(&io_conf);

   int counter = 0;

   while (1) {
         if (gpio_get_level(GPIO_INPUT_PB) == 0) {

               counter++;

               if (counter > 7) {
                     counter = 0;
               }

               vTaskDelay(xDelay);

         }

         if (counter == 0) {
```

```
                gpio_set_level(GPIO_OUTPUT_A, 1);
            } else {
                gpio_set_level(GPIO_OUTPUT_A, 0);
            }

            if (counter == 1) {
                gpio_set_level(GPIO_OUTPUT_B, 1);
            } else {
                gpio_set_level(GPIO_OUTPUT_B, 0);
            }

            if (counter == 2) {
                gpio_set_level(GPIO_OUTPUT_C, 1);
            } else {
                gpio_set_level(GPIO_OUTPUT_C, 0);
            }

            if (counter == 3) {
                gpio_set_level(GPIO_OUTPUT_D, 1);
            } else {
                gpio_set_level(GPIO_OUTPUT_D, 0);
            }

            if (counter == 4) {
                gpio_set_level(GPIO_OUTPUT_E, 1);
            } else {
                gpio_set_level(GPIO_OUTPUT_E, 0);
            }

            if (counter == 5) {
                gpio_set_level(GPIO_OUTPUT_F, 1);
            } else {
                gpio_set_level(GPIO_OUTPUT_F, 0);
            }

            if (counter == 6) {
                gpio_set_level(GPIO_OUTPUT_G, 1);
            } else {
                gpio_set_level(GPIO_OUTPUT_G, 0);
            }

            if (counter == 7) {
                gpio_set_level(GPIO_OUTPUT_H, 1);
            } else {
                gpio_set_level(GPIO_OUTPUT_H, 0);
            }
        }
    }
```

3. Source code 253

```
#include <stdio.h>
#include "driver/gpio.h"
#include "driver/timer.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

#define GPIO_OUTPUT_A 2
#define GPIO_OUTPUT_B 4
#define GPIO_OUTPUT_C 5
#define GPIO_OUTPUT_D 18
```

```
#define GPIO_OUTPUT_E 19
#define GPIO_OUTPUT_F 21
#define GPIO_OUTPUT_G 22
#define GPIO_OUTPUT_H 23
#define GPIO_OUTPUT_PIN_SEL ((1ULL<<GPIO_OUTPUT_A) |
(1ULL<<GPIO_OUTPUT_B) | (1ULL<<GPIO_OUTPUT_C)| (1ULL<<GPIO_OUTPUT_D)|
(1ULL<<GPIO_OUTPUT_E)| (1ULL<<GPIO_OUTPUT_F)| (1ULL<<GPIO_OUTPUT_G)|
(1ULL<<GPIO_OUTPUT_H))

#define TIMER_DIVIDER 16
#define TIMER_SCALE (TIMER_BASE_CLK / TIMER_DIVIDER)
#define DELAY_S 0.25
#define NUMBER_OF_LED 8
#define TIMER1_INTERVAL_SEC (DELAY_S * NUMBER_OF_LED)

void app_main(void) {

   gpio_config_t io_conf;
   io_conf.intr_type = GPIO_INTR_POSEDGE;
   io_conf.mode = GPIO_MODE_OUTPUT;
   io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SEL;
   io_conf.pull_down_en = 0;
   io_conf.pull_up_en = 1;
   gpio_config(&io_conf);

   // Timer menghitung ke atas, nanti baru di start, pakai alarm,
namun tanpa reload counter value setelah alarm event.
   timer_config_t config = {
        .divider = TIMER_DIVIDER,
        .counter_dir = TIMER_COUNT_UP,
        .counter_en = TIMER_PAUSE,
        .alarm_en = TIMER_ALARM_EN,
        .auto_reload = TIMER_AUTORELOAD_EN
    };

   // gunakan timer group dan hardware timer yang valid
   timer_init(TIMER_GROUP_0, TIMER_0, &config);
    timer_set_counter_value(TIMER_GROUP_0, TIMER_0, 0);
    timer_set_alarm_value(TIMER_GROUP_0, TIMER_0, (80000000 /
TIMER_DIVIDER));
    timer_enable_intr(TIMER_GROUP_0, TIMER_0);
    timer_start(TIMER_GROUP_0, TIMER_0);

   // Silahkan melengkapi potongan kode berikut ini untuk membuat
program LED menyala bergeser setiap 250ms dan berulang mulai dari LED
paling awal.

   // Jika ingin membuat potongan kode sendiri sangat dipersilahkan
(sekaligus dapat mengubah konfigurasi timer di atas)
   int count = -1;
   double current_time_sec = 0, last_time_sec = 0,
   last_reset_time = 0;

   while (1) {
         timer_get_counter_time_sec(TIMER_GROUP_0, TIMER_0,
&current_time_sec);

         if (current_time_sec - last_time_sec > DELAY_S) {
               count++;
               last_time_sec = current_time_sec;

         } else if (current_time_sec - last_reset_time >
TIMER1_INTERVAL_SEC) {
```

```
                count = -1;
                last_reset_time = current_time_sec;
            }


            gpio_set_level(count, 1);
            vTaskDelay(1);
        }
    }
```

4. Source code 254

```
#include <stdio.h>
#include "driver/gpio.h"
#include "driver/timer.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

#define GPIO_OUTPUT_A 2
#define GPIO_OUTPUT_B 4
#define GPIO_OUTPUT_C 5
#define GPIO_OUTPUT_D 18
#define GPIO_OUTPUT_E 19
#define GPIO_OUTPUT_F 21
#define GPIO_OUTPUT_G 22
#define GPIO_OUTPUT_H 23
#define GPIO_OUTPUT_PIN_SEL ((1ULL<<GPIO_OUTPUT_A) |
(1ULL<<GPIO_OUTPUT_B) | (1ULL<<GPIO_OUTPUT_C)| (1ULL<<GPIO_OUTPUT_D)|
(1ULL<<GPIO_OUTPUT_E)| (1ULL<<GPIO_OUTPUT_F)| (1ULL<<GPIO_OUTPUT_G)|
(1ULL<<GPIO_OUTPUT_H))
//gunakan timer 0 esp32
#define TIMER_USED TIMER_0
#define TIMER_DIVIDER 16
#define TIMER_SCALE (80000000 / TIMER_DIVIDER)
#define DELAY_S (1.0)
#define NUMBER_OF_LED 8

int led_state= 0;
// isi dengan attribut yang membuat interrupt hanya dapat dipanggil pada
IRAM/ROM dan juga isi nama fungsi interruptnya.
void IRAM_ATTR timer_group0_isr(void* para) {
    // semua timer group pada interrupt harus sama dengan app_main
    timer_spinlock_take(TIMER_GROUP_0);
    int timer_idx = (int)para;
    uint32_t timer_intr =
timer_group_get_intr_status_in_isr(TIMER_GROUP_0);
    if (timer_intr & TIMER_INTR_T0) {
        timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_0);
    } else if (timer_intr & TIMER_INTR_T1) {
        timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_1);
    }
    // merubah kondisi LED (menyala menjadi mati, mati menjadi menyala)
setiap kali interrupt ini dipanggil.
    led_state = !led_state;
    gpio_set_level(GPIO_OUTPUT_A, led_state);
    gpio_set_level(GPIO_OUTPUT_B, led_state);
    gpio_set_level(GPIO_OUTPUT_C, led_state);
    gpio_set_level(GPIO_OUTPUT_D, led_state);
    gpio_set_level(GPIO_OUTPUT_E, led_state);
    gpio_set_level(GPIO_OUTPUT_F, led_state);
    gpio_set_level(GPIO_OUTPUT_G, led_state);
```

```
        gpio_set_level(GPIO_OUTPUT_H, led_state);
        timer_group_enable_alarm_in_isr(TIMER_GROUP_0, timer_idx);
        timer_spinlock_give(TIMER_GROUP_0);
    }

    void app_main(void) {
        gpio_config_t io_conf;
        io_conf.intr_type = GPIO_INTR_DISABLE;
        io_conf.mode = GPIO_MODE_OUTPUT;
        io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SEL;
        io_conf.pull_down_en = 0;
        io_conf.pull_up_en = 0;
        gpio_config(&io_conf);

        timer_config_t config = {
        .divider = TIMER_DIVIDER,
        .counter_dir = TIMER_COUNT_UP,
        .counter_en = TIMER_PAUSE,
        .alarm_en = TIMER_ALARM_EN,
        .auto_reload = true,
        };

        timer_init(TIMER_GROUP_0, TIMER_0, &config);
        timer_set_counter_value(TIMER_GROUP_0, TIMER_0, 0);
        timer_set_alarm_value(TIMER_GROUP_0, TIMER_0, (80000000 /
    TIMER_DIVIDER));
        timer_enable_intr(TIMER_GROUP_0, TIMER_0);
        timer_group0_isr(TIMER_GROUP_0, TIMER_0, timer_group0_isr, NULL, 0);
        timer_start(TIMER_GROUP_0, TIMER_0);
        // Buat interrupt yang dapat dipanggil hanya pada IRAM/ROM
        int intr_alloc_flags = ESP_INTR_FLAG_IRAM | ESP_INTR_FLAG_LEVEL1;

        // jangan lupa isi dengan nama fungsi interrupt dan isi
    intr_alloc_flags agar interrupt hanya dapat dipanggil pada IRAM/ROM
        timer_isr_register(TIMER_GROUP_0, TIMER_USED, timer_group0_isr,
    (void*)TIMER_USED, intr_alloc_flags, NULL);
        timer_start(TIMER_GROUP_0, TIMER_USED);
        while (1) {
            vTaskDelay(1);
        };
    }
```

5. Source code 255

```
    #include <stdio.h>
    #include "driver/gpio.h"
    #include "freertos/FreeRTOS.h"
    #include "freertos/task.h"
    #define GPIO_OUTPUT_A 2
    #define GPIO_OUTPUT_B 4
    #define GPIO_OUTPUT_C 5
    #define GPIO_OUTPUT_D 18
    #define GPIO_OUTPUT_E 19
    #define GPIO_OUTPUT_F 21
    #define GPIO_OUTPUT_G 22
    #define GPIO_OUTPUT_H 23

    #define GPIO_OUTPUT_PIN_SEL ((1ULL<<GPIO_OUTPUT_A) |
    (1ULL<<GPIO_OUTPUT_B) | (1ULL<<GPIO_OUTPUT_C)| (1ULL<<GPIO_OUTPUT_D)|
    (1ULL<<GPIO_OUTPUT_E)| (1ULL<<GPIO_OUTPUT_F)| (1ULL<<GPIO_OUTPUT_G)|
    (1ULL<<GPIO_OUTPUT_H));
    #define GPIO_INPUT_PB 15
```

```
#define GPIO_INPUT_PIN_SEL (1ULL<<GPIO_INPUT_PB)
#define ESP_INTR_FLAG_DEFAULT 0
int on_led_count = 0;
// isi dengan attribut yang membuat interrupt hanya dapat dipanggil pada
IRAM/ROM dan juga isi nama fungsi interruptnya.

static void IRAM_ATTR button_isr_handler(void* arg) {
    int gpio_num = (int) arg;
    printf("Button pressed at GPIO %d\n", gpio_num);
}

void app_main() {
    int i=0;
    gpio_config_t io_conf;
    io_conf.intr_type = GPIO_INTR_POSEDGE;
    io_conf.mode = GPIO_MODE_OUTPUT;
    io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SEL;
    io_conf.pull_down_en = GPIO_PULLDOWN_DISABLE;
    io_conf.pull_up_en = GPIO_PULLUP_ENABLE;
    gpio_config(&io_conf);
    io_conf.pin_bit_mask = GPIO_INPUT_PIN_SEL;
    io_conf.mode = GPIO_MODE_INPUT;
    io_conf.intr_type = GPIO_INTR_POSEDGE;
    io_conf.pull_up_en = GPIO_PULLUP_ENABLE;
    gpio_config(&io_conf);

    // Tambahkan potongan kode gpio_install_isr_service() dengan
ESP_INTR_FLAG_DEFAULT dan gpio_isr_handler_add(), jangan lupa untuk
mengisi parameter kedua fungsi/API tersebut
    gpio_install_isr_service(ESP_INTR_FLAG_DEFAULT);
    gpio_isr_handler_add(GPIO_INPUT_PB, button_isr_handler, (void*)
GPIO_INPUT_PB);

    while (1) {
        if(gpio_get_level(GPIO_INPUT_PB) == 0) {
            gpio_set_level(GPIO_OUTPUT_A, 1);
        }
        vTaskDelay(1);
    }

}
```

6. Source code 256

```
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "driver/gpio.h"
#include "lwip/err.h"
#include "lwip/sys.h"
#include <esp_http_server.h>
// SSID tidak perlu diubah, jika ingin mengubah, sebaiknya jangan
mengubah menjadi ssid yang sama dengan yang sudah ada pada jangkauan.

#define EXAMPLE_ESP_WIFI_SSID "for_robot"
#define EXAMPLE_ESP_WIFI_PASS "qwertyuiop123"
#define EXAMPLE_ESP_WIFI_CHANNEL 1
```

```
#define EXAMPLE_MAX_STA_CONN 4
#define LED_PIN 22
#define GPIO_OUTPUT_PIN_SEL (1ULL<<LED_PIN)
static const char* TAG_WIFI = "wifi softAP";
static const char* TAG_SERVER = "webserver";
static void wifi_event_handler(void* arg, esp_event_base_t event_base,
    int32_t event_id, void* event_data) {
        if (event_id == WIFI_EVENT_AP_STACONNECTED) {
            wifi_event_ap_staconnected_t* event =
(wifi_event_ap_staconnected_t*) event_data;
            ESP_LOGI(TAG_WIFI, "station "MACSTR" join, AID=%d",
MAC2STR(event->mac), event->aid);
        } else if (event_id == WIFI_EVENT_AP_STADISCONNECTED) {
            wifi_event_ap_stadisconnected_t* event =
(wifi_event_ap_stadisconnected_t*)event_data;
            ESP_LOGI(TAG_WIFI, "station "MACSTR" leave, AID=%d",
MAC2STR(event->mac), event->aid);
        }
    }

void wifi_init_softap(void) {
    ESP_ERROR_CHECK(esp_netif_init());
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    esp_netif_create_default_wifi_ap();
    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));
    ESP_ERROR_CHECK(esp_event_handler_instance_register(WIFI_EVENT,
ESP_EVENT_ANY_ID, &wifi_event_handler, NULL,NULL));
    wifi_config_t wifi_config = {
        .ap = {
            .ssid = EXAMPLE_ESP_WIFI_SSID,
            .ssid_len = strlen(EXAMPLE_ESP_WIFI_SSID),
            .channel = EXAMPLE_ESP_WIFI_CHANNEL,
            .password = EXAMPLE_ESP_WIFI_PASS,
            .max_connection = EXAMPLE_MAX_STA_CONN,
            .authmode = WIFI_AUTH_WPA_WPA2_PSK
        },
    };

    if (strlen(EXAMPLE_ESP_WIFI_PASS) == 0) {
        wifi_config.ap.authmode = WIFI_AUTH_OPEN;
    }
    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_AP));
    ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_AP, &wifi_config));
    ESP_ERROR_CHECK(esp_wifi_start());
    ESP_LOGI(TAG_WIFI, "wifi_init_softap finished. SSID:%s password:%s
channel:%d",EXAMPLE_ESP_WIFI_SSID,
EXAMPLE_ESP_WIFI_PASS,EXAMPLE_ESP_WIFI_CHANNEL);
}
static esp_err_t hello_get_handler(httpd_req_t* req) {
    char* buf;
    size_t buf_len;
    buf_len = httpd_req_get_hdr_value_len(req, "Host") + 1;
    if (buf_len > 1) {
        buf = malloc(buf_len);
        if (httpd_req_get_hdr_value_str(req, "Host", buf, buf_len)
==ESP_OK) {
            ESP_LOGI(TAG_SERVER, "Found header => Host: %s", buf);
        }
        free(buf);
    }
    buf_len = httpd_req_get_hdr_value_len(req, "Test-Header-2") +1;
    if (buf_len > 1) {
```

```
            buf = malloc(buf_len);
            if (httpd_req_get_hdr_value_str(req, "Test-Header-2", buf,
buf_len) == ESP_OK) {
                ESP_LOGI(TAG_SERVER, "Found header => Test-Header-2: %s",
buf);
            }
            free(buf);
        }
        buf_len = httpd_req_get_hdr_value_len(req, "Test-Header-1") +1;
        if (buf_len > 1) {
            buf = malloc(buf_len);
            if (httpd_req_get_hdr_value_str(req, "Test-Header-1",
buf,buf_len) == ESP_OK) {
                ESP_LOGI(TAG_SERVER, "Found header => Test-Header-1:
%s",buf);
            }
            free(buf);
        }
        buf_len = httpd_req_get_url_query_len(req) + 1;
        if (buf_len > 1) {
            buf = malloc(buf_len);
            if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK){
                ESP_LOGI(TAG_SERVER, "Found URL query => %s", buf);
                if (strcmp(buf, "on") == 0) {
                    gpio_set_level(LED_PIN, 1);
                } else {
                    gpio_set_level(LED_PIN, 0);
                }
                char param[32];
                if (httpd_query_key_value(buf, "query1", param,
sizeof(param)) == ESP_OK) {
                    ESP_LOGI(TAG_SERVER, "Found URL query parameter =>
query1=%s", param);
                }
                if (httpd_query_key_value(buf, "query3", param,sizeof(param))
== ESP_OK) {
                    ESP_LOGI(TAG_SERVER, "Found URL query parameter =>
query3=%s", param);
                }
                if (httpd_query_key_value(buf, "query2", param,sizeof(param))
== ESP_OK) {
                    ESP_LOGI(TAG_SERVER, "Found URL query parameter =>
query2=%s", param);
                }
            }
            free(buf);
        }
        httpd_resp_set_hdr(req, "Custom-Header-1", "Custom-Value-1");
        httpd_resp_set_hdr(req, "Custom-Header-2", "Custom-Value-2");
        const char* resp_str = (const char*)req->user_ctx;
        httpd_resp_send(req, resp_str, strlen(resp_str));
        if (httpd_req_get_hdr_value_len(req, "Host") == 0) {
            ESP_LOGI(TAG_SERVER, "Request headers lost");
        }
        return ESP_OK;
    }
    static const httpd_uri_t hello = {
        .uri = "/",
        .method = HTTP_GET,
        .handler = hello_get_handler,
        .user_ctx = "<!DOCTYPE html><html><head><meta name=\"viewport\"
    content=\"width=device-width, initial-scale=1\"><link rel=\"icon\"
    href=\"data:,\"><style>html { font-family: Helvetica; display: inline-
```

```
    block; margin: 0px auto; text-align: center;}.button { background-color:
    #4CAF50; border: none; color: white; padding: 16px 40px;text-decoration:
    none; font-size: 30px; margin: 2px; cursor: pointer;}.button2
    {background-color: #555555;}</style></head><body><h1>ESP32 Web
    Server</h1><p><a href=\"?on\"><button
    class=\"button\">ON</button></a></p><p><a href=\"?off\"><button
    class=\"button button2\">OFF</button></a></p>"
        };
    esp_err_t http_404_error_handler(httpd_req_t* req,httpd_err_code_t err) {
        if (strcmp("/", req->uri) == 0) {
            httpd_resp_send_err(req, HTTPD_404_NOT_FOUND, "/ URI is not
    available");
            return ESP_OK;
        }
        httpd_resp_send_err(req, HTTPD_404_NOT_FOUND, "Some 404 error
    message");
        return ESP_FAIL;
    }
    static httpd_handle_t start_webserver(void) {
        httpd_handle_t server = NULL;
        httpd_config_t config = HTTPD_DEFAULT_CONFIG();
        ESP_LOGI(TAG_SERVER, "Starting server on port: '%d'",
    config.server_port);
        if (httpd_start(&server, &config) == ESP_OK) {
            ESP_LOGI(TAG_SERVER, "Registering URI handlers");
            httpd_register_uri_handler(server, &hello);
            return server;
        }
        ESP_LOGI(TAG_SERVER, "Error starting server!");
        return NULL;
    }
    void app_main(void) {
        // jangan lupa diisi
        gpio_config_t io_conf;
        io_conf.intr_type = GPIO_INTR_POSEDGE;
        io_conf.mode = GPIO_MODE_OUTPUT;
        io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SEL;
        io_conf.pull_down_en = GPIO_PULLDOWN_DISABLE;
        io_conf.pull_up_en = GPIO_PULLUP_ENABLE;
        gpio_config(&io_conf);
        esp_err_t ret = nvs_flash_init();
        if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret ==
    ESP_ERR_NVS_NEW_VERSION_FOUND) {
            ESP_ERROR_CHECK(nvs_flash_erase());
            ret = nvs_flash_init();
        }
        ESP_ERROR_CHECK(ret);
        ESP_LOGI(TAG_WIFI, "ESP_WIFI_MODE_AP");
        wifi_init_softap();
        static httpd_handle_t server = NULL;
        server = start_webserver();
    }
```

7. Source code 261

```
    #define A 2
    #define B 4
    #define C 5
    #define D 18
    #define E 19
    #define F 21
    #define G 22
```

```
#define H 23

void setup() {
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(E, OUTPUT);
  pinMode(F, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(H, OUTPUT);

}

void loop() {
    // Buatlah kondisi dimana 8 buah LED tersebut (GPIO_OUTPUT_A hingga
GPIO_OUTPUT_H) menyala bergantian (menggunakanan gpio_set_level())setiap
0,5 detik dengan menggunakan vTaskDelay dan variabel xDelay di atas.

    /*
    digitalWrite(A, !digitalRead(A));
    digitalWrite(B, !digitalRead(B));
    digitalWrite(C, !digitalRead(C));
    digitalWrite(D, !digitalRead(D));
    digitalWrite(E, !digitalRead(E));
    digitalWrite(F, !digitalRead(F));
    digitalWrite(G, !digitalRead(G));
    digitalWrite(H, !digitalRead(H));

    delay(500);
    */
    digitalWrite(A, 1);
    delay(500);
    digitalWrite(A, 0);

    digitalWrite(B, 1);
    delay(500);
    digitalWrite(B, 0);

    digitalWrite(C, 1);
    delay(500);
    digitalWrite(C, 0);

    digitalWrite(D, 1);
    delay(500);
    digitalWrite(D, 0);

    digitalWrite(E, 1);
    delay(500);
    digitalWrite(E, 0);

    digitalWrite(F, 1);
    delay(500);
    digitalWrite(F, 0);

    digitalWrite(G, 1);
    delay(500);
    digitalWrite(G, 0);

    digitalWrite(H, 1);
    delay(500);
    digitalWrite(H, 0);
```

```
}
```

8. Source code 262

```
#define A 2
#define B 4
#define C 5
#define D 18
#define E 19
#define F 21
#define G 22
#define H 23
#define BUTTON 15

void setup() {
    pinMode(A, OUTPUT);
    pinMode(B, OUTPUT);
    pinMode(C, OUTPUT);
    pinMode(D, OUTPUT);
    pinMode(E, OUTPUT);
    pinMode(F, OUTPUT);
    pinMode(G, OUTPUT);
    pinMode(H, OUTPUT);

    digitalWrite(A, 1);
    pinMode(BUTTON, INPUT_PULLDOWN);
}

int counter = 0;
int buttonArr[8] = {2, 4, 5, 18, 19, 21, 22, 23};

void loop() {
  if (counter > 7) {
    counter = 0;
  }

  if (digitalRead(BUTTON) == 1) {
    delay(100);
    digitalWrite(buttonArr[counter], 0);
    counter++;
    digitalWrite(buttonArr[counter], 1);
  }
}
```

9. Source code 263

```
```

10. Source code 264

```
#define A 2
#define B 4
#define C 5
#define D 18
#define E 19
#define F 21
```

```
#define G 22
#define H 23

// setup timer interupt timer 0 esp32
hw_timer_t * My_timer = NULL;

bool counter = 0;

void IRAM_ATTR onTimer(){
    if (counter == 0) {
      digitalWrite(A, !digitalRead(A));
      digitalWrite(B, !digitalRead(B));
      digitalWrite(C, !digitalRead(C));
      digitalWrite(D, !digitalRead(D));
    } else {
      digitalWrite(E, !digitalRead(E));
      digitalWrite(F, !digitalRead(F));
      digitalWrite(G, !digitalRead(G));
      digitalWrite(H, !digitalRead(H));
    }

    counter = !counter;
}

void setup() {
    // put your setup code here, to run once:
    pinMode(A, OUTPUT);
    pinMode(B, OUTPUT);
    pinMode(C, OUTPUT);
    pinMode(D, OUTPUT);
    pinMode(E, OUTPUT);
    pinMode(F, OUTPUT);
    pinMode(G, OUTPUT);
    pinMode(H, OUTPUT);

    // setup timer interupt timer 0 esp32
    My_timer = timerBegin(0, 80, true);
    timerAttachInterrupt(My_timer, &onTimer, true);
    timerAlarmWrite(My_timer, 1000000, true);
    timerAlarmEnable(My_timer);
}

void loop() {
    // put your main code here, to run repeatedly:
}
```

11. Source code 265

```
int ledList[8] = {2, 4, 5, 18, 19, 21, 22, 23};

int i = 0;

//led shift when button pressed
void ledShift(){
    delay(200);
    digitalWrite(ledList[i], !digitalRead(ledList[i]));
    i++;
    if(i == 8){
        i = 0;
    }
}
```

```
void setup() {
    // put your setup code here, to run once:
    for (int i = 0; i < 8; i++){
        pinMode(ledList[i], OUTPUT);
    }

    pinMode(15, INPUT_PULLUP);
    attachInterrupt(15, ledShift, RISING);
}

void loop() {
    // put your main code here, to run repeatedly:
}
```

12. Source code 266

```
// Load Wi-Fi library
#include <WiFi.h>
// Replace with your network credentials
const char* ssid = "for_robot";
const char* password = "qwertyuiop123";
// 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 outputState = "off";
// Assign output variables to GPIO pins
const int output = 2;
const int output2 = 4;
// 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(output, OUTPUT);
    pinMode(output2, OUTPUT);
    // Set outputs to LOW
    digitalWrite(output, 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 = "";
        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 on");
                            outputState = "on";
                            digitalWrite(output, HIGH);
                            digitalWrite(output2, HIGH);
                        } else if (header.indexOf("GET /26/off") >= 0) {
                            Serial.println("GPIO off");
                            outputState = "off";
                            digitalWrite(output, LOW);
                            digitalWrite(output2, 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>");

                        // code for interacting with LED Pin
                        // Display current state, and ON/OFF buttons for
GPIO
                        client.println("<p>GPIO - State " + outputState +
"</p>");
                        // If the outputState is off, it displays the ON
button
                        if (outputState=="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>");
                        }

                        // 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("");
    }
}
```