MODUL 3  
ADC, DAC, DAN KOMUNIKASI

FOTO

Ahmad Aziz (13220034)

Asisten: ()

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 menggunakan ADC, DAC dan juga komunikasi pada sistem mikroposessor. Protokol kmaunikasi yang digunakan pada praktikum modul ini adalah serial uart communication protocol, SPI, dan juga I2C. Percobaan yang dilakukan pada praktikum ini diimplementasikan pada microcontroller Arduino dan juga ESP32.

Kata kunci: SPI, UART*,* I2C, ADC, DAC*.*

1. Pendahuluan

ADC (Analog to Digital Converter) dan DAC (Digital to Analog Converter) adalah dua komponen penting dalam dunia elektronika dan komputasi. ADC digunakan untuk mengubah sinyal analog menjadi sinyal digital, sedangkan DAC digunakan untuk mengubah sinyal digital menjadi sinyal analog. Kedua komponen ini sangat diperlukan dalam banyak aplikasi yang melibatkan pengolahan sinyal seperti pemrosesan suara, video, dan sensor. ADC dan DAC memiliki peran yang sangat penting dalam sistem kontrol dan monitoring, pengukuran, komunikasi dan lainnya. Selain itu, kualitas dari ADC dan DAC sangat mempengaruhi akurasi dan kualitas output dari suatu sistem. Oleh karena itu, pengembangan teknologi ADC dan DAC menjadi fokus utama dalam industri elektronik dan teknologi informasi untuk meningkatkan kinerja dan kehandalan dari berbagai aplikasi yang menggunakan sinyal analog dan digital.

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.
5. Studi Pustaka
   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.

A picture containing text, electronics, circuit

Description automatically generated

A picture containing graphical user interface

Description automatically generated

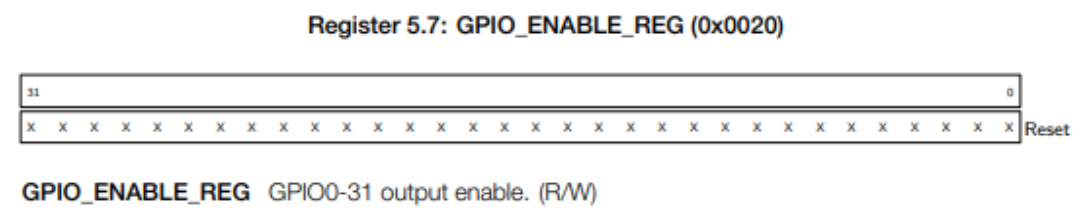
* 1. Register
     1. Konfigurasi I/O ESP32 dan Interrupt

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.

* + 1. 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):



* + 1. 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).

Graphical user interface, application

Description automatically generated

* + 1. INPUT

● GPIO\_OUT\_REG: Membaca langsung input

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

A picture containing text

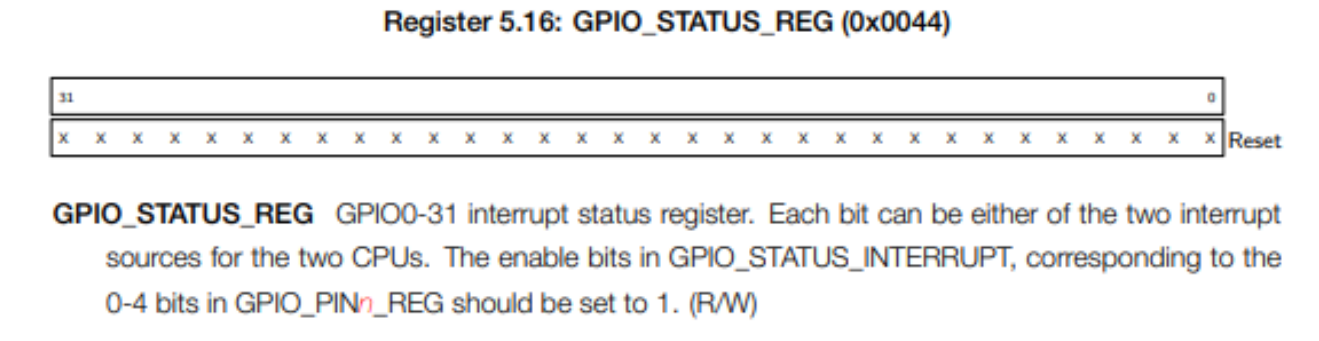
Description automatically generated

* + 1. INTERUPT

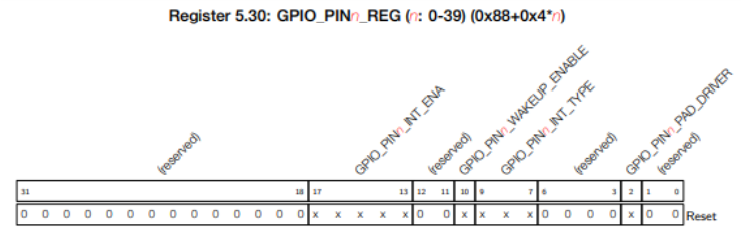
● GPIO\_STATUS\_REG: Membaca status interrupt

● 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).



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



Graphical user interface, text, application, email

Description automatically generated

* + 1. 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).

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

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

Text

Description automatically generated with low confidence

Graphical user interface, text, application

Description automatically generated

* 1. 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 ++.

* 1. API
     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).

* + 1. 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).

* + 1. 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).

* + 1. 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).

* + 1. 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).

1. 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.
8. Hasil dan Analisis

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

* 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:

A picture containing text, crossword puzzle

Description automatically generated

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:

A picture containing text, crossword puzzle

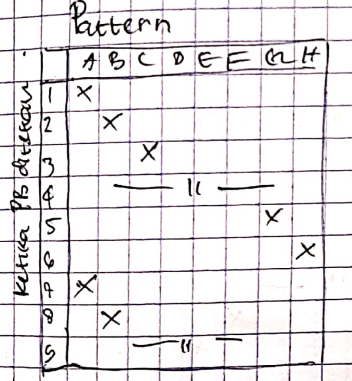
Description automatically generated

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.

* 1. 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**)** **{**

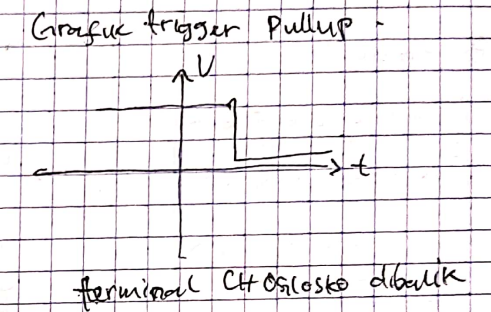
gpio\_set\_level**(**GPIO\_OUTPUT\_A**,** 1**);**

**}** **else** **{**

gpio\_set\_level**(**GPIO\_OUTPUT\_A**,** 0**);**

**}**

Selanjutnya juga dilakukan pengamatan untuk tegangan push button dengan osiloskop

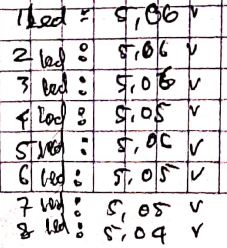


A picture containing text, shoji

Description automatically generated

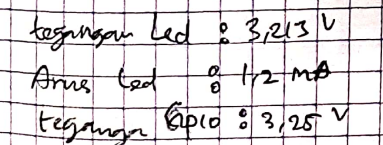
Osiloskop ditrigger ketika push button ditekan. Untuk pin dengan pull-up, terminal osiloskop dibalik untuk dapat mentrigger osiloaskop karena tegangannya terbalik. Grafik output tidak menunjukkan perbedaan yang signifika dari bentuk sinyalnya, namun perbedaan yang paling jelas adalah mode aktif dari 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.

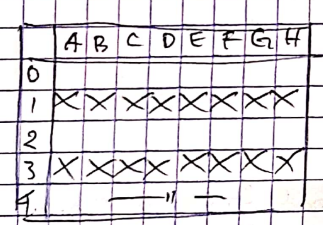
Dari hasil pengukuran, untuk tegangan dan arus pada led adalah sebagai berikut:



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

* 1. 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:

A piece of paper with numbers and letters on it

Description automatically generated with low confidence

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.

* 1. 262-Output Digital

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

Diagram, engineering drawing

Description automatically generated

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,

* 1. 264-Timer Interupt

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

Pola yang didapatkan adalah sebagai berikut:

A picture containing text

Description automatically generated

Text, letter

Description automatically generated

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**));**

**}**

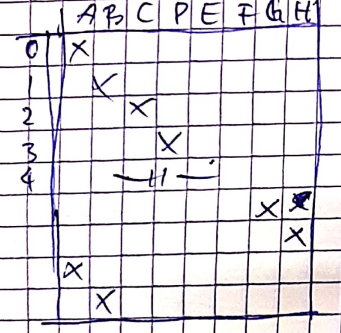
counter **=** **!**counter**;**

**}**

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

* 1. 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**;**

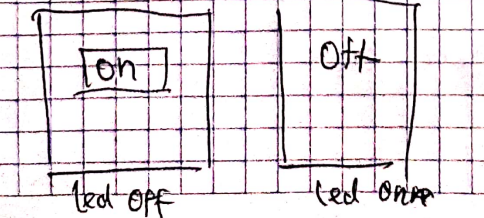
**}**

**}**

* 1. 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.

1. Kesimpulan
2. Interuput dapat ditrigger pada rising atau falling edge.
3. ESP32 memiliki internal resistor pullup dan pulldown yang dapat digunakan.
4. 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**);**  **}**  **}** |

1. 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);  }  }  } |

1. 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);  }  } |

1. 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);  };  } |

1. 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);  }  } |

1. 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**();**  **}** |

1. 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);      } |

1. 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**);**  **}**  **}** |

1. Source code 263

|  |
| --- |
|  |

1. 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:  } |

1. 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:  **}** |

1. 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("");  }  } |