Laporan Tugas 1
Kontrol Lampu LED dengan Finite State Machine (FSM)
EL4121 – Perancangan Sistem Embedded
Semester 1 – 2022/2023

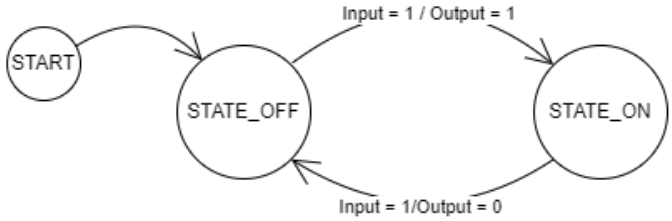
Nama : Vinsensius Liusianto
NIM : 13219036

Link Git : https://github.com/vinlred/FSM_LED

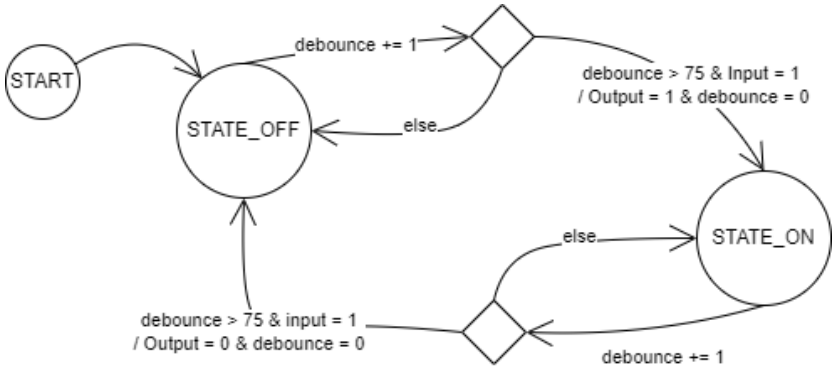
Link Drive Video :
https://drive.google.com/drive/folders/1poWGEOJKW3Bwxbv7eXafpRJTeuxD4tFw?usp=sharing

1 Lampu Toggle

1.1 Desain FSM



Gambar 1.1.1 Finite State Machine Toggle Simulasi



Gambar 1.1.2 Finite State Machine Toggle dengan Debounce

Pada implementasi lampu toggle, dibuat logika finite state machine yang akan mengatur keadaan suatu LED. Keadaan yang diinginkan adalah:

- Kondisi awal LED adalah mati atau STATE_OFF.
- State berubah menjadi nyala atau STATE_ON saat mendapat input berupa push_button dari user saat keadaan STATE_OFF.
- State kembali menjadi mati atau STATE_OFF saat menerima input dalam keadaan STATE_ON.
- Untuk mencegah terjadinya double input dari tekanan button saat implementasi lansung, akan digunakan proses debounce.

Alat akan terdiri dari satu buah LED yang akan diatur keadaannya, serta 1 buah push button sebagai input sinyal untuk mengubah keadaan LED. Pengaturan ini akan diproses dengan mikroprosesor ESP32 yang terhubung lansung dengan LED melalui resistor 1KΩ dan push button.

1.2 Implementasi FSM

Implementasi FSM akan menggunakan bahasa programming C dengan *syntax switch* dan *case*.

FSM tanpa Debounce (Untuk simulasi dan Unit Test) :

```c
#include <stdio.h>
#include <stdlib.h>
#include "fsmtoggle.h"

void fsm(int *state, int input, int *output){
    switch(*state)
    {
    case STATE_OFF:
        {
            if(input==1){
                *state=STATE_ON;
                *output = 1;
            }
            break;
        }
    case STATE_ON:
        {
            if(input==1){
                *state=STATE_OFF;
                *output = 0;
            }
            break;
        }
    default:
        {
        break;
        }
    }
}
```

FSM dengan Debounce (Untuk Implementasi ESP32) :

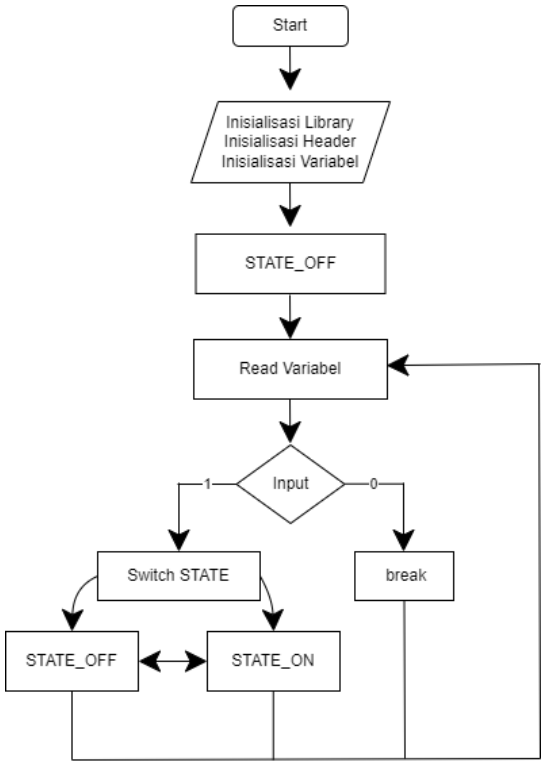
```c
#include <stdio.h>
#include <stdlib.h>
#include "fsmtoggle_debounce.h"

void fsm(int *state, int input, int *output, int *debounce){
    switch(*state)
    {
    case STATE_OFF:
        {
            *debounce += 1;
            if(input==1 && *debounce > 75){
                *state=STATE_ON;
                *output = 1;
                *debounce = 0;
            }
            break;
        }
    case STATE_ON:
```

```
        {
            *debounce += 1;
            if(input==1 && *debounce > 75){
                *state=STATE_OFF;
                *output = 0;
                *debounce = 0;
            }
            break;
        }
    default:
        {
        break;
        }
    }
}
```

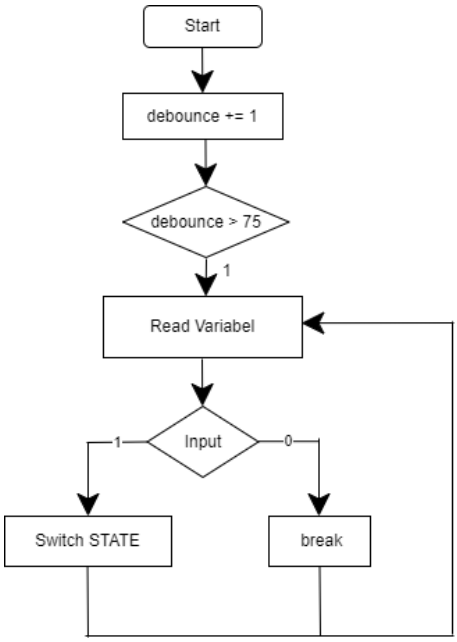
Fungsi fsm() akan menerima masukkan variabel integer berupa state, input, output, dan debounce. Setiap variabel memiliki peran masing-masing berupa:

- *state : variabel integer yang menyimpan nilai state dalam bentuk integer. STATE_OFF bernilai 0 dan STATE_ON bernilai 1. Nilai tersebut didefinisikan pada file header. Variabel dalam bentuk pointer agar nilainya tidak berubah dalam memori antara perulangan program.
- Input : variabel integer yang menyimpan nilai input dengan membaca sinyal yang diterima dari push button. Nilai input adalah 0 dalam kondisi normal dan 1 saat push button ditekan oleh user.
- *output : variabel yang membawa nilai keluaran dari FSM. Nilai variabel akan diatur sesuai dengan keadaan yang ditempati. Output memiliki nilai awal 0, dan berubah menjadi 1 saat push button ditekan dan state berubah dari STATE_OFF menjadi STATE_ON. Saat state berubah sebaliknya, dari STATE_ON menjadi STATE_OFF, nilai output akan diubah menjadi 0.
- *debounce : variabel yang berfungsi sebagai *counter* dalam melakukan proses debounce. Pada saat nilai debounce belum mencapai threshold yang ditentukan, maka input yang dilakukan oleh push button tidak akan diproses oleh mikroprosesor.

Dalam bentuk Flowchart, jalan kerja FSM yang didesain adalah:



Gambar 1.2.1 Flowchart FSM Toggle Overall



Gambar 1.2.2 Flowchart FSM Toggle – State

1.2.1 Simulasi Desktop

Untuk melihat melakukan debugging serta melihat proses kerjanya FSM yang telah didesain, akan dibuat simulasi dengan menggunakan program Desktop yang menirukan proses input nyata. Program ini dibuat dengan bahasa programming C dan menggunakan compiler gcc dengan IDE Code::Blocks.

Program Simulasi :

```
#include <stdio.h>
#include <stdlib.h>
#include "C:\Users\user\Desktop\Scanned\PSE\FSM_LED\Toggle\fsmtoggle\fsmtoggle.h"
#include "C:\Users\user\Desktop\Scanned\PSE\FSM_LED\Toggle\fsmtoggle\fsmtoggle.c"

int main(){
    int state = STATE_OFF;
    int input = 0;
    int output = 0;

    for(int i=0; i<10; i++){
        input = !input; // Alternating Input

        fsm(&state, input, &output);
        printf("Input: %i, State: %i, Output: %i\n",input, state, output);
    }
}
```

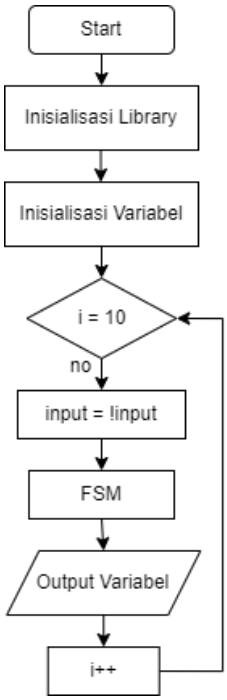
Proses simulasi program akan memanggil fungsi FSM, lalu menirukan proses masukkan dengan nilai input yang bervariasi sehingga mensimulasikan proses input berulang-ulang kali. Dalam simulasi, antar iterasi input, nilai state dan output akan tersimpan dari iterasi sebelumnya sehingga akan tersimulasikan proses perubahan dari state awal yaitu STATE_OFF menjadi STATE_ON, serta dari STATE_ON menjadi STATE_OFF, dan proses perubahan ini akan diulangi berkali-kali. Program ini akan di compile menggunakan Project Code::Blocks, didapatkan hasil:

```
Input: 1, State: 1, Output: 1
Input: 0, State: 1, Output: 1
Input: 1, State: 0, Output: 0
Input: 0, State: 0, Output: 0
Input: 1, State: 1, Output: 1
Input: 0, State: 1, Output: 1
Input: 1, State: 0, Output: 0
Input: 0, State: 0, Output: 0
Input: 1, State: 1, Output: 1
Input: 0, State: 1, Output: 1

Process returned 0 (0x0)   execution time : 0.084 s
Press any key to continue.
```

Gambar 1.2.3 Hasil Simulasi FSM Toggle

Dari hasil simulasi di atas, terlihat Input, state akhir serta output akhir. Pada saat input bernilai 1, maka state akhir akan berubah berikut dengan perubahan nilai output. Hal ini menunjukkan FSM yang dibuat telah berhasil dalam melakukan proses mengubah state LED saat terjadi input push button.

Flowchart dari program simulasi ini:



Gambar 1.2.4 Flowchart Simulasi FSM Toggle

1.2.2 Unit Test di Desktop

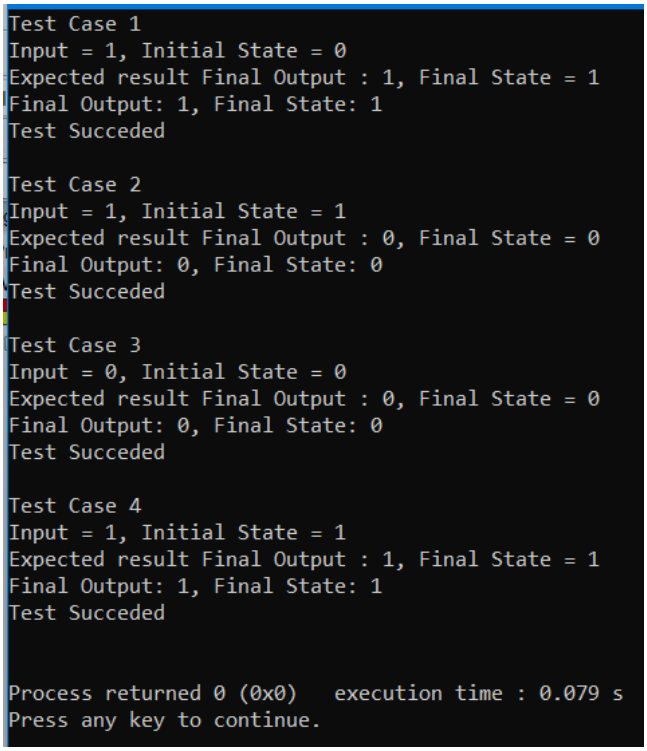
Selain simulasi, juga didesain Unit Test di Desktop dengan menggunakan bahasa C untuk menguji output serta proses kerja dari FSM yang telah dibuat. Unit test akan menguji semua kemungkinan state awal dan input. Hasil dari unit test akan secara detail dan eksplisit menyatakan keberhasilan atau kegagalan dari program dalam menjalankan test case yang diberikan.

Program Test Case:

```
#include <stdio.h>
#include <stdlib.h>
#include
"C:\Users\user\Desktop\Scanned\PSE\FSM_LED\Toggle\fsmtoggle\fsmtoggle.h"
#include
"C:\Users\user\Desktop\Scanned\PSE\FSM_LED\Toggle\fsmtoggle\fsmtoggle.c"

int main(){
    int state = STATE_OFF;
    int input = 0;
    int output = 0;

    printf("Test Case 1\n");
    input = 1;
    output = 0;
    state = STATE_OFF;
    fsm(&state, input, &output);
    printf("Input = 1, Initial State = 0\n");
    printf("Expected result Final Output : 1, Final State = 1\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==1 && state==1){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

    printf("Test Case 2\n");
    input = 1;
    output = 1;
    state = STATE_ON;
    fsm(&state, input, &output);
    printf("Input = 1, Initial State = 1\n");
    printf("Expected result Final Output : 0, Final State = 0\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==0 && state==0){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

    printf("Test Case 3\n");
    input = 0;
    output = 0;
    state = STATE_OFF;
    fsm(&state, input, &output);
    printf("Input = 0, Initial State = 0\n");
    printf("Expected result Final Output : 0, Final State = 0\n");
```

```
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==0 && state==0){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

    printf("Test Case 4\n");
    input = 0;
    output = 1;
    state = STATE_ON;
    fsm(&state, input, &output);
    printf("Input = 1, Initial State = 1\n");
    printf("Expected result Final Output : 1, Final State = 1\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==1 && state==1){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");
}
```

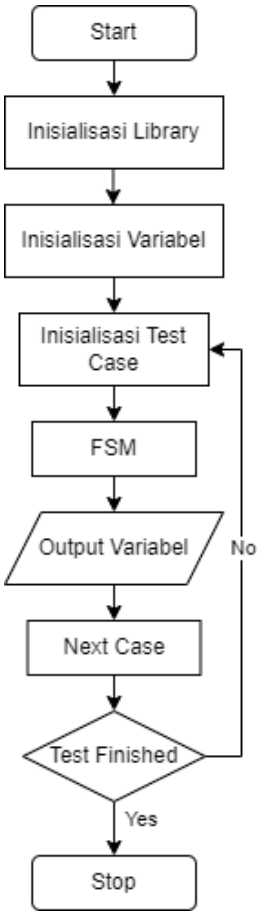
Output dari Unit Test akan menjelaskan input, state awal yang digunakan, hasil yang diharapkan, hasil yang didapatkan, serta keberhasilan atau kegagalan dari test case. Didapatkan hasil dari Unit test terhadap FSM yang telah didesain adalah:



Gambar 1.2.5 Hasil Unit Test FSM Toggle

Dari hasil di atas, telah teruji semua kemungkinan kasus, state awal mati dan hidup dengan input bernilai 0 maupun 1. Hasil yang didapatkan semuanya telah sesuai dengan hasil yang diharapkan sehingga test berhasil dijalankan tanpa kegagalan dengan output yang benar.

Flowchart dari program Unit Test Toggle:



Gambar 1.2.6 Flowchart Unit Test FSM Toggle

1.2.3 Implementasi di ESP32

Implementasi FSM LED Toggle pada ESP32 akan menggunakan ESP-IDF dengan bahasa programming C. Komponen yang digunakan berupa ESP32 DOIT DevKit V1, LED merah, Resistor 1KΩ, push-button, bread-board, serta beberapa buah kabel jumper male-to-male.

Untuk file fsm, digunakan file fsm yang dilengkapi dengan debounce. Program yang diupload ke ESP32 lewat ESP-IDF adalah:

main.c

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

#define GPIO_OUTPUT 4
#define GPIO_OUTPUT_PIN_SEL (1ULL<<GPIO_OUTPUT)
#define GPIO_INPUT_PB 5
#define GPIO_INPUT_PIN_SEL (1ULL<<GPIO_INPUT_PB)

void app_main(){
      gpio_config_t io_conf;
      io_conf.intr_type = 0;
      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 = 1; // menggunakan pull up
        gpio_config(&io_conf);

        int state = 0;
        int output = 0;
        int input = 0;
        int debounce = 0;

        while(1){
              input = !(gpio_get_level(GPIO_INPUT_PB));
              fsm(&state, input, &output, &debounce);
              printf("Input: %i,Output: %i, State: %i\n",
input,output,state);
              gpio_set_level(GPIO_OUTPUT, output);
        }
}
```

fsmtoggle.c

```
#include <stdio.h>
#include <stdlib.h>
#include "fsmtoggle.h"

void fsm(int *state, int input, int *output, int *debounce){
    switch(*state)
    {
    case STATE_OFF:
        {
                  *debounce += 1;
            if(input==1 && *debounce > 75){
                *state=STATE_ON;
                *output = 1;
                        *debounce = 0;
            }
            break;
        }
    case STATE_ON:
        {
                  *debounce += 1;
            if(input==1 && *debounce > 75){
                *state=STATE_OFF;
                *output = 0;
                        *debounce = 0;
            }
            break;
        }
    default:
        {
        break;
        }
    }
}
```

fsmtoggle.h

```
#ifndef FSMTOGGLE_H
#define FSMTOGGLE_H
#define STATE_ON 1
#define STATE_OFF 0
```

```
#endif // FSMTOGGLE_H

void fsm(int *state, int input, int *output, int *debounce);
```

Ketiga file akan digunakan saat melakukan build serta flash dari program ESP32. Resistor terhubung pada kaki positif LED merah dan pin GPIO4 dari ESP32, kaki negatif LED terhubung pada GND, dan Push-button terhubung dengan pin GPIO5 dari ESP32 dan GND.

Program utama akan melakukan inisialisasi library, lalu melakukan setup pada pin GPIO4 dan GPIO5 sesuai dengan mode input dan output-nya. Kemudian melakukan deklarasi variabel, lalu masuk pada looping logikanya. Programa akan terus berulang dan memanggil fungsi FSM pada tiap perulangan. Saat terjadi tekanan button, nilai input akan berubah dari 0 menjadi 1. Fungsi FSM yang terpanggil pada saat input menjadi 1 mengubah state dari LED menjadi STATE_ON melalui fungsi gpio_set_level().

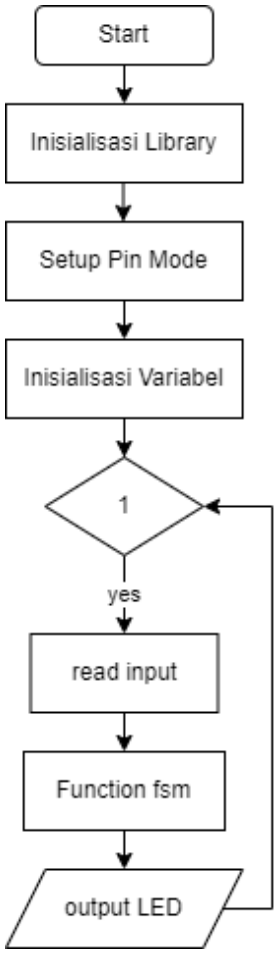
Fungsi fsm dapat diakses oleh program utama melalui file header yang telah di include lewat inisialisasi awal, serta mengatur setting pada file 'CmakeLists.txt'.

Hasil dari implementasi ini dapat dilihat pada video demo.

Link drive untuk video demo:

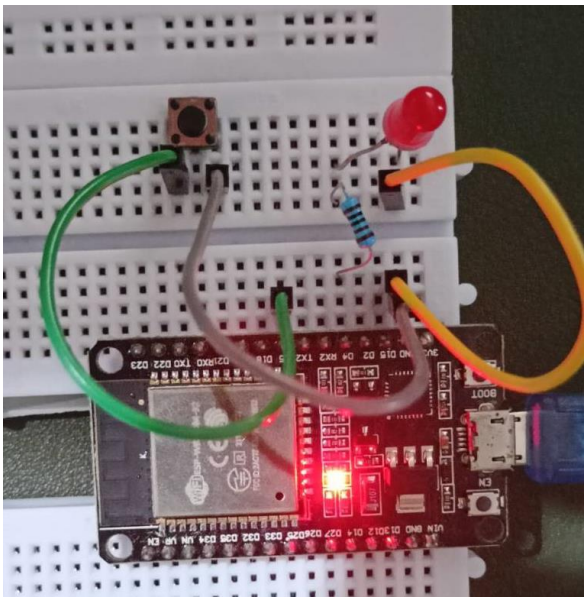
https://drive.google.com/drive/folders/1poWGEOJKW3Bwxbv7eXafpRJTeuxD4tFw?usp=sharing

Flowchart dari program ini:

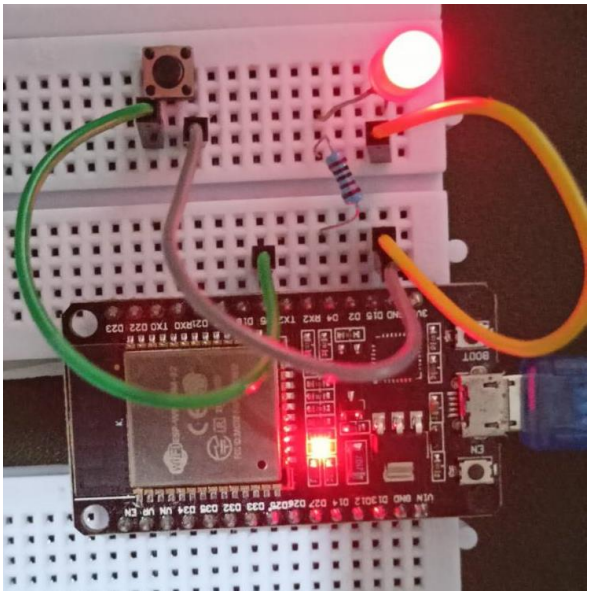


Gambar 1.2.7 Flowchart Implementasi ESP32 FSM Toggle

Foto dari rangkaian ESP32 :



Gambar 1.2.8 Rangkaian Implementasi ESP32 FSM Toggle State Off



Gambar 1.2.9 Rangkaian Implementasi ESP32 FSM Toggle State On

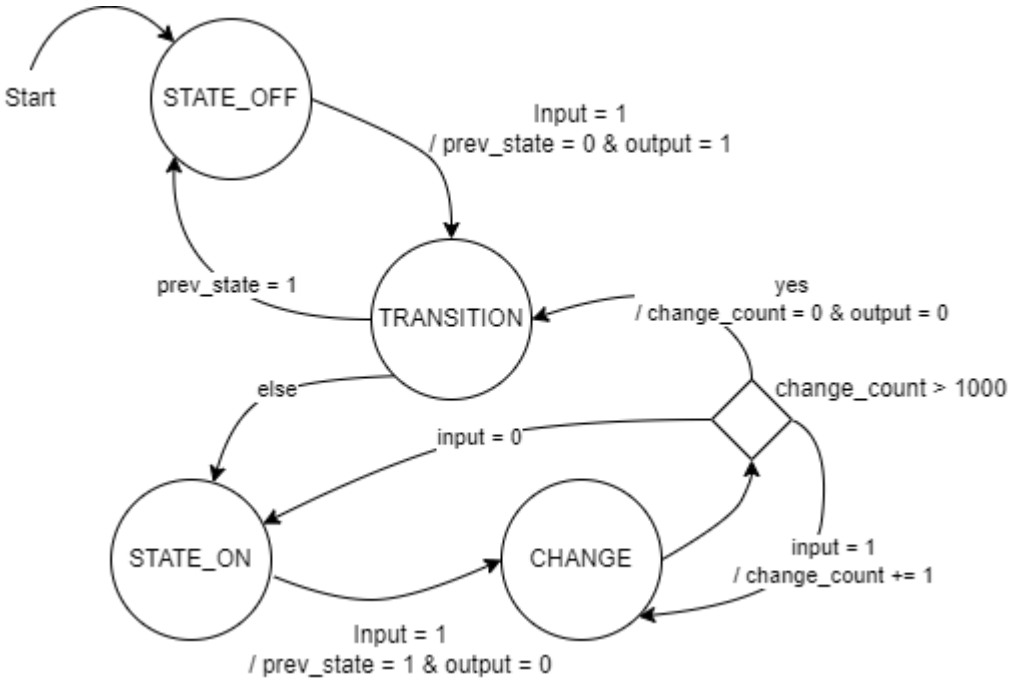
1.3 Kesimpulan & Saran

Dari hasil simulasi, unit test, serta implementasi nyata pada ESP32, desain FSM LED telah berhasil dibuat dan diimplementasikan dengan baik. Nyala LED dapat dikontrol oleh input button sesuai dengan spesifikasi awal yang diberikan. Dari video demo, proses debouncing telah berhasil dilakukan sehingga mencegah terjadinya double input saat button hanya ditekan seketika. Hal ini menunjukkan proses debouncing berjalan dengan seharusnya.

Saran dalam melakukan debouncing adalah pemberian limit pada variabel counter sebab bahasa C rentan terhadap integer overflow. Selain itu penambahan state untuk mencegah double input pada saat button ditekan juga dapat di implementasikan untuk meningkatkan variasi input dari sekedar tekan seketika menjadi tekan dengan durasi berapapun.

2 Sakelar On Off

2.1 Desain FSM



Gambar 2.1.1 FSM On & Off

FSM didesain untuk memenuhi spesifikasi kebutuhan sebagai berikut:

- LED dapat nyala dan mati.
- Kondisi awal mati.
- Kondisi diatur oleh input menggunakan push_button.
- Tekan berapa lama pun dari mati akan menyala.
- Dari menyala minimal tekan selama 1 detik untuk mati.
- Dilengkapi dengan debouncing.

Dari requirement awal, terlihat akan dibutuhkan state-state sebagai berikut:

- State_Off : LED mati, siap menerima input.
- State_On : LED nyala, siap menerima input.
- State_Change : LED nyala, button sedang ditekan, durasi dibawah 1 detik.
- Transition_1 : LED mati, dari State_change, menunggu button dilepas untuk mencegah terjadinya double input dari 1 tekanan tombol.
- Transition_2 : LED nyala, dari State_Off, menunggu button dilepas untuk mencega double input dari 1 tekanan tombol.

State transition 1 dan transition 2 dapat disatukan apabila dimiliki informasi yaitu state sebelumnya. Dimulai dari state_off, saat mendapat input akan pindah pada state transition dengan membawa informasi state sebelumnya berupa state_off. LED sudah dinyalakan, dan menunggu button untuk dilepas. Saat button dilepas, akan pindah ke state_on. Saat button ditekan di state_on, akan pindah ke state_change. Di keadaan ini, saat input masih bernilai 1 atau button tertekan, akan melakukan counting. Apabila mencapai nilai 1000 ms (1 detik), maka LED dimatikan dan pindah ke state transition dengan informasi state sebelumnya berupa state_on. Apabila input sudah menjadi 0 atau button dilepas saat counter masih kurang dari 1000 ms, maka state akan kembali ke state_on.

Dari state_transition, akan menunggu button dilepas untuk pindah kembali ke state_off dan logika diulang dari awal.

Alat akan terdiri dari satu buah LED yang akan diatur keadaannya, serta 1 buah push button sebagai input sinyal untuk mengubah keadaan LED. Pengaturan ini akan diproses dengan mikroprosesor ESP32 yang terhubung lansung dengan LED melalui resistor 1KΩ dan push button.

2.2 Implementasi FSM

FSM diimplementasikan dengan bahasa programming C. File ini disertakan bersama file header untuk proses include pada program simulasi, unit test, serta implementasi lansung pada ESP32. Program FSM yang dibuat adalah:

fsmonoff.c

```
#include <stdio.h>
#include <stdlib.h>
#include "fsmonoff.h"

void fsm(int input,int *prev_state,int *state,int *change_count,int
*output){
	switch(*state)
	{
	case STATE_ON:
	{
		if(input == 0){
		}
		else if(input == 1){
			*state = CHANGE;
			*prev_state = STATE_ON;
			*change_count = 0;
		}
		break;
	}
	case CHANGE:
	{
		if(*change_count <= 1000){ // Jumlah diatur sesuai frekuensi
untuk mencapai 1000 ms
			if(input == 1){
				*change_count += 1;
			}
			else{
				*state = STATE_ON;
			}
		}
		else{
			*state = STATE_TRANS;
			*change_count = 0;
			*output = 0;
		}
		break;
	}
	case STATE_TRANS:
	{
		if(input == 0){
			if(*prev_state == STATE_ON){
				*state = STATE_OFF;
			}
			else{
```

```
                        *state = STATE_ON;
                    }
                }
                break;
        }
        case STATE_OFF:
        {
                if(input == 1){
                        *state = STATE_TRANS;
                        *prev_state = STATE_OFF;
                        *output = 1;
                }
                break;
        }
        default:
        {
                break;
        }
        }
}
```

fsmonoff.h

```
#ifndef FSMONOFF_H
#define FSMONOFF_H
#define STATE_OFF 0
#define STATE_ON 1
#define CHANGE 2
#define STATE_TRANS 3
#endif // FSMONOFF_H

void fsm(int input,int *prev_state,int *state,int *change_count,int
*output);
```

FSM hasil implementasi terdiri dari 4 state dan membutuhkan 5 informasi variabel integer. State yang digunakan:

- STATE_ON : Lampu LED menyala.
- CHANGE : Menghitung lama penekanan button untuk mematikan lampu.
- STATE_TRANS : Mencegah input tambahan hingga button dilepas.
- STATE_OFF : Lampu LED mati.

Variabel yang dibutuhkan oleh fungsi fsm hasil implementasi adalah:

- Input : Nilai lansung dari push button
- Prev_state : Keadaan state sebelumnya, 0 untuk STATE_OFF dan 1 untuk STATE_ON
- State : Keadaan state sekarang
- Change_count : Counter untuk debouncing
- Output : Output berupa keadaan LED, 1 untuk ON dan 0 untuk OFF.

2.2.1 Simulasi Desktop

Simulasi Desktop dilakukan dengan membuat program C yang akan menirukan perilaku menekan tombol dengan variasi berupa durasi menekan dan melepas tombol. Program yang dibuat adalah:

```
#include <stdio.h>
#include <stdlib.h>
#include "C:\Users\user\Desktop\Scanned\PSE\FSM_LED\OnOff\fsm\fsmonoff.c"
#include "C:\Users\user\Desktop\Scanned\PSE\FSM_LED\OnOff\fsm\fsmonoff.h"
```

```c
int main(){
    int state = STATE_OFF;
    int input = 1;
    int output = 0;
    int prev_state = STATE_OFF;
    int change_count = 0;

    int test_case[20] = {100, 100, 100, 100, 750, 750, 750, 750, 1500,
1500, 1500, 1500, 3000, 3000, 3000, 3000, 5000, 5000, 5000, 5000};

    printf("Initial State = 0\n");
    for(int i=0; i<20; i++){
        for (int j=0;j<test_case[i];j++){
            fsm(input, &prev_state, &state, &change_count, &output);
        }
        printf("Input: %i, Duration: %i, State: %i, Output: %i, Prev:
%i\n",input, test_case[i], state, output,prev_state);
        input = !input;
    }
    printf("Finish");
    return 0;
}
```

Pada simulasi, akan divariasikan input bernilai 1 untuk button ditekan dan 0 untuk button tidak tertekan atau lepas, serta durasi dari lama penekanan tombol, akan digunakan 5 durasi yang berbeda, masing-masing diulang sebanyak 2 kali per nilai input.

Hasil dari simulasi ini adalah:

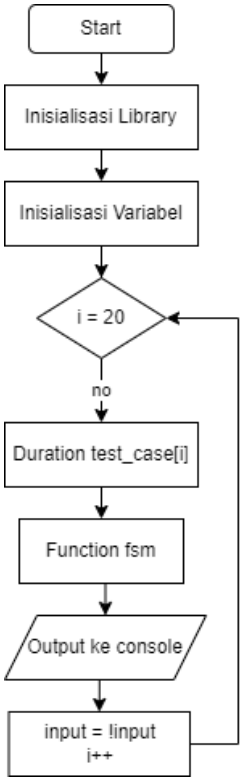
```
Initial State = 0
Input: 1, Duration: 100, State: 3, Output: 1, Prev: 0
Input: 0, Duration: 100, State: 1, Output: 1, Prev: 0
Input: 1, Duration: 100, State: 2, Output: 1, Prev: 1
Input: 0, Duration: 100, State: 1, Output: 1, Prev: 1
Input: 1, Duration: 750, State: 2, Output: 1, Prev: 1
Input: 0, Duration: 750, State: 1, Output: 1, Prev: 1
Input: 1, Duration: 750, State: 2, Output: 1, Prev: 1
Input: 0, Duration: 750, State: 1, Output: 1, Prev: 1
Input: 1, Duration: 1500, State: 3, Output: 0, Prev: 1
Input: 0, Duration: 1500, State: 0, Output: 0, Prev: 1
Input: 1, Duration: 1500, State: 3, Output: 1, Prev: 0
Input: 0, Duration: 1500, State: 1, Output: 1, Prev: 0
Input: 1, Duration: 3000, State: 3, Output: 0, Prev: 1
Input: 0, Duration: 3000, State: 0, Output: 0, Prev: 1
Input: 1, Duration: 3000, State: 3, Output: 1, Prev: 0
Input: 0, Duration: 3000, State: 1, Output: 1, Prev: 0
Input: 1, Duration: 5000, State: 3, Output: 0, Prev: 1
Input: 0, Duration: 5000, State: 0, Output: 0, Prev: 1
Input: 1, Duration: 5000, State: 3, Output: 1, Prev: 0
Input: 0, Duration: 5000, State: 1, Output: 1, Prev: 0
Finish
Process returned 0 (0x0)   execution time : 0.113 s
Press any key to continue.
```

Gambar 2.2.1 Hasil Simulasi FSM On & Off

Dari hasil simulasi, terlihat proses perubahan dari OFF menjadi Transition dengan LED ON akan terjadi saat input bernilai 1 dengan durasi berapapun. Kemudian akan berubah menjadi ON saat button dilepas atau input bernilai 0. Nilai prev_state menjadi 0 sebab berubah dari OFF menjadi ON. Kemudian saat terjadi input bernilai 1 dari state ON dengan durasi kurang dari 1000, akan pindah ke CHANGE sebab masih menunggu counter untuk mencapai 1000. Apabila input berganti menjadi 0, maka kembali ke state ON.

Saat terjadi input 1 dengan durasi lebih dari 1000 saat state ON, maka akan berpindah ke state Transition dengan prev_state bernilai 1. Pada state ini akan ditunggu input bernilai 0 atau button dilepas sebelum berpindah kembali ke state OFF.

Flowchart dari program utama simulasi:



Gambar 2.2.2 Flowchart Simulasi FSM On & Off

2.2.2 Unit Test di Desktop

Selain simulasi, juga didesain Unit Test di Desktop dengan menggunakan bahasa C untuk menguji output serta proses kerja dari FSM yang telah dibuat. Unit test akan menguji semua kemungkinan state awal dan input. Hasil dari unit test akan secara detail dan eksplisit menyatakan keberhasilan atau kegagalan dari program dalam menjalankan test case yang diberikan.

Program Unit test yang dibuat adalah:

```
#include <stdio.h>
#include <stdlib.h>
#include "C:\Users\user\Desktop\Scanned\PSE\FSM_LED\OnOff\fsm\fsmonoff.c"
#include "C:\Users\user\Desktop\Scanned\PSE\FSM_LED\OnOff\fsm\fsmonoff.h"
```

```
int main(){
    int state = STATE_OFF;
    int input = 0;
    int output = 0;
    int prev_state = STATE_OFF;
    int change_count = 0;

    int press_dur = 0;

    printf("Test Case 1\n");
    input = 1;
    output = 0;
    state = STATE_OFF;
    press_dur = 100;

    //printf("Input: %i, Duration: %i, State: %i, Output: %i, Prev:
%i\n",input, press_dur, state, output,prev_state);
    printf("Initial State = %i, Input = %i for %i ms\n",state, input,
press_dur);
    for (int j=0;j<press_dur;j++){
        fsm(input, &prev_state, &state, &change_count, &output);
    }
    printf("Expected result Final Output : LED_ON (1), Final State =
Transition (3)\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==1 && state==3){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

    printf("Test Case 2\n");
    input = 1;
    output = 0;
    state = STATE_OFF;
    press_dur = 10000;

    //printf("Input: %i, Duration: %i, State: %i, Output: %i, Prev:
%i\n",input, press_dur, state, output,prev_state);
    printf("Initial State = %i, Input = %i for %i ms\n",state, input,
press_dur);
    for (int j=0;j<press_dur;j++){
        fsm(input, &prev_state, &state, &change_count, &output);
    }
    printf("Expected result Final Output : LED_ON (1), Final State =
Transition (3)\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==1 && state==3){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

    printf("Test Case 3\n");
    input = 1;
```

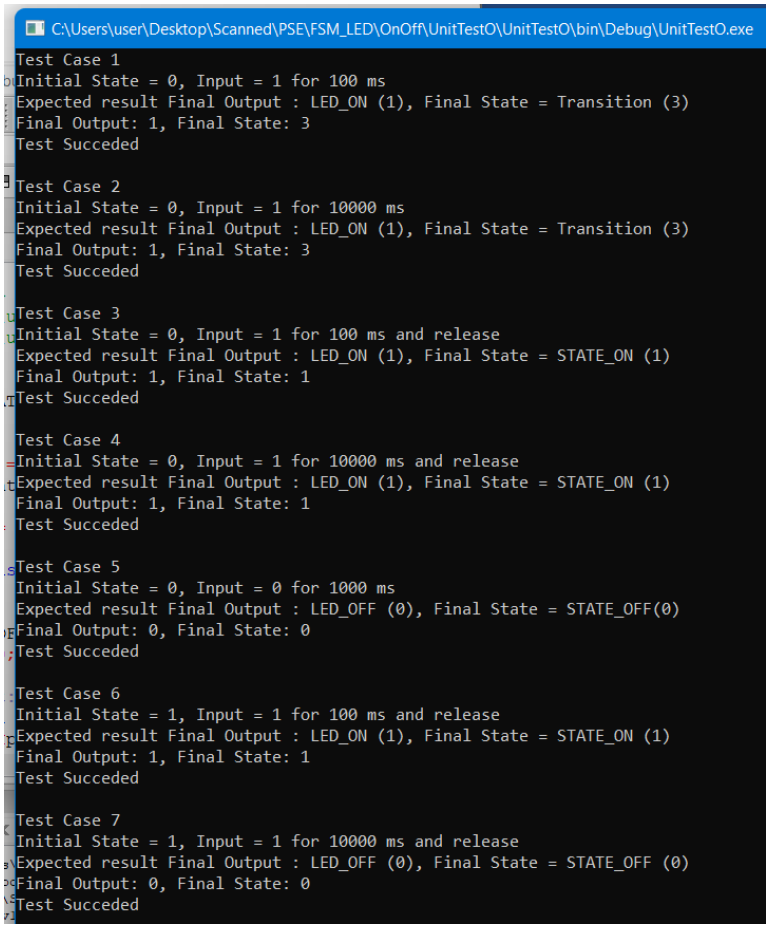
```
    output = 0;
    state = STATE_OFF;
    press_dur = 100;

    //printf("Input: %i, Duration: %i, State: %i, Output: %i, Prev:
%i\n",input, press_dur, state, output,prev_state);
    printf("Initial State = %i, Input = %i for %i ms and
release\n",state, input, press_dur);
    for (int j=0;j<press_dur;j++){
        fsm(input, &prev_state, &state, &change_count, &output);
    }
    input = 0;
    fsm(input, &prev_state, &state, &change_count, &output);
    printf("Expected result Final Output : LED_ON (1), Final State =
STATE_ON (1)\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==1 && state==1){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

    printf("Test Case 4\n");
    input = 1;
    output = 0;
    state = STATE_OFF;
    press_dur = 10000;

    //printf("Input: %i, Duration: %i, State: %i, Output: %i, Prev:
%i\n",input, press_dur, state, output,prev_state);
    printf("Initial State = %i, Input = %i for %i ms and
release\n",state, input, press_dur);
    for (int j=0;j<press_dur;j++){
        fsm(input, &prev_state, &state, &change_count, &output);
    }
    input = 0;
    fsm(input, &prev_state, &state, &change_count, &output);
    printf("Expected result Final Output : LED_ON (1), Final State =
STATE_ON (1)\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==1 && state==1){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

        printf("Test Case 5\n");
    input = 0;
    output = 0;
    state = STATE_OFF;
    press_dur = 1000;

    //printf("Input: %i, Duration: %i, State: %i, Output: %i, Prev:
%i\n",input, press_dur, state, output,prev_state);
```

```
    printf("Initial State = %i, Input = %i for %i ms\n",state, input,
press_dur);
    for (int j=0;j<press_dur;j++){
        fsm(input, &prev_state, &state, &change_count, &output);
    }
    printf("Expected result Final Output : LED_OFF (0), Final State =
STATE_OFF(0)\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==0 && state==0){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

    printf("Test Case 6\n");
    input = 1;
    output = 1;
    state = STATE_ON;
    press_dur = 100;

    //printf("Input: %i, Duration: %i, State: %i, Output: %i, Prev:
%i\n",input, press_dur, state, output,prev_state);
    printf("Initial State = %i, Input = %i for %i ms and
release\n",state, input, press_dur);
    for (int j=0;j<press_dur;j++){
        fsm(input, &prev_state, &state, &change_count, &output);
    }
    input = 0;
    fsm(input, &prev_state, &state, &change_count, &output);
    printf("Expected result Final Output : LED_ON (1), Final State =
STATE_ON (1)\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==1 && state==1){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

    printf("Test Case 7\n");
    input = 1;
    output = 1;
    state = STATE_ON;
    press_dur = 10000;

    //printf("Input: %i, Duration: %i, State: %i, Output: %i, Prev:
%i\n",input, press_dur, state, output,prev_state);
    printf("Initial State = %i, Input = %i for %i ms and
release\n",state, input, press_dur);
    for (int j=0;j<press_dur;j++){
        fsm(input, &prev_state, &state, &change_count, &output);
    }
    input = 0;
    fsm(input, &prev_state, &state, &change_count, &output);
    printf("Expected result Final Output : LED_OFF (0), Final State =
STATE_OFF (0)\n");
```

```
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==0 && state==0){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

    printf("Test Case 8\n");
    input = 1;
    output = 1;
    state = STATE_ON;
    press_dur = 10000;

    //printf("Input: %i, Duration: %i, State: %i, Output: %i, Prev:
%i\n",input, press_dur, state, output,prev_state);
    printf("Initial State = %i, Input = %i for %i ms\n",state, input,
press_dur);
    for (int j=0;j<press_dur;j++){
        fsm(input, &prev_state, &state, &change_count, &output);
    }
    printf("Expected result Final Output : LED_OFF (0), Final State =
TRANSITION (3)\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==0 && state==3){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

    printf("Test Case 9\n");
    input = 1;
    output = 1;
    state = STATE_ON;
    press_dur = 100;

    //printf("Input: %i, Duration: %i, State: %i, Output: %i, Prev:
%i\n",input, press_dur, state, output,prev_state);
    printf("Initial State = %i, Input = %i for %i ms\n",state, input,
press_dur);
    for (int j=0;j<press_dur;j++){
        fsm(input, &prev_state, &state, &change_count, &output);
    }
    printf("Expected result Final Output : LED_ON (1), Final State =
CHANGE (2)\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==1 && state==2){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");

    printf("Test Case 10\n");
```

```
    input = 1;
    output = 1;
    state = STATE_ON;
    press_dur = 100;

    //printf("Input: %i, Duration: %i, State: %i, Output: %i, Prev:
%i\n",input, press_dur, state, output,prev_state);
    printf("Initial State = %i, Input = %i for %i ms and
release\n",state, input, press_dur);
    for (int j=0;j<press_dur;j++){
        fsm(input, &prev_state, &state, &change_count, &output);
    }
    input = 0;
    fsm(input, &prev_state, &state, &change_count, &output);
    printf("Expected result Final Output : LED_ON (1), Final State =
STATE_ON (1)\n");
    printf("Final Output: %i, Final State: %i\n",output,state);
    if(output==1 && state==1){
        printf("Test Succeded\n");
    }
    else{
        printf("Test Failed\n");
    }

    printf("\n");
}
```

Unit test untuk FSM On&Off akan menampilkan hasil yang lebih detail mengenai keadaan awal yang diujikan, keadaan yang diharapkan, serta keadaan akhir. Apabila keadaan akhir sesuai dengan keadaan yang diharapkan, maka test-case tersebut dapat dinyatakan berhasil, sedangkan apabila berbeda, maka dinyatakan gagal. Hasil dari program unit test ini adalah:

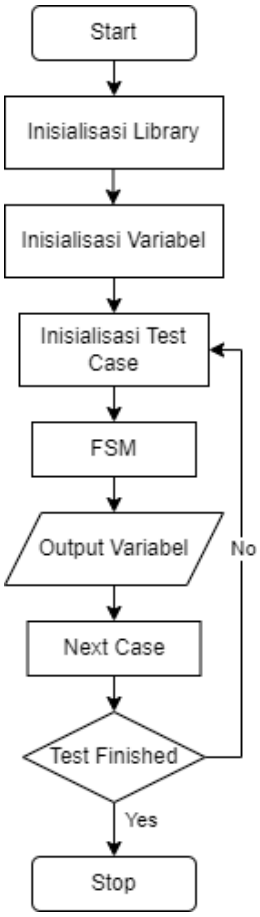


```
Test Case 8
Initial State = 1, Input = 1 for 10000 ms
Expected result Final Output : LED_OFF (0), Final State = TRANSITION (3)
Final Output: 0, Final State: 3
Test Succeded

Test Case 9
Initial State = 1, Input = 1 for 100 ms
Expected result Final Output : LED_ON (1), Final State = CHANGE (2)
Final Output: 1, Final State: 2
Test Succeded

Test Case 10
Initial State = 1, Input = 1 for 100 ms and release
Expected result Final Output : LED_ON (1), Final State = STATE_ON (1)
Final Output: 1, Final State: 1
Test Succeded


Process returned 0 (0x0)   execution time : 0.132 s
Press any key to continue.
```

Gambar 2.2.3 Hasil Unit Test FSM On & Off

Dari hasil unit test, dilihat hasil yang sama dengan simulasi, namun dinyatakan secara eksplisit pada console. Terjadi proses yang sama antara perubahan dari satu state ke state lainnya, sesuai dengan state awal, input, durasi input, dan apakah button dilepaskan atau tidak. Hasil yang terprediksi akan dicantumkan pada expected result, dan apabila hasil output akhir sama dengan expected result, maka test-case yang dijalani dinyatakan berhasil.

Jalan kerja dari Unit Test yang dibuat:



Gambar 2.2.4 Flowchart Unit Test FSM On & Off

2.2.3 Implementasi di ESP32

Implementasi ESP32 dilakukan dengan membuat program bahasa C mengikuti template yang ada untuk ESP-IDF. Kemudian akan diatur frekuensi program main.c di ESP32 untuk melakukan looping dengan durasi minimal yang memungkinkan yaitu 10 ms tiap perulangan, sehingga akan diatur ulang bagian FSM untuk melakukan debouncing hingga counter mencapai 1000 ms atau 100 kali perulangan.

Program yang dibuat:

main.c

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

#define GPIO_OUTPUT 4
#define GPIO_OUTPUT_PIN_SEL (1ULL<<GPIO_OUTPUT)
#define GPIO_INPUT_PB 5
#define GPIO_INPUT_PIN_SEL (1ULL<<GPIO_INPUT_PB)

const TickType_t xDelay = 10 / portTICK_PERIOD_MS; // Set Frekuensi 1/10
ms = 100 Hz

void app_main(){
	gpio_config_t io_conf;
	io_conf.intr_type = 0;
	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 = 1; // menggunakan pull up
	gpio_config(&io_conf);

	int input = 0;
	int prev_state = 0;
	int state = 0;
	int output = 0;
	int change_count = 0;

	while(1){
		input = !(gpio_get_level(GPIO_INPUT_PB));
		fsm(input, &prev_state, &state, &change_count,&output);
		printf("Input: %i,Output: %i, State: %i\n",
input,output,state);
		gpio_set_level(GPIO_OUTPUT, output);
		vTaskDelay(xDelay);
	}
}
```

Variabel xDelay dan vTaskDelay berfungsi untuk mengatur frekuensi perulangan program oleh ESP yang dibutuhkan untuk proses debouncing 1 detik untuk mematikan lampu LED.

fsmonoff.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "fsmonoff.h"

void fsm(int input,int *prev_state,int *state,int *change_count,int
*output){
	switch(*state)
	{
	case STATE_ON:
	{
		if(input == 0){
		}
		else if(input == 1){
			*state = CHANGE;
			*prev_state = STATE_ON;
			*change_count = 0;
		}
		break;
	}
	case CHANGE:
	{
		if(*change_count <= 100){ // Jumlah diatur sesuai frekuensi
untuk mencapai 1000 ms, Frek 100 Hz, 100 perulangan
			if(input == 1){
				*change_count += 1;
			}
			else{
				*state = STATE_ON;
			}
		}
		else{
			*state = STATE_TRANS;
			*change_count = 0;
			*output = 0;
		}
		break;
	}
	case STATE_TRANS:
	{
		if(input == 0){
			if(*prev_state == STATE_ON){
				*state = STATE_OFF;
			}
			else{
				*state = STATE_ON;
			}
		}
		break;
	}
	case STATE_OFF:
	{
		if(input == 1){
			*state = STATE_TRANS;
			*prev_state = STATE_OFF;
			*output = 1;
		}
		break;
	}
	default:
	{
```

```
            break;
        }
        }
}
```

fsmonoff.h

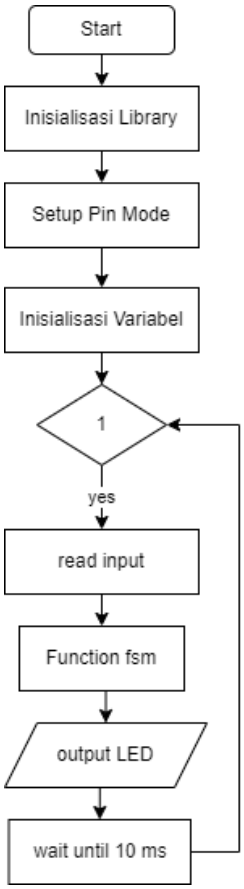
```
#ifndef FSMONOFF_H
#define FSMONOFF_H
#define STATE_OFF 0
#define STATE_ON 1
#define CHANGE 2
#define STATE_TRANS 3
#endif // FSMONOFF_H

void fsm(int input,int *prev_state,int *state,int *change_count,int
*output);
```

Selain file-file tersebut, juga akan diubah isi dari file 'CmakeLists.txt' untuk meng-include file fsmonoff.c dan fsmonoff.h sehingga dapat dikenali oleh compiler ESP-IDF saat proses build dan flash ke ESP32. Pin untuk LED adalah GPIO4 yang terhubung lewat resistor menuju kaki positif LED, kaki negatif terhubung dengan GND, dan button terhubung lansung pada GPIO5 dan GND.

Proses ini serupa dengan implementasi FSM Toggle, tetapi dilengkapi dengan pengaturan frekuensi sebagai basis dari nilai counter debouncing yang dibutuhkan untuk melakukan input hold 1 detik saat mematikan LED.

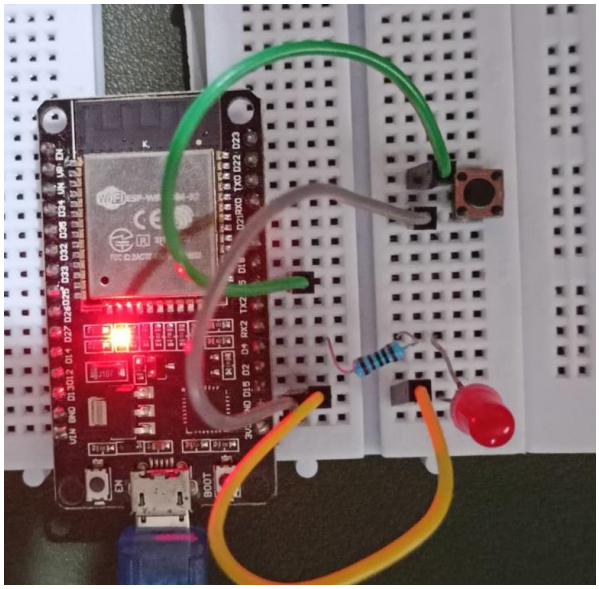
Hasil dari program ini dalam bentuk video demo yang ada pada link:
https://drive.google.com/drive/folders/1poWGEOJKW3Bwxbv7eXafpRJTeuxD4tFw?usp=sharing

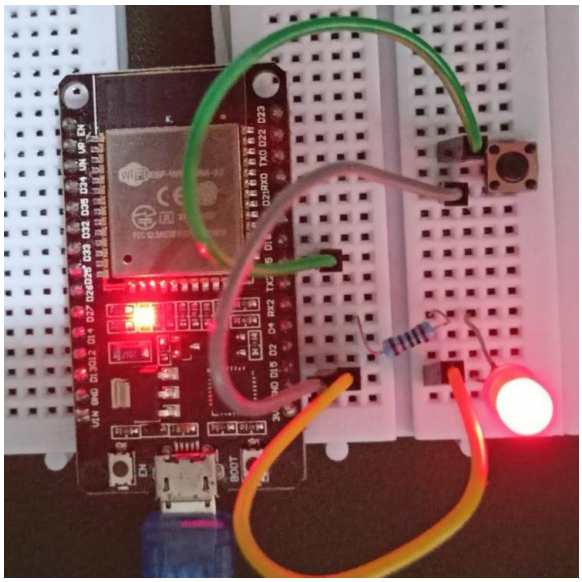
Flowchart dari implementasi FSM LED On&Off:



Gambar 2.2.5 Flowchart Implementasi ESP32 FSM On & Off

Foto sama dengan implementasi FSM Toggle sebab tidak terdapat perubahan rangkaian.





Gambar 2.2.6 & 2.2.7 Rangkaian Implementasi ESP32 FSM On & Off
LED Off **LED On**

2.3 Kesimpulan & Saran

Dari hasil simulasi, unit test, serta implementasi nyata pada ESP32, desain FSM LED telah berhasil dibuat dan diimplementasikan dengan baik. Nyala LED dapat dikontrol oleh input button sesuai dengan spesifikasi awal yang diberikan. Dari video demo, proses debouncing telah berhasil dilakukan dengan akurat sekitar 0.99 detik dari waktu penekanan hingga LED dimatikan. Hal ini menunjukkan proses pengaturan frekuensi serta debouncing berjalan dengan seharusnya.

Saran dalam melakukan debouncing adalah pemberian limit pada variabel counter sebab bahasa C rentan terhadap integer overflow.

Referensi

[1] https://github.com/waskita/2022-fsm-demo

[2] Kuliah EL4121 Perancangan Sistem Embedded

[3] Adijarto. Waskita d.k.k., LDTE, Petunjuk Praktikum Sistem Mikroprosesor, STEI, Institut Teknologi Bandung, Bandung, 2022.