Tugas Lampu Geser dengan Cascade Composition

Disusun oleh:

13219036
Vinsensius Liusianto

INSTITUT TEKNOLOGI BANDUNG

2022

Daftar Isi

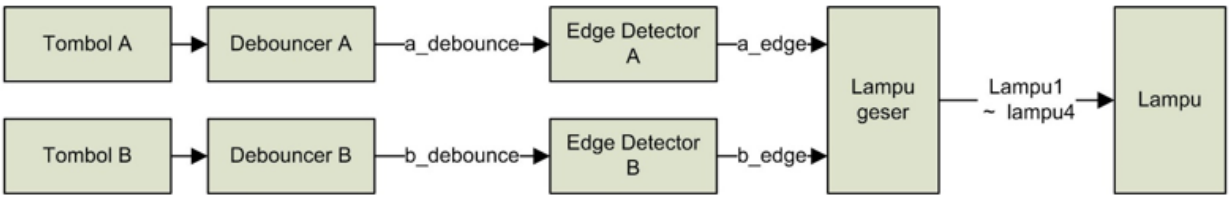
Daftar Gambar

1 Spesifikasi

Spesifikasi dari sistem yang dibuat dalam tugas ini sebagai berikut:

- 4 lampu LED
- 2 push-button untuk geser kiri dan kanan
- LED menyala satu per satu
- Saat tombol kiri ditekan, LED yang menyala bergeser ke kiri
- Saat tombol kanan ditekan, LED yang menyala bergeser ke kanan
- Sistem di lengkapi dengan debouncing.

Desain mengikuti diagram logika berikut.



Gambar 1 Blok Diagram Sistem

Akan ada dua proses yang berjalan bersamaan dalam menerima input dari kedua button, yang mengirimkan informasi menuju proses logika lampu geser yang menentukan apakah lampu bergeser atau tidak.

2 Desain FSM

Dari spesifikasi dan blok diagram sistem, akan disimpulkan bahwa sistem ini terdiri dari beberapa sub-sistem sebagai berikut:

- Debouncer button saat button ditekan
- Edge detector button yang menerima data dari debouncer
- Logika lampu geser berdasarkan input dari edge detector

Sebab logika debouncer dan edge detector berhubungan erat satu sama lain dalam menerima input dari push-button, kedua sub-sistem ini disatukan menjadi satu FSM, kemudian logika lampu geser menjadi FSM kedua.

FSM Push-Button:

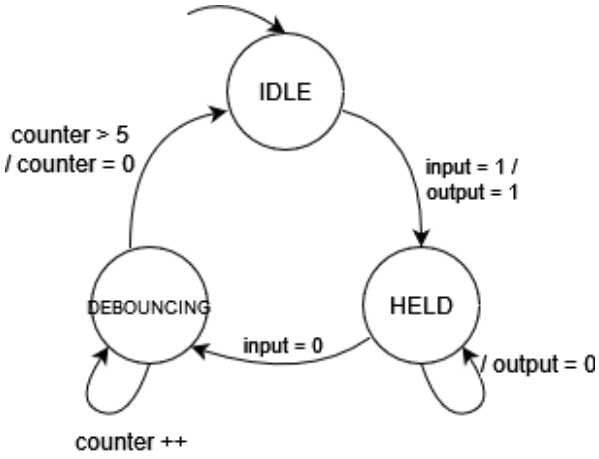
Untuk FSM push-button, diperlukan beberapa state untuk memenuhi spesifikasi, sebagai berikut:

- State idle atau menunggu input.
- State menerima rising edge atau button ditekan

- State tertahan
- State debouncing

State menerima rising edge adalah keluaran seketika dari state idle, kemudian lansung memasuki state tertahan selama push-button masih tertekan, kemudian memasuki state debouncing saat dilepas untuk mencegah terjadinya double-input akibat pelepasan button.

FSM yang didesain untuk memenuhi spesifikasi push-button sebagai berikut:



Gambar 2 FSM push-button

Program untuk fsm push-button dalam file `fsmbutton.h` sebagai berikut:

```
#ifndef FSMBUTTON_H
#define FSMBUTTON_H

#include <stdio.h>
#include <stdlib.h>

#define IDLE        0
#define DEBOUNCING  1
#define HELD        2

void fsmbutton(int input, int *state, int *counter, int *output){
    switch (*state){
        case IDLE:
            {
            if(input == 1){            // input rising edge
                *state = HELD;
                *output = 1;           // trigger action
            }
            break;
            }
            case HELD:
            {
                  *output = 0;
                  if(input == 0){            // falling edge
                        *state = DEBOUNCING;
                  }
                  break;
```

```
            }
        case DEBOUNCING:
            {
                  *counter += 1;
            if(*counter > 5){
                        *state = IDLE;
                        *counter = 0;
                  }
            break;
        }
        default:
            break;
    }
}

#endif
```

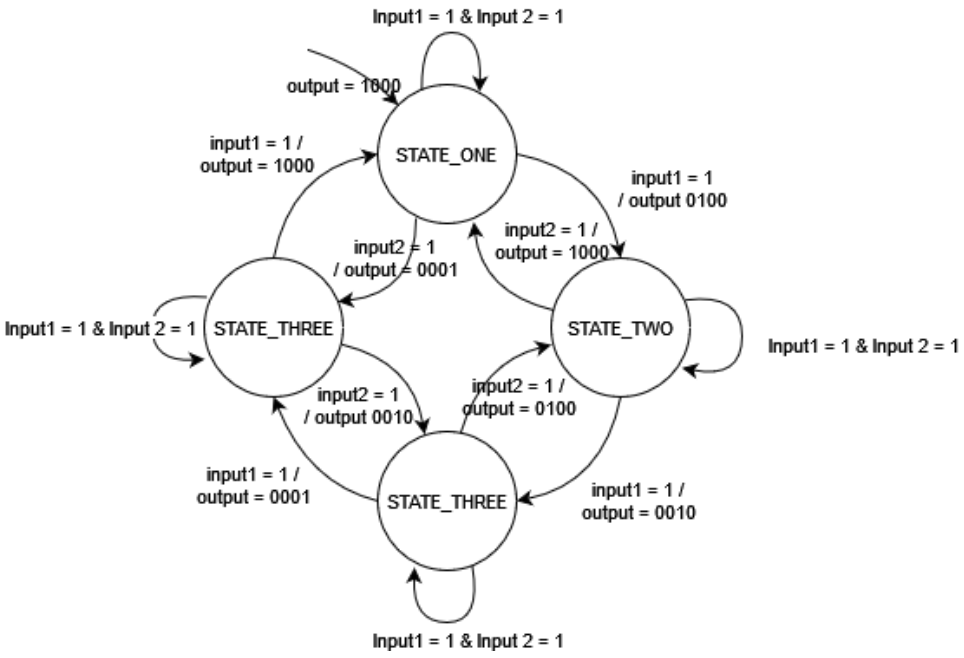
FSM LED Geser:

Untuk FSM LED, terdiri dari beragam state nyala LED, sebagai berikut:

- State 1, LED pertama menyala
- State 2, LED kedua menyala
- State 3, LED ketiga menyala
- State 4, LED keempat menyala

FSM ini menerima informasi mengenai keadaan kedua input, lalu memutuskan logika nyala LED berdasarkan informasi tersebut.

FSM yang didesain untuk memenuhi spesifikasi adalah:



Gambar 3 FSM LED Geser

Program untuk FSM LED geser, `fsmled.h` sebagai berikut:

```
#ifndef FSMLED_H
#define FSMLED_H

#include <stdio.h>
#include <stdlib.h>

#define STATE_ONE    0
#define STATE_TWO    1
#define STATE_THREE  2
#define STATE_FOUR   3


void fsmled(int input1, int input2, int *state, int *led){
    switch (*state){
        case STATE_ONE:
            {
            if(input1 == 1 && input2 == 1){            // both button
                        led[0] = 1;
                        led[1] = 0;
                        led[2] = 0;
                        led[3] = 0;
            }
                  else if(input1 == 1){
                        *state = STATE_TWO;
                        led[0] = 0;
                        led[1] = 1;
                        led[2] = 0;
                        led[3] = 0;
                  }
                  else if(input2 == 1){
                        *state = STATE_FOUR;
                        led[0] = 0;
                        led[1] = 0;
                        led[2] = 0;
                        led[3] = 1;
                  }
                  else{
                      led[0] = 1;
                        led[1] = 0;
                        led[2] = 0;
                        led[3] = 0;
                  }
            break;
            }
        case STATE_TWO:
            {
            if(input1 == 1 && input2 == 1){            // both button
                        led[0] = 0;
                        led[1] = 1;
                        led[2] = 0;
                        led[3] = 0;
            }
                  else if(input1 == 1){
                        *state = STATE_THREE;
                        led[0] = 0;
```

```
                        led[1] = 0;
                        led[2] = 1;
                        led[3] = 0;
                }
                else if(input2 == 1){
                        *state = STATE_ONE;
                        led[0] = 1;
                        led[1] = 0;
                        led[2] = 0;
                        led[3] = 0;
                }
                else{
                        led[0] = 0;
                        led[1] = 1;
                        led[2] = 0;
                        led[3] = 0;
                }
            break;
        }
        case STATE_THREE:
            {
            if(input1 == 1 && input2 == 1){            // both button
                        led[0] = 0;
                        led[1] = 0;
                        led[2] = 1;
                        led[3] = 0;
            }
                else if(input1 == 1){
                        *state = STATE_FOUR;
                        led[0] = 0;
                        led[1] = 0;
                        led[2] = 0;
                        led[3] = 1;
                }
                else if(input2 == 1){
                        *state = STATE_TWO;
                        led[0] = 0;
                        led[1] = 1;
                        led[2] = 0;
                        led[3] = 0;
                }
                else{
                        led[0] = 0;
                        led[1] = 0;
                        led[2] = 1;
                        led[3] = 0;
                }
            break;
            }
        case STATE_FOUR:
            {
            if(input1 == 1 && input2 == 1){            // both button
                        led[0] = 0;
                        led[1] = 0;
                        led[2] = 0;
                        led[3] = 1;
            }
```

```
                    else if(input1 == 1){
                        *state = STATE_ONE;
                        led[0] = 1;
                        led[1] = 0;
                        led[2] = 0;
                        led[3] = 0;
                    }
                    else if(input2 == 1){
                        *state = STATE_THREE;
                        led[0] = 0;
                        led[1] = 0;
                        led[2] = 1;
                        led[3] = 0;
                    }
                    else{
                        led[0] = 0;
                        led[1] = 0;
                        led[2] = 0;
                        led[3] = 1;
                    }
                break;
            }
            default:
                break;
        }
}
#endif
```

3 Simulasi

Untuk melihat dan memastikan spesifikasi terpenuhi dari program yang dibuat terlebih dahulu, dilakukan simulasi setiap sub-sistem satu per satu dan simulasi keseluruhan. Simulasi dilakukan dengan CODE::BLOCKS

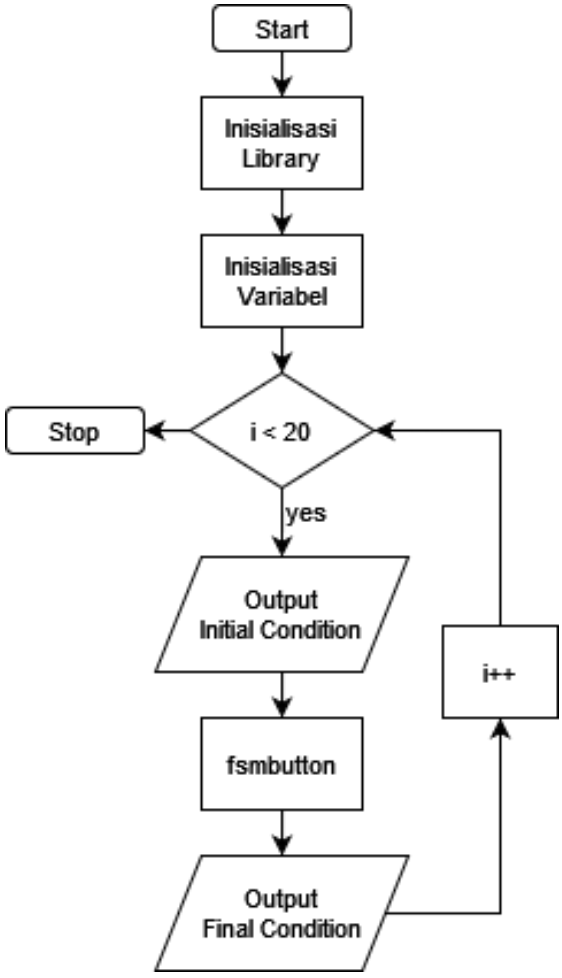
Simulasi Debouncing dan Edge Detector

Digunakan `fsmbutton.h` di atas.
Program simulasi button sebagai berikut:

```
#include <stdio.h>
#include <stdlib.h>
#include
"C:\Users\user\Desktop\Scanned\Semester_7\PSE\Tugas\Tugas_3_Cascade\Code
\fsmbutton.h"

int main(){
    int state = 0;
    int counter = 0;
    int output = 0;
```

```
    int debounce[20] = {0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1,
1, 1, 0, 0};

    for(int i = 0; i < 20; i++){
        printf("Initial State: %i, Input: %i\n", state, debounce[i]);
        fsmbutton(debounce[i], &state, &counter, &output);
        printf("Final State: %i, Output: %i\n", state, output);
        printf("\n");
    }
}
```

Flowchart dari program simulasi button:



Gambar 4 Flowchart Simulasi Button

Hasil simulasi FSM button:

```
Initial State: 0, Input: 0
Final State: 0, Output: 0

Initial State: 0, Input: 0
Final State: 0, Output: 0

Initial State: 0, Input: 1
Final State: 2, Output: 1

Initial State: 2, Input: 1
Final State: 2, Output: 0

Initial State: 2, Input: 0
Final State: 1, Output: 0

Initial State: 1, Input: 0
Final State: 1, Output: 0

Initial State: 1, Input: 1
Final State: 1, Output: 0

Initial State: 1, Input: 0
Final State: 1, Output: 0

Initial State: 1, Input: 0
Final State: 0, Output: 0

Initial State: 0, Input: 1
Final State: 2, Output: 1
```

```
Initial State: 2, Input: 0
Final State: 1, Output: 0

Initial State: 1, Input: 1
Final State: 1, Output: 0

Initial State: 1, Input: 0
Final State: 1, Output: 0

Initial State: 1, Input: 0
Final State: 1, Output: 0

Initial State: 1, Input: 1
Final State: 0, Output: 0

Initial State: 0, Input: 1
Final State: 2, Output: 1

Initial State: 2, Input: 1
Final State: 2, Output: 0

Initial State: 2, Input: 1
Final State: 2, Output: 0

Initial State: 2, Input: 0
Final State: 1, Output: 0

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

Gambar 5 Hasil Simulasi Button

Bisa dilihat dari state 0 atau IDLE, akan berubah saat menerima input 1 ke state 2 atau HELD, yang akan tetap di state 2 saat menerima input 1 atau button masih tertahan, dan pindah ke state 1 atau debouncing saat button di lepas. Di state 1, input button tidak diterima hingga counter mencapai nilai 5, lalu berpindah ke state 0 atau IDLE.

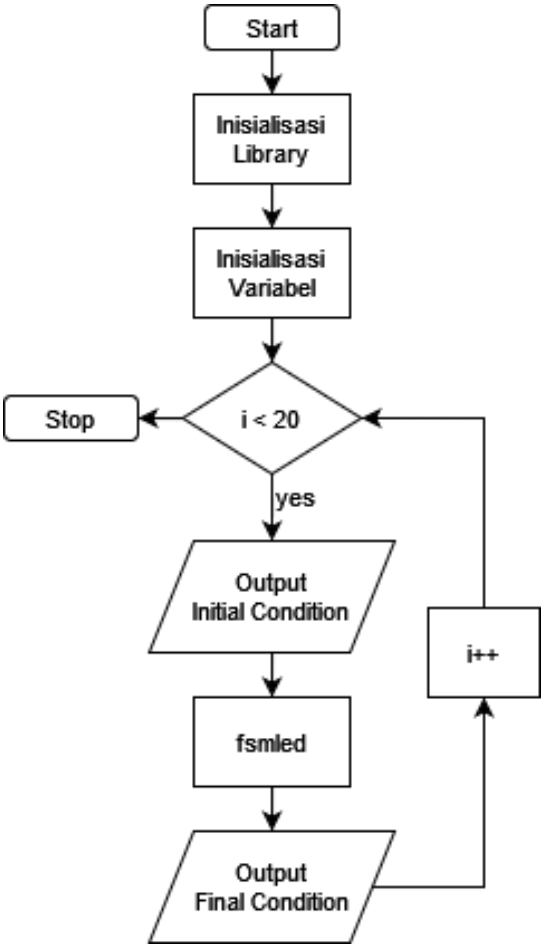
Simulasi LED Geser

Digunakan `fsmled.h`, program yang dibuat untuk simulasi kerja LED sebagai berikut:

```
#include <stdio.h>
#include <stdlib.h>
#include
"C:\Users\user\Desktop\Scanned\Semester_7\PSE\Tugas\Tugas_3_Cascade\Code
\fsmled.h"

int main(){
    int state = 0;
    int output[4] = {1, 0, 0, 0};

    int input1[20] = {0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1,
1, 0, 0};
    int input2[20] = {0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1,
1, 1, 1};
```

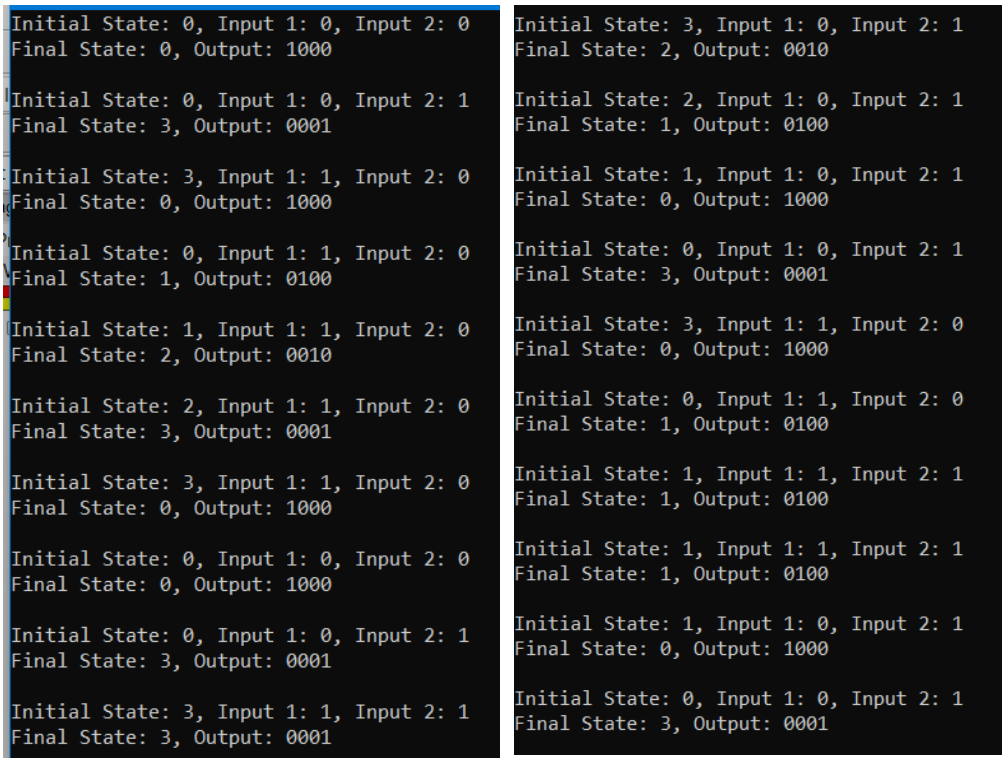
```
    for(int i = 0; i < 20; i++){
        printf("Initial State: %i, Input 1: %i, Input 2: %i\n", state,
input1[i], input2[i]);
        fsmled(input1[i], input2[i], &state, &output);
        printf("Final State: %i, Output: %i%i%i%i\n", state, output[0],
output[1], output[2], output[3]);
        printf("\n");
    }
}
```

Flowchart dari program simulasi LED sebagai berikut:



Gambar 6 Flowchart simulasi LED

Hasil dari simulasi ini sebagai berikut:



Gambar 7 Hasil Simulasi LED

Dapat dilihat dari hasil simulasi LED, dia akan berpindah ke state berikutnya saat input1 (button kiri) bernilai 1, dan berpindah ke state sebelumnya saat input2 (button kanan) bernilai 1. Apabila kedua button ditekan bersamaan, maka tidak terjadi perubahan state.

Simulasi Sistem Keseluruhan

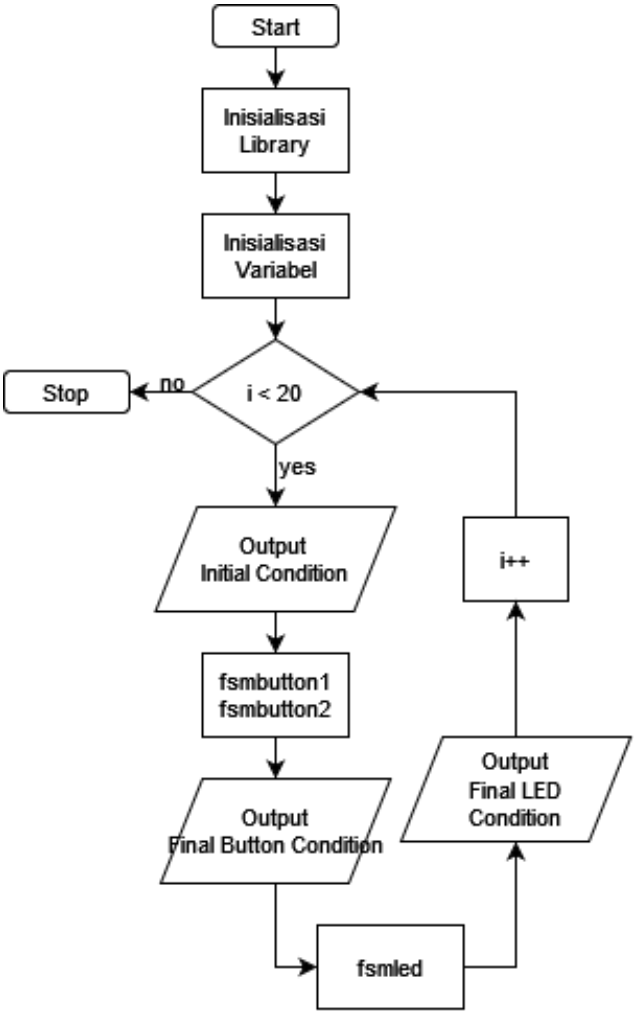
Untuk melihat kerja kedua sub-sistem bersamaan, maka disatukan keduanya menjadi satu simulasi utuh yang menjalankan proses dari awal hingga akhir dengan input kedua button.

Program yang dibuat untuk simulasi sistem:

```
#include <stdio.h>
#include <stdlib.h>
#include
"C:\Users\user\Desktop\Scanned\Semester_7\PSE\Tugas\Tugas_3_Cascade\Code
\fsmbutton.h"
#include
"C:\Users\user\Desktop\Scanned\Semester_7\PSE\Tugas\Tugas_3_Cascade\Code
\fsmled.h"

int main(){
    int input1[20] = {0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
1, 0, 0};
    int state1 = 0;
    int counter1 = 0;
```

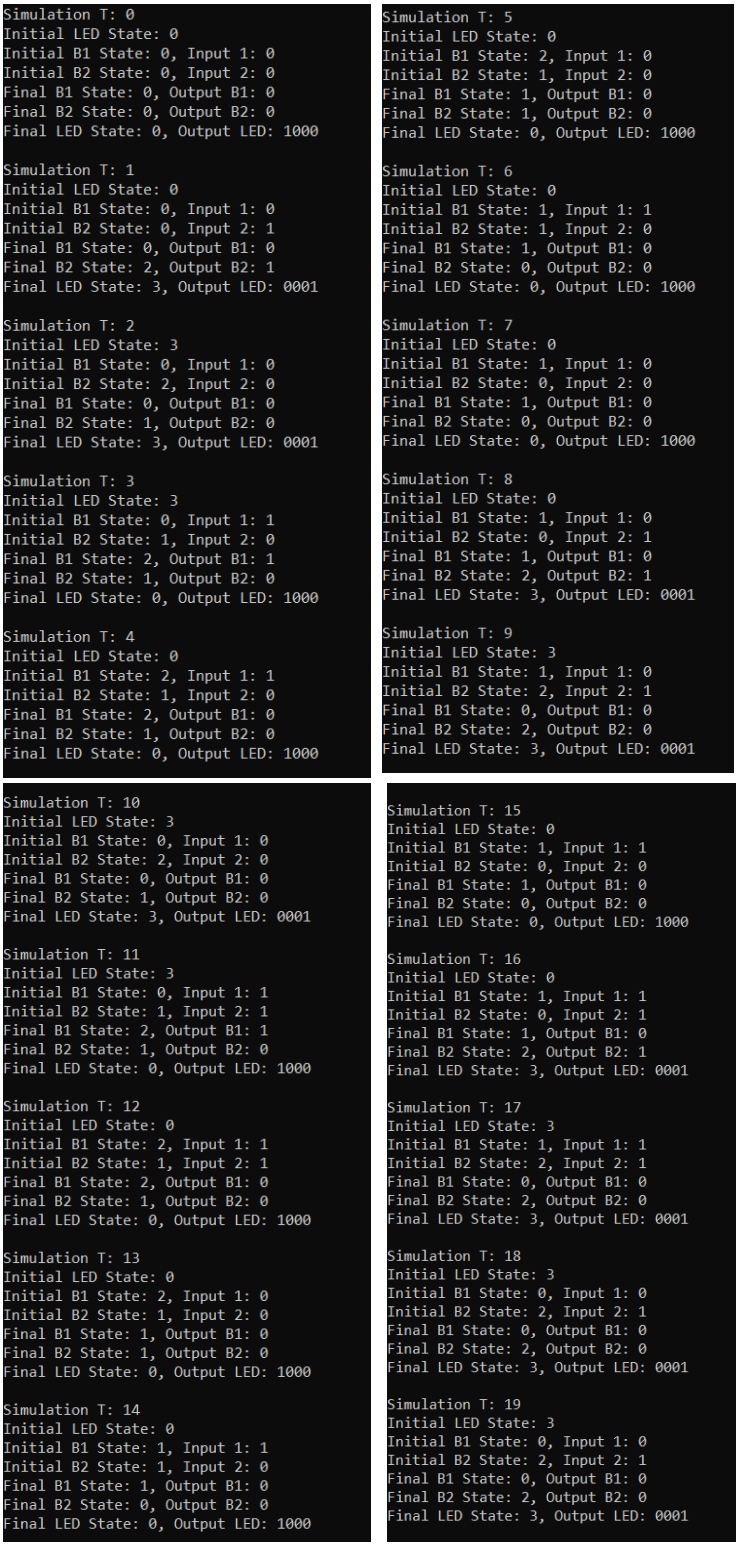
```
    int output1 = 0;

    int input2[20] = {0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1,
1, 1, 1};
    int state2 = 0;
    int counter2 = 0;
    int output2 = 0;

    int stateled = 0;
    int led[4] = {1, 0, 0, 0};

    for(int i = 0; i < 20; i++){
        printf("Simulation T: %i\n", i);
        printf("Initial LED State: %i\n", stateled);
        printf("Initial B1 State: %i, Input 1: %i\n", state1,
input1[i]);
        printf("Initial B2 State: %i, Input 2: %i\n", state2,
input2[i]);

        fsmbutton(input1[i], &state1, &counter1, &output1);
        fsmbutton(input2[i], &state2, &counter2, &output2);

        printf("Final B1 State: %i, Output B1: %i\n", state1, output1);
        printf("Final B2 State: %i, Output B2: %i\n", state2, output2);

        fsmled(output1, output2, &stateled, &led);
        printf("Final LED State: %i, Output LED: %i%i%i%i\n", stateled,
led[0], led[1], led[2], led[3]);
        printf("\n");
    }
}
```

Flowchart dari program simulasi sistem:



Gambar 8 Flowchart simulasi sistem

Hasil dari simulasi sistem:



Gambar 9 Hasil Simulasi Sistem

4 Implementasi

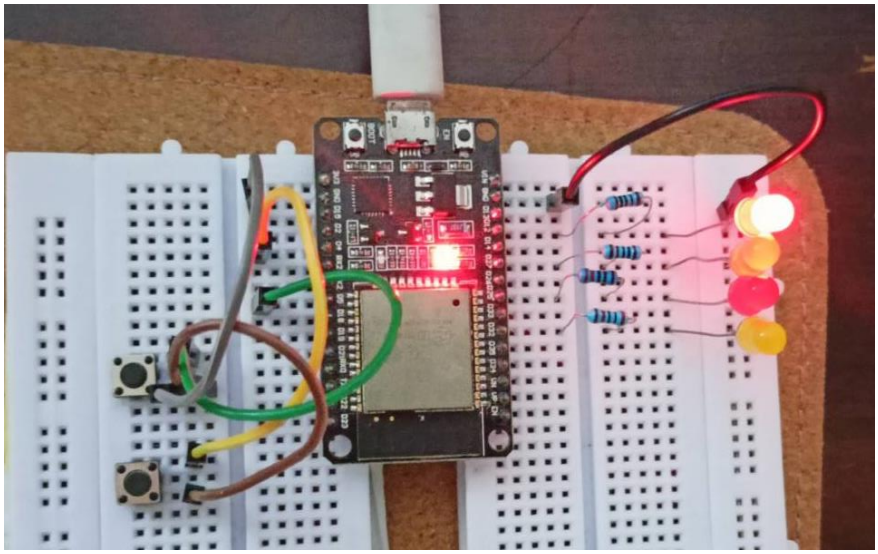
Setelah dilakukan simulasi, akan diimplementasikan sistem pada dunia nyata dengan mikrokontroler ESP32 dan compiler ESP-IDF.

Program yang dibuat untuk implementasi di ESP32 sebagai berikut:

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

#include "fsmbutton.h"
#include "fsmled.h"

#define GPIO_OUTPUT_1 12
#define GPIO_OUTPUT_2 27
#define GPIO_OUTPUT_3 25
#define GPIO_OUTPUT_4 32
#define GPIO_OUTPUT_PIN_SEL
((1ULL<<GPIO_OUTPUT_1)|(1ULL<<GPIO_OUTPUT_2)|(1ULL<<GPIO_OUTPUT_3)|(1ULL
<<GPIO_OUTPUT_4))

#define GPIO_INPUT_PB_1 5
#define GPIO_INPUT_PB_2 4
#define GPIO_INPUT_PIN_SEL
((1ULL<<GPIO_INPUT_PB_1)|(1ULL<<GPIO_INPUT_PB_2))

const TickType_t xDelay = 25 / portTICK_PERIOD_MS;

int input1 = 0;
int state1 = 0;
int counter1 = 0;
int output1 = 0;

int input2 = 0;
int state2 = 0;
int counter2 = 0;
int output2 = 0;

int stateled = 0;
int led[4];

void app_main(void)
{
      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);

    while (1)
      {
            input1 = !(gpio_get_level(GPIO_INPUT_PB_1));
            input2 = !(gpio_get_level(GPIO_INPUT_PB_2));

            fsmbutton(input1, &state1, &counter1, &output1);
            fsmbutton(input2, &state2, &counter2, &output2);

            fsmled(output2, output1, &stateled, &led);

            gpio_set_level(GPIO_OUTPUT_1, led[0]);
            gpio_set_level(GPIO_OUTPUT_2, led[1]);
            gpio_set_level(GPIO_OUTPUT_3, led[2]);
            gpio_set_level(GPIO_OUTPUT_4, led[3]);

        vTaskDelay(xDelay);
    }
}
```

Rangkaian ESP32 dari proses implementasi ini:



Gambar 10 Rangkaian ESP32

Implementasi telah berhasil dilakukan dan proses perpindahan LED geser berjalan dengan baik.

Link Video :

https://drive.google.com/file/d/1FuWqUEIUgH2Tex6OCBlceq9ObV7H-Fs0/view?usp=share_link

Link GitHub :

https://github.com/vinlred/LED_Chaser

5 Kesimpulan

- Kendali Cascade LED Chaser berhasil didesain, disimulasikan dan diimplementasikan.
- Kendali LED Chaser akan terdiri dari berbagai sub-sistem yang berjalan bersamaan untuk menerima input, memproses logika, dan mengubah nyala LED.
- FSM untuk button akan mendeteksi input saat rising edge untuk mengirimkan informasi ke FSM LED, menunggu hingga falling edge, lalu memasuki debouncing. Setelah debouncing akan kembali ke posisi IDLE untuk menunggu input dari user.
- FSM LED menerima input dari kedua button, dan berdasarkan informasi tersebut, menentukan perilaku nyala ke-4 LED. Apabila button kanan ditekan, LED bergeser ke kanan, apabila button kiri ditekan, LED bergeser ke kiri.