

TUGAS LAMPU GESER

Oleh:
Sidartha Prastya. P - 13219033



INSTITUT TEKNOLOGI BANDUNG
2022

DAFTAR ISI

1	Desain Sistem	2
2	Implementasi Hardware	3
3	Implementasi Software	4
4	Pengujian Simulasi Modul Debouncer dan Edge Detector	11
5	Pengujian Simulasi Modul Lampu Geser	13
6	Pengujian Simulasi Sistem Keseluruhan	14
7	Pengujian Sistem Keseluruhan pada <i>Hardware</i>	17
8	Kesimpulan	18
9	Lampiran	19

DAFTAR GAMBAR

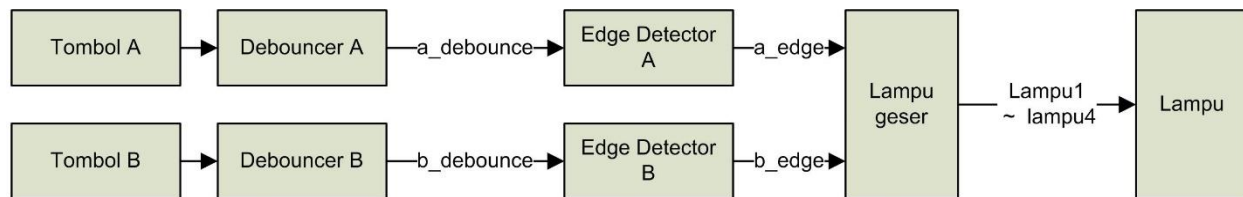
Gambar 1. Blok diagram spesifikasi sistem	2
Gambar 2. Blok diagram sistem yang digunakan	2
Gambar 3. Skematik rangkaian yang digunakan dalam implementasi	3
Gambar 4. Foto rangkaian yang dibuat	3
Gambar 5. State diagram Debouncer + Edge detector	10
Gambar 6. State diagram lampu geser	11
Gambar 7. Flowchart simulasi modul debouncer dan edge detector	12
Gambar 8. Hasil simulasi modul debounce dan edge detector	13
Gambar 9. Hasil simulasi modul lampu geser	14
Gambar 10. Flowchart simulasi sistem keseluruhan	16
Gambar 11. Hasil simulasi sistem keseluruhan	17
Gambar 12. Hasil simulasi sistem keseluruhan pada hardware	18

1 Desain Sistem

Pada tugas “Lampu Geser” terdapat beberapa spesifikasi yang harus dipenuhi, yaitu:

- Terdapat 4 buah LED yang berperan sebagai output.
- Terdapat 2 buah tombol input: kiri dan kanan.
- Dalam satu waktu, hanya ada sebuah LED yang menyala.
- Jika tombol kiri ditekan, maka LED yang menyala bergeser ke kiri.
- Jika tombol kanan ditekan, maka LED yang menyala bergeser ke kanan.
- Sistem dilengkapi dengan *debouncing*.
- Sistem dibuat secara sinkron (*time triggered*).
- Sistem dibuat dengan menggunakan *cascade composition*.

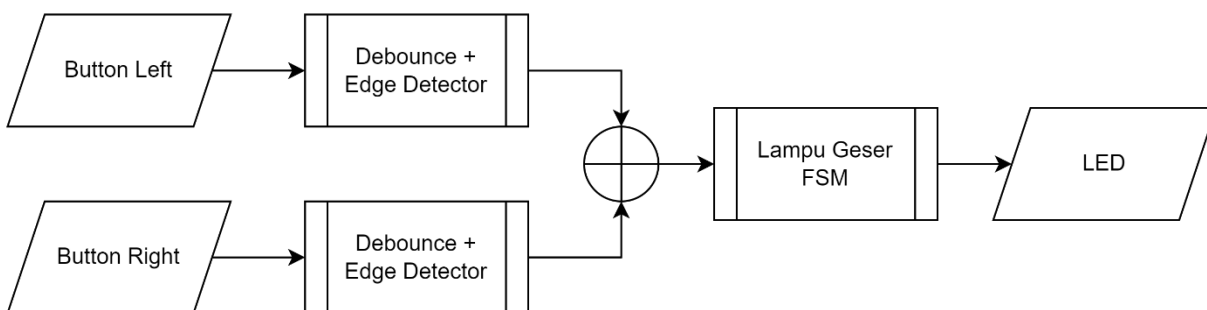
Secara umum, sistem dapat digambarkan dengan diagram blok berikut.



Gambar 1. Blok diagram spesifikasi sistem

Pada diagram di atas, terlihat bahwa masing-masing tombol memiliki *debouncing* secara terpisah. Setelah melalui *debouncing*, maka sinyal masukan akan dideteksi dengan menggunakan *edge detector* untuk mendeteksi adanya masukan pada tombol. Hasil tersebut kemudian dimasukkan ke fsm lampu geser yang kemudian akan menghasilkan keluaran berupa posisi lampu yang menyala.

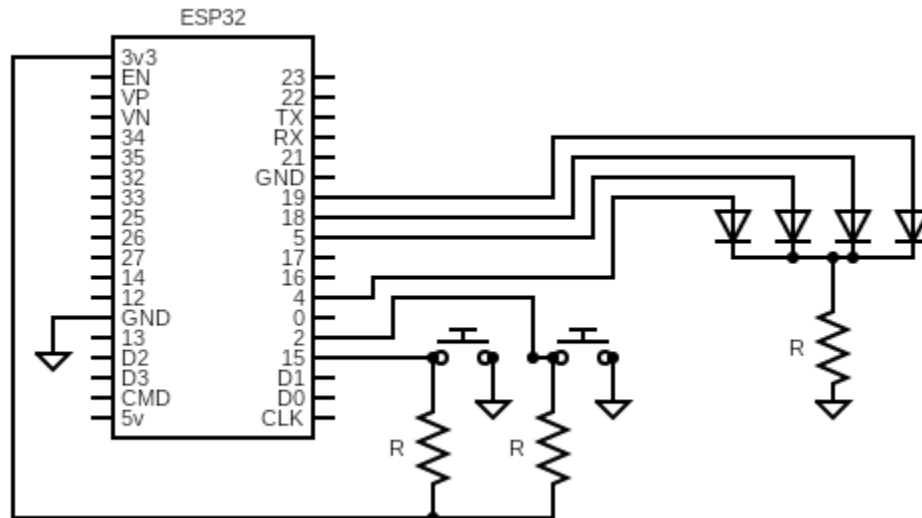
Pada tugas ini, untuk *debouncer* dan *edge detector* dibuat di dalam satu modul yang sama. Tombol kiri dideteksi sebagai -1 dan kanan sebagai 1. Kedua tombol kemudian dijumlahkan yang akan menghasilkan sebuah input untuk lampu geser. Apabila kedua tombol ditekan bersamaan, maka akan memberikan input 0. Berikut adalah gambar sistem yang digunakan.



Gambar 2. Blok diagram sistem yang digunakan

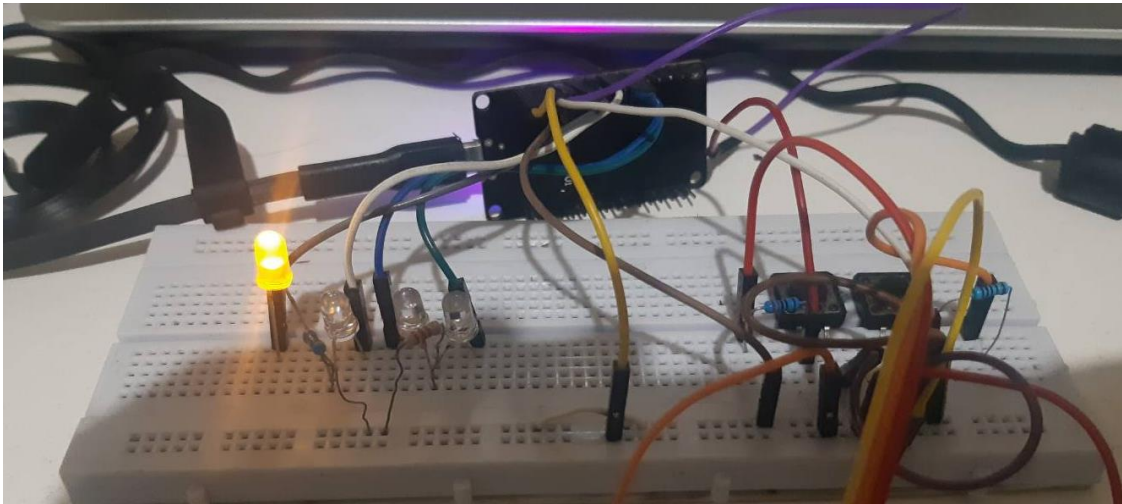
2 Implementasi Hardware

Pada tugas lampu geser, digunakan sebuah mikrokontroler, 2 buah input *push button*, dan 4 buah *output* LED. Berikut adalah ilustrasi skematik yang digunakan pada implementasi *hardware*.



Gambar 3. Skematik rangkaian yang digunakan dalam implementasi

Pada implementasi digunakan input berupa *push button pullup* yang akan bernilai 1 ketika tidak ditekan dan bernilai 0 ketika ditekan. Kemudian, output LED bersifat *common ground*. Masing-masing LED dihubungkan dengan pin GPIO pada ESP32. Lampu akan menyala ketika diberi nilai HIGH dan mati ketika LOW. Berikut adalah rangkaian yang dibuat.



Gambar 4. Foto rangkaian yang dibuat

3 Implementasi Software

Pada tugas ini hanya digunakan sebuah mikrokontroler sehingga hanya terdapat satu jenis *software*. Terdapat 2 bagian utama pada program, yaitu program utama dan FSM. Pada FSM terdapat 2 modul, yaitu modul *debouncer* dan *edge detector* serta modul lampu geser. Kemudian, keduanya dijalankan di dalam program utama. FSM dijalankan di dalam suatu interupsi *timer* sehingga dijalankan dalam frekuensi yang konstan. Input *button* diterima secara terus-menerus, tetapi nilainya baru digunakan setelah interupsi *timer* terjadi, sehingga sistem menjadi *time triggered*. Program yang dibuat dikompilasi dengan menggunakan ESP-IDF.

Berikut adalah *source code* yang digunakan pada “fsm.h”.

```
#ifndef FSM_H
#define FSM_H

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

#define STATE_1    0
#define STATE_2    1
#define STATE_3    2
#define STATE_4    3

#define STATE_DETECT    4
#define STATE_HOLD      5
#define STATE_SEND      6

void lampu_geser(int *button_debouncer, int *state_geser, int *led){
    switch (*state_geser){
        case STATE_1:
            if(*button_debouncer == 1){           // Tombol kanan
                *state_geser = STATE_2;
                led[0] = 0;
                led[1] = 1;
                led[2] = 0;
                led[3] = 0;
                *button_debouncer = 0;
            }
            else if(*button_debouncer == -1){ // tombol kiri
                *state_geser = STATE_4;
                led[0] = 0;
                led[1] = 0;
                led[2] = 0;
                led[3] = 1;
            }
        }
    }
```

```

        *button_debouncer = 0;
    }
    else{
        led[0] = 1;
        led[1] = 0;
        led[2] = 0;
        led[3] = 0;
    }
    break;

case STATE_2:
    if(*button_debouncer == 1){           // Tombol kanan
        *state_geser = STATE_3;
        led[0] = 0;
        led[1] = 0;
        led[2] = 1;
        led[3] = 0;
        *button_debouncer = 0;
    }
    else if(*button_debouncer == -1){ // tombol kiri
        *state_geser = STATE_1;
        led[0] = 1;
        led[1] = 0;
        led[2] = 0;
        led[3] = 0;
        *button_debouncer = 0;
    }
    break;

case STATE_3:
    if(*button_debouncer == 1){           // Tombol kanan
        *state_geser = STATE_4;
        led[0] = 0;
        led[1] = 0;
        led[2] = 0;
        led[3] = 1;
        *button_debouncer = 0;
    }
    else if(*button_debouncer == -1){ // tombol kiri
        *state_geser = STATE_2;
        led[0] = 0;
        led[1] = 1;
        led[2] = 0;
        led[3] = 0;
    }

```

```

        *button_debouncer = 0;
    }
    break;

case STATE_4:
    if(*button_debouncer == 1){           // Tombol kanan
        *state_geser = STATE_1;
        led[0] = 1;
        led[1] = 0;
        led[2] = 0;
        led[3] = 0;
        *button_debouncer = 0;
    }
    else if(*button_debouncer == -1){ // tombol kiri
        *state_geser = STATE_3;
        led[0] = 0;
        led[1] = 0;
        led[2] = 1;
        led[3] = 0;
        *button_debouncer = 0;
    }
    break;

default:
    *state_geser = STATE_1;
    led[0] = 1;
    led[1] = 0;
    led[2] = 0;
    led[3] = 0;
    *button_debouncer = 0;
    break;
}

}

void debouncer(int *button_in, int *counter, int *state_debounce, int *button_fsm){
    switch(*state_debounce){
        case STATE_DETECT:
            if(*button_in != 0){
                *state_debounce = STATE_HOLD;
                *button_fsm = *button_in;
                *button_in = 0;
                *counter = 0;
            }
    }
}

```

```

        break;

    case STATE_HOLD:
        if(*counter > 2){
            *state_debounce = STATE_DETECT;
        }
        else{
            *button_fsm = 0;
            *counter += 1;
        }
        break;

    default:
        *button_in = 0;
        *state_debounce = STATE_DETECT;
        break;
}
}

#endif

```

Kemudian, FSM di atas dijalankan pada program utama dengan *source code* berikut.

```

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

#include "fsm.h"

#define GPIO_OUTPUT_A    GPIO_NUM_4
#define GPIO_OUTPUT_B    GPIO_NUM_5
#define GPIO_OUTPUT_C    GPIO_NUM_18
#define GPIO_OUTPUT_D    GPIO_NUM_19
#define GPIO_OUTPUT_PIN_SEL ((1ULL<<GPIO_OUTPUT_A) | (1ULL<<GPIO_OUTPUT_B) |
(1ULL<<GPIO_OUTPUT_C) | (1ULL<<GPIO_OUTPUT_D))

#define GPIO_INPUT_PB1    GPIO_NUM_15
#define GPIO_INPUT_PB2    GPIO_NUM_2
#define GPIO_INPUT_PIN_SEL ((1ULL<<GPIO_INPUT_PB1) | (1ULL<<GPIO_INPUT_PB2))

#define TIMER_USED        0    // HardwareTimer 0
#define TIMER_DIVIDER      16
#define TIMER_SCALE        (TIMER_BASE_CLK /TIMER_DIVIDER)

```



```

#define DELAY_S                (0.1)

int led_array[8] = {4,5,18,19};
int state_geser = STATE_1; // Initial STATE
int state_debounce_l = STATE_DETECT, state_debounce_r = STATE_DETECT;
int led_out[4];
int button_left = 0, button_right = 0, button_geser = 0;
int counter_left = 0, counter_right = 0;
int button_debounce_l = 0, button_debounce_r = 0;

void IRAM_ATTR timer_group_isr(void* para) {

    timer_spinlock_take(TIMER_USED); // TIMER0
    int timer_idx = (int)para;
    uint32_t timer_intr = timer_group_get_intr_status_in_isr(TIMER_USED);

    if (timer_intr & TIMER_INTR_T0) {
        timer_group_clr_intr_status_in_isr(TIMER_USED, TIMER_0);
    } else if (timer_intr & TIMER_INTR_T1) {
        timer_group_clr_intr_status_in_isr(TIMER_USED, TIMER_1);
    }
    // -----
    // -----Main procedures for ISR-----
    // -----CASCADE FSM-----
    // Debouncer & Edge Detector
    debouncer(&button_left, &counter_left, &state_debounce_l, &button_debounce_l);
    debouncer(&button_right, &counter_right, &state_debounce_r,
&button_debounce_r);

    // DECISION MAKING
    button_geser = button_debounce_l + button_debounce_r;

    // FSM Processed
    lampu_geser(&button_geser, &state_geser, led_out);

    // Turn on desired LED
    for(int i=0; i < 4; i++){
        gpio_set_level(led_array[i], led_out[i]);
    }

    // -----
    // -----

    timer_group_enable_alarm_in_isr(TIMER_USED, timer_idx);
    timer_spinlock_give(TIMER_USED);
}

```

```

}

void app_main(void) {
    // -----
    // -----GPIO CONFIGURATION-----
    // OUTPUT LED
    gpio_config_t io_conf;
    io_conf.intr_type = 0;
    io_conf.mode = GPIO_MODE_DEF_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);
    // -----
    // INPUT BUTTON
    io_conf.pin_bit_mask = GPIO_INPUT_PIN_SEL;
    io_conf.mode = GPIO_MODE_DEF_INPUT;
    io_conf.intr_type = 0; // Falling Edge
    io_conf.pull_up_en = GPIO_PULLUP_ENABLE; // enable interrupt
    io_conf.pull_down_en = 0;
    gpio_config(&io_conf);
    // -----
    // -----TIMER CONFIGURATION-----
    timer_config_t config = {
        .divider = TIMER_DIVIDER,
        .counter_dir = TIMER_COUNT_UP,
        .counter_en = TIMER_START,
        .alarm_en = TIMER_ALARM_EN,
        .auto_reload = TIMER_AUTORELOAD_EN,
    };
    timer_init(TIMER_USED, TIMER_USED, &config);
    timer_set_counter_value(TIMER_USED, TIMER_USED, 0x00000000ULL);
    timer_set_alarm_value(TIMER_USED, TIMER_USED, DELAY_S * TIMER_SCALE);
    timer_enable_intr(TIMER_USED, TIMER_USED);

    // -----TIMER ISR-----
    timer_isr_register(TIMER_USED, TIMER_USED, timer_group_isr, (void*)TIMER_USED,
ESP_INTR_FLAG_IRAM, NULL);
    timer_start(TIMER_USED, TIMER_USED);
    // -----
    // -----

    while (1) {
        // Check Button (0 if pressed)

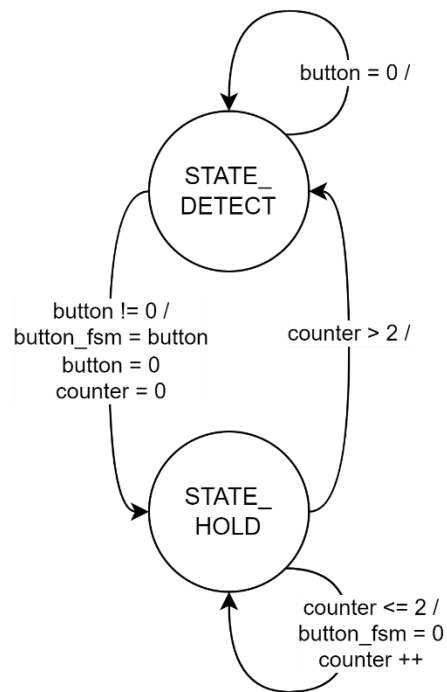
```

```

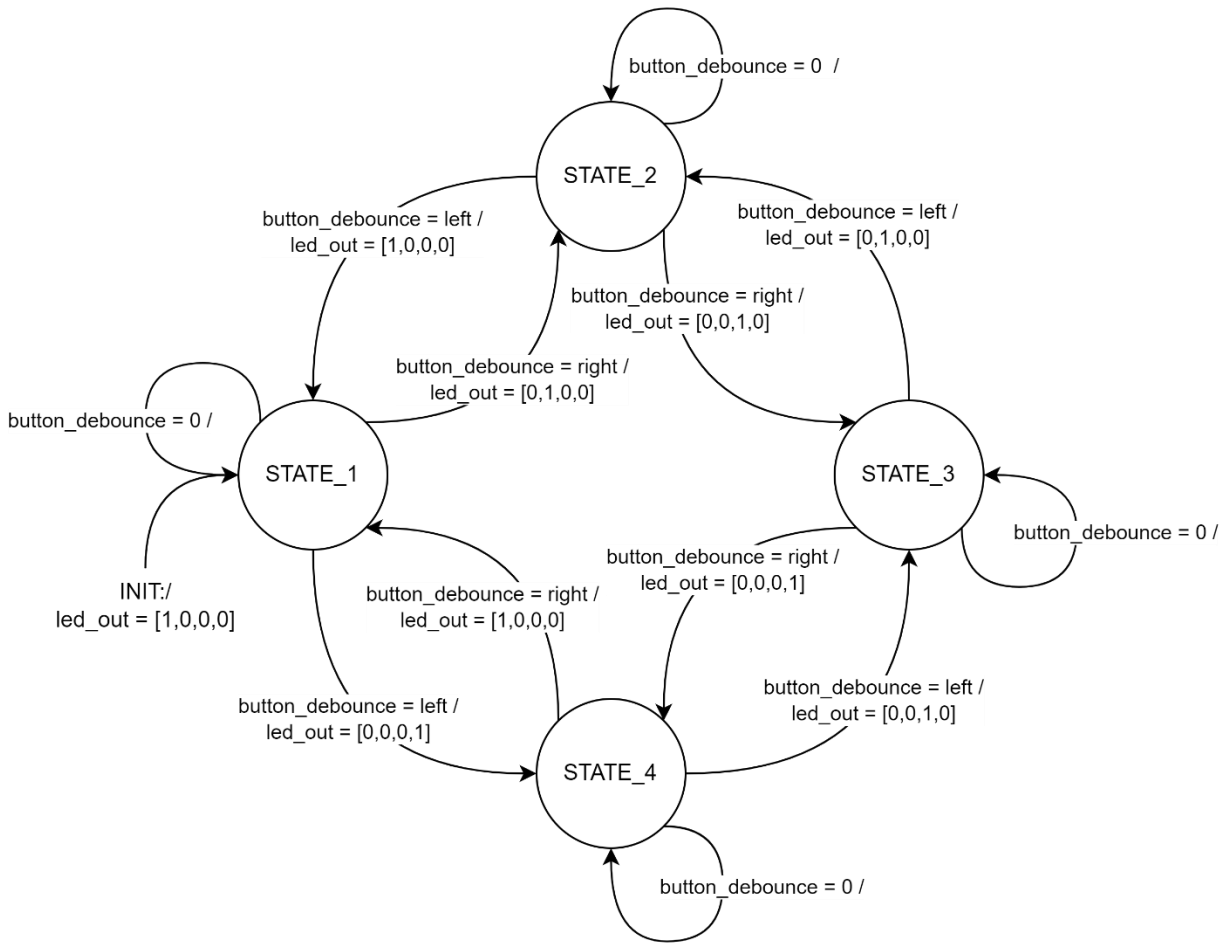
// Button value will back to 0 after FSM finished
if (gpio_get_level(GPIO_INPUT_PB1) == 0) {
    button_left = 1;
}
if(gpio_get_level(GPIO_INPUT_PB2) == 0){
    button_right = -1;
}
vTaskDelay(1);
};
}

```

Apabila digambarkan dalam suatu *state diagram*, maka didapatkan seperti ilustrasi berikut.



Gambar 5. State diagram Debouncer + Edge detector



Gambar 6. State diagram lampu geser

4 Pengujian Simulasi Modul Debouncer dan Edge Detector

Pada tugas ini dilakukan pengujian satu per satu terhadap setiap modul yang dibuat. Pada modul FSM *debouncer* terdapat *debounce* dan *edge detector* yang dijadikan satu modul. *Debouncer* berfungsi untuk mencegah adanya *debouncing* dari pembacaan *push button*. Kemudian, *edge detector* berfungsi untuk membaca nilai input sebagai 1, -1, atau 0. Pengujian dilakukan dengan memberikan *test case* berupa 10 nilai input. Berikut adalah *source code* yang digunakan dalam pengujian simulasi modul *debouncer* dan *edge detector* (*simul_debounce.c*).

```

#include <stdio.h>
#include "../fsm.h"

int main(){
    // Test case:
    int button[10] = {1, 0, 1, 1, 1, 1, 0, 1, 0, 0};
    // -----
  
```

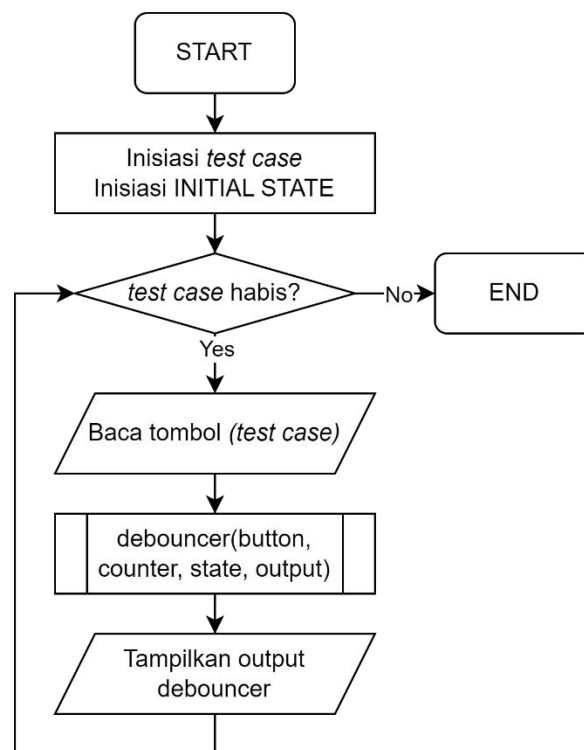
```

int out;
int counter = 0;
int state = STATE_DETECT;
printf("Test Case: ");
for (int k=0; k<11; k++){
    printf("%d, ", button[k]);
}
printf("\n");
for (int i=0; i<10; i++){
    int prev = button[i];
    debouncer(&button[i], &counter, &state, &out);
    printf("INPUT %d : %d\t | OUT : %d\n", (i+1), prev, out);
}

return 0;
}

```

Apabila *source code* di atas diilustrasikan menggunakan suatu *flowchart*, maka akan menjadi seperti berikut.



Gambar 7. Flowchart simulasi modul debouncer dan edge detector

Simulasi di atas menghasilkan keluaran sebagai berikut.

```
Test Case: 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0,
INPUT 1 : 1 | OUT : 1
INPUT 2 : 0 | OUT : 0
INPUT 3 : 1 | OUT : 0
INPUT 4 : 1 | OUT : 0
INPUT 5 : 1 | OUT : 0
INPUT 6 : 1 | OUT : 1
INPUT 7 : 0 | OUT : 0
INPUT 8 : 1 | OUT : 0
INPUT 9 : 0 | OUT : 0
INPUT 10 : 0 | OUT : 0
```

Gambar 8. Hasil simulasi modul debounce dan edge detector

Pada hasil simulasi di atas, didapatkan bahwa program hanya akan memvalidasi nilai *button* setiap setelah 4 siklus setelah pembacaan sebelumnya. Walaupun pada program FSM dituliskan batas *count* hingga 2, akan tetapi, input baru dikeluarkan ketika *count* mencapai 3. Ditambah dengan siklus ketika memasuki state HOLD, maka benar bahwa jeda siklus yang dibutuhkan untuk kemudian membaca sebuah input berikutnya adalah 4 siklus, sehingga periode pembacaan adalah per 5 siklus.

5 Pengujian Simulasi Modul Lampu Geser

Pengujian juga dilakukan pada modul lampu geser. Sama seperti simulasi sebelumnya, digunakan 11 buah *test case* dengan input ke-1 merupakan *initial state* yang ada. Alur algoritma yang digunakan sama persis dengan yang digunakan pada simulasi modul *debouncer* dan *edge detector*. Berikut adalah *source code* yang digunakan pada simulasi ini (*simul_lamp.c*).

```
#include <stdio.h>
#include "../fsm.h"

int main(){
    // Test case:
    int lamp_in[11] = {0, 1, 0, 1, 1, 1, -1, 0, -1, 0, -1};
    // -----

    int led_out[4];
    int state = STATE_1;
    int prev;

    for (int i=0; i<11; i++){
        prev = lamp_in[i];
        lampu_geser(&lamp_in[i], &state, led_out);
        printf("INPUT %d : %d\t | OUT : %d %d %d %d\n", i, prev,
               led_out[0], led_out[1], led_out[2], led_out[3]);
    }
}
```

```

    return 0;
}

```

Program di atas menghasilkan tampilan sebagai berikut:

INPUT 0 : 0		OUT : 1 0 0 0
INPUT 1 : 1		OUT : 0 1 0 0
INPUT 2 : 0		OUT : 0 1 0 0
INPUT 3 : 1		OUT : 0 0 1 0
INPUT 4 : 1		OUT : 0 0 0 1
INPUT 5 : 1		OUT : 1 0 0 0
INPUT 6 : -1		OUT : 0 0 0 1
INPUT 7 : 0		OUT : 0 0 0 1
INPUT 8 : -1		OUT : 0 0 1 0
INPUT 9 : 0		OUT : 0 0 1 0
INPUT 10 : -1		OUT : 0 1 0 0

Gambar 9. Hasil simulasi modul lampu geser

Pada simulasi ini tidak digunakan *debouncer* sehingga tidak berlaku *delay* input dan sistem dianggap ideal (memproses input apapun yang diterima). Dapat dilihat bahwa input 1 (kanan) akan menggeser LED yang menyala ke arah kanan dan input -1 (kiri) akan menggeser LED yang menyala ke arah kiri. Ketika LED berada di ujung suatu sisi dan bergerak ke arah yang sama, maka LED akan pindah ke ujung yang berlawanan dan melanjutkan ke arah yang seharusnya, seperti yang terlihat pada input ke-5 dan ke-6.

6 Pengujian Simulasi Sistem Keseluruhan

Pada simulasi sistem secara keseluruhan, maka input/*test case* mewakili input tombol yang ada, kemudian output dari *debouncer* dan *edge detector* diwakili oleh “*debounce_out*”. Pada simulasi ini digunakan 2 buah tombol/*set test case* yang mewakili tombol kanan (*right/r* dan *left/l*), sehingga modul *debouncer* dipanggil 2 kali yang masing-masing dipasang pada setiap *input*. Keluaran dari masing-masing *debouncer* kemudian dijumlahkan dan dijadikan *input* untuk FSM lampu geser. Berikut adalah *source code* yang digunakan pada simulasi ini (*simul_system.c*).

```

#include <stdio.h>
#include "../fsm.h"

int main(){
    // Test case:
    int button_in_left[11] = {0, -1, 0, -1, 0, -1, 0, 0, 0, 0, 0};
    int button_in_right[11] = {0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1};
    // -----
    int debounce_out_left = 0, debounce_out_right = 0;
    int counter_left = 0, counter_right = 0;
    int lamp_in;

```

```

int led_out[4];
int state_lamp = STATE_1;
int state_deb_left = STATE_DETECT, state_deb_right = STATE_DETECT;
int prev;
int i;

printf("BUTTON LEFT\t: ");
for (i=0; i<11; i++){
    printf("%d\t", button_in_left[i]);
}
printf("\n");
printf("BUTTON RIGHT\t: ");
for (i=0; i<11; i++){
    printf("%d\t", button_in_right[i]);
}
printf("\n");

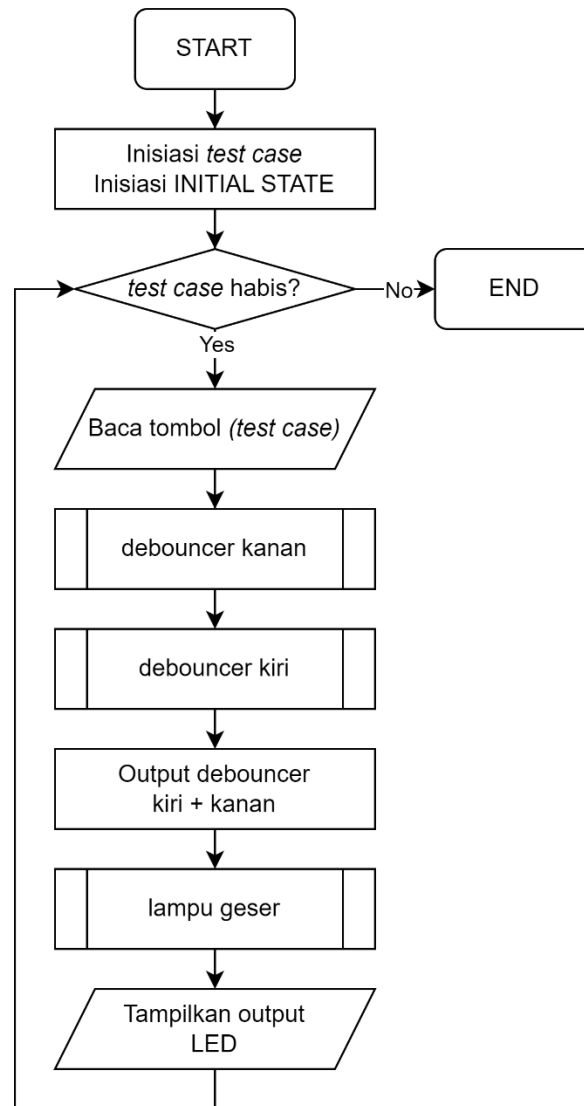
for (i=0; i<11; i++){
    // prev_left = button_in_left[i];
    // prev_right = button_in_right[i];
    debouncer(&button_in_left[i], &counter_left, &state_deb_left,
&debounce_out_left);
    debouncer(&button_in_right[i], &counter_right, &state_deb_right,
&debounce_out_right);

    lamp_in = debounce_out_left + debounce_out_right;
    prev = lamp_in;
    lampu_geser(&lamp_in, &state_lamp, led_out);
    printf("INPUT %d : %d\t | OUT : %d %d %d %d\n", i, prev,
        led_out[0], led_out[1], led_out[2], led_out[3]);
}

return 0;
}

```

Apabila digambarkan ke dalam diagram alur, maka akan dihasilkan diagram seperti berikut.



Gambar 10. Flowchart simulasi sistem keseluruhan

Simulai di atas menghasilkan tampilan sebagai berikut.

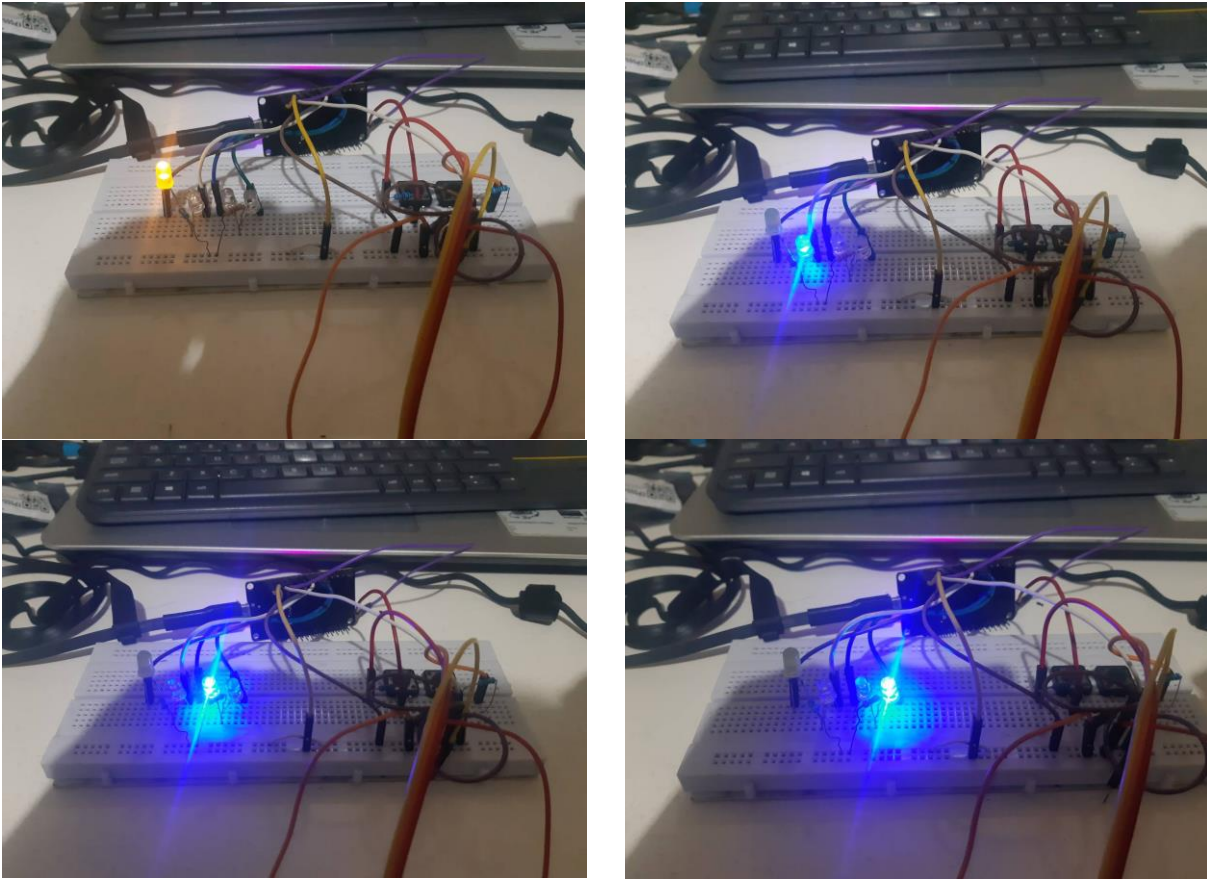
BUTTON LEFT	:	0	-1	0	-1	0	-1	0	0	0	0	0
BUTTON RIGHT	:	0	0	0	0	1	0	1	0	1	0	1
INPUT 0 :	0		OUT :	1	0	0	0					
INPUT 1 :	-1		OUT :	0	0	0	1					
INPUT 2 :	0		OUT :	0	0	0	1					
INPUT 3 :	0		OUT :	0	0	0	1					
INPUT 4 :	1		OUT :	1	0	0	0					
INPUT 5 :	0		OUT :	1	0	0	0					
INPUT 6 :	0		OUT :	1	0	0	0					
INPUT 7 :	0		OUT :	1	0	0	0					
INPUT 8 :	0		OUT :	1	0	0	0					
INPUT 9 :	0		OUT :	1	0	0	0					
INPUT 10 :	1		OUT :	0	1	0	0					

Gambar 11. Hasil simulasi sistem keseluruhan

Pada “button left” dan “button right” menunjukkan input/test case untuk masing-masing tombol. Pada simulasi ini, modul *debouncer* dan *edge detector* dijalankan secara bersamaan dan paralel sehingga keluarannya tidak tertunda apabila salah satu tombol telah dideteksi. Output dari kedua debouncer kemudian dijumlahkan, sehingga apabila hanya salah satu tombol yang terbaca, maka input akan ada, sedangkan input dianggap tidak ada apabila kedua tombol ditekan secara bersamaan. Pada baris INPUT merupakan input yang diterima oleh FSM lampu geser. Dari simulasi yang dilakukan, maka terlihat bahwa sistem keseluruhan telah berjalan dengan sesuai.

7 Pengujian Sistem Keseluruhan pada *Hardware*

Program yang ada pada subbab “Implementasi Software” sebelumnya diprogram ke dalam ESP32 dengan rangkaian yang dibuat seperti pada subbab “Implementasi Hardware” menghasilkan keluaran seperti pada beberapa citra/foto berikut.



Gambar 12. Hasil simulasi sistem keseluruhan pada hardware

Pada simulasi yang dilakukan, didapatkan hasil yang sudah sesuai dengan yang diinginkan. Akan tetapi, terkadang terdapat *debouncing* yang terjadi ketika menekan tombol. Hal tersebut berarti bahwa perlu ditambahkan *count* pada modul *debouncer*. Pada simulasi ini digunakan interupsi *timer* per 0.1 detik. Dengan siklus *debouncer* yang sekarang digunakan sebesar 5 siklus, maka *debounce* berada di sekitar 400ms. Besarnya waktu *debouncing* ini dapat disebabkan oleh tombol yang kurang responsif. Selain itu, program memang dibuat untuk selalu membaca input ketika ditekan. Ada kemungkinan bahwa penulis kurang cepat dalam mengangkat tombol sehingga masih terjadi *debouncing*.

8 Kesimpulan

Dari simulasi yang dilakukan, didapatkan kesimpulan sebagai berikut:

- Sistem lampu geser telah berhasil dibuat dengan menggunakan 2 buah input *push button* dan 4 buah output LED. Sistem dibuat dengan memanfaatkan 2 buah FSM, yaitu modul *debouncer* dan *edge detector* serta modul lampu geser.
- Modul *debouncer* dan *edge detector* berfungsi untuk menghindari/meminimalisir *debouncing* pada tombol serta menentukan jenis input yang diterima (kiri atau kanan).
- Modul lampu geser berfungsi untuk menentukan keluaran LED selanjutnya.
- Setiap modul telah diuji menggunakan simulasi pada *desktop* dan dihasilkan keluaran yang sesuai.

- Masih terdapat debounce (walaupun jarang terjadi) yang menunjukkan kurangnya *count* yang digunakan pada modul *debouncer*. Hal tersebut juga bisa diatasi apabila input hanya dibaca sekali hingga tombol dilepas dan ditekan kembali.

9 Lampiran

Seluruh *source code* pada simulasi ini dapat diakses melalui tautan *Github* berikut:

https://github.com/sidarthaprastya/lampu_geser