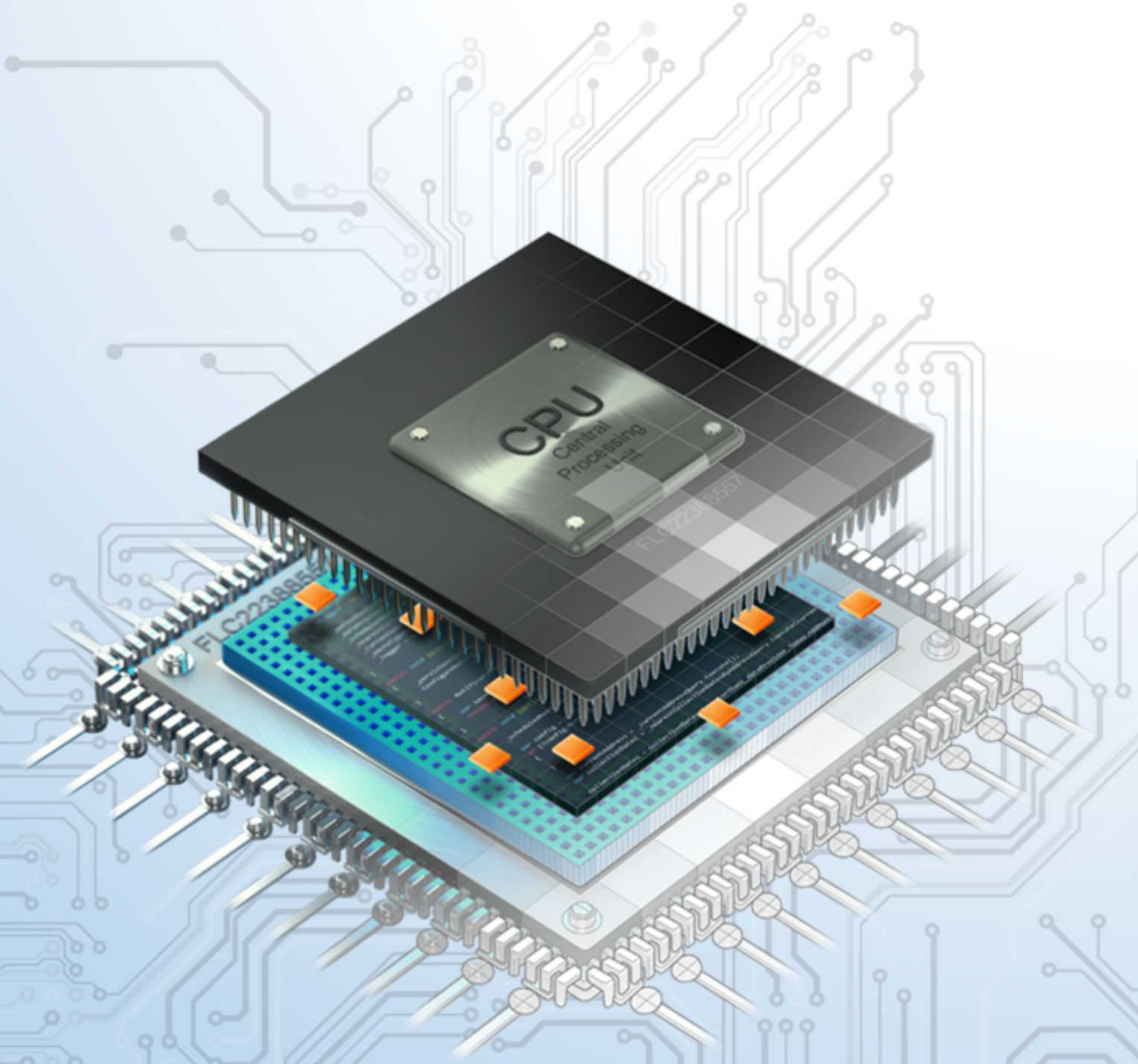
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller



Dr. Le Trong Nhan

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

VI XỬ LÝ-ĐIỀU KHIỂN (CO3009)

Report

LAB 3 : BUTTONS/SWITCHES

GVHD: Lê Trọng Nhân
Huỳnh Phúc Nghị

SV thực hiện: Nguyễn Quốc Việt – 2115279

Tp. Hồ Chí Minh, Tháng 11/2023

Mục lục

1 Exercise and Report 4

1.1 Specifications 4

1.2 Exercise 1: Sketch an FSM 5

1.3 Exercise 2: Proteus Schematic 6

1.4 Exercise 3: Create STM32 Project 6

1.5 Exercise 4: Modify Timer Parameters 7

1.6 Exercise 5: Adding code for button debouncing 8

1.7 Exercise 6: Adding code for displaying modes 11

1.8 Exercise 7: Adding code for increasing time duration value for the red LEDs 15

1.9 Exercise 8: Adding code for increasing time duration value for the amber LEDs 18

1.10 Exercise 9: Adding code for increasing time duration value for the green LEDs 19

1.11 Exercise 10: To finish the project 20

1.12 Link Github 20

1 Exercise and Report

1.1 Specifications

You are required to build an application of a traffic light in a cross road which includes some features as described below:

- The application has 12 LEDs including 4 red LEDs, 4 amber LEDs, 4 green LEDs.
- The application has 4 seven segment LEDs to display time with 2 for each road. The 2 seven segment LEDs will show time for each color LED corresponding to each road.
- The application has three buttons which are used

  - to select modes,

  - to modify the time for each color led on the fly, and

  - to set the chosen value.

- The application has at least 4 modes which is controlled by the first button. Mode 1 is a normal mode, while modes 2 3 4 are modification modes. You can press the first button to change the mode. Modes will change from 1 to 4 and back to 1 again.

  Mode 1 - Normal mode:

  - The traffic light application is running normally.

  Mode 2 - Modify time duration for the red LEDs: This mode allows you to change the time duration of the red LED in the main road. The expected behaviours of this mode include:

  - All single red LEDs are blinking in 2 Hz.

  - Use two seven-segment LEDs to display the value.

  - Use the other two seven-segment LEDs to display the mode.

  - The second button is used to increase the time duration value for the red LEDs.

  - The value of time duration is in a range of 1 - 99.

  - The third button is used to set the value.

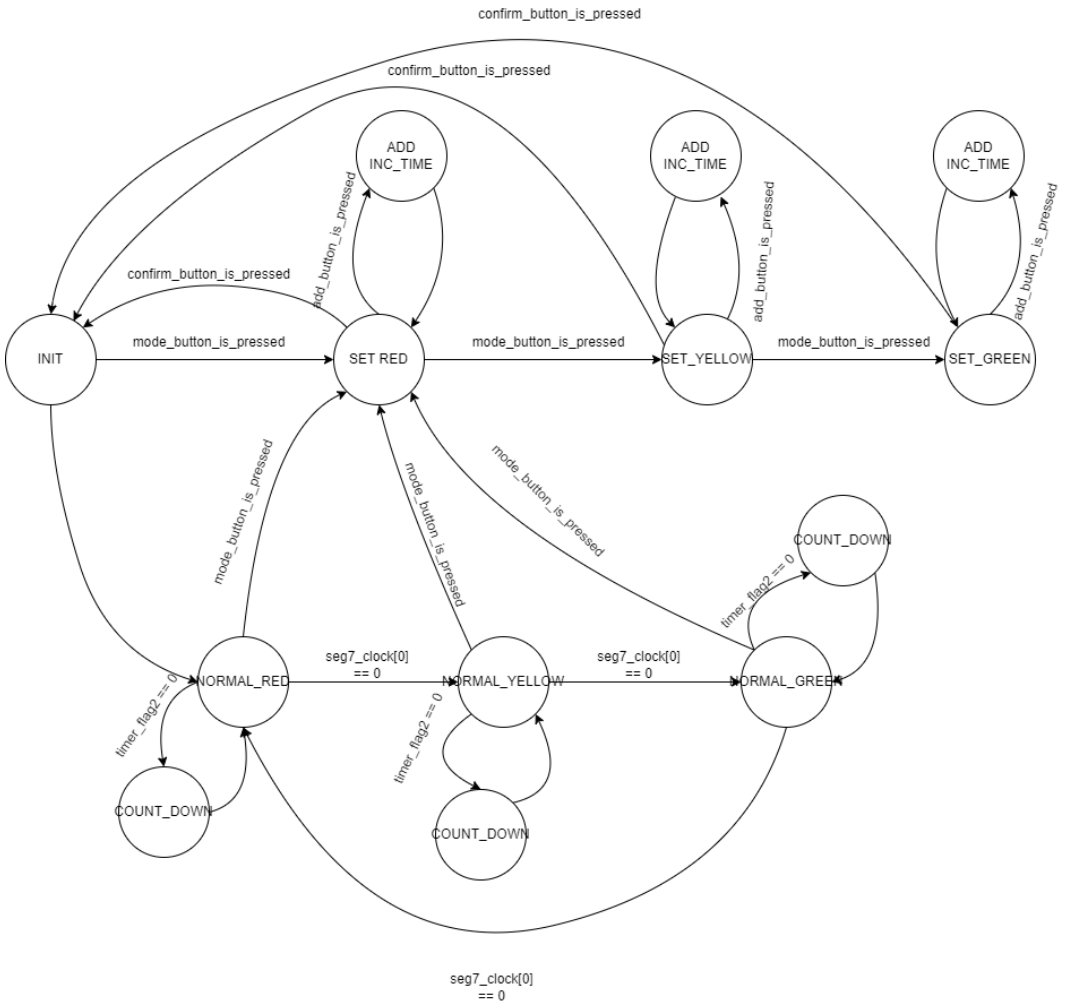
  Mode 3 - Modify time duration for the amber LEDs: Similar for the red LEDs described above with the amber LEDs.

  Mode 4 - Modify time duration for the green LEDs: Similar for the red LEDs described above with the green LEDs.

1.2 Exercise 1: Sketch an FSM

Your task in this exercise is to sketch an FSM that describes your idea of how to solve the problem.

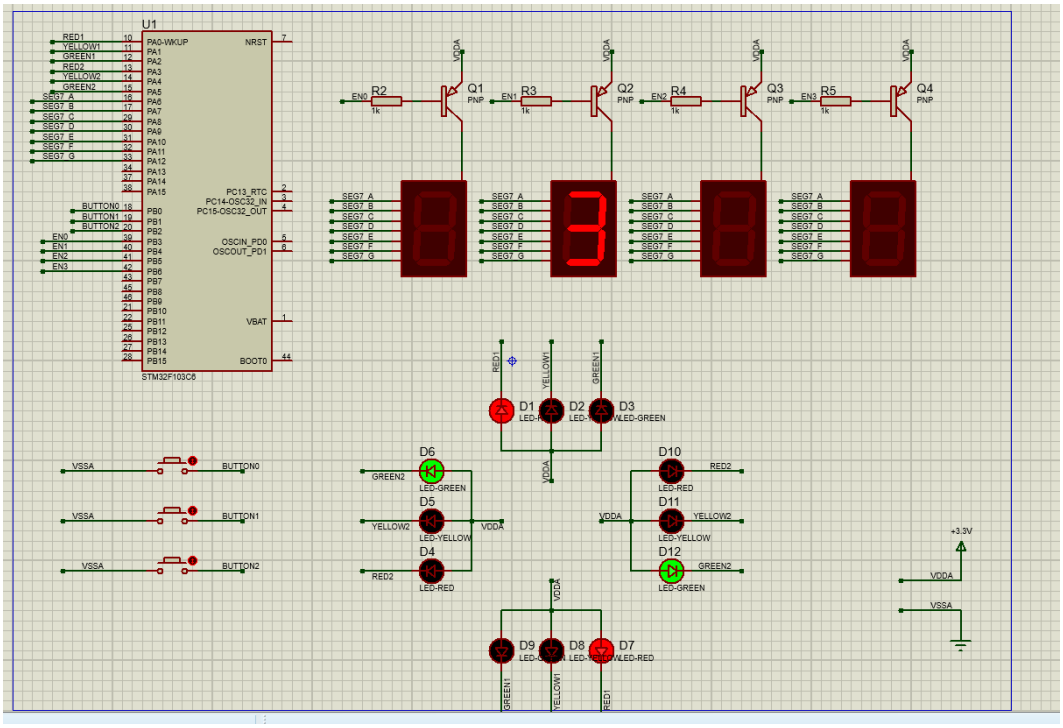
Please add your report here.
Report: FSM



Hình 1: *Sketch FSM*

1.3 Exercise 2: Proteus Schematic

Your task in this exercise is to draw a Proteus schematic for the problem above.
Report:

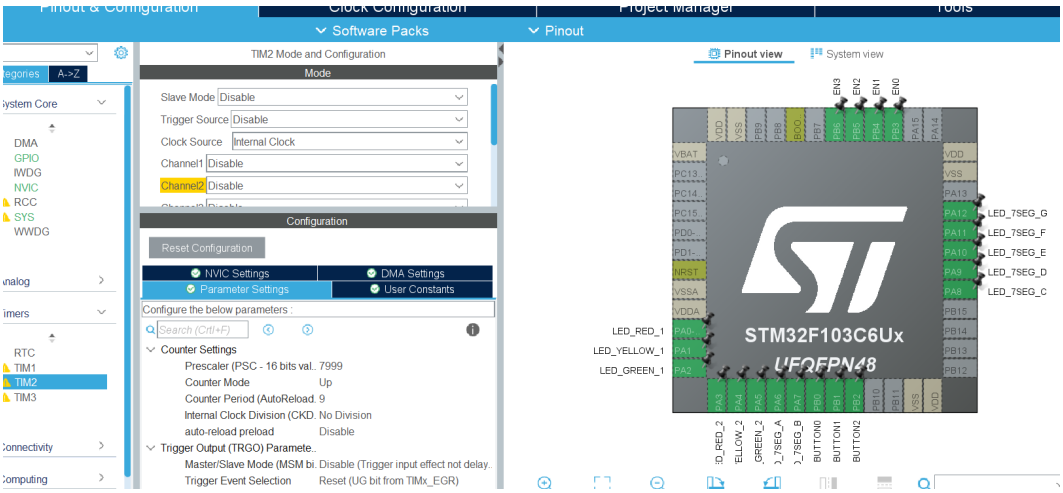


Hình 2: Simulation Protues FSM

1.4 Exercise 3: Create STM32 Project

Your task in this exercise is to create a project that has pin corresponding to the Proteus schematic that you draw in previous section. You need to set up your timer interrupt is about 10ms.

Report:



Hình 3: STM32 Project

1.5 Exercise 4: Modify Timer Parameters

Your task in this exercise is to modify the timer settings so that when we want to change the time duration of the timer interrupt, we change it the least and it will not affect the overall system. For example, the current system we have implemented is that it can blink an LED in 2 Hz, with the timer interrupt duration is 10ms. However, when we want to change the timer interrupt duration to 1ms or 100ms, it will not affect the 2Hz blinking LED.

```
#include "software_timer.h"

int timer_counter[NUMBER_OF_TIMER] = {0};
int timer_flag[NUMBER_OF_TIMER] = {0};
void set_timer(int index, int duration){
  timer_counter[index] = duration/TIME_CYCLE;
}
int timer1_counter = 0;
int timer1_flag = 0;
void set_timer1(int duration){
  timer1_counter = duration;
  timer1_flag = 0;
}
void clear_timer(int index){
  timer_counter[index] = 0;
  timer_flag[index] = 0;
}
int get_time_of_counter(int index){
  return timer_counter[index]*TIME_CYCLE;
}
int is_timer_timeout(int index){
  if(timer_flag[index]){
    timer_flag[index] = 0;
    return 1;
```

```
  }
  return 0;
}
void timer_run(){
  for(int i = 0 ; i < NUMBER_OF_TIMER; ++i){
    if(timer_counter[i] > 0){
      timer_counter[i]--;
      if(timer_counter[i] <= 0) timer_flag[i] = 1;
    }
  }
  if (timer1_counter > 0){
      timer1_counter--;
      if (timer1_counter <= 0){
        timer1_flag = 1;
      }
    }
}
```

Program 1: Source Code

1.6 Exercise 5: Adding code for button debouncing

Following the example of button reading and debouncing in the previous section, your tasks in this exercise are:

- To add new files for input reading and output display,
- To add code for button debouncing,
- To add code for increasing mode when the first button is pressed.

Report:

```
//file BUTTON.h
#ifndef INC_BUTTON_H_
#define INC_BUTTON_H_

#include "global.h"
#include "main.h"
#include "software_timer.h"
extern int state_of_button[NUMBER_OF_BUTTON];
extern int flag_for_pressed[NUMBER_OF_BUTTON];
extern int flag_for_pressed_3s[NUMBER_OF_BUTTON];

extern int register0_key[NUMBER_OF_BUTTON];
extern int register1_key[NUMBER_OF_BUTTON];
extern int register2_key[NUMBER_OF_BUTTON];
extern int button_buffer[NUMBER_OF_BUTTON];

int is_pressed(int index);
```

```
18 int is_pressed_3s(int index);
19 int is_button_released(int index);
20
21 void read_input();
22 void fsm_for_button();
23
24 //file BUTTON.c
25 #include "BUTTON.h"
26 // gan cac gia  t r  ban dau cua state_of_button =
   BUTTON_RELEASED
27 int state_of_button[NUMBER_OF_BUTTON] = {[0 ...
   NUMBER_OF_BUTTON - 1] = BUTTON_RELEASED};
28 int flag_for_pressed[NUMBER_OF_BUTTON] = {0};
29 int flag_for_pressed_3s[NUMBER_OF_BUTTON] = {0};
30
31 int is_pressed(int index){
32   if(index > NUMBER_OF_BUTTON) return 0;
33   if(flag_for_pressed[index]){
34     flag_for_pressed[index] = 0;
35     return 1;
36   }
37   return 0;
38 }
39 int is_pressed_3s(int index){
40   if(index > NUMBER_OF_BUTTON) return 0;
41   if(flag_for_pressed_3s[index]){
42     flag_for_pressed_3s[index] = 0;
43     return 1;
44   }
45   return 0;
46 }
47 int is_button_released(int index){
48   if(button_buffer[index] == BUTTON_IS_RELEASED) return 1;
49   return 0;
50 }
51
52 int register0_key[NUMBER_OF_BUTTON] = {[0 ...
   NUMBER_OF_BUTTON-1] = BUTTON_IS_RELEASED};
53 int register1_key[NUMBER_OF_BUTTON] = {[0 ...
   NUMBER_OF_BUTTON-1] = BUTTON_IS_RELEASED};
54 int register2_key[NUMBER_OF_BUTTON] = {[0 ...
   NUMBER_OF_BUTTON-1] = BUTTON_IS_RELEASED};
55 int button_buffer[NUMBER_OF_BUTTON] = {[0 ...
   NUMBER_OF_BUTTON-1] = BUTTON_IS_RELEASED};
56 void read_input(){
57   for(int i = 0 ; i < NUMBER_OF_BUTTON; ++i){
58     register0_key[i] = register1_key[i];
59     register1_key[i] = register2_key[i];
60     switch(i){
```

```
61      case 0 :
62          register2_key[i] = HAL_GPIO_ReadPin(GPIOB,
    BUTTON0_Pin);
63        break;
64      case 1:
65        register2_key[i] = HAL_GPIO_ReadPin(GPIOB,
    BUTTON1_Pin);
66        break;
67      case 2:
68        register2_key[i] = HAL_GPIO_ReadPin(GPIOB,
    BUTTON2_Pin);
69        break;
70      default:
71        break;
72      }
73      if(register0_key[i] == register1_key[i] &&
    register1_key[i] == register2_key[i]){
74        button_buffer[i] = register2_key[i];
75      }
76    }
77  }
78
79  void fsm_for_button(){
80    for(int i = 0; i < NUMBER_OF_BUTTON; ++i){
81      switch(state_of_button[i]){
82      case BUTTON_RELEASED:
83        if(button_buffer[i] == BUTTON_IS_PRESSED){
84          flag_for_pressed[i] = 1;
85          set_timer(i, 3000);
86          state_of_button[i] = BUTTON_PRESSED;
87        }
88        break;
89      case BUTTON_PRESSED:
90        if(is_timer_timeout(i)){
91          flag_for_pressed_3s[i] = 1;
92          state_of_button[i] = BUTTON_PRESSED_3S;
93        }
94        if(button_buffer[i] == BUTTON_IS_RELEASED){
95          clear_timer(i);
96          state_of_button[i] = BUTTON_RELEASED;
97        }
98        break;
99      case BUTTON_PRESSED_3S:
100       if(button_buffer[i] == BUTTON_IS_RELEASED){
101         state_of_button[i] = BUTTON_RELEASED;
102       }
103       break;
104     default:
105       break;
```

```
    }
  }
}
```

1.7 Exercise 6: Adding code for displaying modes

Your tasks in this exercise are:

- To add code for display mode on seven-segment LEDs, and
- To add code for blinking LEDs depending on the mode that is selected.

```
//file led_7_seg.h
#ifndef INC_LED_7_SEG_H_
#define INC_LED_7_SEG_H_

#include "global.h"
#include "main.h"
#include "software_timer.h"
//LED 7 SEGMENT
extern int led_7_seg_buffer[4];
extern int index_led_7_seg;
//turn_on_led_7seg control the ENi pin
void turn_on_led_7seg(int index);
//display_led_7seg display led 7 segment (0 - 9);
void display_led_7seg(int number);
//update value of buffer of led 7 seg at index 0 and  index 1
//number1 will update buffer[0] and buffer[1]
void update_2_buffer_led_7seg_left(int number1);
//update value of buffer of led 7 seg at index 2 and  index 3
//number2 will update buffer[2] and buffer[3]
void update_2_buffer_led_7seg_right(int number2);
// running in while(1), display led 7 segment 0  to 3 and repeat this index(0 - 1 - 2 - 3 - 0 - 1 -....).
void led_7seg_run();

//file led_7_seg.c
#include "led_7_seg.h"

int led_7_seg_buffer[4] = {0};
int index_led_7_seg = 0;

void turn_on_led_7seg(int index){
  switch(index){
  case 0:
    HAL_GPIO_WritePin( GPIOB, EN0_Pin, RESET);
```

```
    HAL_GPIO_WritePin( GPIOB, EN1_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN2_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN3_Pin, SET);
    break;
  case 1:
    HAL_GPIO_WritePin( GPIOB, EN0_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN1_Pin, RESET);
    HAL_GPIO_WritePin( GPIOB, EN2_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN3_Pin, SET);
    break;
  case 2:
    HAL_GPIO_WritePin( GPIOB, EN0_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN1_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN2_Pin, RESET);
    HAL_GPIO_WritePin( GPIOB, EN3_Pin, SET);
    break;
  case 3:
    HAL_GPIO_WritePin( GPIOB, EN0_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN1_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN2_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN3_Pin, RESET);
    break;
  default:
    HAL_GPIO_WritePin( GPIOB, EN0_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN1_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN2_Pin, SET);
    HAL_GPIO_WritePin( GPIOB, EN3_Pin, SET);
    break;
  }
}


void display_led_7seg(int number){
  switch(number){
  case 0:
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_A_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_B_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_C_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_D_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_E_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_F_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_G_Pin, SET);
    break;
  case 1:
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_A_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_B_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_C_Pin, RESET);
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_D_Pin, SET);
    HAL_GPIO_WritePin(GPIOA, LED_7SEG_E_Pin, SET);
```

```
83      HAL_GPIO_WritePin(GPIOA, LED_7SEG_F_Pin, SET);
84      HAL_GPIO_WritePin(GPIOA, LED_7SEG_G_Pin, SET);
85      break;
86    case 2:
87      HAL_GPIO_WritePin(GPIOA, LED_7SEG_A_Pin, RESET);
88      HAL_GPIO_WritePin(GPIOA, LED_7SEG_B_Pin, RESET);
89      HAL_GPIO_WritePin(GPIOA, LED_7SEG_C_Pin, SET);
90      HAL_GPIO_WritePin(GPIOA, LED_7SEG_D_Pin, RESET);
91      HAL_GPIO_WritePin(GPIOA, LED_7SEG_E_Pin, RESET);
92      HAL_GPIO_WritePin(GPIOA, LED_7SEG_F_Pin, SET);
93      HAL_GPIO_WritePin(GPIOA, LED_7SEG_G_Pin, RESET);
94      break;
95    case 3:
96      HAL_GPIO_WritePin(GPIOA, LED_7SEG_A_Pin, RESET);
97      HAL_GPIO_WritePin(GPIOA, LED_7SEG_B_Pin, RESET);
98      HAL_GPIO_WritePin(GPIOA, LED_7SEG_C_Pin, RESET);
99      HAL_GPIO_WritePin(GPIOA, LED_7SEG_D_Pin, RESET);
100     HAL_GPIO_WritePin(GPIOA, LED_7SEG_E_Pin, SET);
101     HAL_GPIO_WritePin(GPIOA, LED_7SEG_F_Pin, SET);
102     HAL_GPIO_WritePin(GPIOA, LED_7SEG_G_Pin, RESET);
103     break;
104   case 4:
105     HAL_GPIO_WritePin(GPIOA, LED_7SEG_A_Pin, SET);
106     HAL_GPIO_WritePin(GPIOA, LED_7SEG_B_Pin, RESET);
107     HAL_GPIO_WritePin(GPIOA, LED_7SEG_C_Pin, RESET);
108     HAL_GPIO_WritePin(GPIOA, LED_7SEG_D_Pin, SET);
109     HAL_GPIO_WritePin(GPIOA, LED_7SEG_E_Pin, SET);
110     HAL_GPIO_WritePin(GPIOA, LED_7SEG_F_Pin, RESET);
111     HAL_GPIO_WritePin(GPIOA, LED_7SEG_G_Pin, RESET);
112     break;
113   case 5:
114     HAL_GPIO_WritePin(GPIOA, LED_7SEG_A_Pin, RESET);
115     HAL_GPIO_WritePin(GPIOA, LED_7SEG_B_Pin, SET);
116     HAL_GPIO_WritePin(GPIOA, LED_7SEG_C_Pin, RESET);
117     HAL_GPIO_WritePin(GPIOA, LED_7SEG_D_Pin, RESET);
118     HAL_GPIO_WritePin(GPIOA, LED_7SEG_E_Pin, SET);
119     HAL_GPIO_WritePin(GPIOA, LED_7SEG_F_Pin, RESET);
120     HAL_GPIO_WritePin(GPIOA, LED_7SEG_G_Pin, RESET);
121     break;
122   case 6:
123     HAL_GPIO_WritePin(GPIOA, LED_7SEG_A_Pin, RESET);
124     HAL_GPIO_WritePin(GPIOA, LED_7SEG_B_Pin, SET);
125     HAL_GPIO_WritePin(GPIOA, LED_7SEG_C_Pin, RESET);
126     HAL_GPIO_WritePin(GPIOA, LED_7SEG_D_Pin, RESET);
127     HAL_GPIO_WritePin(GPIOA, LED_7SEG_E_Pin, RESET);
128     HAL_GPIO_WritePin(GPIOA, LED_7SEG_F_Pin, RESET);
129     HAL_GPIO_WritePin(GPIOA, LED_7SEG_G_Pin, RESET);
130     break;
131   case 7:
```

```
132     HAL_GPIO_WritePin(GPIOA, LED_7SEG_A_Pin, RESET);
133     HAL_GPIO_WritePin(GPIOA, LED_7SEG_B_Pin, RESET);
134     HAL_GPIO_WritePin(GPIOA, LED_7SEG_C_Pin, RESET);
135     HAL_GPIO_WritePin(GPIOA, LED_7SEG_D_Pin, SET);
136     HAL_GPIO_WritePin(GPIOA, LED_7SEG_E_Pin, SET);
137     HAL_GPIO_WritePin(GPIOA, LED_7SEG_F_Pin, SET);
138     HAL_GPIO_WritePin(GPIOA, LED_7SEG_G_Pin, SET);
139     break;
140   case 8:
141     HAL_GPIO_WritePin(GPIOA, LED_7SEG_A_Pin, RESET);
142     HAL_GPIO_WritePin(GPIOA, LED_7SEG_B_Pin, RESET);
143     HAL_GPIO_WritePin(GPIOA, LED_7SEG_C_Pin, RESET);
144     HAL_GPIO_WritePin(GPIOA, LED_7SEG_D_Pin, RESET);
145     HAL_GPIO_WritePin(GPIOA, LED_7SEG_E_Pin, RESET);
146     HAL_GPIO_WritePin(GPIOA, LED_7SEG_F_Pin, RESET);
147     HAL_GPIO_WritePin(GPIOA, LED_7SEG_G_Pin, RESET);
148     break;
149   case 9:
150     HAL_GPIO_WritePin(GPIOA, LED_7SEG_A_Pin, RESET);
151     HAL_GPIO_WritePin(GPIOA, LED_7SEG_B_Pin, RESET);
152     HAL_GPIO_WritePin(GPIOA, LED_7SEG_C_Pin, RESET);
153     HAL_GPIO_WritePin(GPIOA, LED_7SEG_D_Pin, RESET);
154     HAL_GPIO_WritePin(GPIOA, LED_7SEG_E_Pin, SET);
155     HAL_GPIO_WritePin(GPIOA, LED_7SEG_F_Pin, RESET);
156     HAL_GPIO_WritePin(GPIOA, LED_7SEG_G_Pin, RESET);
157     break;
158   default:
159     break;
160   }
161 }
162
163 void update_2_buffer_led_7seg_left(int number1){
164     led_7_seg_buffer[0] = (number1/1000)/10;
165     led_7_seg_buffer[1] = (number1/1000)%10;
166
167 }
168 void update_2_buffer_led_7seg_right(int number2){
169     led_7_seg_buffer[2] =( number2/1000)/10;
170     led_7_seg_buffer[3] = ( number2/1000)%10;
171 }
172 void led_7seg_run(){
173   // enable the ENi pin ( when index = 0, EN0 = RESET ...);
174   turn_on_led_7seg(index_led_7_seg);
175   display_led_7seg(led_7_seg_buffer[index_led_7_seg]);
176   //switch index after 500ms, index increase 1 and index
       assign 0 when index = 4;
177   if(timer1_flag == 1)//each LED_7SEG run25ms-> 4LED = 1S
178   {
179     index_led_7_seg= (index_led_7_seg+1)%4;
```

```
    set_timer1(25);
  }
}
```

1.8 Exercise 7: Adding code for increasing time duration value for the red LEDs

Your tasks in this exercise are:

- to use the second button to increase the time duration value of the red LEDs
- to use the third button to set the value for the red LEDs.

See full code on github file trafic_fsm.c and led_traffic

```

#include "traffic_fsm.h"

void fsm_traffic_1_run(){
  switch(state_led_traffic_1){
    case RED:
      turn_on_traffic_led_1();
      //STATE STRANSION
      // time out, RED-->GREEN
      if(is_timer_timeout(4)){
        state_led_traffic_1 =GREEN;
        set_timer(4,duration_time_of_GREEN);
      }
    break;
    case YELLOW:
      turn_on_traffic_led_1();
      //STATE STRANSION
      // time out, YELLOW --> RED
      if(is_timer_timeout(4)){
        state_led_traffic_1 = RED;
        set_timer(4,duration_time_of_RED);
      }
    break;
    case GREEN:
      turn_on_traffic_led_1();
      //STATE STRANSION
      // timeout, GREEN-->YELLOW
      if(is_timer_timeout(4)){
        state_led_traffic_1 = YELLOW;
        set_timer(4,duration_time_of_YELLOW);
      }
    break;
    default:
```

```
    break;
  }
}

void fsm_traffic_2_run(){
  switch(state_led_traffic_2){
    case RED:

      turn_on_traffic_led_2();
      //STATE STRANSION
      // time out, RED-->GREEN
      if(is_timer_timeout(5)){
        state_led_traffic_2 = GREEN;
        set_timer(5,duration_time_of_GREEN);
      }
    break;
    case YELLOW:

      turn_on_traffic_led_2();
      //STATE STRANSION
      // time out, YELLOW --> RED
      if(is_timer_timeout(5)){
        state_led_traffic_2 = RED;
        set_timer(5,duration_time_of_RED);
      }
    break;
    case GREEN:

      turn_on_traffic_led_2();
      //STATE STRANSION
      // timeout, GREEN-->YELLOW
      if(is_timer_timeout(5)){
        state_led_traffic_2 = YELLOW;
        set_timer(5,duration_time_of_YELLOW);
      }
    break;
    default:
    break;
  }
}

void fsm_system_run(){
  switch(mode){
    case INIT_SYSTEM:

      state_led_traffic_1 = RED;
      state_led_traffic_2 = GREEN;
      set_timer(3, 500);
      set_timer(4, duration_time_of_RED);
```

```
83      set_timer(5, duration_time_of_GREEN);
84      //STATE STRANSITION
85      mode = NORMAL_MODE;
86    break;
87    case NORMAL_MODE:
88
89      fsm_traffic_1_run();
90      fsm_traffic_2_run();
91      update_2_buffer_led_7seg_left(get_time_of_counter(4))
    ;
92      update_2_buffer_led_7seg_right(get_time_of_counter(5)
    );
93      //STATE STRANSITION
94      //button 0 is pressed,  NORMAl_MODE ->
    MODIFY_RED_MODE
95      if(is_pressed(0)){
96        clear_timer(4);
97        clear_timer(5);
98        state_led_traffic_1 = RED;
99        state_led_traffic_2 = RED;
100       set_timer(6, 500);
101       turn_on_traffic_led_1();
102       turn_on_traffic_led_2();
103       buffer_duration_time = duration_time_of_RED;
104       mode = MODIFY_RED_MODE;
105     }
106   break;
107   case MODIFY_RED_MODE:// LED_RED
108
109     update_2_buffer_led_7seg_left(mode*1000);
110     update_2_buffer_led_7seg_right(buffer_duration_time);
111     //blinking every 500ms
112     if(is_timer_timeout(6)){
113       blinkind_led_traffic_1();
114       blinkind_led_traffic_2();
115       set_timer(6, 500);
116     }
117     //button 1 is pressed, buffer_duration_time increase
    1, if buffer exceed 99, buffer = 0;
118     if(is_pressed(1)){
119       buffer_duration_time = (buffer_duration_time +1000)
    % (100*1000);
120     }
121     //button 2 is pressed,  duration time of system =
    buffer_duration_time;
122     if(is_pressed(2)){
123       duration_time_of_RED = buffer_duration_time;
124     }
125     //STATE STRANSITION
```

```
126       //button 0 is pressed, MODIFY_RED_MODE ->
    MODIFY_YELLOW_MODE
127       if(is_pressed(0)){
128         state_led_traffic_1 = YELLOW;
129         state_led_traffic_2 = YELLOW;
130         set_timer(6, 500);
131         turn_on_traffic_led_1();
132         turn_on_traffic_led_2();
133         buffer_duration_time = duration_time_of_YELLOW;
134         mode = MODIFY_YELLOW_MODE;
135       }
136     break;
```

1.9 Exercise 8: Adding code for increasing time duration value for the amber LEDs

Your tasks in this exercise are:

- to use the second button to increase the time duration value of the amber LEDs
- to use the third button to set the value for the amber LEDs.

Report:

```
1  case MODIFY_YELLOW_MODE:
2      update_2_buffer_led_7seg_left(mode*1000);
3      update_2_buffer_led_7seg_right(buffer_duration_time);
4    //blinking every 500ms
5      if(is_timer_timeout(6)){
6    blinkind_led_traffic_1();
7    blinkind_led_traffic_2();
8    set_timer(6, 500);
9      }
10       //button 1 is pressed, buffer_duration_time increase
     1, buffer exceed 99, buffer = 0;
11     if(is_pressed(1)){
12   buffer_duration_time = (buffer_duration_time +1000)%
     (100*1000);
13     }
14       //button 2 is pressed,  duration time of system =
     buffer_duration_time;
15       if(is_pressed(2)){
16         duration_time_of_YELLOW = buffer_duration_time;
17       }
18
19       //STATE STRANSITION
20       //button 0 is pressed,MODIFY_YELLOW_MODE  ->
    MODIFY_GREEN_MODE
```

```
      if(is_pressed(0)){
        state_led_traffic_1 = GREEN;
        state_led_traffic_2 = GREEN;
        set_timer(6, 500);
        turn_on_traffic_led_1();
        turn_on_traffic_led_2();
        buffer_duration_time = duration_time_of_GREEN;
        mode = MODIFY_GREEN_MODE;
      }
    break;
```

1.10 Exercise 9: Adding code for increasing time duration value for the green LEDs

Your tasks in this exercise are:

- to use the second button to increase the time duration value of the green LEDs
- to use the third button to set the value for the green LEDs.

Report:

```
case MODIFY_GREEN_MODE:
      update_2_buffer_led_7seg_left(mode*1000);
      update_2_buffer_led_7seg_right(buffer_duration_time);
      //blinking every 500ms
      if(is_timer_timeout(6)){
        blinkind_led_traffic_1();
        blinkind_led_traffic_2();
        set_timer(6, 500);
      }
      //button 1 is pressed, buffer_duration_time increase
   1, if buffer exceed 99, buffer = 0;
      if(is_pressed(1)){
        buffer_duration_time = (buffer_duration_time +1000)
   % (100*1000);
      }
      //button 2 is pressed,  duration time of system =
   buffer_duration_time;
      if(is_pressed(2)){
        duration_time_of_GREEN = buffer_duration_time;
      }

      //STATE STRANSITION
      //button 0 is pressed, MODIFY_GREEN_MODE ->
   NORMAL_MODE
      if(is_pressed(0)){
        clear_timer(6);
```

```
        state_led_traffic_1 = RED;
        state_led_traffic_2 = GREEN;
        set_timer(4, duration_time_of_RED);
        set_timer(5, duration_time_of_GREEN);
        mode = NORMAL_MODE;
      }
          break;
    default:
          break;
  }
```

1.11 Exercise 10: To finish the project

Your tasks in this exercise are:

- To integrate all the previous tasks to one final project
- To create a video to show all features in the specification
- To add a report to describe your solution for each exercise.
- To submit your report and code on the BKeL

Report:
All code of Project on Github

https://github.com/quocviet1502/VXL_LAB3/tree/main/STM32PROJECT/Core/Src

1.12 Link Github

Link Github of LAB 3: BUTTONS / SWITCHES
Including STM32Project_traffic_light + Proteus

Link Github: https://github.com/quocviet1502/VXL_LAB3.git

End.