

VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY  
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY  
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



## Microprocessor - Microcontroller

---

### Lab Report - CO3010

# Lab 3

---

Advisor(s): Phan Văn Sỹ

Student(s): Chu Le Hoang 2352346

HO CHI MINH CITY, OCTOBER 2025





## Contents

|   |          |
|---|----------|
| <b>1 Overall</b>                                  | <b>4</b> |
| <b>2 Specifications</b>                           | <b>4</b> |
| <b>3 Exercise 1: Sketch an FSM</b>                | <b>5</b> |
| <b>4 Exercise 2: Proteus Schematic</b>            | <b>6</b> |
| <b>5 Exercise 3: Create STM32 Project</b>         | <b>6</b> |
| 5.1 timer.c . . . . .                             | 6        |
| <b>6 Exercise 4: Modify Timer Parameters</b>      | <b>8</b> |
| <b>7 Exercise 5 to 10</b>                         | <b>8</b> |
| 7.1 main.c . . . . .                              | 8        |
| 7.2 global.h and global.c . . . . .               | 9        |
| 7.3 led_display.c . . . . .                       | 10       |
| 7.4 traffic.c . . . . .                           | 14       |
| 7.5 input_reading.c and input_reading.h . . . . . | 17       |
| 7.6 input_processing.c . . . . .                  | 19       |
| 7.7 Summary . . . . .                             | 22       |



# 1 Overall

Lab schematics and the source codes are submitted via GitHub link: <https://github.com/KennyLe298/MPU-MCU>

# 2 Specifications

An application of a traffic light in a crossroad 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.

### 3 Exercise 1: Sketch an FSM

This is the FSM for the Traffic Light system:

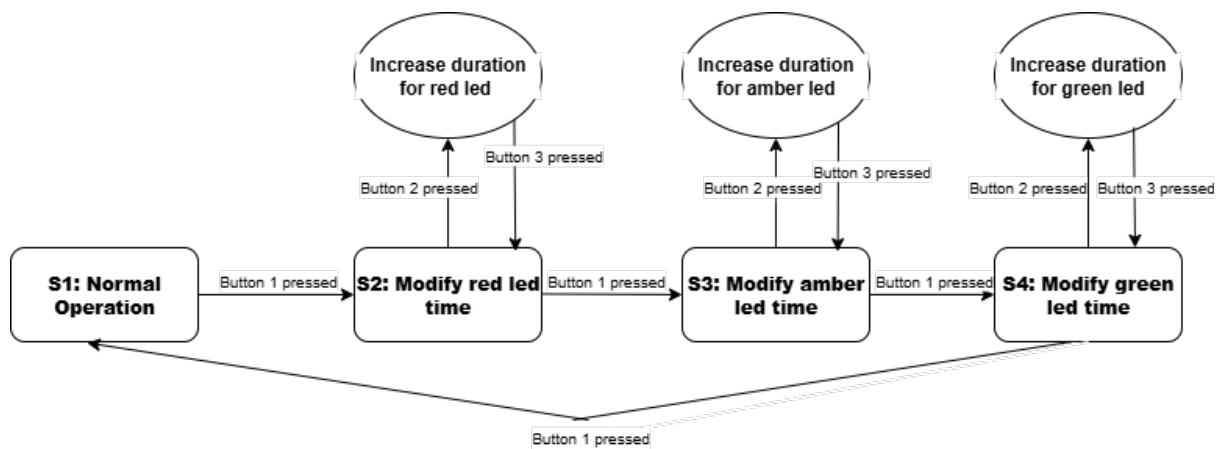


Figure 3.1: FSM sketch

## 4 Exercise 2: Proteus Schematic

Here is the schematic for the Traffic Light system:

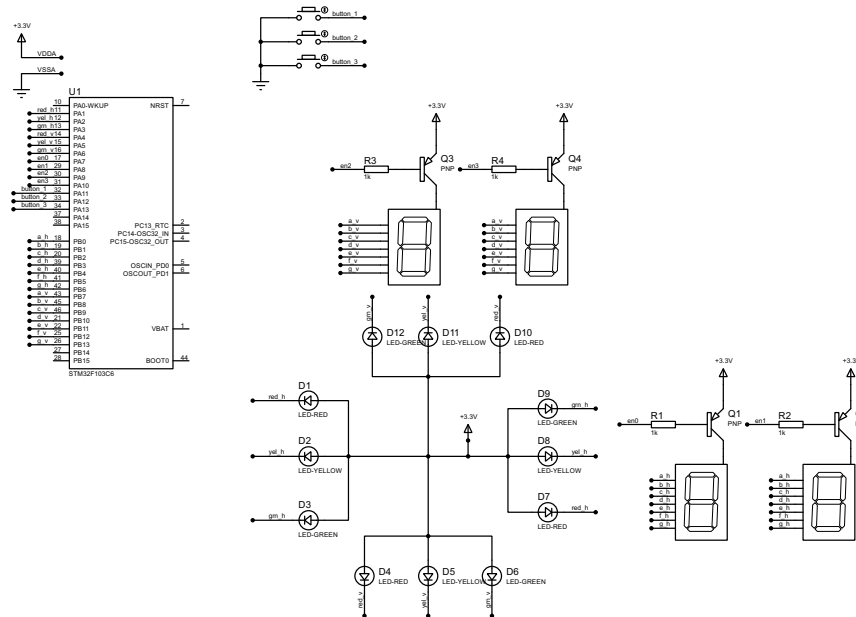


Figure 4.1: Schematic for lab 3

## 5 Exercise 3: Create STM32 Project

### 5.1 timer.c

Source code for timer interrupt 10ms:

```
1 #define TIMER_CYCLE 10
2 int timer1 = 0, timer2 = 0, timer3 = 0, timer4 = 0, timer5 =
   0;
3
4 int flag1 = 0, flag2 = 0, flag3 = 0, flag4 = 0, flag5 = 0;
5
6 void setTimer1(int timer){
7     timer1 = timer / TIMER_CYCLE;
```



```
8   flag1 = 0;
9 }
10 void setTimer2(int timer){
11     timer2 = timer / TIMER_CYCLE;
12     flag2 = 0;
13 }
14 void setTimer3(int timer){
15     timer3 = timer / TIMER_CYCLE;
16     flag3 = 0;
17 }
18 void setTimer4(int timer){
19     timer4 = timer / TIMER_CYCLE;
20     flag4 = 0;
21 }
22 void setTimer5(int timer){
23     timer5 = timer / TIMER_CYCLE;
24     flag5 = 0;
25 }
26
27 void timerRun(){
28     timer1--;
29     timer2--;
30     timer3--;
31     timer4--;
32     timer5--;
33     if(timer1 == 0){
34         flag1 = 1;
35     }
36     if(timer2 == 0){
37         flag2 = 1;
38     }
39     if(timer3 == 0){
40         flag3 = 1;
41     }
42     if(timer4 == 0){
```



```
43     flag4 = 1;
44 }
45 if(timer5 == 0){
46     flag5 = 1;
47 }
48 }
49
50 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
51     if(htim->Instance == TIM2){
52         button_reading ();
53         timerRun();
54     }
55 }
```

## 6 Exercise 4: Modify Timer Parameters

```
1 #define TIMER_CYCLE 10
```

If TIM2 interrupt = 10 ms, we set `TIMER_CYCLE` to 10.

If TIM2 interrupt = 1 ms, we set `TIMER_CYCLE` 1.

If TIM2 interrupt = 100 ms, we set `TIMER_CYCLE` 100.

So when we want to change the timer interrupt duration to 1ms or 100ms, it will not affect the 2 Hz blinking LED

## 7 Exercise 5 to 10

The implementation for this lab project is quite compact so I will create files and implement the functionalities of FSM below as a list of files.

### 7.1 main.c

In the main.c will add these following code and function into the main()

```
1 traffic_reset(); //this function is for initialization of the
    traffic light system which is implemented in traffic.c
2 setTimer5(500); //control the scanning frequency
```





```
3 while (1)
4 {
5     if(flag5){
6         flag5 = 0;
7         scan7seg(scan_state);
8         if(scan_state >=1) scan_state = 0;
9         else scan_state++;
10
11         setTimer5(500);
12     }
13
14     fsm_for_input_processing();
15 }
```

## 7.2 global.h and global.c

The global variables are defined here

```
1 // global.c
2
3 // 1 : normal, 2 : modify red, 3 : modify amber, 4 : modify
  green
4 int current_mode = 1;
5
6 // default duration
7 int red_duration = 5;
8 int amber_duration = 2;
9 int green_duration = 3;
10
11 int scan_state = 0;
```

```
1 //global.h
2 extern int current_mode;
3 extern int red_duration;
4 extern int amber_duration;
5 extern int green_duration;
```

```
6 #define RED    1
7 #define YEL    2
8 #define GRN    3
9 #define OFF    0
10 extern int scan_state;
```

### 7.3 led\_display.c

These led display functions are inherited from Lab 1 exercises. With additional function named updateSeg2Digits for updating only 2 7-segment-led.

```
1 // led_display.c
2 int horstate = 0;
3 int verstate = 0;
4 void setHorLed(int color){
5     switch(color){
6         case RED:
7             HAL_GPIO_WritePin(red_h_GPIO_Port , red_h_Pin ,
8                               GPIO_PIN_RESET);
9             HAL_GPIO_WritePin(yel_h_GPIO_Port , yel_h_Pin ,
10                               GPIO_PIN_SET);
11             HAL_GPIO_WritePin(grn_h_GPIO_Port , grn_h_Pin ,
12                               GPIO_PIN_SET);
13             horstate = RED;
14             break;
15         case YEL:
16             HAL_GPIO_WritePin(red_h_GPIO_Port , red_h_Pin ,
17                               GPIO_PIN_SET);
18             HAL_GPIO_WritePin(yel_h_GPIO_Port , yel_h_Pin ,
19                               GPIO_PIN_RESET);
20             HAL_GPIO_WritePin(grn_h_GPIO_Port , grn_h_Pin ,
21                               GPIO_PIN_SET);
22             horstate = YEL;
23             break;
24         case GRN:
25             HAL_GPIO_WritePin(red_h_GPIO_Port , red_h_Pin ,
```



```
GPIO_PIN_SET);
20 HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin,
GPIO_PIN_SET);
21 HAL_GPIO_WritePin(grn_h_GPIO_Port, grn_h_Pin,
GPIO_PIN_RESET);
22 horstate = GRN;
23 break;
24 default:
25 HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin,
GPIO_PIN_SET);
26 HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin,
GPIO_PIN_SET);
27 HAL_GPIO_WritePin(grn_h_GPIO_Port, grn_h_Pin,
GPIO_PIN_SET);
28 horstate = OFF;
29 break;
30 }
31 }
32 void setVerLed(int color){
33 switch(color){
34 case RED:
35 HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin,
GPIO_PIN_RESET);
36 HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin,
GPIO_PIN_SET);
37 HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin,
GPIO_PIN_SET);
38 verstate = RED;
39 break;
40 case YEL:
41 HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin,
GPIO_PIN_SET);
42 HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin,
GPIO_PIN_RESET);
43 HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin,
```

```
GPIO_PIN_SET);
44     verstate = YEL;
45     break;
46     case GRN:
47         HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin,
GPIO_PIN_SET);
48         HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin,
GPIO_PIN_SET);
49         HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin,
GPIO_PIN_RESET);
50         verstate = GRN;
51         break;
52         default:
53             HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin,
GPIO_PIN_SET);
54             HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin,
GPIO_PIN_SET);
55             HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin,
GPIO_PIN_SET);
56             verstate = OFF;
57             break;
58     }
59 }
60 int segment_buffer[4] = {0};
61 GPIO_PinState PinMap[11][7] = {
62     {0, 0, 0, 0, 0, 0, 1}, //0
63     {1, 0, 0, 1, 1, 1, 1}, //1
64     {0, 0, 1, 0, 0, 1, 0}, //2
65     {0, 0, 0, 0, 1, 1, 0}, //3
66     {1, 0, 0, 1, 1, 0, 0}, //4
67     {0, 1, 0, 0, 1, 0, 0}, //5
68     {0, 1, 0, 0, 0, 0, 0}, //6
69     {0, 0, 0, 1, 1, 1, 1}, //7
70     {0, 0, 0, 0, 0, 0, 0}, //8
71     {0, 0, 0, 0, 1, 0, 0}, //9
```



```
72 {1, 1, 1, 1, 1, 1, 1} //ALL LED TURN OFF
73 };
74 GPIO_TypeDef* segHorPorts[7] = {a_h_GPIO_Port, b_h_GPIO_Port,
    c_h_GPIO_Port, d_h_GPIO_Port, e_h_GPIO_Port,
    f_h_GPIO_Port, g_h_GPIO_Port};
75 uint16_t segHorPins[7] = {a_h_Pin, b_h_Pin, c_h_Pin, d_h_Pin,
    e_h_Pin, f_h_Pin, g_h_Pin};
76 GPIO_TypeDef* segVerPorts[7] = {a_v_GPIO_Port, b_v_GPIO_Port,
    c_v_GPIO_Port, d_v_GPIO_Port, e_v_GPIO_Port,
    f_v_GPIO_Port, g_v_GPIO_Port};
77 uint16_t segVerPins[7] = {a_v_Pin, b_v_Pin, c_v_Pin, d_v_Pin,
    e_v_Pin, f_v_Pin, g_v_Pin};
78 void set7segHor(int num){
79     if(num < 0 || num > 10) num = 10;
80     for(int s = 0; s < 7; s++){
81         HAL_GPIO_WritePin(segHorPorts[s], segHorPins[s], (PinMap[
            num][s] == 0) ? GPIO_PIN_RESET : GPIO_PIN_SET);
82     }
83 }
84 void set7segVer(int num){
85     if(num < 0 || num > 10) num = 10;
86     for(int s = 0; s < 7; s++){
87         HAL_GPIO_WritePin(segVerPorts[s], segVerPins[s], (PinMap[
            num][s] == 0) ? GPIO_PIN_RESET : GPIO_PIN_SET);
88     }
89 }
90 void scan7seg(int state){
91     state = state%2; // only scan 2 leds
92     switch(state){
93         case 0:
94             HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, GPIO_PIN_RESET)
95             ;
96             HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, GPIO_PIN_SET);
97             HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, GPIO_PIN_RESET)
98             ;
```

```
97     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, GPIO_PIN_SET);
98     set7segHor(segment_buffer[0]);
99     set7segVer(segment_buffer[2]);
100     break;
101     case 1:
102     HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, GPIO_PIN_SET);
103     HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, GPIO_PIN_RESET)
104     ;
105     HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, GPIO_PIN_SET);
106     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, GPIO_PIN_RESET)
107     ;
108     set7segHor(segment_buffer[1]);
109     set7segVer(segment_buffer[3]);
110     break;
111     default:
112     break;
113 }
114 }
115 void updateSeg(int a, int b, int c, int d){
116     segment_buffer[0] = a;
117     segment_buffer[1] = b;
118     segment_buffer[2] = c;
119     segment_buffer[3] = d;
120 }
121 void updateSeg2Digits(int a, int b){
122     segment_buffer[0] = a/10;
123     segment_buffer[1] = a%10;
124     segment_buffer[2] = b/10;
125     segment_buffer[3] = b%10;
126 }
```

## 7.4 traffic.c

This file holds the implementation of the traffic light system running automatically.

```
1 static int state = 0;
```



```
2 static int counter = 0;
3 void traffic_fsm(void) {
4     switch(state) {
5         case 0: // green hor, red ver
6             setHorLed(GRN);
7             setVerLed(RED);
8             if(flag2) {
9                 flag2 = 0;
10                counter--;
11                if(counter <= 0) {
12                    counter = amber_duration;
13                    setTimer2(1000);
14                    state = 1;
15                } else {
16                    setTimer2(1000);
17                }
18            }
19            updateSeg2Digits(counter, counter + amber_duration);
20            break;
21
22         case 1: // yel hor, red ver
23             setHorLed(YEL);
24             setVerLed(RED);
25             if(flag2) {
26                 flag2 = 0;
27                 counter--;
28                 if(counter <= 0) {
29                     counter = green_duration;
30                     setTimer2(1000);
31                     state = 2;
32                 } else {
33                     setTimer2(1000);
34                 }
35            }
36            updateSeg2Digits(counter, counter);
```

```
37     break;
38
39     case 2: // red hor , green ver
40         setHorLed(RED);
41         setVerLed(GRN);
42         if(flag2) {
43             flag2 = 0;
44             counter--;
45             if(counter <= 0) {
46                 counter = amber_duration;
47                 setTimer2(1000);
48                 state = 3;
49             } else {
50                 setTimer2(1000);
51             }
52         }
53         updateSeg2Digits(counter + amber_duration, counter);
54         break;
55
56     case 3: // red hor, yel ver
57         setHorLed(RED);
58         setVerLed(YEL);
59         if(flag2) {
60             flag2 = 0;
61             counter--;
62             if(counter <= 0) {
63                 counter = green_duration;
64                 setTimer2(1000);
65                 state = 0;
66             } else {
67                 setTimer2(1000);
68             }
69         }
70         updateSeg2Digits(counter, counter);
71         break;
```





```
72 }
73 }
74 void traffic_reset(void) {
75     state = 0;
76     counter = green_duration;
77     setTimer2(1000);
78 }
79 void traffic_set_counter(int value) {
80     counter = value;
81 }
```

## 7.5 input\_reading.c and input\_reading.h

This file holds the implementation for reading input from 3 buttons

```
1 //input_reading.h
2 #define NO_OF_BUTTONS 3 // 3 buttons
3
4 #define DURATION_FOR_AUTO_INCREASING 100 //1s
5
6 #define PRESSED GPIO_PIN_RESET
7 #define RELEASED GPIO_PIN_SET
8
9
10 //input_reading.c
11 static GPIO_PinState buttonBuffer[NO_OF_BUTTONS]; //buffer
12     that the final result is stored after debouncing
13 static GPIO_PinState debounceButtonBuffer1[NO_OF_BUTTONS]; //
14     two buffer for debouncing
15 static GPIO_PinState debounceButtonBuffer2[NO_OF_BUTTONS];
16 static uint8_t flagForButtonPress1s[NO_OF_BUTTONS]; //flag
17     for a button to be pressed for more than 1 sec
18 static uint16_t counterForButtonPress1s[NO_OF_BUTTONS]; //
19     counter for auto increase value after button pressed more
20     than 1 sec
21 GPIO_TypeDef* buttonPorts[NO_OF_BUTTONS] = {
22     button_1_GPIO_Port, button_2_GPIO_Port, button_3_GPIO_Port
23 };
```

```
8 uint16_t buttonPins[NO_OF_BUTTONS] = {button_1_Pin,
    button_2_Pin, button_3_Pin};
9 void button_reading(void){
10     for(int i = 0; i < NO_OF_BUTTONS; i++){
11         debounceButtonBuffer2[i] = debounceButtonBuffer1[i];
12         debounceButtonBuffer1[i] = HAL_GPIO_ReadPin(buttonPorts[i],
            buttonPins[i]);
13         if(debounceButtonBuffer1[i] == debounceButtonBuffer2[i]){
14             buttonBuffer[i] = debounceButtonBuffer1[i]; //accept if
                stable
15             if(buttonBuffer[i] == PRESSED){ //start counting when
                button is pressed
16                 if(counterForButtonPress1s[i] <
                    DURATION_FOR_AUTO_INCREASING){
17                     counterForButtonPress1s[i]++;
18                 }
19                 else{
20                     flagForButtonPress1s[i] = 1; //flag when 1 sec
                        passed since button pressed
21                     counterForButtonPress1s[i] = 0;
22                 }
23             }
24             else{ // released
25                 counterForButtonPress1s[i] = 0;
26                 flagForButtonPress1s[i] = 0; //reset for repetition
27             }
28         }
29     }
30 }
31 unsigned char is_button_pressed(uint8_t index){
32     if(index >= NO_OF_BUTTONS) return 0;
33     return (buttonBuffer[index] == PRESSED);
34 }
35 unsigned char is_button_pressed_1s (unsigned char index){
36     if(index >= NO_OF_BUTTONS) return 0xff ;
```

```
37     return (flagForButtonPress1s[index] == 1) ;  
38 }
```

## 7.6 input\_processing.c

In this file is the implementation of the FSM and also the processing of the inputs from 3 buttons

```
1 // for modify (1-99)  
2 int temp_value = 1;  
3 int blink_state = 0;  
4 void fsm_for_input_processing(void){  
5     static uint8_t last_mode_button = 1;  
6     static uint8_t last_inc_button = 0;  
7     static uint8_t last_set_button = 0;  
8     if(is_button_pressed(0) && !last_mode_button){  
9         current_mode++;  
10        if(current_mode > 4) current_mode = 1;  
11  
12        if(current_mode == 1) {  
13            traffic_reset();  
14        }  
15        else {  
16            HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin|yel_h_Pin|  
17            grn_h_Pin, GPIO_PIN_SET);  
18            HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin|yel_v_Pin|  
19            grn_v_Pin, GPIO_PIN_SET);  
20        }  
21        if(current_mode == 2) temp_value = red_duration;  
22        else if(current_mode == 3) temp_value = amber_duration;  
23        else if(current_mode == 4) temp_value = green_duration;  
24        //2Hz blink  
25        setTimer1(250); // flag1  
26        last_inc_button = 1;  
27        last_set_button = 1;  
28    }
```

```
27 last_mode_button = is_button_pressed(0);
28 if(current_mode == 1){
29     if(flag2 == 0 && timer2 == 0){
30         setTimer2(1000);
31     }
32     traffic_fsm();
33 }
34 else if(current_mode == 2 || current_mode == 3 ||
35         current_mode == 4){
36     updateSeg2Digits(current_mode, temp_value);
37     if(flag1){
38         flag1 = 0;
39         blink_state = !blink_state;
40         setTimer1(250);
41         if(current_mode == 2){
42             if(blink_state){
43                 HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin,
44                                     GPIO_PIN_RESET);
45                 HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin,
46                                     GPIO_PIN_RESET);
47             }
48             else{
49                 HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin,
50                                     GPIO_PIN_SET);
51                 HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin,
52                                     GPIO_PIN_SET);
53             }
54         }
55         else if(current_mode == 3){
56             if(blink_state){
57                 HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin,
58                                     GPIO_PIN_RESET);
59                 HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin,
60                                     GPIO_PIN_RESET);
61             }
62         }
63     }
64 }
```



```
55     else{
56         HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin,
GPIO_PIN_SET);
57         HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin,
GPIO_PIN_SET);
58     }
59 }
60 else if(current_mode == 4){
61     if(blink_state){
62         HAL_GPIO_WritePin(grn_h_GPIO_Port, grn_h_Pin,
GPIO_PIN_RESET);
63         HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin,
GPIO_PIN_RESET);
64     }
65     else{
66         HAL_GPIO_WritePin(grn_h_GPIO_Port, grn_h_Pin,
GPIO_PIN_SET);
67         HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin,
GPIO_PIN_SET);
68     }
69 }
70 }
71 if(is_button_pressed(1) && !last_inc_button){
72     temp_value++;
73     if(temp_value > 99) temp_value = 1;
74 }
75 if(is_button_pressed_1s(1)){
76     temp_value++;
77     if(temp_value > 99) temp_value = 1;
78 }
79 last_inc_button = is_button_pressed(1);
80 if(is_button_pressed(2) && !last_set_button){
81     if(current_mode == 2){
82         if(temp_value >= 2){
83             red_duration = temp_value;
```

```
84         if(red_duration > amber_duration){
85             green_duration = red_duration - amber_duration;
86         }
87         else{
88             amber_duration = 1; //auto set amber to be 1 when
red is small
89             green_duration = red_duration - 1;
90         }
91     }
92 }
93 else if(current_mode == 3){
94     if(temp_value > 0 && temp_value < red_duration){
95         amber_duration = temp_value;
96         green_duration = red_duration - amber_duration;
97     }
98 }
99 else if(current_mode == 4){
100     if(temp_value > 0 && temp_value < red_duration){
101         green_duration = temp_value;
102         amber_duration = red_duration - green_duration;
103     }
104 }
105 }
106 last_set_button = is_button_pressed(2);
107 }
108 }
```

## 7.7 Summary

There are several other header files but those files are short and only for defining function names so I did not put in this report. The demo of this project will be presented onsite and graded by lecturer.