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

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:

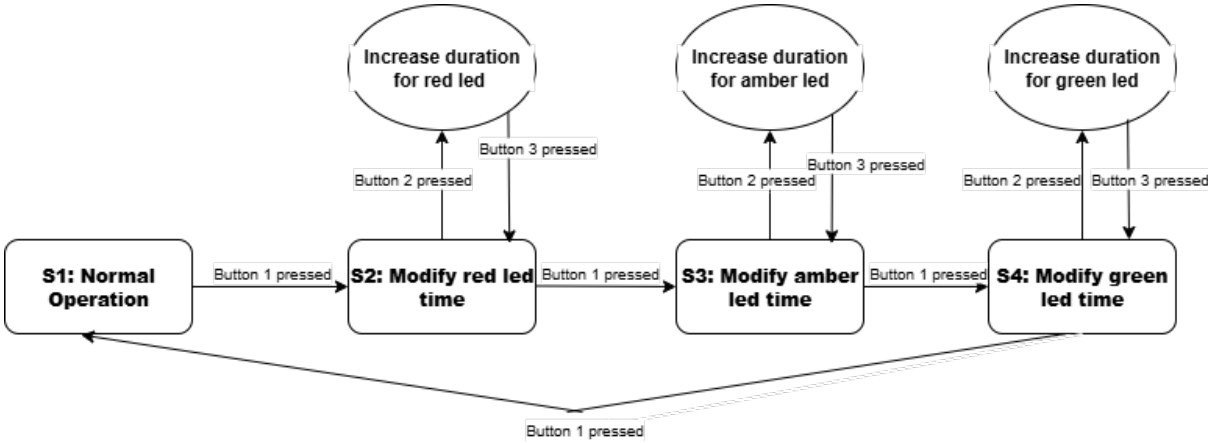

Figure 3.1: FSM sketch

4 Exercise 2: Proteus Schematic

Here is the schematic for the Traffic Light system:

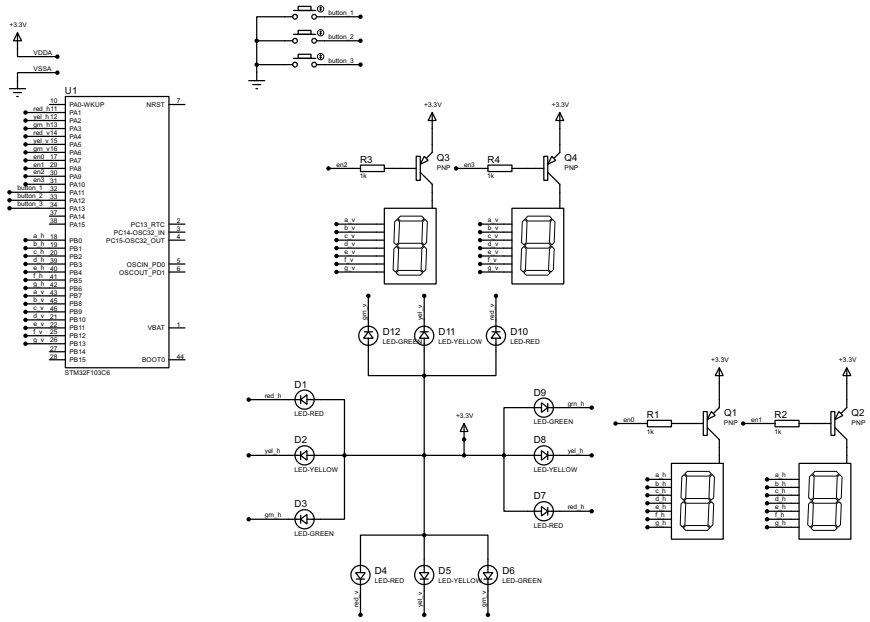

Figure 4.1: Schematic for lab 3

5 Exercise 3: Create STM32 Project

5.1 timer.c

Source code for timer interrupt 10ms:

```
#define TIMER_CYCLE 10
int timer1 = 0, timer2 = 0, timer3 = 0, timer4 = 0, timer5 =
    0;

int flag1 = 0, flag2 = 0, flag3 = 0, flag4 = 0, flag5 = 0;

void setTimer1(int timer){
  timer1 = timer / TIMER_CYCLE;
```

```
  flag1 = 0;
}
void setTimer2(int timer){
  timer2 = timer / TIMER_CYCLE;
  flag2 = 0;
}
void setTimer3(int timer){
  timer3 = timer / TIMER_CYCLE;
  flag3 = 0;
}
void setTimer4(int timer){
  timer4 = timer / TIMER_CYCLE;
  flag4 = 0;
}
void setTimer5(int timer){
  timer5 = timer / TIMER_CYCLE;
  flag5 = 0;
}

void timerRun(){
  timer1--;
  timer2--;
  timer3--;
  timer4--;
  timer5--;
  if(timer1 == 0){
    flag1 = 1;
  }
  if(timer2 == 0){
    flag2 = 1;
  }
  if(timer3 == 0){
    flag3 = 1;
  }
  if(timer4 == 0){
```

```
    flag4 = 1;
  }
  if(timer5 == 0){
    flag5 = 1;
  }
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
  if(htim->Instance == TIM2){
    button_reading ();
    timerRun();
  }
}
```

6 Exercise 4: Modify Timer Parameters

```
#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()

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

```
while (1)
{
  if(flag5){
    flag5 = 0;
    scan7seg(scan_state);
    if(scan_state >=1) scan_state = 0;
    else scan_state++;

    setTimer5(500);
  }

  fsm_for_input_processing();
}
```

7.2 global.h and global.c

The global variables are defined here

```
// global.c

// 1 : normal, 2 : modify red, 3 : modify amber, 4 : modify
   green
int current_mode = 1;

// default duration
int red_duration = 5;
int amber_duration = 2;
int green_duration = 3;

int scan_state = 0;
```

```
//global.h
extern int current_mode;
extern int red_duration;
extern int amber_duration;
extern int green_duration;
```

```
#define RED   1
#define YEL   2
#define GRN   3
#define OFF   0
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.

```
// led_display.c
int horstate = 0;
int verstate = 0;
void setHorLed(int color){
  switch(color){
    case RED:
    HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin,
   GPIO_PIN_RESET);
    HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin,
   GPIO_PIN_SET);
    HAL_GPIO_WritePin(grn_h_GPIO_Port, grn_h_Pin,
   GPIO_PIN_SET);
    horstate = RED;
    break;
    case YEL:
    HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin,
   GPIO_PIN_SET);
    HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin,
   GPIO_PIN_RESET);
    HAL_GPIO_WritePin(grn_h_GPIO_Port, grn_h_Pin,
   GPIO_PIN_SET);
    horstate = YEL;
    break;
    case GRN:
    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      HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, GPIO_PIN_SET);
96      HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, GPIO_PIN_RESET)
       ;
```

```
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     HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, GPIO_PIN_SET);
105     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, GPIO_PIN_RESET)
    ;
106     set7segHor(segment_buffer[1]);
107     set7segVer(segment_buffer[3]);
108     break;
109     default:
110     break;
111   }
112 }
113 void updateSeg(int a, int b, int c, int d){
114   segment_buffer[0] = a;
115   segment_buffer[1] = b;
116   segment_buffer[2] = c;
117   segment_buffer[3] = d;
118 }
119 void updateSeg2Digits(int a, int b){
120   segment_buffer[0] = a/10;
121   segment_buffer[1] = a%10;
122   segment_buffer[2] = b/10;
123   segment_buffer[3] = b%10;
124 }
```

7.4 traffic.c

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

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

```
static int counter = 0;
void traffic_fsm(void) {
  switch(state) {
    case 0: // green hor, red ver
    setHorLed(GRN);
    setVerLed(RED);
    if(flag2) {
      flag2 = 0;
      counter--;
      if(counter <= 0) {
        counter = amber_duration;
        setTimer2(1000);
        state = 1;
      } else {
        setTimer2(1000);
      }
    }
    updateSeg2Digits(counter, counter + amber_duration);
    break;

    case 1: // yel hor, red ver
    setHorLed(YEL);
    setVerLed(RED);
    if(flag2) {
      flag2 = 0;
      counter--;
      if(counter <= 0) {
        counter = green_duration;
        setTimer2(1000);
        state = 2;
      } else {
        setTimer2(1000);
      }
    }
    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;
```

```
  }
}
void traffic_reset(void) {
  state = 0;
  counter = green_duration;
  setTimer2(1000);
}
void traffic_set_counter(int value) {
  counter = value;
}
```

7.5 input_reading.c and input_reading.h

This file holds the implementation for reading input from 3 buttons

```
//input_reading.h
#define NO_OF_BUTTONS 3 // 3 buttons

#define DURATION_FOR_AUTO_INCREASING 100 //1s

#define PRESSED GPIO_PIN_RESET
#define RELEASED GPIO_PIN_SET
```

```
//input_reading.c
static GPIO_PinState buttonBuffer[NO_OF_BUTTONS]; //buffer
   that the final result is stored after debouncing
static GPIO_PinState debounceButtonBuffer1[NO_OF_BUTTONS]; //
   two buffer for debouncing
static GPIO_PinState debounceButtonBuffer2[NO_OF_BUTTONS];
static uint8_t flagForButtonPress1s[NO_OF_BUTTONS]; //flag
   for a button to be pressed for more than 1 sec
static uint16_t counterForButtonPress1s[NO_OF_BUTTONS]; //
   counter for auto increase value after button pressed more
   than 1 sec
GPIO_TypeDef* buttonPorts[NO_OF_BUTTONS] = {
   button_1_GPIO_Port, button_2_GPIO_Port, button_3_GPIO_Port
   };
```

```
uint16_t buttonPins[NO_OF_BUTTONS] = {button_1_Pin, button_2_Pin, button_3_Pin};
void button_reading(void){
  for(int i = 0; i < NO_OF_BUTTONS; i++){
    debounceButtonBuffer2[i] = debounceButtonBuffer1[i];
    debounceButtonBuffer1[i] = HAL_GPIO_ReadPin(buttonPorts[i], buttonPins[i]);
    if(debounceButtonBuffer1[i] == debounceButtonBuffer2[i]){
      buttonBuffer[i] = debounceButtonBuffer1[i]; //accept if stable
      if(buttonBuffer[i] == PRESSED){ //start counting when button is pressed
        if(counterForButtonPress1s[i] < DURATION_FOR_AUTO_INCREASING){
          counterForButtonPress1s[i]++;
        }
        else{
          flagForButtonPress1s[i] = 1; //flag when 1 sec passed since button pressed
          counterForButtonPress1s[i] = 0;
        }
      }
      else{ // released
        counterForButtonPress1s[i] = 0;
        flagForButtonPress1s[i] = 0; //reset for repetition
      }
    }
  }
}
unsigned char is_button_pressed(uint8_t index){
  if(index >= NO_OF_BUTTONS) return 0;
  return (buttonBuffer[index] == PRESSED);
}
unsigned char is_button_pressed_1s (unsigned char index){
  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 = 0;
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|
     grn_h_Pin, GPIO_PIN_SET);
17       HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin|yel_v_Pin|
     grn_v_Pin, GPIO_PIN_SET);
18     }
19     if(current_mode == 2) temp_value = red_duration;
20     else if(current_mode == 3) temp_value = amber_duration;
21     else if(current_mode == 4) temp_value = green_duration;
22     //2Hz blink
23     setTimer1(250); // flag1
24     last_inc_button = 1;
25     last_set_button = 1;
26   }
```

```
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 ||
     current_mode == 4){
35    updateSeg2Digits(current_mode, temp_value);
36    if(flag1){
37      flag1 = 0;
38      blink_state = !blink_state;
39      setTimer1(250);
40      if(current_mode == 2){
41        if(blink_state){
42          HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin,
     GPIO_PIN_RESET);
43          HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin,
     GPIO_PIN_RESET);
44        }
45        else{
46          HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin,
     GPIO_PIN_SET);
47          HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin,
     GPIO_PIN_SET);
48        }
49      }
50      else if(current_mode == 3){
51        if(blink_state){
52          HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin,
     GPIO_PIN_RESET);
53          HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin,
     GPIO_PIN_RESET);
54        }
```

```
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.