

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY  
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY  
Faculty of Computer Science and Engineering



---

CC02 — Lab Report

# Microprocessor - Microcontroller Lab 3

---

**Supervisors:** Nguyen Thien An  
**Students:** Nguyen Thanh Tai 2252722

Ho Chi Minh City, October 29, 2024



## Contents

<b>1</b>	<b>Exercise</b>	<b>2</b>
1.1	Exercise 1 . . . . .	3
1.2	Exercise 2 . . . . .	5
1.3	Exercise 3 . . . . .	6
1.4	Exercise 4 . . . . .	7
1.5	Exercise 5 . . . . .	9
1.6	Exercise 6 . . . . .	11
1.7	Exercise 7 . . . . .	18
1.8	Exercise 8 . . . . .	19
1.9	Exercise 9 . . . . .	20
1.10	Exercise 10 . . . . .	21
	<b>References</b>	<b>22</b>

## 1 Exercise

The GitHub link for the lab schematics is at [here](https://github.com/ThanhTaiNguyen24/mcu-mpu-lab1) or in this link: <https://github.com/ThanhTaiNguyen24/mcu-mpu-lab1>.

The schematic for the exercises from 1 to 10 is located here:

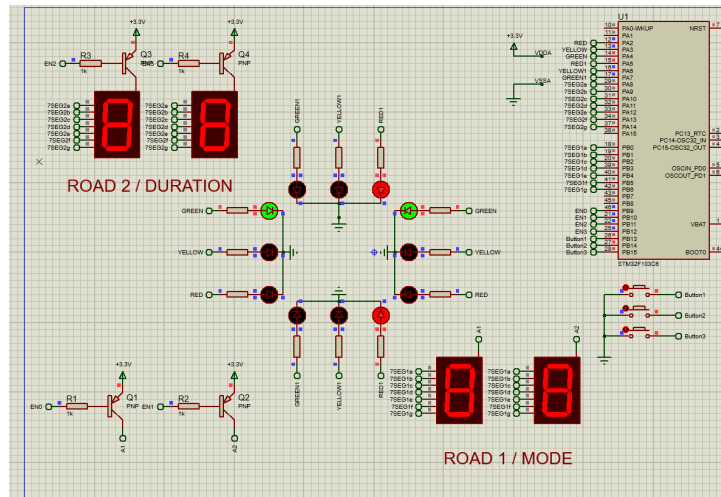
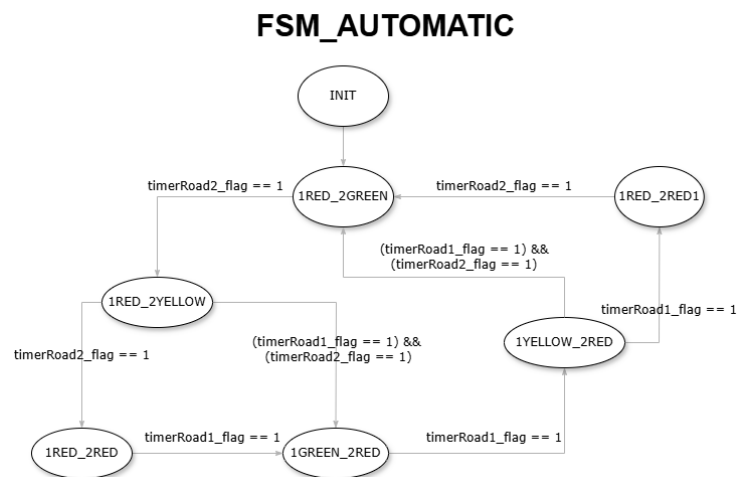


Figure 1: The schematic for Lab 3.

## 1.1 Exercise 1

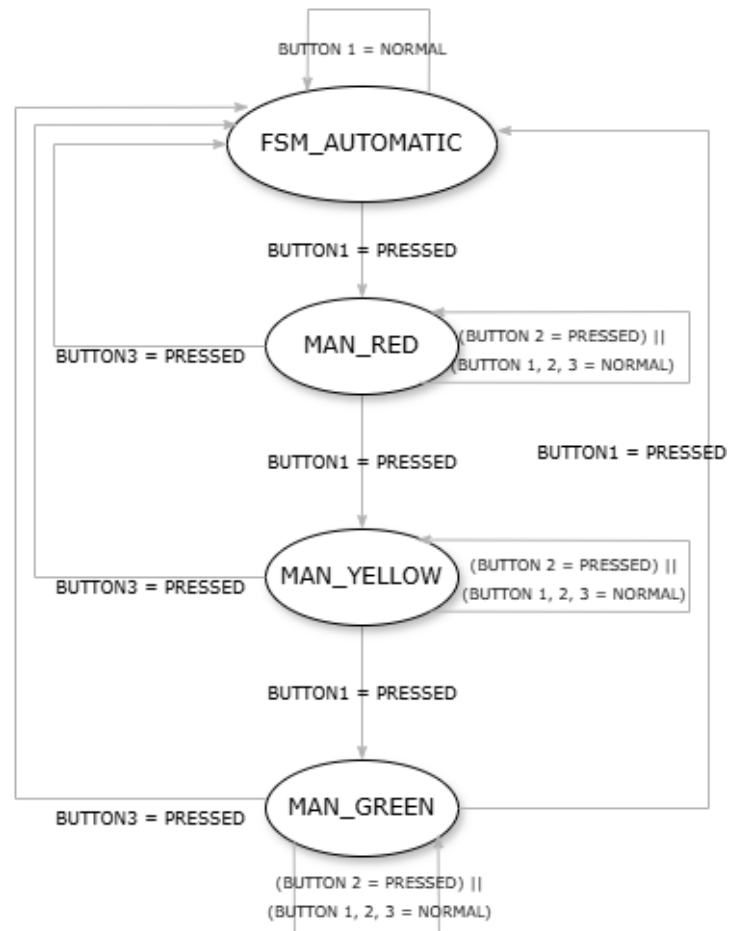
An Finite State Machine for Mode 1 - Normal mode: The traffic light application is running normally is located here:



**Figure 2:** The Finite State Machine for normal mode.

An Finite State Machine for 3 buttons application is located here:

## FSM\_MANUAL



**Figure 3:** The Finite State Machine for 3 buttons application.



## 1.2 Exercise 2

Can be found at 1.

### 1.3 Exercise 3

A project that has pin corresponding to the Proteus schematic with 10ms timer interrupt.

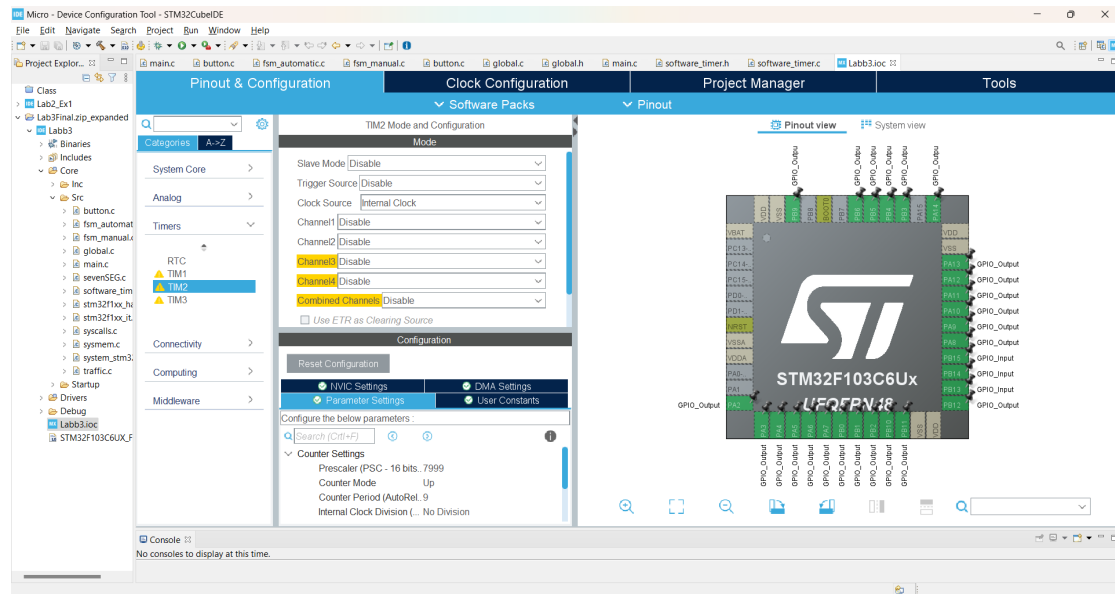


Figure 4: Lab 3 project.

## 1.4 Exercise 4

This is the source code for the exercise 4:

```
1 int timerRoad1_counter = 0;
2 int timerRoad2_counter = 0;
3 int timer2_counter = 0;
4
5
6 int timerRoad1_flag = 0;
7 int timerRoad2_flag = 0;
8 int timer2_flag = 0;
9
10
11
12
13 void setTimerRoad1(int duration){
14     timerRoad1_counter = duration/timercycle;
15     timerRoad1_flag = 0;
16 }
17
18 void setTimerRoad2(int duration){
19     timerRoad2_counter = duration/timercycle;
20     timerRoad2_flag = 0;
21 }
22
23 void setTimer2(){
24     timer2_counter = (1000 / MAX_LED) / timercycle;
25     timer2_flag = 0;
26 }
27
28 void timerRun(){
29     if (timerRoad1_counter > 0){
30         timerRoad1_counter--;
31         if(timerRoad1_counter <= 0){
32             timerRoad1_flag = 1;
33         }
34     }
35     if (timerRoad2_counter > 0){
36         timerRoad2_counter--;
37         if(timerRoad2_counter <= 0){
38             timerRoad2_flag = 1;
39         }
40     }
41     if (timer2_counter > 0){
42         timer2_counter--;
43         road1duration-=10;
44         road2duration-=10;
45         if(timer2_counter <= 0){
46             timer2_flag = 1;
```





```
47     }  
48 }  
49 }
```

## 1.5 Exercise 5

This is the source code for button debouncing in the exercise 5:

```
1  GPIO_TypeDef* portB[number_of_buttons] = {GPIOB, GPIOB, GPIOB};
2  uint16_t buttons[number_of_buttons] = {GPIO_PIN_13, GPIO_PIN_14, GPIO_PIN_15};
3
4  int button_flag[number_of_buttons] = {0, 0, 0};
5
6  int keyReg0[number_of_buttons];
7  int keyReg1[number_of_buttons];
8  int keyReg2[number_of_buttons];
9  int keyReg3[number_of_buttons];
10
11
12  int button_pressed_counter[number_of_buttons] = {200, 200, 200};
13
14
15  int checkflag(int index){
16      if(button_flag[index] == 1){
17          button_flag[index] = 0;
18          return 1;
19      }
20      return 0;
21  }
22
23
24  void getKeyInput(void) {
25      for(int i = 0; i < number_of_buttons; i++){
26          keyReg0[i] = keyReg1[i];
27          keyReg1[i] = keyReg2[i];
28          keyReg2[i] = HAL_GPIO_ReadPin(portB[i], buttons[i]);
29
30          if ((keyReg0[i] == keyReg1[i]) && (keyReg1[i] == keyReg2[i])) {
31              if (keyReg3[i] != keyReg2[i]){
32                  keyReg3[i] = keyReg2[i];
33                  if (keyReg2[i] <= pressed_button){
34                      button_pressed_counter[i] = 200;
35                      button_flag[i] = 1;
36                  }
37              } else {
38                  button_pressed_counter[i]--;
39                  if (button_pressed_counter[i] <= 0){
40                      keyReg3[i] = normal_button;
41                  }
42              }
43          }
44      }
45  }
```

This is the source code for increasing mode when the first button is pressed in the exercise 5:



```
1 void checkManual(){  
2     updateLedBuffer();  
3     if (checkflag(0) == 1){  
4         traffic_turnoffall();  
5         status = MAN_RED;  
6         setTimerRoad1(100);  
7         setTimerRoad2(100);  
8     }  
9 }
```

## 1.6 Exercise 6

This is the source code for display mode on seven-segment LEDS in the exercise 6:

```
1  const int MAX_LED = 2;
2  int index_led = 0;
3  int led_bufferRoad1 [2] = {0, 0};
4  int led_bufferRoad2 [2] = {0, 0};
5
6
7
8  void updateLedBuffer(){
9      if (timer2_flag == 1){
10         led_bufferRoad1[0] = (road1duration / 1000) / 10;
11         led_bufferRoad1[1] = (road1duration / 1000) % 10;
12         led_bufferRoad2[0] = (road2duration / 1000) / 10;
13         led_bufferRoad2[1] = (road2duration / 1000) % 10;
14
15
16         update7SEGRoad1(index_led);
17         update7SEGRoad2(index_led);
18         index_led++;
19         if (index_led >= MAX_LED) index_led = 0;
20         setTimer2();
21     }
22 }
23 void updateManualRedLedBuffer(){
24     if (timer2_flag == 1){
25         led_bufferRoad1[0] = status / 10;
26         led_bufferRoad1[1] = status % 10;
27         led_bufferRoad2[0] = (currentred / 1000) / 10;
28         led_bufferRoad2[1] = (currentred / 1000) % 10;
29
30         update7SEGRoad1(index_led);
31         update7SEGRoad2(index_led);
32         index_led++;
33         if (index_led >= MAX_LED) index_led = 0;
34         setTimer2();
35     }
36 }
37 void updateManualYellowLedBuffer(){
38     if (timer2_flag == 1){
39         led_bufferRoad1[0] = status / 10;
40         led_bufferRoad1[1] = status % 10;
41         led_bufferRoad2[0] = (currentyellow / 1000) / 10;
42         led_bufferRoad2[1] = (currentyellow / 1000) % 10;
43
44         update7SEGRoad1(index_led);
45         update7SEGRoad2(index_led);
46         index_led++;
```

```
47 if (index_led >= MAX_LED) index_led = 0;
48 setTimer2();
49 }
50 }
51 void updateManualGreenLedBuffer(){
52     if (timer2_flag == 1){
53         led_bufferRoad1[0] = status / 10;
54         led_bufferRoad1[1] = status % 10;
55         led_bufferRoad2[0] = (currentgreen / 1000) / 10;
56         led_bufferRoad2[1] = (currentgreen / 1000) % 10;
57
58         update7SEGRoad1(index_led);
59         update7SEGRoad2(index_led);
60         index_led++;
61         if (index_led >= MAX_LED) index_led = 0;
62         setTimer2();
63     }
64 }
65 void update7SEGRoad1 (int index){
66     switch(index){
67     case 0:
68         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_RESET);
69         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10, GPIO_PIN_SET);
70
71         display7SEGRoad1(led_bufferRoad1[0]);
72         break;
73     case 1:
74         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10, GPIO_PIN_RESET);
75         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_SET);
76
77         display7SEGRoad1(led_bufferRoad1[1]);
78         break;
79
80     default:
81         break;
82     }
83 }
84
85 void update7SEGRoad2 (int index){
86     switch(index){
87     case 0:
88         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11, GPIO_PIN_RESET);
89         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_SET);
90
91         display7SEGRoad2(led_bufferRoad2[0]);
92         break;
93     case 1:
94         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11, GPIO_PIN_SET);
95         HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_RESET);
```

```
96
97     display7SEGRoad2(led_bufferRoad2[1]);
98     break;
99 default:
100     break;
101 }
102 }
103 void display7SEGRoad1(int value)
104 {
105     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG0_Pin, SET);
106     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG1_Pin, SET);
107     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG2_Pin, SET);
108     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG3_Pin, SET);
109     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG4_Pin, SET);
110     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG5_Pin, SET);
111     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG6_Pin, SET);
112
113     switch(value){
114         case 0:
115             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG0_Pin, RESET);
116             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG1_Pin, RESET);
117             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG2_Pin, RESET);
118             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG3_Pin, RESET);
119             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG4_Pin, RESET);
120             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG5_Pin, RESET);
121             break;
122         case 1:
123             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG1_Pin, RESET);
124             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG2_Pin, RESET);
125             break;
126         case 2:
127             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG0_Pin, RESET);
128             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG1_Pin, RESET);
129             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG3_Pin, RESET);
130             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG4_Pin, RESET);
131             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG6_Pin, RESET);
132             break;
133         case 3:
134             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG0_Pin, RESET);
135             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG1_Pin, RESET);
136             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG2_Pin, RESET);
137             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG3_Pin, RESET);
138             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG6_Pin, RESET);
139             break;
140         case 4:
141             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG1_Pin, RESET);
142             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG2_Pin, RESET);
143             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG5_Pin, RESET);
144             HAL_GPIO_WritePin(SEG_GPIO_Port, SEG6_Pin, RESET);
```



```
145     break;
146 case 5:
147     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG0_Pin, RESET);
148     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG2_Pin, RESET);
149     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG3_Pin, RESET);
150     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG5_Pin, RESET);
151     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG6_Pin, RESET);
152     break;
153 case 6:
154     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG0_Pin, RESET);
155     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG2_Pin, RESET);
156     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG3_Pin, RESET);
157     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG4_Pin, RESET);
158     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG5_Pin, RESET);
159     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG6_Pin, RESET);
160     break;
161 case 7:
162     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG0_Pin, RESET);
163     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG1_Pin, RESET);
164     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG2_Pin, RESET);
165     break;
166 case 8:
167     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG0_Pin, RESET);
168     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG1_Pin, RESET);
169     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG2_Pin, RESET);
170     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG3_Pin, RESET);
171     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG4_Pin, RESET);
172     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG5_Pin, RESET);
173     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG6_Pin, RESET);
174     break;
175 case 9:
176     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG0_Pin, RESET);
177     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG1_Pin, RESET);
178     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG2_Pin, RESET);
179     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG3_Pin, RESET);
180     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG5_Pin, RESET);
181     HAL_GPIO_WritePin(SEG_GPIO_Port, SEG6_Pin, RESET);
182     break;
183 default:
184     break;
185 }
186 }
187 void display7SEGRoad2(int value)
188 {
189     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, SET);
190     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, SET);
191     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, SET);
192     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG13_Pin, SET);
193     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG14_Pin, SET);
```



```
194 HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG15_Pin, SET);
195 HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG16_Pin, SET);
196
197 switch(value){
198     case 0:
199         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, RESET);
200         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, RESET);
201         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
202         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG13_Pin, RESET);
203         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG14_Pin, RESET);
204         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG15_Pin, RESET);
205         break;
206     case 1:
207         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, RESET);
208         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
209         break;
210     case 2:
211         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, RESET);
212         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, RESET);
213         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG13_Pin, RESET);
214         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG14_Pin, RESET);
215         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG16_Pin, RESET);
216         break;
217     case 3:
218         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, RESET);
219         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, RESET);
220         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
221         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG13_Pin, RESET);
222         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG16_Pin, RESET);
223         break;
224     case 4:
225         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, RESET);
226         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
227         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG15_Pin, RESET);
228         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG16_Pin, RESET);
229         break;
230     case 5:
231         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, RESET);
232         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
233         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG13_Pin, RESET);
234         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG15_Pin, RESET);
235         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG16_Pin, RESET);
236         break;
237     case 6:
238         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, RESET);
239         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
240         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG13_Pin, RESET);
241         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG14_Pin, RESET);
242         HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG15_Pin, RESET);
```



```
243     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG16_Pin, RESET);
244     break;
245 case 7:
246     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, RESET);
247     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, RESET);
248     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
249     break;
250 case 8:
251     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, RESET);
252     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, RESET);
253     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
254     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG13_Pin, RESET);
255     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG14_Pin, RESET);
256     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG15_Pin, RESET);
257     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG16_Pin, RESET);
258     break;
259 case 9:
260     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, RESET);
261     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, RESET);
262     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
263     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG13_Pin, RESET);
264     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG15_Pin, RESET);
265     HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG16_Pin, RESET);
266     break;
267 default:
268     break;
269 }
270 }
```

This is the source code for blinking LEDs depending on the mode that selected in the exercise 6:

```
1 void fsm_manual_run(){
2     switch (status) {
3         case MAN_RED:
4             if ((timerRoad1_flag == 1) & (timerRoad2_flag ==1)){
5                 traffic_manualred();
6                 updateManualRedLedBuffer();
7                 setTimerRoad1(500);
8                 setTimerRoad2(500);
9             }
10
11             break;
12         case MAN_YELLOW:
13             if ((timerRoad1_flag == 1) & (timerRoad2_flag ==1)){
14                 traffic_manuallyellow();
15                 updateManualYellowLedBuffer();
16                 setTimerRoad1(500);
17                 setTimerRoad2(500);
18             }
19     }
```

```
20     break;
21     case MAN_GREEN:
22         if ((timerRoad1_flag == 1) & (timerRoad2_flag ==1)){
23             traffic_manualgreen();
24             updateManualGreenLedBuffer();
25             setTimerRoad1(500);
26             setTimerRoad2(500);
27         }
28
29         break;
30     default:
31         break;
32 }
33 }
34
35 void traffic_manualred(){
36     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
37     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_2);
38     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
39     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
40     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_RESET);
41     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
42 }
43 void traffic_manualgreen(){
44     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
45     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
46     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
47     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
48     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_RESET);
49     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);
50 }
51 void traffic_manuallyellow(){
52     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_6);
53     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_3);
54     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
55     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
56     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
57     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);
58 }
```

## 1.7 Exercise 7

This is the code for the exercise 7:

```
1 void fsm_manual_run(){
2     switch (status) {
3         case MAN_RED:
4             if ((timerRoad1_flag == 1) & (timerRoad2_flag ==1)){
5                 traffic_manualred();
6                 updateManualRedLedBuffer();
7                 setTimerRoad1(500);
8                 setTimerRoad2(500);
9             }
10            if (checkflag(0) == 1){
11                status = MAN_YELLOW;
12                currentred = initialduration;
13                initialduration = currentyellow;
14            }
15            if (checkflag(1) == 1){
16                if(currentred <= 99000){
17                    currentred += 1000;
18                } else {
19                    currentred = 1000;
20                }
21            }
22            if (checkflag(2) == 1){
23                status = INIT;
24            }
25            break;
26        default:
27            break;
28    }
29 }
```

## 1.8 Exercise 8

This is the code for the exercise 8:

```
1 void fsm_manual_run(){
2     switch (status) {
3         case MAN_YELLOW:
4             if ((timerRoad1_flag == 1) & (timerRoad2_flag ==1)){
5
6                 traffic_manuallyellow();
7                 updateManualYellowLedBuffer();
8                 setTimerRoad1(500);
9                 setTimerRoad2(500);
10            }
11            if (checkflag(0) == 1){
12                status = MAN_GREEN;
13                currentyellow = initialduration;
14                initialduration = currenttgreen;
15            }
16            if (checkflag(1) == 1){
17                if(currentyellow <= 99000 ){
18                    if (currentred > currentyellow + currenttgreen){
19                        currentyellow += 1000;
20                    } else {
21                        currentyellow = currentred - currenttgreen;
22                    }
23                } else {
24                    currentyellow = 1000;
25                }
26            }
27            if (checkflag(2) == 1){
28                status = INIT;
29            }
30            break;
31        default:
32            break;
33    }
34 }
```

## 1.9 Exercise 9

This is the source code for the exercise 9:

```
1 void fsm_manual_run(){
2     switch (status) {
3         case MAN_GREEN:
4             if ((timerRoad1_flag == 1) & (timerRoad2_flag ==1)){
5                 traffic_manualgreen();
6                 updateManualGreenLedBuffer();
7                 setTimerRoad1(500);
8                 setTimerRoad2(500);
9             }
10            if (checkflag(0) == 1){
11                status = INIT;
12                currentgreen = initialduration;
13                initialduration = 0;
14            }
15            if (checkflag(1) == 1){
16                if (currentgreen <= 99000){
17                    if (currentred > (currentgreen + currentyellow)){
18                        currentgreen += 1000;
19                    } else {
20                        currentgreen = currentred - currentyellow;
21                    }
22                } else {
23                    currentgreen = 1000;
24                }
25            }
26        }
27        if (checkflag(2) == 1){
28            status = INIT;
29        }
30        break;
31    default:
32        break;
33    }
34 }
```

## 1.10 Exercise 10

In Lab 3, the project running with 2 finite state machines, fsm\_automatic and fsm\_manual.

- In fsm\_automatic (Normal mode), the traffic light in a cross road is performed by 12 LEDS including 4 Red LEDS, 4 Amber LEDS, 4 green LEDS and 7 states (including Start state). When starting to run schematic, the traffic light is displayed by given duration for each road.
- While running fsm\_automatic, if the first button is pressed, the system will immediately switch to the red manual state in fsm\_manual.
- In fsm\_manual, there are 3 states including MAN\_RED, MAN\_YELLOW, MAN\_GREEN for editing and setting the duration of each LED color. Button 2 is used for increasing duration of a certain LED by 1 second; with a provided condition that the duration of the red light must be greater than or equal to the total duration of the yellow light and the green light. Button 3 is used for saving and setting the value after button 2 is pressed and return to the Start state of fsm\_automatic.
- If button 1 is long-pressed, after every one second, it will jump to the next state.
- If button 2 is long-pressed, after every one second, it will increasing the duration of a certain LED by 1 second.
- If button 3 is long-pressed, after the first one second, it will set the duration's value of a certain LED and return to the Start state of fsm\_automatic.



## References