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

1 Exercise

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

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

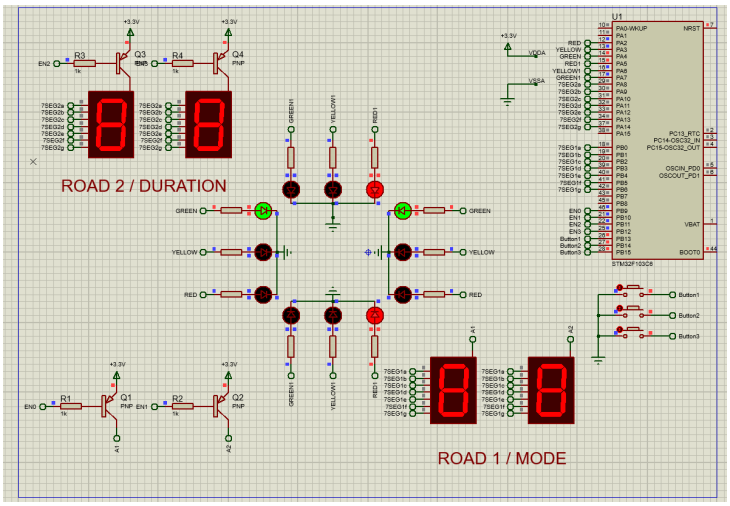

Figure 1: *The schematic for Lab 3.*

1.1 Exercise 1

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

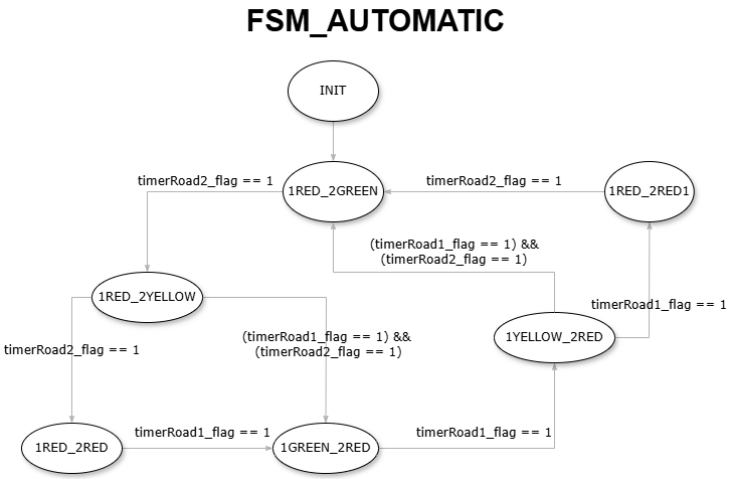

Figure 2: *The Finite State Machine for normal mode.*

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

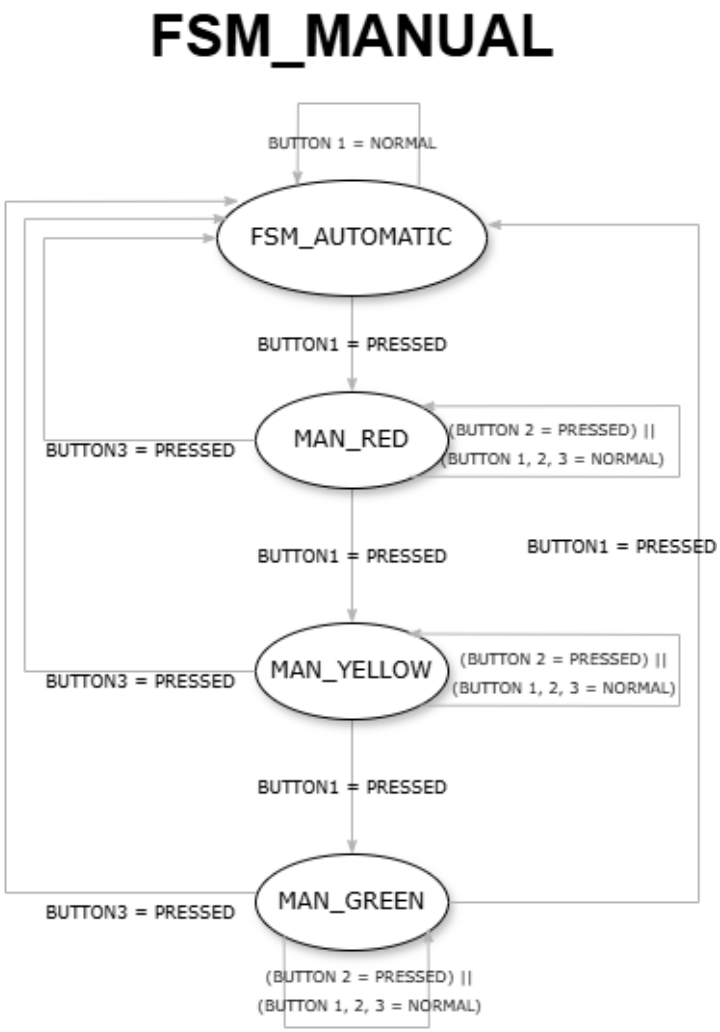

Figure 3: *The Finite State Machine for 3 buttons application.*

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.

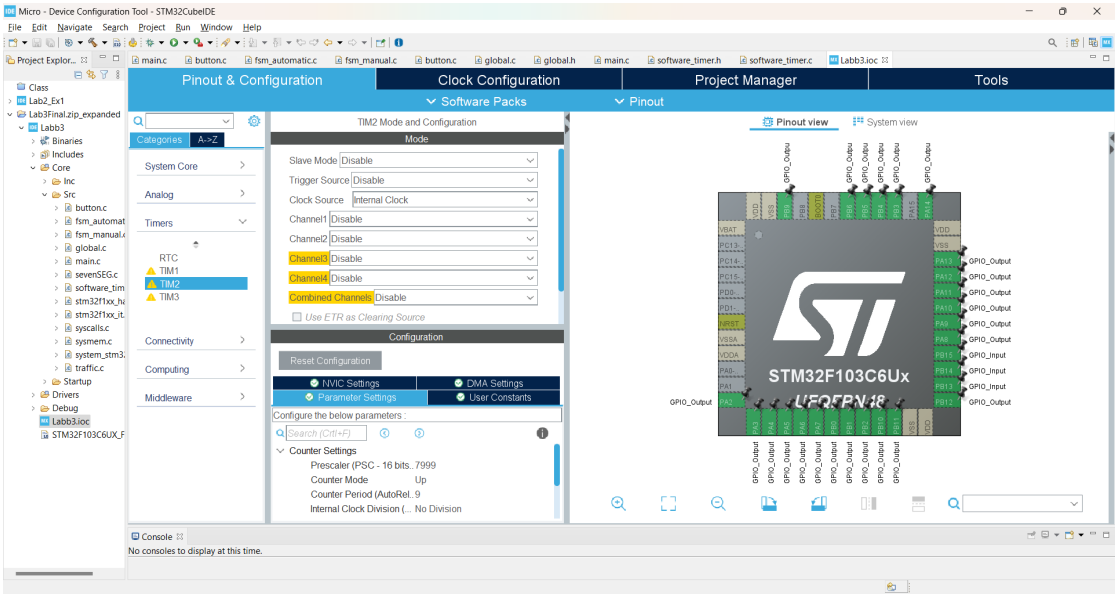

Figure 4: *Lab 3 project.*

1.4 Exercise 4

1.4.1 Report 1

This is the source code for the exercise 4:

```
int timerRoad1_counter = 0;
int timerRoad2_counter = 0;
int timer2_counter = 0;


int timerRoad1_flag = 0;
int timerRoad2_flag = 0;
int timer2_flag = 0;




void setTimerRoad1(int duration){
  timerRoad1_counter = duration/timercycle;
  timerRoad1_flag = 0;
}

void setTimerRoad2(int duration){
  timerRoad2_counter = duration/timercycle;
  timerRoad2_flag = 0;
}

void setTimer2(){
  timer2_counter = (1000 / MAX_LED) / timercycle;
  timer2_flag = 0;
}

void timerRun(){
  if (timerRoad1_counter > 0){
    timerRoad1_counter--;
    if(timerRoad1_counter <= 0){
      timerRoad1_flag = 1;
    }
  }
  if (timerRoad2_counter > 0){
    timerRoad2_counter--;
    if(timerRoad2_counter <= 0){
      timerRoad2_flag = 1;
    }
  }
  if (timer2_counter > 0){
    timer2_counter--;
    road1duration-=10;
    road2duration-=10;
    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:

```
GPIO_TypeDef* portB[number_of_buttons] = {GPIOB, GPIOB, GPIOB};
uint16_t buttons[number_of_buttons] = {GPIO_PIN_13, GPIO_PIN_14, GPIO_PIN_15};

int button_flag[number_of_buttons] = {0, 0, 0};

int keyReg0[number_of_buttons];
int keyReg1[number_of_buttons];
int keyReg2[number_of_buttons];
int keyReg3[number_of_buttons];


int button_pressed_counter[number_of_buttons] = {200, 200, 200};


int checkflag(int index){
  if(button_flag[index] == 1){
    button_flag[index] = 0;
    return 1;
  }
  return 0;
}


void getKeyInput(void) {
  for(int i = 0; i < number_of_buttons; i++){
    keyReg0[i] = keyReg1[i];
    keyReg1[i] = keyReg2[i];
    keyReg2[i] = HAL_GPIO_ReadPin(portB[i], buttons[i]);

    if ((keyReg0[i] == keyReg1[i])&&(keyReg1[i] == keyReg2[i])) {
      if (keyReg3[i] != keyReg2[i]){
        keyReg3[i] = keyReg2[i];
        if (keyReg2[i] <= pressed_button){
          button_pressed_counter[i] = 200;
                button_flag[i] = 1;
           }
        } else {
           button_pressed_counter[i]--;
             if (button_pressed_counter[i] <= 0){
               keyReg3[i] = normal_button;
             }
        }
    }
  }
}
```

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

```
void checkManual(){
  updateLedBuffer();
  if (checkflag(0) == 1){
    traffic_turnoffall();
    status = MAN_RED;
    setTimerRoad1(100);
    setTimerRoad2(100);
  }
}
```

1.6 Exercise 6

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

```
const int MAX_LED = 2;
int index_led = 0;
int led_bufferRoad1 [2] = {0, 0};
int led_bufferRoad2 [2] = {0, 0};



void updateLedBuffer(){
  if (timer2_flag == 1){
  led_bufferRoad1[0] = (road1duration / 1000) / 10;
  led_bufferRoad1[1] = (road1duration / 1000) % 10;
  led_bufferRoad2[0] = (road2duration / 1000) / 10;
  led_bufferRoad2[1] = (road2duration / 1000) % 10;


  update7SEGRoad1(index_led);
  update7SEGRoad2(index_led);
  index_led++;
  if (index_led >= MAX_LED) index_led = 0;
  setTimer2();
  }
}
void updateManualRedLedBuffer(){
  if (timer2_flag == 1){
  led_bufferRoad1[0] = status / 10;
  led_bufferRoad1[1] = status % 10;
  led_bufferRoad2[0] = (currentred / 1000) / 10;
  led_bufferRoad2[1] = (currentred / 1000) % 10;

  update7SEGRoad1(index_led);
  update7SEGRoad2(index_led);
  index_led++;
  if (index_led >= MAX_LED) index_led = 0;
  setTimer2();
  }
}
void updateManualYellowLedBuffer(){
  if (timer2_flag == 1){
  led_bufferRoad1[0] = status / 10;
  led_bufferRoad1[1] = status % 10;
  led_bufferRoad2[0] = (currentyellow / 1000) / 10;
  led_bufferRoad2[1] = (currentyellow / 1000) % 10;

  update7SEGRoad1(index_led);
  update7SEGRoad2(index_led);
  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);
```

```
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG16_Pin, RESET);
      break;
    case 7:
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
      break;
    case 8:
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG13_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG14_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG15_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG16_Pin, RESET);
      break;
    case 9:
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG10_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG11_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG12_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG13_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG15_Pin, RESET);
      HAL_GPIO_WritePin(SEG1_GPIO_Port, SEG16_Pin, RESET);
      break;
    default:
      break;
  }
}
```

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

```
void fsm_manual_run(){
  switch (status) {
    case MAN_RED:
      if ((timerRoad1_flag == 1) & (timerRoad2_flag ==1)){
        traffic_manualred();
        updateManualRedLedBuffer();
        setTimerRoad1(500);
        setTimerRoad2(500);
      }

      break;
    case MAN_YELLOW:
      if ((timerRoad1_flag == 1) & (timerRoad2_flag ==1)){
        traffic_manualyellow();
        updateManualYellowLedBuffer();
        setTimerRoad1(500);
        setTimerRoad2(500);
      }

```

```
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_manualyellow(){
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:

```
void fsm_manual_run(){
  switch (status) {
    case MAN_RED:
      if ((timerRoad1_flag == 1) & (timerRoad2_flag ==1)){
        traffic_manualred();
        updateManualRedLedBuffer();
        setTimerRoad1(500);
        setTimerRoad2(500);
      }
      if (checkflag(0) == 1){
        status = MAN_YELLOW;
        currentred = initialduration;
        initialduration = currentyellow;
      }
      if (checkflag(1) == 1){
        if(currentred <= 99000){
          currentred += 1000;
        } else {
          currentred = 1000;
        }
      }
      if (checkflag(2) == 1){
        status = INIT;
      }
      break;
    default:
      break;
  }
}
```

1.8 Exercise 8

This is the code for the exercise 8:

```
void fsm_manual_run(){
  switch (status) {
    case MAN_YELLOW:
      if ((timerRoad1_flag == 1) & (timerRoad2_flag ==1)){

        traffic_manualyellow();
        updateManualYellowLedBuffer();
        setTimerRoad1(500);
        setTimerRoad2(500);
      }
      if (checkflag(0) == 1){
        status = MAN_GREEN;
        currentyellow = initialduration;
        initialduration = currentgreen;
      }
      if (checkflag(1) == 1){
        if(currentyellow <= 99000 ){
          if (currentred > currentyellow + currentgreen){
            currentyellow += 1000;
          } else {
            currentyellow = currentred - currentgreen;
          }
        } else {
          currentyellow = 1000;
        }
      }
      if (checkflag(2) == 1){
        status = INIT;
      }
      break;
    default:
      break;
  }
}
```

1.9 Exercise 9

This is the source code for the exercise 9:

```
void fsm_manual_run(){
  switch (status) {
    case MAN_GREEN:
      if ((timerRoad1_flag == 1) & (timerRoad2_flag ==1)){
        traffic_manualgreen();
        updateManualGreenLedBuffer();
        setTimerRoad1(500);
        setTimerRoad2(500);
      }
      if (checkflag(0) == 1){
        status = INIT;
        currentgreen = initialduration;
        initialduration = 0;
      }
      if (checkflag(1) == 1){
        if (currentgreen <= 99000){
          if (currentred > (currentgreen + currentyellow)){
            currentgreen += 1000;
          } else {
            currentgreen = currentred - currentyellow;
          }
        } else {
          currentgreen = 1000;
        }

      }
      if (checkflag(2) == 1){
        status = INIT;
      }
      break;
    default:
      break;
  }
}
```

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