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

CC02 – CO3010 - MICROPROCESSOR - MICROCONTROLLER (LAB)

Lab 3 REPORT

Supervisors: Nguyen Thien An
Students: Nguyen Dang Duy 2252116

Ho Chi Minh City, November 11, 2024

Contents

Information about the Project:

GitHub repository to the project: https://github.com/l1ttled1no/mcu-mpu

YouTube link to the demo of the lab simulator (in case the simulator cannot run): https://youtu.be/5rKPf6nUgBg

1 Schematic

This is the Proteus schematic. The schematic can be downloaded from the upper GitHub repository:

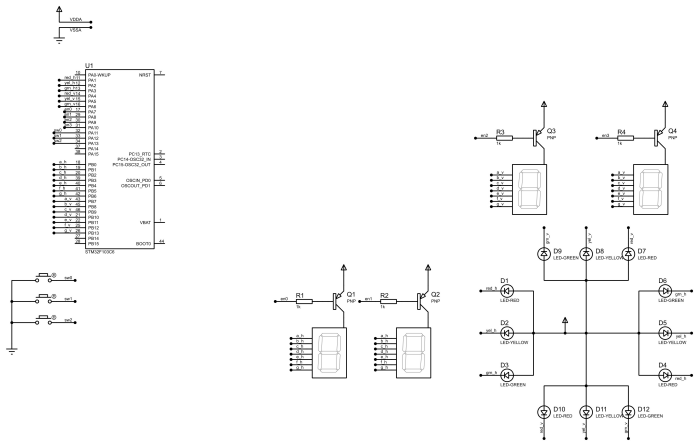

Figure 1: *Proteus Schematic.*

2 File Layout

- button.c: reading the buttons.

- timer.c and timer.h: define the timer interrrupt.

- led.c and led.h: modifying the LED lights

- segment.c and segment.h: Modifying 4 7-segment LEDs to display counters and modes, depends on each mode.

- global.h: define global variables for the FSM.

- fsm.c: contain the FSM.

- main.h and main.c: contain main program.

3 Finite State Machine

This is the FSM for the Traffic Light system:

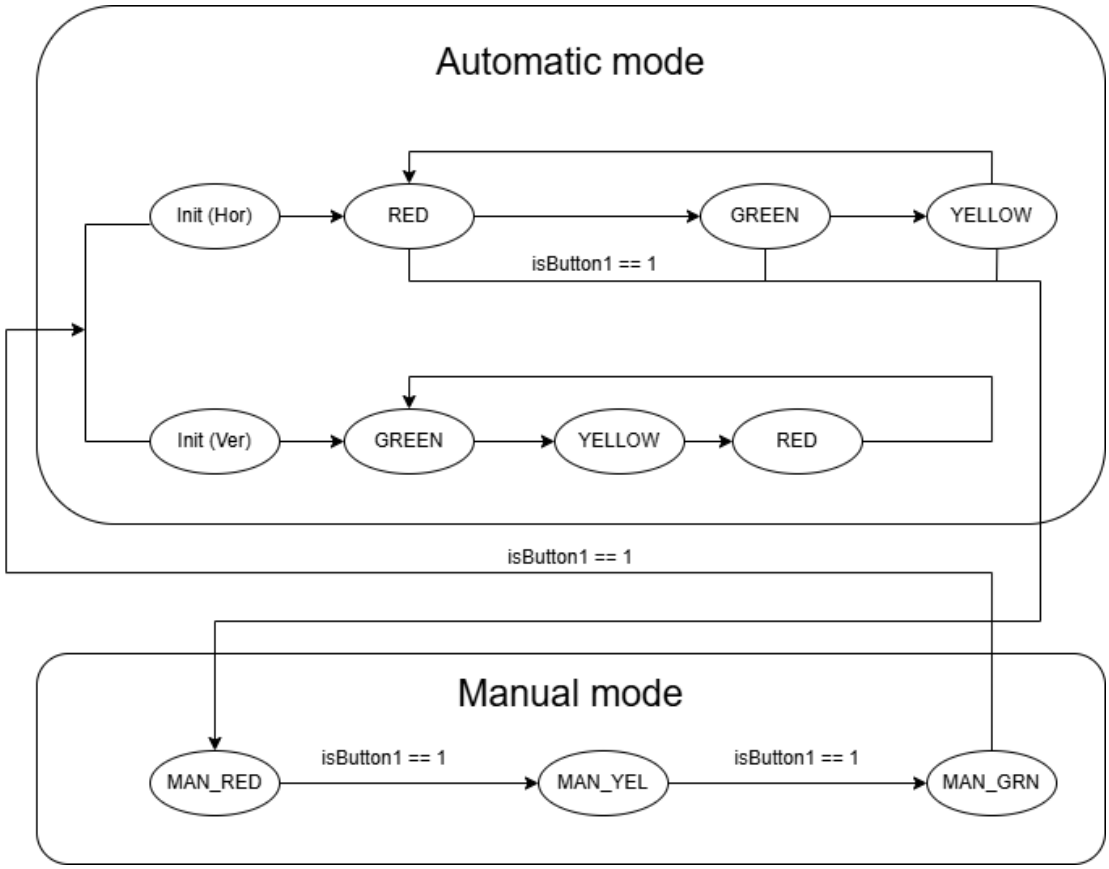

Figure 2: *Finite State Machine.*

4 Functions

4.1 button.c & button.h

button.c and button.h is used to implement the buttons. There are 3 buttons:

- Button 1 (`sw0`): use to modify modes.

- Button 2 (`sw1`): use to modify values for each LEDs (only available in `MAN_RED`, `MAN_YEL` and `MAN_GRN` state of `void fsm_man()`).

- Button 3 (`sw0`): To confirm the values of each LEDs (only available in `MAN_RED`, `MAN_YEL` and `MAN_GRN` state of `void fsm_man()`)

Implementing for button.h:

```
#ifndef INC_BUTTON_H_
#define INC_BUTTON_H_

#include "global.h"

#define NO_BUTTON 3

#define PRESSED 0
#define RELEASED 1

int isButtonNoPressed(int);
void buttonRead();


#endif /* INC_BUTTON_H_ */
```

Implementing for button.c:

```
#include "button.h"

int buttonFlag[NO_BUTTON] = {0, 0, 0};
int timeout[NO_BUTTON] = {100, 100, 100};

int KeyReg0[NO_BUTTON] = {RELEASED, RELEASED, RELEASED};
int KeyReg1[NO_BUTTON] = {RELEASED, RELEASED, RELEASED};
int KeyReg2[NO_BUTTON] = {RELEASED, RELEASED, RELEASED};
int KeyReg3[NO_BUTTON] = {RELEASED, RELEASED, RELEASED};

int isButtonNoPressed(int no){
  if (buttonFlag[no] == 1){
    buttonFlag[no] = 0;
    return 1;
  }
  return 0;
}

```

```
void buttonRead(){
  for (int i = 0; i < NO_BUTTON; i++){
    KeyReg2[i] = KeyReg1[i];
    KeyReg1[i] = KeyReg0[i];
    KeyReg0[i] = HAL_GPIO_ReadPin(sw0_GPIO_Port, sw0_Pin << i);
    if ((KeyReg1[i] == KeyReg0[i]) && (KeyReg1[i] == KeyReg2[i])){
      if (KeyReg2[i] != KeyReg3[i]){ //reg2 != reg3
        KeyReg3[i] = KeyReg2[i];
        if (KeyReg0[i] == PRESSED){
          timeout[i] = 100;
          buttonFlag[i] = 1;
        }
      }
      else { //reg2 = reg3
        timeout[i]--;
        if (timeout[i] == 0){
          timeout[i] = 10;
          if (KeyReg3[i] == PRESSED){
            buttonFlag[i] = 1;
          }
        }
      }
    }
  }
}
```

4.2 timer.h & timer.c

timer.h & timer.c are used to implementing the timer interrupt instead of using `HAL_Delay(uint32_t)`.

Implementing timer.h:

```
#ifndef INC_TIMER_H_
#define INC_TIMER_H_
#include "global.h"
#define CYCLE   10
extern int  flag1,
            flag2,
            flag3,
            flag4,
            flag5;

void set1(int);
void set2(int);
void set3(int);
void set4(int);
void set5(int);
void resetTimer(int);

```

```
void timerRun();

#endif /* INC_TIMER_H_ */
```

Implementing timer.c:

```
#include "timer.h"

int timer1 = 0,
    timer2 = 0,
    timer3 = 0,
    timer4 = 0,
    timer5 = 0;

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

void set1(int timer){
    timer1 = timer / CYCLE;
    flag1 = 0;
}

void set2(int timer){
    timer2 = timer / CYCLE;
    flag2 = 0;
}

void set3(int timer){
    timer3 = timer / CYCLE;
    flag3 = 0;
}

void set4(int timer){
    timer4 = timer / CYCLE;
    flag4 = 0;
}
void set5(int timer){
    timer5 = timer / CYCLE;
    flag5 = 0;
}

void resetTimer(int timer){
  switch(timer){
  case 1:
    timer1 = 0;
    flag1 = 0;
    break;
```

```
  case 2:
    timer2 = 0;
    flag2 = 0;
    break;
  case 3:
    timer3 = 0;
    flag3 = 0;
    break;
  case 4:
    timer4 = 0;
    flag4 = 0;
        break;
    case 5:
        timer5 = 0;
        flag5 = 0;
    break;
  default:
    timer1 = 0;
    timer2 = 0;
    timer3 = 0;
    timer4 = 0;
        timer5 = 0;
    flag1 = 0;
    flag2 = 0;
    flag3 = 0;
    flag4 = 0;
        flag5 = 0;
    break;
  }
}

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;
    }
```

```
94      if (timer5 == 0){
95          flag5 = 1;
96      }
97  }
```

4.3 led.c & led.h

led.c & led.h are used to implement LEDs. There are some functions and definitions:

- `setLedH(int color)`: setting horizontal LEDs color. There are 5 modes: `RED`, `YELLOW`, `GREEN`, `ALL` (all lights are turning on) and default (all lights are turning off).

- `setLedV(int color)`: setting vertical LEDs color. There are 5 modes: `RED`, `YELLOW`, `GREEN`, `ALL` (all lights are turning on) and default (all lights are turning off).

Implementing for led.h:

```
1  #ifndef INC_LED_H_
2  #define INC_LED_H_
3  #include "global.h"
4
5  #define LED_ON    0
6  #define LED_OFF   1
7
8  extern int horState;
9  extern int verState;
10
11 void setLedH (int);
12 void setLedV (int);
13
14 #endif /* INC_LED_H_ */
```

Implementing for led.c:

```
1  #include "led.h"
2  int horState = NONE;
3  int verState = NONE;
4  void setLedH(int color){
5    switch(color){
6    case RED:
7      HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin, LED_ON);
8      HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin, LED_OFF);
9      HAL_GPIO_WritePin(grn_h_GPIO_Port, grn_h_Pin, LED_OFF);
10     horState = RED;
11     break;
12   case YEL:
13     HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin, LED_OFF);
14     HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin, LED_ON);
15     HAL_GPIO_WritePin(grn_h_GPIO_Port, grn_h_Pin, LED_OFF);
16     horState = YEL;
17     break;
```

```
18    case GRN:
19      HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin, LED_OFF);
20      HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin, LED_OFF);
21      HAL_GPIO_WritePin(grn_h_GPIO_Port, grn_h_Pin, LED_ON);
22      horState = GRN;
23      break;
24    case ALL:
25      HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin, LED_ON);
26      HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin, LED_ON);
27      HAL_GPIO_WritePin(grn_h_GPIO_Port, grn_h_Pin, LED_ON);
28      horState = ALL;
29      break;
30    default:
31      HAL_GPIO_WritePin(red_h_GPIO_Port, red_h_Pin, LED_OFF);
32      HAL_GPIO_WritePin(yel_h_GPIO_Port, yel_h_Pin, LED_OFF);
33      HAL_GPIO_WritePin(grn_h_GPIO_Port, grn_h_Pin, LED_OFF);
34      horState = NONE;
35      break;
36    }
37  }
38
39  void setLedV(int color){
40    switch(color){
41    case RED:
42      HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin, LED_ON);
43      HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin, LED_OFF);
44      HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin, LED_OFF);
45      verState = RED;
46      break;
47    case YEL:
48      HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin, LED_OFF);
49      HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin, LED_ON);
50      HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin, LED_OFF);
51      verState = YEL;
52      break;
53    case GRN:
54      HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin, LED_OFF);
55      HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin, LED_OFF);
56      HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin, LED_ON);
57      verState = GRN;
58      break;
59    case ALL:
60      HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin, LED_ON);
61      HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin, LED_ON);
62      HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin, LED_ON);
63      verState = ALL;
64      break;
65    default:
66      HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin, LED_OFF);
```

```
67      HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin, LED_OFF);
68      HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin, LED_OFF);
69      verState = NONE;
70      break;
71    }
72  }
```

4.4 segment.c & segment.h

segment.c & segment.h are used to implements 4 7-segment LEDs. We use scanning LEDs method inorder to control 4 of them. There are some functions and definitions:

- `segment_buffer[4]`: segment buffer for each LEDs, with:
  - index `0` for `en0` LED.
  - index `1` for `en1` LED.
  - index `2` for `en2` LED.
  - index `3` for `en3` LED.

`-GPIO_PinState pinArr[11][7]`: using the shifting method to utilize the code, make it shorter.

`-void set7SegH(int)`: set the 7-segment LEDs on horizontal path.

`-void set7SegH(int)`: set the 7-segment LEDs on vertical path.

`-void scan7Seg(int)`: scan the first LEDs on horizontal/vertical and second LEDs on horizontal/vertical repeatedly.

`void updateSegment(int, int, int, int)`: set the number to the segment to the right index respectively.

`void updateSegment2Digits(int, int)`: set 2-digit numbers to the segment, with the first 2 LEDs is for the first integer, the last 2 LEDs is for the second integer.

Implementing for segment.h:

```
1  #ifndef INC_SEGMENT_H_
2  #define INC_SEGMENT_H_
3
4  #include "global.h"
5  extern int segment_buffer[4];
6
7  void set7SegH(int);
8  void set7SegV(int);
9  void scan7Seg(int);
10
11 void updateSegment(int, int, int, int);
12 void updateSegment2Digits(int, int);
13
14 #endif /* INC_SEGMENT_H_ */
```

Implement for segment.c:

```c
#include "segment.h"

int segment_buffer[4] = {0};

GPIO_PinState pinArr[11][7] = {
      {0, 0, 0, 0, 0, 0, 1}, //0
      {1, 0, 0, 1, 1, 1, 1}, //1
      {0, 0, 1, 0, 0, 1, 0}, //2
      {0, 0, 0, 0, 1, 1, 0}, //3
      {1, 0, 0, 1, 1, 0, 0}, //4
      {0, 1, 0, 0, 1, 0, 0}, //5
      {0, 1, 0, 0, 0, 0, 0}, //6
      {0, 0, 0, 1, 1, 1, 1}, //7
      {0, 0, 0, 0, 0, 0, 0}, //8
      {0, 0, 0, 0, 1, 0, 0}, //9
    {1, 1, 1, 1, 1, 1, 1}  //ALL LED TURN OFF
};

void set7SegH(int num){
  if (num >= 0 && num <= 9){
    for (int state = 0; state < 7; state++){
      HAL_GPIO_WritePin(a_h_GPIO_Port, a_h_Pin << state, pinArr[num][state]);
    }
  }
  else{
    for(int state = 0; state < 7; state++){ // Turn off
      HAL_GPIO_WritePin(a_h_GPIO_Port, a_h_Pin << state, pinArr[10][state]);
    }
  }
}

void set7SegV(int num){
  if (num >= 0 && num <= 9){
    for (int state = 0; state < 7; state++){
      HAL_GPIO_WritePin(a_v_GPIO_Port, a_v_Pin << state, pinArr[num][state]);
    }
  }
  else {
    for(int state = 0; state < 7; state++){ // Turn off
      HAL_GPIO_WritePin(a_v_GPIO_Port, a_v_Pin << state, pinArr[10][state]);
    }
  }
}


void scan7Seg (int state){
  state = state % 2;
  switch (state){
  case 0:
```

```
    HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 0);
    HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 1);
    HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 0);
    HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 1);
    set7SegH(segment_buffer[0]);
    set7SegV(segment_buffer[2]);
    break;
  case 1:
    HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, 1);
    HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, 0);
    HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, 1);
    HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, 0);
    set7SegH(segment_buffer[1]);
    set7SegV(segment_buffer[3]);
    break;
  default:
    break;
  }
}

void updateSegment(int a, int b, int c, int d){
  segment_buffer[0] = a % 10;
  segment_buffer[1] = b % 10;
  segment_buffer[2] = c % 10;
  segment_buffer[3] = d % 10;
}

void updateSegment2Digits(int firstNum, int secNum){
  segment_buffer[0] = firstNum / 10;
  segment_buffer[1] = firstNum % 10;
  segment_buffer[2] = secNum / 10;
  segment_buffer[3] = secNum % 10;
}
```

4.5 global.h

global.h is the library to define the global variable, and define the FSM we are using: - `segmentUpdateAuto()`: updating the counter into the `segment_buffer` array while being in auto mode.

- `void fsm_run()`: run all FSM, put in the `while(1)` function in main.c.

- `void fsm_auto_hor()`: FSM for the horizontal traffic way.

- `void fsm_auto_ver()`: FSM for the vertical traffic way.

- `void fsm_man()`: modifying each LEDs duration, activate by pressing `sw0` button.

- `void button0Signal()`: checking if `sw0` button is pressed or not. Only using in `void fsm_auto_hor()`.

Implement for global.h:

```
#ifndef INC_GLOBAL_H_
#define INC_GLOBAL_H_

#include "button.h"
#include "main.h"
#include "timer.h"
#include "segment.h"
#include "led.h"

#define RED   11
#define YEL   22
#define GRN   33
#define ALL   44
#define NONE  0

#define INIT 1

#define MAN_RED 10
#define MAN_YEL 20
#define MAN_GRN 30

#define IDLE  -1

void segmentUpdateAuto();

void button0Signal();

void fsm_run();
void fsm_auto_hor();
void fsm_auto_ver();
void fsm_man();

#endif /* INC_GLOBAL_H_ */
```

4.6 fsm.c

fsm.c is used to implementing the FSM in global.h:

```
#include "global.h"

int autoState_H = INIT;
int autoState_V = INIT;
int manState = IDLE;

int redDur = 5;
int yelDur = 2;
int grnDur = 3;

```

```
int tempRed = 1;
int tempYel = 1;
int tempGrn = 1;

int horCount = 0;
int verCount = 0;
int scan = 0;
void segmentUpdateAuto(){
  updateSegment2Digits(horCount, verCount);
}

void fsm_run(){
  fsm_auto_hor();
  fsm_auto_ver();
  fsm_man();
}

void button0Signal(){
  if (isButtonNoPressed(0) == 1){
    resetTimer(-1);
    horCount = 0, verCount = 0;
    updateSegment(IDLE, IDLE, IDLE, IDLE);
    scan = 0;
    setLedH(IDLE);
    setLedV(IDLE);
    autoState_H = IDLE;
    autoState_V = IDLE;
    manState = MAN_RED;
    set1(100);
    set3(100);
    return;
  }
}

void fsm_auto_hor(){
  switch(autoState_H){
  case INIT:
    set1(redDur * 1000); //timer1 is for the hor_state
    set2(1000); //counter
    set3(100);  //scanning LED, does not need at fsm_auto_ver
    horCount = redDur - 1;
    autoState_H = RED;
    break;
  case RED:
    setLedH(RED);
    if (flag2 == 1){
      horCount--;
      if (verCount < 0) verCount = 9;
      set2(1000);
```

```
60      }
61      if (flag1 == 1){ //switch state
62        set1(grnDur * 1000);
63        horCount = grnDur - 1;
64        autoState_H = GRN;
65      }
66      if (flag3 == 1){
67        segmentUpdateAuto();
68        scan = (scan == 1) ? 0 : 1;
69        scan7Seg(scan);
70        set3(100);
71      }
72      button0Signal();
73      break;
74    case GRN:
75      setLedH(GRN);
76      if (flag2 == 1){
77        horCount--;
78        if (verCount < 0) verCount = 9;
79        set2(1000);
80      }
81
82      if (flag1 == 1){
83        set1(yelDur * 1000);
84        horCount = yelDur - 1;
85        autoState_H = YEL;
86      }
87      if (flag3 == 1){
88        segmentUpdateAuto();
89        scan = (scan == 1) ? 0 : 1;
90        scan7Seg(scan);
91        set3(100);
92      }
93      button0Signal();
94      break;
95    case YEL:
96      setLedH(YEL);
97      if (flag2 == 1){
98        horCount--;
99        if (verCount < 0) verCount = 9;//sync with vertical
100       set2(1000);
101     }
102
103     if (flag1 == 1){
104       set1(redDur * 1000);
105       horCount = redDur - 1;
106       autoState_H = RED;
107     }
108     if (flag3 == 1){
```

```
109       segmentUpdateAuto();
110       scan = (scan == 1) ? 0 : 1;
111       scan7Seg(scan);
112       set3(100);
113     }
114     button0Signal();
115     break;
116   default: //IDLE
117     break;
118   }
119 }
120 
121 void fsm_auto_ver(){
122     switch(autoState_V){
123     case INIT:
124         set4(grnDur * 1000); //timer4 is for the ver_state
125         set5(1000);
126         verCount = grnDur - 1;
127         autoState_V = GRN;
128         break;
129     case GRN:
130       setLedV(GRN);
131       if (flag5 == 1){
132         verCount--;
133         set5(1000);
134       }
135       if (flag4 == 1){
136         set4(yelDur * 1000);
137         verCount = yelDur - 1;
138         autoState_V = YEL;
139       }
140       break;
141     case YEL:
142       setLedV(YEL);
143       if (flag5 == 1){
144         verCount--;
145         set5(1000);
146       }
147       if (flag4 == 1){
148         set4(redDur * 1000);
149         verCount = redDur - 1;
150         autoState_V = RED;
151       }
152       break;
153     case RED:
154       setLedV(RED);
155       if (flag5 == 1){
156         verCount--;
157         set5(1000);
```

```
      }
      if (flag4 == 1){
        set4(grnDur * 1000);
        verCount = grnDur - 1;
        autoState_V = GRN;
      }
      break;
    default: break;
    }
}

void fsm_man(){
  switch(manState){
  case MAN_RED:
    updateSegment2Digits(tempRed, 22);
    if (isButtonNoPressed(0) == 1){
      //switch to man yellow and discard the change
      tempRed = 1;
      manState = MAN_YEL;
      setLedV(IDLE);
      setLedH(IDLE);
      set1(100);
      set3(100);
    }

    if (isButtonNoPressed(1) == 1){
      //add the tempRed 1 unit, if tempred > 99 -> 1
      tempRed = (tempRed == 99) ? 1 : tempRed + 1;

    }
    if (isButtonNoPressed(2) == 1){
      //save the config of the light
      redDur = tempRed;
    }
    if (flag1 == 1){ //flickering the LEDs light
      HAL_GPIO_TogglePin(red_h_GPIO_Port, red_h_Pin);
      HAL_GPIO_TogglePin(red_v_GPIO_Port, red_v_Pin);
      set1(500);
    }
    if (flag3 == 1){
      scan = (scan + 1)%2;
      scan7Seg(scan);
      //set the flag again; then displaying modes
      set3(100);
    }
    break;
  case MAN_YEL:
    updateSegment2Digits(tempYel, 33);
    if (isButtonNoPressed(0) == 1){
```

```
207       //switch to man grn and discard the change
208       tempYel = 1;
209       manState = MAN_GRN;
210       setLedV(IDLE);
211       setLedH(IDLE);
212       set1(100);
213       set3(100);
214     }
215 
216     if (isButtonNoPressed(1) == 1){
217       //add the tempRed 1 unit, if tempred > 99 -> 1
218       tempYel = (tempYel == 99) ? 1 : tempYel + 1;
219 
220     }
221     if (isButtonNoPressed(2) == 1){
222       //save the config of the light
223       yelDur = tempYel;
224     }
225     if (flag1 == 1){ //flickering the LEDs light
226       HAL_GPIO_TogglePin(yel_h_GPIO_Port, yel_h_Pin);
227       HAL_GPIO_TogglePin(yel_v_GPIO_Port, yel_v_Pin);
228       set1(500);
229     }
230     if (flag3 == 1){
231       scan = (scan + 1)%2;
232       scan7Seg(scan);
233       //set the flag again; then displaying modes
234       set3(100);
235     }
236     break;
237   case MAN_GRN:
238     updateSegment2Digits(tempGrn, 44);
239     if (isButtonNoPressed(0) == 1){
240 
241       //Checking the value of all the LED; if it is logical, it can be used; else
      modify it.
242       if (yelDur > grnDur){
243         grnDur += yelDur;
244       }
245       if (redDur < grnDur + yelDur){
246         redDur = grnDur + yelDur;
247       }
248       if (grnDur >= redDur + yelDur){
249         grnDur = redDur - yelDur;
250       }
251       setLedH(ALL);
252       setLedV(ALL);
253       set7SegH(8);
254       set7SegV(8);
```

```
      for (int i = 0; i < 4; i++){
        HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin << 1, 0);
      }
      HAL_Delay(3000);
      setLedV(IDLE);
      setLedH(IDLE);
      resetTimer(NONE); //reset all timer;
      manState = IDLE;
      autoState_H = INIT;
      autoState_V = INIT;
      return;
    }

    if (isButtonNoPressed(1) == 1){
      //add the tempRed 1 unit, if tempred > 99 -> 1
      tempGrn = (tempGrn == 99) ? 1 : tempGrn + 1;
    }
    if (isButtonNoPressed(2) == 1){
      //save the config of the light
      grnDur = tempGrn;
    }
    if (flag1 == 1){ //flickering the LEDs light
      HAL_GPIO_TogglePin(grn_h_GPIO_Port, grn_h_Pin);
      HAL_GPIO_TogglePin(grn_v_GPIO_Port, grn_v_Pin);
      set1(500);
    }
    if (flag3 == 1){
      scan = (scan + 1)%2;
      scan7Seg(scan);
      //set the flag again; then displaying modes
      set3(100);
    }
    break;
  default: break;
  }
}
```

4.7 main.c

Implementing the main.c and timer interrupt:

```

#include "main.h"
#include "global.h"

TIM_HandleTypeDef htim2;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);
```

```
9
10 int main(void)
11 {
12   HAL_Init();
13   SystemClock_Config();
14   MX_GPIO_Init();
15   MX_TIM2_Init();
16   HAL_TIM_Base_Start_IT(&htim2);
17   while (1)
18   {
19     fsm_run();
20   }
21 }
22
23 void SystemClock_Config(void)
24 {
25   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
26   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
27   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
28   RCC_OscInitStruct.HSIState = RCC_HSI_ON;
29   RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
30   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
31   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
32   {
33     Error_Handler();
34   }
35
36   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
37                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
38   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
39   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
40   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
41   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
42
43   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
44   {
45     Error_Handler();
46   }
47 }
48
49 static void MX_TIM2_Init(void)
50 {
51   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
52   TIM_MasterConfigTypeDef sMasterConfig = {0};
53
54   htim2.Instance = TIM2;
55   htim2.Init.Prescaler = 7999;
56   htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
57   htim2.Init.Period = 9;
```

```
58    htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
59    htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
60    if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
61    {
62      Error_Handler();
63    }
64    sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
65    if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
66    {
67      Error_Handler();
68    }
69    sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
70    sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
71    if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
72    {
73      Error_Handler();
74    }
75
76  }
77
78  static void MX_GPIO_Init(void)
79  {
80    GPIO_InitTypeDef GPIO_InitStruct = {0};
81
82    __HAL_RCC_GPIOA_CLK_ENABLE();
83    __HAL_RCC_GPIOB_CLK_ENABLE();
84
85
86    HAL_GPIO_WritePin(GPIOA, red_h_Pin|yel_h_Pin|grn_h_Pin|red_v_Pin
87                            |yel_v_Pin|grn_v_Pin|en0_Pin|en1_Pin
88                            |en2_Pin|en3_Pin, GPIO_PIN_RESET);
89
90
91    HAL_GPIO_WritePin(GPIOB, a_h_Pin|b_h_Pin|c_h_Pin|d_v_Pin
92                            |e_v_Pin|f_v_Pin|g_v_Pin|d_h_Pin
93                            |e_h_Pin|f_h_Pin|g_h_Pin|a_v_Pin
94                            |b_v_Pin|c_v_Pin, GPIO_PIN_RESET);
95
96
97    GPIO_InitStruct.Pin = red_h_Pin|yel_h_Pin|grn_h_Pin|red_v_Pin
98                            |yel_v_Pin|grn_v_Pin|en0_Pin|en1_Pin
99                            |en2_Pin|en3_Pin;
100   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
101   GPIO_InitStruct.Pull = GPIO_NOPULL;
102   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
103   HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
104
105
106   GPIO_InitStruct.Pin = a_h_Pin|b_h_Pin|c_h_Pin|d_v_Pin
```

```
                          |e_v_Pin|f_v_Pin|g_v_Pin|d_h_Pin
                          |e_h_Pin|f_h_Pin|g_h_Pin|a_v_Pin
                          |b_v_Pin|c_v_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = sw0_Pin|sw1_Pin|sw2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);


}


void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
    buttonRead();
    timerRun();
}

void Error_Handler(void)
{

  __disable_irq();
  while (1)
  {
  }

}

#ifdef  USE_FULL_ASSERT

void assert_failed(uint8_t *file, uint32_t line)
{

}
#endif /* USE_FULL_ASSERT */
```

References