VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

REPORT

LAB 3

Class: Microprocessors - Microcontrollers – CC04

Lecturer: PHAN VĂN SỸ

No.	Full Name	ID Student
1.	**Lê Mạnh Quân**	**2352993**

Ho Chi Minh City, October 2025

Contents

1 Exercise 1

1.1 FSM Mode

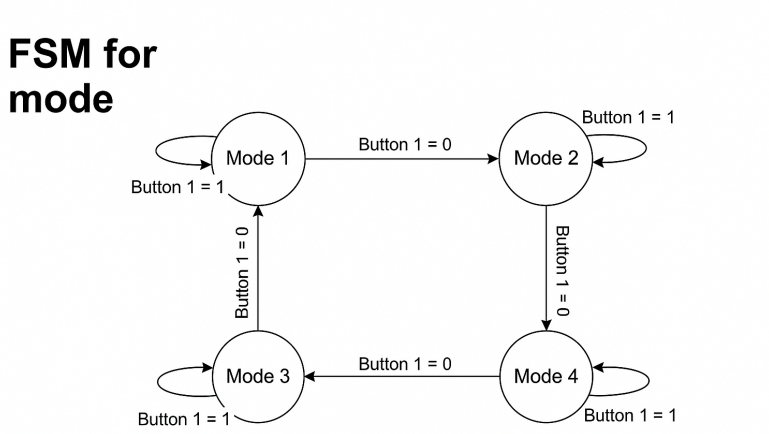


Figure 1: FSM 4 Mode

1.2 FSM Button

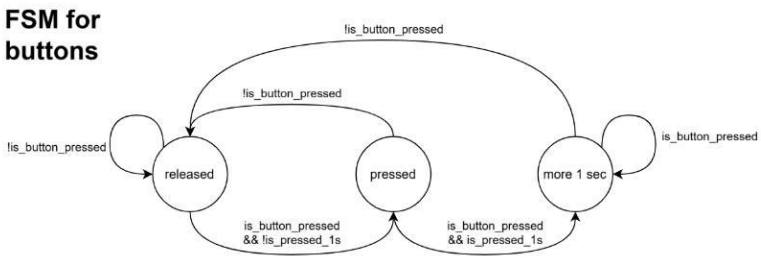


Figure 2: FSM for Button

2 Exercise 2

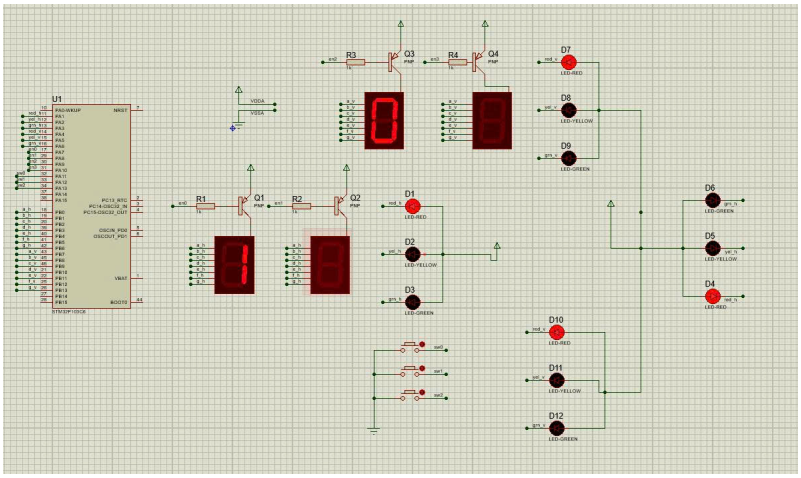
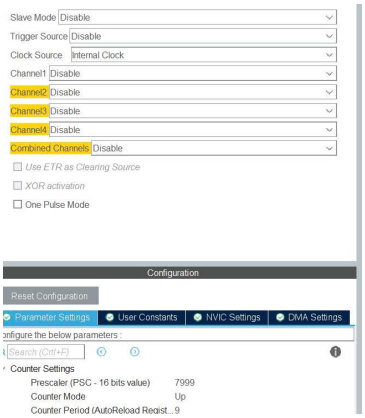
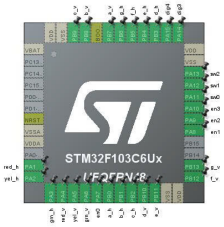


Figure 3: The schematic of Lab3

3 Exercise 3


Slave Mode Disable
Trigger Source Disable
Clock Source Internal Clock
Channel1 Disable
Channel2 Disable
Channel3 Disable
Channel4 Disable
Combined Channels Disable
Use ETR as Clearing Source
XOR activation
One Pulse Mode
Configuration
Reset Configuration
Parameter Settings
User Constants
NVIC Settings
DMA Settings
onfigure the below parameters :
Search (Ctrl+F)
Counter Settings
Prescaler (PSC - 16 bits value) 7999
Counter Mode Up
Counter Period (AutoReload Regist... 9


STM32F103C6Ux

4 timer.h 7 timer.c

4.1 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_ */
```

4.2 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--;
```

```
81      timer3--;
82      timer4--;
83      timer5--;
84      if (timer1 == 0){
85          flag1 = 1;
86      }
87      if (timer2 == 0){
88          flag2 = 1;
89      }
90      if (timer3 == 0){
91          flag3 = 1;
92      }
93      if (timer4 == 0){
94          flag4 = 1;
95      }
96      if (timer5 == 0){
97          flag5 = 1;
98      }
99  }
```

5 button.h button.c

5.1 button.h

```
1
2  #ifndef INC_BUTTON_H_
3  #define INC_BUTTON_H_
4
5  #include "global.h"
6
7  #define NO_BUTTON 3
8
9  #define PRESSED 0
10 #define RELEASED 1
11
12 int isButtonNoPressed(int);
13 void buttonRead();
14
15
16 #endif /* INC_BUTTON_H_ */
```

5.2 timer.c

```
1
2
3
4  #include "button.h"
5
6  int buttonFlag[NO_BUTTON] = {0, 0, 0};
7  //int pressed[NO_BUTTON] = {0, 0, 0};
8  //int longPressed[NO_BUTTON] = {0, 0, 0};
9  int timeout[NO_BUTTON] = {100, 100, 100};
10
11 int KeyReg0[NO_BUTTON] = {RELEASED, RELEASED, RELEASED};
12 int KeyReg1[NO_BUTTON] = {RELEASED, RELEASED, RELEASED};
13 int KeyReg2[NO_BUTTON] = {RELEASED, RELEASED, RELEASED};
14 int KeyReg3[NO_BUTTON] = {RELEASED, RELEASED, RELEASED};
15
16 int isButtonNoPressed(int no){
17         if (buttonFlag[no] == 1){
18                 buttonFlag[no] = 0;
19                 return 1;
20         }
21         return 0;
22 }
23
24 void buttonRead(){
25         for (int i = 0; i < NO_BUTTON; i++){
26                 KeyReg2[i] = KeyReg1[i];
27                 KeyReg1[i] = KeyReg0[i];
28                 KeyReg0[i] = HAL_GPIO_ReadPin(sw0_GPIO_Port,
                       sw0_Pin << i);
```

```
29                  if ((KeyReg1[i] == KeyReg0[i]) && (KeyReg1[i] ==
                        KeyReg2[i])){
30                      if (KeyReg2[i] != KeyReg3[i]){ //reg2 !=
                            reg3
31                          KeyReg3[i] = KeyReg2[i];
32                          if (KeyReg0[i] == PRESSED){
33                                  timeout[i] = 100;
34                                  buttonFlag[i] = 1;
35                          }
36                      }
37                      else { //reg2 = reg3
38                              timeout[i]--;
39                              if (timeout[i] == 0){
40                                      timeout[i] = 10;
41                                      if (KeyReg3[i] == PRESSED){
42                                              buttonFlag[i] = 1;
43                                      }
44                              }
45                      }
46                  }
47          }
48  }
```

6 segment.h segment.c

6.1 segment.h

```
1
2  #ifndef INC_SEGMENT_H_
3  #define INC_SEGMENT_H_
4
5  #include "global.h"
6
7
8  extern int segment_buffer[4];
9
10 void set7SegH(int);
11 void set7SegV(int);
12 void scan7Seg(int);
13
14 void updateSegment(int, int, int, int);
15 void updateSegment2Digits(int, int);
16
17 #endif /* INC_SEGMENT_H_ */
```

6.2 segment.c

```
1
2
3
4
5  #include "segment.h"
6
7  int segment_buffer[4] = {0};
8
9  GPIO_PinState pinArr[11][7] = {
10             {0, 0, 0, 0, 0, 0, 1}, //0
11             {1, 0, 0, 1, 1, 1, 1}, //1
12             {0, 0, 1, 0, 0, 1, 0}, //2
13             {0, 0, 0, 0, 1, 1, 0}, //3
14             {1, 0, 0, 1, 1, 0, 0}, //4
15             {0, 1, 0, 0, 1, 0, 0}, //5
16             {0, 1, 0, 0, 0, 0, 0}, //6
17             {0, 0, 0, 1, 1, 1, 1}, //7
18             {0, 0, 0, 0, 0, 0, 0}, //8
19             {0, 0, 0, 0, 1, 0, 0}, //9
20                 {1, 1, 1, 1, 1, 1, 1}  //ALL LED TURN OFF
21 };
22
23 void set7SegH(int num){
24         if (num >= 0 && num <= 9){
25                 for (int state = 0; state < 7; state++){
26                         HAL_GPIO_WritePin(a_h_GPIO_Port, a_h_Pin <<
                               state, pinArr[num][state]);
27                 }
28         }
```

```
        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;
        segment_buffer[1] = b;
        segment_buffer[2] = c;
        segment_buffer[3] = d;
}

void updateSegment2Digits(int firstNum, int secNum){
```

```
82          segment_buffer[0] = firstNum / 10;
83          segment_buffer[1] = firstNum % 10;
84          segment_buffer[2] = secNum / 10;
85          segment_buffer[3] = secNum % 10;
86  }
```

7 led.h led.c

7.1 led.h

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

7.2 led.c

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

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

```
63          case ALL:
64                  HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin,
                        LED_ON);
65                  HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin,
                        LED_ON);
66                  HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin,
                        LED_ON);
67                  verState = ALL;
68                  break;
69          default:
70                  HAL_GPIO_WritePin(red_v_GPIO_Port, red_v_Pin,
                        LED_OFF);
71                  HAL_GPIO_WritePin(yel_v_GPIO_Port, yel_v_Pin,
                        LED_OFF);
72                  HAL_GPIO_WritePin(grn_v_GPIO_Port, grn_v_Pin,
                        LED_OFF);
73                  verState = NONE;
74                  break;
75          }
76  }
```

8 global.h fsm.c

8.1 global.h

```
1
2  #ifndef INC_GLOBAL_H_
3  #define INC_GLOBAL_H_
4
5  #include "button.h"
6  #include "main.h"
7  #include "timer.h"
8  #include "segment.h"
9  #include "led.h"
10
11 #define RED             11
12 #define YEL             22
13 #define GRN             33
14 #define ALL     44
15 #define NONE    0
16 //#define delay         HAL_Delay
17 #define INIT 1
18
19 #define MAN_RED 10
20 #define MAN_YEL 20
21 #define MAN_GRN 30
22
23 #define IDLE    -1
24
25 void segmentUpdateAuto();
26
27 //void button0Signal();
28
29 void fsm_run();
30 void fsm_auto_hor();
31 void fsm_auto_ver();
32 void fsm_man();
33
34 #endif /* INC_GLOBAL_H_ */
```

8.2 fsm.c

```
1
2  #include "global.h"
3
4  int autoState_H = INIT;
5  int autoState_V = INIT;
6  int manState    = IDLE;
7
8  int redDur = 5;
9  int yelDur = 2;
10 int grnDur = 3;
11
12 int tempRed = 1;
```

```
13 int tempYel = 1;
14 int tempGrn = 1;
15
16 int horCount = 0;
17 int verCount = 0;
18 int scan     = 0;
19
20 void segmentUpdateAuto(void){
21   updateSegment2Digits(horCount, verCount);
22 }
23
24 static void button0Signal(void){
25   if (isButtonNoPressed(0) == 1){
26     resetTimer(-1);
27     horCount = 0; verCount = 0;
28     updateSegment(IDLE, IDLE, IDLE, IDLE);
29     scan = 0;
30     setLedH(IDLE);
31     setLedV(IDLE);
32     autoState_H = IDLE;
33     autoState_V = IDLE;
34     manState    = MAN_RED;
35     set1(100);
36     set3(250);
37   }
38 }
39
40 void fsm_run(void){
41   fsm_auto_hor();
42   fsm_auto_ver();
43   fsm_man();
44 }
45
46 /* -------------------- AUTO HORIZONTAL -------------------- */
47 void fsm_auto_hor(void){
48   switch (autoState_H){
49     case INIT:
50       set1(redDur * 1000);
51       set2(1000);
52       set3(250);
53       horCount   = redDur;
54       autoState_H = RED;
55       break;
56
57     case RED:
58       setLedH(RED);
59
60       if (flag1){
61         set1(grnDur * 1000);
62         horCount    = grnDur;
63         set2(1000);
64         autoState_H = GRN;
65       }
66
67       if (flag2){
68         if (horCount > 0) horCount--;
```

```
69          set2(1000);
70        }
71
72        if (flag3){
73          segmentUpdateAuto();
74          scan ^= 1;
75          scan7Seg(scan);
76          set3(250);
77        }
78        button0Signal();
79        break;
80
81      case GRN:
82        setLedH(GRN);
83
84        if (flag1){
85          set1(yelDur * 1000);
86          horCount    = yelDur;          //      KH NG -1
87          set2(1000);
88          autoState_H = YEL;
89        }
90
91        if (flag2){
92          if (horCount > 0) horCount--;
93          set2(1000);
94        }
95
96        if (flag3){
97          segmentUpdateAuto();
98          scan ^= 1;
99          scan7Seg(scan);
100         set3(250);
101       }
102       button0Signal();
103       break;
104
105     case YEL:
106       setLedH(YEL);
107
108       if (flag1){
109         set1(redDur * 1000);
110         horCount    = redDur;
111         set2(1000);
112         autoState_H = RED;
113       }
114
115       if (flag2){
116         if (horCount > 0) horCount--;
117         set2(1000);
118       }
119
120       if (flag3){
121         segmentUpdateAuto();
122         scan ^= 1;
123         scan7Seg(scan);
124         set3(250);
```

```
      }
      button0Signal();
      break;

    default: /* IDLE */ break;
  }
}

void fsm_auto_ver(void){
  switch (autoState_V){
    case INIT:
      set4(grnDur * 1000);   // timer pha V
      set5(1000);            //  n h p  1s V
      verCount    = grnDur;
      autoState_V = GRN;
      break;

    case GRN:
      setLedV(GRN);

      if (flag4){                       //    i    pha  t r   c
        set4(yelDur * 1000);
        verCount    = yelDur;
        set5(1000);
        autoState_V = YEL;
      }

      if (flag5){
        if (verCount > 0) verCount--;
        set5(1000);
      }
      break;

    case YEL:
      setLedV(YEL);

      if (flag4){
        set4(redDur * 1000);
        verCount    = redDur;
        set5(1000);
        autoState_V = RED;
      }

      if (flag5){
        if (verCount > 0) verCount--;
        set5(1000);
      }
      break;

    case RED:
      setLedV(RED);

      if (flag4){
        set4(grnDur * 1000);
        verCount    = grnDur;
        set5(1000);
```

```
181         autoState_V = GRN;
182       }
183 
184       if (flag5){
185         if (verCount > 0) verCount--;
186         set5(1000);
187       }
188       break;
189 
190     default: break;
191   }
192 }
193 
194 /* -------------------- MANUAL MODES -------------------- */
195 void fsm_man(void){
196   switch (manState){
197     case MAN_RED:
198       updateSegment2Digits(tempRed, 02);
199 
200       if (isButtonNoPressed(0) == 1){
201         tempRed  = 1;
202         manState = MAN_YEL;
203         setLedV(IDLE);
204         setLedH(IDLE);
205         set1(100);
206         set3(250);
207       }
208 
209       if (isButtonNoPressed(1) == 1){
210         tempRed = (tempRed == 99) ? 1 : tempRed + 1;
211       }
212 
213       if (isButtonNoPressed(2) == 1){
214         redDur = tempRed;
215       }
216 
217       if (flag1){
218         HAL_GPIO_TogglePin(red_h_GPIO_Port, red_h_Pin);
219         HAL_GPIO_TogglePin(red_v_GPIO_Port, red_v_Pin);
220         set1(500);
221       }
222 
223       if (flag3){
224         scan ^= 1;
225         scan7Seg(scan);
226         set3(250);
227       }
228       break;
229 
230     case MAN_YEL:
231       updateSegment2Digits(tempYel, 03);
232 
233       if (isButtonNoPressed(0) == 1){
234         tempYel  = 1;
235         manState = MAN_GRN;
236         setLedV(IDLE);
```

```
237         setLedH(IDLE);
238         set1(100);
239         set3(250);
240       }
241
242       if (isButtonNoPressed(1) == 1){
243         tempYel = (tempYel == 99) ? 1 : tempYel + 1;
244       }
245
246       if (isButtonNoPressed(2) == 1){
247         yelDur = tempYel;
248       }
249
250       if (flag1){
251         HAL_GPIO_TogglePin(yel_h_GPIO_Port, yel_h_Pin);
252         HAL_GPIO_TogglePin(yel_v_GPIO_Port, yel_v_Pin);
253         set1(500);
254       }
255
256       if (flag3){
257         scan ^= 1;
258         scan7Seg(scan);
259         set3(250);
260       }
261       break;
262
263     case MAN_GRN:
264       updateSegment2Digits(tempGrn, 04);
265
266       if (isButtonNoPressed(0) == 1){
267
268         if (yelDur > grnDur){
269           grnDur += yelDur;
270         }
271         if (redDur < grnDur + yelDur){
272           redDur = grnDur + yelDur;
273         }
274         if (grnDur >= redDur + yelDur){
275           grnDur = redDur - yelDur;
276         }
277
278         setLedH(ALL);
279         setLedV(ALL);
280
281
282
283
284         setLedV(IDLE);
285         setLedH(IDLE);
286         resetTimer(NONE);          // reset all timer
287
288         manState    = IDLE;
289         autoState_H = INIT;
290         autoState_V = INIT;
291         return;
292       }
```

```
293
294       if (isButtonNoPressed(1) == 1){
295         tempGrn = (tempGrn == 99) ? 1 : tempGrn + 1;
296       }
297
298       if (isButtonNoPressed(2) == 1){
299         grnDur = tempGrn;
300       }
301
302       if (flag1){
303         HAL_GPIO_TogglePin(grn_h_GPIO_Port, grn_h_Pin);
304         HAL_GPIO_TogglePin(grn_v_GPIO_Port, grn_v_Pin);
305         set1(500);
306       }
307
308       if (flag3){
309         scan ^= 1;
310         scan7Seg(scan);
311         set3(250);
312       }
313       break;
314
315     default: break;
316   }
317 }
```

9 main.c

9.1 main.c

```
while (1)
  {
//          fsm_auto_hor();
            fsm_run();
  /* USER CODE END 3 */
  }
}
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
    buttonRead();
    timerRun();
}
```

10 Source

GG Drive Link: My Source Code Github Link: My Source Code