

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



THÍ NGHIỆM

VI XỬ LÝ - VI ĐIỀU KHIỂN (CO 3010)

BÁO CÁO LAB 3

Giảng viên: PHAN VĂN SỸ

Họ và tên	Mã số sinh viên
Nguyễn Văn Sang	2212912

TP.Hồ Chí Minh, 05/11/2025

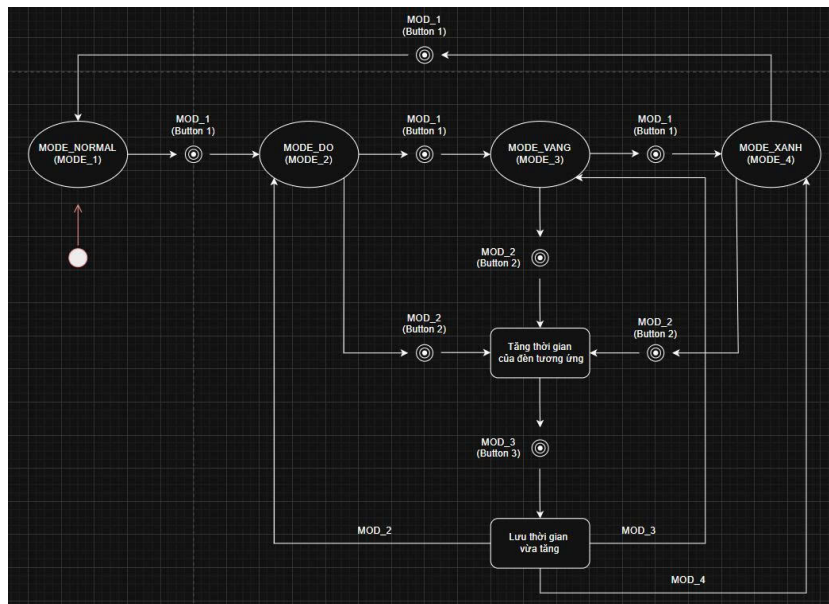


Mục lục

1	Bài 1 Sketch an FSM	2
2	Bài 2 Proteus Schematic	2
3	Bài 3 Create STM32 Project	3
4	Bài 4 Modify Timer Parameters	4
5	Bài 5 Adding code for button debouncing	4
6	Bài 6 Adding code for displaying modes	13
7	Bài 7 Adding code for increasing time duration value for the red LEDs	15
8	Bài 8 Adding code for increasing time duration value for the amber LEDs	16
9	Bài 9 Adding code for increasing time duration value for the green LEDs	16
10	Tổng hợp LAB 3	16

1 Bài 1 Sketch an FSM

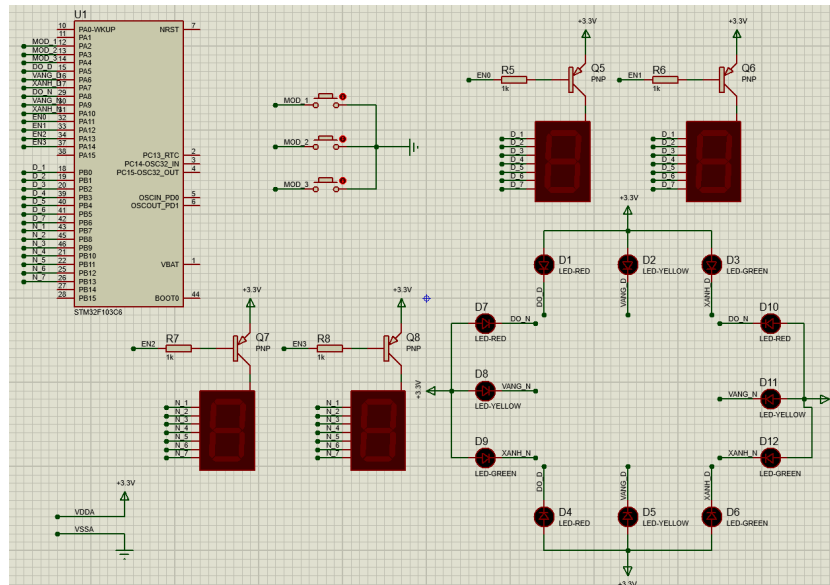
Your task in this exercise is to sketch an FSM that describes your idea of how to solve the problem.



Hình 1: FSM state diagram

2 Bài 2 Proteus Schematic

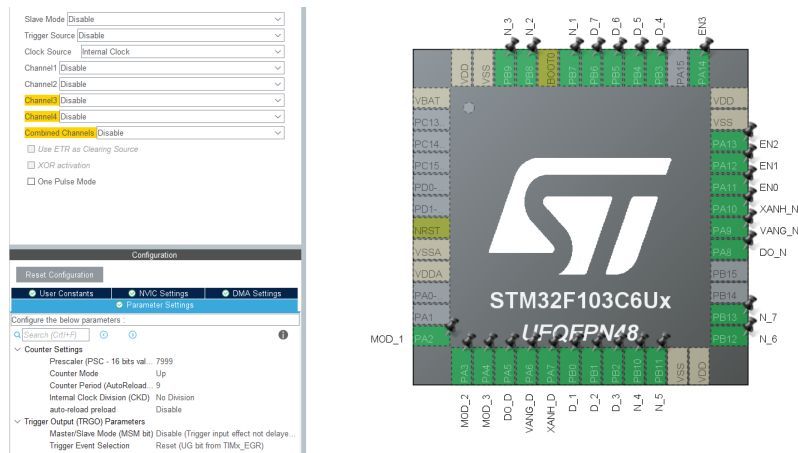
Your task in this exercise is to draw a Proteus schematic for the problem above.



Hình 2: Schematic of exercise

3 Bài 3 Create STM32 Project

Your task in this exercise is to create a project that has pin corresponding to the Proteus schematic that you draw in previous section. You need to set up your timer interrupt is about 10ms.



Hình 3: Set up pins and timer_interrupt for the project

4 Bài 4 Modify Timer Parameters

Your task in this exercise is to modify the timer settings so that when we want to change the time duration of the timer interrupt, we change it the least and it will not affect the overall system. For example, the current system we have implemented is that it can blink an LED in 2 Hz, with the timer interrupt duration is 10ms. However, when we want to change the timer interrupt duration to 1ms or 100ms, it will not affect the 2Hz blinking LED.

```
1 // ----- software_timer.h ----- //
2 #ifndef TIMER_TICK_MS
3 #define TIMER_TICK_MS 10
4 #endif
5
6 // ----- main.c ----- //
7 htim2.Init.Prescaler = 7999;
8 htim2.Init.Period = TIMER_TICK_MS - 1;
```

Để đạt được yêu cầu thì ta sẽ sửa đổi thông số $Period = TIMER_TICK_MS - 1$.

Vì $T = \frac{1}{\frac{(Prescaler + 1)(Period + 1)}{8.10^6}} = \frac{1}{\frac{(Prescaler + 1)(TIMER_TICK_MS)}{8.10^6}}$ (s) nên khi ta

tiến hành thành đổi $TIMER_TICK_MS$, thông số $Period$ cũng sẽ thay đổi theo, kéo theo việc T cũng thay đổi, khi đó sẽ thỏa mãn được yêu cầu đặt ra của đề bài.

5 Bài 5 Adding code for button debouncing

Following the example of button reading and debouncing in the previous section, your tasks in this exercise are:

- To add new files for input reading and output display.

```
1 // ----- button.h ----- //
2 #ifndef INC_BUTTON_H_
3 #define INC_BUTTON_H_
4
5 #include "main.h"
6
7 #ifndef NORMAL_STATE
8 #define NORMAL_STATE GPIO_PIN_SET
9 #endif
10 #ifndef ACTIVE_STATE
11 #define ACTIVE_STATE GPIO_PIN_RESET
12 #endif
13
14 extern int MOD_1_ACTIVE;
15 int check_MOD_1();
16
17 extern int MOD_2_ACTIVE;
18 int check_MOD_2();
19
20 extern int MOD_3_ACTIVE;
21 int check_MOD_3();
22
23 int check_MOD_1_long();
24 void reset_MOD_flags();
25 void process_input();
26
```

```
27 #endif /* INC_BUTTON_H */
28
29 // ----- button.c ----- //
30 #include "button.h"
31 #include "software_timer.h"
32
33 #ifndef AUTO_REPEAT_MS
34 #define AUTO_REPEAT_MS 100
35 #endif
36
37 static int Tmp_0 = NORMAL_STATE;
38 static int Tmp_1 = NORMAL_STATE;
39 static int Tmp_2 = NORMAL_STATE;
40 static int Tmp_3 = NORMAL_STATE;
41
42 static int MOD_1_state = NORMAL_STATE;
43 int MOD_1_ACTIVE = 0;
44
45 static int MOD_2_state = NORMAL_STATE;
46 int MOD_2_ACTIVE = 0;
47
48 static int MOD_3_state = NORMAL_STATE;
49 int MOD_3_ACTIVE = 0;
50
51 static int KeyTimeout = AUTO_REPEAT_MS / TIMER_TICK_MS;
52 static int MOD_1_long = 0;
53
54 int check_MOD_1() {
55     if (MOD_1_ACTIVE) {
56         MOD_1_ACTIVE = 0;
57         return 1;
58     }
59     return 0;
60 }
61
62 int check_MOD_2() {
63     if (MOD_2_ACTIVE) {
64         MOD_2_ACTIVE = 0;
65         return 1;
66     }
67     return 0;
68 }
69
70 int check_MOD_3() {
71     if (MOD_3_ACTIVE) {
72         MOD_3_ACTIVE = 0;
73         return 1;
74     }
75     return 0;
76 }
77
78 int check_MOD_1_long() {
79     if (MOD_1_long) {
80         MOD_1_long = 0;
81         return 1;
82     }
83     return 0;
84 }
85
86 void reset_MOD_flags() {
87     MOD_1_ACTIVE = 0;
88     MOD_2_ACTIVE = 0;
```

```
89     MOD_3_ACTIVE = 0;
90 }
91
92 static void sample_inputs(void) {
93     Tmp_2 = Tmp_1;
94     Tmp_1 = Tmp_0;
95     MOD_1_state = HAL_GPIO_ReadPin(MOD_1_GPIO_Port, MOD_1_Pin);
96     MOD_2_state = HAL_GPIO_ReadPin(MOD_2_GPIO_Port, MOD_2_Pin);
97     MOD_3_state = HAL_GPIO_ReadPin(MOD_3_GPIO_Port, MOD_3_Pin);
98     Tmp_0 = MOD_1_state && MOD_2_state && MOD_3_state;
99 }
100
101 static void set_active_for_current_mod(void) {
102     if (MOD_1_state == ACTIVE_STATE) {
103         MOD_1_ACTIVE = 1;
104     } else if (MOD_2_state == ACTIVE_STATE) {
105         MOD_2_ACTIVE = 1;
106     } else if (MOD_3_state == ACTIVE_STATE) {
107         MOD_3_ACTIVE = 1;
108     }
109 }
110
111 void process_input() {
112     sample_inputs();
113     if ((Tmp_1 == Tmp_0) && (Tmp_1 == Tmp_2)) {
114         if (Tmp_2 != Tmp_3) {
115             Tmp_3 = Tmp_2;
116             if (Tmp_3 == ACTIVE_STATE) {
117                 KeyTimeout = AUTO_REPEAT_MS;
118                 set_active_for_current_mod();
119             }
120         } else {
121             KeyTimeout--;
122             if (KeyTimeout == 0) {
123                 KeyTimeout = AUTO_REPEAT_MS;
124                 if (Tmp_3 == ACTIVE_STATE) {
125                     set_active_for_current_mod();
126                 }
127             }
128         }
129     }
130 }
131
132 // ----- global.h ----- //
133 #ifndef INC_GLOBAL_H_
134 #define INC_GLOBAL_H_
135
136 #include "software_timer.h"
137 #include "button.h"
138
139 #define MODE_NORMAL 1
140 #define MODE_DO 2
141 #define MODE_VANG 3
142 #define MODE_XANH 4
143
144 extern int MOD_DO_TIMER;
145 extern int DO_duration;
146
147 extern int MOD_VANG_TIMER;
148 extern int VANG_duration;
149
150 extern int MOD_XANH_TIMER;
```

```
151 extern int XANH_duration;
152
153 void apply_changes(int MODE_CURRENT);
154 void init_traffic_timers(int MODE_CURRENT);
155 void reset_to_defaults(int MODE_CURRENT);
156 void run_traffic_lights(void);
157 void configure_leds(int MODE_CURRENT);
158 void show_7seg(int value, uint32_t pin);
159 void show_mode(int value);
160 void update_buffer_mode(int value);
161 void update_buffer_time(int value);
162 void scan_7seg(int idx);
163 void show_time(int value);
164 void verify_timers(void);
165
166 #endif /* INC_GLOBAL_H_ */
167
168 // ----- global.c ----- //
169 #include "global.h"
170
171 #ifndef SCAN_INTERVAL_MS
172 #define SCAN_INTERVAL_MS 125
173 #endif
174
175 #ifndef LIGHT_STEP_MS
176 #define LIGHT_STEP_MS 400
177 #endif
178
179 int buf_mode[2] = {1, 2};
180 int buf_time[2] = {3, 4};
181
182 int DO_duration = 10;
183 int MOD_DO_TIMER = 10;
184 int DO_modified = 0;
185
186 int VANG_duration = 3;
187 int MOD_VANG_TIMER = 3;
188 int VANG_modified = 0;
189
190 int XANH_duration = 7;
191 int MOD_XANH_TIMER = 7;
192 int XANH_modified = 0;
193
194 int idx_mode = 0;
195 int idx_time = 2;
196
197 int direction = 1;
198
199 void show_7seg(int value, uint32_t pin) {
200     if(value < 0 || value > 9)
201         return;
202     char seg_patterns[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0
203         x80, 0x90};
204     for (int j = 0; j < 7; j++) {
205         HAL_GPIO_WritePin(GPIOB, pin << j, (seg_patterns[value] >> j) & 1);
206     }
207
208 void scan_7seg(int idx) {
209     switch (idx) {
210     case 0:
211         HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, GPIO_PIN_SET);
```



```
212         show_7seg(buf_mode[0], GPIO_PIN_0);
213         HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, GPIO_PIN_RESET);
214         break;
215     case 1:
216         HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, GPIO_PIN_SET);
217         show_7seg(buf_mode[1], GPIO_PIN_0);
218         HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, GPIO_PIN_RESET);
219         break;
220     case 2:
221         HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, GPIO_PIN_SET);
222         show_7seg(buf_time[0], GPIO_PIN_7);
223         HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, GPIO_PIN_RESET);
224         break;
225     case 3:
226         HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, GPIO_PIN_SET);
227         show_7seg(buf_time[1], GPIO_PIN_7);
228         HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, GPIO_PIN_RESET);
229         break;
230     default:
231         break;
232     }
233 }
234
235 void update_buffer_mode(int value) {
236     if (value >= 0) {
237         buf_mode[0] = value / 10;
238         buf_mode[1] = value % 10;
239     }
240 }
241
242 void update_buffer_time(int value) {
243     if (value >= 0) {
244         buf_time[0] = value / 10;
245         buf_time[1] = value % 10;
246     }
247 }
248
249 void show_mode(int value) {
250     update_buffer_mode(value);
251     if (tmr3_done == 1) {
252         scan_7seg(idx_mode);
253         idx_mode = (idx_mode + 1) % 2;
254         init_tmr3(SCAN_INTERVAL_MS);
255     }
256 }
257
258 void show_time(int value) {
259     update_buffer_time(value);
260     if (tmr4_done == 1) {
261         scan_7seg(idx_time);
262         idx_time++;
263         if (idx_time > 3) {
264             idx_time = 2;
265         }
266         init_tmr4(SCAN_INTERVAL_MS);
267     }
268 }
269
270 void verify_timers() {
271     if (DO_modified == 0) {
272         MOD_DO_TIMER = DO_duration;
273     } else {
```

```
274         if (VANG_modified == 0) {
275             MOD_VANG_TIMER = VANG_duration;
276         } else {
277             if (XANH_modified == 0) {
278                 MOD_XANH_TIMER = XANH_duration;
279             }
280         }
281     }
282     if (DO_modified == 1 || VANG_modified == 1 || XANH_modified == 1) {
283         if (MOD_DO_TIMER == (MOD_VANG_TIMER + MOD_XANH_TIMER)) {
284             DO_duration = MOD_DO_TIMER;
285             VANG_duration = MOD_VANG_TIMER;
286             XANH_duration = MOD_XANH_TIMER;
287             DO_modified = 0;
288             VANG_modified = 0;
289             XANH_modified = 0;
290         }
291     }
292 }
293
294 void reset_to_defaults(int MODE_CURRENT) {
295     switch (MODE_CURRENT) {
296         case MODE_NORMAL:
297             verify_timers();
298             MOD_DO_TIMER = DO_duration;
299             MOD_VANG_TIMER = VANG_duration;
300             MOD_XANH_TIMER = XANH_duration;
301             break;
302         case MODE_DO:
303             MOD_DO_TIMER = DO_duration;
304             break;
305         case MODE_VANG:
306             MOD_VANG_TIMER = VANG_duration;
307             break;
308         case MODE_XANH:
309             MOD_XANH_TIMER = XANH_duration;
310             break;
311         default:
312             break;
313     }
314 }
315
316 void init_traffic_timers(int MODE_CURRENT) {
317     switch (MODE_CURRENT) {
318         case MODE_DO:
319             MOD_DO_TIMER = (MOD_DO_TIMER % 99) + 1;
320             break;
321         case MODE_VANG:
322             MOD_VANG_TIMER = (MOD_VANG_TIMER % 99) + 1;
323             break;
324         case MODE_XANH:
325             MOD_XANH_TIMER = (MOD_XANH_TIMER % 99) + 1;
326             break;
327         default:
328             break;
329     }
330 }
331
332 void configure_leds(int MODE_CURRENT) {
333     if (tmr2_done == 1) {
334         if (MODE_CURRENT == MODE_DO) {
335             HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5|GPIO_PIN_8);
```

```
336     } else if (MODE_CURRENT == MODE_VANG) {
337         HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_6|GPIO_PIN_9);
338     } else if (MODE_CURRENT == MODE_XANH) {
339         HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7|GPIO_PIN_10);
340     }
341     init_tmr2(250);
342 }
343 }
344
345 static inline void config_north(GPIO_PinState r, GPIO_PinState y,
346     GPIO_PinState g) {
347     HAL_GPIO_WritePin(DO_N_GPIO_Port, DO_N_Pin, r);
348     HAL_GPIO_WritePin(VANG_N_GPIO_Port, VANG_N_Pin, y);
349     HAL_GPIO_WritePin(XANH_N_GPIO_Port, XANH_N_Pin, g);
350 }
351
352 static inline void config_south(GPIO_PinState r, GPIO_PinState y,
353     GPIO_PinState g) {
354     HAL_GPIO_WritePin(DO_D_GPIO_Port, DO_D_Pin, r);
355     HAL_GPIO_WritePin(VANG_D_GPIO_Port, VANG_D_Pin, y);
356     HAL_GPIO_WritePin(XANH_D_GPIO_Port, XANH_D_Pin, g);
357 }
358
359 void run_traffic_lights() {
360     if (tmr1_done == 1) {
361         if (direction == 1) {
362             if (MOD_DO_TIMER > 0) {
363                 update_buffer_time(MOD_DO_TIMER);
364                 if (MOD_DO_TIMER > VANG_duration) {
365                     if (MOD_XANH_TIMER > 0) {
366                         config_north(GPIO_PIN_RESET, GPIO_PIN_SET,
367                             GPIO_PIN_SET);
368                         config_south(GPIO_PIN_SET, GPIO_PIN_SET,
369                             GPIO_PIN_RESET);
370                         update_buffer_mode(MOD_XANH_TIMER);
371                         MOD_XANH_TIMER -= 1;
372                     }
373                 } else {
374                     if (MOD_DO_TIMER <= VANG_duration && MOD_DO_TIMER >= 1) {
375                         if (MOD_VANG_TIMER > 0) {
376                             config_north(GPIO_PIN_RESET, GPIO_PIN_SET,
377                                 GPIO_PIN_SET);
378                             config_south(GPIO_PIN_SET, GPIO_PIN_RESET,
379                                 GPIO_PIN_SET);
380                             update_buffer_mode(MOD_VANG_TIMER);
381                             MOD_VANG_TIMER -= 1;
382                         }
383                     }
384                 }
385             }
386             MOD_DO_TIMER -= 1;
387         }
388         if (MOD_VANG_TIMER == 0) {
389             MOD_VANG_TIMER = VANG_duration;
390         }
391         if (MOD_XANH_TIMER == 0) {
392             MOD_XANH_TIMER = XANH_duration;
393         }
394         if (MOD_DO_TIMER == 0) {
395             MOD_DO_TIMER = DO_duration;
396             direction = 2;
397         }
398     } else {
```

```
392         if (MOD_DO_TIMER > 0) {
393             update_buffer_mode(MOD_DO_TIMER);
394             if (MOD_DO_TIMER > VANG_duration) {
395                 if (MOD_XANH_TIMER > 0) {
396                     config_south(GPIO_PIN_RESET, GPIO_PIN_SET,
397                                 GPIO_PIN_SET);
398                     config_north(GPIO_PIN_SET, GPIO_PIN_SET,
399                                GPIO_PIN_RESET);
400                     update_buffer_time(MOD_XANH_TIMER);
401                     MOD_XANH_TIMER -= 1;
402                 }
403             } else {
404                 if (MOD_DO_TIMER <= VANG_duration && MOD_DO_TIMER >= 1) {
405                     if (MOD_VANG_TIMER > 0) {
406                         config_south(GPIO_PIN_RESET, GPIO_PIN_SET,
407                                     GPIO_PIN_SET);
408                         config_north(GPIO_PIN_SET, GPIO_PIN_RESET,
409                                    GPIO_PIN_SET);
410                         update_buffer_time(MOD_VANG_TIMER);
411                         MOD_VANG_TIMER -= 1;
412                     }
413                 }
414             }
415             MOD_DO_TIMER -= 1;
416         }
417         if (MOD_VANG_TIMER == 0) {
418             MOD_VANG_TIMER = VANG_duration;
419         }
420         if (MOD_XANH_TIMER == 0) {
421             MOD_XANH_TIMER = XANH_duration;
422         }
423         if (MOD_DO_TIMER == 0) {
424             MOD_DO_TIMER = DO_duration;
425             direction = 1;
426         }
427     }
428     init_tmr1(LIGHT_STEP_MS);
429 }
430 if (tmr5_done == 1) {
431     scan_7seg(idx_mode);
432     idx_mode = (idx_mode + 1) % 2;
433     scan_7seg(idx_time);
434     idx_time++;
435     if (idx_time > 3) idx_time = 2;
436     init_tmr5(SCAN_INTERVAL_MS);
437 }
438 }
439
440 void apply_changes(int MODE_CURRENT) {
441     switch (MODE_CURRENT) {
442         case MODE_DO:
443             DO_modified = 1;
444             break;
445         case MODE_VANG:
446             VANG_modified = 1;
447             break;
448         case MODE_XANH:
449             XANH_modified = 1;
450             break;
451         default:
452             break;
453     }
454 }
```

450 }

- To add code for button debouncing.

```
1 // ----- button.c ----- //
2 static void sample_inputs(void) {
3     Tmp_2 = Tmp_1;
4     Tmp_1 = Tmp_0;
5     MOD_1_state = HAL_GPIO_ReadPin(MOD_1_GPIO_Port, MOD_1_Pin);
6     MOD_2_state = HAL_GPIO_ReadPin(MOD_2_GPIO_Port, MOD_2_Pin);
7     MOD_3_state = HAL_GPIO_ReadPin(MOD_3_GPIO_Port, MOD_3_Pin);
8     Tmp_0 = MOD_1_state && MOD_2_state && MOD_3_state;
9 }
10
11 static void set_active_for_current_mod(void) {
12     if (MOD_1_state == ACTIVE_STATE) {
13         MOD_1_ACTIVE = 1;
14     } else if (MOD_2_state == ACTIVE_STATE) {
15         MOD_2_ACTIVE = 1;
16     } else if (MOD_3_state == ACTIVE_STATE) {
17         MOD_3_ACTIVE = 1;
18     }
19 }
20
21 void process_input() {
22     sample_inputs();
23     if ((Tmp_1 == Tmp_0) && (Tmp_1 == Tmp_2)) {
24         if (Tmp_2 != Tmp_3) {
25             Tmp_3 = Tmp_2;
26             if (Tmp_3 == ACTIVE_STATE) {
27                 KeyTimeout = AUTO_REPEAT_MS;
28                 set_active_for_current_mod();
29             }
30         } else {
31             KeyTimeout--;
32             if (KeyTimeout == 0) {
33                 KeyTimeout = AUTO_REPEAT_MS;
34                 if (Tmp_3 == ACTIVE_STATE) {
35                     set_active_for_current_mod();
36                 }
37             }
38         }
39     }
40 }
```

- To add code for increasing mode when the first button is pressed.

```
1 // ----- fsm.c ----- //
2 int MODE_CURRENT = 1;
3
4 static void handle_mode() {
5     switch (MODE_CURRENT) {
6         case MODE_NORMAL:
7             run_traffic_lights();
8             break;
9         case MODE_DO:
10            configure_and_display_mode(MODE_DO, MOD_DO_TIMER);
11            break;
12         case MODE_VANG:
13            configure_and_display_mode(MODE_VANG, MOD_VANG_TIMER);
14            break;
15         case MODE_XANH:
```

```
16         configure_and_display_mode(MODE_XANH, MOD_XANH_TIMER);
17         break;
18     default:
19         break;
20     }
21 }
22
23 void execute_fsm() {
24     handle_mode();
25     if (check_MOD_1()) {
26         MODE_CURRENT = (MODE_CURRENT % 4) + 1;
27         turn_Off_LEDs();
28         reset_to_defaults(MODE_CURRENT);
29         reset_MOD_flags();
30     }
31     if (MODE_CURRENT != 1) {
32         if (check_MOD_2()) {
33             init_traffic_timers(MODE_CURRENT);
34         }
35     }
36     if (check_MOD_3()) {
37         apply_changes(MODE_CURRENT);
38     }
39 }
```

6 Bài 6 Adding code for displaying modes

Your tasks in this exercise are:

- To add code for display mode on seven-segment LEDs.

```
1 // ----- fsm.c ----- //
2 static void configure_and_display_mode(int mode, int timer_constant) {
3     configure_leds(mode);
4     show_mode(mode);
5     show_time(timer_constant);
6 }
7
8 static void handle_mode() {
9     switch (MODE_CURRENT) {
10         case MODE_NORMAL:
11             run_traffic_lights();
12             break;
13         case MODE_DO:
14             configure_and_display_mode(MODE_DO, MOD_DO_TIMER);
15             break;
16         case MODE_VANG:
17             configure_and_display_mode(MODE_VANG, MOD_VANG_TIMER);
18             break;
19         case MODE_XANH:
20             configure_and_display_mode(MODE_XANH, MOD_XANH_TIMER);
21             break;
22         default:
23             break;
24     }
25 }
26
27 // ----- global.c ----- //
28 void show_7seg(int value, uint32_t pin) {
29     if(value < 0 || value > 9)
```

```
30         return;
31     char seg_patterns[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0
32         x80, 0x90};
33     for (int j = 0; j < 7; j++) {
34         HAL_GPIO_WritePin(GPIOB, pin << j, (seg_patterns[value] >> j) & 1);
35     }
36 }
37 void scan_7seg(int idx) {
38     switch (idx) {
39         case 0:
40             HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, GPIO_PIN_SET);
41             show_7seg(buf_mode[0], GPIO_PIN_0);
42             HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, GPIO_PIN_RESET);
43             break;
44         case 1:
45             HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, GPIO_PIN_SET);
46             show_7seg(buf_mode[1], GPIO_PIN_0);
47             HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, GPIO_PIN_RESET);
48             break;
49         case 2:
50             HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, GPIO_PIN_SET);
51             show_7seg(buf_time[0], GPIO_PIN_7);
52             HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, GPIO_PIN_RESET);
53             break;
54         case 3:
55             HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, GPIO_PIN_SET);
56             show_7seg(buf_time[1], GPIO_PIN_7);
57             HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, GPIO_PIN_RESET);
58             break;
59         default:
60             break;
61     }
62 }
63
64 void update_buffer_mode(int value) {
65     if (value >= 0) {
66         buf_mode[0] = value / 10;
67         buf_mode[1] = value % 10;
68     }
69 }
70
71 void update_buffer_time(int value) {
72     if (value >= 0) {
73         buf_time[0] = value / 10;
74         buf_time[1] = value % 10;
75     }
76 }
77
78 void show_mode(int value) {
79     update_buffer_mode(value);
80     if (tmr3_done == 1) {
81         scan_7seg(idx_mode);
82         idx_mode = (idx_mode + 1) % 2;
83         init_tmr3(SCAN_INTERVAL_MS);
84     }
85 }
86
87 void show_time(int value) {
88     update_buffer_time(value);
89     if (tmr4_done == 1) {
90         scan_7seg(idx_time);
```

```
91     idx_time++;
92     if (idx_time > 3) {
93         idx_time = 2;
94     }
95     init_tmr4(SCAN_INTERVAL_MS);
96 }
97 }
```

- To add code for blinking LEDs depending on the mode that is selected.

```
1  // ----- fsm.c ----- //
2  static void configure_and_display_mode(int mode, int timer_constant) {
3      configure_leds(mode);
4      show_mode(mode);
5      show_time(timer_constant);
6  }
7
8  static void handle_mode() {
9      switch (MODE_CURRENT) {
10         case MODE_NORMAL:
11             run_traffic_lights();
12             break;
13         case MODE_DO:
14             configure_and_display_mode(MODE_DO, MOD_DO_TIMER);
15             break;
16         case MODE_VANG:
17             configure_and_display_mode(MODE_VANG, MOD_VANG_TIMER);
18             break;
19         case MODE_XANH:
20             configure_and_display_mode(MODE_XANH, MOD_XANH_TIMER);
21             break;
22         default:
23             break;
24     }
25 }
26
27 // ----- global.c ----- //
28 void configure_leds(int MODE_CURRENT) {
29     if (tmr2_done == 1) {
30         if (MODE_CURRENT == MODE_DO) {
31             HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5|GPIO_PIN_8);
32         } else if (MODE_CURRENT == MODE_VANG) {
33             HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_6|GPIO_PIN_9);
34         } else if (MODE_CURRENT == MODE_XANH) {
35             HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7|GPIO_PIN_10);
36         }
37         init_tmr2(250);
38     }
39 }
```

7 Bài 7 Adding code for increasing time duration value for the red LEDs

Your tasks in this exercise are:

- To use the second button to increase the time duration value of the red LEDs.

```
1  // ----- global.c ----- //
2  case MODE_DO:
```



```
3 MOD_DO_TIMER = (MOD_DO_TIMER % 99) + 1;  
4 break;
```

- To use the third button to set the value for the red LEDs.

```
1 // ----- global.c ----- //  
2 case MODE_DO:  
3     DO_modified = 1;  
4     break;
```

8 Bài 8 Adding code for increasing time duration value for the amber LEDs

Your tasks in this exercise are:

- To use the second button to increase the time duration value of the amber LEDs.

```
1 // ----- global.c ----- //  
2 case MODE_VANG:  
3     MOD_VANG_TIMER = (MOD_VANG_TIMER % 99) + 1;  
4     break;
```

- To use the third button to set the value for the amber LEDs.

```
1 // ----- global.c ----- //  
2 case MODE_VANG:  
3     VANG_modified = 1;  
4     break;
```

9 Bài 9 Adding code for increasing time duration value for the green LEDs

Your tasks in this exercise are:

- To use the second button to increase the time duration value of the green LEDs.

```
1 // ----- global.c ----- //  
2 case MODE_XANH:  
3     MOD_XANH_TIMER = (MOD_XANH_TIMER % 99) + 1;  
4     break;
```

- To use the third button to set the value for the green LEDs.

```
1 // ----- global.c ----- //  
2 case MODE_XANH:  
3     XANH_modified = 1;  
4     break;
```

10 Tổng hợp LAB 3

Link video demo: https://drive.google.com/file/d/1iXA3bZZBbTN_pZSBT3JlWQ2iapAx0gyc/view?usp=sharing

Link Github: <https://github.com/SangNguyen-232/VXLVDK-LAB3>