

VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



Microprocessor - Microcontroller

Lab Report - CO3010 - CC02

Lab 3

Advisor(s): Phan Văn Sỹ

Student(s): Lương Đức Hiếu 2352324

HO CHI MINH CITY, OCTOBER 2025



Contents

1	Overall	3
2	Exercise	3
2.1	Exercise 1: Sketch an FSM	3
2.2	Exercise 2: Proteus Schematic	4
2.3	Exercise 3: Create STM32 Project	4
2.3.1	main.h	4
2.3.2	main.c	6
2.4	Exercise 4: Modify Timer Parameters	9
2.5	Exercise 5: Button debouncing	10
2.5.1	Input reading	10
2.5.2	sw0 handling	11
2.6	Exercise 6: Displaying modes	12
2.6.1	Mode handling	12
2.6.2	Display led	15
2.7	Exercise 7,8,9: Increasing time duration value for the red, amber and green LEDs	16
2.7.1	sw1 handling	16
2.7.2	sw2 handling	17
2.8	Exercise 10: To finish the project	18

1 Overall

The GitHub link for the Lab 3 project is at : <https://github.com/hieuld1003/MPU-MCU>

2 Exercise

2.1 Exercise 1: Sketch an FSM

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

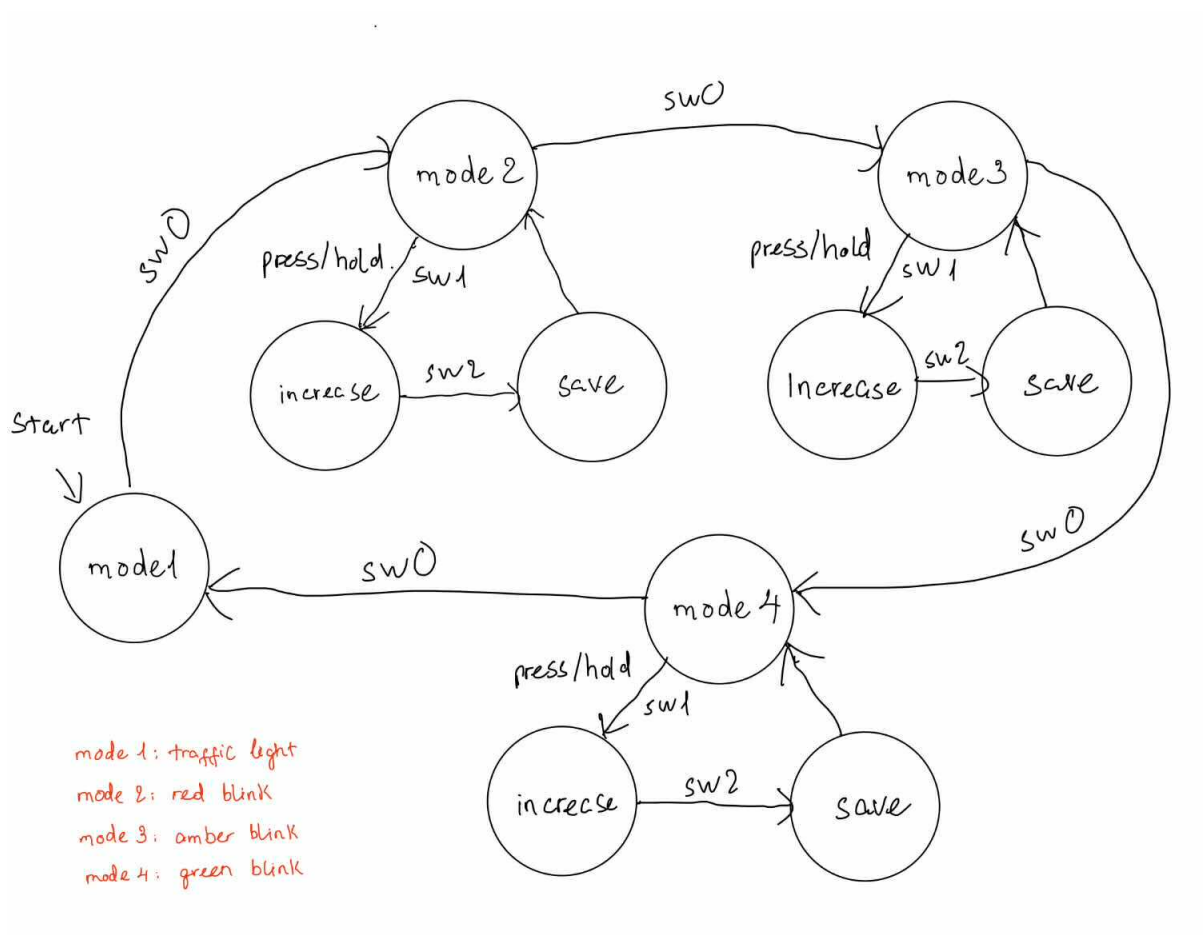


Figure 2.1: FSM Sketch



```
6 #define yellow_h_GPIO_Port GPIOA
7 #define green_h_Pin GPIO_PIN_3
8 #define green_h_GPIO_Port GPIOA
9 #define red_v_Pin GPIO_PIN_4
10 #define red_v_GPIO_Port GPIOA
11 #define yellow_v_Pin GPIO_PIN_5
12 #define yellow_v_GPIO_Port GPIOA
13 #define green_v_Pin GPIO_PIN_6
14 #define green_v_GPIO_Port GPIOA
15 #define en0_Pin GPIO_PIN_7
16 #define en0_GPIO_Port GPIOA
17 #define a_h_Pin GPIO_PIN_0
18 #define a_h_GPIO_Port GPIOB
19 #define b_h_Pin GPIO_PIN_1
20 #define b_h_GPIO_Port GPIOB
21 #define c_h_Pin GPIO_PIN_2
22 #define c_h_GPIO_Port GPIOB
23 #define d_v_Pin GPIO_PIN_10
24 #define d_v_GPIO_Port GPIOB
25 #define e_v_Pin GPIO_PIN_11
26 #define e_v_GPIO_Port GPIOB
27 #define f_v_Pin GPIO_PIN_12
28 #define f_v_GPIO_Port GPIOB
29 #define g_v_Pin GPIO_PIN_13
30 #define g_v_GPIO_Port GPIOB
31 #define en1_Pin GPIO_PIN_8
32 #define en1_GPIO_Port GPIOA
33 #define en2_Pin GPIO_PIN_9
34 #define en2_GPIO_Port GPIOA
35 #define en3_Pin GPIO_PIN_10
36 #define en3_GPIO_Port GPIOA
37 #define sw0_Pin GPIO_PIN_11
38 #define sw0_GPIO_Port GPIOA
39 #define sw1_Pin GPIO_PIN_12
40 #define sw1_GPIO_Port GPIOA
41 #define sw2_Pin GPIO_PIN_13
42 #define sw2_GPIO_Port GPIOA
43 #define d_h_Pin GPIO_PIN_3
```



```
44 #define d_h_GPIO_Port GPIOB
45 #define e_h_Pin GPIO_PIN_4
46 #define e_h_GPIO_Port GPIOB
47 #define f_h_Pin GPIO_PIN_5
48 #define f_h_GPIO_Port GPIOB
49 #define g_h_Pin GPIO_PIN_6
50 #define g_h_GPIO_Port GPIOB
51 #define a_v_Pin GPIO_PIN_7
52 #define a_v_GPIO_Port GPIOB
53 #define b_v_Pin GPIO_PIN_8
54 #define b_v_GPIO_Port GPIOB
55 #define c_v_Pin GPIO_PIN_9
56 #define c_v_GPIO_Port GPIOB
57 /* USER CODE BEGIN Private defines */
58
59 /* USER CODE END Private defines */
60
61 #ifndef __cplusplus
```

2.3.2 main.c

Also in main.c:

```
1 static void MX_TIM2_Init(void)
2 {
3
4     /* USER CODE BEGIN TIM2_Init 0 */
5
6     /* USER CODE END TIM2_Init 0 */
7
8     TIM_ClockConfigTypeDef sClockSourceConfig = {0};
9     TIM_MasterConfigTypeDef sMasterConfig = {0};
10
11     /* USER CODE BEGIN TIM2_Init 1 */
12
13     /* USER CODE END TIM2_Init 1 */
14     htim2.Instance = TIM2;
15     htim2.Init.Prescaler = 7999;
16     htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
```



```
17 htim2.Init.Period = 9;
18 htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
19 htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
20 if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
21 {
22     Error_Handler();
23 }
24 sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
25 if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) !=
    HAL_OK)
26 {
27     Error_Handler();
28 }
29 sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
30 sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
31 if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &
    sMasterConfig) != HAL_OK)
32 {
33     Error_Handler();
34 }
35 /* USER CODE BEGIN TIM2_Init 2 */
36
37 /* USER CODE END TIM2_Init 2 */
38
39 }
40
41 /**
42  * @brief GPIO Initialization Function
43  * @param None
44  * @retval None
45  */
46 static void MX_GPIO_Init(void)
47 {
48     GPIO_InitTypeDef GPIO_InitStruct = {0};
49
50     /* GPIO Ports Clock Enable */
51     __HAL_RCC_GPIOA_CLK_ENABLE();
52     __HAL_RCC_GPIOB_CLK_ENABLE();
```

```
53
54  /*Configure GPIO pin Output Level */
55  HAL_GPIO_WritePin(GPIOA, red_h_Pin|yellow_h_Pin|green_h_Pin|
56      red_v_Pin
57      |yellow_v_Pin|green_v_Pin|en0_Pin|
58      en1_Pin
59      |en2_Pin|en3_Pin, GPIO_PIN_RESET);
60
61  /*Configure GPIO pin Output Level */
62  HAL_GPIO_WritePin(GPIOB, a_h_Pin|b_h_Pin|c_h_Pin|d_v_Pin
63      |e_v_Pin|f_v_Pin|g_v_Pin|d_h_Pin
64      |e_h_Pin|f_h_Pin|g_h_Pin|a_v_Pin
65      |b_v_Pin|c_v_Pin, GPIO_PIN_RESET);
66
67  /*Configure GPIO pins : red_h_Pin yellow_h_Pin green_h_Pin
68      red_v_Pin
69      yellow_v_Pin green_v_Pin en0_Pin
70      en1_Pin
71      en2_Pin en3_Pin */
72  GPIO_InitStruct.Pin = red_h_Pin|yellow_h_Pin|green_h_Pin|
73      red_v_Pin
74      |yellow_v_Pin|green_v_Pin|en0_Pin|
75      en1_Pin
76      |en2_Pin|en3_Pin;
77  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
78  GPIO_InitStruct.Pull = GPIO_NOPULL;
79  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
80  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
81
82  /*Configure GPIO pins : a_h_Pin b_h_Pin c_h_Pin d_v_Pin
83      e_v_Pin f_v_Pin g_v_Pin d_h_Pin
84      e_h_Pin f_h_Pin g_h_Pin a_v_Pin
85      b_v_Pin c_v_Pin */
86  GPIO_InitStruct.Pin = a_h_Pin|b_h_Pin|c_h_Pin|d_v_Pin
87      |e_v_Pin|f_v_Pin|g_v_Pin|d_h_Pin
88      |e_h_Pin|f_h_Pin|g_h_Pin|a_v_Pin
89      |b_v_Pin|c_v_Pin;
90  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
```




```
85  GPIO_InitStruct.Pull = GPIO_NOPULL;
86  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
87  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
88
89  /*Configure GPIO pins : sw0_Pin sw1_Pin sw2_Pin */
90  GPIO_InitStruct.Pin = sw0_Pin|sw1_Pin|sw2_Pin;
91  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
92  GPIO_InitStruct.Pull = GPIO_PULLUP;
93  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
94
95 }
```

2.4 Exercise 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.

The system uses separate timers for different functions, each of which can be adjusted independently:

```
1 void setTimer_TrafficLight(int duration) {
2     timer_traffic_light_counter = duration / 10;
3     timer_traffic_light_flag = 0;
4 }
5
6 void setTimer_7seg_scan(int duration) {
7     timer_7seg_scan_counter = duration / 10;
8     timer_7seg_scan_flag = 0;
9 }
10
11 void setTimer_Blink(int duration) {
12     timer_blink_counter = duration / 10;
13     timer_blink_flag = 0;
14 }
```

This approach allows changing the timer interrupt duration with minimal impact on the system, as each module can be adjusted independently

2.5 Exercise 5: 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,
- To add code for button debouncing,
- To add code for increasing mode when the first button is pressed.

2.5.1 Input reading

Button handling code is implemented in the input reading C file with debouncing functionality and long press detection:

```
1 void button_reading(void) {
2     for(int i = 0; i < NO_OF_BUTTONS; i++) {
3         debounceButtonBuffer2[i] = debounceButtonBuffer1[i];
4         debounceButtonBuffer1[i] = HAL_GPIO_ReadPin(button_ports[
5             i], button_pins[i]);
6         if (debounceButtonBuffer1[i] == debounceButtonBuffer2[i])
7             buttonBuffer[i] = debounceButtonBuffer1[i];
8
9         if (buttonBuffer[i] == BUTTON_IS_PRESSED) {
10             if (counterForButtonPress1s[i] <
11                 DURATION_FOR_AUTO_INCREASING) {
12                 counterForButtonPress1s[i]++;
13             } else {
14                 flagForButtonPress1s[i] = 1;
15             }
16         } else {
17             counterForButtonPress1s[i] = 0;
18             flagForButtonPress1s[i] = 0;
19         }
20     }
21 }
```

2.5.2 sw0 handling

Button sw0 to switch between modes is implemented in the input processing C file :

```
1 if (is_button_pressed(0) && !button0_flag) {
2     button0_flag = 1;
3     prev_mode = mode;
4     mode = (mode % 4) + 1;
5
6     switch(mode) {
7         case MODIFY_RED_MODE:
8             temp_value = time_red;
9             break;
10        case MODIFY_YELLOW_MODE:
11            temp_value = time_yellow;
12            break;
13        case MODIFY_GREEN_MODE:
14            temp_value = time_green;
15            break;
16    }
17
18    if (prev_mode == MODIFY_GREEN_MODE && mode == NORMAL_MODE) {
19        status_h = H_RED;
20        counter_h = time_red;
21        set_red_h(1); set_yellow_h(0); set_green_h(0);
22
23        status_v = V_GREEN;
24        counter_v = time_green;
25        set_red_v(0); set_yellow_v(0); set_green_v(1);
26
27        setTimer_TrafficLight(1000);
28    }
29 } else if (!is_button_pressed(0)) {
30     button0_flag = 0;
31 }
```

2.6 Exercise 6: Displaying modes

Your tasks in this exercise are:

- To add code for display mode on seven-segment LEDs, and
- To add code for blinking LEDs depending on the mode that is selected.

2.6.1 Mode handling

Display of the mode on 7-segment LEDs and controlling LED blinking depending on the mode in input processing C file:

```
1 switch(mode) {
2     case NORMAL_MODE:
3         updateLedBuffer(counter_h, counter_v);
4
5         if (timer_traffic_light_flag == 1) {
6             setTimer_TrafficLight(500); // set traffic light
7             timer here
8             counter_h--;
9             counter_v--;
10        }
11
12        switch(status_h) {
13            case H_RED:
14                if (counter_h <= 0) {
15                    status_h = H_GREEN;
16                    counter_h = time_green;
17                    set_red_h(0); set_green_h(1);
18                }
19                break;
20            case H_GREEN:
21                if (counter_h <= 0) {
22                    status_h = H_YELLOW;
23                    counter_h = time_yellow;
24                    set_green_h(0); set_yellow_h(1);
25                }
26                break;
27            case H_YELLOW:
28                if (counter_h <= 0) {
```



```
28         status_h = H_RED;
29         counter_h = time_red;
30         set_yellow_h(0); set_red_h(1);
31     }
32     break;
33 }
34
35 switch(status_v) {
36     case V_RED:
37         if (counter_v <= 0) {
38             status_v = V_GREEN;
39             counter_v = time_green;
40             set_red_v(0); set_green_v(1);
41         }
42         break;
43     case V_GREEN:
44         if (counter_v <= 0) {
45             status_v = V_YELLOW;
46             counter_v = time_yellow;
47             set_green_v(0); set_yellow_v(1);
48         }
49         break;
50     case V_YELLOW:
51         if (counter_v <= 0) {
52             status_v = V_RED;
53             counter_v = time_red;
54             set_yellow_v(0); set_red_v(1);
55         }
56         break;
57 }
58 break;
59
60 case MODIFY_RED_MODE:
61     led_buffer[2] = 0;
62     led_buffer[3] = 2;
63     led_buffer[0] = temp_value / 10;
64     led_buffer[1] = temp_value % 10;
65 }
```

```
66     if (timer_blink_flag == 1) {
67         setTimer_Blink(250);
68         blink_status = 1 - blink_status;
69
70         set_red_h(blink_status);
71         set_red_v(blink_status);
72         set_yellow_h(0);
73         set_yellow_v(0);
74         set_green_h(0);
75         set_green_v(0);
76     }
77     break;
78
79 case MODIFY_YELLOW_MODE:
80     led_buffer[2] = 0;
81     led_buffer[3] = 3;
82     led_buffer[0] = temp_value / 10;
83     led_buffer[1] = temp_value % 10;
84
85     if (timer_blink_flag == 1) {
86         setTimer_Blink(250);
87         blink_status = 1 - blink_status;
88
89         set_red_h(0);
90         set_red_v(0);
91         set_yellow_h(blink_status);
92         set_yellow_v(blink_status);
93         set_green_h(0);
94         set_green_v(0);
95     }
96     break;
97
98 case MODIFY_GREEN_MODE:
99     led_buffer[2] = 0;
100    led_buffer[3] = 4;
101    led_buffer[0] = temp_value / 10;
102    led_buffer[1] = temp_value % 10;
103
```



```
104     if (timer_blink_flag == 1) {
105         setTimer_Blink(250);
106         blink_status = 1 - blink_status;
107
108         set_red_h(0);
109         set_red_v(0);
110         set_yellow_h(0);
111         set_yellow_v(0);
112         set_green_h(blink_status);
113         set_green_v(blink_status);
114     }
115     break;
116 }
```

2.6.2 Display led

7-segment LEDs are scanned in two stages for simultaneous display in led display C file:

```
1 static int scan_stage = 0;
2 void scan7SEG() {
3     if (timer_7seg_scan_flag == 1) {
4         setTimer_7seg_scan(100); // set led scan here
5
6         if (scan_stage == 0) {
7             HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, GPIO_PIN_SET);
8             HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, GPIO_PIN_SET);
9
10            display7SEG_H(led_buffer[0]);
11            display7SEG_V(led_buffer[2]);
12
13            HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, GPIO_PIN_RESET)
14                ;
15            HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, GPIO_PIN_RESET)
16                ;
17
18            scan_stage = 1;
19        } else {
```

```
18     HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, GPIO_PIN_SET);
19     HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, GPIO_PIN_SET);
20
21     display7SEG_H(led_buffer[1]);
22     display7SEG_V(led_buffer[3]);
23
24     HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, GPIO_PIN_RESET)
25         ;
26     HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, GPIO_PIN_RESET)
27         ;
28     scan_stage = 0;
29 }
```

2.7 Exercise 7,8,9: Increasing time duration value for the red, amber and green LEDs

2.7.1 sw1 handling

Button sw1 is used to increase the time value, with auto-increment functionality when held:

```
1 if (is_button_pressed(1)) {
2     if (!button1_flag) {
3         button1_flag = 1;
4
5         if (mode != NORMAL_MODE) {
6             temp_value++;
7             if (temp_value > 99) temp_value = 1;
8         }
9     }
10
11     if (is_button_pressed_1s(1)) {
12         auto_increase_counter++;
13         if (auto_increase_counter >= 2500) {
14             auto_increase_counter = 0;
15             if (mode != NORMAL_MODE) {
```




```
16         temp_value++;
17         if (temp_value > 99) temp_value = 1;
18     }
19 }
20 }
21 } else {
22     button1_flag = 0;
23     auto_increase_counter = 0;
24 }
```

2.7.2 sw2 handling

Button sw2 confirms the value and maintains constraints between the lights:

```
1 if (is_button_pressed(2) && !button2_flag) {
2     button2_flag = 1;
3
4     if (mode != NORMAL_MODE) {
5         switch(mode) {
6             case MODIFY_RED_MODE:
7                 time_red = temp_value;
8                 if (time_green + time_yellow != time_red) {
9                     int diff = time_red - time_yellow;
10
11                     if (diff > 1) {
12                         time_green = diff;
13                     } else {
14                         time_green = 1;
15                         time_yellow = time_red - time_green;
16                     }
17                 }
18                 break;
19
20             case MODIFY_YELLOW_MODE:
21                 time_yellow = temp_value;
22                 if (time_green + time_yellow != time_red) {
23                     time_green = time_red - time_yellow;
24                     if (time_green < 1) {
25                         time_green = 1;
```

```
26         time_red = time_green + time_yellow;
27     }
28 }
29 break;
30
31 case MODIFY_GREEN_MODE:
32     time_green = temp_value;
33     if (time_green + time_yellow != time_red) {
34         time_yellow = time_red - time_green;
35         if (time_yellow < 1) {
36             time_yellow = 1;
37             time_red = time_green + time_yellow;
38         }
39     }
40     break;
41 }
42 }
43 } else if (!is_button_pressed(2)) {
44     button2_flag = 0;
45 }
```

2.8 Exercise 10: To finish the project

Your tasks in this exercise are:

- To integrate all the previous tasks to one final project ✓
- To create a video to show all features in the specification ✓
- To add a report to describe your solution for each exercise ✓
- To submit your report and code on the BKeL ✓