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

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.

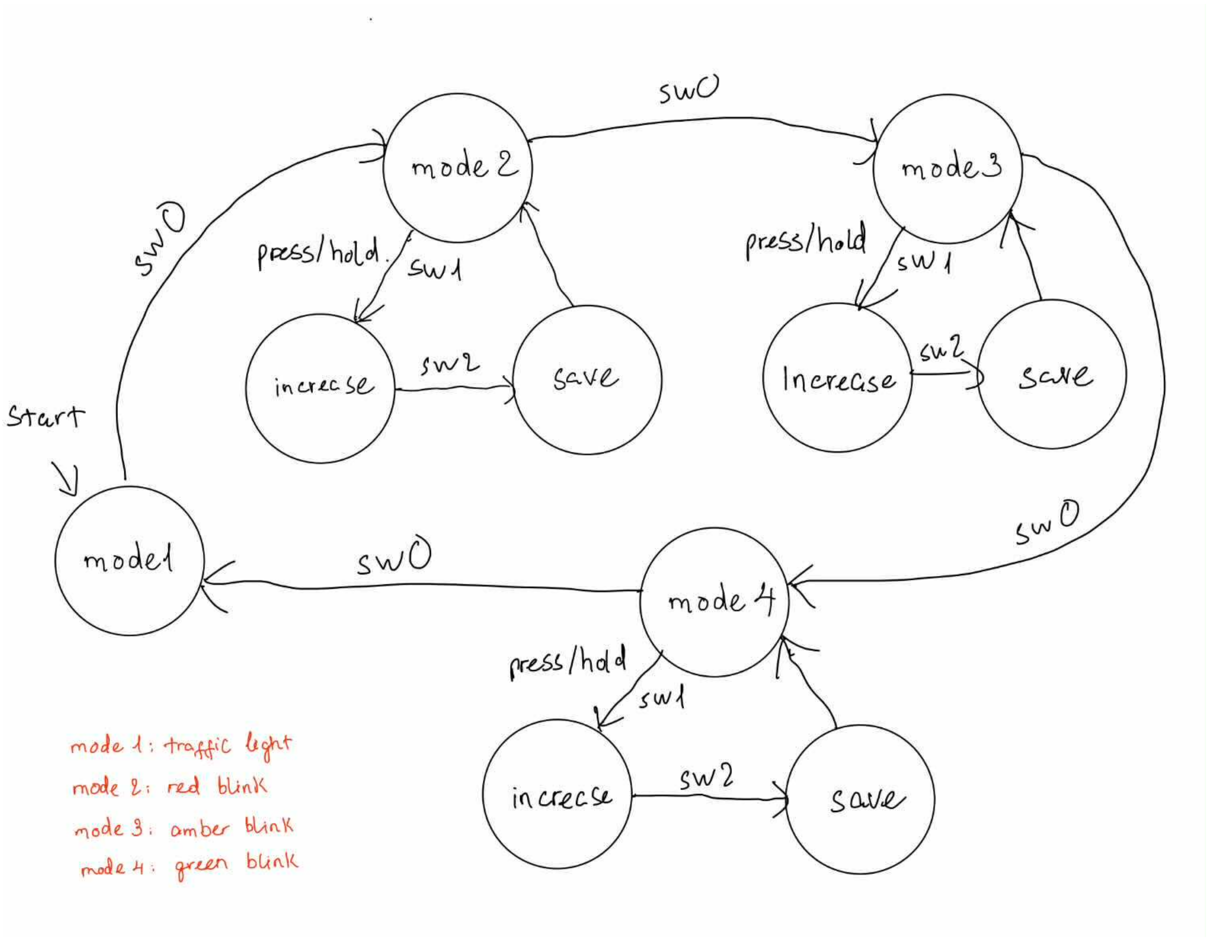

Figure 2.1: FSM Sketch

## 2.2 Exercise 2: Proteus Schematic

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

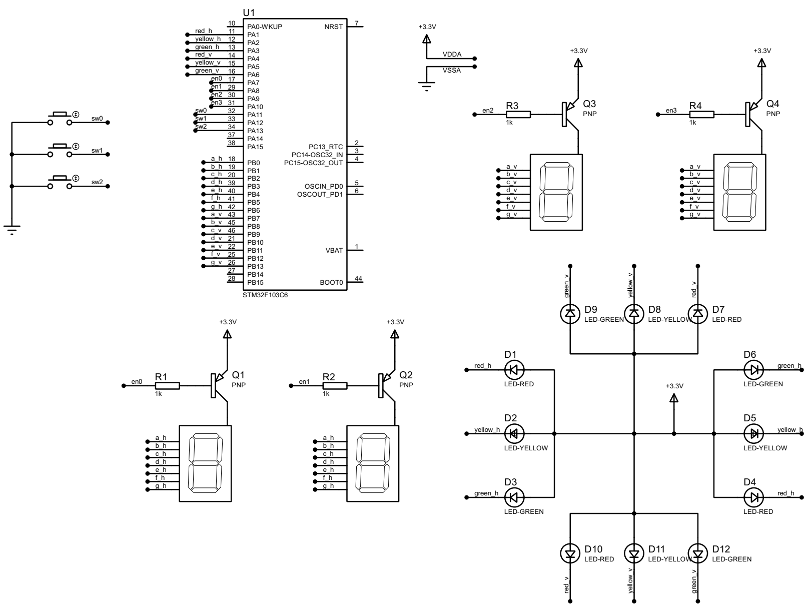

Figure 2.2: Lab3 schematic

## 2.3 Exercise 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.

### 2.3.1 main.h

Pin defined in main.h:

```
void Error_Handler(void);

#define red_h_Pin GPIO_PIN_1
#define red_h_GPIO_Port GPIOA
#define yellow_h_Pin GPIO_PIN_2
```

```
#define yellow_h_GPIO_Port GPIOA
#define green_h_Pin GPIO_PIN_3
#define green_h_GPIO_Port GPIOA
#define red_v_Pin GPIO_PIN_4
#define red_v_GPIO_Port GPIOA
#define yellow_v_Pin GPIO_PIN_5
#define yellow_v_GPIO_Port GPIOA
#define green_v_Pin GPIO_PIN_6
#define green_v_GPIO_Port GPIOA
#define en0_Pin GPIO_PIN_7
#define en0_GPIO_Port GPIOA
#define a_h_Pin GPIO_PIN_0
#define a_h_GPIO_Port GPIOB
#define b_h_Pin GPIO_PIN_1
#define b_h_GPIO_Port GPIOB
#define c_h_Pin GPIO_PIN_2
#define c_h_GPIO_Port GPIOB
#define d_v_Pin GPIO_PIN_10
#define d_v_GPIO_Port GPIOB
#define e_v_Pin GPIO_PIN_11
#define e_v_GPIO_Port GPIOB
#define f_v_Pin GPIO_PIN_12
#define f_v_GPIO_Port GPIOB
#define g_v_Pin GPIO_PIN_13
#define g_v_GPIO_Port GPIOB
#define en1_Pin GPIO_PIN_8
#define en1_GPIO_Port GPIOA
#define en2_Pin GPIO_PIN_9
#define en2_GPIO_Port GPIOA
#define en3_Pin GPIO_PIN_10
#define en3_GPIO_Port GPIOA
#define sw0_Pin GPIO_PIN_11
#define sw0_GPIO_Port GPIOA
#define sw1_Pin GPIO_PIN_12
#define sw1_GPIO_Port GPIOA
#define sw2_Pin GPIO_PIN_13
#define sw2_GPIO_Port GPIOA
#define d_h_Pin GPIO_PIN_3
```

```
#define d_h_GPIO_Port GPIOB
#define e_h_Pin GPIO_PIN_4
#define e_h_GPIO_Port GPIOB
#define f_h_Pin GPIO_PIN_5
#define f_h_GPIO_Port GPIOB
#define g_h_Pin GPIO_PIN_6
#define g_h_GPIO_Port GPIOB
#define a_v_Pin GPIO_PIN_7
#define a_v_GPIO_Port GPIOB
#define b_v_Pin GPIO_PIN_8
#define b_v_GPIO_Port GPIOB
#define c_v_Pin GPIO_PIN_9
#define c_v_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

#ifdef __cplusplus
```

### 2.3.2 main.c

Also in main.c:

```
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 7999;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
```

```c
  htim2.Init.Period = 9;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) !=
     HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &
     sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */

  /* USER CODE END TIM2_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
```

```

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, red_h_Pin|yellow_h_Pin|green_h_Pin|red_v_Pin
                        |yellow_v_Pin|green_v_Pin|en0_Pin|en1_Pin
                        |en2_Pin|en3_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOB, 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_PIN_RESET);

/*Configure GPIO pins : red_h_Pin yellow_h_Pin green_h_Pin red_v_Pin
                         yellow_v_Pin green_v_Pin en0_Pin en1_Pin
                         en2_Pin en3_Pin */
GPIO_InitStruct.Pin = red_h_Pin|yellow_h_Pin|green_h_Pin|red_v_Pin
                        |yellow_v_Pin|green_v_Pin|en0_Pin|en1_Pin
                        |en2_Pin|en3_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pins : 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.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);

  /*Configure GPIO pins : sw0_Pin sw1_Pin sw2_Pin */
  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);

}
```

## 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:

```
void setTimer_TrafficLight(int duration) {
    timer_traffic_light_counter = duration / 10;
    timer_traffic_light_flag = 0;
}

void setTimer_7seg_scan(int duration) {
    timer_7seg_scan_counter = duration / 10;
    timer_7seg_scan_flag = 0;
}

void setTimer_Blink(int duration) {
    timer_blink_counter = duration / 10;
    timer_blink_flag = 0;
}
```

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:

```c
void button_reading(void) {
    for(int i = 0; i < NO_OF_BUTTONS; i++) {
        debounceButtonBuffer2[i] = debounceButtonBuffer1[i];
        debounceButtonBuffer1[i] = HAL_GPIO_ReadPin(button_ports[
           i], button_pins[i]);
        if (debounceButtonBuffer1[i] == debounceButtonBuffer2[i])
        buttonBuffer[i] = debounceButtonBuffer1[i];

        if (buttonBuffer[i] == BUTTON_IS_PRESSED) {
            if (counterForButtonPress1s[i] <
               DURATION_FOR_AUTO_INCREASING) {
                counterForButtonPress1s[i]++;
            } else {
                flagForButtonPress1s[i] = 1;
            }
        } else {
            counterForButtonPress1s[i] = 0;
            flagForButtonPress1s[i] = 0;
        }
    }
}
```

### 2.5.2 sw0 handling

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

```c
if (is_button_pressed(0) && !button0_flag) {
    button0_flag = 1;
    prev_mode = mode;
    mode = (mode % 4) + 1;

    switch(mode) {
        case MODIFY_RED_MODE:
            temp_value = time_red;
            break;
        case MODIFY_YELLOW_MODE:
            temp_value = time_yellow;
            break;
        case MODIFY_GREEN_MODE:
            temp_value = time_green;
            break;
    }

    if (prev_mode == MODIFY_GREEN_MODE && mode == NORMAL_MODE) {
        status_h = H_RED;
        counter_h = time_red;
        set_red_h(1); set_yellow_h(0); set_green_h(0);

        status_v = V_GREEN;
        counter_v = time_green;
        set_red_v(0); set_yellow_v(0); set_green_v(1);

        setTimer_TrafficLight(1000);
    }
} else if (!is_button_pressed(0)) {
    button0_flag = 0;
}
```

## 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:

```c
switch(mode) {
    case NORMAL_MODE:
        updateLedBuffer(counter_h, counter_v);

        if (timer_traffic_light_flag == 1) {
            setTimer_TrafficLight(500); // set traffic light
               timer here
            counter_h--;
            counter_v--;
        }

        switch(status_h) {
            case H_RED:
                if (counter_h <= 0) {
                    status_h = H_GREEN;
                    counter_h = time_green;
                    set_red_h(0); set_green_h(1);
                }
                break;
            case H_GREEN:
                if (counter_h <= 0) {
                    status_h = H_YELLOW;
                    counter_h = time_yellow;
                    set_green_h(0); set_yellow_h(1);
                }
                break;
            case H_YELLOW:
                if (counter_h <= 0) {
                    status_h = H_RED;
                    counter_h = time_red;
                    set_yellow_h(0); set_red_h(1);
                }
                break;
        }

        switch(status_v) {
            case V_RED:
                if (counter_v <= 0) {
                    status_v = V_GREEN;
                    counter_v = time_green;
                    set_red_v(0); set_green_v(1);
                }
                break;
            case V_GREEN:
                if (counter_v <= 0) {
                    status_v = V_YELLOW;
                    counter_v = time_yellow;
                    set_green_v(0); set_yellow_v(1);
                }
                break;
            case V_YELLOW:
                if (counter_v <= 0) {
                    status_v = V_RED;
                    counter_v = time_red;
                    set_yellow_v(0); set_red_v(1);
                }
                break;
        }
        break;

    case MODIFY_RED_MODE:
        led_buffer[2] = 0;
        led_buffer[3] = 2;
        led_buffer[0] = temp_value / 10;
        led_buffer[1] = temp_value % 10;

        if (timer_blink_flag == 1) {
            setTimer_Blink(250);
            blink_status = 1 - blink_status;

            set_red_h(blink_status);
            set_red_v(blink_status);
            set_yellow_h(0);
            set_yellow_v(0);
            set_green_h(0);
            set_green_v(0);
        }
        break;

    case MODIFY_YELLOW_MODE:
        led_buffer[2] = 0;
        led_buffer[3] = 3;
        led_buffer[0] = temp_value / 10;
        led_buffer[1] = temp_value % 10;

        if (timer_blink_flag == 1) {
            setTimer_Blink(250);
            blink_status = 1 - blink_status;

            set_red_h(0);
            set_red_v(0);
            set_yellow_h(blink_status);
            set_yellow_v(blink_status);
            set_green_h(0);
            set_green_v(0);
        }
        break;

    case MODIFY_GREEN_MODE:
        led_buffer[2] = 0;
        led_buffer[3] = 4;
        led_buffer[0] = temp_value / 10;
        led_buffer[1] = temp_value % 10;

        if (timer_blink_flag == 1) {
            setTimer_Blink(250);
            blink_status = 1 - blink_status;

            set_red_h(0);
            set_red_v(0);
            set_yellow_h(0);
            set_yellow_v(0);
            set_green_h(blink_status);
            set_green_v(blink_status);
        }
        break;
}
```

### 2.6.2 Display led

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

```
static int scan_stage = 0;
void scan7SEG() {
if (timer_7seg_scan_flag == 1) {
    setTimer_7seg_scan(100); // set led scan here

    if (scan_stage == 0) {
        HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, GPIO_PIN_SET);

        display7SEG_H(led_buffer[0]);
        display7SEG_V(led_buffer[2]);

        HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, GPIO_PIN_RESET)
            ;
        HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, GPIO_PIN_RESET)
            ;

        scan_stage = 1;
    } else {
        HAL_GPIO_WritePin(en0_GPIO_Port, en0_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(en2_GPIO_Port, en2_Pin, GPIO_PIN_SET);

        display7SEG_H(led_buffer[1]);
        display7SEG_V(led_buffer[3]);

        HAL_GPIO_WritePin(en1_GPIO_Port, en1_Pin, GPIO_PIN_RESET)
            ;
        HAL_GPIO_WritePin(en3_GPIO_Port, en3_Pin, GPIO_PIN_RESET)
            ;

        scan_stage = 0;
    }
}
```

## 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:

```
if (is_button_pressed(1)) {
    if (!button1_flag) {
        button1_flag = 1;

        if (mode != NORMAL_MODE) {
            temp_value++;
            if (temp_value > 99) temp_value = 1;
        }
    }

    if (is_button_pressed_1s(1)) {
        auto_increase_counter++;
        if (auto_increase_counter >= 2500) {
            auto_increase_counter = 0;
            if (mode != NORMAL_MODE) {
                temp_value++;
                if (temp_value > 99) temp_value = 1;
            }
        }
    }
} else {
    button1_flag = 0;
    auto_increase_counter = 0;
}
```

### 2.7.2 sw2 handling

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

```
if (is_button_pressed(2) && !button2_flag) {
    button2_flag = 1;

    if (mode != NORMAL_MODE) {
        switch(mode) {
            case MODIFY_RED_MODE:
                time_red = temp_value;
                if (time_green + time_yellow != time_red) {
                    int diff = time_red - time_yellow;

                    if (diff > 1) {
                        time_green = diff;
                    } else {
                        time_green = 1;
                        time_yellow = time_red - time_green;
                    }
                }
                break;

            case MODIFY_YELLOW_MODE:
                time_yellow = temp_value;
                if (time_green + time_yellow != time_red) {
                    time_green = time_red - time_yellow;
                    if (time_green < 1) {
                        time_green = 1;
                        time_red = time_green + time_yellow;
                    }
                }
                break;

            case MODIFY_GREEN_MODE:
                time_green = temp_value;
                if (time_green + time_yellow != time_red) {
                    time_yellow = time_red - time_green;
                    if (time_yellow < 1) {
                        time_yellow = 1;
                        time_red = time_green + time_yellow;
                    }
                }
                break;
        }
    }
} else if (!is_button_pressed(2)) {
    button2_flag = 0;
}
```

## 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 ✓