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 LÍ - VI ĐIỀU KHIỂN (CO 3010)

BÁO CÁO LAB 3

**Giảng viên: PHAN VĂN SỸ**

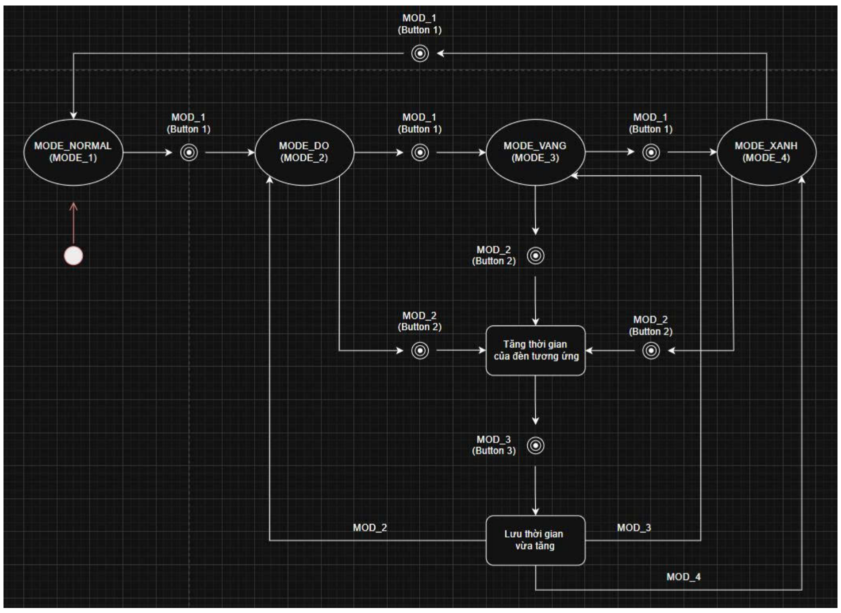
| **Họ và tên** | **Mã số sinh viên** |
|---|---|
| Nguyễn Văn Sang | 2212912 |

TP.Hồ Chí Minh, 05/11/2025

# Mục lục

## 1 Bài 1 Sketch an FSM

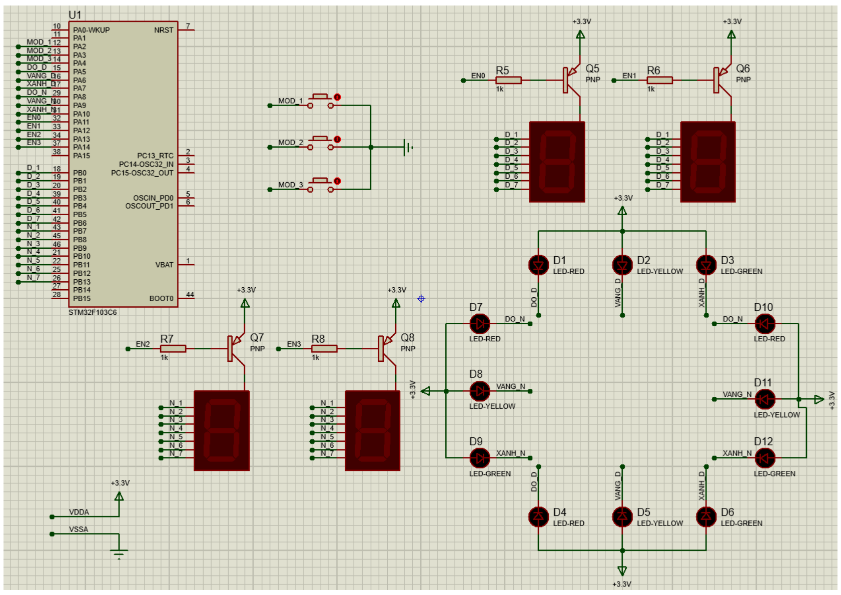
Your task in this exercise is to sketch an FSM that describes your idea of how to solve the problem.



Hình 1: FSM state diagram

## 2 Bài 2 Proteus Schematic

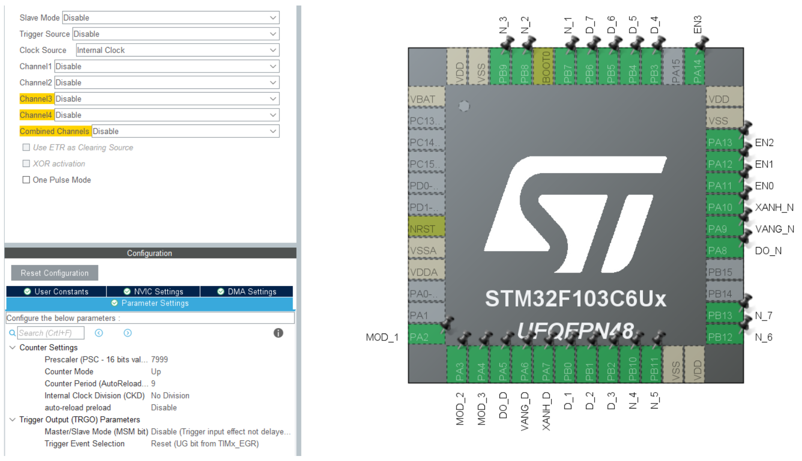
Your task in this exercise is to draw a Proteus schematic for the problem above.



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.


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
Configure the below parameters :
Counter Settings
Prescaler (PSC - 16 bits val... 7999
Counter Mode Up
Counter Period (AutoReload... 9
Internal Clock Division (CKD) No Division
auto-reload preload Disable
Trigger Output (TRGO) Parameters
Master/Slave Mode (MSM bit) Disable (Trigger input effect not delaye...
Trigger Event Selection Reset (UG bit from TIMx_EGR)

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.

```
// ----- software_timer.h ----- //
#ifndef TIMER_TICK_MS
#define TIMER_TICK_MS 10
#endif

// ----- main.c ----- //
htim2.Init.Prescaler = 7999;
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 $T = \frac{1}{\frac{8.10^6}{(Prescaler+1)(Period+1)}} = \frac{1}{\frac{8.10^6}{(Prescaler+1)(TIMER\_TICK\_MS)}}(s)$ 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.

```
// ----- button.h ----- //
#ifndef INC_BUTTON_H_
#define INC_BUTTON_H_

#include "main.h"

#ifndef NORMAL_STATE
#define NORMAL_STATE GPIO_PIN_SET
#endif
#ifndef ACTIVE_STATE
#define ACTIVE_STATE GPIO_PIN_RESET
#endif

extern int MOD_1_ACTIVE;
int check_MOD_1();

extern int MOD_2_ACTIVE;
int check_MOD_2();

extern int MOD_3_ACTIVE;
int check_MOD_3();

int check_MOD_1_long();
void reset_MOD_flags();
void process_input();

```

```c
#endif /* INC_BUTTON_H_ */

// ----- button.c ----- //
#include "button.h"
#include "software_timer.h"

#ifndef AUTO_REPEAT_MS
#define AUTO_REPEAT_MS 100
#endif

static int Tmp_0 = NORMAL_STATE;
static int Tmp_1 = NORMAL_STATE;
static int Tmp_2 = NORMAL_STATE;
static int Tmp_3 = NORMAL_STATE;

static int MOD_1_state = NORMAL_STATE;
int MOD_1_ACTIVE = 0;

static int MOD_2_state = NORMAL_STATE;
int MOD_2_ACTIVE = 0;

static int MOD_3_state = NORMAL_STATE;
int MOD_3_ACTIVE = 0;

static int KeyTimeout = AUTO_REPEAT_MS / TIMER_TICK_MS;
static int MOD_1_long = 0;

int check_MOD_1() {
    if (MOD_1_ACTIVE) {
        MOD_1_ACTIVE = 0;
        return 1;
    }
    return 0;
}

int check_MOD_2() {
    if (MOD_2_ACTIVE) {
        MOD_2_ACTIVE = 0;
        return 1;
    }
    return 0;
}

int check_MOD_3() {
    if (MOD_3_ACTIVE) {
        MOD_3_ACTIVE = 0;
        return 1;
    }
    return 0;
}

int check_MOD_1_long() {
    if (MOD_1_long) {
        MOD_1_long = 0;
        return 1;
    }
    return 0;
}

void reset_MOD_flags() {
    MOD_1_ACTIVE = 0;
    MOD_2_ACTIVE = 0;
    MOD_3_ACTIVE = 0;
}

static void sample_inputs(void) {
    Tmp_2 = Tmp_1;
    Tmp_1 = Tmp_0;
    MOD_1_state = HAL_GPIO_ReadPin(MOD_1_GPIO_Port, MOD_1_Pin);
    MOD_2_state = HAL_GPIO_ReadPin(MOD_2_GPIO_Port, MOD_2_Pin);
    MOD_3_state = HAL_GPIO_ReadPin(MOD_3_GPIO_Port, MOD_3_Pin);
    Tmp_0 = MOD_1_state && MOD_2_state && MOD_3_state;
}

static void set_active_for_current_mod(void) {
    if (MOD_1_state == ACTIVE_STATE) {
        MOD_1_ACTIVE = 1;
    } else if (MOD_2_state == ACTIVE_STATE) {
        MOD_2_ACTIVE = 1;
    } else if (MOD_3_state == ACTIVE_STATE) {
        MOD_3_ACTIVE = 1;
    }
}

void process_input() {
    sample_inputs();
    if ((Tmp_1 == Tmp_0) && (Tmp_1 == Tmp_2)) {
        if (Tmp_2 != Tmp_3) {
            Tmp_3 = Tmp_2;
            if (Tmp_3 == ACTIVE_STATE) {
                KeyTimeout = AUTO_REPEAT_MS;
                set_active_for_current_mod();
            }
        } else {
            KeyTimeout--;
            if (KeyTimeout == 0) {
                KeyTimeout = AUTO_REPEAT_MS;
                if (Tmp_3 == ACTIVE_STATE) {
                    set_active_for_current_mod();
                }
            }
        }
    }
}

// ----- global.h ----- //
#ifndef INC_GLOBAL_H_
#define INC_GLOBAL_H_

#include "software_timer.h"
#include "button.h"

#define MODE_NORMAL 1
#define MODE_DO     2
#define MODE_VANG   3
#define MODE_XANH   4

extern int MOD_DO_TIMER;
extern int DO_duration;

extern int MOD_VANG_TIMER;
extern int VANG_duration;

extern int MOD_XANH_TIMER;
```

```c
extern int XANH_duration;

void apply_changes(int MODE_CURRENT);
void init_traffic_timers(int MODE_CURRENT);
void reset_to_defaults(int MODE_CURRENT);
void run_traffic_lights(void);
void configure_leds(int MODE_CURRENT);
void show_7seg(int value, uint32_t pin);
void show_mode(int value);
void update_buffer_mode(int value);
void update_buffer_time(int value);
void scan_7seg(int idx);
void show_time(int value);
void verify_timers(void);

#endif /* INC_GLOBAL_H_ */

// ----- global.c ----- //
#include "global.h"

#ifndef SCAN_INTERVAL_MS
#define SCAN_INTERVAL_MS 125
#endif

#ifndef LIGHT_STEP_MS
#define LIGHT_STEP_MS 400
#endif

int buf_mode[2] = {1, 2};
int buf_time[2] = {3, 4};

int DO_duration = 10;
int MOD_DO_TIMER = 10;
int DO_modified = 0;

int VANG_duration = 3;
int MOD_VANG_TIMER = 3;
int VANG_modified = 0;

int XANH_duration = 7;
int MOD_XANH_TIMER = 7;
int XANH_modified = 0;

int idx_mode = 0;
int idx_time = 2;

int direction = 1;

void show_7seg(int value, uint32_t pin) {
    if(value < 0 || value > 9)
        return;
    char seg_patterns[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0
        x80, 0x90};
    for (int j = 0; j < 7; j++) {
        HAL_GPIO_WritePin(GPIOB, pin << j, (seg_patterns[value] >> j) & 1);
    }
}

void scan_7seg(int idx) {
    switch (idx) {
        case 0:
            HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, GPIO_PIN_SET);
            show_7seg(buf_mode[0], GPIO_PIN_0);
            HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, GPIO_PIN_RESET);
            break;
        case 1:
            HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, GPIO_PIN_SET);
            show_7seg(buf_mode[1], GPIO_PIN_0);
            HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, GPIO_PIN_RESET);
            break;
        case 2:
            HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, GPIO_PIN_SET);
            show_7seg(buf_time[0], GPIO_PIN_7);
            HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, GPIO_PIN_RESET);
            break;
        case 3:
            HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, GPIO_PIN_SET);
            show_7seg(buf_time[1], GPIO_PIN_7);
            HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, GPIO_PIN_RESET);
            break;
        default:
            break;
    }
}

void update_buffer_mode(int value) {
    if (value >= 0) {
        buf_mode[0] = value / 10;
        buf_mode[1] = value % 10;
    }
}

void update_buffer_time(int value) {
    if (value >= 0) {
        buf_time[0] = value / 10;
        buf_time[1] = value % 10;
    }
}

void show_mode(int value) {
    update_buffer_mode(value);
    if (tmr3_done == 1) {
        scan_7seg(idx_mode);
        idx_mode = (idx_mode + 1) % 2;
        init_tmr3(SCAN_INTERVAL_MS);
    }
}

void show_time(int value) {
    update_buffer_time(value);
    if (tmr4_done == 1) {
        scan_7seg(idx_time);
        idx_time++;
        if (idx_time > 3) {
            idx_time = 2;
        }
        init_tmr4(SCAN_INTERVAL_MS);
    }
}

void verify_timers() {
    if (DO_modified == 0) {
        MOD_DO_TIMER = DO_duration;
    } else {
```

```c
        if (VANG_modified == 0) {
            MOD_VANG_TIMER = VANG_duration;
        } else {
            if (XANH_modified == 0) {
                MOD_XANH_TIMER = XANH_duration;
            }
        }
    }
    if (DO_modified == 1 || VANG_modified == 1 || XANH_modified == 1) {
        if (MOD_DO_TIMER == (MOD_VANG_TIMER + MOD_XANH_TIMER)) {
            DO_duration = MOD_DO_TIMER;
            VANG_duration = MOD_VANG_TIMER;
            XANH_duration = MOD_XANH_TIMER;
            DO_modified = 0;
            VANG_modified = 0;
            XANH_modified = 0;
        }
    }
}

void reset_to_defaults(int MODE_CURRENT) {
    switch (MODE_CURRENT) {
        case MODE_NORMAL:
            verify_timers();
            MOD_DO_TIMER = DO_duration;
            MOD_VANG_TIMER = VANG_duration;
            MOD_XANH_TIMER = XANH_duration;
            break;
        case MODE_DO:
            MOD_DO_TIMER = DO_duration;
            break;
        case MODE_VANG:
            MOD_VANG_TIMER = VANG_duration;
            break;
        case MODE_XANH:
            MOD_XANH_TIMER = XANH_duration;
            break;
        default:
            break;
    }
}

void init_traffic_timers(int MODE_CURRENT) {
    switch (MODE_CURRENT) {
        case MODE_DO:
            MOD_DO_TIMER = (MOD_DO_TIMER % 99) + 1;
            break;
        case MODE_VANG:
            MOD_VANG_TIMER = (MOD_VANG_TIMER % 99) + 1;
            break;
        case MODE_XANH:
            MOD_XANH_TIMER = (MOD_XANH_TIMER % 99) + 1;
            break;
        default:
            break;
    }
}

void configure_leds(int MODE_CURRENT) {
    if (tmr2_done == 1) {
        if (MODE_CURRENT == MODE_DO) {
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5|GPIO_PIN_8);
        } else if (MODE_CURRENT == MODE_VANG) {
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_6|GPIO_PIN_9);
        } else if (MODE_CURRENT == MODE_XANH) {
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7|GPIO_PIN_10);
        }
        init_tmr2(250);
    }
}

static inline void config_north(GPIO_PinState r, GPIO_PinState y,
    GPIO_PinState g) {
    HAL_GPIO_WritePin(DO_N_GPIO_Port, DO_N_Pin, r);
    HAL_GPIO_WritePin(VANG_N_GPIO_Port, VANG_N_Pin, y);
    HAL_GPIO_WritePin(XANH_N_GPIO_Port, XANH_N_Pin, g);
}

static inline void config_south(GPIO_PinState r, GPIO_PinState y,
    GPIO_PinState g) {
    HAL_GPIO_WritePin(DO_D_GPIO_Port, DO_D_Pin, r);
    HAL_GPIO_WritePin(VANG_D_GPIO_Port, VANG_D_Pin, y);
    HAL_GPIO_WritePin(XANH_D_GPIO_Port, XANH_D_Pin, g);
}

void run_traffic_lights() {
    if (tmr1_done == 1) {
        if (direction == 1) {
            if (MOD_DO_TIMER > 0) {
                update_buffer_time(MOD_DO_TIMER);
                if (MOD_DO_TIMER > VANG_duration) {
                    if (MOD_XANH_TIMER > 0) {
                        config_north(GPIO_PIN_RESET, GPIO_PIN_SET,
                            GPIO_PIN_SET);
                        config_south(GPIO_PIN_SET, GPIO_PIN_SET,
                            GPIO_PIN_RESET);
                        update_buffer_mode(MOD_XANH_TIMER);
                        MOD_XANH_TIMER -= 1;
                    }
                } else {
                    if (MOD_DO_TIMER <= VANG_duration && MOD_DO_TIMER >= 1) {
                        if (MOD_VANG_TIMER > 0) {
                            config_north(GPIO_PIN_RESET, GPIO_PIN_SET,
                                GPIO_PIN_SET);
                            config_south(GPIO_PIN_SET, GPIO_PIN_RESET,
                                GPIO_PIN_SET);
                            update_buffer_mode(MOD_VANG_TIMER);
                            MOD_VANG_TIMER -= 1;
                        }
                    }
                }
                MOD_DO_TIMER -= 1;
            }
            if (MOD_VANG_TIMER == 0) {
                MOD_VANG_TIMER = VANG_duration;
            }
            if (MOD_XANH_TIMER == 0) {
                MOD_XANH_TIMER = XANH_duration;
            }
            if (MOD_DO_TIMER == 0) {
                MOD_DO_TIMER = DO_duration;
                direction = 2;
            }
        } else {
            if (MOD_DO_TIMER > 0) {
                update_buffer_mode(MOD_DO_TIMER);
                if (MOD_DO_TIMER > VANG_duration) {
                    if (MOD_XANH_TIMER > 0) {
                        config_south(GPIO_PIN_RESET, GPIO_PIN_SET,
                            GPIO_PIN_SET);
                        config_north(GPIO_PIN_SET, GPIO_PIN_SET,
                            GPIO_PIN_RESET);
                        update_buffer_time(MOD_XANH_TIMER);
                        MOD_XANH_TIMER -= 1;
                    }
                } else {
                    if (MOD_DO_TIMER <= VANG_duration && MOD_DO_TIMER >= 1) {
                        if (MOD_VANG_TIMER > 0) {
                            config_south(GPIO_PIN_RESET, GPIO_PIN_SET,
                                GPIO_PIN_SET);
                            config_north(GPIO_PIN_SET, GPIO_PIN_RESET,
                                GPIO_PIN_SET);
                            update_buffer_time(MOD_VANG_TIMER);
                            MOD_VANG_TIMER -= 1;
                        }
                    }
                }
                MOD_DO_TIMER -= 1;
            }
            if (MOD_VANG_TIMER == 0) {
                MOD_VANG_TIMER = VANG_duration;
            }
            if (MOD_XANH_TIMER == 0) {
                MOD_XANH_TIMER = XANH_duration;
            }
            if (MOD_DO_TIMER == 0) {
                MOD_DO_TIMER = DO_duration;
                direction = 1;
            }
        }
        init_tmr1(LIGHT_STEP_MS);
    }
    if (tmr5_done == 1) {
        scan_7seg(idx_mode);
        idx_mode = (idx_mode + 1) % 2;
        scan_7seg(idx_time);
        idx_time++;
        if (idx_time > 3) idx_time = 2;
        init_tmr5(SCAN_INTERVAL_MS);
    }
}

void apply_changes(int MODE_CURRENT) {
    switch (MODE_CURRENT) {
        case MODE_DO:
            DO_modified = 1;
            break;
        case MODE_VANG:
            VANG_modified = 1;
            break;
        case MODE_XANH:
            XANH_modified = 1;
            break;
        default:
            break;
    }
```

```
}
```

- To add code for button debouncing.

```c
// ----- button.c ----- //
static void sample_inputs(void) {
    Tmp_2 = Tmp_1;
    Tmp_1 = Tmp_0;
    MOD_1_state = HAL_GPIO_ReadPin(MOD_1_GPIO_Port, MOD_1_Pin);
    MOD_2_state = HAL_GPIO_ReadPin(MOD_2_GPIO_Port, MOD_2_Pin);
    MOD_3_state = HAL_GPIO_ReadPin(MOD_3_GPIO_Port, MOD_3_Pin);
    Tmp_0 = MOD_1_state && MOD_2_state && MOD_3_state;
}

static void set_active_for_current_mod(void) {
    if (MOD_1_state == ACTIVE_STATE) {
        MOD_1_ACTIVE = 1;
    } else if (MOD_2_state == ACTIVE_STATE) {
        MOD_2_ACTIVE = 1;
    } else if (MOD_3_state == ACTIVE_STATE) {
        MOD_3_ACTIVE = 1;
    }
}

void process_input() {
    sample_inputs();
    if ((Tmp_1 == Tmp_0) && (Tmp_1 == Tmp_2)) {
        if (Tmp_2 != Tmp_3) {
            Tmp_3 = Tmp_2;
            if (Tmp_3 == ACTIVE_STATE) {
                KeyTimeout = AUTO_REPEAT_MS;
                set_active_for_current_mod();
            }
        } else {
            KeyTimeout--;
            if (KeyTimeout == 0) {
                KeyTimeout = AUTO_REPEAT_MS;
                if (Tmp_3 == ACTIVE_STATE) {
                    set_active_for_current_mod();
                }
            }
        }
    }
}
```

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

```c
// ----- fsm.c ----- //
int MODE_CURRENT = 1;

static void handle_mode() {
    switch (MODE_CURRENT) {
        case MODE_NORMAL:
            run_traffic_lights();
            break;
        case MODE_DO:
            configure_and_display_mode(MODE_DO, MOD_DO_TIMER);
            break;
        case MODE_VANG:
            configure_and_display_mode(MODE_VANG, MOD_VANG_TIMER);
            break;
        case MODE_XANH:
                configure_and_display_mode(MODE_XANH, MOD_XANH_TIMER);
                break;
            default:
                break;
        }
}

void execute_fsm() {
    handle_mode();
    if (check_MOD_1()) {
        MODE_CURRENT = (MODE_CURRENT % 4) + 1;
        turn_Off_LEDs();
        reset_to_defaults(MODE_CURRENT);
        reset_MOD_flags();
    }
    if (MODE_CURRENT != 1) {
        if (check_MOD_2()) {
            init_traffic_timers(MODE_CURRENT);
        }
    }
    if (check_MOD_3()) {
        apply_changes(MODE_CURRENT);
    }
}
```

## 6 Bài 6 Adding code for displaying modes

Your tasks in this exercise are:

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

```c
// ----- fsm.c ----- //
static void configure_and_display_mode(int mode, int timer_constant) {
    configure_leds(mode);
    show_mode(mode);
    show_time(timer_constant);
}

static void handle_mode() {
    switch (MODE_CURRENT) {
        case MODE_NORMAL:
            run_traffic_lights();
            break;
        case MODE_DO:
            configure_and_display_mode(MODE_DO, MOD_DO_TIMER);
            break;
        case MODE_VANG:
            configure_and_display_mode(MODE_VANG, MOD_VANG_TIMER);
            break;
        case MODE_XANH:
            configure_and_display_mode(MODE_XANH, MOD_XANH_TIMER);
            break;
        default:
            break;
    }
}

// ----- global.c ----- //
void show_7seg(int value, uint32_t pin) {
    if(value < 0 || value > 9)
```

```c
        return;
    char seg_patterns[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0
        x80, 0x90};
    for (int j = 0; j < 7; j++) {
        HAL_GPIO_WritePin(GPIOB, pin << j, (seg_patterns[value] >> j) & 1);
    }
}

void scan_7seg(int idx) {
    switch (idx) {
        case 0:
            HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, GPIO_PIN_SET);
            show_7seg(buf_mode[0], GPIO_PIN_0);
            HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, GPIO_PIN_RESET);
            break;
        case 1:
            HAL_GPIO_WritePin(EN0_GPIO_Port, EN0_Pin, GPIO_PIN_SET);
            show_7seg(buf_mode[1], GPIO_PIN_0);
            HAL_GPIO_WritePin(EN1_GPIO_Port, EN1_Pin, GPIO_PIN_RESET);
            break;
        case 2:
            HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, GPIO_PIN_SET);
            show_7seg(buf_time[0], GPIO_PIN_7);
            HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, GPIO_PIN_RESET);
            break;
        case 3:
            HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, GPIO_PIN_SET);
            show_7seg(buf_time[1], GPIO_PIN_7);
            HAL_GPIO_WritePin(EN3_GPIO_Port, EN3_Pin, GPIO_PIN_RESET);
            break;
        default:
            break;
    }
}

void update_buffer_mode(int value) {
    if (value >= 0) {
        buf_mode[0] = value / 10;
        buf_mode[1] = value % 10;
    }
}

void update_buffer_time(int value) {
    if (value >= 0) {
        buf_time[0] = value / 10;
        buf_time[1] = value % 10;
    }
}

void show_mode(int value) {
    update_buffer_mode(value);
    if (tmr3_done == 1) {
        scan_7seg(idx_mode);
        idx_mode = (idx_mode + 1) % 2;
        init_tmr3(SCAN_INTERVAL_MS);
    }
}

void show_time(int value) {
    update_buffer_time(value);
    if (tmr4_done == 1) {
        scan_7seg(idx_time);
        idx_time++;
        if (idx_time > 3) {
            idx_time = 2;
        }
        init_tmr4(SCAN_INTERVAL_MS);
    }
}
```

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

```c
// ----- fsm.c ----- //
static void configure_and_display_mode(int mode, int timer_constant) {
    configure_leds(mode);
    show_mode(mode);
    show_time(timer_constant);
}

static void handle_mode() {
    switch (MODE_CURRENT) {
        case MODE_NORMAL:
            run_traffic_lights();
            break;
        case MODE_DO:
            configure_and_display_mode(MODE_DO, MOD_DO_TIMER);
            break;
        case MODE_VANG:
            configure_and_display_mode(MODE_VANG, MOD_VANG_TIMER);
            break;
        case MODE_XANH:
            configure_and_display_mode(MODE_XANH, MOD_XANH_TIMER);
            break;
        default:
            break;
    }
}

// ----- global.c ----- //
void configure_leds(int MODE_CURRENT) {
    if (tmr2_done == 1) {
        if (MODE_CURRENT == MODE_DO) {
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5|GPIO_PIN_8);
        } else if (MODE_CURRENT == MODE_VANG) {
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_6|GPIO_PIN_9);
        } else if (MODE_CURRENT == MODE_XANH) {
            HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7|GPIO_PIN_10);
        }
        init_tmr2(250);
    }
}
```

## 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.

```c
// ----- global.c ----- //
case MODE_DO:
    MOD_DO_TIMER = (MOD_DO_TIMER % 99) + 1;
    break;
```

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

```
// ----- global.c ----- //
case MODE_DO:
    DO_modified = 1;
    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.

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

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

```
// ----- global.c ----- //
case MODE_VANG:
    VANG_modified = 1;
    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.

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

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

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

## 10 Tổng hợp LAB 3

Link video demo: `https://drive.google.com/file/d/1iXA3bZZBbTN_pZSBT3JlWQ2iapAxOgyc/view?usp=sharing`
Link Github: `https://github.com/SangNguyen-232/VXLVDK-LAB3`