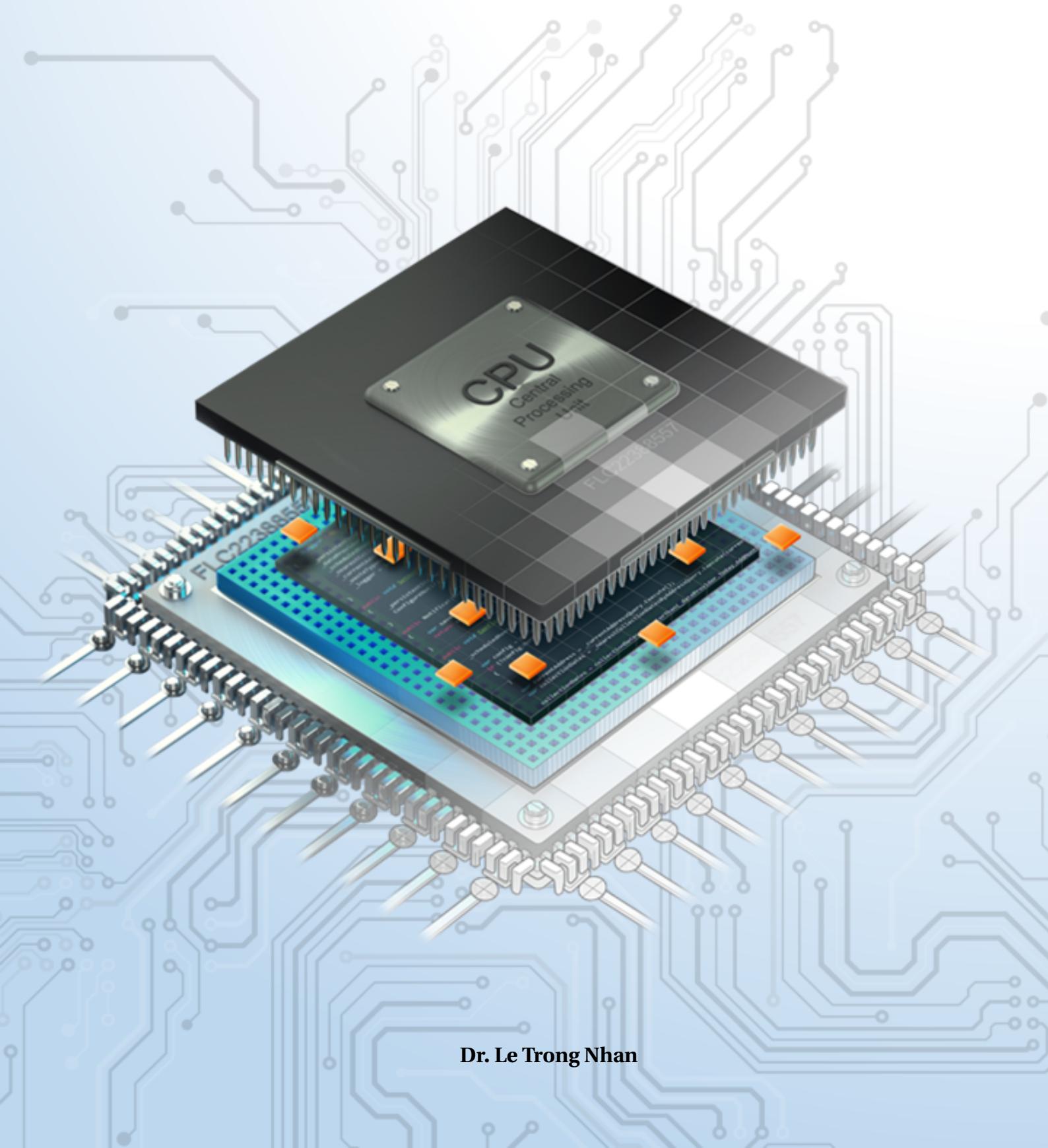




HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller



Dr. Le Trong Nhan

Mục lục

1	Giới thiệu	4
2	Tổng quan hệ thống	4
2.1	Mô tả máy trạng thái	4
2.2	Cấu hình các chân (Pin Configuration)	5
3	Báo cáo chi tiết	7
3.1	Cấu trúc chương trình	7
3.2	Module Software Timer	8
3.3	Module Button (Xử lý nút nhấn)	8
3.4	Module LCD (Hiển thị)	9
3.5	Module FSM (Logic điều khiển)	9
4	Video demo	10
5	Lời cảm ơn	10

1 Giới thiệu

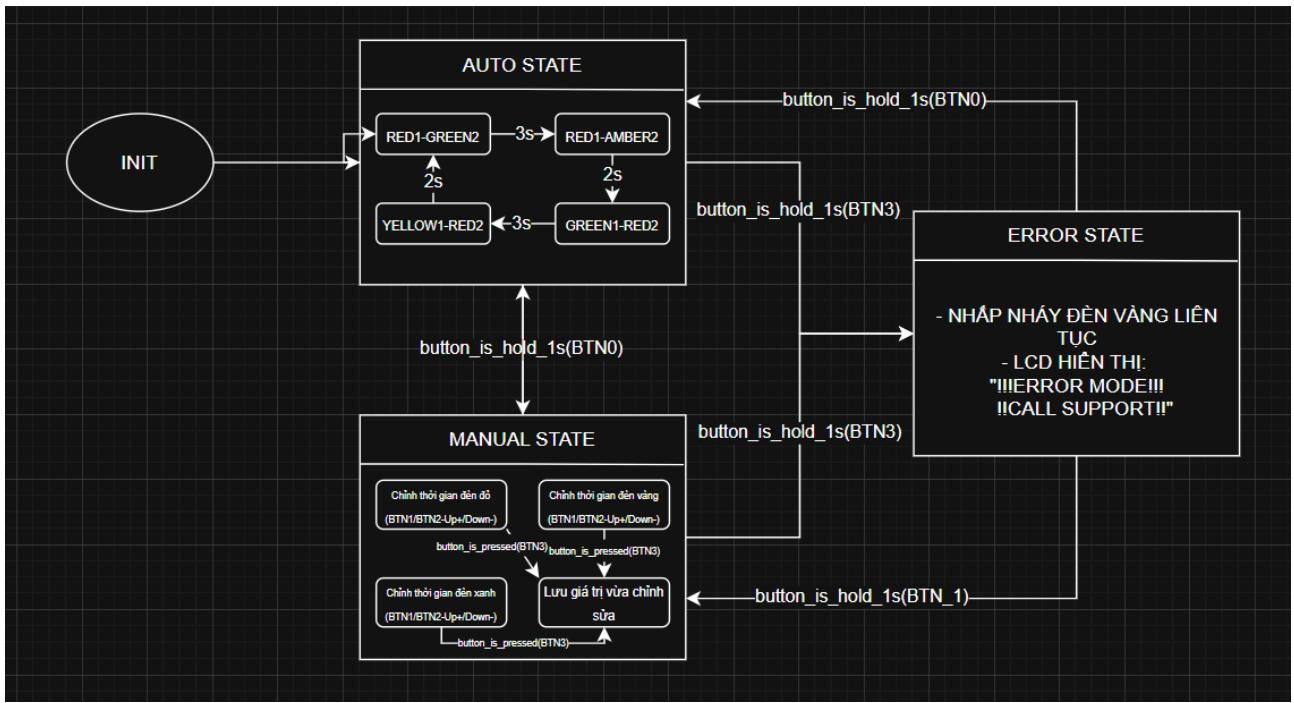
Hệ thống đèn giao thông được thiết kế trên bo mạch Nucleo-F103RB, mô phỏng điều khiển giao thông tại ngã tư. Hệ thống sử dụng màn hình LCD 16x2 để hiển thị thời gian đêm ngược và trạng thái, cùng 4 nút nhấn để điều khiển. Hệ thống hoạt động dựa trên mô hình máy trạng thái hữu hạn FSM để đảm bảo tính ổn định và dễ mở rộng.

2 Tổng quan hệ thống

2.1 Mô tả máy trạng thái

Hệ thống được thiết kế với 4 trạng thái chính, chuyển đổi qua lại dựa trên sự kiện timer và nút nhấn:

- **INIT (Khởi tạo):** Khi cấp nguồn, hệ thống vào trạng thái này để thiết lập các thông số ban đầu, hiển thị màn hình chào mừng. Sau khoảng thời gian định trước (Timer Init), hệ thống tự động chuyển sang chế độ AUTO.
- **AUTO (Tự động):** Đây là chế độ hoạt động mặc định. Đèn giao thông thay đổi theo chu trình: Đỏ → Xanh → Vàng. Thời gian đêm ngược được hiển thị trên LCD.
 - Sự kiện chuyển đổi: Nhấn giữ nút BTN_MODE (1 giây) để sang chế độ MANUAL. Nhấn giữ BTN_ER (1 giây) để sang chế độ ERROR.
- **MANUAL (Thủ công):** Cho phép người dùng cài đặt thời gian cho đèn Đỏ và Xanh. Thời gian đèn Vàng được tự động tính toán hoặc cố định để đảm bảo logic (Đỏ = Xanh + Vàng).
 - Sự kiện chuyển đổi: Nhấn giữ nút BTN_MODE để lưu cài đặt và quay về AUTO.
- **ERROR (Lỗi):** Chế độ cảnh báo. Đèn Vàng ở cả hai tuyến đường nhấp nháy liên tục, LCD hiển thị thông báo lỗi.
 - Sự kiện chuyển đổi: Nhấn giữ nút BTN_MODE để reset về chế độ AUTO.



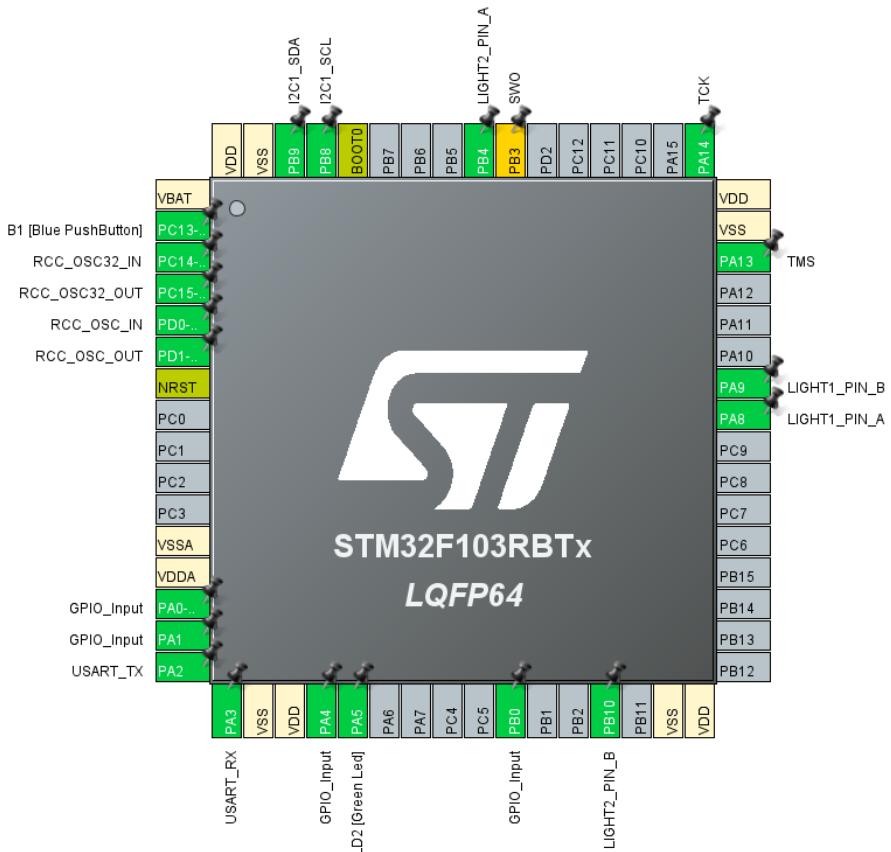
Hình 1: Sơ đồ máy trạng thái (FSM) của hệ thống

2.2 Cấu hình các chân (Pin Configuration)

Cấu hình chân vi điều khiển STM32F103RB được thực hiện trên STM32CubeIDE như sau:

STT	Tên Pin	Chân MCU	Chức năng
1	BTN_MODE	PA0	Nút nhấn chuyển chế độ (Input Pull-up)
2	BTN_UP	PA1	Nút tăng giá trị (Input Pull-up)
3	BTN_DOWN	PA4	Nút giảm giá trị (Input Pull-up)
4	BTN_ER	PB0	Nút chuyển chế độ lỗi (Input Pull-up)
5	LED_DEBUG	PA5	Đèn LED trạng thái trên Kit (Output)
6	LIGHT1_A	PA8	Điều khiển Đèn giao thông 1 (Bit 0)
7	LIGHT1_B	PA9	Điều khiển Đèn giao thông 1 (Bit 1)
8	LIGHT2_A	PB4	Điều khiển Đèn giao thông 2 (Bit 0)
9	LIGHT2_B	PB10	Điều khiển Đèn giao thông 2 (Bit 1)
10	I2C1_SCL	PB8	Giao tiếp xung nhịp màn hình LCD
11	I2C1_SDA	PB9	Giao tiếp dữ liệu màn hình LCD

Bảng 1: Bảng phân bổ chân tín hiệu



Hình 2: Cấu hình chân trên STM32CubeIDE

TIM2 Mode and Configuration

Mode

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

<input checked="" type="radio"/> NVIC Settings	<input checked="" type="radio"/> DMA Settings
<input checked="" type="radio"/> Parameter Settings	<input checked="" type="radio"/> User Constants

Configure the below parameters :

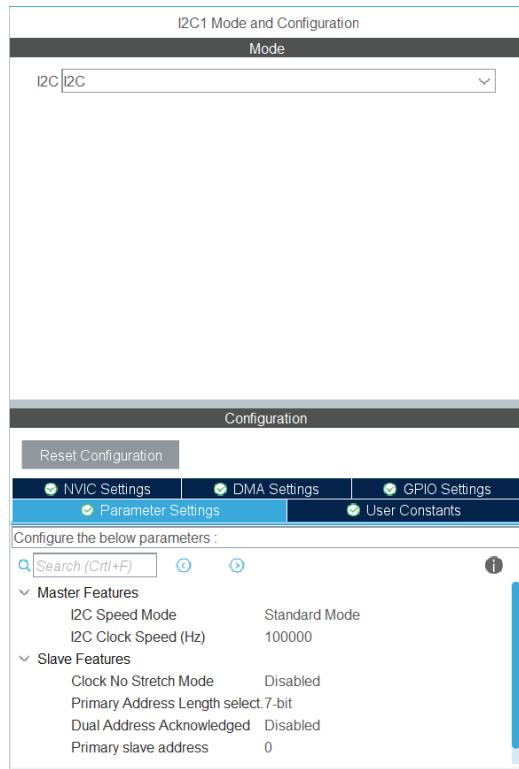
Counter Settings

- Prescaler (PSC - 16 bits value) 63999
- 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 delayed)

Hình 3: Cấu hình Timer



Hình 4: Cấu hình I2C

3 Báo cáo chi tiết

Phần này trình bày chi tiết việc hiện thực các module chức năng trên vi điều khiển STM32.

3.1 Cấu trúc chương trình

Chương trình chính sử dụng vòng lặp vô tận kết hợp với ngắt Timer (System Tick 10ms). Hàm main đóng vai trò là bộ điều phối (Dispatcher), gọi các máy trạng thái tương ứng dựa trên biến toàn cục STATUS.

```

1 while (1)
2 {
3     switch (STATUS) {
4         case INIT:
5             FSM_INIT_RUN();
6             break;
7         case AUTO:
8             FSM_AUTO_RUN();
9             break;
10        case MANUAL:
11            FSM_MANUAL_RUN();
12            break;
13        case ERROR:
14            FSM_ERROR_RUN();
15            break;

```

```

16    }
17    updateLCD(); // Update LCD
18 }
```

Program 1: Vòng lặp chính trong main.c

3.2 Module Software Timer

Để tránh việc sử dụng hàm HAL_Delay gây dừng chương trình (blocking), một bộ định thời mềm (Software Timer) được xây dựng. Timer này được cập nhật trong ngắt TIM2_IRQHandler mỗi 10ms (giá trị TICK).

```

1 void timer_run() {
2     for (int i = 0; i < TIMER_LIMIT; i++) {
3         if (timer_counter[i] > 0) {
4             timer_counter[i]--;
5             if (timer_counter[i] <= 0) {
6                 timer_flag[i] = 1; // Flag on
7             }
8         }
9     }
10 }
```

Program 2: Software timer

3.3 Module Button (Xử lý nút nhấn)

Module nút nhấn chịu trách nhiệm đọc trạng thái 4 nút nhấn, khử rung (debouncing) và phát hiện các sự kiện: nhấn (pressed), nhấn giữ 1 giây (long press).

Giải thuật khử rung đọc tín hiệu đầu vào 3 lần liên tiếp trong các chu kỳ ngắt timer. Chỉ khi 3 lần đọc giống nhau thì trạng thái nút mới được xác nhận.

```

1 void GetKeyInput(void) {
2     for (int i = 0; i < NUM_BUTTONS; i++) {
3         // ... (Debouncing)
4
5         if (button_state[i] == BUTTON_IS_PRESSED) {
6             counter_for_hold[i]++;
7             if (counter_for_hold[i] == TICKS_FOR_1S_HOLD) {
8                 flag_btn_hold_1s[i] = 1; // Hold flag on
9             }
10        } else {
11            // Press flag on
12            if (counter_for_hold[i] < TICKS_FOR_1S_HOLD) {
13                flag_btn_press[i] = 1;
14            }
15        }
16    }
}
```

Program 3: Read and button function

3.4 Module LCD (Hiển thị)

Module LCD sử dụng giao tiếp I2C để tiết kiệm chân vi điều khiển. Để tối ưu hiệu năng, kỹ thuật **Caching** (Bộ nhớ đệm trạng thái) được áp dụng. Màn hình chỉ được gửi lệnh vẽ lại khi nội dung thực sự thay đổi so với lần hiển thị trước đó.

```

1 static LCD_Cache_t lcd_cache = {-1, -1, -1, -1};

2

3 void updateLCD(void) {
4     // If state changes, then reset cache
5     if (lcd_cache.status != STATUS) {
6         LCD_Reset_Cache();
7         lcd_cache.status = STATUS;
8     }
9     // Call draw function
10    switch (STATUS) {
11        case AUTO: LCD_Draw_Auto(); break;
12        case MANUAL: LCD_Draw_Manual(); break;
13        case ERROR: LCD_Draw_Error(); break;
14    }
15 }
```

Program 4: Caching in LCD.c

3.5 Module FSM (Logic điều khiển)

Logic điều khiển đèn giao thông được chia thành các file `FSM_AUTO.c`, `FSM_MANUAL.c`, v.v. Ví dụ, trong chế độ AUTO, hệ thống kiểm tra cờ timer để chuyển đổi giữa các trạng thái đèn (R1-G2, R1-Y2, G1-R2, Y1-R2).

```

1 void FSM_AUTO_RUN() {
2     switch (fsm_auto_state) {
3         case STATE_RED1_GREEN2:
4             setTrafficLight(STATE_RED1_GREEN2);
5             // Change State when time out
6             if (getTimerFlag(TIMER_FSM_AUTO) == 1) {
7                 fsm_auto_state = STATE_RED1_YELLOW2;
8                 setTimer(TIMER_FSM_AUTO, time_yellow * 1000);
9                 remaining_time_road2 = time_yellow;
10            }
11            break;
12            // ... Other
13    }
14 }
```

Program 5: Logic chuyển trạng thái tự động

4 Video demo

Video mô phỏng hoạt động của hệ thống trên phần mềm Proteus và giải thích chi tiết các trường hợp kiểm thử (Test Cases):

https://github.com/namhcmutpd/VXL_PROJECT

5 Lời cảm ơn

- **Giảng viên hướng dẫn:** Dr. Lê Trọng Nhân
- **Sinh viên thực hiện:**
 - Trương Hoàng Nam (MSSV: 2312201)
 - Vương Quốc Khánh (MSSV: 2311542)

Nhóm chúng em xin cảm ơn sự dạy dỗ và quan tâm tận tình của thầy Nhân trong khoá học này suốt thời gian vừa qua. Qua khoá học này, nhóm chúng em ngày càng hoàn thiện hơn những kiến thức và kỹ năng của một trong những một chuyên ngành cốt lõi và quan trọng của ngành Kỹ thuật máy tính: "Vi xử lý - Vi điều khiển". Một lần nữa, nhóm chúng em xin cảm ơn thầy và kính chúc thầy thật nhiều sức khoẻ!