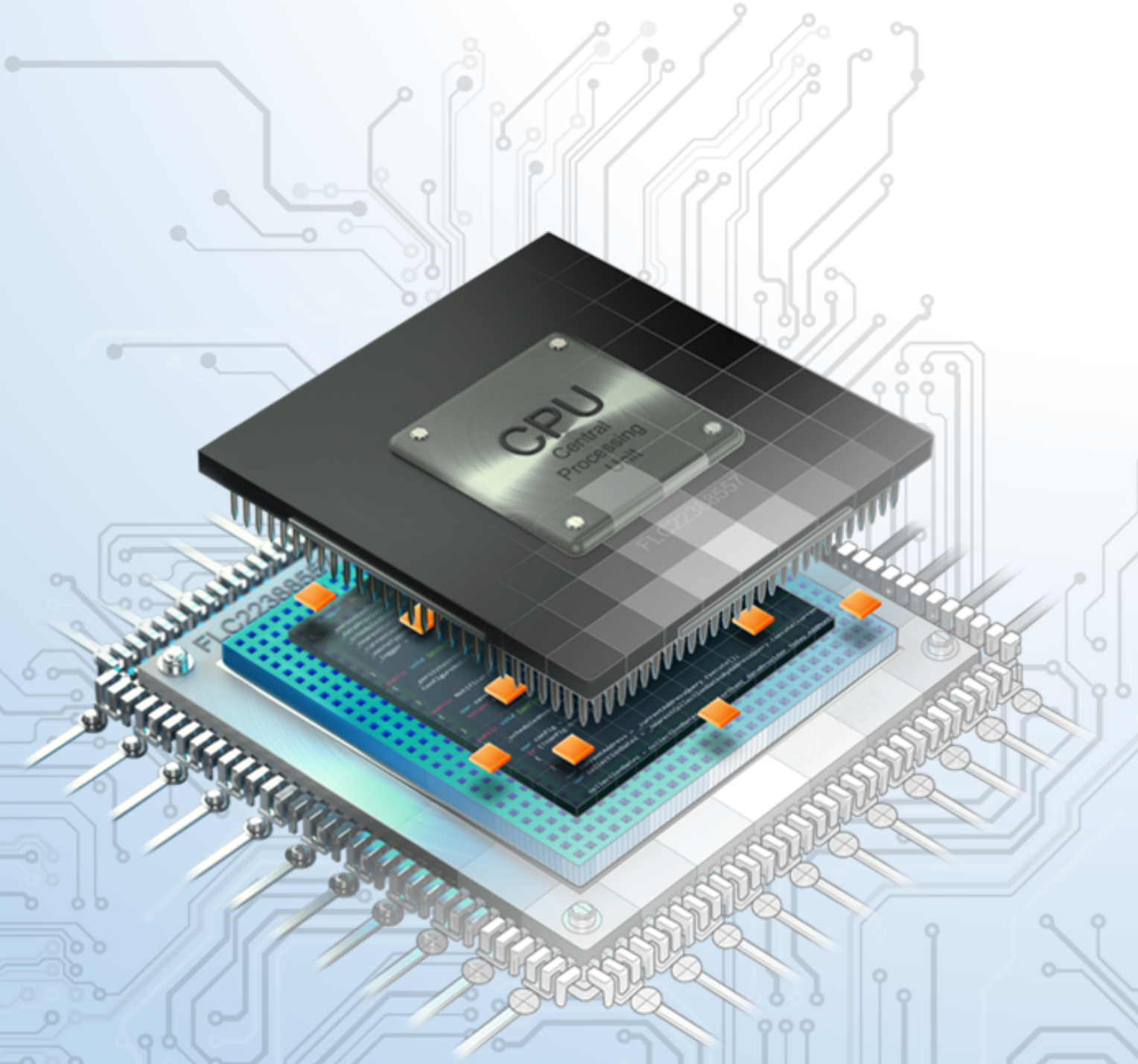
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

# Microcontroller




**Dr. Le Trong Nhan**

# Mục lục

# 1 Giới thiệu

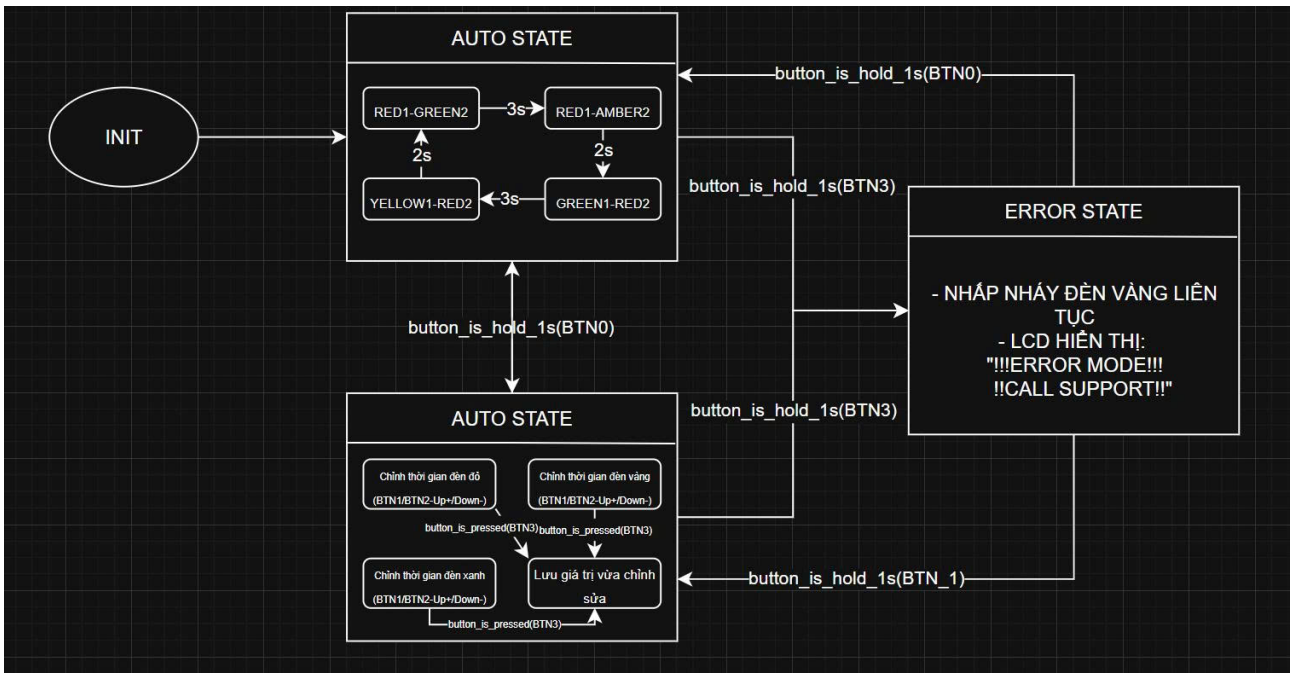
Hệ thống đèn giao thông được thiết kế trên bo mạch Nucleo-F103RB, mô phỏng điều khiển giao thông tại ngã tư. Hệ thống sử dụng màn hình LCD 16x2 để hiển thị thời gian đếm ngược và trạng thái, cùng 4 nút nhấn để điều khiển. Hệ thống hoạt động dựa trên mô hình máy trạng thái hữu hạn FSM để đảm bảo tính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



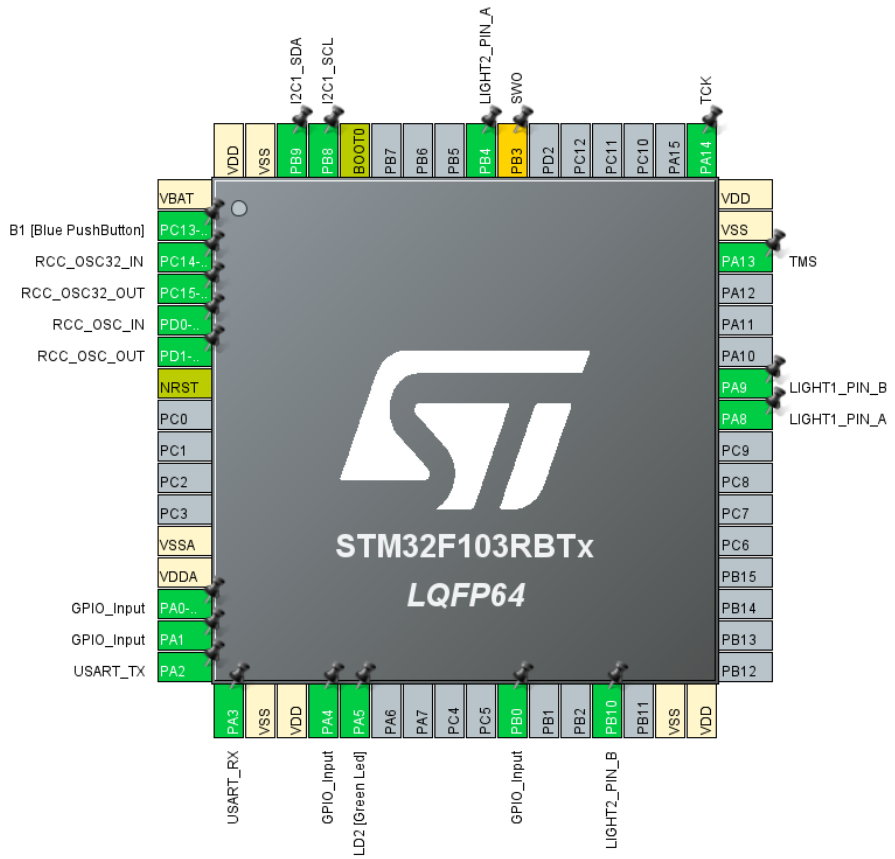

Hình 1: Sơ đồ máy trạng thái (FSM) của hệ thống

## 2.2 Cấu hình các chân (Pin Configuration)

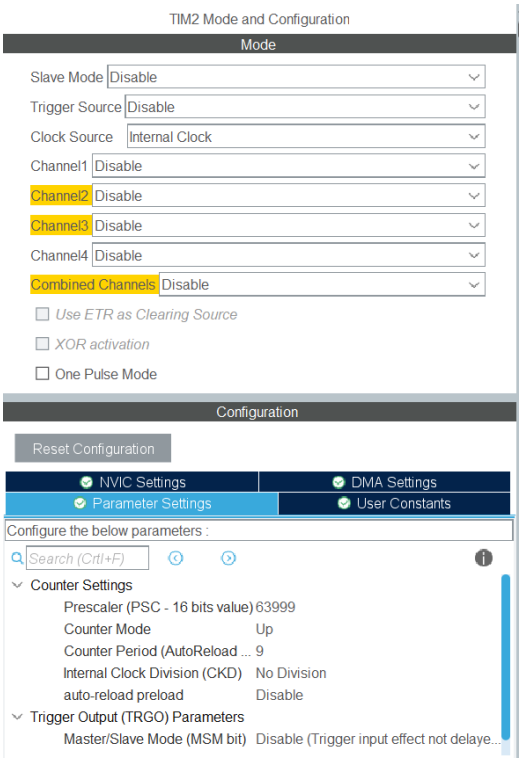
Cấu hình chân vi điều khiển STM32F103RB được thực hiện trên STM32CubeIDE như sau:

| 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 |

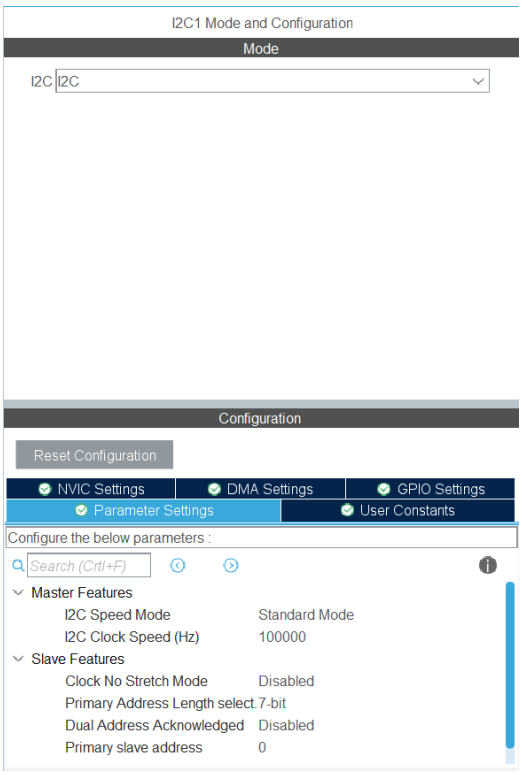
Bảng 1: Bảng phân bố chân tín hiệu




Hình 2: Cấu hình chân trên STM32CubeIDE




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`.

```
while (1)
{
  switch (STATUS) {
      case INIT:
           FSM_INIT_RUN();
           break;
      case AUTO:
           FSM_AUTO_RUN();
           break;
      case MANUAL:
           FSM_MANUAL_RUN();
           break;
      case ERROR:
           FSM_ERROR_RUN();
           break;
```

```
 }
  updateLCD(); // Update LCD
}
```

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).

```
void timer_run() {
    for (int i = 0; i < TIMER_LIMIT; i++) {
        if (timer_counter[i] > 0) {
            timer_counter[i]--;
            if (timer_counter[i] <= 0) {
                timer_flag[i] = 1; // Flag on
            }
        }
    }
}
```

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.

```
void GetKeyInput(void) {
    for (int i = 0; i < NUM_BUTTONS; i++) {
        // ... (Debouncing)

        if (button_state[i] == BUTTON_IS_PRESSED) {
            counter_for_hold[i]++;
            if (counter_for_hold[i] == TICKS_FOR_1S_HOLD) {
                flag_btn_hold_1s[i] = 1; // Hold flag on
            }
        } else {
             // Press flag on
             if (counter_for_hold[i] < TICKS_FOR_1S_HOLD) {
                 flag_btn_press[i] = 1;
             }
        }
    }
```

```
}
```

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.

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

void updateLCD(void) {
    // If state changes, then reset cache
    if (lcd_cache.status != STATUS) {
        LCD_Reset_Cache();
        lcd_cache.status = STATUS;
    }
    // Call draw function
    switch (STATUS) {
        case AUTO: LCD_Draw_Auto(); break;
        case MANUAL: LCD_Draw_Manual(); break;
        case ERROR: LCD_Draw_Error(); break;
    }
}
```

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).

```
void FSM_AUTO_RUN() {
    switch (fsm_auto_state) {
        case STATE_RED1_GREEN2:
            setTrafficLight(STATE_RED1_GREEN2);
            // Change State when time out
            if (getTimerFlag(TIMER_FSM_AUTO) == 1) {
                fsm_auto_state = STATE_RED1_YELLOW2;
                setTimer(TIMER_FSM_AUTO, time_yellow * 1000);
                remaining_time_road2 = time_yellow;
            }
            break;
        // ... Other
    }
}
```

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 môt chuyên ngành cốt lõi và quan trọng của ngành Kĩ 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!***