**Bài tập 1:** Các dải đèn LED RGB địa chỉ hóa (addressable RGB LED strips) như WS2812B (còn gọi là NeoPixel) rất phổ biến trong các dự án DIY và sản phẩm thương mại. Để điều khiển chúng, một vi điều khiển không gửi đi các tín hiệu Analog riêng biệt.

Thay vào đó, nó gửi một luồng dữ liệu số (a stream of digital data) nối tiếp nhau. Mỗi con chip LED trên dải sẽ "lấy" phần dữ liệu dành cho mình, sau đó truyền phần còn lại cho con LED tiếp theo.

Dữ liệu cho mỗi con LED là một gói 24-bit (3 bytes) định nghĩa màu sắc của nó: 8 bit cho Green, 8 bit cho Red, và 8 bit cho Blue (Thứ tự G-R-B là một đặc điểm phổ biến của loại LED này). Nhiệm vụ của bạn là viết một "driver" phần mềm cấp cao, cung cấp các hàm tiện ích để tạo và quản lý một bộ đệm (buffer) chứa toàn bộ dữ liệu màu cho cả dải LED, sẵn sàng để được gửi đi.

**Yêu cầu phần mềm**:

1. **Cấu trúc Dữ liệu Màu (Color Data Structure):**
    a. Mỗi pixel LED được điều khiển bởi một giá trị 32-bit uint32_t để dễ dàng thao tác. Tuy nhiên, chỉ 24 bit được sử dụng để chứa dữ liệu màu.
    b. Định dạng 24-bit trong uint32_t:

| **Bits (31-24)** | **Bits (23-16)** | **Bits (15-8)** | **Bits (7-0)** |
|---|---|---|---|
| Không sử dụng (8 bits) | Green (8 bits) | Red (8 bits) | Red (8 bits)Blue (8 bits) |
| 0x00 | 0-255 | 0-255 | 0-255 |

2. **Cấu trúc mã nguồn:**
    a. led_driver.h: Định nghĩa API công khai để điều khiển dải LED.
    b. led_driver.c: Triển khai logic của driver.
    c. main.c: Chương trình chính để mô phỏng và kiểm thử driver.
3. **led_driver.h**

| **Hàm** | **Mô tả** |
|---|---|
| int led_init(size_t num_pixels); | Initializes a buffer for the LED strip with the given number of pixels.<br>Returns 0 on success, -1 on failure (e.g., memory allocation failed). |
| void led_shutdown(); | Frees the memory allocated for the LED strip. |
| void led_set_pixel_color(size_t index, uint8_t r, uint8_t g, uint8_t b); | Sets the color of a specific pixel at the given 'index'.<br>Uses 3 separate 8-bit values for Red, Green, and Blue. |
| void led_fill(uint8_t r, uint8_t g, uint8_t b); | Fills the entire strip with a single color. |

| | |
|---|---|
| void led_clear(); | Turns off all pixels (by setting their color to black). |
| const uint32_t* led_get_buffer(); | Gets a constant (read-only) pointer to the data buffer, ready to be "sent" out. This function is used for testing purposes in this exercise. |
| size_t led_get_pixel_count(); | Gets the number of pixels on the LED strip. |

**Yêu cầu chức năng:**

1. **Quản lý Bộ nhớ:**
    a. led_init(): Phải cấp phát động một mảng uint32_t có kích thước bằng num_pixels. Toàn bộ mảng phải được khởi tạo về 0 (màu đen).
    b. led_shutdown(): Phải giải phóng bộ nhớ đã được cấp phát bởi led_init() để tránh rò rỉ bộ nhớ.
2. **Thao tác Bit để Tạo Màu (Đây là phần cốt lõi):**
    a. Hàm led_set_pixel_color() phải nhận 3 giá trị r, g, b (mỗi giá trị 8-bit) và dùng các toán tử thao tác bit để "gói" chúng lại thành một số uint32_t duy nhất theo đúng định dạng G-R-B đã quy định.
    b. Gợi ý: Bạn sẽ cần sử dụng toán tử dịch trái (<<) để đẩy các giá trị Green và Red vào đúng vị trí của chúng, và toán tử OR (|) để kết hợp chúng lại với nhau.
3. **Các Hàm Tiện ích:**
    a. led_fill(): Phải tính toán giá trị màu 32-bit một lần, sau đó lặp qua toàn bộ buffer và gán giá trị này cho tất cả các pixel.
    b. led_clear(): Một trường hợp đặc biệt của led_fill() với màu là (0, 0, 0).
4. **An toàn và Kiểm tra Biên:**
    a. Tất cả các hàm nhận index (như led_set_pixel_color) phải kiểm tra xem index có nằm trong phạm vi hợp lệ (0 đến num_pixels - 1) hay không. Nếu không, hàm phải lặng lẽ bỏ qua để tránh ghi ra ngoài vùng nhớ.

**Hướng dẫn Kiểm thử (main.c)**

File main.c của bạn nên thực hiện các kịch bản sau:

1. Khởi tạo một dải LED với 10 pixel bằng led_init(10).
2. Kiểm tra xem buffer có được khởi tạo về 0 hết không.
3. Đặt màu cho pixel đầu tiên (index 0) thành màu Đỏ (255, 0, 0).
4. Đặt màu cho pixel cuối cùng (index 9) thành màu Xanh dương (0, 0, 255).
5. Đặt màu cho một pixel ở giữa (ví dụ: index 4) thành màu Trắng (255, 255, 255).
6. Sử dụng hàm led_get_buffer() để lấy con trỏ tới buffer dữ liệu.

7. In ra giá trị hexa của các pixel tại index 0, 4, và 9 để xác thực.
    - Giá trị tại index 0 phải là 0x0000FF00 (Red).
    - Giá trị tại index 9 phải là 0x000000FF (Blue).
    - Giá trị tại index 4 phải là 0x00FFFFFF (White).
8. Gọi hàm led_fill() để đặt tất cả các pixel thành màu Xanh lá (0, 255, 0).
9. Kiểm tra lại buffer để đảm bảo tất cả 10 pixel đều có giá trị 0x00FF0000.
10. Gọi led_shutdown() để giải phóng bộ nhớ.