**Báo cáo thực hành Lab 1 (29/03)**

Họ và tên: Phạm Đức Long

Mã số sinh viên: 20225737

**MỤC LỤC**

[**I, Lưu ý: 2**](#_8e5h7aeyha3x)

[**1. Cam kết 2**](#_ocbu2prrpkmw)

[**2. Video, hình ảnh 2**](#_xs4m6fyvweo0)

[**II, Nội dung 2**](#_9mxta0d708fb)

[**3.1, Giải thích hoạt động của phần code viết thêm 2**](#_uynl7w24wz3s)

[3.1.1, Hàm DisplayLeds: 2](#_r4kc926y2vk)

[3.1.2, Hàm Running\_spot\_L: Bật led từ phải qua trái 3](#_jmtsz7ah5two)

[3.1.3, Hàm Running\_spot\_R: Bật led từ trái qua phải 3](#_tjtppac5dfne)

[3.1.4, Hàm Flash: Bật tất cả 8 led, trễ thích hợp sau đó, tắt tất cả 8 led 4](#_95ayqwomky1a)

[3.1.5, Hàm Spot\_bumber: Bật lần lượt 2 led đối xứng từ ngoài vào trong rồi từ trong ra ngoài 4](#_j07hp1o1jtyl)

[3.1.6, Hiệu ứng lặp lại 2 chu kỳ 5](#_4c94imiio64s)

[**3.2, Ghép nối LED 7 thanh 5**](#_1d8xf62f7b52)

[**3.3, Giải mã tín hiệu điều khiển hồng ngoại 6**](#_2zwzbmo26742)

[3.3.1, Biểu đồ giá trị tín hiệu trên chân PG5 theo thời gian khi có 1 nút được bấm trên điều khiển 7](#_t5nu1ug2tx2h)

[3.3.2, Ngắt ngoài EXTI5 trên chân PG5 được đặt là Rising edge hay Falling edge? Tại sao? 9](#_nza94rnvm8yl)

[3.3.3, Làm thế nào để đo thời gian của các bit? 10](#_nidi3whjc9n)

[3.3.4, Căn cứ vào đâu để xác định các bit start, 0, 1? 11](#_nlc6u0hagw8b)

[**4. Bài tập tự làm: Giải mã lệnh điều khiển từ xa hồng ngoại 11**](#_4h90wc69ugri)

[4.1, Khai báo biến last\_button 11](#_wg4fcxyj5lvd)

[4.2, Code logic trong vòng lặp while: 12](#_l8gsxuae6lsv)

# **I, Lưu ý:**

## 1. Cam kết

- Nội dung và mã nguồn trong báo cáo thực hành này là do em và bạn Nguyễn Bá Hoàng (20225844) cùng tự làm. Bất cứ nội dung nào tham khảo từ bên ngoài thì sẽ được nêu rõ nguồn gốc và tác giả.

## 2. Hình ảnh, video

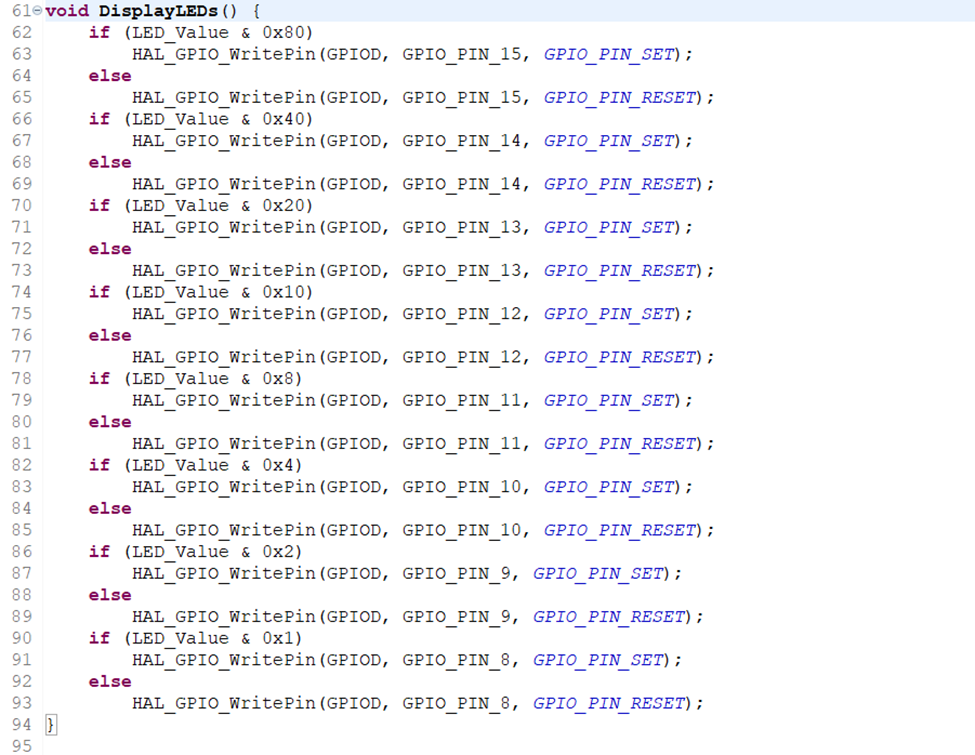
- Tất cả các hình ảnh, video về các bài tập trong buổi thực hành được nhóm em chụp và quay sẽ có ở [link](https://drive.google.com/drive/folders/1yXehBJs3CPROLCyO7LkIJxdSfS4YTllv?usp=sharing) để đảm bảo chất lượng hình ảnh, video và dung lượng file báo cáo ạ.

# **II, Nội dung**

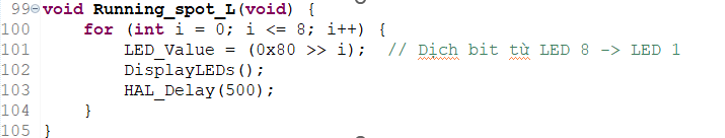
## **3.1, Giải thích hoạt động của phần code viết thêm**

- [Link phần 3.1](https://drive.google.com/file/d/1RLXdwTFnAIrNlU4AjDN6JygYZYkyO2_Q/view?usp=sharing)

### **3.1.1, Hàm DisplayLeds:**



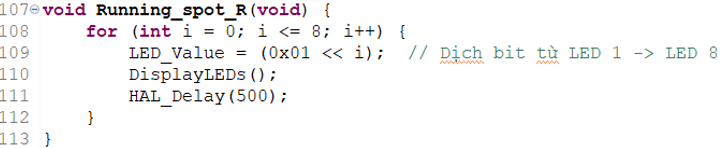
### **3.1.2, Hàm Running\_spot\_L: Bật led từ phải qua trái**



- Giải thích: Ban đầu, biến điều khiển đèn LED được gán giá trị 0x80 (1000 0000 hệ nhị phân) để bật LED đầu tiên từ bên phải. Trong vòng lặp for, biến i tăng dần và giá trị của biến này được dùng để dịch bit sang phải, sau đó gọi hàm DisplayLeds để hiển thị trạng thái LED. Sau mỗi lần hiển thị, chương trình tạm dừng 0,5 giây trước khi tiếp tục vòng lặp. Giá trị của Led\_Value lần lượt thay đổi theo trình tự sau:

1000 0000 -> 0100 0000 -> 0010 0000 -> 0001 0000 -> 0000 1000 -> 0000 0100 -> 0000 0010 -> 0000 0001

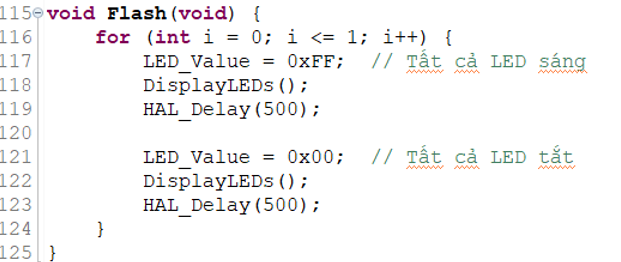
### **3.1.3, Hàm Running\_spot\_R: Bật led từ trái qua phải**



- Giải thích: tương tự Running\_spot\_L, nhưng dịch bit ngược lại. Giá trị của Led\_Value lần lượt thay đổi theo trình tự sau:

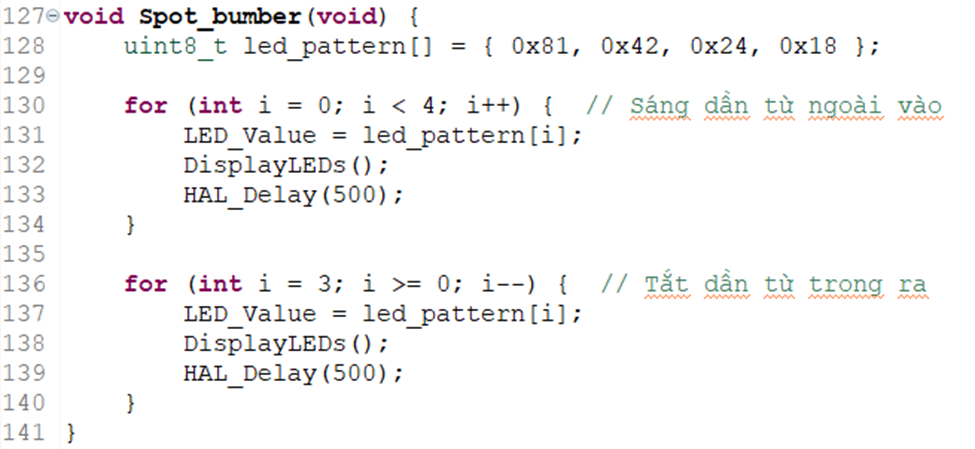
0000 0001 -> 0000 0010 -> 0000 0100 -> 0000 1000-> 0001 0000-> 0010 0000 -> 0100 0000 -> 1000 0000

### **3.1.4, Hàm Flash: Bật tất cả 8 led, trễ thích hợp sau đó, tắt tất cả 8 led**



- Giải thích: Truyền 0xFF vào Led\_Value để hiện 8 đèn, sau delay(500) = 0.5s thì truyền Led\_Value = 0x00 để tắt hết đèn.

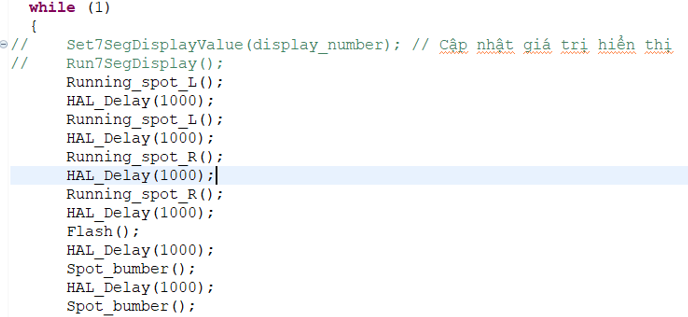
### **3.1.5, Hàm Spot\_bumber: Bật lần lượt 2 led đối xứng từ ngoài vào trong rồi từ trong ra ngoài**



- Giải thích:

* Biến led\_pattern lưu trữ 4 giá trị đại diện cho các cặp led đối xứng: (1-8), (2-7), (3-6), (4-5), tương ứng với các giá trị nhị phân: (1000 0001), (0100 0010), (0010 0100), (0001 1000).
* Vòng lặp for đầu tiên gán lần lượt các giá trị trong led\_pattern vào Led\_Value theo thứ tự từ trái sang phải, giúp các LED sáng dần từ hai bên ngoài vào giữa.
* Vòng lặp for thứ hai thực hiện ngược lại, duyệt led\_pattern từ phải sang trái để các LED sáng dần từ trung tâm ra hai bên ngoài.

### **3.1.6, Hiệu ứng lặp lại 2 chu kỳ**



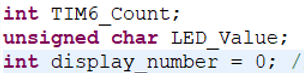
- Giải thích: Vòng lặp while lần lượt gọi các hàm điều khiển LED để thực hiện các hiệu ứng theo trình tự. Mỗi hiệu ứng được chạy hai lần, với khoảng dừng 1 giây giữa mỗi lần. Sau khi tất cả các hiệu ứng đã hoàn thành, vòng lặp quay lại từ đầu, tạo nên chuỗi hiệu ứng lặp liên tục.

## **3.2, Ghép nối LED 7 thanh**

- [Ảnh bài mẫu phần 3.2](https://drive.google.com/file/d/1Y_en_-eoOafudDyhCfwBKW6GedyYMJLH/view?usp=sharing), [Video bài tự làm phần 3.2](https://drive.google.com/file/d/1lc83FG0FU1cZt55N4f0R4ytDzB-HkPVT/view?usp=sharing)

- Giải thích hoạt động code viết thêm:

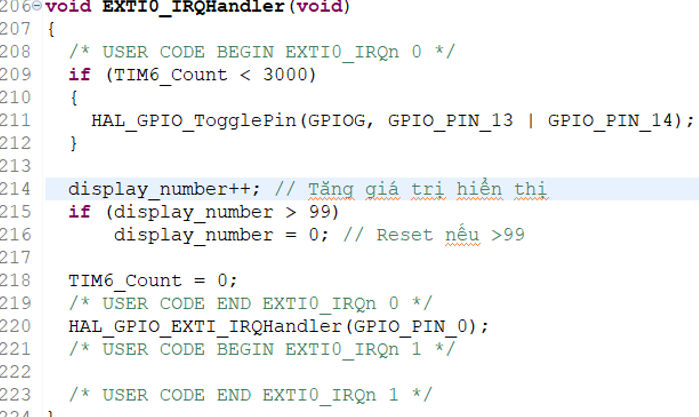
* Đầu tiên ta tạo biến giá trị hiển thị để lưu giá trị hiển thị ra led 7 thanh và để làm tham số truyền vào hàm Set7SegDisplayValue trong thư viện 7seg.c.:



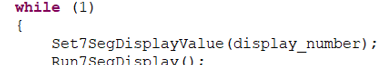
* Ta khai báo biến dùng chung vào file stm32f4xx\_it.c vì biến này cần được sử dụng trong hàm EXTI0 trong file it.c, ta khai báo biến này trong it.c với từ khóa extern để trình biên dịch hiểu rằng biến này là biến toàn cục, được chia sẻ giữa main.c và it.c.



* Để tăng giá trị hiển thị trên led 7 thanh khi ấn nút B1



* Khi nút B1 được bấm thì sẽ có 1 ngắt xuất hiện. Ngắt đó sẽ gọi hàm EXT0\_IRQHandler đã được cấu hình trước đó như trên hình. Khi đó, ta sẽ thực hiện tăng giá trị biến display\_number lên 1 đơn vị và nếu giá trị đó vượt quá 99 thì sẽ được reset về giá trị 0.
* Cuối cùng, hiển thị giá trị ra led 7 thanh bằng cách truyền giá trị của biến display\_number vào hàm Set7SegDisplayValue() để thiết lập số hiển thị trên LED 7 thanh. Sau đó, gọi hàm Run7SegDisplay() để hiển thị giá trị này lên led.



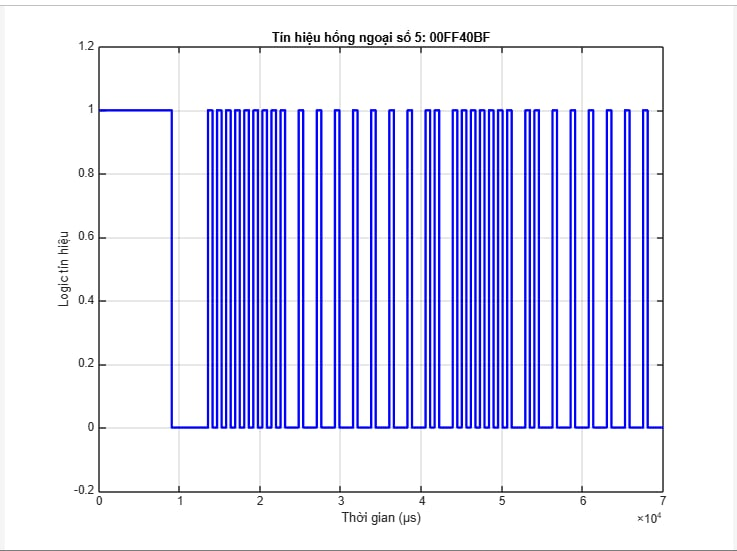
## **3.3, Giải mã tín hiệu điều khiển hồng ngoại**

- [Video bài 3.3 mẫu](https://drive.google.com/file/d/1BWOPzJk6JB9JfyBxlLeozX0YI50e7vPN/view?usp=sharing)

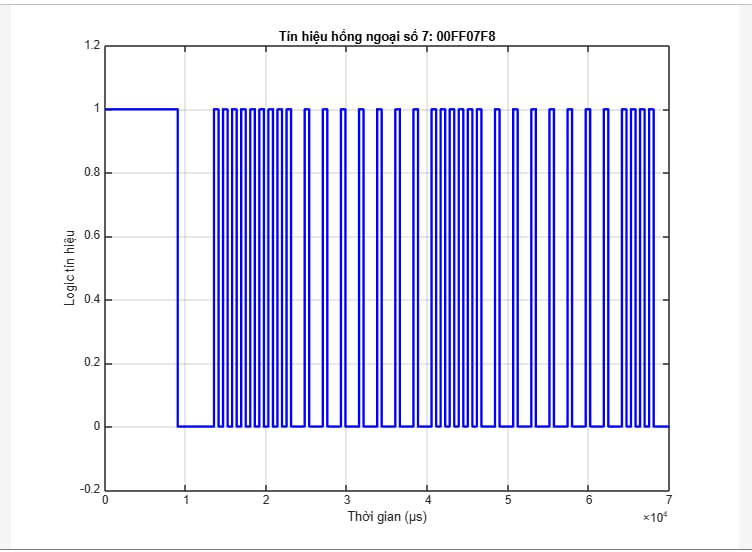
### **3.3.1, Biểu đồ giá trị tín hiệu trên chân PG5 theo thời gian khi có 1 nút được bấm trên điều khiển**

- Vì em có MSSV là 20225737 nên em xin biểu diễn các tín hiệu nút bấm 5, 3, 7, các tín hiệu còn lại tương tự ạ.

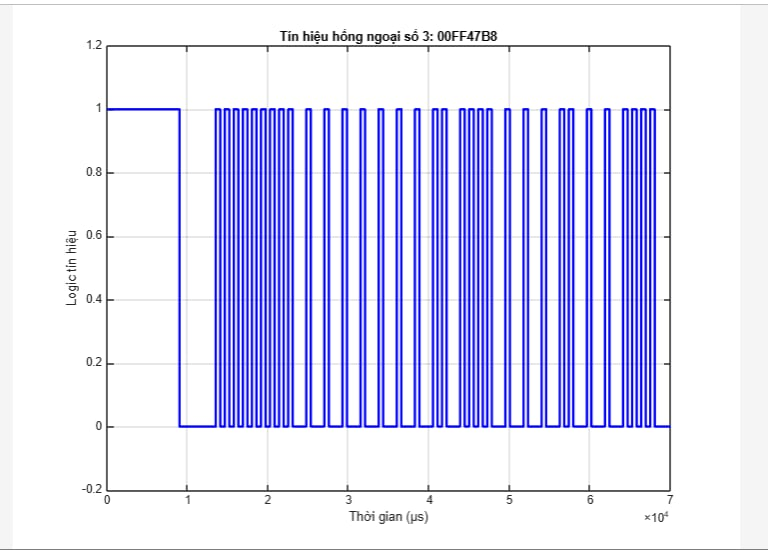
* Tín hiệu nút bấm 5:



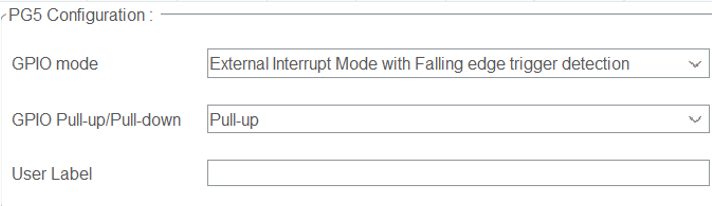
* Tìn hiệu nút bấm 7:



* Tín hiệu nút bấm 3:

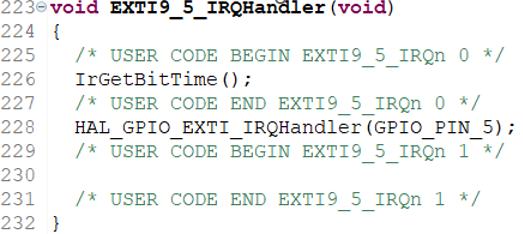


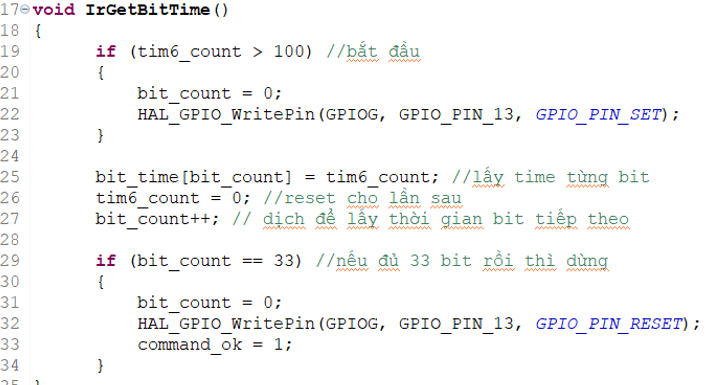
### **3.3.2, Ngắt ngoài EXTI5 trên chân PG5 được đặt là Rising edge hay Falling edge? Tại sao?**



- Cần cấu hình ngắt ở **falling edge** vì tín hiệu mặc định từ bộ thu hồng ngoại ở trạng thái cao (mức 1) khi không nhận tín hiệu. Khi có một chuỗi burst đến, tín hiệu này sẽ chuyển xuống mức thấp (mức 0). Chính tại thời điểm chuyển từ 1 xuống 0 này, ta mới gọi hàm ngắt để bắt đầu quá trình đo thời gian của chuỗi xung; từ đó phân tích xem đó là bit 0, bit 1 hay là tín hiệu bắt đầu.

### **3.3.3, Làm thế nào để đo thời gian của các bit?**

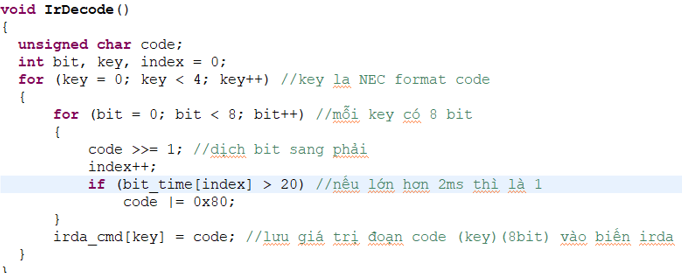




- Giải thích:

* Mỗi khi chân PG5 chuyển từ mức logic cao (1) xuống thấp (0), một ngắt ngoài sẽ được kích hoạt. Trong hàm xử lý ngắt EXTI9\_5\_IRQHandler, hàm IrGetBitTime() được gọi để đo và lưu khoảng thời gian giữa hai lần ngắt liên tiếp vào mảng bit\_time.
* Khoảng thời gian này được tính dựa trên biến tim6\_count, được thiết lập để tăng dần với tần số 10.000Hz (tương ứng mỗi đơn vị là 0,1ms) và được reset sau mỗi lần đo (theo dòng code số 26). Sau khi đã ghi nhận đủ 33 bit dữ liệu, chương trình sẽ tắt chân GPIO (bằng GPIO\_PIN\_RESET) và đặt command\_ok = 1 để báo hiệu quá trình thu tín hiệu đã hoàn tất.

### **3.3.4, Căn cứ vào đâu để xác định các bit start, 0, 1?**



- Dựa vào khoảng thời gian giữa các lần ngắt được ghi trong mảng bit\_time, ta có thể phân biệt giá trị của từng bit. Cụ thể, nếu giá trị thời gian tại một phần tử trong bit\_time lớn hơn 2ms thì ta xem đó là bit 1, ngược lại là bit 0.

## **4. Bài tập tự làm: Giải mã lệnh điều khiển từ xa hồng ngoại**

- [Video bài 4](https://drive.google.com/file/d/1FU7_8v1vyRYgRlCh6FAMrJEX34f3sdgC/view?usp=sharing)

### **4.1, Khai báo biến last\_button**



- Giải thích:

* Biến này được sử dụng để xử lý nút bấm OK, nhằm tránh tình trạng đèn G14 bị toggle liên tục do vòng lặp while liên tục kiểm tra byte thứ 3 trong mảng irda\_cmd. Khi chưa có nút mới được bấm, mảng irda\_cmd vẫn giữ nguyên giá trị gồm 4 byte cũ, trong đó có dữ liệu của nút OK. Điều này khiến đèn G14 liên tục chuyển trạng thái trong mỗi vòng lặp với tốc độ rất cao.
* Vì tốc độ thay đổi trạng thái quá nhanh, mắt người không kịp nhận biết sự thay đổi nên chỉ thấy đèn G14 luôn sáng. Việc sử dụng biến này giúp tránh lặp lại hành động toggle không mong muốn và đảm bảo đèn chỉ thay đổi trạng thái mỗi khi nút OK được bấm thực sự.

### **4.2, Code logic trong vòng lặp while:**

| *while (1)*  *{*  *Set7SegDisplayValue(display\_number); // Cập nhật giá trị hiển thị*  *Run7SegDisplay();*  *if (command\_ok)*  *{*  *char buf[256];*  *IrDecode();*  *sprintf(buf, "%02x%02x%02x%02x\n", irda\_cmd[0], irda\_cmd[1], irda\_cmd[2], irda\_cmd[3]);*  *HAL\_UART\_Transmit(&huart1, (const uint8\_t \*)buf, strlen(buf), 2);*  *command\_ok = 0;*  *}*  *if (irda\_cmd[2] == 0x45){ //1*  *display\_number = 45;*  *Running\_spot\_L();*  *last\_button = 0x45;*  *}else if (irda\_cmd[2] == 0x46){ //2*  *display\_number = 70;*  *Running\_spot\_R();*  *last\_button = 0x46;*  *}else if (irda\_cmd[2] == 0x47){ //3*  *display\_number = 71;*  *Flash();*  *last\_button = 0x47;*  *}else if (irda\_cmd[2] == 0x44){ //4*  *display\_number = 68;*  *Spot\_bumber();*  *last\_button = 0x44;*  *}else if (irda\_cmd[2] == 0x40){ //5*  *display\_number = 40;*  *last\_button = 0x40;*  *}else if (irda\_cmd[2] == 0x43){ //6*  *display\_number = 43;*  *last\_button = 0x43;*  *}else if (irda\_cmd[2] == 0x07){ //7*  *display\_number = 07;*  *last\_button = 0x07;*  *}else if (irda\_cmd[2] == 0x15){ //8*  *display\_number = 15;*  *last\_button = 0x15;*  *}else if (irda\_cmd[2] == 0x09){ //9*  *display\_number = 9;*  *last\_button = 0x09;*  *}else if (irda\_cmd[2] == 0x19){ //0*  *display\_number = 19;*  *last\_button = 0x19;*  *}else if (irda\_cmd[2] == 0x1c){ //ok*  *HAL\_GPIO\_TogglePin(GPIOG, GPIO\_PIN\_14);*  *irda\_cmd[2] = last\_button;*  *}*  *}*  *}* |
| --- |

- Giải thích:

* Đầu tiên ta dùng đoạn code có sẵn để hiện ra màn hình hercules giá trị 4byte theo hệ hexa qua cơ chế UART, Sprintf giúp lưu giá trị 4 byte vào mảng buf và sau đó gửi lên hercules qua hàm HAL\_UART\_Transmit. Lưu ý irda\_cmd[2] tương ứng với giá trị byte số 3 trong dãy 4 byte mã lệnh.
* Sau đó là các lệnh điều kiện if else để hiện ra led 7 thanh giá trị tương ứng với byte số 3 của tín hiệu thu được ở bài 3.3, đồng thời gán giá trị last\_button bằng với giá trị byte số 3 của nút đang bấm (nhằm xử lí nút OK).
* Riêng với các nút từ 1 đến 4 thì ta sẽ vừa thực hiện để led 7 thanh hiển thị giá trị byte số 3, vừa thực hiện gọi các hàm để cho các led nháy theo yêu cầu của đề bài, tương ứng với bài tự làm 3.1.
* Với nút OK có chức năng bật/tắt đèn LED4, ta gán lại giá trị irda\_cmd[2] bằng byte số 3 của tín hiệu trước khi bấm OK, để LED4 sẽ không bị vòng lặp while gọi bật tắt liên tục và đồng thời cũng không làm gián đoạn đến việc chạy chương trình