|  |  |
| --- | --- |
| Học trực tuyến CNTT, học lập trình từ cơ bản đến nâng cao |  |

**IOT303 – LẬP TRÌNH C NHÚNG NÂNG CAO CHO VI ĐIỀU KHIỂN**

**THUYẾT MINH ASSIGNMENT 1**

**Công tắc cảm ứng ĐIỀU KHIỂN ĐÓNG/MỞ RÈM CỬA**

|  |  |
| --- | --- |
| Tên học viên:  Mã học viên:  Ngày báo cáo: | Phạm Ngọc Hiệp  FX19756  16/03/2023 |

**HÀ NỘI, 03/2023**

**MỤC LỤC**

[1. Danh sách thư viện 1](#_Toc129963880)

[2. Danh sách các biến toàn cục và các macro 1](#_Toc129963881)

[3. Danh sách các hàm 3](#_Toc129963882)

[4. Hàm static void LedBuzz\_Init(void) 4](#_Toc129963883)

[5. Hàm static void Button\_Init(void) 5](#_Toc129963884)

[6. Hàm static void LedControl\_SetState(uint8\_t led\_id, uint8\_t led\_state) 5](#_Toc129963885)

[7. Hàm static void delay(uint32\_t ms) 5](#_Toc129963886)

[8. Hàm static void BuzzerControl\_SetBeep(void) 6](#_Toc129963887)

[9. Hàm static void Blinkled\_StatusPower(void) 6](#_Toc129963888)

[10. Hàm uint32\_t CalculatorTime(uint32\_t dwTimeInit, uint32\_t dwTimeCurrent) 7](#_Toc129963889)

[11. Hàm static void B3Interruppt\_Init(void) 7](#_Toc129963890)

[12. Hàm static void B2Interruppt\_Init(void) 7](#_Toc129963891)

[13. Hàm static void B4Interruppt\_Init(void) 8](#_Toc129963892)

[14. Hàm static void Interruppt\_Init(void) 8](#_Toc129963893)

[15. Hàm void EXTI4\_IRQHandler() 9](#_Toc129963894)

[16. Hàm void EXTI3\_IRQHandler() 9](#_Toc129963895)

[17. Hàm void EXTI0\_IRQHandler() 9](#_Toc129963896)

[18. Hàm void Button2\_Scan() 10](#_Toc129963897)

[19. Hàm void Button3\_Scan() 10](#_Toc129963898)

[20. Hàm void Button4\_Scan() 11](#_Toc129963899)

[21. Hàm static void AppInit (void) 11](#_Toc129963900)

[22. Hàm int main(void) 12](#_Toc129963901)

[23. Luồng xử lý code 12](#_Toc129963902)

**DANH MỤC HÌNH VẼ, BẢNG BIỂU**

**NỘI DUNG THUYẾT MINH**

# Danh sách thư viện

|  |  |  |
| --- | --- | --- |
| **STT** | **Thư viện** | **Mô tả** |
| 1. 2 | stdint.h | Thư viện lưu trữ và làm việc với các dữ liệu ở cấp độ bit  + uint8\_t : 8 bit  +uint16\_t: 16 bit  +uint32\_t: 32 bit  ….. |
| 1. 4 | system\_stm32f4xx.h | Thư viện cung cấp hàm để thao tác với stm32f4 |
| 1. 5 | timer.h | Thư viện cung cấp hàm xử lý thời gian |
| 1. 6 | misc.h | Thư viện cung cấp xử lý NVIC |
| 1. 7 | stm32f401re\_rcc.h | Thư viện cung cấp địa chỉ các thanh ghi |
| 1. 8 | stm32f401re\_gpio.h | Thư viện cung cấp cấu hình GPIO |
| 1. 9 | stm32f401re\_exti.h | Thư viện cung cấp EXTI firmware |
| 1. 10 | stm32f401re\_syscfg.h | Thư viện cung cấp SYSCFG firmware |

# Danh sách các biến toàn cục và các macro

|  |  |  |
| --- | --- | --- |
| **STT** | **Biến toàn cục và macro** | **Mô tả** |
| 1 | GPIO\_InitStructure | Kiểu dữ liệu struct GPIO\_InitTypedef để cấu hình GPIO |
| 2 | EXTI\_InitStructure | Kiểu dữ liệu struct EXTI\_InitTypedef để cấu hình EXTI |
| 3 | NVIC\_InitStructure | Kiểu dữ liệu struct NVIC\_InitTypedef để cấu hình NVIC |
| 4 | GPIO\_PIN\_SET | Macro set giá trị bằng 1 cho GPIO Pin |
| 5 | GPIO\_PIN\_RESET | Macro reset giá trị bằng 0 cho GPIO Pin |
| 6 | LED\_BOARD\_GPIO\_PORT  LED\_Green\_1\_GPIO\_PORT  LED\_Green\_2\_GPIO\_PORT  LED\_BLUE\_1\_GPIO\_PORT  LED\_BLUE\_2\_GPIO\_PORT  LED\_RED\_1\_GPIO\_PORT  LED\_RED\_2\_GPIO\_PORT  BUZZER\_GPIO\_PORT  BUTTONB2\_IT\_GPIO\_PORT  BUTTONB4\_IT\_GPIO\_PORT  BUTTONB3\_IT\_GPIO\_PORT | Macro gắn giá trị Port GPIO cho ngoại vi |
| 6 | LED\_BOARD\_GPIO\_PIN  LED\_Green\_1\_GPIO\_ PIN  LED\_Green\_2\_GPIO\_ PIN  LED\_BLUE\_1\_GPIO\_ PIN  LED\_BLUE\_2\_GPIO\_ PIN  LED\_RED\_1\_GPIO\_ PIN  LED\_RED\_2\_GPIO\_ PIN  BUZZER\_GPIO\_ PIN  BUTTONB2\_IT\_GPIO\_ PIN  BUTTONB4\_IT\_GPIO\_ PIN  BUTTONB3\_IT\_GPIO\_ PIN | Macro gắn giá trị PIN GPIO cho ngoại vi |
| 7 | LED\_BOARD\_GPIO\_PIN5  LED\_Green\_1\_PIN0  LED\_Green\_2\_PIN11  LED\_BLUE\_1\_GPIO\_PIN10  LED\_BLUE\_2\_GPIO\_PIN3  LED\_RED\_1\_GPIO\_PIN13  LED\_RED\_2\_GPIO\_PIN1 | Giá trị của Pin mà ngoại vi kết nối |
| 8 | LED\_BOARD\_GPIO\_SetClock  LED\_Green\_1\_GPIO\_ SetClock  LED\_Green\_2\_GPIO\_ SetClock  LED\_BLUE\_1\_GPIO\_ SetClock  LED\_BLUE\_2\_GPIO\_ SetClock  LED\_RED\_1\_GPIO\_ SetClock  LED\_RED\_2\_GPIO\_ SetClock  BUZZER\_GPIO\_ SetClock  BUTTONB2\_IT\_GPIO\_ SetClock  BUTTONB4\_IT\_GPIO\_ SetClock  BUTTONB3\_IT\_GPIO\_ SetClock | Macro gắn giá trị thanh ghi cấp Clock cho ngoại vi |
| 9 | uint32\_t releaseTimeB3 = 0;  uint32\_t pressTimeB3 = 0;  uint32\_t releaseTimeB2 = 0;  uint32\_t pressTimeB2 = 0;  uint32\_t releaseTimeB4 = 0;  uint32\_t pressTimeB4 = 0; | Biến thuộc kiểu dữ liệu uint32\_t cho biết thời gian nhấn nhả của nút B2/3/4 |
| 10 | uint8\_t countB3 = 0;  uint8\_t countB2 = 0;  uint8\_t countB4 = 0; | Biến thuộc kiểu dữ liệu uint8\_t đếm số lần nhả nút B2/3/4 |

# Danh sách các hàm

|  |  |  |
| --- | --- | --- |
| **STT** | **Danh sách hàm** | **Mô tả** |
| 1 | int main() | Hàm chính của chương trình |
| 2 | static void LedBuzz\_Init(void) | Hàm khởi tạo các chân điều khiển Led và Buzzer |
| 3 | static void Button\_Init(void) | Hàm khởi tạo các chân điều khiển nút nhấn |
| 4 | static void LedControl\_SetState(uint8\_t led\_id, uint8\_t led\_state) | Hàm điều khiển bật/tắt Led. |
| 5 | static void delay(uint32\_t ms) | Hàm điều khiển led nháy |
| 6 | static void BuzzerControl\_SetBeep(void) | Hàm điều khiển bật/tắt còi |
| 7 | static void Blinkled\_StatusPower(void) | Hàm hiển thị trạng thái led GREEN bật/tắt bốn lần khi cấp nguồn cho thiết bị |
| 8 | uint32\_t CalculatorTime(uint32\_t dwTimeInit, uint32\_t dwTimeCurrent) | Hàm tính thời gian nhấn nút với tham số truyền vào là thời gian khi bắt đầu nhấn nút và thời gian hiện tại |
| 9 | static void B3Interruppt\_Init(void) | Hàm cấu hình trình phục ngắt cho nút B3 |
| 10 | static void B2Interruppt\_Init(void) | Hàm cấu hình trình phục ngắt cho nút B2 |
| 11 | static void B4Interruppt\_Init(void) | Hàm cấu hình trình phục ngắt cho nút B4 |
| 12 | static void Interruppt\_Init(void) | Hàm gọi các hàm cấu hình trình phục ngắt cho nút bấm |
| 13 | void EXTI4\_IRQHandler() | Hàm xử lý ngắt từ bộ xử lý ngoại vi EXTI cho chân GPIO số 4 |
| 14 | void EXTI3\_IRQHandler() | Hàm xử lý ngắt từ bộ xử lý ngoại vi EXTI cho chân GPIO số 3 |
| 15 | void EXTI0\_IRQHandler() | Hàm xử lý ngắt từ bộ xử lý ngoại vi EXTI cho chân GPIO số 0 |
| 16 | void Button2\_Scan() | Hàm xử lý trả về sự kiện nút bấm B2 |
| 17 | void Button3\_Scan() | Hàm xử lý trả về sự kiện nút bấm B3 |
| 18 | void Button4\_Scan() | Hàm xử lý trả về sự kiện nút bấm B4 |
| 18 | static void AppInit (void) | Hàm khởi tạo các hàm cấu hình |

# Hàm static void LedBuzz\_Init(void)

1. *Mô tả*

Hàm static void LedBuzz\_Init(void) thực hiện việc cấu hình các chân điều khiển của module led và buzzer

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Không có giá trị trả về.

# Hàm static void Button\_Init(void)

1. *Mô tả*

Hàm static void Button\_Init(void) thực hiện cấu hình các nút nhấn B2, B3, B4 với chức năng ngắt.

1. *Tham số truyền vào*

Không có tham số truyền vào

1. *Giá trị trả về*

Không có giá trị trả về.

# Hàm static void LedControl\_SetState(uint8\_t led\_id, uint8\_t led\_state)

1. *Mô tả*

Hàm static void LedControl\_SetState(uint8\_t led\_id, uint8\_t led\_state) thực hiện việc điều khiển bật/tắt Led.

1. *Tham số truyền vào*

**led\_id**: id của led cần điều khiển.

**led\_state**: trạng thái điều khiển bật/tắt led.

1. *Giá trị trả về*

Không có giá trị trả về.

# Hàm static void delay(uint32\_t ms)

1. *Mô tả*

Hàm static void delay(uint32\_t ms) thực hiện việc delay thời gian thực hiện hàm tiếp theo, dùng để nháy led.

1. *Tham số truyền vào*

uint32\_t ms : Thời gian delay

1. *Giá trị trả về*

Không có giá trị trả về

# Hàm static void BuzzerControl\_SetBeep(void)

1. *Mô tả*

Hàm static void BuzzerControl\_SetBeep(void) thực hiện điều khiển buzzer kêu

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Không có giá trị trả về

# Hàm static void Blinkled\_StatusPower(void)

1. *Mô tả*

Hàm static void Blinkled\_StatusPower(void) thực hiện điểu khiển bật/tắt Led trên mạch 4 lần để thông báo trạng thái của thiết bị đã được cấp nguồn

1. *Tham số truyền vào*

Không có tham số truyền vào

1. *Giá trị trả về*

Không có giá trị trả về

# Hàm uint32\_t CalculatorTime(uint32\_t dwTimeInit, uint32\_t dwTimeCurrent)

1. *Mô tả*

Hàm uint32\_t CalculatorTime(uint32\_t dwTimeInit, uint32\_t dwTimeCurrent) thực hiện tính thời gian nhấn nút với tham số truyền vào là thời gian khi bắt đầu nhấn nút và thời gian hiện tại.

1. *Tham số truyền vào*

uint32\_t dwTimeInit: thời gian bắt đầu nhấn nút

uint32\_t dwTimeCurrent: thời gian hiện tại

1. *Giá trị trả về*

uint32\_t dwTimeTotal: thời gian nhấn nút

# Hàm static void B3Interruppt\_Init(void)

1. *Mô tả*

Hàm static void B3Interruppt\_Init(void) thực hiện cấu hình trình phục ngắt cho nút B3.

1. *Tham số truyền vào*

Hàm không có tham số truyền vào

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm static void B2Interruppt\_Init(void)

1. *Mô tả*

Hàm static void B2Interruppt\_Init(void) thực hiện cấu hình trình phục ngắt cho nút B3.

1. *Tham số truyền vào*

Hàm không có tham số truyền vào

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm static void B4Interruppt\_Init(void)

1. *Mô tả*

Hàm static void B4Interruppt\_Init(void) thực hiện cấu hình trình phục ngắt cho nút B3.

1. *Tham số truyền vào*

Hàm không có tham số truyền vào

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm static void Interruppt\_Init(void)

1. *Mô tả*

Hàm static void Interruppt\_Init(void) thực hiện gọi các hàm cấu hình trình phục ngắt cho nút bấm.

1. *Tham số truyền vào*

Hàm không có tham số truyền vào

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm void EXTI4\_IRQHandler()

1. *Mô tả*

Hàm void EXTI4\_IRQHandler() thực hiện xử lý ngắt từ bộ xử lý ngoại vi EXTI cho chân GPIO số 4.

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm void EXTI3\_IRQHandler()

1. *Mô tả*

Hàm void EXTI3\_IRQHandler() thực hiện xử lý ngắt từ bộ xử lý ngoại vi EXTI cho chân GPIO số 3.

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm void EXTI0\_IRQHandler()

1. *Mô tả*

Hàm void EXTI0\_IRQHandler() thực hiện xử lý ngắt từ bộ xử lý ngoại vi EXTI cho chân GPIO số 0.

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm void Button2\_Scan()

1. *Mô tả*

Hàm void Button2\_Scan() thực hiện xử lý sự kiện trả về nút bấm B2.

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm void Button3\_Scan()

1. *Mô tả*

Hàm void Button3\_Scan() thực hiện xử lý sự kiện trả về nút bấm B3.

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm void Button4\_Scan()

1. *Mô tả*

Hàm void Button4\_Scan() thực hiện xử lý sự kiện trả về nút bấm B4.

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm static void AppInit (void)

1. *Mô tả*

Hàm **static void AppInit (void)** thực hiện việc khởi tạo các tài nguyên cần sử dụng

Hàm **SystemCoreClockUpdate** để cấu hình clock của hệ thống là 84Mhz

Hàm **TimerInit** để khởi tạo timer system tick của hệ thống.

Hàm **Button\_Init** để cấu hình chân GPIO của các nút nhấn trên mạch.

Hàm **LedBuzzer\_Init** để cấu hình chân GPIO của các led và còi trên mạch.

Hàm **Interruppt\_Init** để cấu hình trình phục ngắt cho nút bấm

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm int main(void)

1. *Mô tả*

Hàm **int main(void)** thực hiện việc gọi hàm :

**App\_Init ()** để khởi tạo các tài nguyên cần sử dụng

**Blinkled\_StatusPower**: Đảo trạng thái led bốn lần báo hiệu thiết bị đã được cấp nguồn.

Trong hàm while(1) gọi các hàm:

**Button3\_Scan()** để xử lý các sư kiện từ nút B3.

**Button2\_Scan()** để xử lý các sư kiện từ nút B2.

**Button4\_Scan()** để xử lý các sư kiện từ nút B4.

1. *Tham số truyền vào*

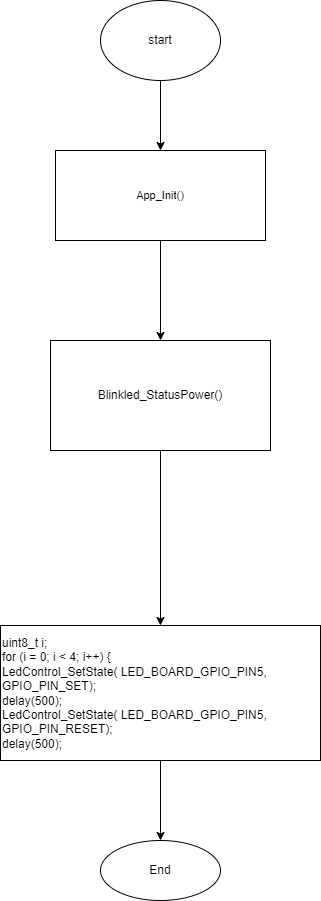
Không có tham số truyền vào

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Luồng xử lý code

**Câu 1** Thiết bị sẽ nháy led GREEN trên Board STM32 bốn lần để thông báo trạng thái của thiết bị đã được cấp nguồn.



**Các bước thực xử lý:**

***Bước 1:*** Chương trình chạy hàm main :

* Chạy hàm App\_Init để khởi tạo các tài nguyên cần sử dụng
* Trong hàm App\_Init chạy hàm :

+ Chạy hàm Hàm **SystemCoreClockUpdate** để cấu hình clock của hệ thống là 84Mhz

+Chạy hàm **TimerInit** để khởi tạo timer system tick của hệ thống.

+Chạy hàm **Button\_Init** để cấu hình chân GPIO của các nút nhấn trên mạch.

+Chạy hàm **LedBuzzer\_Init** để cấu hình chân GPIO của các led và còi trên mạch.

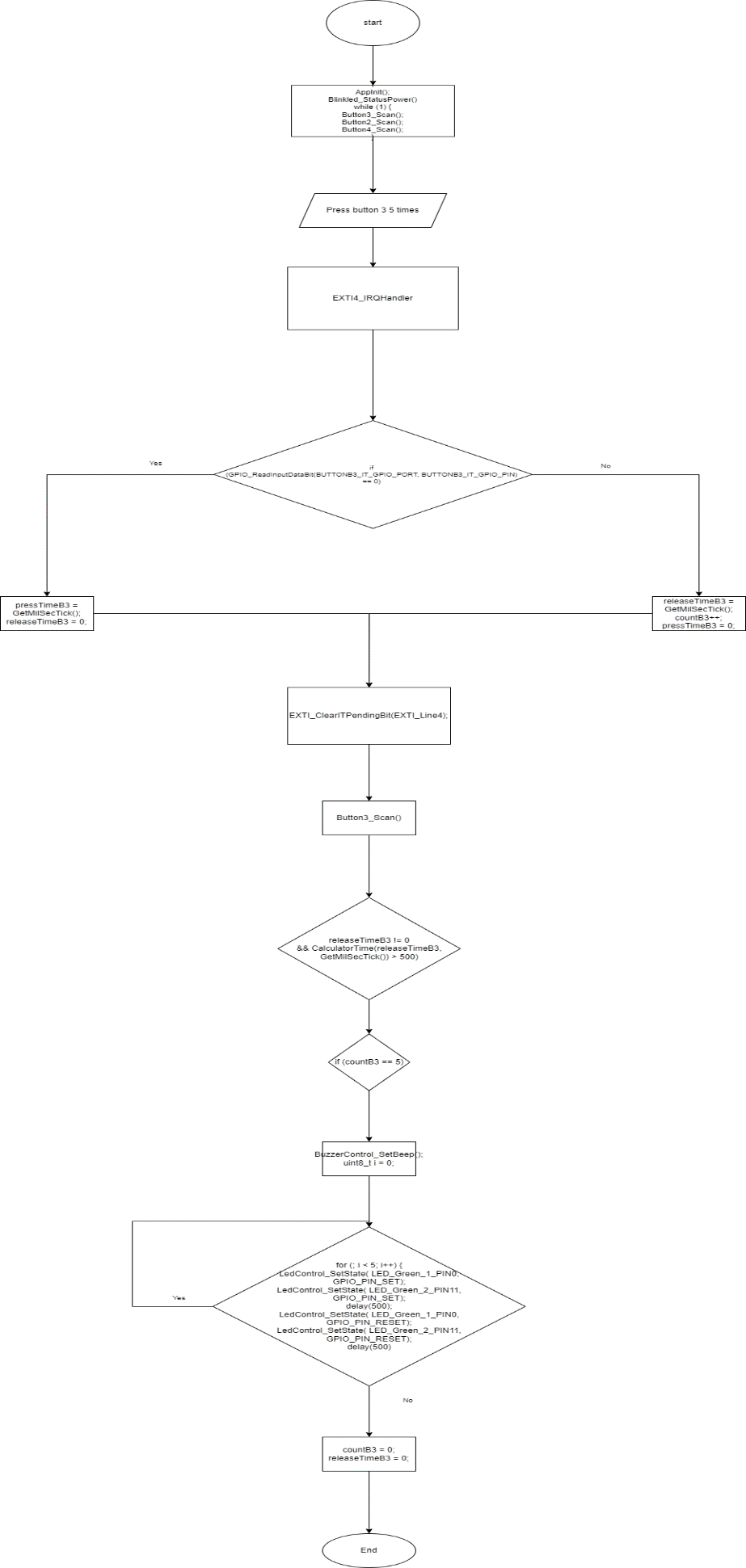
+Chạy hàm **Interruppt\_Init** để cấu hình trình phục ngắt cho nút bấm

* Chạy hàm **Blinkled\_StatusPower**

***Bước 2 :*** Trong hàm **Blinkled\_StatusPower** với:

* Tạo vòng lặp for với biến uint8\_t i có giá trị bằng 0 , giá trị i < 4 thực hiện vòng lặp
* Trong vòng lặp sử dụng hàm **LedControl\_SetState**và **delay** để điều khiển led bật tắt 4 lần

**Câu 2** Nhấn nút B3 năm lần sẽ điều khiển nháy tất cả các led RGB màu GREEN năm lần và còi sẽ kêu bíp hai lần để thông báo thiết bị đang gia nhập mạng.



**Các bước thực xử lý:**

***Bước 1:*** Trong hàm main:

* Tại vòng lặp while thực hiện hàm Button3\_Scan để quét sự kiện xảy ra
* Khi người sử dụng bấm nút B3 5 lần -> chuyển sang hàm **EXTI4\_IRQHandler**

***Bước 2 :*** Tại hàm **EXTI4\_IRQHandler:**

* Thực hiện kiểm tra điều kiện :

+ Trường hợp nút B3 được nhấn -> Gắn giá trị biến pressTimeB3 với hàm GetMilSecTick để lấy thời gian khi nút được nhấn. -> Gắn giá trị của biến releaseTimeB3 = 0

+ Trường hợp nút B3 được nhả ra -> Gắn giá trị biến releaseTime B3 với hàm GetMilSecTick để lấy thời gian khi nút được nhả ra -> Gắn giá trị của biến pressTimeB3 = 0-> Thêm 1 giá trị cho biến đếm countB3.

* Sử dụng hàm EXTI\_ClearITPendingBit(EXTI\_Line4) để xóa bit thoát khỏi ngắt.
* Chuyển sang hàm -> Button3\_Scan

***Bước 3:*** Tại hàm **Button3\_Scan:**

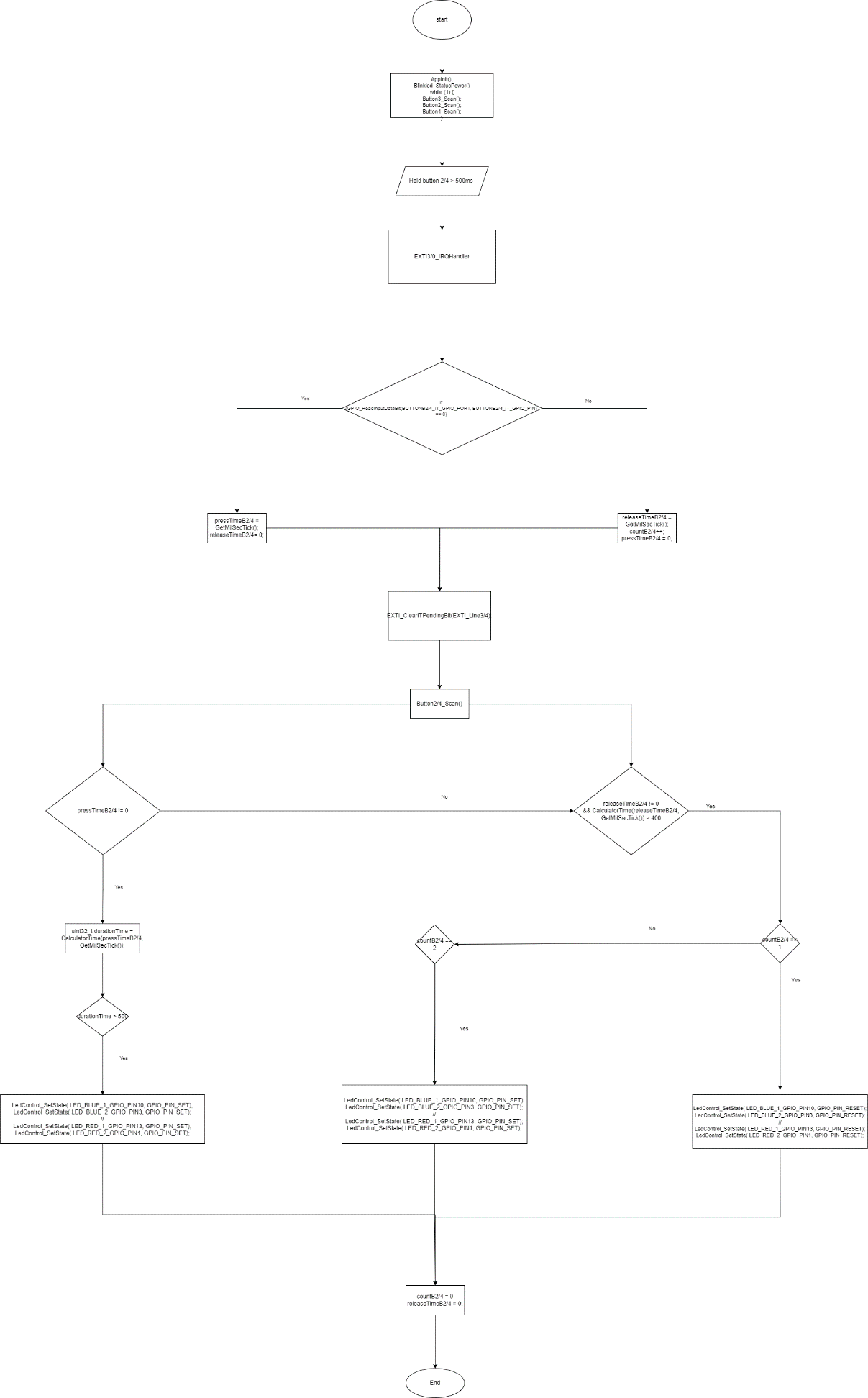
* Thực hiện kiểm tra điều kiện nếu biến releaseTime B3 khác giá trị 0 và thời releaseTime > 500ms
* Kiểm tra điều kiện biến countB3 có bằng năm không

+ Nếu đúng thực hiện bật còi bằng hàm BuzzerControl\_SetBeep

+ Thực hiện bật tắt Led qua LedControl\_SetState và hàm delay trong vòng lặp for

* Đặt giá trị biến đếm countB3 = 0 và biến releasetimeB3 = 0

**Câu 3 :** Nhấn giữ nút B2/B4 để điều khiển đóng/mở rèm thông qua mô phỏng trạng thái bật/tắt các led RGB:



**Các bước thực xử lý:**

***Bước 1*** :

* Tương tự như câu 2 thay **Button3\_Scan** bằng hàm **Button2\_Scan** hoặc **Button4\_Scan** tùy thuộc vào việc thực hiện nút B2 hay B4.

***Bước 2 :***

* Tương tự như câu 2 thay hàm **EXTI4\_IRQHandler** bằng hàm **EXTI3\_IRQHandler** hoặc hàm **EXTI0\_IRQHandler** tùy thuộc vào việc thực hiện nút B2 hay B4

***Bước 3 :*** Tại hàm **Button2\_Scan** hoặc **Button4\_Scan:**

* Kiểm tra biến pressTimeB2/B4 có giá trị khác 0 không

+ Nếu đúng thực hiện tạo biến thời gian giữ nút durationTime gắn giá trị bằng hàm CalculatorTime với tham số truyền vào là pressTimeB2/B4 và GetMilSecTick.

+ Thực hiện kiểm tra biến durationTime > 400 ms không . Nếu có thực hiện hàm LedControl\_SetState để bật Led Blue/Red tùy thuộc vào việc nhấn giữ nút nào

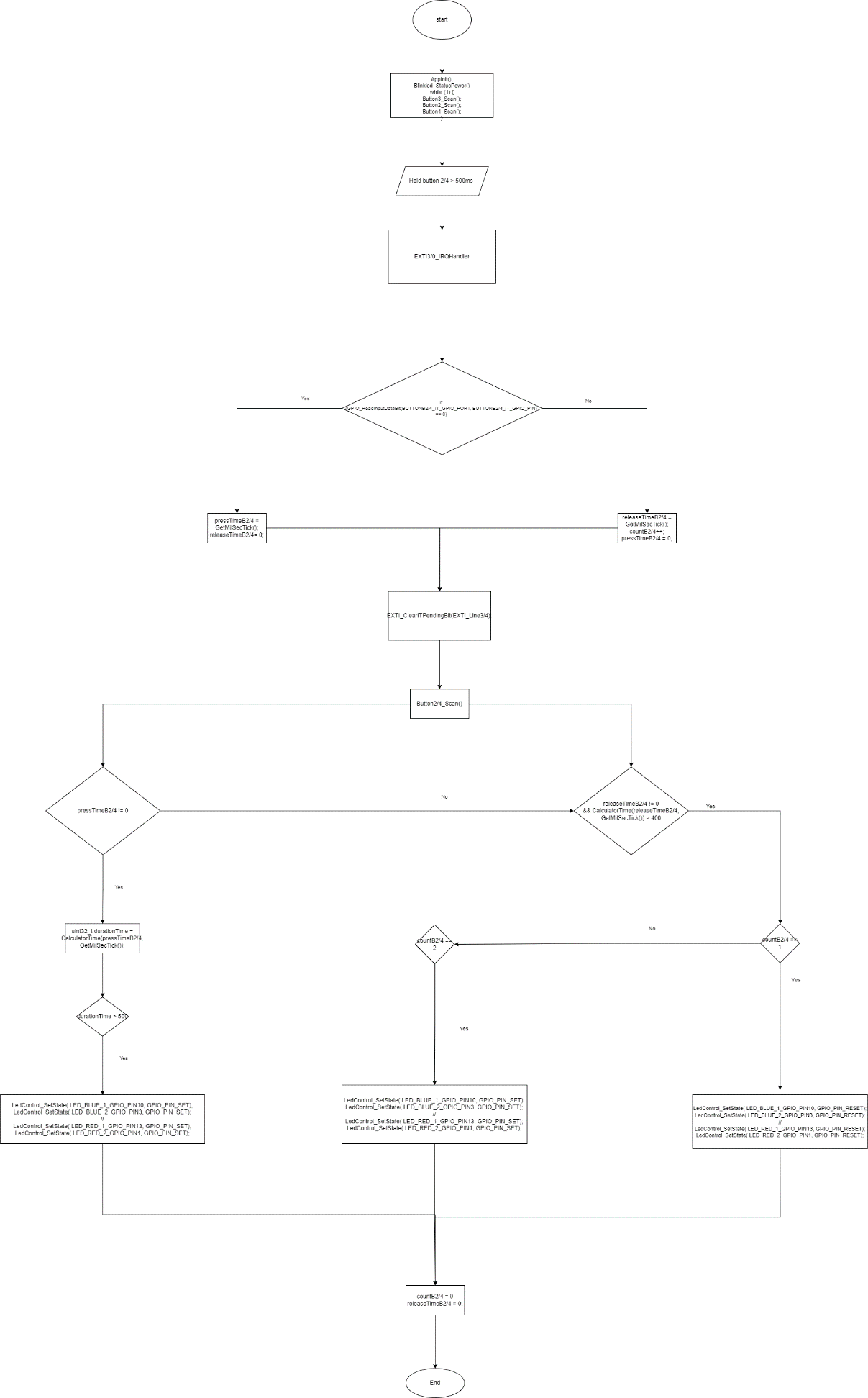
* Khi nhả tay thực hiện kiểm tra releaseTimeB2/B4 có giá trị khác 0 không

+ Nếu đúng kiểm tra biến countB2/B4 có bằng 1 và thời gian nhả có >500ms không

+ Nếu đúng sử dụng hàm LedControl\_SetState để tắt Led.

* Đặt giá trị biến đếm countB2/4 = 0 và biến releasetimeB2/4 = 0 .

**Câu 4** Nhấn nút B2/B4 hai lần sẽ điều khiển đóng/mở rèm thông qua trạng thái bật/tắt led RGB.



**Các bước thực xử lý:**

***Bước 1*** :

* Tương tự như câu 3 thay **Button2\_Scan** hoặc **Button4\_Scan** tùy thuộc vào việc thực hiện nút B2 hay B4.

***Bước 2 :***

* Tương tự như câu 3 thay hàm **EXTI3\_IRQHandler** hoặc hàm **EXTI0\_IRQHandler** tùy thuộc vào việc thực hiện nút B2 hay B4

***Bước 3 :*** Tại hàm **Button2\_Scan** hoặc **Button4\_Scan:**

* Khi bấm B2/B4 2 lần cách nhau 400ms thực hiện kiểm tra releaseTimeB2/B4 có giá trị khác 0 không và thời gian nhấn bấm có > 500ms không
* Trường hợp biến countB2/B4 bằng 2

+ Thực hiện sử dụng hàm LedControl\_SetState để bật Led.

* Trường hợp biến countB2/B4 bằng 1

+ Thực hiện sử dụng hàm LedControl\_SetState để tắt Led

* Đặt giá trị biến đếm countB2/4 = 0 và biến releasetimeB2/4 = 0 .