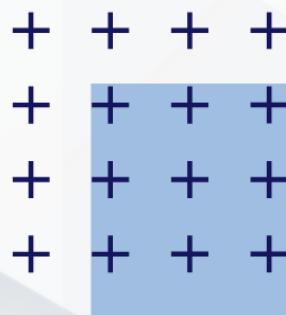
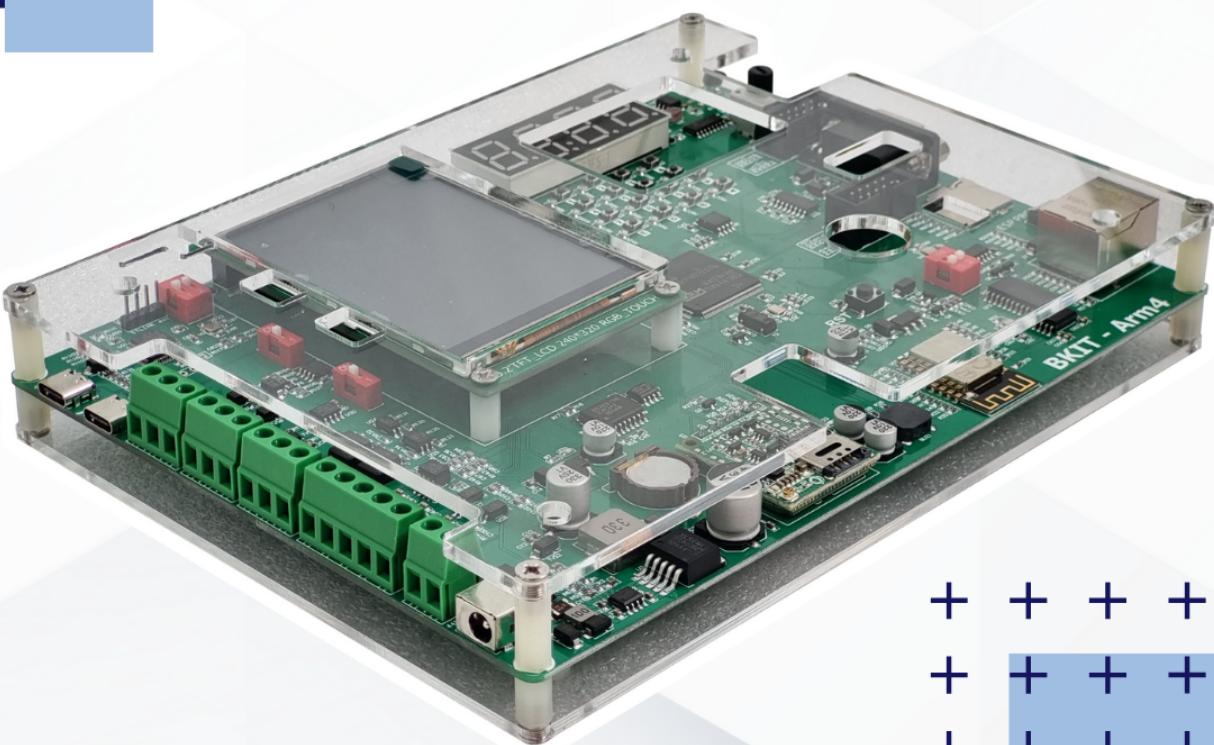


# KIT THÍ NGHIỆM BKIT

# ARM4



---

# Mục lục

---

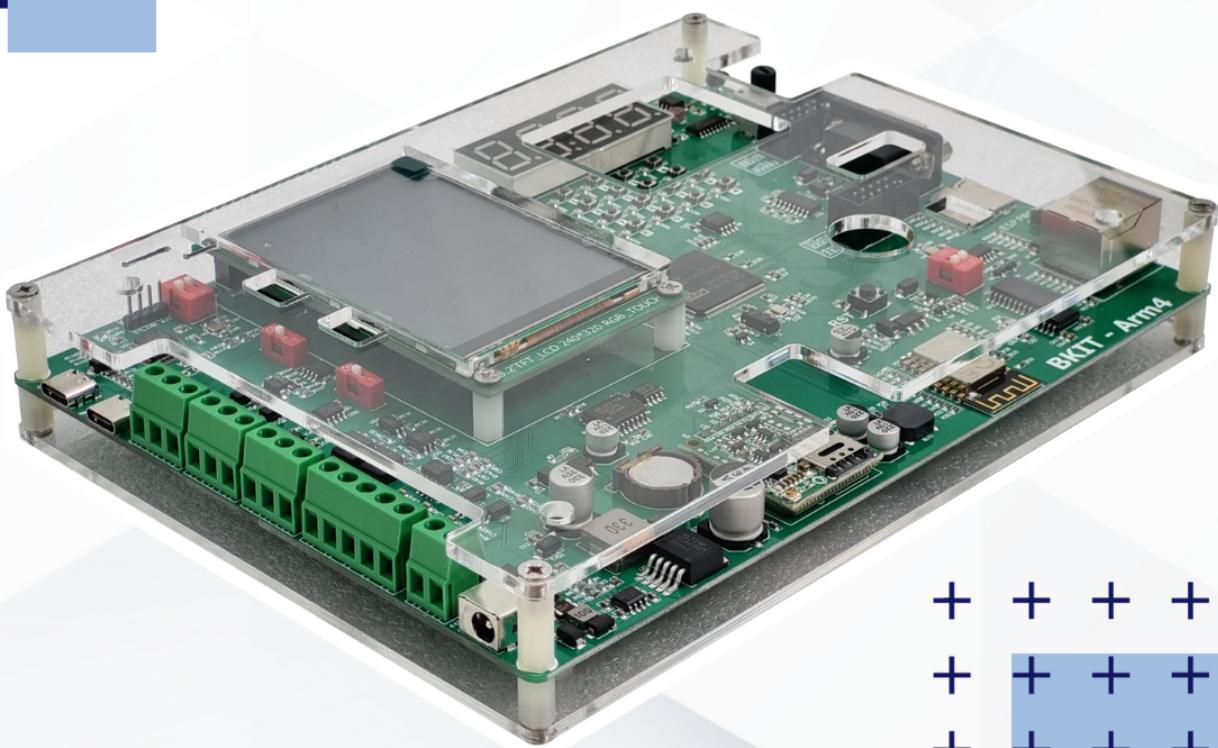
<b>Chapter 1. General Purpose Input Output</b>	<b>3</b>
1    Mục tiêu . . . . .	4
2    Giới thiệu . . . . .	4
3    Project đầu tiên trên STM32Cube . . . . .	5
4    Cơ sở lý thuyết . . . . .	10
4.1    GPIO Output Mode . . . . .	10
4.2    GPIO Input Mode . . . . .	11
5    Hướng dẫn config . . . . .	12
6    Bài tập và Báo cáo . . . . .	18
6.1    Bài tập 1 . . . . .	18
6.2    Bài tập 2 . . . . .	18
6.3    Bài tập 3 . . . . .	18

# CHƯƠNG 1

---

## General Purpose Input Output

---



# 1 Mục tiêu

- Biết cách sử dụng phần mềm STM32CubeIDE để xây dựng ứng dụng trên vi điều khiển.
- Biết cách config và lập trình các chân GPIO.
- Biết cách điều khiển các ngoại vi liên quan đến GPIO trên kit thí nghiệm.

# 2 Giới thiệu

Trong hướng dẫn này, STM32CubeIDE được sử dụng làm trình soạn thảo để lập trình vi điều khiển ARM. STM32CubeIDE là platform nổi bật với các tính năng config ngoại vi, sinh code, biên dịch và gỡ lỗi cho vi điều khiển và vi xử lý STM32.



Hình 1.1: Phần mềm STM32Cube IDE

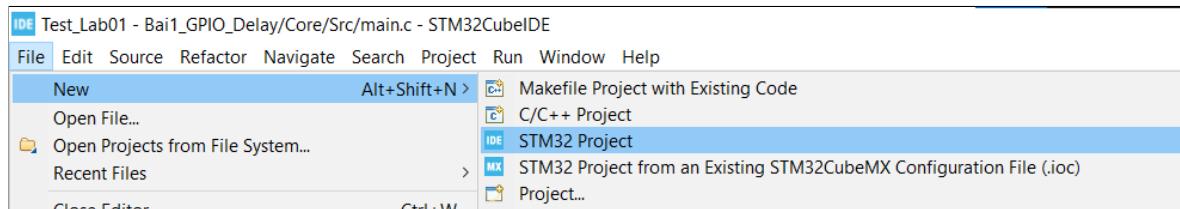
Điều thú vị nhất của STM32CubeIDE là sau khi chọn một MPU hoặc MCU STM32, hay một bộ vi điều khiển, bộ vi xử lý được config sẵn từ việc chọn bo mạch, mã khởi tạo sẽ được tạo tự động. Bất cứ khi nào trong quá trình lập trình, người dùng đều có thể quay lại quá trình khởi tạo và thay đổi config của các thiết bị ngoại vi. Việc config này không ảnh hưởng đến phần code đã được người lập trình viết trước đó. Tính năng này có thể đơn giản hóa quá trình khởi tạo các ngoại vi và dễ dàng phát triển các ứng dụng trên vi điều khiển STM32. Phần mềm có thể được tải xuống từ liên kết dưới đây:

[https://ubc.sgp1.digitaloceanspaces.com/BKU\\_Softwares/STM32/stm32cubeide\\_1.7.0.zip](https://ubc.sgp1.digitaloceanspaces.com/BKU_Softwares/STM32/stm32cubeide_1.7.0.zip)

Phần còn lại của hướng dẫn này bao gồm tạo một project trên STM32Cube IDE, chạy kiểm thử trên mạch thực. Sau đó, sinh viên sẽ hoàn thành một số bài tập để hiểu thêm.

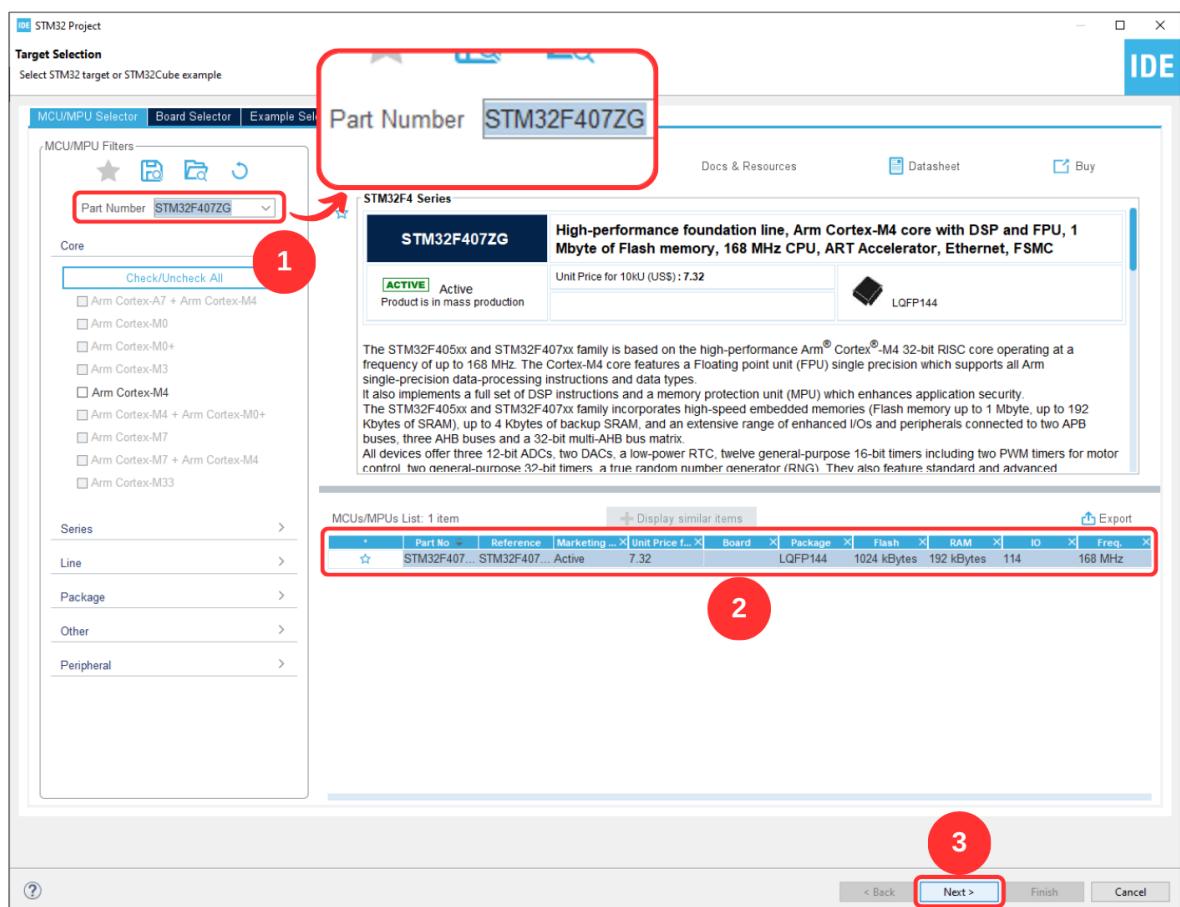
### 3 Project đầu tiên trên STM32Cube

**Bước 1:** Chạy phần mềm STM32CubeIDE, chọn menu **File**, chọn **New**, sau đó chọn **STM32 Project**. STM32CubeIDE sẽ cần tải xuống một số packages, thường tốn một ít thời gian trong lần đầu tiên tạo một dự án mới.



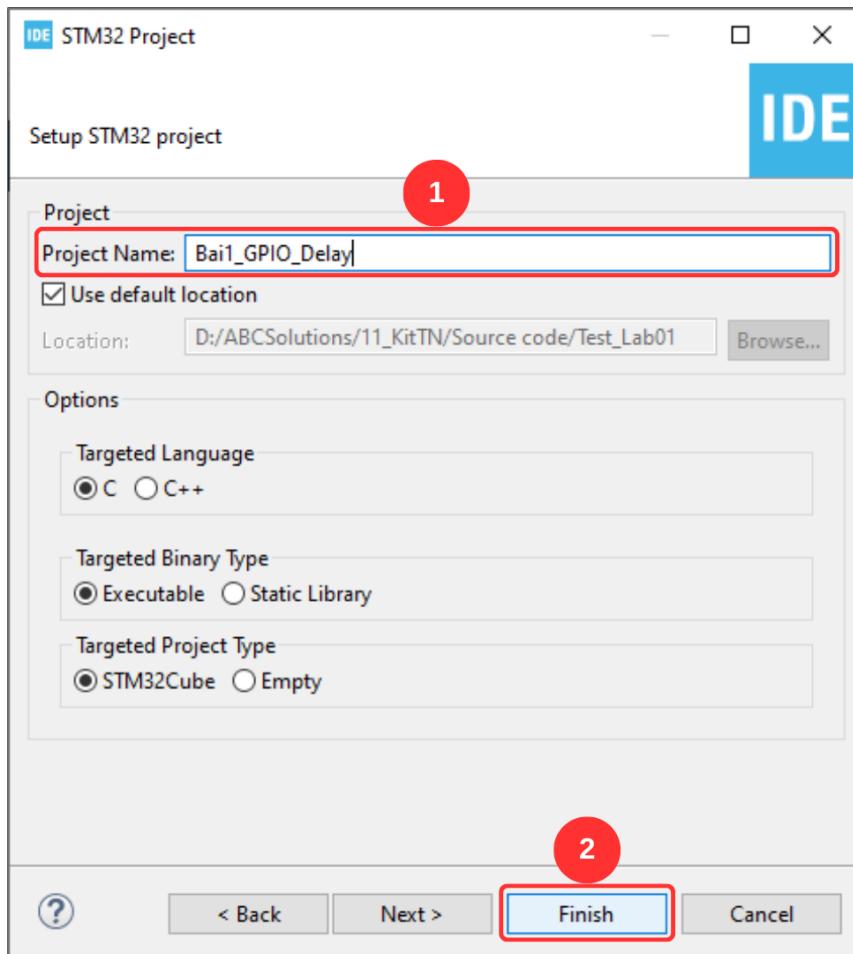
Hình 1.2: Tạo một project mới trên STM32CubeIDE

**Bước 2:** Tìm chip STM32F407ZG. Để dễ dàng tìm, chúng ta nhập tên vi điều khiển trong thanh tìm kiếm **Part Number**. Sau đó, chọn chip tại phần **MCUs/MPUs List** và chọn next để tiếp tục.



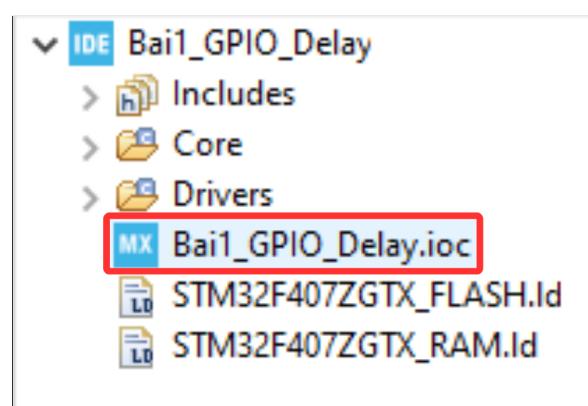
Hình 1.3: Tìm kiếm chip

**Bước 3:** Đặt tên project trong **Project Name** và đường dẫn lưu project trong **Location**. Lưu ý: Đường dẫn không được chứa ký tự tiếng Việt và ký tự đặc biệt (ví dụ như space).



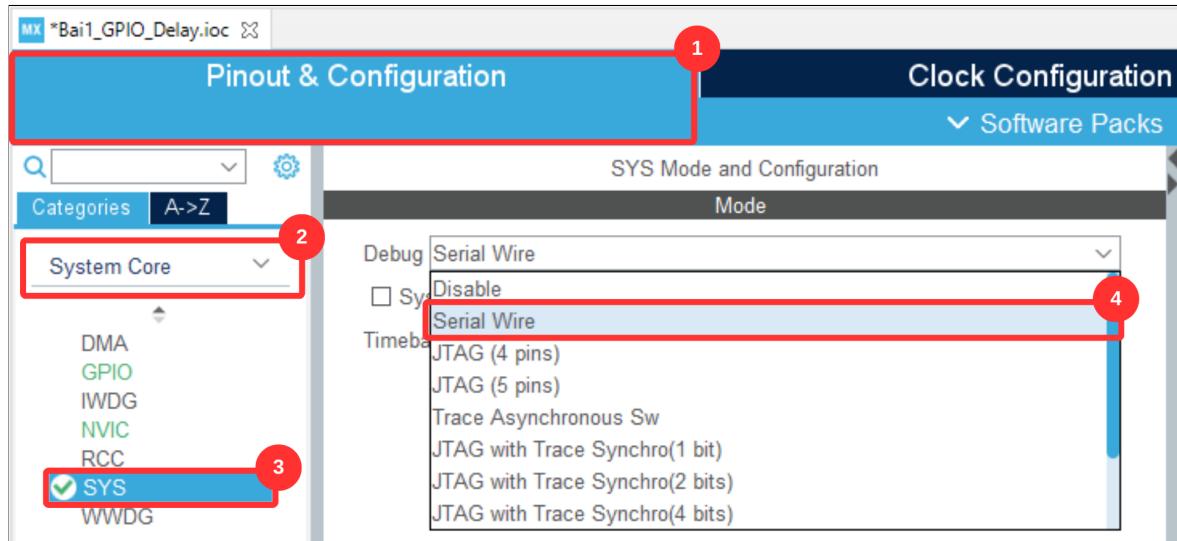
Hình 1.4: Đặt tên và tìm vị trí lưu project

**Bước 4:** Sau khi tạo xong project, màn hình config được hiển thị (file .ioc). Tính năng này của CubeIDE có thể đơn giản hóa quy trình config cho bộ vi điều khiển ARM như STM32. Trường hợp file chưa được mở, ta có thể mở file trong project.



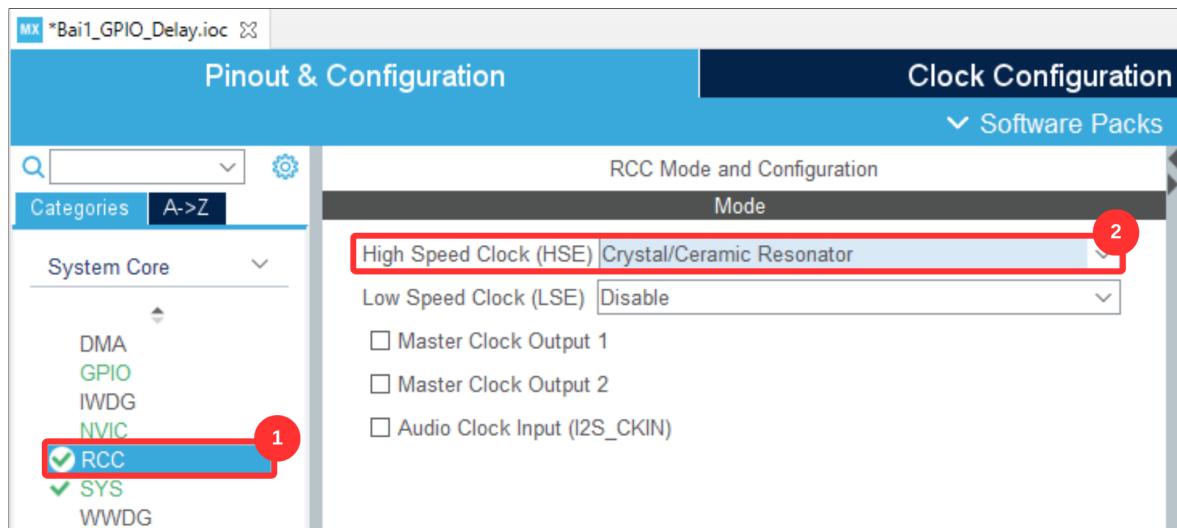
Hình 1.5: File config các chân của vi điều khiển

### Bước 5: Điều chỉnh phương thức gõ lỗi:



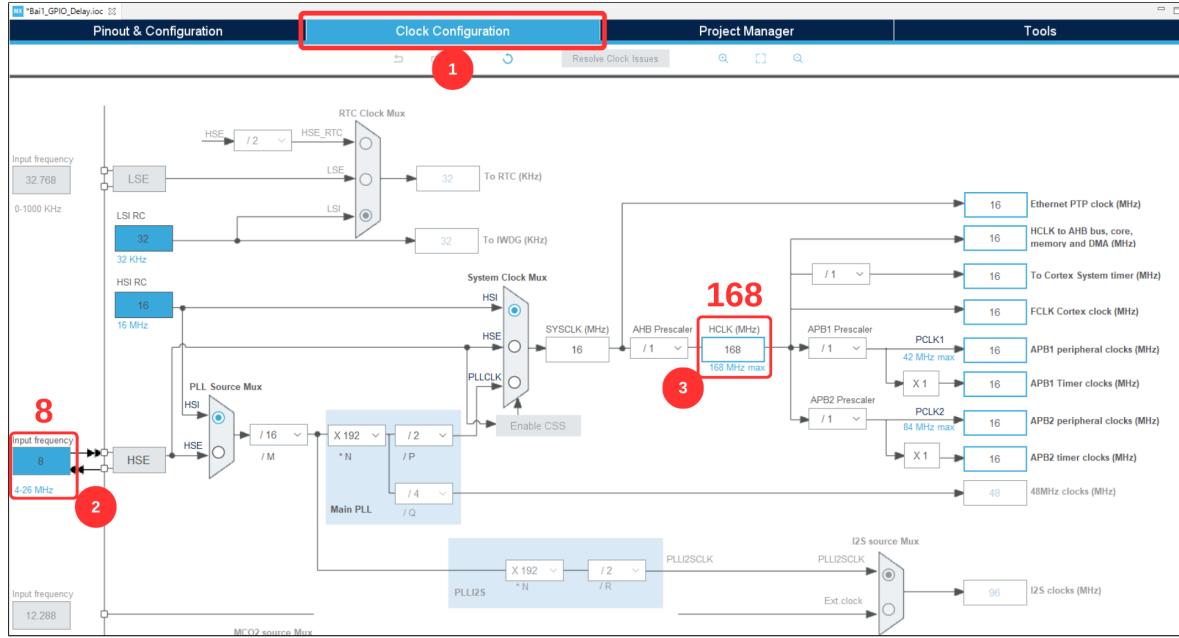
Hình 1.6: Chọn phương thức gõ lỗi

**Bước 6:** Thạch anh ngoài giúp tối ưu hóa tần số hoạt động của vi điều khiển. Để sử dụng thạch anh ngoài, chúng ta sẽ cần config trong file .ioc như hình 1.7.



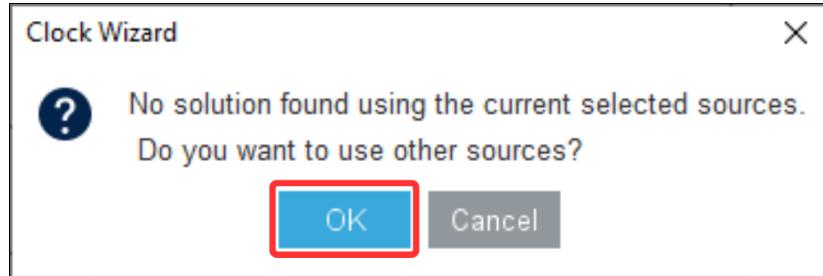
Hình 1.7: Config thạch anh ngoài

Chỉnh tần số trong cửa sổ **Clock Configuration**. Vì kit thí nghiệm sử dụng thạch anh ngoài có thông số **8MHz** và chúng ta muốn điều chỉnh tối đa tần số hoạt động của kit thí nghiệm (**168MHz**) nên ta sẽ điều chỉnh các thông số như hình 1.8.



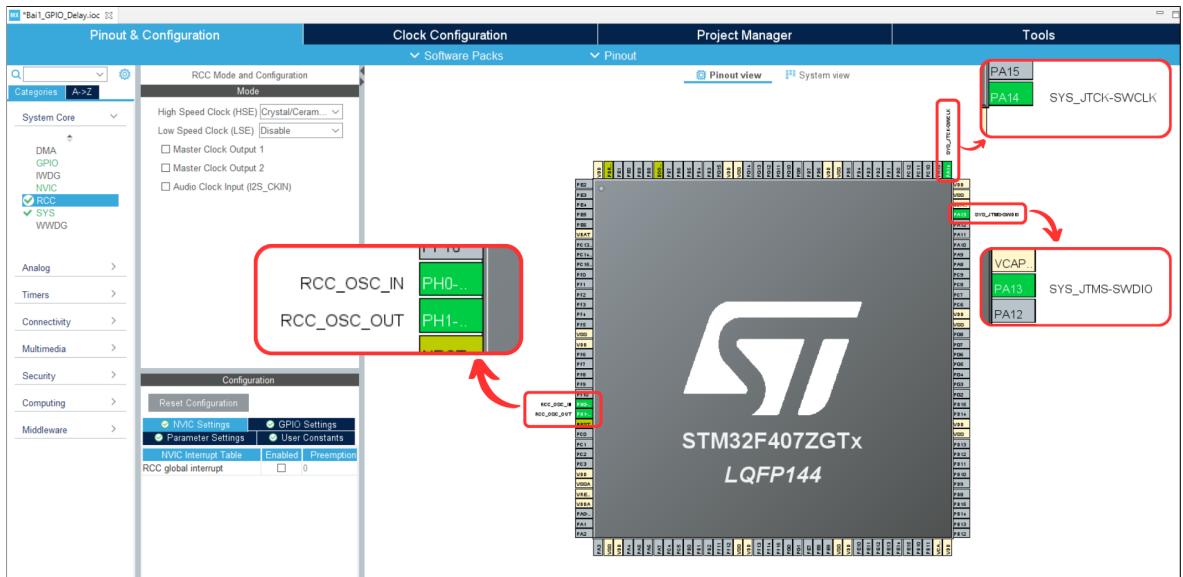
Hình 1.8: Config thạch anh ngoài

Chọn OK và đợi tính toán:



Hình 1.9: Config thạch anh ngoài

Sau khi config xong thì có 4 chân được config như hình 1.10. Sau đó, bấm tổ hợp phím **Ctrl + S** để phần mềm sinh code.



Hình 1.10: Vị trí điều khiển sau khi config

## 4 Cơ sở lý thuyết

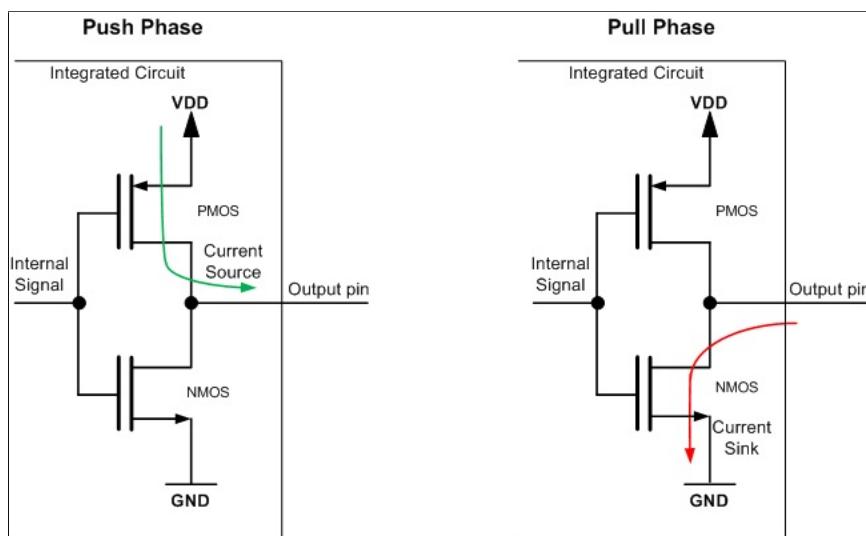
Trong phần này, chúng ta sẽ tìm hiểu về GPIO (viết tắt của General Purpose Input Output), là một thuật ngữ phổ biến trong lĩnh vực nhúng. Tín hiệu trên các chân GPIO cho phép thực hiện chức năng và giao tiếp với các thiết bị bên ngoài.

### 4.1 GPIO Output Mode

GPIO được sử dụng để truyền tín hiệu điện (cao hoặc thấp) đến chân khi nó được config làm output. Có hai tùy chọn config chủ yếu:

- **Push-pull**

- Trạng thái này là trạng thái mặc định của chế độ đầu ra GPIO. Chân này có thể “push” tín hiệu lên cao hoặc “pull” tín hiệu xuống thấp bằng cách sử dụng transistor PMOS hoặc transistor NMOS.
- Không cần điện trở kéo lên hay kéo xuống vì các transistor đã thực hiện công việc đó.

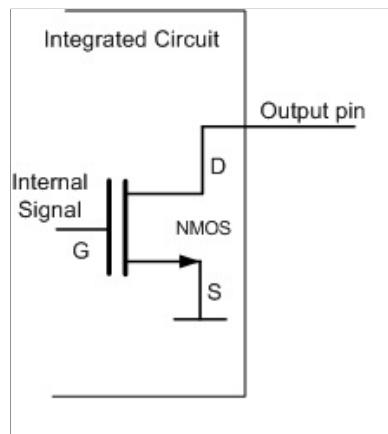


Hình 1.11: Sơ đồ config của push-pull

- **Open-drain**

- Trong Output mode, có thể sử dụng transistor PMOS và NMOS để tạo bộ đệm đầu ra. Khi loại bỏ transistor PMOS, trạng thái này sẽ trở thành "open-drain".
- Tên gọi "open-drain" xuất phát từ việc MOSFET không có kết nối nội bộ với bất kỳ điểm nào.

- Khi ta đưa NMOS lên mức cao, nó sẽ cung cấp một mức đất (GND), đồng thời đầu ra GPIO sẽ ở mức thấp.
- Khi tắt NMOS, đầu ra GPIO sẽ ở trạng thái nổi, không kết nối hoặc nối với Vcc hoặc GND. Do đó, đầu ra có thể là mức thấp hoặc mức cao trôi (nổi), có khả năng kéo chân xuống mức đất, nhưng không thể đẩy lên mức cao.
- Chế độ open-drain thường được sử dụng trong các giao diện truyền thông, nơi nhiều thiết bị được kết nối trên cùng một đường truyền (ví dụ: I2C, One-Wire). Khi tắt cả các đầu ra của các thiết bị kết nối với đường truyền ở trạng thái Hi-Z (không kết nối), đường truyền được đẩy đến một mức logic mặc định 1 bằng cách sử dụng trở kéo lên. Bất kỳ thiết bị nào cũng có thể pull xuống mức logic 0 bằng cách sử dụng open-drain và tất cả các thiết bị có thể nhận thấy mức này.



Hình 1.12: Sơ đồ config của open-drain

## 4.2 GPIO Input Mode

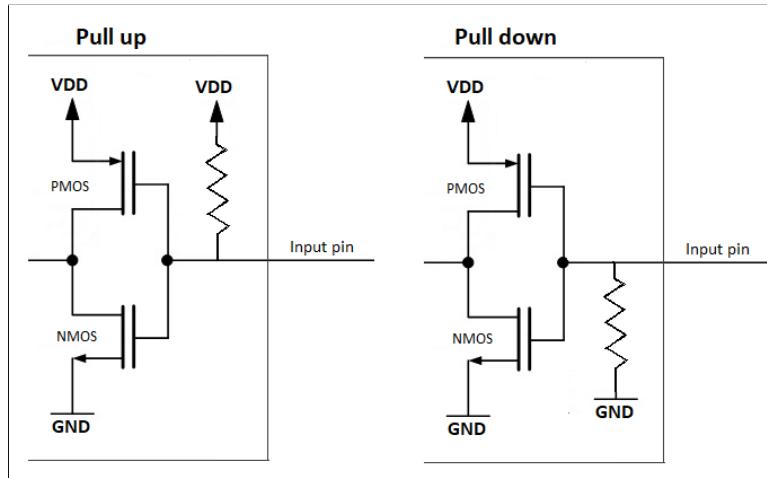
Để tránh hiện tượng thả nổi và đảm bảo rằng chân input có giá trị xác định khi không được kết nối với bất kỳ thiết bị nào, có hai chế độ để xác định giá trị ban đầu của một chân input: pull up và pull down.

- Pull up

- Điện trở kéo lên bên trong được kết nối với chân vi điều khiển. Vì vậy, trạng thái sẽ ở mức Cao trừ khi sử dụng điện trở kéo xuống bên ngoài.
- Việc sử dụng điện trở kéo lên có thể được config bởi người dùng.

- Pull down

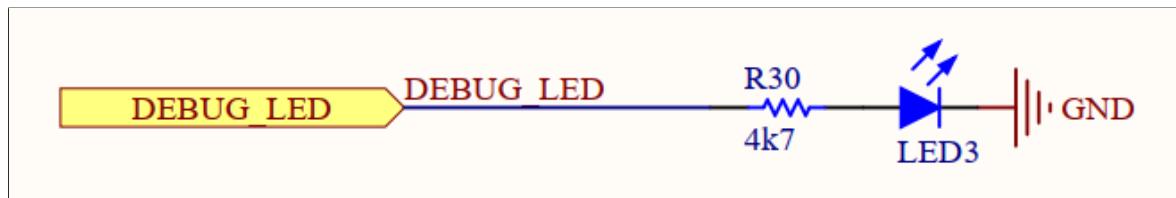
- Điện trở kéo xuống bên trong được kết nối với chân vi điều khiển. Vì vậy, trạng thái sẽ ở mức Thấp trừ khi sử dụng điện trở kéo lên bên ngoài.
- Việc sử dụng điện trở kéo xuống có thể được config bởi người dùng.



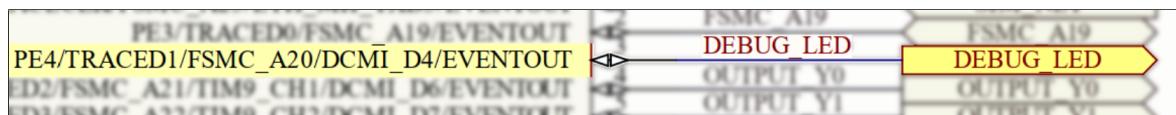
Hình 1.13: Sơ đồ config của input

## 5 Hướng dẫn config

Trong nội dung này, ta sẽ tiến hành kiểm thử trên kit thí nghiệm bằng cách điều khiển LED3 (LED DEBUG). Theo như schematic, LED3 sẽ được điều khiển bằng chân PE4 của MCU và sẽ tích cực mức cao (LED sẽ sáng khi đầu ra của vi điều khiển ở mức logic 1).



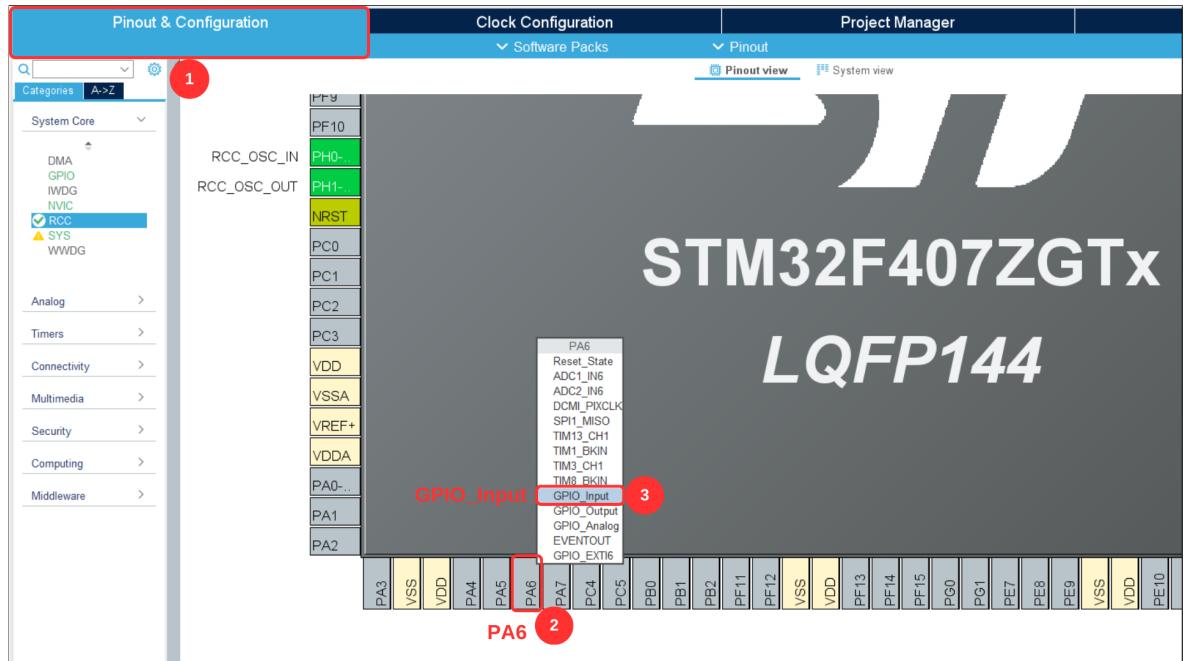
Hình 1.14: Sơ đồ nguyên lý của LED3



Hình 1.15: MCU điều khiển LED3 bằng chân PE4

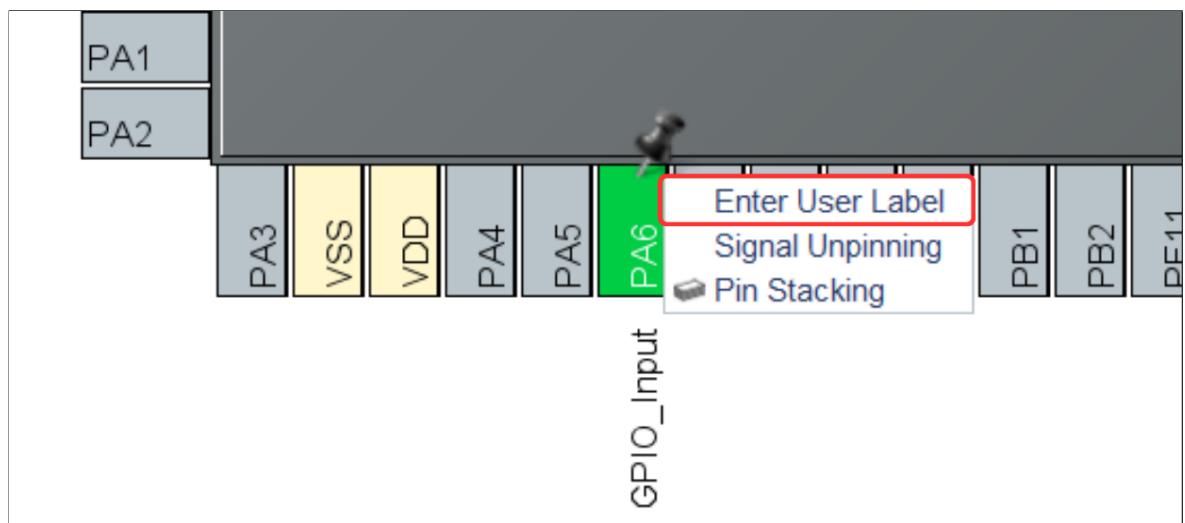
Ngoài ra, ta cũng có thể luyện tập config trước các chân ở khối input X0->X3 và khối output Y0,Y1. Lưu ý: Các khối input X và output Y trên kit thí nghiệm được thiết kế thông qua opto, dùng để tương tác với các thiết bị công suất lớn. Tương ứng với mỗi input và output nói trên, kit thí nghiệm sẽ có một đèn LED.

Để config các chân của vi điều khiển, ta mở file cấu hình (.ioc) và chọn vào cửa sổ **Pinout & Configuration**. Tiếp đó nhấp chọn chuột trái vào chân muốn config. Để config input, ta chọn **GPIO\_Input**.

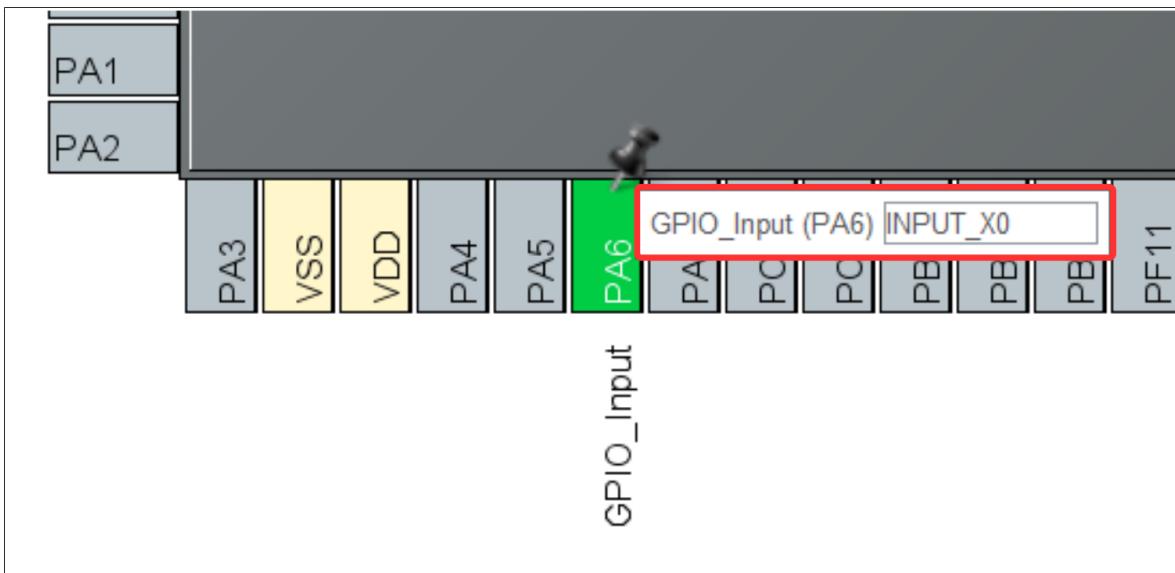


Hình 1.16: Config PA6 thành chân input

Tiếp theo, ta có thể đặt tên mới cho các chân này để thuận tiện cho việc lập trình. Đặt tên mới bằng cách nhấp chuột phải vào chân muốn đặt tên. Sau đó chọn Enter User Label và nhập tên mới.

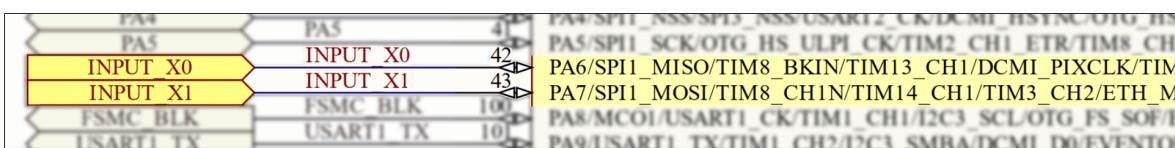


Hình 1.17: Nhấp chuột phải vào chân PA6 và chọn Enter User Label

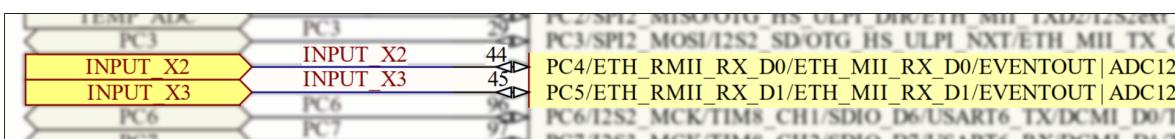


Hình 1.18: Nhập tên mới cho chân input PA6 là INPUT\_X0

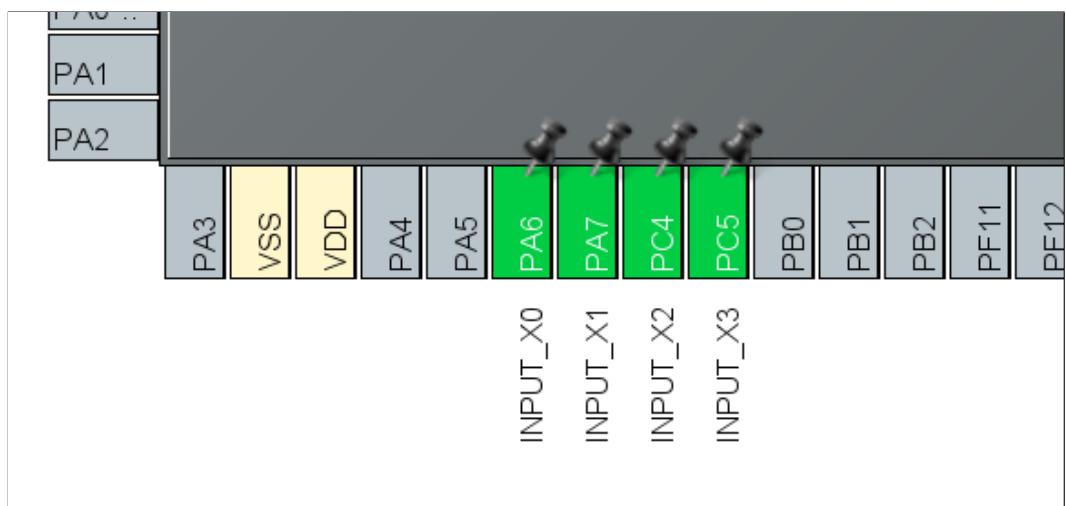
Thực hiện tương tự với các chân input còn lại.



Hình 1.19: Sơ đồ nguyên lý của các chân input X0 và X1

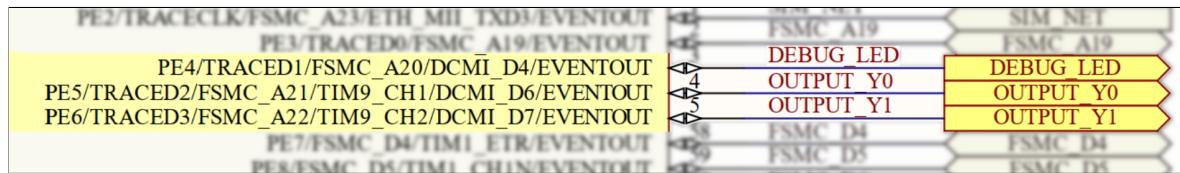


Hình 1.20: Sơ đồ nguyên lý của các chân input X2 và X3

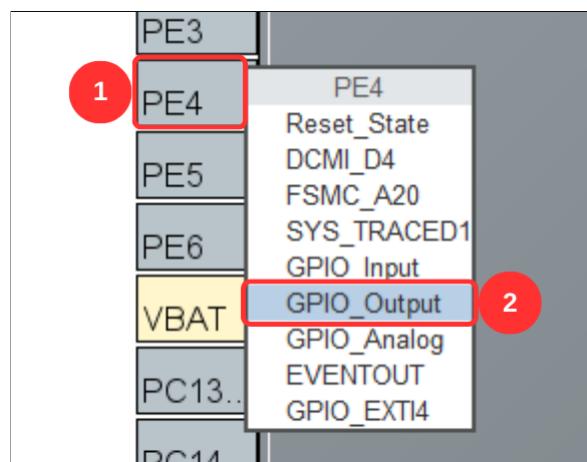


Hình 1.21: Các chân input sau khi được config và đặt lại tên

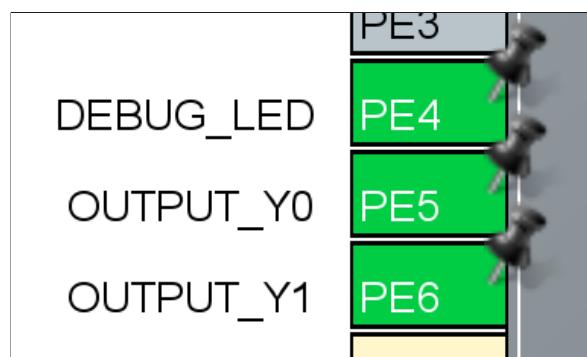
Tương tự như vậy, ta có thể config các chân output.



Hình 1.22: Sơ đồ nguyên lý của các chân output DEBUG\_LED, Y0 và Y1



Hình 1.23: Config PE4 thành chân output



Hình 1.24: Các chân output sau khi được config và đặt lại tên

Sau đó ta lưu file config lại để phần mềm sinh code. Sau khi sinh code, ta được file **main.c** như sau:

```
1 int main(void)
2 {
3     /* USER CODE BEGIN 1 */
4
5     /* USER CODE END 1 */
6
7     /* MCU configuration
8     -----
9     */
10
11    /* Reset of all peripherals, Initializes the Flash
12       interface and the Systick. */
13    HAL_Init();
14
15
16    /* USER CODE BEGIN Init */
17
18
19    /* USER CODE END Init */
20
21
22    /* configure the system clock */
23    SystemClock_Config();
24
25
26    /* USER CODE BEGIN SysInit */
27
28
29    /* Initialize all configured peripherals */
30    MX_GPIO_Init();
31
32    /* USER CODE BEGIN 2 */
33
34    /* USER CODE END 2 */
35
36
37    /* Infinite loop */
38    /* USER CODE BEGIN WHILE */
39
40
41    while (1)
42    {
43        HAL_GPIO_TogglePin(DEBUG_LED_GPIO_Port, DEBUG_LED_Pin);
44    }
45}
```

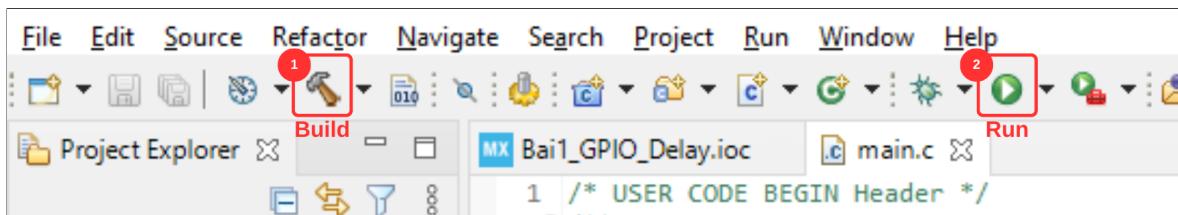
```

35     HAL_Delay(1000);
36     /* USER CODE END WHILE */
37
38     /* USER CODE BEGIN 3 */
39 }
40 /* USER CODE END 3 */
41 }
```

Program 1.1: Project chớp tắt led đầu tiên

Dòng 34 và 35 là các dòng ta chủ động thêm vào. Lưu ý khi ta lập trình cần phải để mã nguồn vào đúng các vị trí được cho phép, nếu không mã nguồn sẽ mất khi chỉnh sửa config và sinh lại code. Khi lập trình nên tập thói quen sử dụng phím tắt **Ctrl + Space** để gợi ý mã nguồn.

Đoạn mã trên là chương trình mẫu chớp tắt LED mỗi 1 giây. Bước tiếp theo là **Build** chương trình, bước này sẽ giúp kiểm tra các lỗi về mặt cú pháp. Sau đó, để nạp chương trình vào kit thí nghiệm, ta chọn **Run**. Lưu ý: SW1 trên kit thí nghiệm phải ở trạng thái ON để nạp code.



Hình 1.25: Build và Run chương trình

Bên cạnh sử dụng câu lệnh **Toggle**, ta còn có thể sử dụng câu lệnh **Set** và **Reset**.

```

1 while (1){
2     HAL_GPIO_WritePin(DEBUG_LED_GPIO_Port, DEBUG_LED_Pin, 1);
3     HAL_Delay(1000);
4     HAL_GPIO_WritePin(DEBUG_LED_GPIO_Port, DEBUG_LED_Pin, 0);
5     HAL_Delay(1000);
6 }
```

Program 1.2: Ví dụ chương trình chớp tắt LED

Sau khi nạp chương trình, ta được kết quả LED3 trên kit thí nghiệm sẽ chớp tắt mỗi 1 giây. Đoạn code trên có thể được chỉnh sửa để điều khiển output Y0, Y1, hoặc có thể tham khảo project mẫu **Bai1\_GPIO\_Delay**.

## **6 Bài tập và Báo cáo**

### **6.1 Bài tập 1**

Viết chương trình điều khiển LED3 sáng trong 2 giây, sau đó tắt trong 4 giây, quá trình này được lặp lại mãi mãi.

### **6.2 Bài tập 2**

Lập trình lại chương trình ở Bài tập 1 nhưng chỉ sử dụng một câu lệnh delay duy nhất ở cuối vòng lặp while. Gợi ý: sử dụng biến đếm và biến để lưu trạng thái của đèn.

### **6.3 Bài tập 3**

Sử dụng LED3 và các LED ở output Y0, Y1 để mô phỏng tín hiệu đèn giao thông. Giả sử mỗi LED đại diện cho một tín hiệu trên đèn giao thông với chu kì đèn đỏ là 5 giây, đèn xanh là 3 giây, đèn vàng là 1 giây. Chỉ sử dụng duy nhất 1 câu lệnh delay.