**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

\*\*\*\*\*\*\*\*\*\*

A close up of a sign

Description automatically generated

**BÁO CÁO THỰC HÀNH**

**HỌC PHẦN: HỆ NHÚNG**

**BÀI THỰC HÀNH SỐ 3**

**Xây dựng ứng dụng trên Hệ nhúng ARM Cortex-M4**

|  |  |  |
| --- | --- | --- |
| Họ và tên | : | Trương Văn Hiển |
| Mã số sinh viên | : | 20194276 |
| Lớp | : | 727602 |
| Giảng viên hướng dẫn | : | TS. Ngô Lam Trung |

*Hà Nội, tháng 07 năm 2023*

**MỤC LỤC**

[3.0. Làm quen với Kit STM32F429 3](#_Toc141017505)

[3.1. Làm quen với STM32CubeIDE 3](#_Toc141017506)

[3.2. Giao tiếp với ngoại vi 5](#_Toc141017507)

[3.3. Xây dựng ứng dụng với TouchGFX và FreeRTOS 7](#_Toc141017508)

[3.4. Bài tập tự làm 8](#_Toc141017509)

# **3.0. Làm quen với Kit STM32F429**

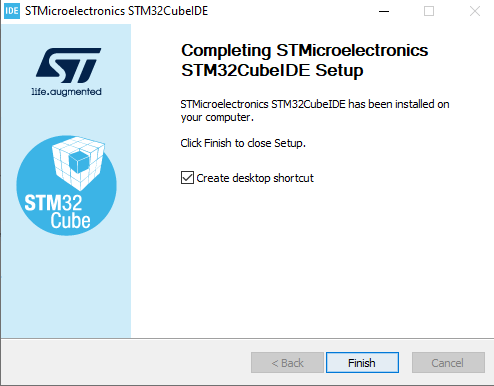
Kit STM32F429

* STM32F429ZIT6 ARM Cortex-M4.
* Tốc độ CPU 180 MHz (max).
* 2 MB flash, 256 KB SRAM, 8MB SDRAM.
* Màn hình LCD 2.4” 240x320, cảm ứng điện trở.
* 6 LEDs (2 user LEDs).
* 3 axis gyros.

A close up of a circuit board

Description automatically generated

# **3.1. Làm quen với STM32CubeIDE**



A screenshot of a computer

Description automatically generated

* Quan sát 2 đèn LED màu xanh và cam cùng bật sáng:

A rectangular object with a rectangular frame on a wooden surface

Description automatically generated

* Ý nghĩa của 2 dòng code được thêm vào hàm **main()**:

1. Bật đèn LED3:

*HAL\_GPIO\_WritePin(LD3\_GPIO\_Port, LD3\_Pin, GPIO\_PIN\_SET); //turn on LED3*

* **HAL\_GPIO\_WritePin**: hàm trong thư viện HAL của STM32 để điều khiển đầu ra GPIO
* **LD3\_GPIO\_Port**: hằng số cổng GPIO của LD3
* **LD3\_Pin**: hằng số chân GPIO của LD3
* **GPIO\_PIN\_SET**: giá trị thể hiện trạng thái cao (ON) của chân GPIO

Khi thực thi dòng code trên, nó sẽ đưa chân GPIO của LD3 lên trạng thái cao, từ đó bật đèn LED LD3.

1. Bật đèn LED4:

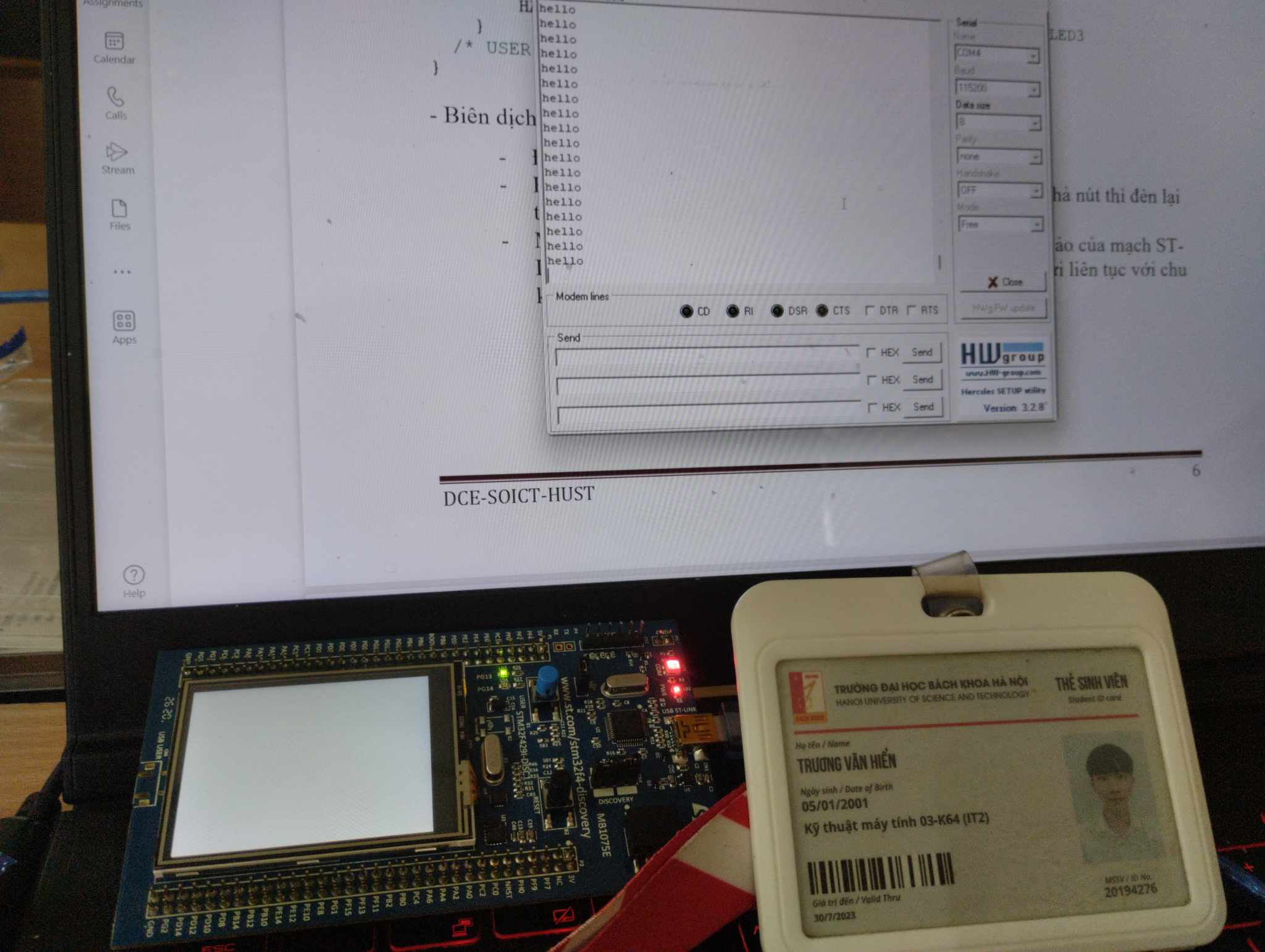
*HAL\_GPIO\_WritePin(LD4\_GPIO\_Port, LD4\_Pin, GPIO\_PIN\_SET); //turn on LED4*

* **HAL\_GPIO\_WritePin**: hàm trong thư viện HAL của STM32 để điều khiển đầu ra GPIO
* **LD4\_GPIO\_Port**: hằng số cổng GPIO của LD4
* **LD4\_Pin**: hằng số chân GPIO của LD4
* **GPIO\_PIN\_SET**: giá trị thể hiện trạng thái cao (ON) của chân GPIO

Khi thực thi dòng code trên, nó sẽ đưa chân GPIO của LD4 lên trạng thái cao, từ đó bật đèn LED LD4.

⇒ 2 dòng code trên được sử dụng để bật hai đèn LED LD3 và LD4 thông qua việc điều khiển các chân GPIO của vi điều khiển STM32.

# **3.2. Giao tiếp với ngoại vi**



Giải thích:

* Vai trò của các đoạn code được thêm vào:

Với Config ở phần trên, chúng ta đã đặt cấu hình Prescaler = 89 , Counter preiod = 49999 , cùng với HSE = 180MHz , ta sẽ tính được chu kì ngắt của Timer 6 là

*(Prescaler + 1 ) x (Counter + 1 ) / 180000000 = 25ms*

* Do ta đã thiết lập ngắt Timer ở Timer6, nên cứ đều đặn 25ms lại xuất hiện ngắt 1 lần, mỗi lần ngắt sẽ tực hiện hàm **TIM6\_DAC\_IRQHandler()** 1 lần để tăng biến *TimerCount* lên 1 đơn vị , khi biến TimerCount đạt giá trị 20 tiến hành đảo trạng thái của LD3\_Pin, có nghĩa là cứ *20x25ms = 500ms* thì đèn sẽ đảo trạng thái 1 lần.
* Điều này có nghĩa là cứ 1000ms thì đèn mới bật lại 1 lần nữa, đó chính là lí do tại sao đèn LED3 lại nháy với tần số 1Hz.
* Khi bấm nút màu xanh , đèn LED4 bật lên bởi vòng lặp *while* trong hàm **main**, biến *GPIO\_PinState* dược dùng để đọc giá trị của cổng PA0(USER\_BUTTON), mỗi khi PA0 ở trạng thái HIGH(nút được bấm) thì sẽ đồng thời thiết lập trạng thái HIGH cho LED4, còn nếu nút không được bấm thì sẽ thiết lập trạng thái LOW cho LED4, vì vật mà LED4 sẽ bật sáng mỗi khi USR\_BUTTON được kích hoạt.
* Đoạn code được thêm vào vòng lặp while để bật đèn LED4 dựa vào USER\_BUTTON được thiết lập ở PA0, đồng thời truyền buffer là một xâu có chứa kí tự “hello” thông qua giao tiếp UART.
* Đoạn code chứa hàm **TIM6\_DAC\_IRQHandler** chính là hàm xử lý ngắt , sẽ tự thực hiện ngắt mỗi khi có ngắt do Timer6 xảy ra.
* Vai trò của đoạn code dưới đây trong hàm **MX\_GPIO\_Init()**:

**A close-up of a computer screen

Description automatically generated**

Đoạn code trên được dùng để thiết lập các thuộc tính và enable cho cổng của một nút bấm có tên là USER\_BUTTON.

# **3.3. Xây dựng ứng dụng với TouchGFX và FreeRTOS**

**A group of id cards and electronics on a table

Description automatically generated**

* Đoạn video chạy ứng dụng em đã có gửi kèm theo Assignment
* Giải thích hoạt động của chương trình:

Ta khai báo một hàng đợi Queue1Handler để thủ tục hóa các tín hiệu đầu vào từ các chân GPIO , với hàm **StartDefaultTask()**, ta sẽ định nghĩa các message từ các chân GPIO để Put vào hàng đợi chờ xử lý . Sau đó hàm **handleTickEvent()** trong Screen1View sẽ thực hiện lấy các message từ hàng đợi ra và thực thi chương trình như mong muốn

1. Hình tròn rơi từ trên xuống :

Trong hàm **handleTickEvent()** , biến *tickCount* sẽ được tăng dần sau mỗi vòng lặp , đồng thời trong mỗi vòng lặp ta thấy *image1.setY(tickCount%320)* có nghĩa rằng tọa độ trục Y trên màn hình của image 1 sẽ tăng dần cho đến khi *tickCount* vượt quá 320 sẽ quay lại từ 0, đấy chính là lí do tại sao hình tròn rơi từ trên xuống (320 là chiều dài cơ sở của màn hình)

1. Khi bấm nút, hình tròn đồng thời di chuyển sang bên phải cùng với tự động đi xuống

Khi bấm nút, *BUTTON\_USER* sẽ gửi một message tới *Queue1Handler*, và khi message được lấy ra so sánh trong **handleTickEvent()**, tọa độ trục X trên màn hình của image 1 sẽ tăng dần lên 2 đơn vị đồng thời tọa độ trục Y sẽ vẫn tăng dần như phần trên. Chính vì vậy mà hình tròn sẽ di chuyển về phía dưới sang bên phải mỗi khi bấm nút.

# **3.4. Bài tập tự làm**

Sử dụng mã nguồn mẫu đã cho “MyApplication-D01”, xây dựng giao diện và phần hoạt hình cơ bản cho 1 game đua xe.

* Đường đua di chuyển theo chiều từ trên xuống. - Xe chỉ có thể di chuyển theo chiều ngang.
* Ghép nối thêm 2 nút bấm (sử dụng chân PG2 và PG3), để điều khiển xe dịch chuyển sang trái/phải. Chú ý: cần tham khảo hàm MX\_GPIO\_Init() trong bài 3.2 để tự viết code cấu hình PG2 và PG3 ở chế độ.

Các file asset mẫu được để trong cùng thư mục.