Bài thực hành 1

GHÉP NỐI VỚI GPIO, INTERRUPT, TIMER

1. Mục đích

- Củng cố kiến thức về các ngoại vi cơ bản của vi điều khiển: GPIO, ngắt ngoài, timer.

- Thực hành ghép nối với LED đơn và LED 7 thanh.

- Tìm hiểu chuẩn NEC Protocol ứng dụng trong điều khiển từ xa hồng ngoại. Lập trình kết hợp ngắt ngoài và timer để nhận và giải mã lệnh từ điều khiển từ xa.

- Xây dựng ứng dụng đơn giản với LED 7 thanh, remote control.

2. Chuẩn bị

- Bộ KIT STM32F429-DISC.

- Phụ kiện: điều khiển từ xa hồng ngoại, module thu hồng ngoại HS0038, module LED 7 thanh, LED đơn, điện trở, transistor, breadboard và dây nối.

- Tài liệu hướng dẫn thực hành, mã nguồn mẫu, datasheet của HS0038.

- Phần mềm: STM32CubeIDE, CubeMX, Hercules.

3. Nội dung thực hành

3.0. Project mẫu

- Mở STM32CubeIDE, vào menu File → Import, rồi thực hiện import project trong file **Lab1_20242.zip**.

- Dịch và chạy chương trình trên kit STM32F429. Kiểm tra hoạt động của mạch. Nếu thấy đèn LED3 và LED4 đảo trạng thái khi bấm đúp vào nút B1 tức là project đã biên dịch đúng và mạch hoạt động đúng.

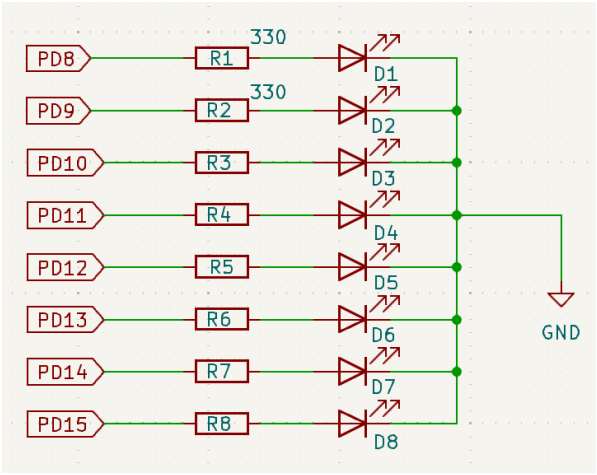
- Kiểm tra lại các thiết lập tại project này (trong file ioc).

  - Chân PA0 được cấu hình ở chế độ External interrupt + Rising edge.
  - CPU được cấu hình chạy ở tốc độ 180 MHz, các timer chạy ở tốc độ 90 MHz.
  - Timer 6 được cấu hình để sinh ngắt liên tục với tần số 10000 Hz.
  - Sự kiện bấm đúp trên nút xanh được phát hiện và xử lý trong hàm xử lý ngắt ở file **stm32f4xx_it.c** Sinh viên cần đọc hiểu kỹ project mẫu này.

3.1. Ghép nối LED đơn

- Lắp mạch để ghép nối dãy 8 LED đơn với các chân từ PD8 đến PD15 của STM32F429 theo sơ đồ trên Hình 1.

- Mở file ioc, thiết lập cấu hình cho các chân PD8 - PD15 là GPIO_Output. Lưu project và sinh mã nguồn tự động (Ctrl + S).



Hình 1: Ghép nối LED đơn

- Khai báo thêm biến LED_Value

```
/* USER CODE BEGIN PV */
int TIM6_Count;
unsigned char LED_Value;
/* USER CODE END PV */
```

- Thêm vào file main.c hàm **void DisplayLEDs(int** mode) dưới đây tại vị trí thích hợp trong vùng *Private user code*.

```
/* Private user code ---------------------------------------------------*/
/* USER CODE BEGIN 0 */
void DisplayLEDs(int mode)
{
	if (mode == 1)
	{
		LED_Value = (LED_Value >> 1) | (LED_Value << 7);
	}

	if (LED_Value & 0x80)
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET);
	else
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET);

	if (LED_Value & 0x40)
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
	else
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);

	if (LED_Value & 0x20)
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
	else
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);

	if (LED_Value & 0x10)
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
	else
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);

	if (LED_Value & 0x8)
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_11, GPIO_PIN_SET);
	else
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_11, GPIO_PIN_RESET);

	if (LED_Value & 0x4)
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_10, GPIO_PIN_SET);
	else
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_10, GPIO_PIN_RESET);

	if (LED_Value & 0x2)
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_9, GPIO_PIN_SET);
	else
```

```
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_9, GPIO_PIN_RESET);

        if (LED_Value & 0x1)
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_8, GPIO_PIN_SET);
        else
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_8, GPIO_PIN_RESET);
}
```

- Sử dụng hàm **DisplayLEDs** trong main loop để tạo hiệu ứng LED chạy

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
LED_Value = 1;
while (1)
{
    HAL_Delay(500);
    DisplayLEDs(1);
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

- Biên dịch và nạp chương trình lên kit. Hiệu chỉnh lỗi nếu có để mạch chạy như trong video **Bai 1 - Nhay LED don.mp4**.

Phần tự làm:

Viết thêm code cho hàm DisplayLEDs để tạo thêm các hiệu ứng tương ứng với giá trị tham số mode.

1): Running spot L: Bật lần lượt từng led từ phải qua trái

∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ → ∘ ∘ ∘ ∘ ∘ ∘ ∘ O → ∘ ∘ ∘ ∘ ∘ ∘ O ∘ → ∘ ∘ ∘ ∘ ∘ O ∘ ∘ →

∘ ∘ ∘ ∘ O ∘ ∘ ∘ → …... → O ∘ ∘ ∘ ∘ ∘ ∘ ∘ → ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘

2): Running spot R: Bật lần lượt từng led từ trái qua phải

∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ → O ∘ ∘ ∘ ∘ ∘ ∘ ∘ → ∘ O ∘ ∘ ∘ ∘ ∘ ∘ → ∘ ∘ O ∘ ∘ ∘ ∘ ∘ →

∘ ∘ ∘ O ∘ ∘ ∘ ∘ → …... → ∘ ∘ ∘ ∘ ∘ ∘ ∘ O → ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘

3) Flash: Bật tất cả 8 led, trễ thích hợp sau đó, tắt tất cả 8 led.

∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ → O O O O O O O O → ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘

4): Spot bumber: Bật lần lượt 2 led đối xứng từ ngoài vào trong rồi từ trong ra ngoài

∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ → O ∘ ∘ ∘ ∘ ∘ ∘ O → ∘ O ∘ ∘ ∘ ∘ O ∘ → ∘ ∘ O ∘ ∘ O ∘ ∘ →

∘ ∘ ∘ O O ∘ ∘ ∘ → ∘ ∘ ∘ O O ∘ ∘ ∘ → ∘ ∘ O ∘ ∘ O ∘ ∘ → ∘ O ∘ ∘ ∘ ∘ O ∘ →

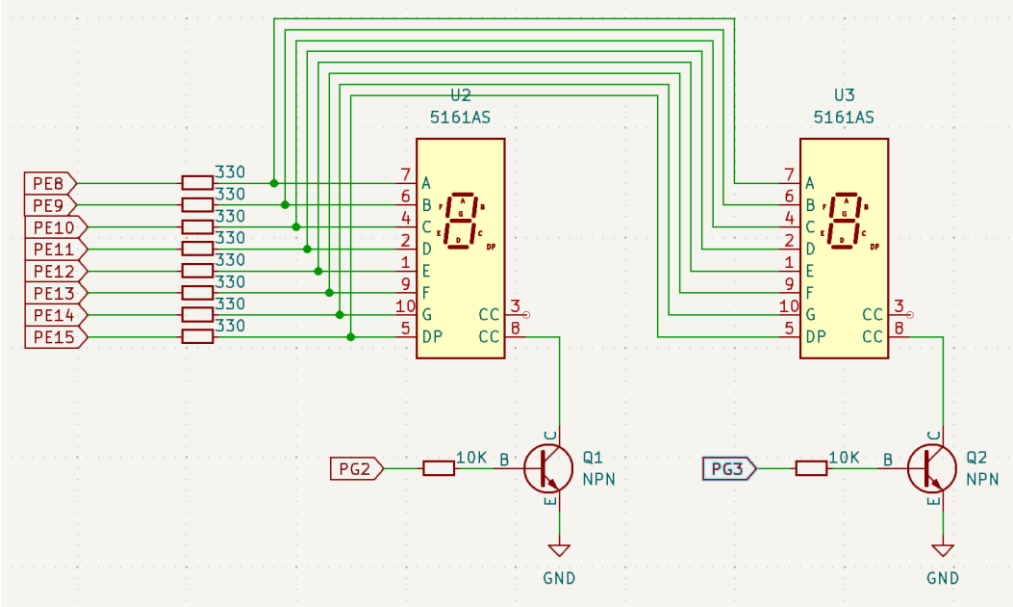
O ∘ ∘ ∘ ∘ ∘ ∘ O → ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘

Các hiệu ứng này được lặp lại 2 chu kỳ trước khi lần lượt chuyển sang hiệu ứng kế tiếp theo vòng tròn. Sinh viên tùy chọn thời gian trễ khi chuyển trạng thái đèn.

3.2. Ghép nối LED 7 thanh

- Lắp mạch để ghép nối 2 module LED 7 thanh với STM32F4 như trên Hình 2.

- Các chân điều khiển thanh LED nối với PE8-PE15.
- Các chân điều khiển chọn module LED nối với PG2 và PG3.



Hình 2: Ghép nối 2 module LED 7 thanh

- Thêm các file 7seg.h và 7seg.c vào project.

- Thêm code vào main loop để hiển thị số 25 trên 2 module LED 7 thanh.

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim6);
Set7SegDisplayValue(25);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
LED_Value = 1;
while (1)
{
        HAL_Delay(5);
        DisplayLEDs(1);
        Run7SegDisplay();
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
```

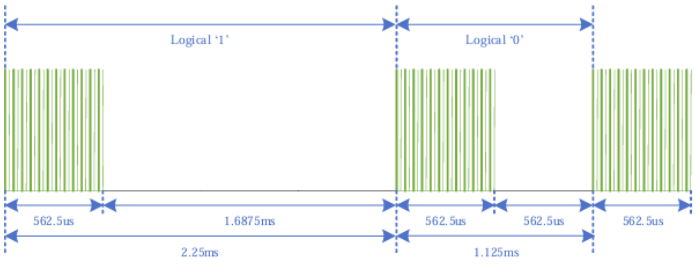
- Biên dịch và nạp lên kit. Hiệu chỉnh lỗi nếu có để mạch hiển thị số 25 lên 2 module LED 7 thanh, trong khi dãy đèn LED đơn nháy với tần số cao.

Phần tự làm:

- Lập trình để mỗi lần bấm nút B1 thì giá trị hiển thị trên 2 module LED 7 thanh tăng một đơn vị.

3.3. Giải mã tín hiệu điều khiển hồng ngoại

a. Mã hóa tín hiệu điều khiển từ xa hồng ngoại theo chuẩn NEC.



Giao thức truyền hồng ngoại của NEC sử dụng phương pháp "pulse distance encoding" để mã hóa bit 0 và 1. Bộ điều khiển hồng ngoại gửi liên tiếp các chuỗi xung hồng ngoại được điều chế ở tần số 38 KHz gọi là pulse burst. Mỗi pulse burst có độ rộng 562.5 micro giây (vùng màu xanh) và tương ứng với 1 bit. Khoảng cách giữa pulse burst hiện tại với pulse burst kế tiếp sẽ cho biết giá trị của bit là 1 (dài) hay 0 (ngắn).

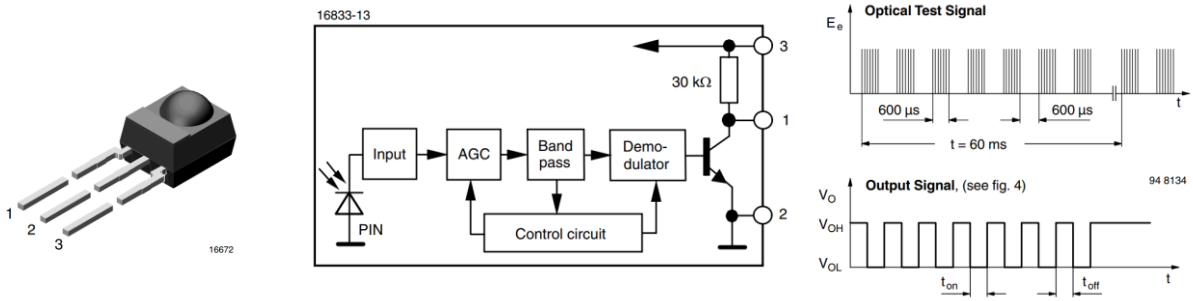
Một lệnh điều khiển hồng ngoại theo chuẩn NEC là một chuỗi nhị phân 32 bit, được đóng gói thành một bản tin tương ứng với 34 pulse burst.

- Start: burst rộng 9ms, theo sau là khoảng lặng 4.5 ms.

- 32 bit dữ liệu, mỗi bit là một burst rộng 562.5 micro giây theo sau là khoảng lặng 562.5 micro giây (bit 0) hoặc 1687.5 micro giây (bit 1). Xem thêm trong phụ lục.

- Stop: burst rộng 562.5 micro giây, đánh dấu đã hết 32 bit.

b. Giải điều chế và giải mã tín hiệu hồng ngoại.



Hình 3: Nguyên tắc hoạt động của bộ thu HS0038

Hình 3 mô tả nguyên tắc hoạt động của bộ thu hồng ngoại HS0038 và các chip tương đương. Ở trạng thái bình thường khi bộ thu không phát hiện pulse bust thì giá trị logic tại đầu ra ở chân 1 được thiết lập bằng 1. Khi bộ thu phát hiện có pulse burst (phát từ điều khiển hồng ngoại) thì mức logic tại đầu ra bằng 0. Như vậy khi có một lệnh điều khiển được gửi tới bộ thu, thì tại đầu ra của bộ thu sẽ có một chuỗi xung tương ứng. Chuỗi xung này cần được giải mã để tìm ra giá trị mã lệnh ban đầu được gửi bởi điều khiển.

Nguyên tắc thực hiện giải mã lệnh hồng ngoại là do thời gian của các bit sau khi giải điều chế. Nếu thời gian của bit tương ứng 13.5 ms thì đây là bit start bắt đầu một lệnh. Nếu thời gian là 1.25 ms thì đây là bit 0. Và nếu thời gian là 2.25 ms thì đây là bit 1. Một lệnh sẽ được giải mã khi nhận đủ 33 bit, gồm 1 bit start và 32 bit dữ liệu.

c. Ghép nối và lập trình giải mã bản tin hồng ngoại.

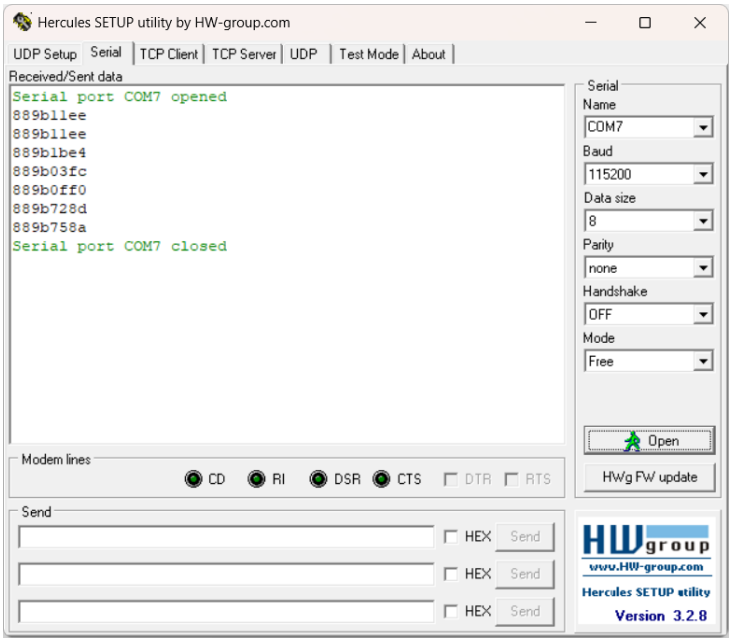
- Lắp mạch kết nối module bộ thu hồng ngoại HS0038 với STM32F429 theo sơ đồ.

HS0038	STM32F429
1 (Out)	PG5
2 (-)	GND
3 (+)	3V

- Import project IrRemoteControl trong file zip tương ứng vào STM32CubeIDE.

- Dịch và chạy trên kit.

- Mở phần mềm Hercules, mở cổng Serial ở tab Serial để quan sát dữ liệu gửi từ kit về PC qua UART. Chú ý: xem số hiệu cổng Serial trong Device Manager của PC. Mỗi khi có một nút trên điều khiển được nhấn, mã lệnh tương ứng của nút sẽ hiện trên Hercules.



Phần tự làm:

- Đọc mã nguồn project và mô tả chi tiết phương pháp giải mã lệnh điều khiển hồng ngoại.

- Chân Out của HS0038 được nối vào chân PG5. Hãy vẽ biểu đồ giá trị tín hiệu trên chân PG5 theo thời gian khi có 1 nút được bấm trên điều khiển (HS0038 nhận được lệnh từ điều khiển).
- Ngắt ngoài EXTI5 trên chân PG5 được đặt là Rising edge hay Falling edge? Tại sao?
- Làm thế nào để đo thời gian của các bit?
- Căn cứ vào đâu để xác định các bit start, 0, 1?
- Khi nào thì một lệnh từ điều khiển được giải mã xong? Mã lệnh sau khi giải mã được để ở đâu?

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

- Xây dựng chương trình thực hiện chức năng:

+ Giải mã lệnh gửi từ điều khiển từ xa tới mạch.

+ Hiển thị giá trị nút bấm (byte số 3 trong dãy 4 byte mã lệnh) lên LED 7 thanh.

+ Nếu mã lệnh tương ứng với nút OK thì bật/tắt đèn LED4.

+ Nếu mã lệnh tương ứng với một trong các nút 1, 2, 3, 4 thì hiển thị hiệu ứng tương ứng trên dãy LED đơn. Các hiệu ứng này đã thực hiện ở bài 3.1.

5. Viết báo cáo thực hành

- Giải thích các nội dung làm trong phần 3 và 4 (kèm hình ảnh minh họa, mã nguồn).

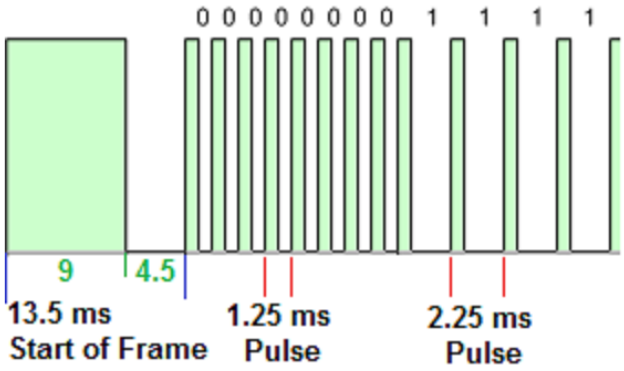
Phụ lục: khuôn dạng và cách mã hóa lệnh của điều khiển từ xa theo NEC Protocol

NEC Protocol

NEC IR protocol encodes the keys using a 32-bit frame format as shown below.

NEC Frame Format			
Address	Complement of Address	Command	Complement of Command
LSB-MSB(0-7)	LSB-MSB(8-15)	LSB-MSB(16-23)	LSB-MSB(24-31)

Each bit is transmitted using the pulse distance as shown in the image.
Logical '0': A 562.5µs pulse burst followed by a 562.5µs space, with a total transmit time of 1.125ms
Logical '1': A 562.5µs pulse burst followed by a 1.6875ms space, with a total transmit time of 2.25ms



When a key is pressed on the remote controller, the message transmitted consists of the following, in order:

1. A 9ms leading pulse burst (16 times the pulse burst length used for a logical data bit)
2. A 4.5ms space
3. The 8-bit address for the receiving device
4. The 8-bit logical inverse of the address
5. The 8-bit command
6. The 8-bit logical inverse of the command
7. A final 562.5µs pulse burst to signify the end of message transmission.

The four bytes of data bits are each sent least significant bit first. Below image illustrates the format of an NEC IR transmission frame, for an address of 00h (00000000b) and a command of ADh (10101101b).

A total of 67.5ms is required to transmit a message frame. It needs 27ms to transmit the 16 bits of address (address + inverse) and the 16 bits of command (command + inverse).