ĐẠI HỌC QUỐC GIA

ĐẠI HỌC BÁCH KHOA TP HỒ CHÍ MINH

– Embedded System Lab 02 –

ESP32 GPIO and FreeRTOS task

Nhóm 2 – L02

Họ và tên	MSSV
Trần Nguyễn Minh Duy	1910095
Đặng Trung Kiên	1911437
Nguyễn Hải Long	1911517
Nguyễn Nhật Trường	1912344

Thành phố Hồ Chí Minh – 2022

ĐÓNG GÓP CỦA CÁC THÀNH VIÊN

Họ và tên	MSSV	Đóng góp
Trần Nguyễn Minh Duy	1910095	25%
Đặng Trung Kiên	1911437	25%
Nguyễn Hải Long	1911517	25%
Nguyễn Nhật Trường	1912344	25%

BÁO CÁO

1. Chuẩn bị linh kiện

Để hiện thực bài Lab 2, ta cần chuẩn bị các linh kiện sau:

- 1 mạch ESP32
- 1 dây cáp USB để nạp code cho mạch ESP32
- 1 nút nhấn (button)
- Dây điện nối nút nhấn với mạch ESP32

2. Quy trình thực hiện

2.1. Lắp đặt phần cứng

- Đầu tiên, ta cắm cáp USB. Một đầu ta cắm vào mạch ESP32, đầu USB ta cắm vào máy tính (laptop).
- Thứ hai, ta dùng dây dẫn và nối nút nhấn vào cổng GPIO 2 của mạch ESP32

2.2. Cài đặt môi trường ESP-IDF

- Để hiện thực lab 2, ta cần tải và cài đặt ESP-IDF để lập trình.

2.3. Lập trình cho mạch ESP32

- Ta lập trình để hiện thực lab 2 sao cho thỏa yêu cầu đề bài: tạo 2 task và lên lịch chúng bằng các chức năng quản lý tác vụ của FreeRTOS
    - Cyclic task in mã nhận dạng học sinh mỗi giây.
    - Acyclic task đọc nút nhấn và in “ESP32” mỗi khi nhấn nút.

2.4. Nạp code và kiểm tra chương trình

- Sau khi lập trình, ta build code và nạp code vào mạch ESP32.
- Tiếp theo, ta kiểm tra xem mạch có hoạt động đúng với mô tả của đề bài.

3. Hiện thực chương trình

3.1. Khai báo biến và hằng số

```
define BUTTON_GPIO 2


#define IDENTIFIER "Group 2:\n1. Tran Nguyen Minh Duy - 1910095\n2. Dang Trung Kien - 1911437\n3. Nguyen Hai Long - 1911517\n4. Nguyen Nhat Truong - 1912344\n"


short key_code = 0;
```

Giải thích code:

- #define BUTTON_GPIO 2: hằng số BUTTON_GPIO có giá trị là 2, đại diện cho Pin GPIO 2 cách mạch ESP32.
- #define IDENTIFIER "Group 2:\n1. Tran Nguyen Minh Duy - 1910095\n2. Dang Trung Kien - 1911437\n3. Nguyen Hai Long - 1911517\n4. Nguyen Nhat Truong - 1912344\n": Student identifier in ra mỗi giây
- short key_code = 0;: biến này được sự dụng để lưu giá trị nút nhấn khi nút được nhấn hoặc không nhấn

3.2. Hàm khởi tạo hệ thống (init system)

```
void init_system()
{
  key_code = 0;


  // button
  gpio_pad_select_gpio(BUTTON_GPIO);
  gpio_set_direction(BUTTON_GPIO, GPIO_MODE_INPUT);
}
```

Giải thích code:

- Đầu tiên, ta khởi tạo key_code = 0 (không nhấn nút)

- Thứ hai, ta dùng hàm gpio_pad_select_gpio, và gpio_set_direction để thiết lập chân Pin GPIO của mạch dùng cho nút nhấn. Đồng thời đặt cho Pin này là input.

3.3. Hàm in student identifier mỗi giây

```
void monitor_task(void *pvParameter)
{
        while(1)
        {
           printf(IDENTIFIER);
           vTaskDelay(1000 / portTICK_RATE_MS);
        }

   vTaskDelete(NULL);
}
```

Giải thích code:

- Đầu tiên, ta tạo một vòng lặp vô hạn while(true).
- Trong vòng while, ta dùng lệnh printf() để in IDENTIFIER (đã định nghĩa từ trước). Đồng thời dùng lệnh vTaskDelay(1000 / portTICK_RATE_MS); để delay mỗi 1 giây.
- Cuối hàm, ta dùng vTaskDelete(NULL); để xóa task.

3.4. Hàm đọc nút nhấn

```
void buttton_task(void *pvParameter)
{

   while(1) {
      if (gpio_get_level(BUTTON_GPIO) == 1)
      {
         key_code++;
```

```
        }
        else
        {
            key_code = 0;
        }

        vTaskDelay(10 / portTICK_RATE_MS);
    }

    vTaskDelete(NULL);
}
```

Giải thích code:

- Đầu tiên, ta tạo một vòng lặp vô hạn while(true).
- Trong vòng while, ta đọc chân GPIO 2 (nối với nút nhấn). Nếu nút được nhấn, ta cộng 1 cho key_code. Ngược lại, key_code = 0.
- Ta dùng vTaskDelay(10 / portTICK_RATE_MS); để delay (10ms) – đọc nút nhấn mỗi 10ms.
- Cuối hàm, ta dùng vTaskDelete(NULL); để xóa task.

3.5. Hàm in "ESP32" khi nút nhấn được nhấn

```
void is_button_pressed(void *pvParameter)
{
    while(1) {
        if (key_code == 5)
        {
            printf("ESP32\n");
        }

        vTaskDelay(10 / portTICK_RATE_MS);
    }
```

```
    vTaskDelete(NULL);
}
```

- Đầu tiên, ta tạo một vòng lặp vô hạn while(true).
- Trong vòng while, nếu nút nhấn được nhấn (không run - liên tục trong 10*5 = 50 ms), thì ta in “ESP32”.
- Ta dùng vTaskDelay(10 / portTICK_RATE_MS); để delay (10ms).
- Cuối hàm, ta dùng vTaskDelete(NULL); để xóa task.

3.6. Hàm main

```
void app_main()
{
    init_system();

    xTaskCreate(&monitor_task, "monitor_task", 1024, NULL, 1, NULL);
    xTaskCreate(&buttton_task, "buttton_task", 1024, NULL, 2, NULL);
    xTaskCreate(&is_button_pressed, "is_button_pressed", 1024, NULL, 2, NULL);
    vTaskStartScheduler();
}
```

Giải thích code:

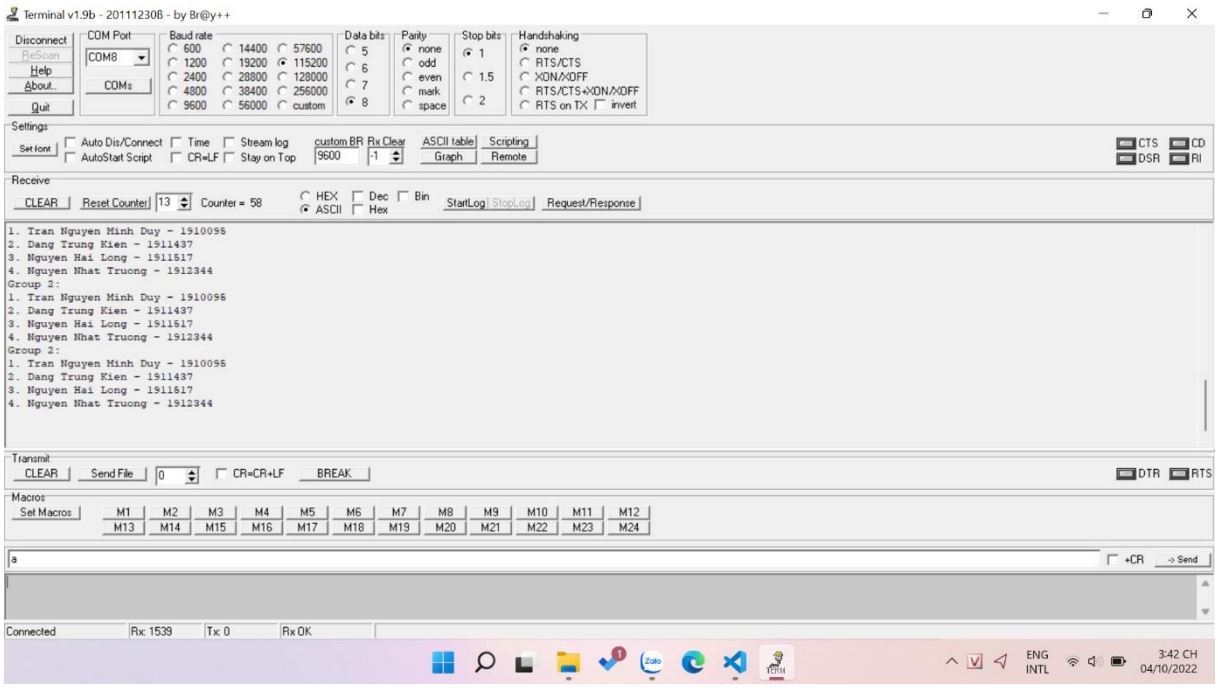
- Đầu tiên, ta gọi hàm init_system để khởi tạo hệ thống.
- Tiếp theo, ta dùng hàm xTaskCreate, để thêm task vào scheduler:
    - monitor_task (priority = 1). Priority = 1 (thấp nhất), vì tần số gọi hàm này ít nhất (mỗi 1s).
    - buttton_task (priority = 2), và is_button_pressed (priority = 2). Priority = 2 (cao hơn monitor_task), vì tần số gọi 2 hàm này bằng nhau (mỗi 10ms).

4. Link github

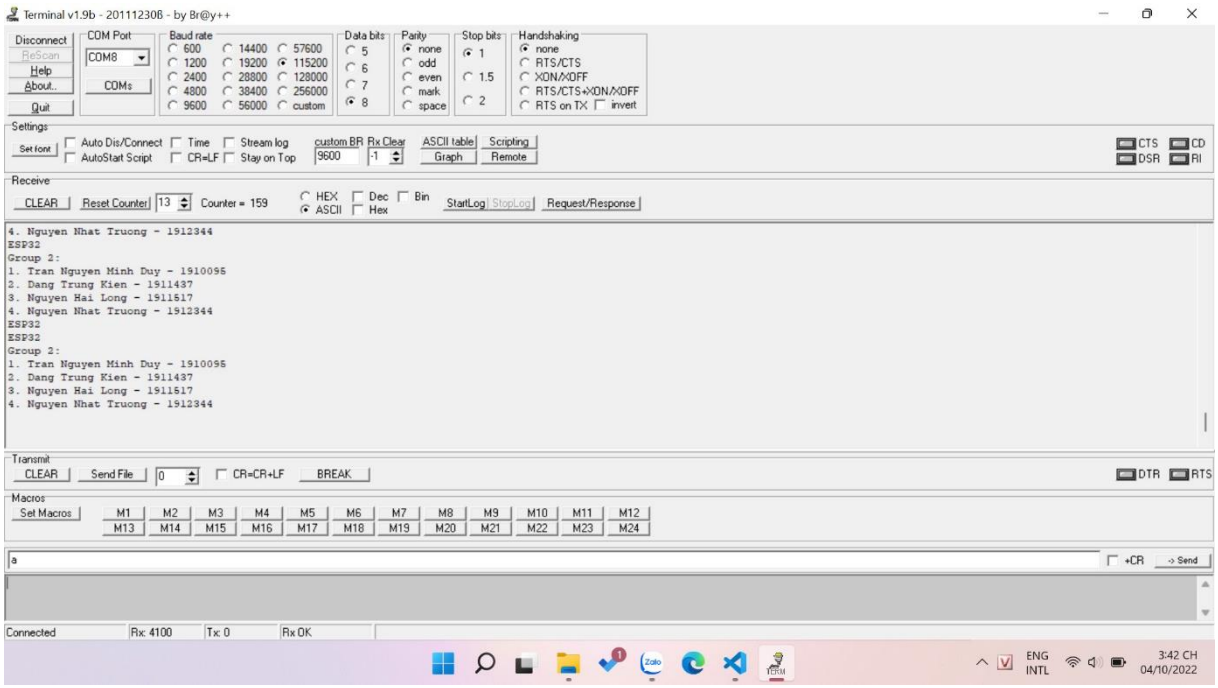
https://github.com/nhokkoranh1511/Embedded_System_LAB/tree/main/lab2/Code

5. Kết quả LAB 2

- Khi không nhấn nút: cứ mỗi giây mạch sẽ in ra student identifier.



- Khi nút được nhấn, mạch sẽ in ra “ESP32”.



6. Câu hỏi thêm

Does the ESP-IDF need the vTaskStartScheduler() routine?

ESP-IDF có cần quy trình vTaskStartScheduler () không?

- Câu trả lời: Ta không cần gọi vTaskStartScheduler nếu đang sử dụng ESP-IDF. Bởi vì, nó đã được gọi trước khi main() bắt đầu.