**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

1. **Quy trình thực hiện**
   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
  1. *Cài đặt môi trường ESP-IDF*
* Để hiện thực lab 2, ta cần tải và cài đặt ESP-IDF để lập trình.
  1. *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.
  1. *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.

1. **Hiện thực chương trình**
   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
  1. *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.
  1. *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.
  1. *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.
  1. *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.
  1. *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:

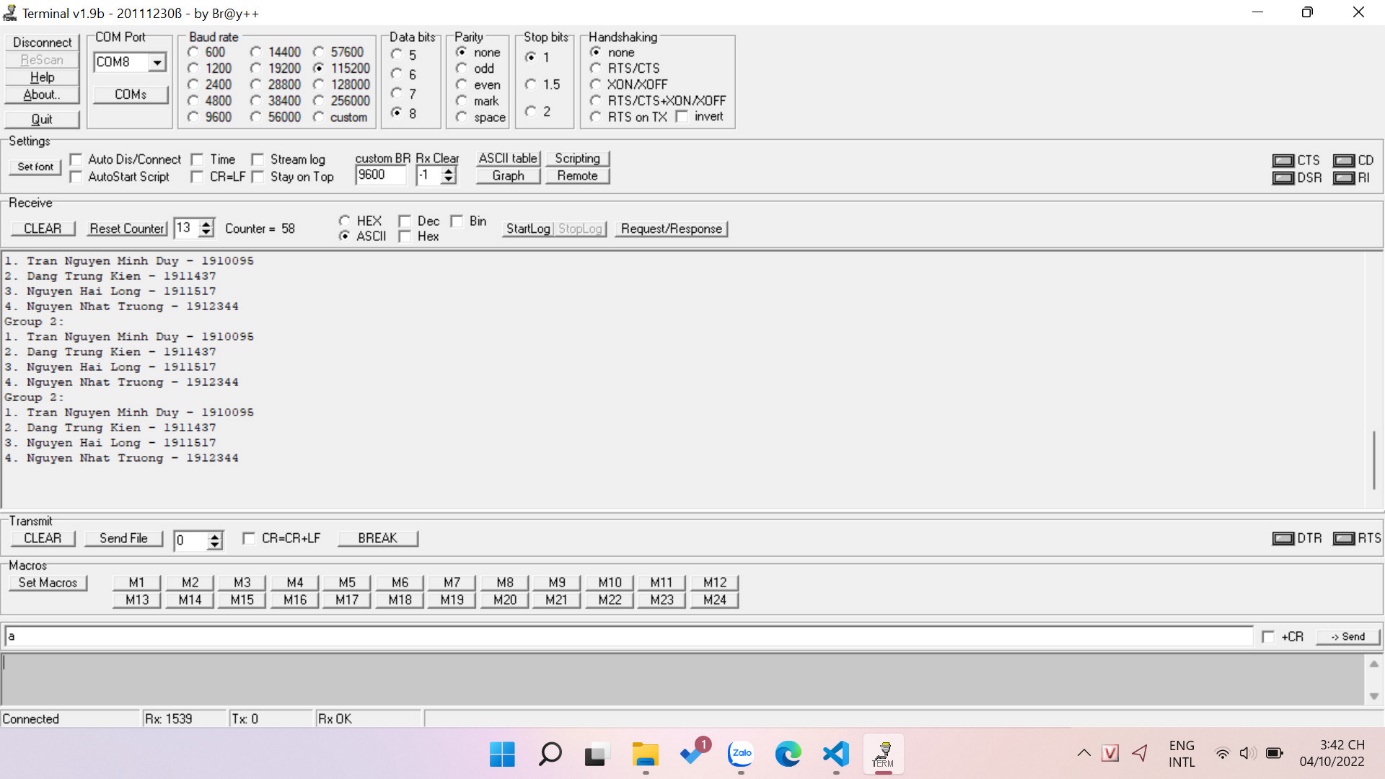
* Đầu tiên, ta gọi hàm init\_system để khởi tạo hệ thống.
* Tiếp theo, ta dùng hàm xTaskCreate, để thêm task vào scheduler:
  + monitor\_task (priority = 1). Priority = 1 (thấp nhất), vì tần số gọi hàm này ít nhất (mỗi 1s).
  + buttton\_task (priority = 2), và is\_button\_pressed (priority = 2). Priority = 2 (cao hơn monitor\_task), vì tần số gọi 2 hàm này bằng nhau (mỗi 10ms).

1. **Link github**

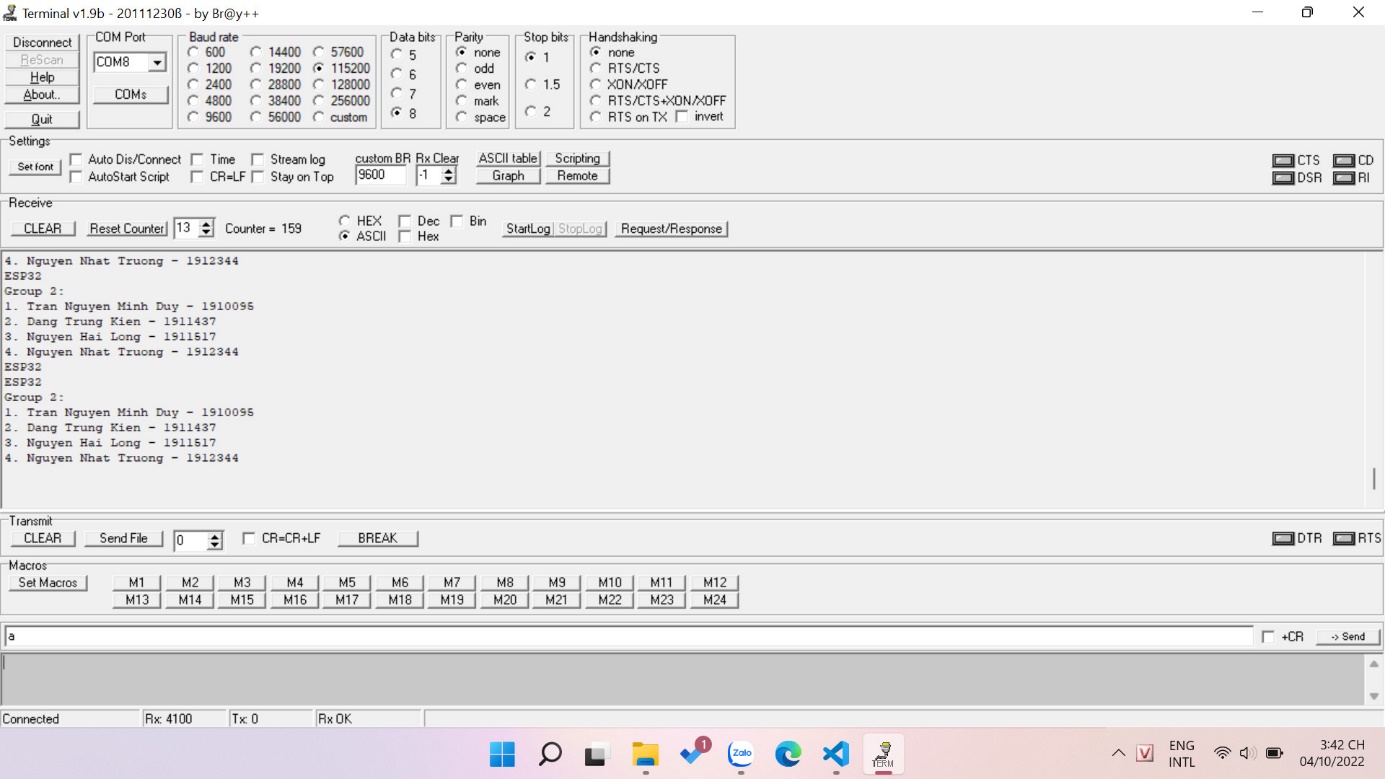
https://github.com/nhokkoranh1511/Embedded\_System\_LAB/tree/main/lab2/Code

1. **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”.



1. **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 đầ