

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



– Embedded System Lab 02 –

ESP32 GPIO and FreeRTOS task

Giáo viên hướng dẫn: Vũ Trọng Thiên

Sinh viên thực hiện: Nguyễn Hữu Hiếu – 2013552 – L03

Lê Bá Dũng – 2012863 – L04

Tp.Hồ Chí Minh, Tháng 10/2023

MỤC LỤC

I.	Giới thiệu	1
1.	Yêu cầu bài của bài lab 02	1
2.	Mục tiêu đề ra	1
3.	ESP32-DevKitC V4 Module WiFi Bluetooth 2.4GHz	1
II.	Hiện thực	3
1.	Yêu cầu về thiết bị và môi trường	3
2.	Mục tiêu đề ra	4
3.	Kết quả	4
	Tài liệu tham khảo	9

I. Giới thiệu

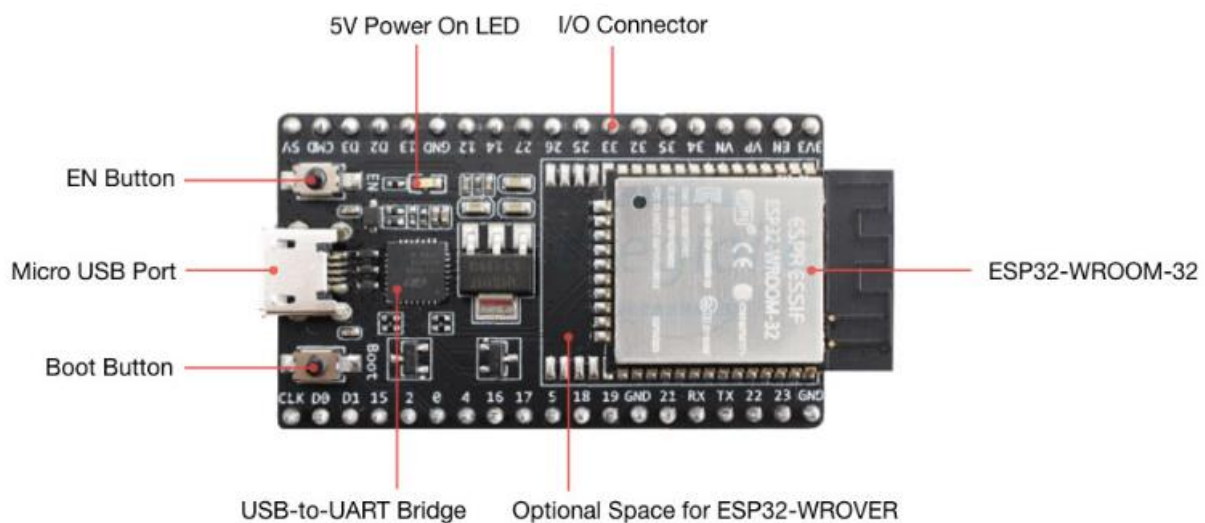
1. Yêu cầu bài của bài lab 02

- Đọc Input và ghi Output cho GPIO pin
- Tạo, lên lịch và xóa FreeRTOS tasks trên mạch lập trình ESP32
- Thực hiện thành công bài tập cuối cùng của bài lab

2. Mục tiêu đề ra

3. ESP32-DevKitC V4 Module WiFi Bluetooth 2.4GHz

ESP32-DevKitC V4 là bo mạch phát triển dựa trên ESP32 cỡ nhỏ do Espressif sản xuất. Hầu hết các chân I/O được chia thành các đầu chân cắm ở cả hai bên để dễ dàng giao tiếp. Các nhà phát triển có thể kết nối các thiết bị ngoại vi bằng dây nhảy hoặc gắn ESP32-DevKitC V4 trên bảng mạch khung.



Hình 1.1 Hình ảnh mạch xử lý

4. Tùy chọn cung cấp điện

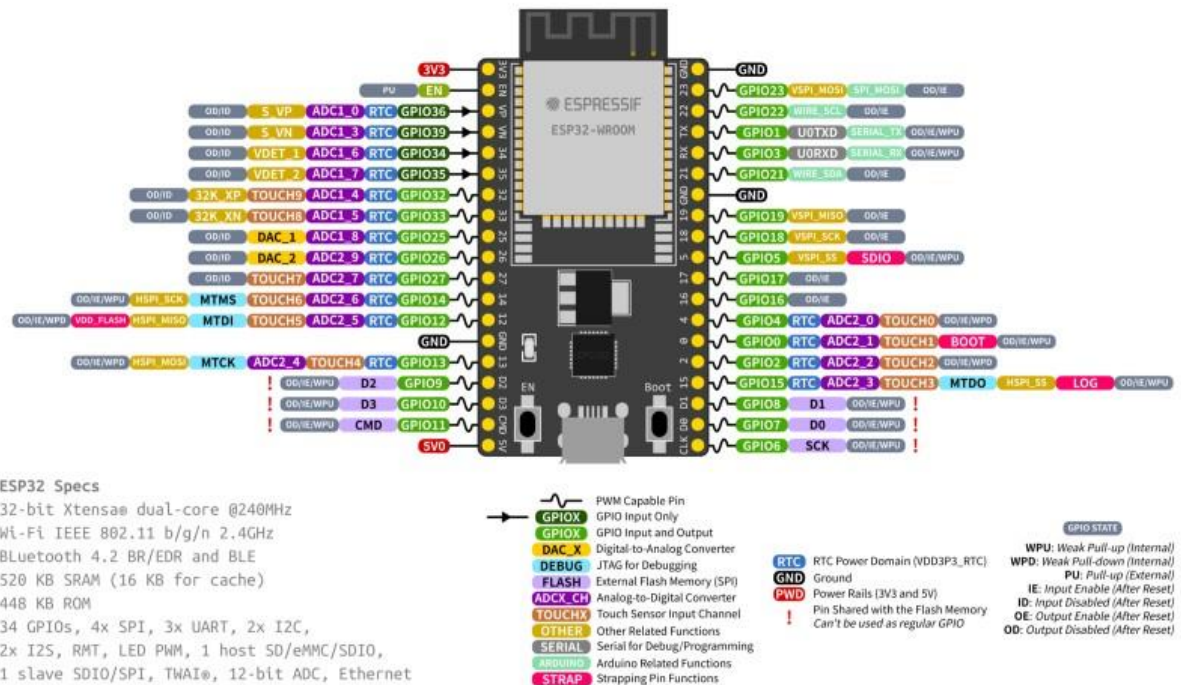
- Đầu nối micro USB cho giao tiếp và nguồn PC
- Cấp nguồn qua đầu nối: 5 V hoặc 3,3 V.

Lưu ý: Nguồn điện phải được cung cấp bằng một và chỉ một trong các tùy chọn ở trên, nếu không bo mạch và/hoặc nguồn điện có thể bị hỏng.

Thông số kỹ thuật

- Module: ESP32-WROOM-32D, 4MB Flash
 - . Bộ xử lý không dây ESP32-D0W (SoC)
 - . CPU kép Xtensa LX6
 - . WiFi 802.11 b/g/n (802.11n - 150 Mbps)
 - . Thông số kỹ thuật Bluetooth v4.2 BR / EDR và BLE
 - . Tần số bộ xử lý được điều chỉnh từ 80 MHz đến 240 MHz.
 - . Số lượng GPIO: 34
 - Cảm biến trường
 - Cảm biến cảm ứng điện dung

- Ăng-ten PCB MIFA
- Bộ chuyển đổi USB-UART (tốc độ trao đổi lên tới 3 Mbit/s)
- Nút boot
- Nút khởi động: Khởi động và EN
- Đèn LED trạng thái nguồn
- Pin I/O: PWM, ADC, DAC, I2C, I2S, SPI,...
- Kích thước : 54.4mm x 27.9mm



Hình 1.2. Pin layout

II. Hiện thực

1. Yêu cầu về thiết bị và môi trường

Thiết bị:

- ESP32-DevKitC V4 Module WiFi Bluetooth 2.4GHz

Môi trường:

- ESP-IDF Version 5.11
- Python Version 3.9.13
- Hệ điều hành máy tính: Window 11
- IDE: VSCode
- Extension: Espressif IDF

2. Mục tiêu đề ra

Về kiến thức:

- Hiểu được về FreeRTOS tasks và các thao tác cơ bản với các tasks: tạo, xóa, sửa, sắp xếp
- Nắm được các tham số của các hàm liên quan đến task.
- Nắm được sơ đồ các chân GPIO của mạch lập trình ESP32 và cách thức sử dụng cơ bản.

Về hiện thực:

- a. Hiện thực một tác vụ tuần hoàn in mã định danh học sinh của bạn mỗi giây.
- b. Hiện thực một tác vụ không theo chu kỳ thăm dò một nút và in "ESP32" mỗi khi nút được nhấn.
- c. Trả lời cho câu hỏi: ESP-IDF có cần hàm lập lịch `vTaskStartScheduler()` không ?

Sự kiện Button - khi nhấn nút bấm :

- In ra ESP32
- Bật sáng đèn LED

Yêu cầu cho nút nhấn:

- Hiện thực debounce cho nút nhấn
- Hiện thực nhấn đè (long press)

3. Kết quả

FreeRTOS trên ESP – IDF.

Trong ESP-IDF, bạn có thể tạo và quản lý các nhiệm vụ FreeRTOS để thực hiện các hoạt động song song trên vi điều khiển ESP3. Dưới đây là các bước cơ bản để tạo và quản lý các nhiệm vụ FreeRTOS trong ESP-IDF:

- . *Khởi tạo FreeRTOS*: ESP-IDF khởi tạo FreeRTOS cho bạn, vì vậy bạn không cần cài đặt FreeRTOS thủ công.
- . *Tạo một tác vụ (Create a Task)*: Bạn có thể tạo một nhiệm vụ mới bằng cách sử dụng hàm `xTaskCreate`. Hàm này có nhiều tham số, bao gồm hàm nhiệm vụ (mã để chạy trong nhiệm vụ), tên nhiệm vụ, kích thước ngăn xếp, độ ưu tiên và nhiều tham số khác.
- . *Đồng Bộ Hóa (Synchronization)*: FreeRTOS cung cấp các cơ chế đồng bộ hóa như semaphore, hàng đợi và mutex để giao tiếp và đồng bộ hóa giữa các nhiệm vụ.
- . *Xóa tác vụ (Task Deletion)*: Bạn có thể xóa một nhiệm vụ khi nó không còn cần bằng cách sử dụng hàm `vTaskDelete`.
- . *Tác vụ ưu tiên (Task Prioritization)*: Bạn có thể gán các độ ưu tiên cho các nhiệm vụ. Các nhiệm vụ có độ ưu tiên cao sẽ chạy trước các nhiệm vụ có độ ưu tiên thấp hơn.
- . *Tác vụ Rảnh Rỗi (Idle Task)*: Bản triển khai FreeRTOS của ESP-IDF cũng có nhiệm vụ rảnh rỗi chạy khi không có nhiệm vụ nào khác sẵn sàng thực hiện. Đây là nơi hệ thống vào chế độ tiết kiệm năng lượng khi thích hợp.

Cấu hình các chân GPIO của ESP32.

- Biết cách cấu hình các chân GPIO của mạch ESP32
- Biết cách nạp và xuất dữ liệu thông qua các chân GPIO

Hiện thực các yêu cầu đề ra.

Bảng II.1. Check list yêu cầu đề ra

<i>No.</i>	<i>Nội dung yêu cầu</i>	<i>Mức độ hoàn thành</i>
a.	Hiện thực một tác vụ tuần hoàn in mã định danh học sinh của bạn mỗi giây.	Hoàn thành. In ra mã số sinh viên mỗi 1 giây.
b.	Hiện thực một tác vụ không theo chu kỳ thăm dò một nút và in "ESP32" mỗi khi nút được nhấn.	Hoàn thành. Mỗi lần nhấn nút nhấn, monitor in ra chuỗi chứa "ESP32". Đèn led sáng

c.	Trả lời cho câu hỏi: ESP-IDF có cần hàm lập lịch <code>vTaskStartScheduler()</code> không ?	Đã trả lời
----	---------------------------------------------------------------------------------------------	------------

Hiện thực.

Set up.

- Hàm setup để định nghĩa các GPIO mà hệ thống sử dụng.
- Chọn các GPIO bằng hàm `gpio_pad_select_gpio(GPIO_PIN);`
- Lựa chọn input/output cho GPIO bằng hàm `gpio_set_direction(GPIO_PIN, MODE_GPIO);`
MODE_GPIO có 2 lựa chọn `GPIO_MODE_INPUT` hoặc `GPIO_MODE_OUTPUT`
- Chọn chế độ cho nút nhấn bằng hàm `gpio_set_pull_mode(GPIO_PIN, MODE)`
- Trong bài lab này nút nhấn được định nghĩa ở chế độ `GPIO_PULLUP_ONLY`

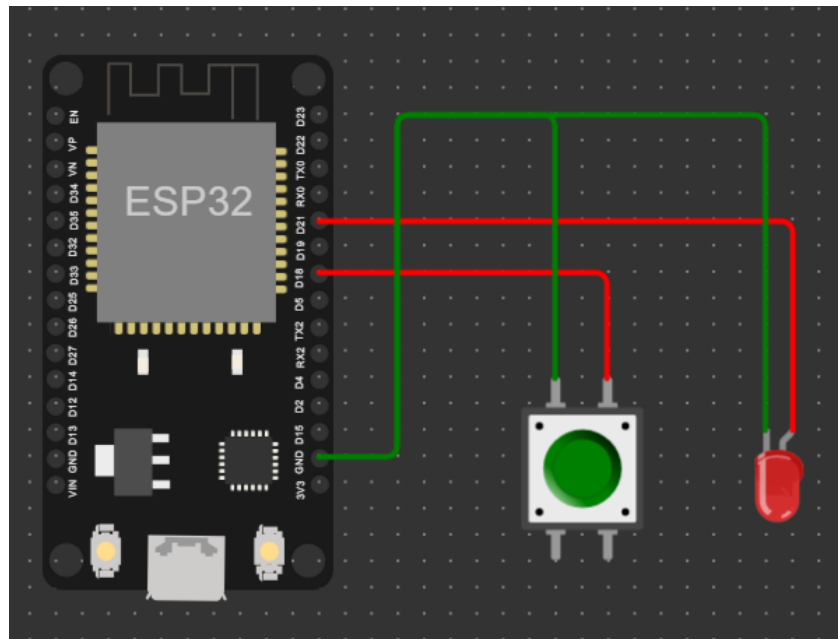
a. Task 1.

- Tạo một task in ra monitor ID của sinh viên và số lần task 1 được thực thi.
- Trong hàm task1, tác vụ đặt trong hàm `while(1)`, điều phối mỗi giây sẽ thực thi lệnh in bằng hàm `vTaskDelay(const TickType_t xTicksToDelay);` `xTicksToDelay` ở đây được tính bằng `time / portTICK_PERIOD_MS` với `portTICK_PERIOD_MS = 1000` (đã được định nghĩa sẵn)

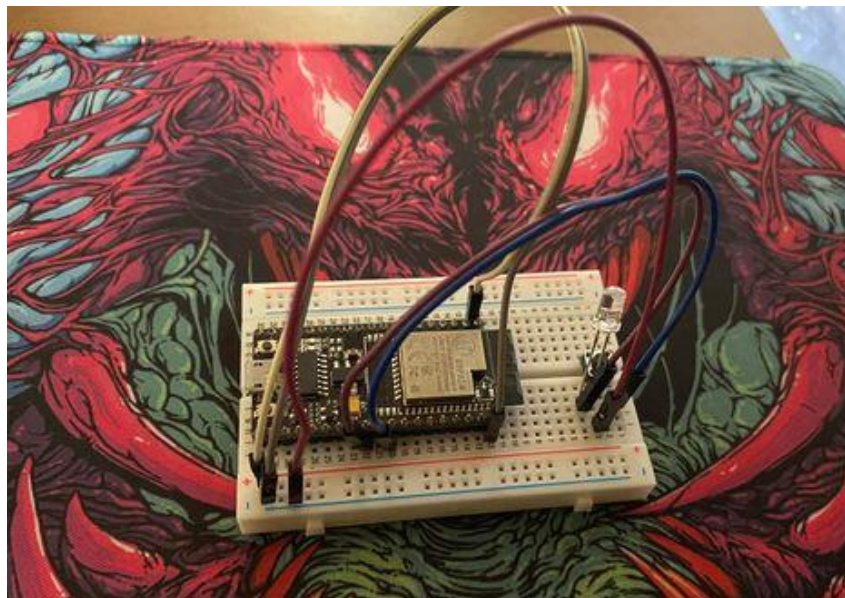
b. Task 2

- Trong hàm hiện thực task 2, tác vụ đặt trong hàm `while(1)`, việc nhận biết sự kiện nút nhấn sẽ nằm ở trong câu lệnh điều kiện, với điều kiện:
`gpio_get_level(BUTTON) == button_state`
- Khi thỏa điều kiện, sẽ in dòng chữ có chứa ký tự “ESP32” ra monitor và bật sáng LED.
- Để debounce cho nút nhấn, ở ngoài câu lệnh điều kiện, nhóm sử dụng `vTaskDelay(time_Delay / portTICK_PERIOD_MS);` với `time_delay = 200`.

Kết quả hiện thực.



Hình II.1. Schematic của hệ thống

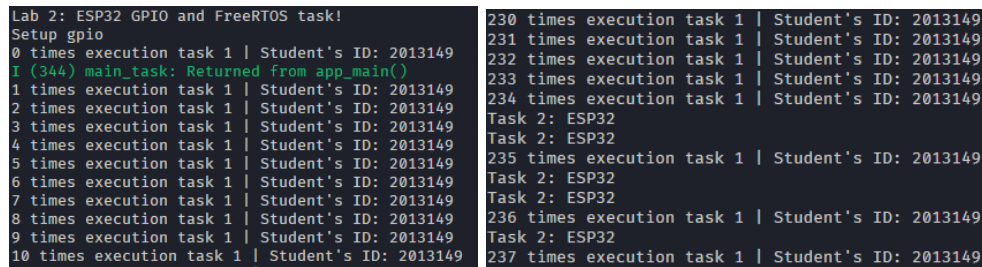


Hình II.2. Sơ đồ thực tế

a,b.

. Hiện thực một tác vụ tuần hoàn in mã định danh học sinh của bạn mỗi giây.

. Hiện thực một tác vụ không theo chu kỳ thăm dò một nút và in "ESP32" mỗi khi nút được nhấn



The image shows two side-by-side terminal windows displaying the output of a FreeRTOS application on an ESP32. The left window shows the execution of Task 1, which prints 'times execution task 1 | Student's ID: 2013149' repeatedly. The right window shows the execution of Task 2, which prints 'Task 2: ESP32' repeatedly. Both tasks are running concurrently, as evidenced by the interleaved output lines.

```
Lab 2: ESP32 GPIO and FreeRTOS task!
Setup gpio
0 times execution task 1 | Student's ID: 2013149
1 (344) main_task: Returned from app_main()
1 times execution task 1 | Student's ID: 2013149
2 times execution task 1 | Student's ID: 2013149
3 times execution task 1 | Student's ID: 2013149
4 times execution task 1 | Student's ID: 2013149
5 times execution task 1 | Student's ID: 2013149
6 times execution task 1 | Student's ID: 2013149
7 times execution task 1 | Student's ID: 2013149
8 times execution task 1 | Student's ID: 2013149
9 times execution task 1 | Student's ID: 2013149
10 times execution task 1 | Student's ID: 2013149
230 times execution task 1 | Student's ID: 2013149
231 times execution task 1 | Student's ID: 2013149
232 times execution task 1 | Student's ID: 2013149
233 times execution task 1 | Student's ID: 2013149
234 times execution task 1 | Student's ID: 2013149
Task 2: ESP32
Task 2: ESP32
235 times execution task 1 | Student's ID: 2013149
Task 2: ESP32
Task 2: ESP32
236 times execution task 1 | Student's ID: 2013149
Task 2: ESP32
237 times execution task 1 | Student's ID: 2013149
```

Hình II.3. Kết quả in ra ở monitor; Task 1 (bên trái) & Task 2 (bên phải)

c. Trả lời cho câu hỏi: ESP-IDF có cần hàm lập lịch `vTaskStartScheduler()` không ?

ESP-IDF không cần hay thậm chí là không nên gọi `vTaskStartScheduler()`.

Giải thích; ESP-IDF FreeRTOS được **khởi động tự động**. Điểm vào là hàm `void app_main(void)` do người dùng xác định. Thông thường, người dùng sẽ sinh ra phần còn lại của tác vụ ứng dụng của họ từ `app_main` .

Video demo kết quả:

https://drive.google.com/file/d/1JooLO_zw9d_qFPXC0FvvuTKVtPGXuIik/view?usp=sharing

Source Code: Đính kèm trong file nén.

Tài liệu tham khảo

- [1] *Get Started*, Truy cập từ: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html>
- [2] *Embedded System Lab 02: ESP32 GPIO and FreeRTOS task*, Pham Hoang Anh & Huynh Hoang Kha