1. **TẠO 3 TASK VỚI PLATFORM IO.**

***File code mình đã up lên text, các bạn theo dõi file code để dễ hiểu hơn trong việc phân tích dưới đây nhé.***

**Yêu cầu đề bài**

Chạy được 3 task cơ bản:

+ Task 1: Chớp tắt led.

+ Task 2: 2 nút nhấn, điều khiển 1 led.

+ Task 3: 1 nút nhấn Pause/Run Task 1

**Ý tưởng**

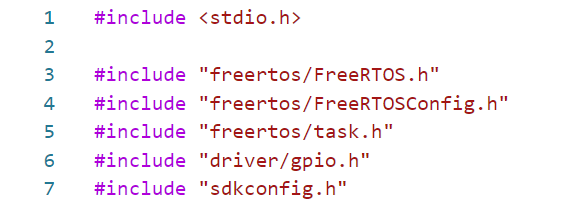
Task 1: Thực hiện chớp tắt LED trên chân 21, mỗi lần chớp tắt 1s.

Task 2: Thực hiện giao tiếp với 2 nút nhấn và LED(chân 18), nút nhấn ON(chân 22) sẽ làm sáng LED, nút nhấn OFF(chân 23) sẽ làm tắt LED.

Task 3: Thực hiện điều khiển task 1 bằng nút nhấn(chân 19), điều khiển task 1 bằng hàm vTaskSuspend() và vTaskResume().

3 task trên sẽ được tạo với mức độ ưu tiên bằng nhau, do đó, chúng sẽ được chạy trên chiến lược điều phối round robin, các hàm đợi ngoại vi giao tiếp (nút nhấn) sẽ đi vào trạng thái ngủ qua hàm vTaskDelay().

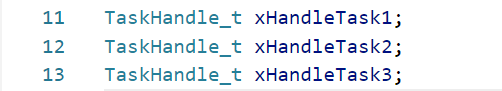
**Phân tích và giải thích code**



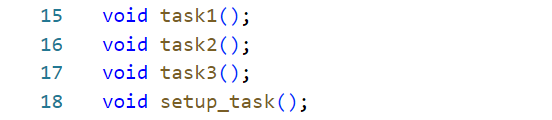
Khai báo các thư viện cần thiết.



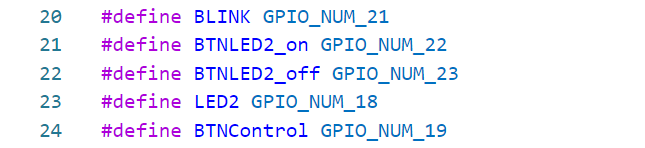
Khai báo biến TURN toàn cục để sử dụng trong task 3, mình sẽ nói kỹ hơn trong phân tích công việc của task 3.



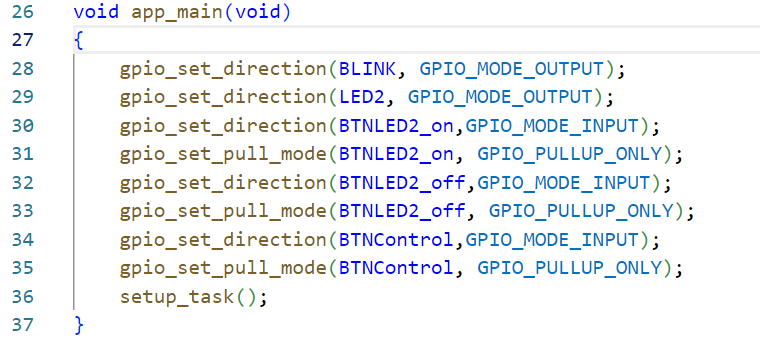
Khai báo các biến điều khiển, biến này sẽ sử dụng để điều khiển task ở bên ngoài task bị điều khiển.



Khởi tạo các hàm công việc trong task và hàm khởi tạo task.



Gán tên cho các chân mình sẽ thực hiện như trong phần ý tưởng để tiện theo dõi.

****

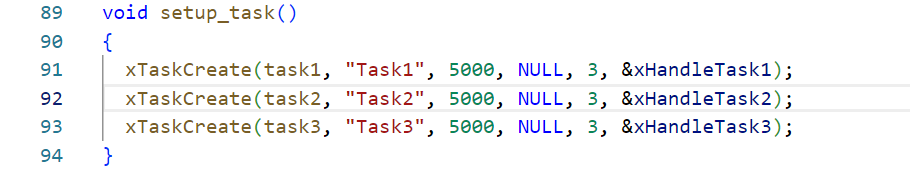
Cấu hình cho các chân trong hàm main() và thực hiện set up task. Các hàm cấu hình:

**gpio\_set\_direction (**[**gpio\_num\_t**](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html#_CPPv410gpio_num_t)**gpio\_num ,**[**gpio\_mode\_t**](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html#_CPPv411gpio_mode_t)**mode ):** Định cấu hình hướng GPIO, chẳng hạn như output\_only, input\_only, output\_and\_input.

[**esp\_err\_t**](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/esp_err.html#_CPPv49esp_err_t)**gpio\_set\_pull\_mode (**[**gpio\_num\_t**](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html#_CPPv410gpio_num_t)**gpio\_num ,**[**gpio\_pull\_mode\_t**](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html#_CPPv416gpio_pull_mode_t)**pull ):** Cấu hình điện trở kéo lên / kéo xuống của GPIO.

Ở đây, các chân điều khiển LED (BLINK, LED2) sẽ cấu hình là output, các nút nhấn sẽ thực hiện giao tiếp nên cấu hình là input, và thực hiện kéo lên (không nhấn thì input là0, nhấn thì input là 1).

Hàm khởi tạo các task:



Khởi tạo các task theo cú pháp:

**xTaskCreate( TaskFunction\_t pvTaskCode,**

**const char \* const pcName,**

[configSTACK\_DEPTH\_TYPE](https://www.freertos.org/a00110.html#configSTACK_DEPTH_TYPE) **usStackDepth,**

**void \*pvParameters,**

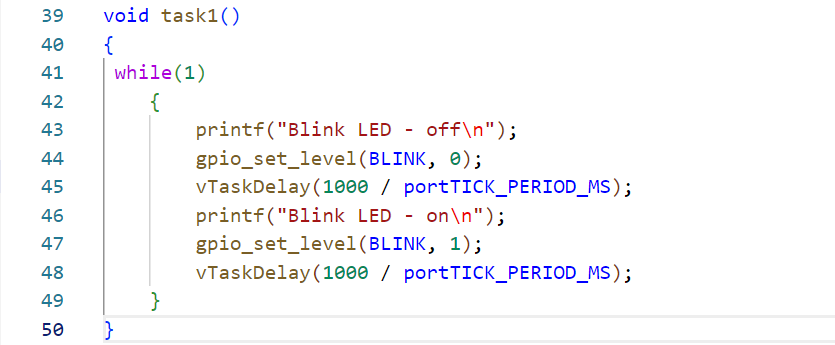
**UBaseType\_t uxPriority,**

**TaskHandle\_t \*pxCreatedTask**

**);**

Với các thông số mình đã giải thích trong bài hướng dẫn đầu. Nhưng ở đây, sẽ có khai báo thêm biến điều khiển trong mục **TaskHandle\_t \*pxCreatedTask,** được khởi tạo trong dòng 11-13. Biến này sẽ được dùng để điều khiển các task.

**Phân tích công việc các task.**



Thực hiện nhấp nháy LED, mỗi lần nhấp nháy 1s.

**gpio\_set\_level (**[**gpio\_num\_t**](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html#_CPPv410gpio_num_t)**gpio\_num , uint32\_t level ):** GPIO đặt mức đầu ra.



Thực hiện giao tiếp nút nhấn với LED, thực hiện kiểm tra nút nhấn ON và nút nhấn OFF của LED:

Nếu nút nhấn ON được nhấn(1) task sẽ đưa mức đầu ra LED lên mức 1, LED sáng.

Nếu nút nhấn OFF được nhấn(1) task sẽ đưa mức đầu ra LED xuống mức 0, LED tắt.

Hàm ở dòng 56 và 61 sẽ có tác dụng treo task 2 nếu nút nhấn ON, OFF được giữ lại (nhấn và giữ), khi đó hết quantum round robin của task 2 thì task khác sẽ vẫn chạy.



Task 3 sẽ thực hiện kiểm tra nút nhấn và điều khiển task 1, khi nút được nhấn, task 3 sẽ ngừng task 1 bằng vTaskSuspend().

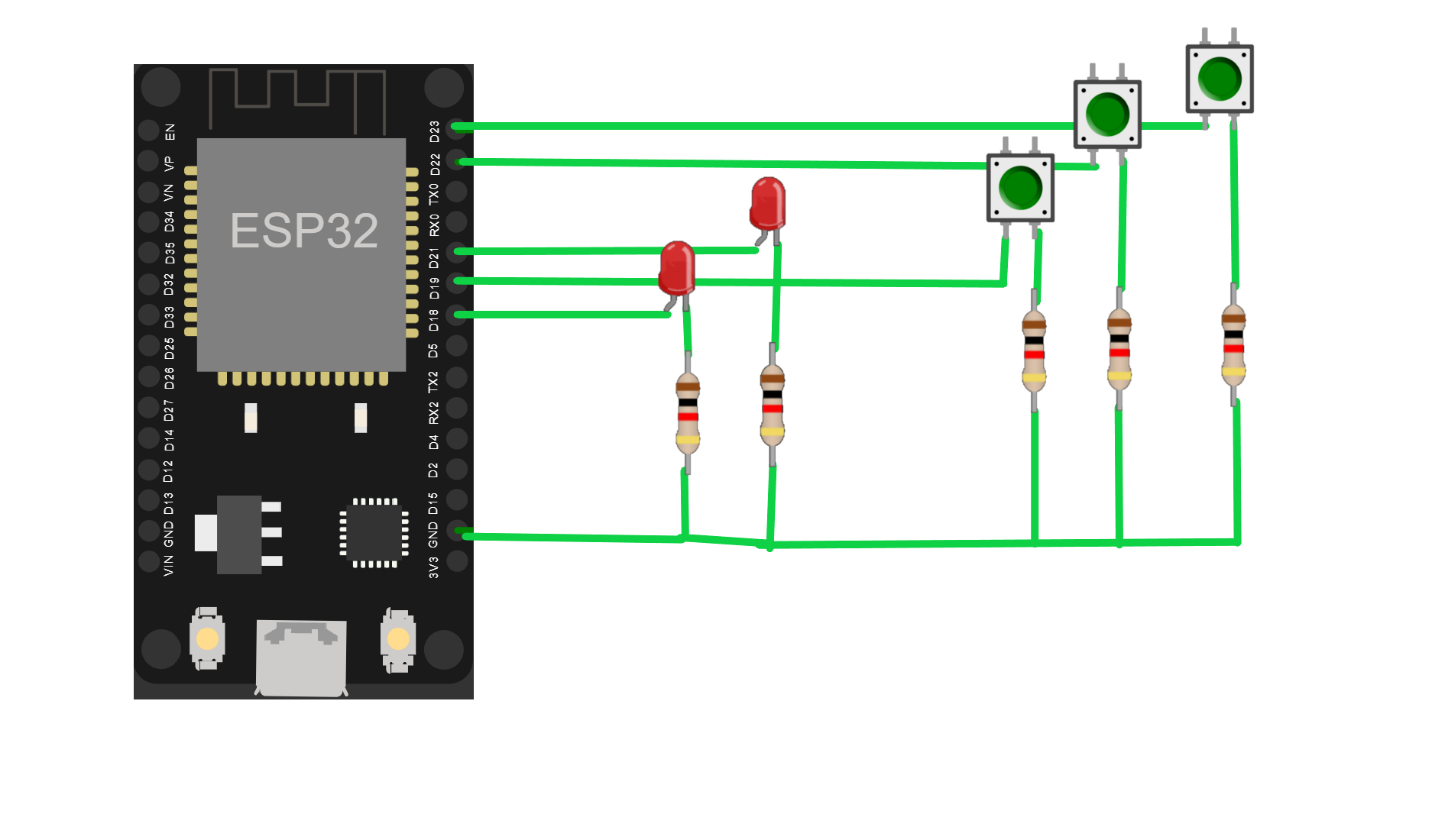
**vTaskSuspend (TaskHandle\_t xTaskToSuspend):** Tạm dừng bất kỳ nhiệm vụ nào. Khi bị treo một tác vụ sẽ không bao giờ nhận được bất kỳ thời gian xử lý vi điều khiển nào, bất kể mức độ ưu tiên của nó là gì.

Khi nút nhấn được nhấn lần nữa, task 3 sẽ điều khiển task 1 tiếp tục chạy bằng vTaskresume() .

**vTaskResume( TaskHandle\_t xTaskToResume ):** Tiếp tục nhiệm vụ bị tạm ngừng.Một tác vụ đã bị tạm ngưng bởi một hoặc nhiều lệnh gọi đến vTaskSuspend () sẽ khả dụng để chạy lại bằng một lệnh gọi duy nhất tới vTaskResume ().

Ở đây biến TURN sẽ có tác dụng là 1 chìa khóa để thực hiện 2 tiến trình con. Tiến trình con thứ nhất là Suspend task 1, tiến trình thứ 2 là Resume task 1, việc thực hiện nhấn nút sẽ thực hiện lần lượt 2 tiến trình con này trong task 3.

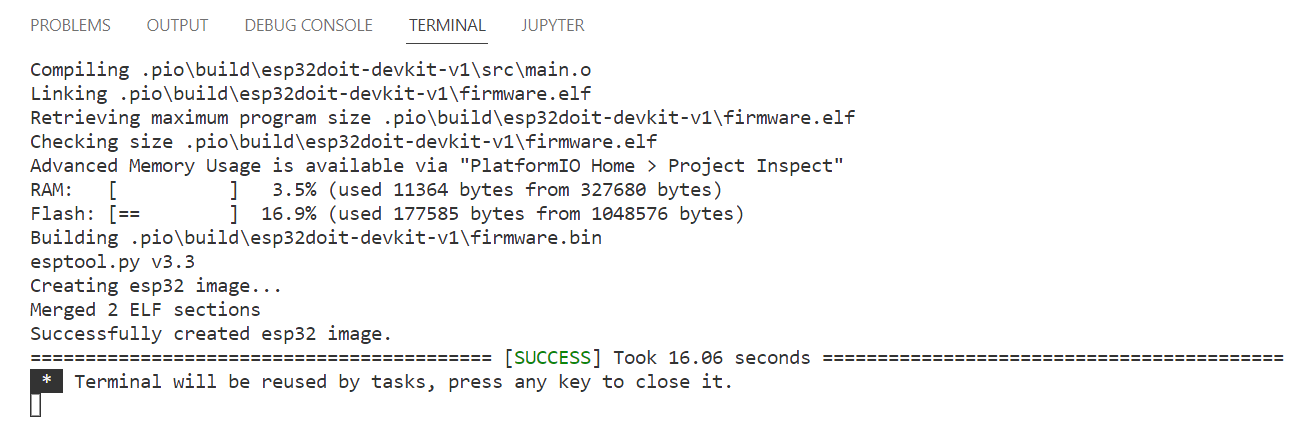
**Sơ đồ nối chân**

****

**Chạy chương trình và nạp code**

****

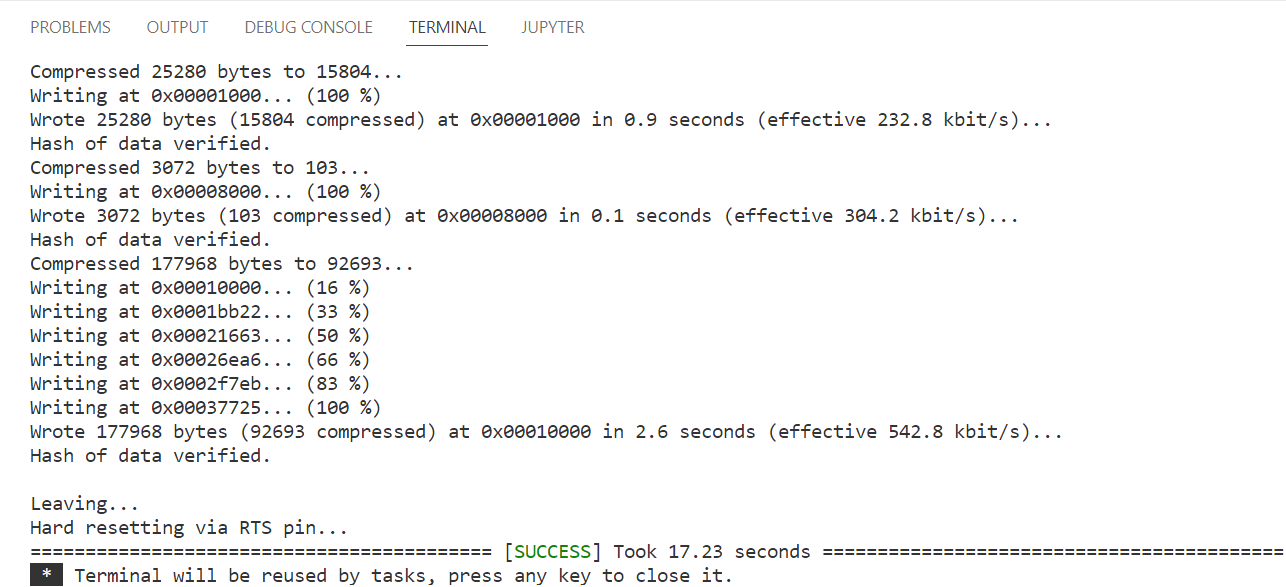
Sau khi code xong, sử dụng nút build để xây dựng chương trình.



Màn hình trên sẽ hiện ra khi build thành công.



Up load code xuống esp32.



Màn hình sau khi up load thành công.

**Kết quả xem tại link:** https://youtu.be/wgbzw1Ixvn8