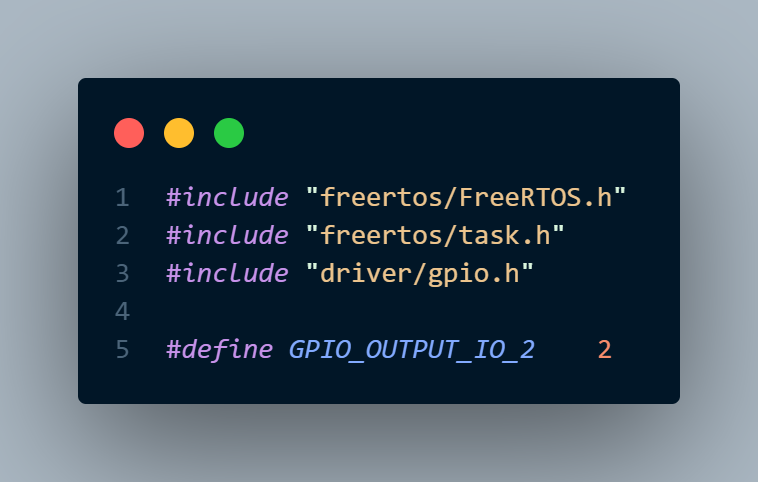
Giải thích Source:



Trong phần này, ta khai báo các thư viện cần thiết để sử dụng FreeRTOS, thư viện GPIO và khai báo một macro để định nghĩa chân GPIO 2 làm chân xuất.

Giải thích thêm: Tại sao lại chọn GPIO 2 làm chân xuất?  
  
-> Module để nhúng đoạn code trên là ESP32 DEV KIT v1 và GPIO 2 được nối với led màu xanh có sẵn trên module.

Text

Description automatically generated

Phần chính của code này là hàm **app\_main()** - đây là hàm chính được thực thi khi chương trình được khởi động. Trong phần này, ta sử dụng hàm **esp\_rom\_gpio\_pad\_select\_gpio()** để chọn chân GPIO 2 làm chân GPIO. Sau đó, ta sử dụng hàm **gpio\_set\_direction()** để cấu hình chân GPIO 2 làm chân xuất. Ta khai báo biến **led** với giá trị ban đầu là **false**. Trong vòng lặp, ta sử dụng toán tử **!** để đảo ngược trạng thái của biến **led**, sau đó sử dụng hàm **gpio\_set\_level()** để điều khiển trạng thái của LED tương ứng với giá trị của biến **led**. Cuối cùng, ta sử dụng hàm **printf()** để in ra thông tin trạng thái của LED (ON hoặc OFF) và sử dụng hàm **vTaskDelay()** để tạm dừng thực thi chương trình trong 1 giây. Lưu ý rằng **vTaskDelay()** sử dụng thời gian tính bằng mili giây, do đó ta phải chia cho **portTICK\_PERIOD\_MS** để chuyển đổi thành đơn vị ticks của FreeRTOS.

**portTICK\_PERIOD\_MS** là một hằng số được định nghĩa trong các thư viện của FreeRTOS để chuyển đổi giữa thời gian tính bằng mili giây và ticks của FreeRTOS. Giá trị của **portTICK\_PERIOD\_MS** phụ thuộc vào tần số của tick được cấu hình cho FreeRTOS. Mặc định, tần số tick được cấu hình là 1000Hz (1ms), do đó giá trị của **portTICK\_PERIOD\_MS** là 1.

Vì vậy, khi ta sử dụng **vTaskDelay(1000 / portTICK\_PERIOD\_MS)**, thì ta đang yêu cầu tạm dừng thực thi chương trình trong 1000ms (1 giây). Sau khi chạy xong hàm này, FreeRTOS sẽ đếm số tick cần thiết để tạm dừng chương trình trong 1 giây.