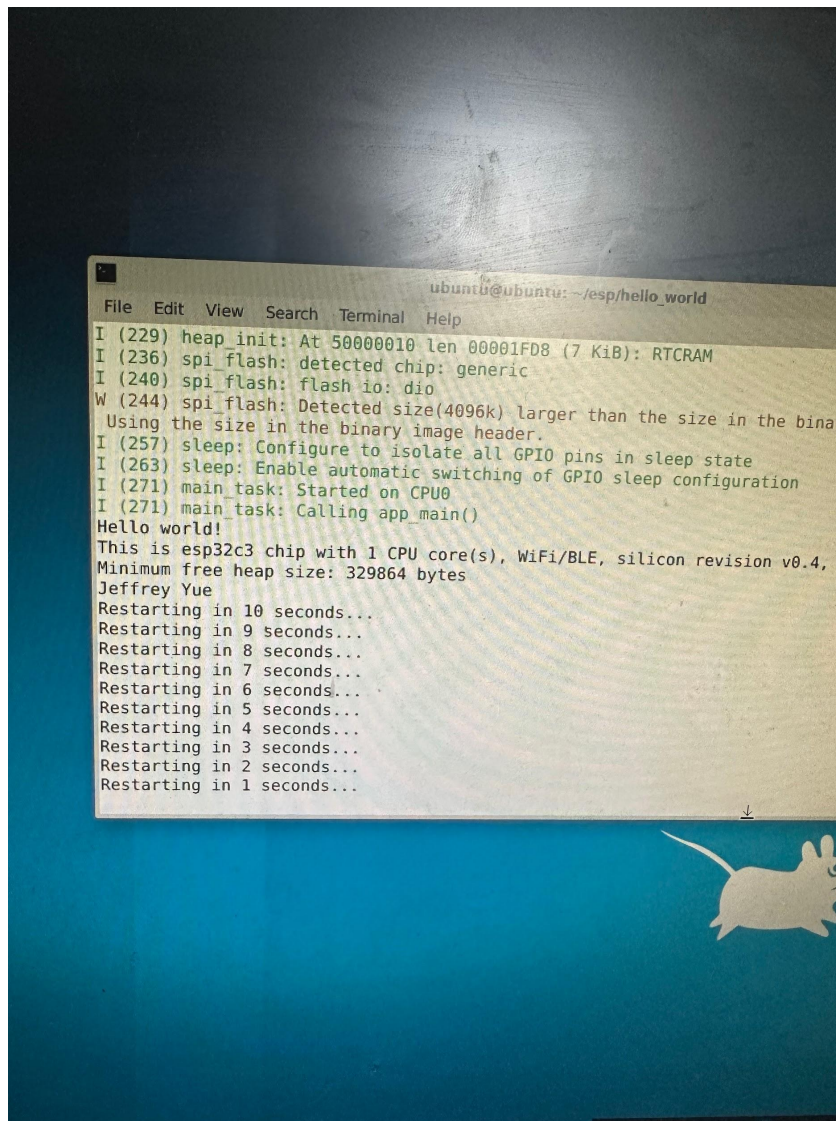


The first lab in the class basically just demonstrated how to use the Raspberry Pi and its programs, mainly the ESP32C3 board, and its subsequent program, esp-idf.

### Lab 1\_2 Hello World:

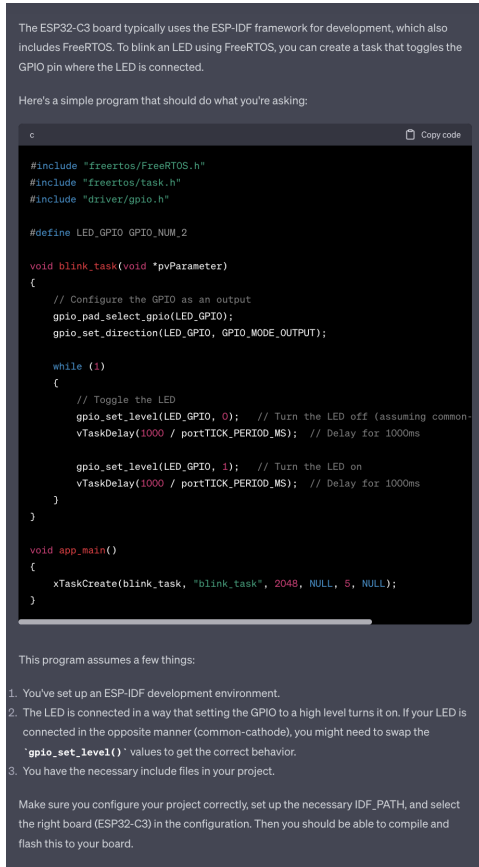
All I did was change the name on the file to my name, Jeffrey Yue, this was the result when I ran the program. It restarts every 10 seconds and prints the default message along with my name.



### Lab 1\_3

Asked Chatgpt: “How do i write a simple c program that flashes an led on/off every second using an esp32c3 board, where it has an led connected to GPIO2?”

This was the response that I received from the app.



The ESP32-C3 board typically uses the ESP-IDF framework for development, which also includes FreeRTOS. To blink an LED using FreeRTOS, you can create a task that toggles the GPIO pin where the LED is connected.

Here's a simple program that should do what you're asking:

```
c
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"

#define LED_GPIO GPIO_NUM_2

void blink_task(void *pvParameter)
{
    // Configure the GPIO as an output
    gpio_pad_select_gpio(LED_GPIO);
    gpio_set_direction(LED_GPIO, GPIO_MODE_OUTPUT);

    while (1)
    {
        // Toggle the LED
        gpio_set_level(LED_GPIO, 0); // Turn the LED off (assuming common
        vTaskDelay(1000 / portTICK_PERIOD_MS); // Delay for 1000ms

        gpio_set_level(LED_GPIO, 1); // Turn the LED on
        vTaskDelay(1000 / portTICK_PERIOD_MS); // Delay for 1000ms
    }
}

void app_main()
{
    xTaskCreate(blink_task, "blink_task", 2048, NULL, 5, NULL);
}
```

This program assumes a few things:

1. You've set up an ESP-IDF development environment.
2. The LED is connected in a way that setting the GPIO to a high level turns it on. If your LED is connected in the opposite manner (common-cathode), you might need to swap the 'gpio\_set\_level()' values to get the correct behavior.
3. You have the necessary include files in your project.

Make sure you configure your project correctly, set up the necessary IDF\_PATH, and select the right board (ESP32-C3) in the configuration. Then you should be able to compile and flash this to your board.

The esp board flashes as it should, and it should be able to be tested from the zip file. Hard to time a picture for it, unfortunately.

Resources: ChatGPT  
chat.openai.com

<https://docs.espressif.com/projects/esp-matter/en/latest/esp32/faq.html>

APACHE license:

<https://drive.google.com/drive/u/1/folders/15uqJ8zO42qNE-cszcmcdxg0Aig6x94aC>