Oregon State University

ECE 406: Project

To-Do List and Log

Travis Fredrickson

fredritr@oregonstate.edu

Date: 2024-12-01

# Table of Contents

[Table of Contents 2](#_Toc183941770)

[To-Do List 3](#_Toc183941771)

[Done 3](#_Toc183941772)

[To-Do 3](#_Toc183941773)

[Log 3](#_Toc183941774)

[2024-11-28 3](#_Toc183941775)

[2024-11-30 3](#_Toc183941776)

[10:00-11:15 4](#_Toc183941777)

[12:15-15:15 4](#_Toc183941778)

[2024-12-01 4](#_Toc183941779)

[9:15-10:00 4](#_Toc183941780)

# To-Do List

## Done

1. Set up VS Code and ESP toolchain.
2. Get an example program working.

## To-Do

1. Custom program 1: Combine blink and hello world functionality, such as blink the RGB LED and print to terminal.
2. Custom program 1: Change RGB LED color and print statements based on external inputs to GPIOs being high or low.
3. FreeRTOS: Learn to configure FreeRTOS.
4. FreeRTOS: Demonstrate real-time or preemptive software.
5. Rust: Get Rust code to compile within C code project.
6. Rust: Demonstrate Rust code functioning.
7. GUI.
8. Inter-communication.

# Log

## 2024-11-28

* Got VS Code with the ESP toolchain set up.
* Built and flashed an example program.
* COM port was tricky, it was not being detected properly. The computer did not have a USB to UART driver, I could have found one, but switching to the USB only port on the development board worked instead. I also had to switch the cable from the USB 2.0 to 3.0 port on the computer for it to work, weird.
* FreeRTOS seems to already be included in the ESP toolchain, as evidenced by it in the blink example main file included header files. Now I must figure out if is enough and how I can use it to demonstrate RTOS functionality.

## 2024-11-30

### 10:00-11:15

* Hello world example project. It outputs information to the VS Code terminal.
* Compared example projects to understand default code better.

### 12:15-15:15

* Setting up my own custom project.
* Investigated RGB LED. Toggled by GPIO 8, yet I am unable to measure a voltage change on it, weird. Initially, CONFIG\_BLINK\_GPIO is set to 5, but after building it changes to 8. This is apparently set using the configuration menu (idf.py menuconfig).
* idf.py menuconfig can be easily accessed via the gear icon; how magical.
  + A screenshot of a computer

    Description automatically generated
* Further, apparently the RGB LED is not driven by GPIO 8, rather GPIO 8 is used as a data signal for the RMT protocol which programs the RGB LED. This would explain why my cheap multimeter is unable to detect a voltage; the voltage is probably changed in quick, narrow bursts of data, not prolonged (1 second) power levels.
* Trying to build custom program. Unsure about if led\_strip.h will auto-magically be included or where it comes from. It is not auto-magically included.

## 2024-12-01

### 9:15-11:45

* Trying to understand how led\_strip.h or similar can be included in custom projects. What is it exactly, an API?
  + <https://docs.espressif.com/projects/esp-idf/en/v5.3.1/esp32/api-reference/api-conventions.html>
  + <https://docs.espressif.com/projects/esp-idf/en/v5.3.1/esp32/api-guides/tools/idf-component-manager.html>
  + <https://components.espressif.com/components?q=target:esp32c6>
  + <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/build-system.html>
* In VS code, type >component, click ESP-IDF: Create New ESP-IDF Component, then browse for a component, and click install. Magic!
* Setting up a GitHub repo.
* Set up GitHub repo.
* Got LED to work.
* Added hello world print info to custom project.