CSE 121: Lab 1 Report

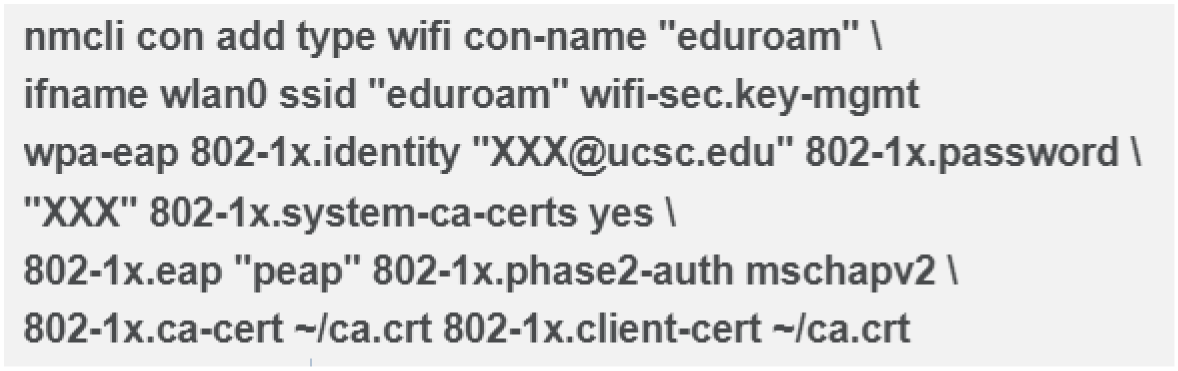
By Daniel Beltran

10/10/2023

Lab 1: Part 1

During the first part of the lab, everything went smoothly up until step 11 where we needed to connect to eduroam. I was attempting to setup the raspberry pi with the Ubuntu Server early on which led to problems since BELS had not whitelisted the chip onto eduroam. It wasn't until a piazza post had gathered the attention of the staff that they had provided us with additional information to fix the issue. Initially, I had thought it was an issue with the Ubuntu Server I had downloaded and there were a few extra do-overs. The terminal command and certification that helped with connection to eduroam listed here:

Certification: https://its.ucsc.edu/wireless/eduroam-manual-config.html



```
nmcli con add type wifi con-name "eduroam" \
ifname wlan0 ssid "eduroam" wifi-sec.key-mgmt
wpa-eap 802-1x.identity "XXX@ucsc.edu" 802-1x.password \
"XXX" 802-1x.system-ca-certs yes \
802-1x.eap "peap" 802-1x.phase2-auth mschapv2 \
802-1x.ca-cert ~/ca.crt 802-1x.client-cert ~/ca.crt
```

Lab 1: Part 2

The second part of the lab was configuring the ESP32C board onto our raspberry pi and running the hello_world_main.c file. Overall, the instructions and installation of the many packages needed were fairly easy with no serious problems. The only real issues I had was attempting to run **idf.py flash** and **idf.py monitor,** it did not run the command but I realized that I needed to be in the directory before to test onto the chip.

Lab 1: Part 3

During the third part of the lab there were many resources that were utilized for lighting up the GPIO2 LED slot while experiencing issues throughout. Under the slide "Main Docs", given by Professor Marcelo's, provided a link that allowed me to properly look into the hardware of an ESP32C3, along with its source code. The link below to the first link is below:

Lecture 1 Slides: https://www.espressif.com/en/products/devkits

As the lab1_3.c file's purpose was to blink the GPIO2 LED for a second each time, I was able to find the blink example within "examples/get-started/blink/main/blink_example_main.c." My reasoning behind this was to get a better understanding of the certain functionalities that would be needed to properly implement the blinking. The link and code that was utilized will be shared below:

ESP32C3 Source Code:
https://github.com/espressif/esp-idf/blob/master/examples/get-started/blink/main/blink_example_main.c

```
static void blink_led(void)
{
    /* If the addressable LED is enabled */
    if (s_led_state) {
        /* Set the LED pixel using RGB from 0 (0%) to 255 (100%) for each color */
        led_strip_set_pixel(led_strip, 0, 16, 16, 16);
        /* Refresh the strip to send data */
        led_strip_refresh(led_strip);
    } else {
        /* Set all LED off to clear all pixels */
        led_strip_clear(led_strip);
    }
}
```

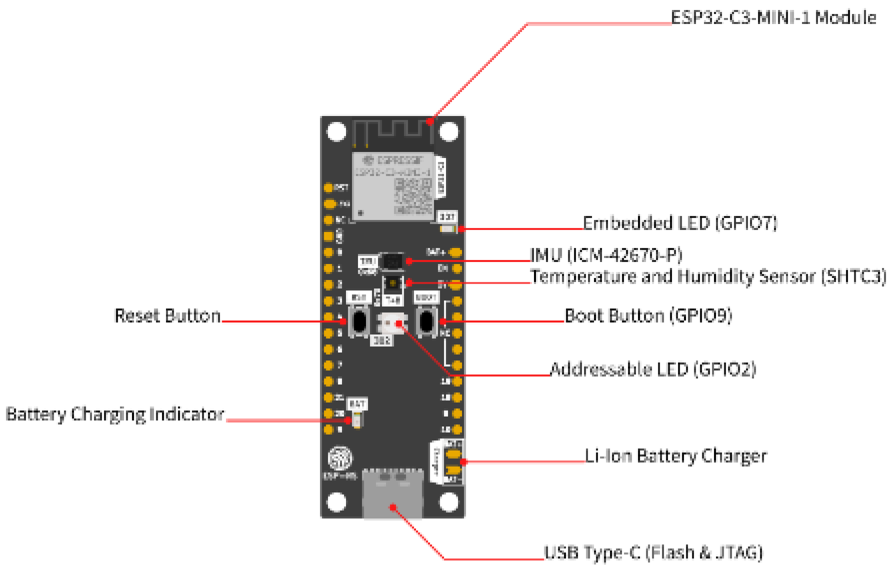
```
static void configure_led(void)
{
    ESP_LOGI(TAG, "Example configured to blink addressable LED!");
    /* LED strip initialization with the GPIO and pixels number*/
    led_strip_config_t strip_config = {
        .strip_gpio_num = BLINK_GPIO,
        .max_leds = 1, // at least one LED on board
    };
#if CONFIG_BLINK_LED_STRIP_BACKEND_RMT
    led_strip_rmt_config_t rmt_config = {
        .resolution_hz = 10 * 1000 * 1000, // 10MHz
        .flags.with_dma = false,
    };
    ESP_ERROR_CHECK(led_strip_new_rmt_device(&strip_config, &rmt_config, &led_strip));
#elif CONFIG_BLINK_LED_STRIP_BACKEND_SPI
    led_strip_spi_config_t spi_config = {
        .spi_bus = SPI2_HOST,
        .flags.with_dma = true,
    };
    ESP_ERROR_CHECK(led_strip_new_spi_device(&strip_config, &spi_config, &led_strip));
```

```
void app_main(void)
{

    /* Configure the peripheral according to the LED type */
    configure_led();

    while (1) {
        ESP_LOGI(TAG, "Turning the LED %s!", s_led_state == true ? "ON" : "OFF");
        blink_led();
        /* Toggle the LED state */
        s_led_state = !s_led_state;
        vTaskDelay(CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
    }
}
```

Many hours went by by myself looking over the documentation and seeing if anything would piece together. Initially, I had caused blinking for GPIO7 as I had followed the "app_main" function's description and realized later on that though there two LEDS they have their own differences. Below is a screenshot of the Professor's lecture showing the two LEDs Addressable LED (GPIO2) and Embedded LED (GPIO7)



Looking up what an addressable LED is and gave me a better understanding of what it is, a strip LED. With that in mind, searching through the source code with and seeing the header inclusion of "include led_strip.h" which was located at the top of the file. From the screenshots above, I was able to piece the code together for the GPIO2 LED to blink properly for a second! Of course the download of the dependency to use "led_strip.h" below. Also when creating the blink_task function we passed in some pointer in the parameter labeled **"pvParameters "** which will be cited below as well.

GPIO2 Info: https://core-electronics.com.au/guides/fully-addressable-rgb-raspberry-pi/

Dependency: https://components.espressif.com/components/espressif/led_strip

Pointer Parameter: https://www.freertos.org/a00125.html