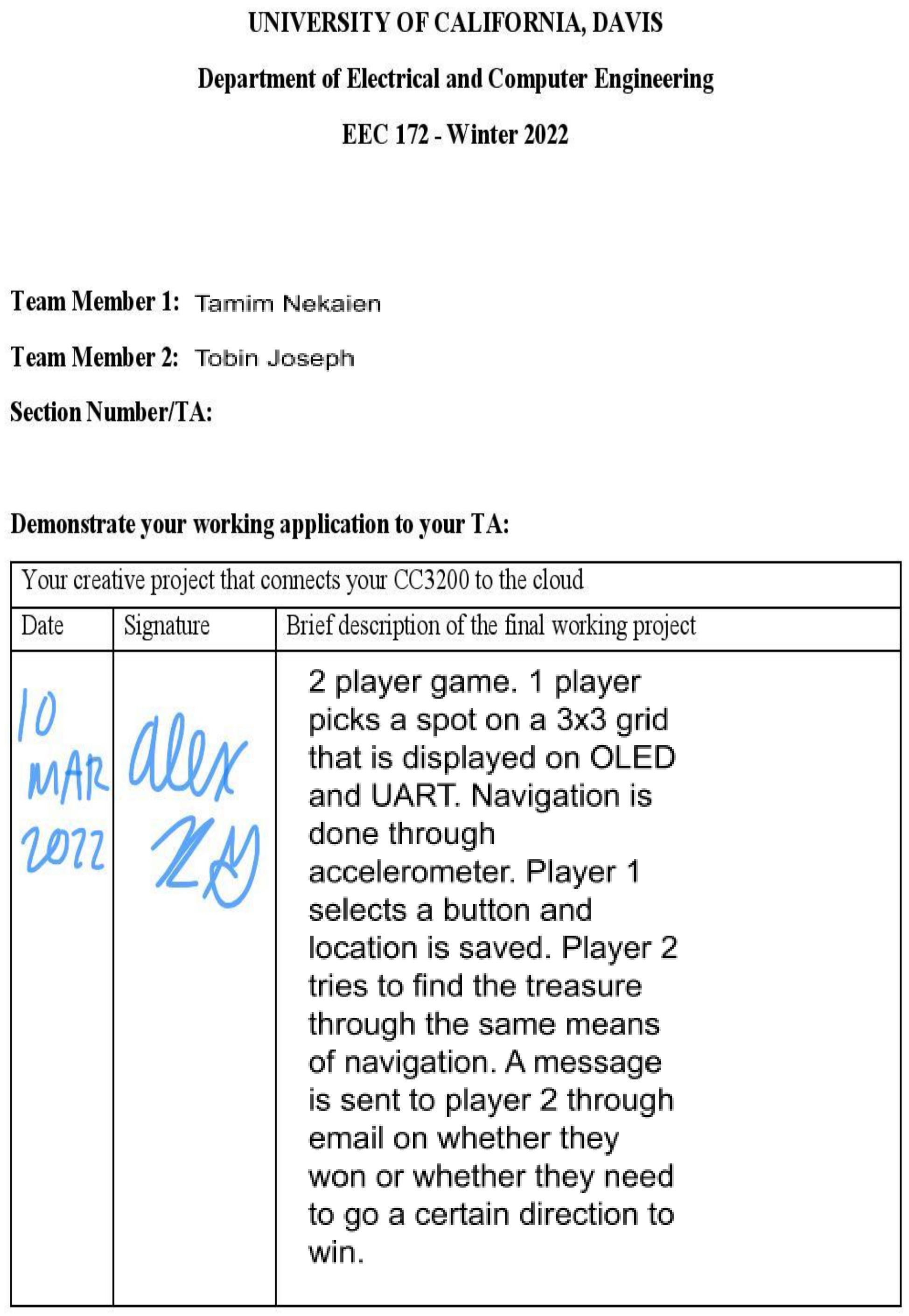


UNIVERSITY OF CALIFORNIA, DAVIS

Department of Electrical and Computer Engineering

EEC 172 - Winter 2022

Team Member 1: Tamim Nekaien

Team Member 2: Tobin Joseph

Section Number/TA:

Demonstrate your working application to your TA:

Your creative project that connects your CC3200 to the cloud		
Date	Signature	Brief description of the final working project
10 MAR 2022	alex [illegible]	2 player game. 1 player picks a spot on a 3x3 grid that is displayed on OLED and UART. Navigation is done through accelerometer. Player 1 selects a button and location is saved. Player 2 tries to find the treasure through the same means of navigation. A message is sent to player 2 through email on whether they won or whether they need to go a certain direction to win.

Tamim Nekaien Tobin Joseph

EEC172 with Professor Ghiasi

Lab#5 Comprehensive Report

Open-Ended IoT Project: Treasure Hunt

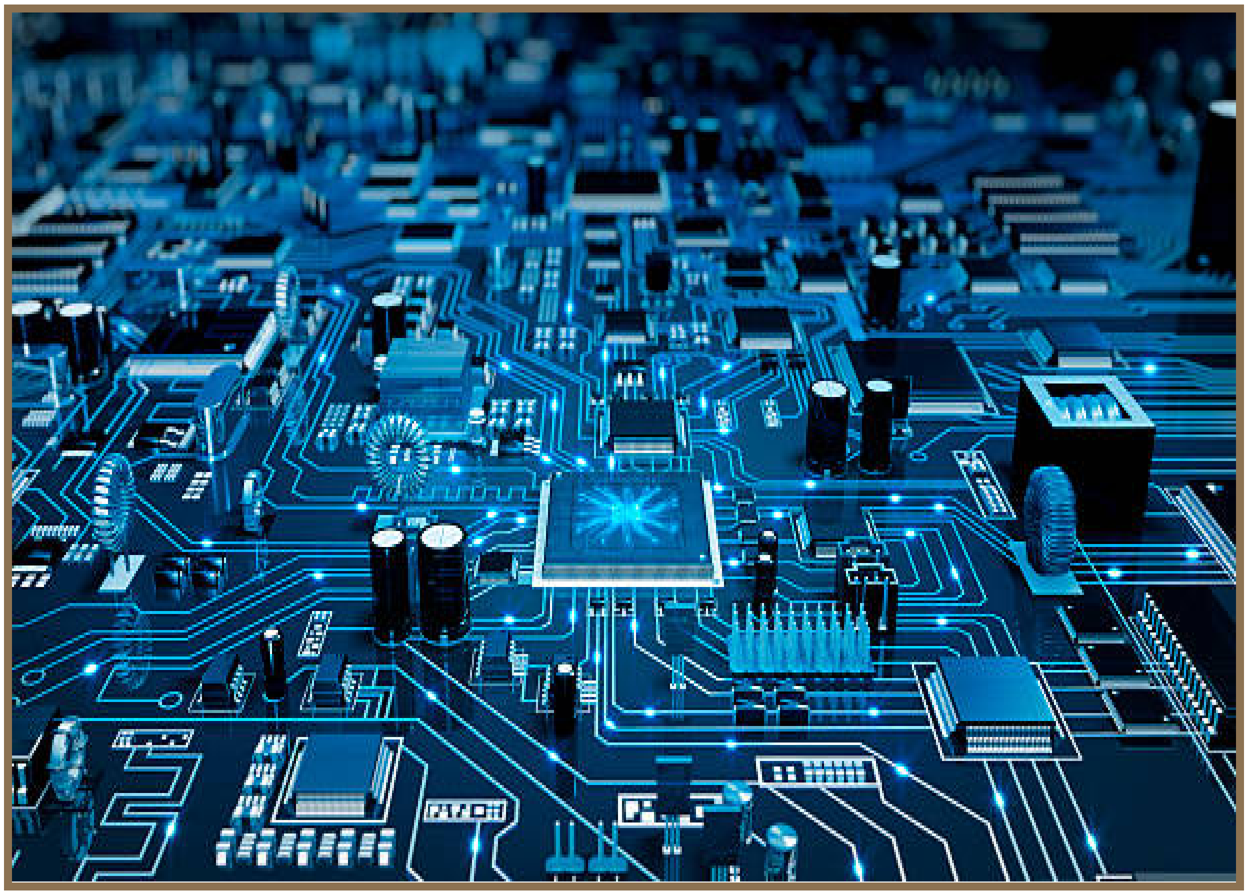


Table of Contents

Introduction

Overview

The purpose of this lab was to create an open-ended lab that would make use of the concepts from the previous labs. The requirements were that the project be:

- Interactive
- Control or Monitor a thing over the internet using a combination of devices (sensor, Oled, remote, switch)
- Run on its own with binary file flashed onto device. The CC3200 can, however, still be powered through a laptop USB. External battery attachment is also possible.

Considering these tasks, we have fulfilled all requirements. Our device interacts with users through button pushes and accelerometer data. Our thing monitors whether the user wins or loses and sends a message through Amazon AWS SNS. And the device was able to run on its own.

Files Used

- main.c
- common.h
- Pinmux.c
- Certificates (starfield, private, and client)

Hardware Used

- CC3200 device (BMA222, buttons)
- OLED

Software Used

- Code Composer Studio
- CCS Uniflash
- Pin Mux tool
- Amazon IoT Core, AWS, and RESTful API

Background

General Description of Tasks

1. Task 1: Establishing internet connection
    - Get new certificates, private, and public keys
    - Establish thing device shadow, set up SNS email endpoint
    - Edit main.c to include new host header, post header, and server name
    - Reflash device
    - Edit common.h with correct wifi router information
2. Task 2: Program game so that it works on UART
    - Create 3x3 game board function
    - Take in accelerometer input and have it move marker “X” around the grid
    - Create game conditions for playing state, winning state, and losing state
3. Task 3: Program game so that it works on OLED in sync with UART serial terminal
    - Create 3x3 game board function
    - Take in accelerometer input and have it move marker “X” around the grid
    - Create game conditions for playing state, winning state, and losing state
4. Task 4: Combining previous tasks and debugging
    - Make sure the terminal and OLED display the same output at the same time. Make sure the http_post displays the correct message depending on the losing hit. Create a state machine to clarify which player is supposed to be playing.

Goals

What concepts are being reviewed

1. Task 1
    - In this part, we are reviewing what we learned in lab 2. By recreating the certificate and converting the file to .der format, we are solidifying our previously gained knowledge.
2. Task 2

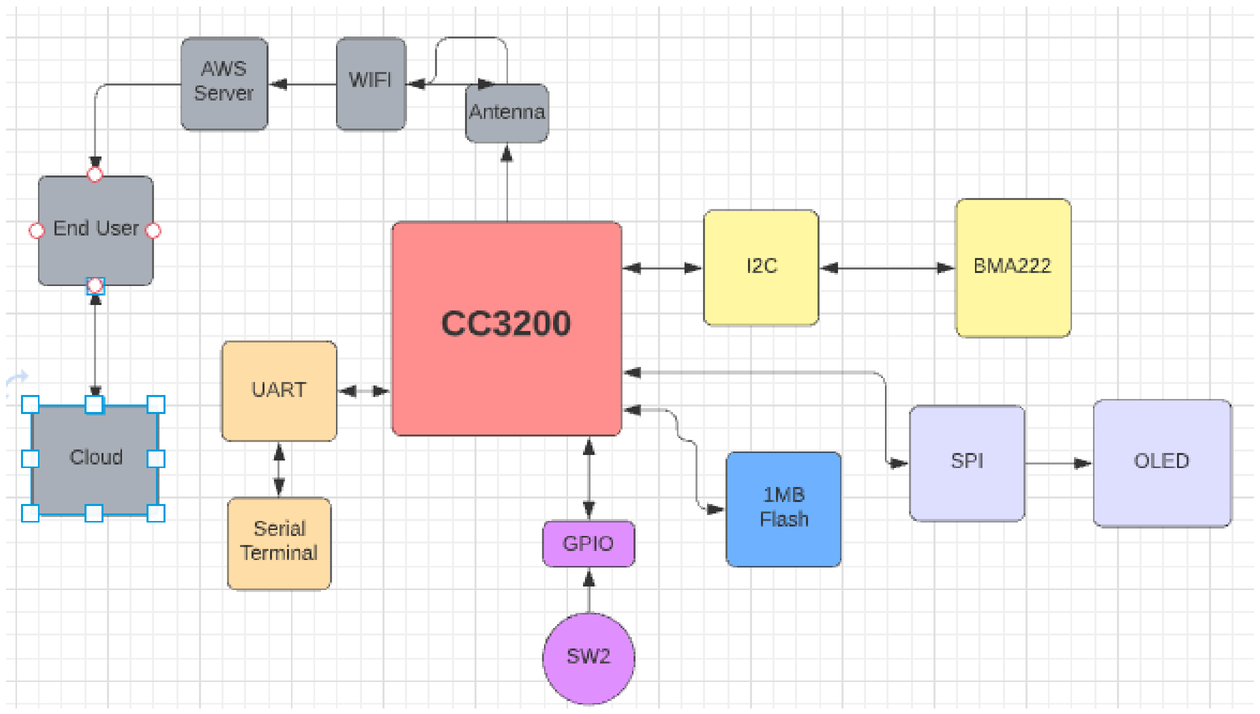
   - This is a software heavy project. So for this part, we relearn how to create a matrix in our C code to then be displayed on the terminal. We also relearn reading accelerometer data to have the information displayed on the terminal.

3. Task 3
   - This part of the lab is a review of lab 3. Here we use the pinmux to make sure the OLED is connected correctly, although we made no changes in pin configuration since lab 3. We also did not change any of the supporting files for adafruit display including GFX and OLED.c. We do however create a unique function for displaying our grid and messages to the player.

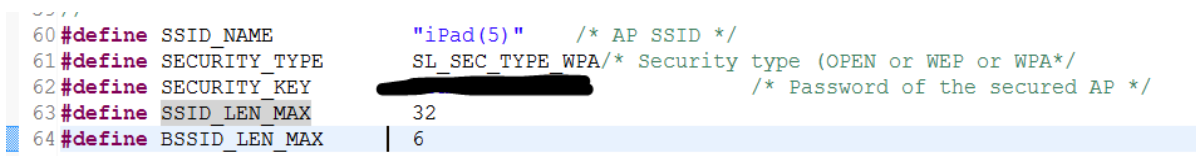
Methods

Top Level Description of PROJECT w/ Block Diagram of Hardware

- Gameplay
  - Player 1 uses CC3200 to mark a spot to play treasure
    - 3x3 grid is displayed on terminal and OLED simultaneously
    - Button on CC3200 marks spot, accelerometer used to move the cursor to the desired cell in the grid.
  - Player 2 uses CC3200 to find the treasure spot
    - 3x3 grid is displayed on terminal and OLED simultaneously
    - The button, SW2, on CC3200 picks a spot, accelerometer used to move the cursor to the desired cell in the grid.
    - Amazon Aws sends sns message to the email account of player 2. Telling the player they either won, or should have moved “up, down, left, right” to win.
- Hardware and Software Connections
  - We start with the CC3200
    - Interface with accelerometer through I2C.
    - Interface with OLED using SPI
    - Interface with a serial terminal via UART
    - Interface with Amazon AWS through the cloud.

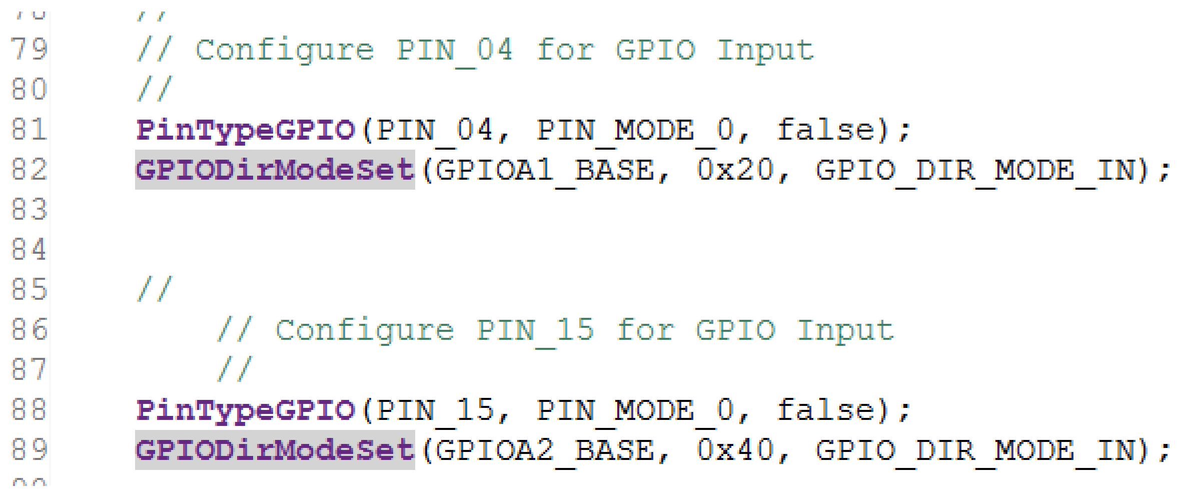


Detailed Description of Internet Connection

- Common.h:
  - In this file, I change the #define SSID_NAME to "iPad(5)". This is my personal hotspot.
  - I also inputted #define SECURITY_KEY. This is the password to my hotspot
  - 

```
#define SSID_NAME           "iPad(5)"    /* AP SSID */
#define SECURITY_TYPE       SL_SEC_TYPE_WPA/* Security type (OPEN or WEP or WPA*/
#define SECURITY_KEY     [redacted]                 /* Password of the secured AP */
#define SSID_LEN_MAX        32
#define BSSID_LEN_MAX       6
```

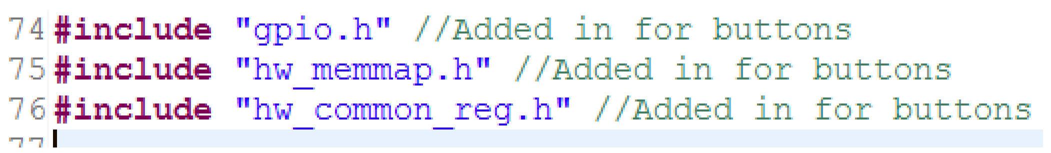
- Pinmux.c:
  - Configure Pin4 and Pin15 for GPIO Input and setup peripheral clock.
  - This ensures that buttons are active



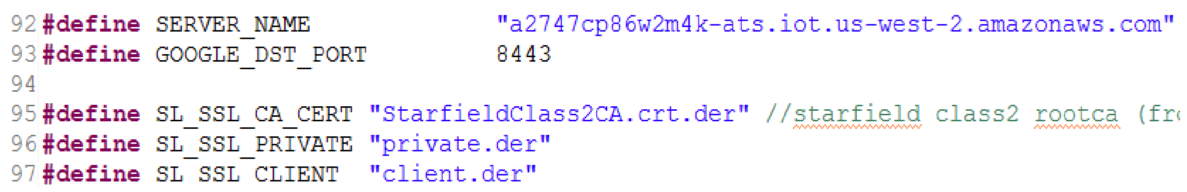
```
    //
    // Configure PIN_04 for GPIO Input
    //
    PinTypeGPIO(PIN_04, PIN_MODE_0, false);
    GPIODirModeSet(GPIOA1_BASE, 0x20, GPIO_DIR_MODE_IN);


    //
        // Configure PIN_15 for GPIO Input
        //
    PinTypeGPIO(PIN_15, PIN_MODE_0, false);
    GPIODirModeSet(GPIOA2_BASE, 0x40, GPIO_DIR_MODE_IN);
```

- Main.c:
    - Add #include hw_memmap.h so that the GPIO inputs are recognized.
    - Change #define SERVER_NAME to the endpoint given in AWS
    - Change #define SL_SSL_CA_CERT so that CC3200 recognizes code and AWS can be accessed.
    - Change #define DATE to the current date so that timing is accurate between device and AWS
    - Change #define POST HEADER for communication with device shadow

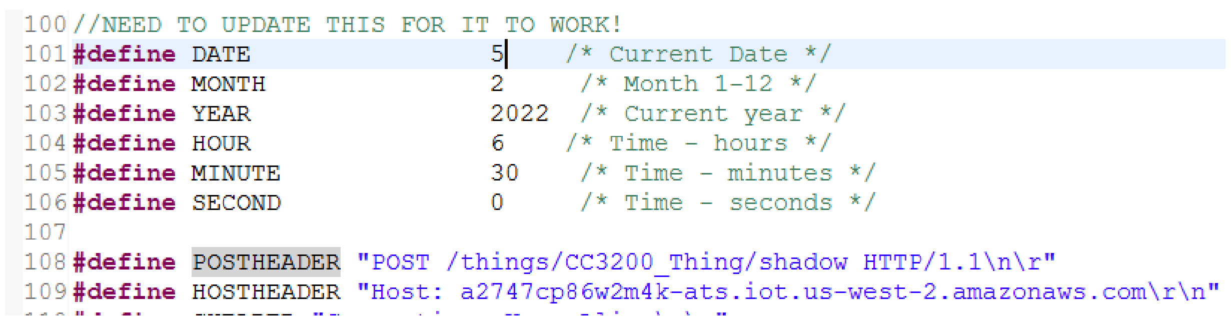


```
#include "gpio.h" //Added in for buttons
#include "hw_memmap.h" //Added in for buttons
#include "hw_common_reg.h" //Added in for buttons
```



```
#define SERVER_NAME                "a2747cp86w2m4k-ats.iot.us-west-2.amazonaws.com"
#define GOOGLE_DST_PORT            8443

#define SL_SSL_CA_CERT "StarfieldClass2CA.crt.der" //starfield class2 rootca (fr
#define SL_SSL_PRIVATE "private.der"
#define SL_SSL_CLIENT  "client.der"
```

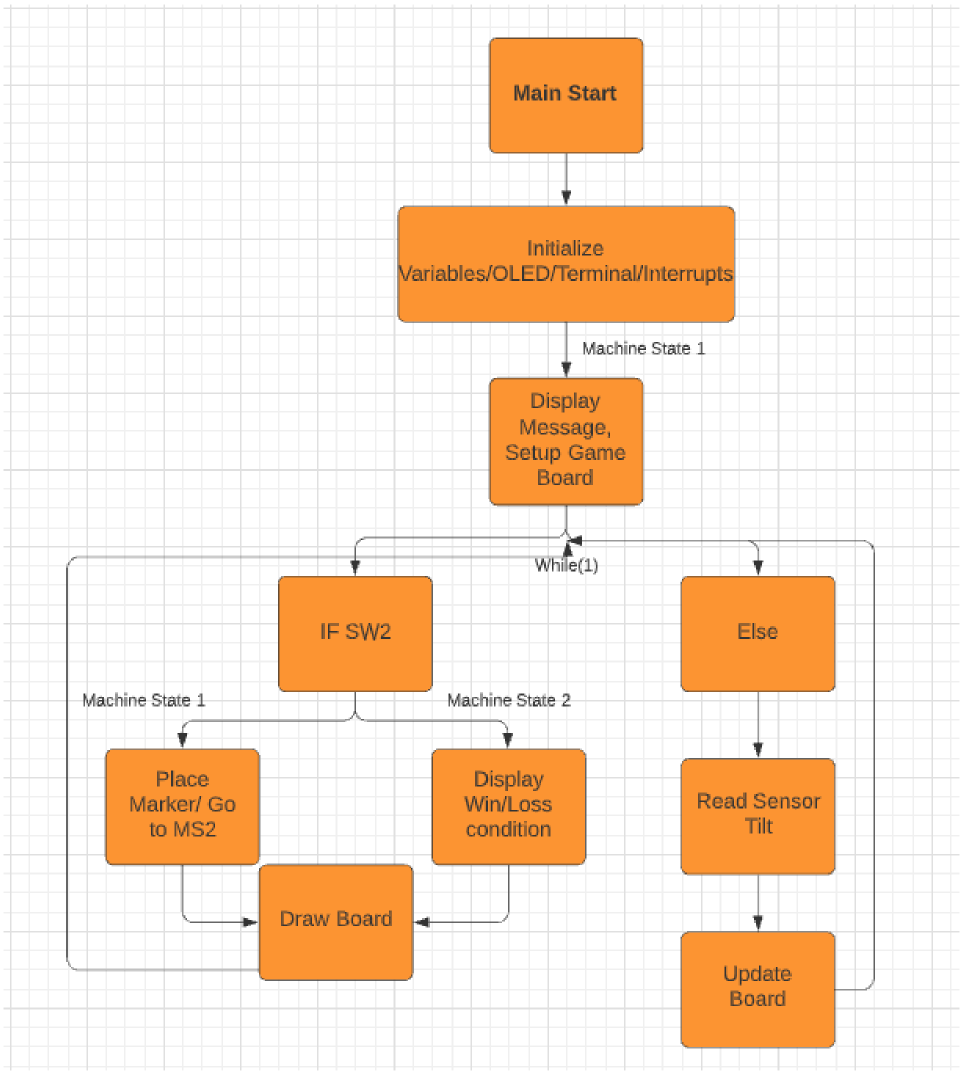


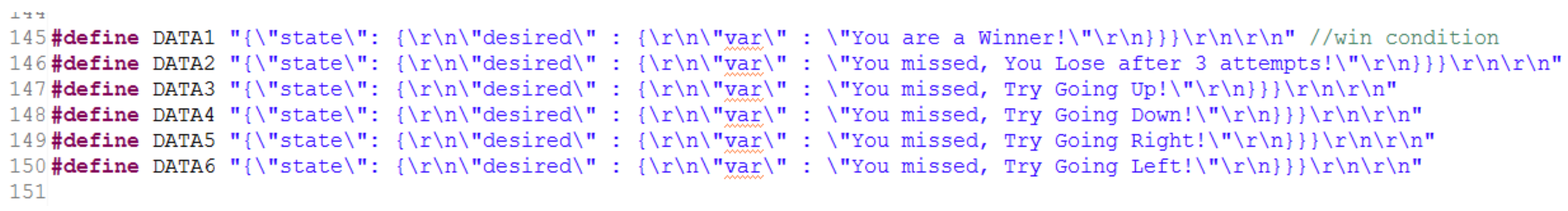
```
//NEED TO UPDATE THIS FOR IT TO WORK!
#define DATE                5     /* Current Date */
#define MONTH               2      /* Month 1-12 */
#define YEAR                2022   /* Current year */
#define HOUR                6     /* Time - hours */
#define MINUTE              30     /* Time - minutes */
#define SECOND              0      /* Time - seconds */

#define POSTHEADER "POST /things/CC3200_Thing/shadow HTTP/1.1\n\r"
#define HOSTHEADER "Host: a2747cp86w2m4k-ats.iot.us-west-2.amazonaws.com\r\n"
```

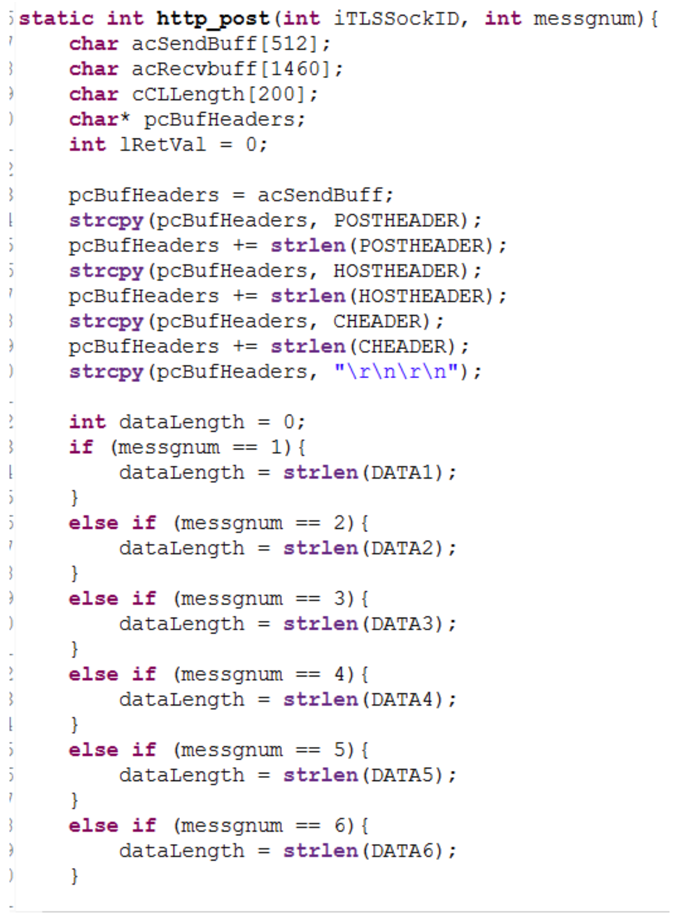
Top Level Description of CODE

Base Top Level Scheme:





```
#define DATA1 "{\"state\": {\r\n\"desired\" : {\r\n\"var\" : \"You are a Winner!\"\r\n}}}\r\n\r\n" //win condition
#define DATA2 "{\"state\": {\r\n\"desired\" : {\r\n\"var\" : \"You missed, You Lose after 3 attempts!\"\r\n}}}\r\n\r\n"
#define DATA3 "{\"state\": {\r\n\"desired\" : {\r\n\"var\" : \"You missed, Try Going Up!\"\r\n}}}\r\n\r\n"
#define DATA4 "{\"state\": {\r\n\"desired\" : {\r\n\"var\" : \"You missed, Try Going Down!\"\r\n}}}\r\n\r\n"
#define DATA5 "{\"state\": {\r\n\"desired\" : {\r\n\"var\" : \"You missed, Try Going Right!\"\r\n}}}\r\n\r\n"
#define DATA6 "{\"state\": {\r\n\"desired\" : {\r\n\"var\" : \"You missed, Try Going Left!\"\r\n}}}\r\n\r\n"
```



```
static int http_post(int iTLSSockID, int messgnum){
    char acSendBuff[512];
    char acRecvbuff[1460];
    char cCLLength[200];
    char* pcBufHeaders;
    int lRetVal = 0;

    pcBufHeaders = acSendBuff;
    strcpy(pcBufHeaders, POSTHEADER);
    pcBufHeaders += strlen(POSTHEADER);
    strcpy(pcBufHeaders, HOSTHEADER);
    pcBufHeaders += strlen(HOSTHEADER);
    strcpy(pcBufHeaders, CHEADER);
    pcBufHeaders += strlen(CHEADER);
    strcpy(pcBufHeaders, "\r\n\r\n");

    int dataLength = 0;
    if (messgnum == 1){
        dataLength = strlen(DATA1);
    }
    else if (messgnum == 2){
        dataLength = strlen(DATA2);
    }
    else if (messgnum == 3){
        dataLength = strlen(DATA3);
    }
    else if (messgnum == 4){
        dataLength = strlen(DATA4);
    }
    else if (messgnum == 5){
        dataLength = strlen(DATA5);
    }
    else if (messgnum == 6){
        dataLength = strlen(DATA6);
    }
```

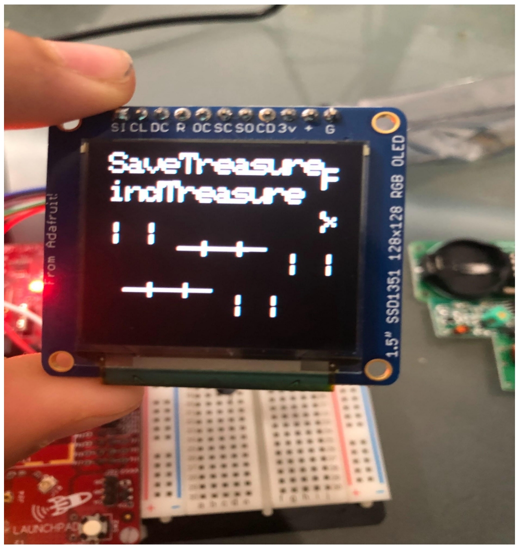
We did make changes to http_post. By adding different loss conditions depending on player 2's choice, we needed to add other DATA definitions to print to the endpoint. In total, we have 6 unique messages.

Discussion/Conclusion

Challenges

Our first challenge was using the files we already had from lab 2 to work on another project. There were posts on slack to get lab 2 code to work on a lab 5 folder, but we found it more time-efficient to recreate the certificates and keys and redo lab 2 setups. An additional benefit to this was a more solid understanding of the processes to establish a connection.

Another challenge was getting the terminal to display the same output on the OLED, but on our first attempt, the terminal appeared to display the grid correctly while the OLED was completely wrong. So our solution to the problem was to build the grid and place the "X" mark through different codes. Here is our old attempt's result.



One of our difficulties that carried over from lab 4. We originally planned to use the remote to navigate on the 3x3 grid. But given the challenges encountered in that approach, we decided to use the BMA222 accelerometer to navigate the grid.

Possible Improvements or Different Projects (within reach)

As stated in the previous section, one major improvement would be to have navigation done with the remote so that all previous labs would be thoroughly reviewed. Another improvement would be to control through AWS using a lambda function. There is a report written by Tina Sorensen published on slack that outlines the process for future labs. Another improvement would be to have custom grid choices. And the option to bury multiple treasures.

We have considered alternative project ideas as well. I ran and tested the getweather file available in the “example” folder. I was planning on receiving data from NASA API about the meteors and asteroids and displaying the information on the OLED. The problem was that getweather was not working for us even after following all the directions on the README file.

Another idea was having my personal FS1000A receiver and transmitter facilitate communication between CC3200 and an Arduino. The Arduino would connect to a raindrop module that I also personally own. Then, if the module detects water, the CC3200 would connect to openweather.gov and confirm whether it was raining or not at Davis.