Lab 1 README

ECE 13

Sean Manger

Prof: Steve McGuire

F24

Author: <Sean Manger> (smanger@ucsc.edu)

Collaborators: I did not collaborate with anyone for this Lab. All code is my own original work. I did however discuss the lab with / assist Donny Tang & Nic Jorgenson.

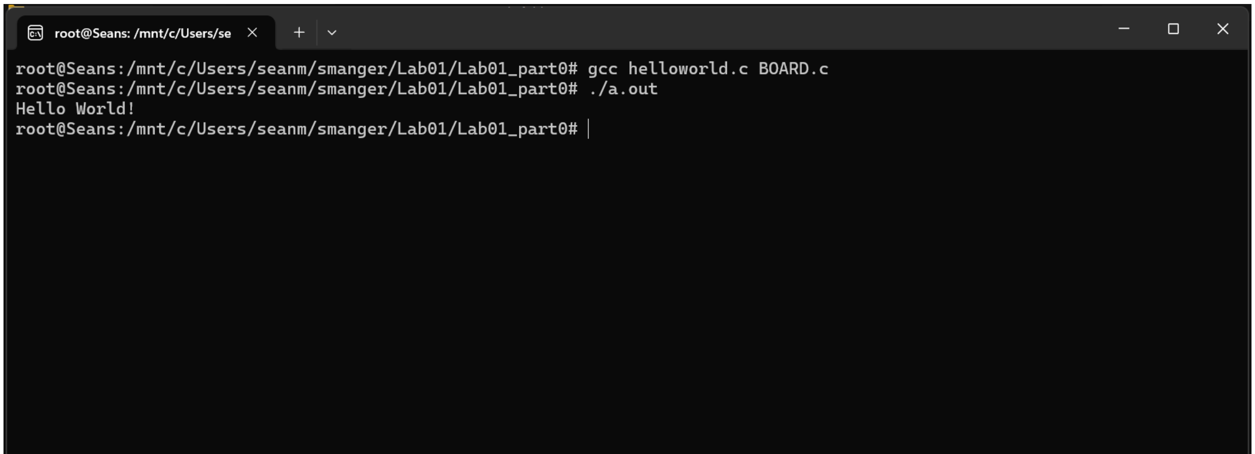
Introduction:

The goal of this lab was to be a gentle introduction to ECE 13 and allow us to set up our development environment which we will use for the rest of the quarter. We looked to get comfortable with using the command line terminal. We also learned about version control using our online GIT Repository.

Part 0: Hello World program on a Terminal & Embedded:

For Part I, we were asked to run a simple "Hello World!" program using the provided code on both the terminal & in an embedded environment on our Nucleo board. Here are my results:

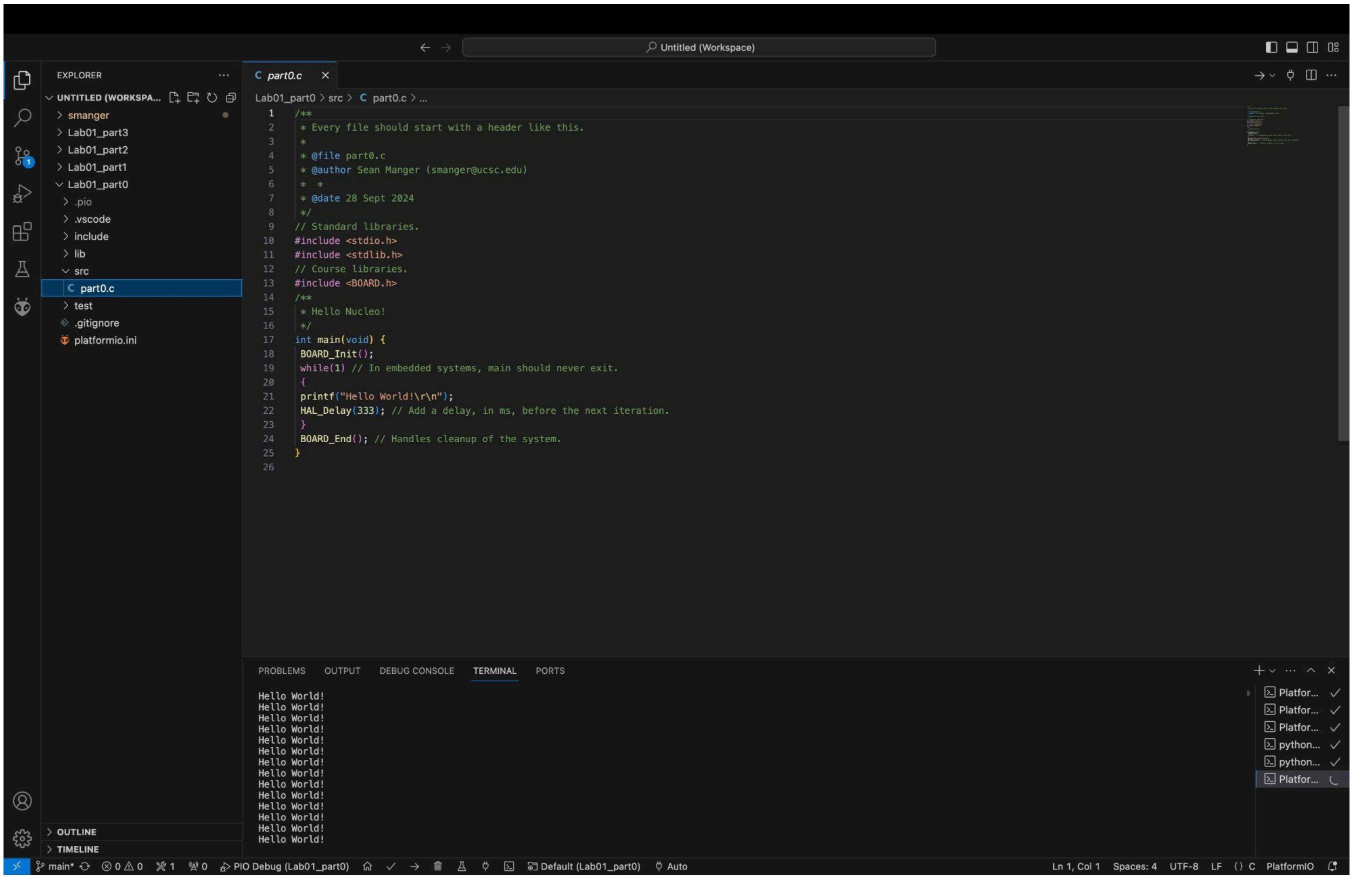
Terminal:



```
root@Seans:/mnt/c/Users/seanm/smanger/Lab01/Lab01_part0# gcc helloworld.c BOARD.c
root@Seans:/mnt/c/Users/seanm/smanger/Lab01/Lab01_part0# ./a.out
Hello World!
root@Seans:/mnt/c/Users/seanm/smanger/Lab01/Lab01_part0# |
```

- Hello World! running on Windows terminal using gcc.

Embedded:



```c
/**
 * Every file should start with a header like this.
 *
 * @file part0.c
 * @author Sean Manger (smanger@ucsc.edu)
 *  *
 * @date 28 Sept 2024
 */
// Standard libraries.
#include <stdio.h>
#include <stdlib.h>
// Course libraries.
#include <BOARD.h>
/**
 * Hello Nucleo!
 */
int main(void) {
 BOARD_Init();
 while(1) // In embedded systems, main should never exit.
 {
 printf("Hello World!\r\n");
 HAL_Delay(333); // Add a delay, in ms, before the next iteration.
 }
 BOARD_End(); // Handles cleanup of the system.
}
```

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

- Hello World! running on ST Nucleo 411FE Board

Part I: Using a Debugger & Sorting an Array:

For Part 1, following the provided directions I set a breakpoint and stepped through the array and recorded as it was sorted and changed. I recorded what I saw under "Part 1 Values". After this, the debugger continued to the final print statement after the for loop and the final value of the sorted array was displayed. Which I also recorded down below.

Part 1 Values (Using Debugger):

[45,207,70,41,4] // i = 0

[45,207,70,41,4] // i = 1

[45,207,70,41,4] // i = 2

[45,70,207,41,4] // i = 3

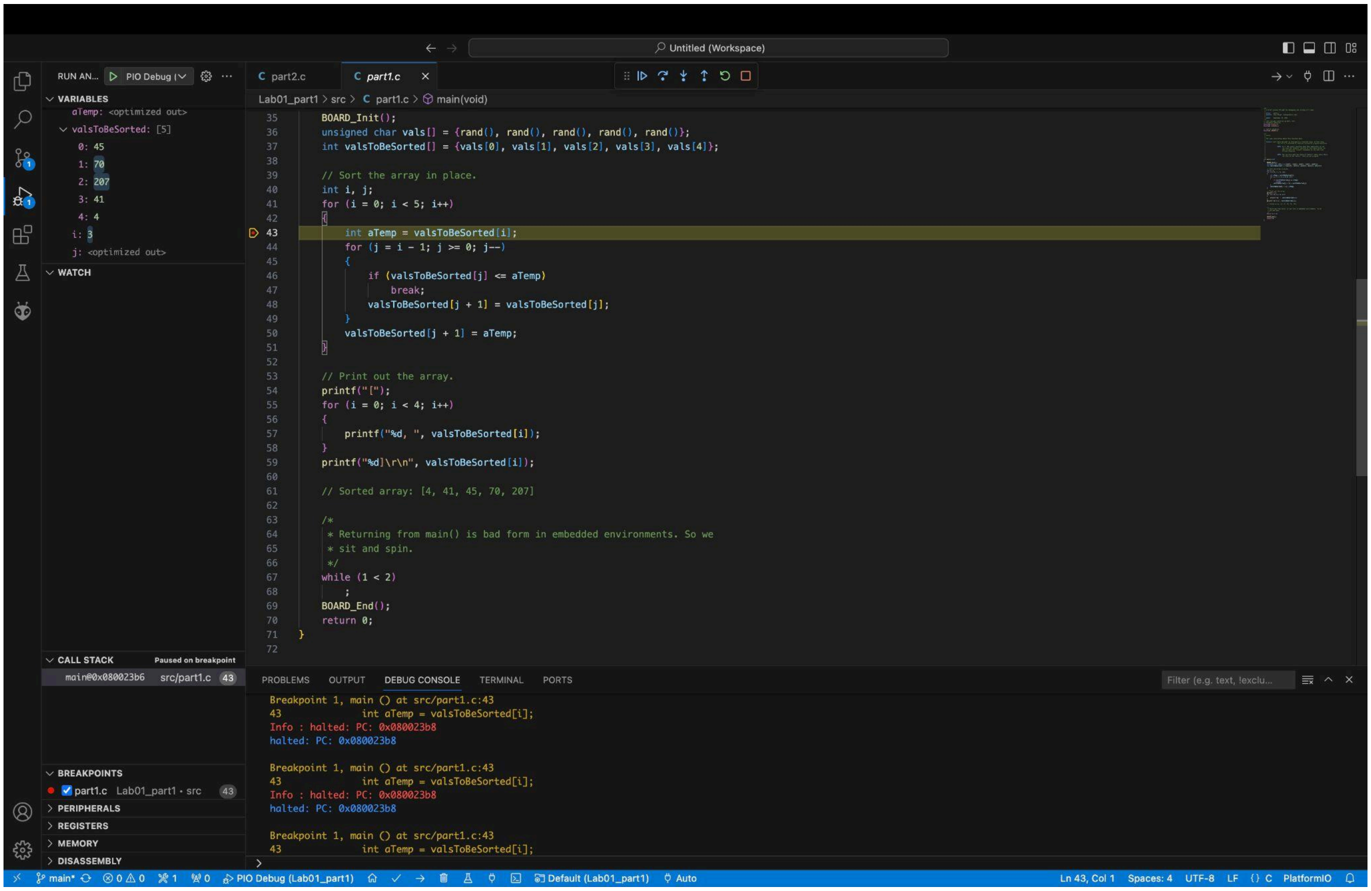
[41,45,70,207,4] // i = 4

Final Result (After exiting Breakpoint):

[4, 41, 45, 70, 207]

I also took the following two screenshots for Part 1 to further verify functionality:

Using Debugger:



```
aTemp: <optimized out>
valsToBeSorted: [5]
  0: 45
  1: 70
  2: 207
  3: 41
  4: 4
i: 3
j: <optimized out>
```

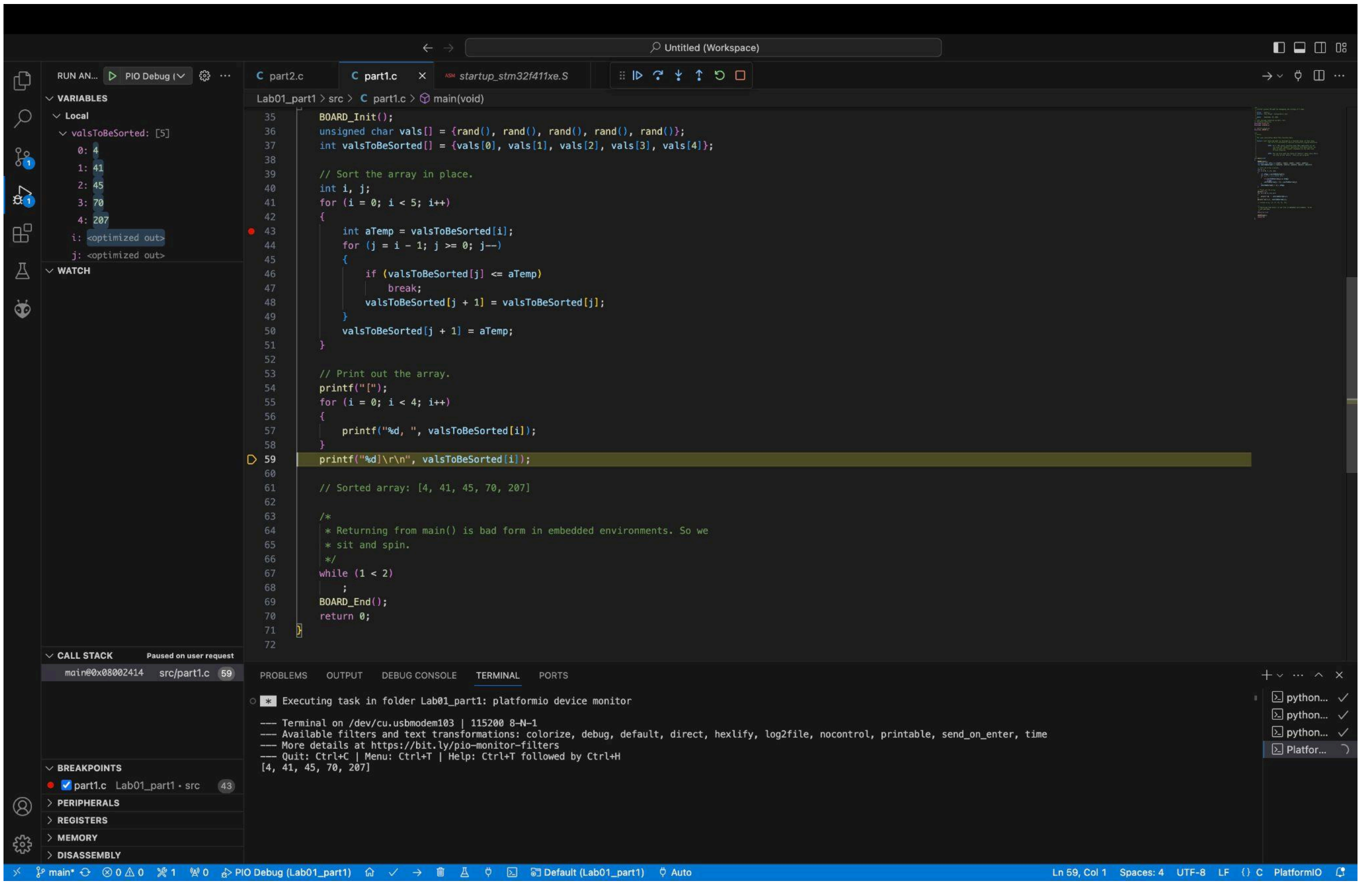
```
main@0x080023b6   src/part1.c 43
```

```
☑ part1.c  Lab01_part1 • src  43
```

```c
    BOARD_Init();
    unsigned char vals[] = {rand(), rand(), rand(), rand(), rand()};
    int valsToBeSorted[] = {vals[0], vals[1], vals[2], vals[3], vals[4]};

    // Sort the array in place.
    int i, j;
    for (i = 0; i < 5; i++)
    {
        int aTemp = valsToBeSorted[i];
        for (j = i - 1; j >= 0; j--)
        {
            if (valsToBeSorted[j] <= aTemp)
                break;
            valsToBeSorted[j + 1] = valsToBeSorted[j];
        }
        valsToBeSorted[j + 1] = aTemp;
    }

    // Print out the array.
    printf("[");
    for (i = 0; i < 4; i++)
    {
        printf("%d, ", valsToBeSorted[i]);
    }
    printf("%d]\r\n", valsToBeSorted[i]);

    // Sorted array: [4, 41, 45, 70, 207]

    /*
     * Returning from main() is bad form in embedded environments. So we
     * sit and spin.
     */
    while (1 < 2)
        ;
    BOARD_End();
    return 0;
}
```

```
Breakpoint 1, main () at src/part1.c:43
43              int aTemp = valsToBeSorted[i];
Info : halted: PC: 0x080023b8
halted: PC: 0x080023b8

Breakpoint 1, main () at src/part1.c:43
43              int aTemp = valsToBeSorted[i];
Info : halted: PC: 0x080023b8
halted: PC: 0x080023b8

Breakpoint 1, main () at src/part1.c:43
43              int aTemp = valsToBeSorted[i];
```

Sorted Array Final Result:



VARIABLES
Local
valsToBeSorted: [5]
0: 4
1: 41
2: 45
3: 70
4: 207
i: <optimized out>
j: <optimized out>
WATCH

CALL STACK Paused on user request
main@0x08002414 src/part1.c 59

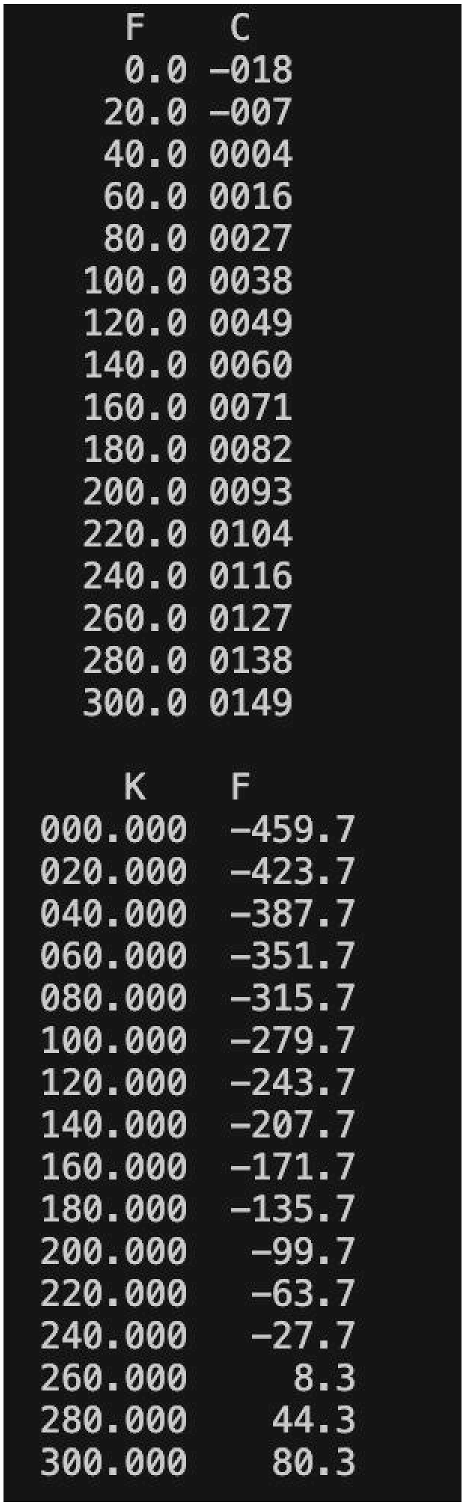
BREAKPOINTS
☑ part1.c Lab01_part1 • src 43
PERIPHERALS
REGISTERS
MEMORY
DISASSEMBLY

```c
    BOARD_Init();
    unsigned char vals[] = {rand(), rand(), rand(), rand(), rand()};
    int valsToBeSorted[] = {vals[0], vals[1], vals[2], vals[3], vals[4]};

    // Sort the array in place.
    int i, j;
    for (i = 0; i < 5; i++)
    {
        int aTemp = valsToBeSorted[i];
        for (j = i - 1; j >= 0; j--)
        {
            if (valsToBeSorted[j] <= aTemp)
                break;
            valsToBeSorted[j + 1] = valsToBeSorted[j];
        }
        valsToBeSorted[j + 1] = aTemp;
    }

    // Print out the array.
    printf("[");
    for (i = 0; i < 4; i++)
    {
        printf("%d, ", valsToBeSorted[i]);
    }
    printf("%d]\r\n", valsToBeSorted[i]);

    // Sorted array: [4, 41, 45, 70, 207]

    /*
     * Returning from main() is bad form in embedded environments. So we
     * sit and spin.
     */
    while (1 < 2)
        ;
    BOARD_End();
    return 0;
}
```

```
 *  Executing task in folder Lab01_part1: platformio device monitor

--- Terminal on /dev/cu.usbmodem103 | 115200 8-N-1
--- Available filters and text transformations: colorize, debug, default, direct, hexlify, log2file, nocontrol, printable, send_on_enter, time
--- More details at https://bit.ly/pio-monitor-filters
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H
[4, 41, 45, 70, 207]
```

Part 2: Temperature Conversion Tables:

For Part 2, I followed the provided instructions and wrote the desired code for both the Farenheit to Celcius Table & Kelvin to Fahrenheit Table. I also set the desired character widths and precisions for each table to what was specified in the lab document. Here is a picture of my results which match what was expected in the lab doc:



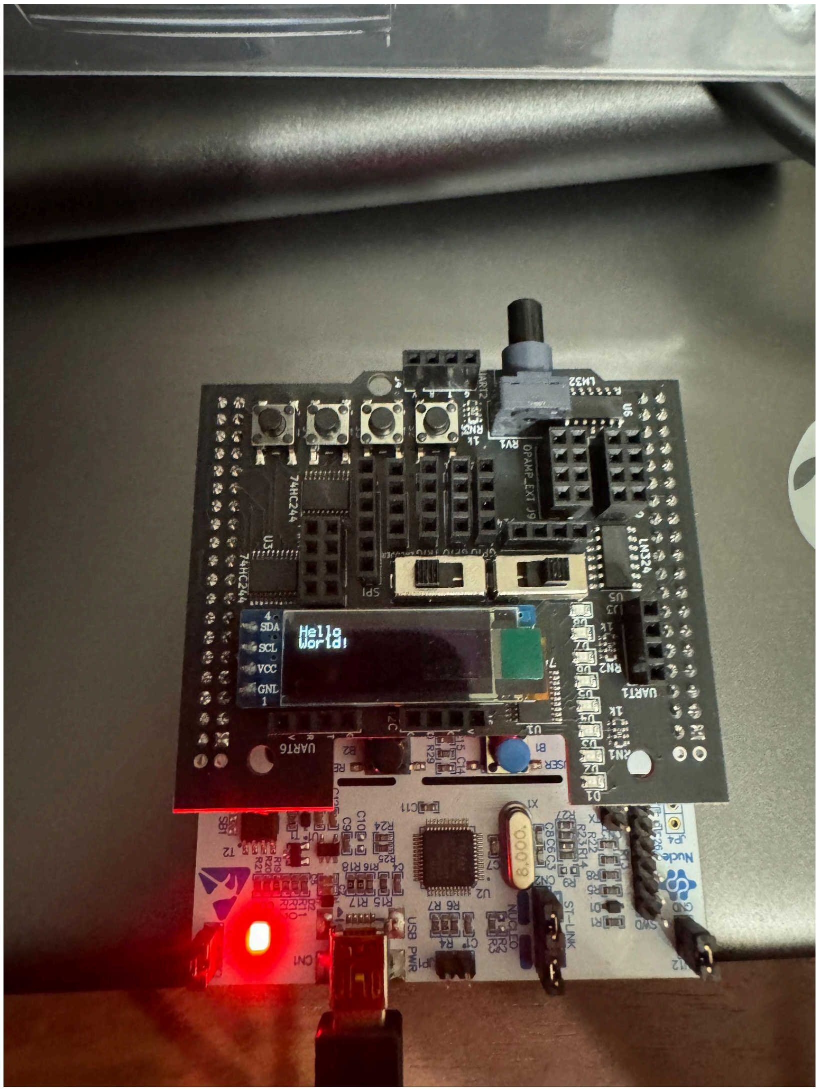
```
    F    C
    0.0 -018
   20.0 -007
   40.0 0004
   60.0 0016
   80.0 0027
  100.0 0038
  120.0 0049
  140.0 0060
  160.0 0071
  180.0 0082
  200.0 0093
  220.0 0104
  240.0 0116
  260.0 0127
  280.0 0138
  300.0 0149

    K    F
000.000  -459.7
020.000  -423.7
040.000  -387.7
060.000  -351.7
080.000  -315.7
100.000  -279.7
120.000  -243.7
140.000  -207.7
160.000  -171.7
180.000  -135.7
200.000   -99.7
220.000   -63.7
240.000   -27.7
260.000     8.3
280.000    44.3
300.000    80.3
```

- Part 2 Temperature Conversion Tables

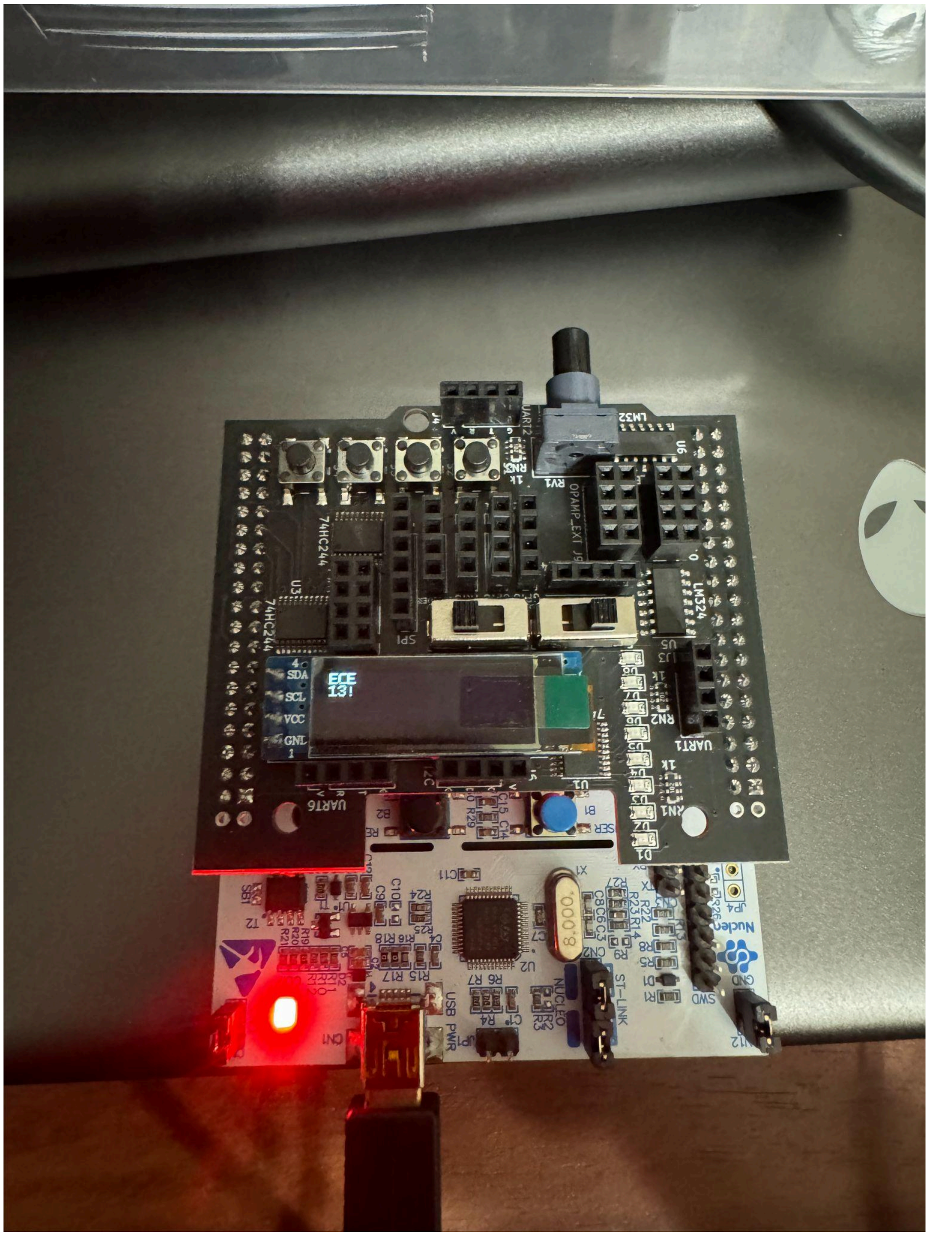
-

Part 3: OLED Screen Extra Credit:

For Part 3 we were asked to use the oled screen on our Nucleo board to display static text reading "Hello World!". However, I thought that this was rather uninteresting. So, I modified the code to first display "Hello World!" for 3 seconds followed by "ECE 13!" for another 3 seconds before returning to the initial "Hello World!" repeatedly in a loop. Here are pictures of my results:



- Hello World! displayed on the OLED screen.



- Followed by ECE 13 in a repeating loop.

This concluded Lab 1.

<u>Personal Comments & Observations:</u>

Overall, I thought this was an enjoyable lab. It served as a good introduction to the course and the development environment we will be using throughout. I thought the hardest part of this lab was just getting everything setup and working properly. One problem I encountered was installing the proper drivers on Windows. However, I thankfully found the files on STM's website. Besides that, the actual coding part of the lab was pretty straightforward. I also thought that the lab document was fairly easy to follow.