CPE403/ECG603/ECG710 – Advanced Embedded Systems/Real-Time Embedded Systems

Design Assignment 4

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Joshua Martinez

Email: marti87@unlv.nevada.edu

Github Repository link (root): https://github.com/JoshuaMa2003/submission_da_0011

Youtube Playlist link (root):
https://www.youtube.com/playlist?list=PLZ09MA_uFt61Vw2JCAtw-_8vAdmoCKXpZ

Follow the submission guideline to be awarded points for this Assignment.

Submit the following for all Assignments:

1. In the document, for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include comments. If no base code is provided, submit the base code for the first task only.
2. Create a private Github repository with a random name (no CPE403/603/710, Lastname, Firstname). Place all labs under the root folder MSP432E4/CC1352, sub-folder named Assignment1, with one document and one video link file for each lab, place modified c files named as asng_taskxx.c.
3. If multiple c files or other libraries are used, create a folder asng1_t01 and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) with startup_ccs.c and other include files, c) text file with youtube video links (see template).
5. Submit the doc file in canvas before the due date. The root folder of the github assignment directory should have the documentation and the text file with youtube video links.
6. Organize your videos as a playlist under the name “ADVEMBSYS”. The playlist should

have the video sequence arranged as submission or due dates.

7. Only submit the PDF. Upload this document in the github repository and in canvas.

1. **Code for tasks: For each task, submit the relevant modified/section or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the initialization and execution section of each task separately. Use a separate page for each task.**

Task 1 – Base Project Zero Application (Unmodified Reference)

- **Original Behavior:**
  - **The original *project_zero.c* comes from TI's Project Zero example.**
  - **It already:**
    - **Initializes GAP, GATT, Device Information Service.**
    - **Adds three Bluetooth services:**
      1. **LED Service**
      2. **Button Service**
      3. **Data Service**
    - **Registers callbacks for:**
      1. **LED Writes**
      2. **Button notifications config**
      3. **Data Services writes / CCCD**
    - **Handles buttons presses locally by updating the Button Service characteristic in *ProjectZero_handleButtonPress()* and letting the BLE stack send notifications.**
- **No ADC or joystick code exists in the original file; all hardware interactions are just two LEDs and two push-buttons.**

- **Unchanged Base Code:**
  - **Service registrations in ProjectZero_init:**

```
LedService_AddService(selfEntity);
ButtonService_AddService(selfEntity);
DataService_AddService(selfEntity);
```

  - **Callback Registration:**

```
LedService_RegisterAppCBs(&ProjectZero_LED_ServiceCBs);
ButtonService_RegisterAppCBs(&ProjectZero_Button_ServiceCBs);
DataService_RegisterAppCBs(&ProjectZero_Data_ServiceCBs);
```

Task 2 – Add an ADC-based Joystick Input

2.1 SysConfig / Hardware Changes

- **In SysConfig → ADC, I enabled a single ADC instance:**
  - **Name:** Config_ADC_0
  - **Reference Source:** Fixed
  - **Reference Voltage:** 3300000 uV
  - **Sampling Duration:** 2.7 uS
  - **ADC Peripheral:** Any (ADC0)
  - **ADC Pin:** Any (DI023/2 Header)
- **In TIDrivers section, I made sure the ADC Driver was enabled**

2.2 Code Changes: Initialize and Read ADC

1. **Included the ADC driver header at the top with the other TIDriver Includes:**

```
#include <ti/drivers/ADC.h>
#include "ti_drivers_config.h"
```

2. **Added global variables for the joystick ADC:**

```
static ADC_Handle   adcJoystickHandle;
static ADC_Params   adcJoystickParams;
static uint16_t     adcJoystickRaw = 0;
```

3. **In ProjectZero_init(), after the button GPIO initialization, I opened the ADC:**

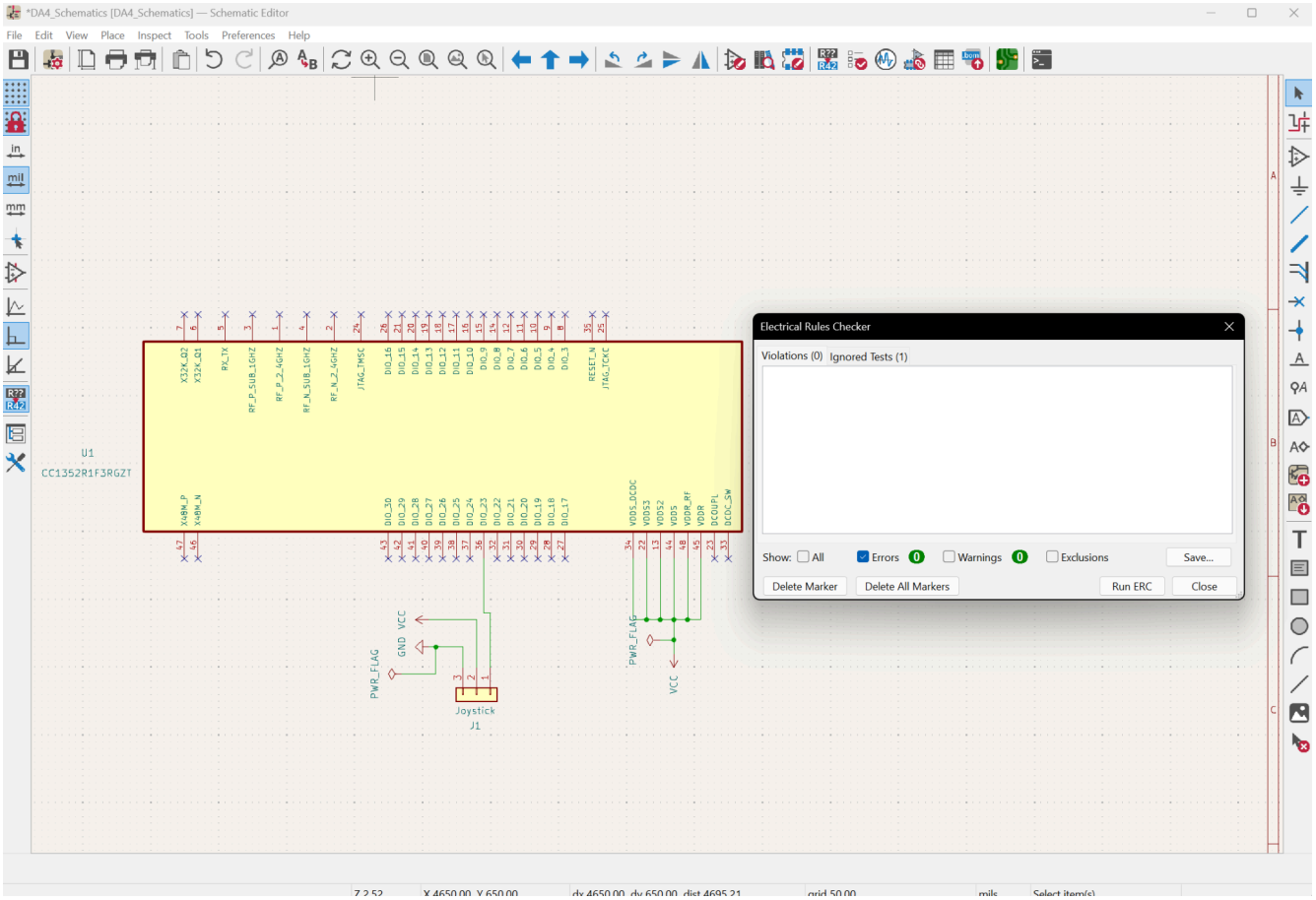
```
// --- Joystick ADC initialization ---
ADC_init();
ADC_Params_init(&adcJoystickParams);
adcJoystickParams.isProtected = false;
adcJoystickHandle = ADC_open(CONFIG_ADC_0, &adcJoystickParams);

if (adcJoystickHandle == NULL)
{
    Log_error0("Error initializing joystick ADC (CONFIG_ADC_0)");
}
else
{
    Log_info0("Joystick ADC initialized on CONFIG_ADC_0");
}
```

4. **Created a small helper function to read the joystick:**

```
static void ProjectZero_readJoystick(void)
{
    int_fast16_t res;

    if (adcJoystickHandle == NULL)
    {
        return;
    }

    res = ADC_convert(adcJoystickHandle, &adcJoystickRaw);

    if (res == ADC_STATUS_SUCCESS)
    {
        // adcJoystickRaw now holds a 12-bit value 0–4095
        Log_info1("Joystick ADC raw = %d", adcJoystickRaw);
    }
    else
    {
        Log_error1("ADC_convert failed (%d)", res);
    }
}
```

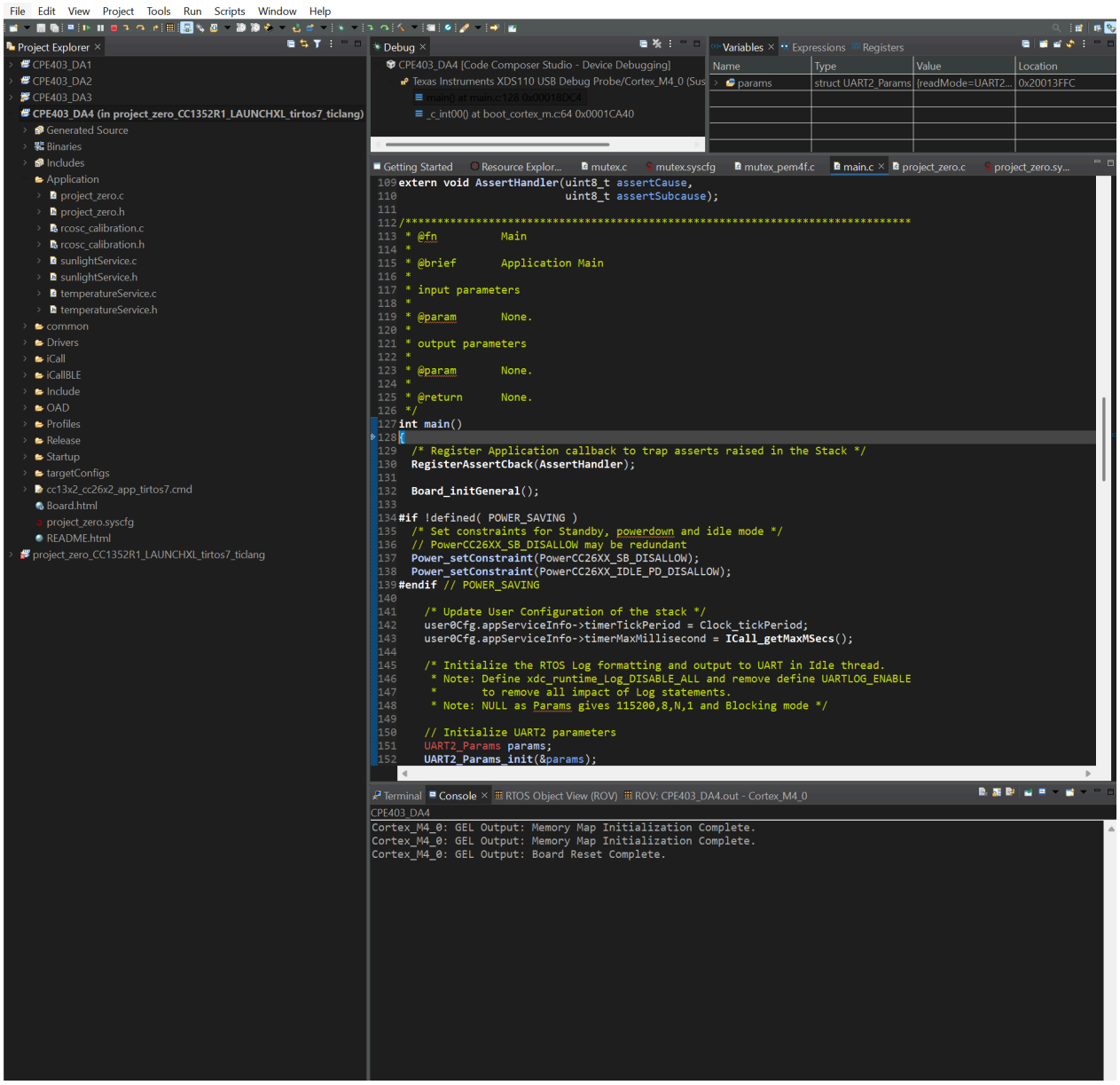
2. Block diagram and/or Schematics showing the components, pins used, and interface. You can use KiCAD/Eagle/Altium to get the schematics. KiCAD Symbol libraries for TI uCs are @ https://kicad.github.io/symbols/MCU_Texas.html and https://www.snapeda.com/parts/CC1352P1F3RGZT/Texas%20Instruments/view-part/

KiCad Design Schematics (ERC Compliant):



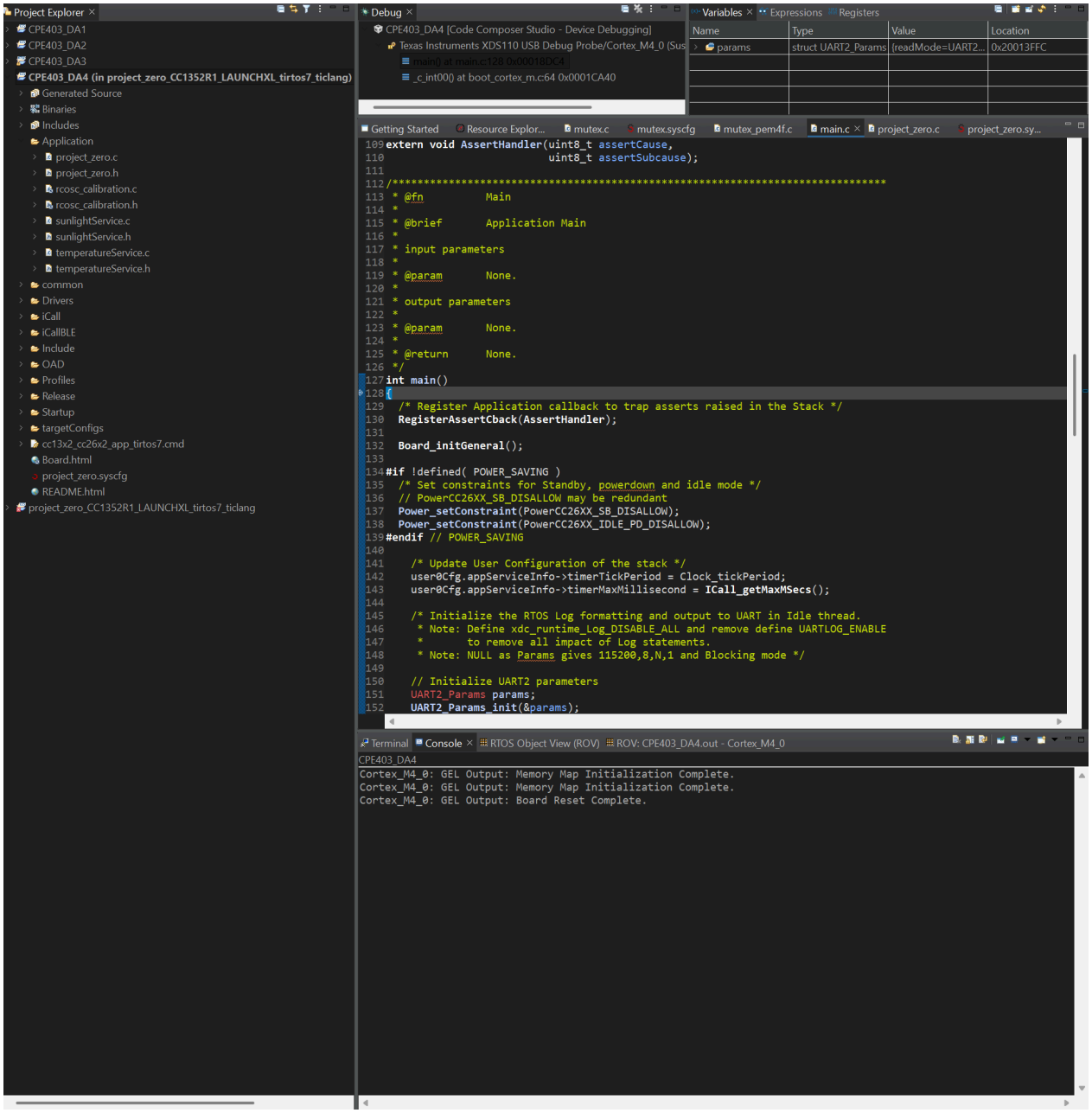
3. Screenshots of the IDE, physical setup, debugging process – Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.

IDE Workspace:

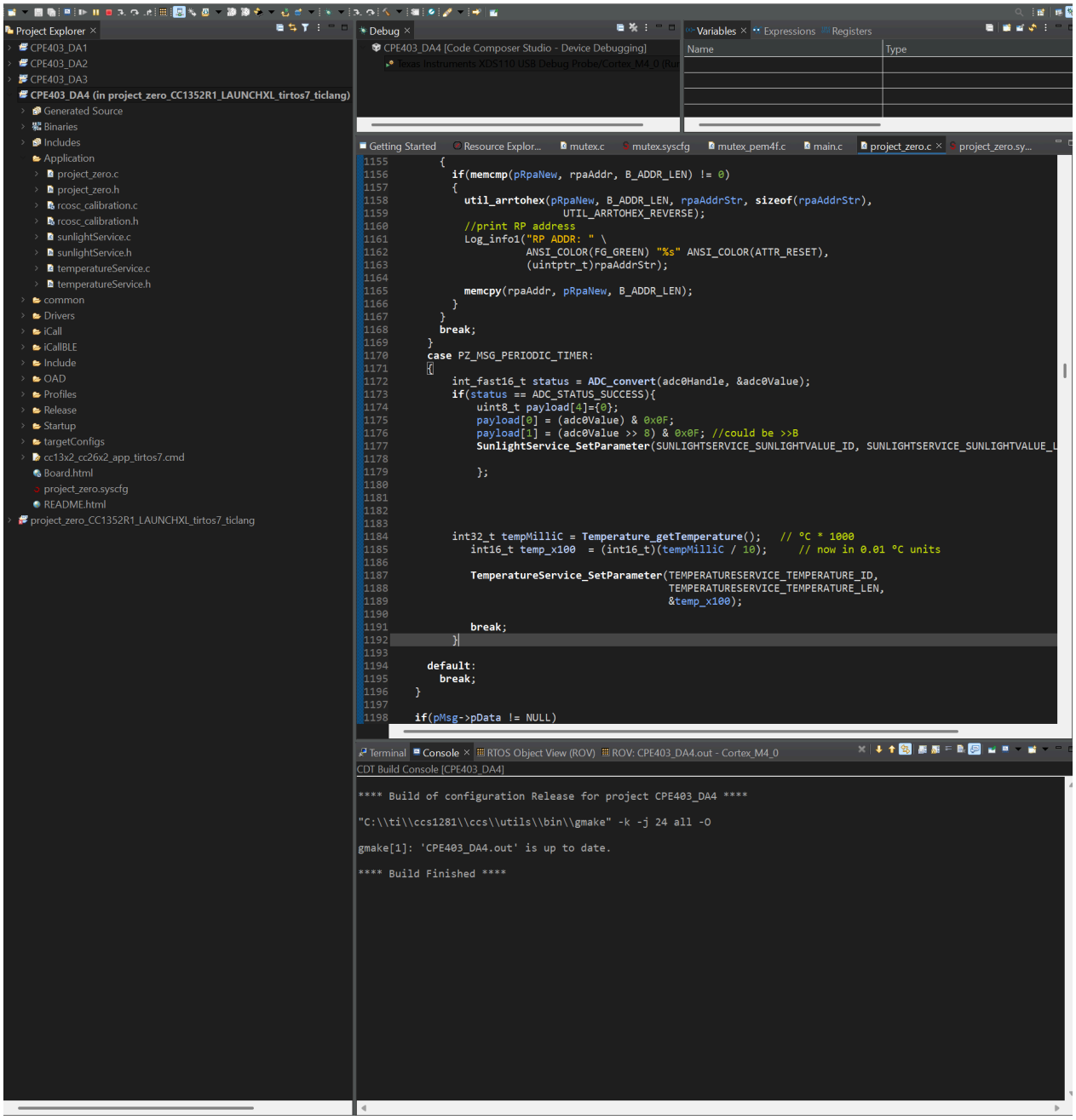


Physical Setup:

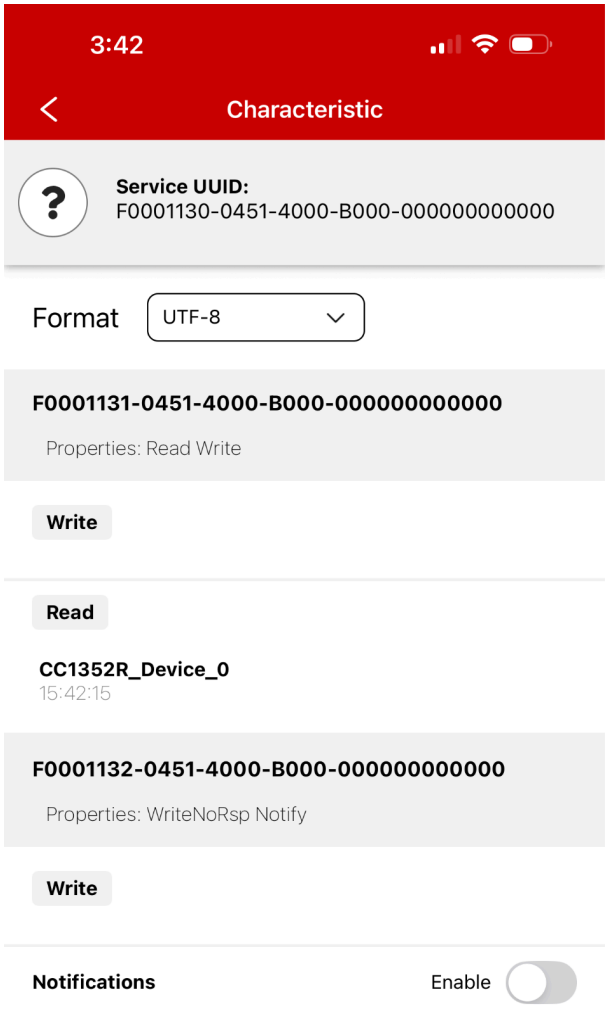
Debugging Process:



Successful Compilation:



Fixed Identification Message:



4. Declaration

I understand the Student Academic Misconduct Policy - http://studentconduct.unlv.edu/misconduct/policy.html

"This assignment submission is my own, original work".
Joshua Martinez