Fuel Cell Monitor

Rana Kortam

Russell Wells

Sameer Osama

Jessica Odutola

**SUBSYSTEM REPORT**

REVISION – Original

04 December, 2022

Subsystem Reports

for

Fuel Cell Monitor

Prepared by:

Fuel Cell Team\_\_\_\_\_\_\_\_12/04/2022

Author Date

Approved by:

Russell Wells\_\_\_\_\_\_\_\_ \_12/04/2022

Project Leader Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

John Lusher, P.E. Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dalton Cyr Date

**Change Record**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rev.** | **Date** | **Originator** | **Approvals** | **Description** |
| **1** | 12/04/2022 | Fuel Cell Monitor Team |  | Original |

Table of Contents

[Table of Contents III](#_Toc441585281)

[List of Tables IV](#_Toc441585282)

[List of Figures V](#_Toc441585283)

[1. Overview 1](#_Toc441585284)

[2. Power Subsystem Report 1](#_Toc441585285)

[2.1 Subsystem Introduction 1](#_Toc441585286)

[2.2 Subsystem Details 1](#_Toc441585287)

[2.3 Subsystem Validation](#_Toc441585287) 2

[2.4 Diagnostic](#_Toc441585287) 3

[2.5 Midigation](#_Toc441585287) 4

[2.6 Subsystem Conclusion](#_Toc441585287) 4

[3. Internal Signal and Data Manipulation Subsystem](#_Toc441585289) 5

[3.1 Subsystem Introduction](#_Toc441585290) 5

[3.2 Subsystem Details](#_Toc441585291) 5

[3.3 Subsystem Validation](#_Toc441585293) 6-7

[3.4 Diagnostic and Mitigation](#_Toc441585290) 7-8

[3.5 Subsystem Conclusion](#_Toc441585291) 8

[4. ESP32 and PIC32 Subsystem](#_Toc441585296) 9

[4.1 Subsystem Introduction](#_Toc441585297) 9

[4.2 Subsystem Details](#_Toc441585297) 9

[4.3 Subsystem Validation](#_Toc441585297) 9

[4.4 Diagnostic and Midigation](#_Toc441585297) 13

[4.5 Subsystem Conclusion](#_Toc441585297) 14

[5. Android Application Subsystem](#_Toc441585298) 15

[5.1. Subsystem Introduction](#_Toc441585299) 15

[5.2. App Running on Android Device 1](#_Toc441585300)5

5.3. App Connecting to Database……………………………………………………………16

5.4. Properly Display Accurate Data to Users………………………………………………17

5.5. Alerts to Users Regarding Errors……………………………………………………….18

5.6. Subsystem Diagnostic and Mitigation………………………………………………….19

5.7. Subsystem Conclusion…………………………………………………………………..19

List of Tables

Table 1: Power Subsystem Electronic Loads ……………….……………………2

List of Figures

Figure 1: Power Subsystem Output to Output Voltage, LM2595sx-3.3……………………2

Figure 2: Power Subsystem Output to Output Voltage, LM2595sx-3.3……………………3

Figure 3: Power Subsystem Output to Output Voltage, LM2595s-5.0…………………….3

Figure 4: Power Subsystem DC/DC Converter Noise Measurement LM2595sx-3.3…….4

Figure 5: Power Subsystem DC/DC Converter Noise Measurement LM2595s-5.0……..4

Figure 6: Internal Signal Transfer and Data Manipulation High Level Circuit Schematic..6

Figure 7: ISDMS Oscilloscope Signal Verification………………………………………...…7

Figure 8: Block diagram of the ESP32 and PIC32…………………………………………..9

Figure 9:ESP32 connection to Wi-Fi………………………………………………………..10

Figure 10:ESP32 UART communication……………………………………………………10

Figure 11: ESP32 database connection…………………………………………………….11

Figure 12: UART code for PIC32 …………………………………………………………….11

Figure 13: Array code for PIC32……………………………………………………………..12

Figure 14: code for PIC32…………………………………………………………………….13

Figure 15: Wrong ADC code for PIC32……………………………………………………..14

Figure 16: App running on a hardware device………………………………………………15

Figure 17: Connectivity between the app and database…………………………………..16

Figure 18: Status Graph………………………………………………………………………17

Figure 19: Fuel Cell Table…………………………………………………………………….17

Figure 20: Error Detection…………………………………………………………………….18

# Introduction

# The Fuel Cell Monitor will be created to provide real-time monitoring of every fuel cell in each fuel stack. The system will take voltage readings from the fuel cell and sends the voltages to database, where an android application will be able display the voltages. The system is broken down into the power, internal signal transfer and data manipulation subsystem, PIC32 and ESP32, and the android application subsystem. The following sections outline the fuel cell monitor subsystem’s validations as well as their diagnostic and mitigation as necessary.

# Power Subsystem

## Subsystem Introduction:

The Power Subsystem is designed to provide 3.3 V and 5 V to the components that require it. The 5 V also gets passed over an isolation barrier to provide power to the components on the isolated side of the circuit. It also involves building differential amplifiers.

## Subsystem Details:

The Power Subsystem uses two DC/DC converters (LM2595s-5.0 and LM2595sx-3.3). These converters take an input of 12 VDC from the wall wart and output 5V and 3.3V respectively. The 5V is passed to the input of the isolated DC/DC converter (DCP020505U/1K), which outputs 5V to provide power to the components on the isolated side of the circuit. There is also a voltage reference that goes to the differential amplifiers (LT1990ACS8). The differential amplifiers are used to filter the fuel cell signals and pass those signals to the Internal Signal Transfer and Data Manipulation Subsystem. The high-level schematic for this is shown in Figure 1.

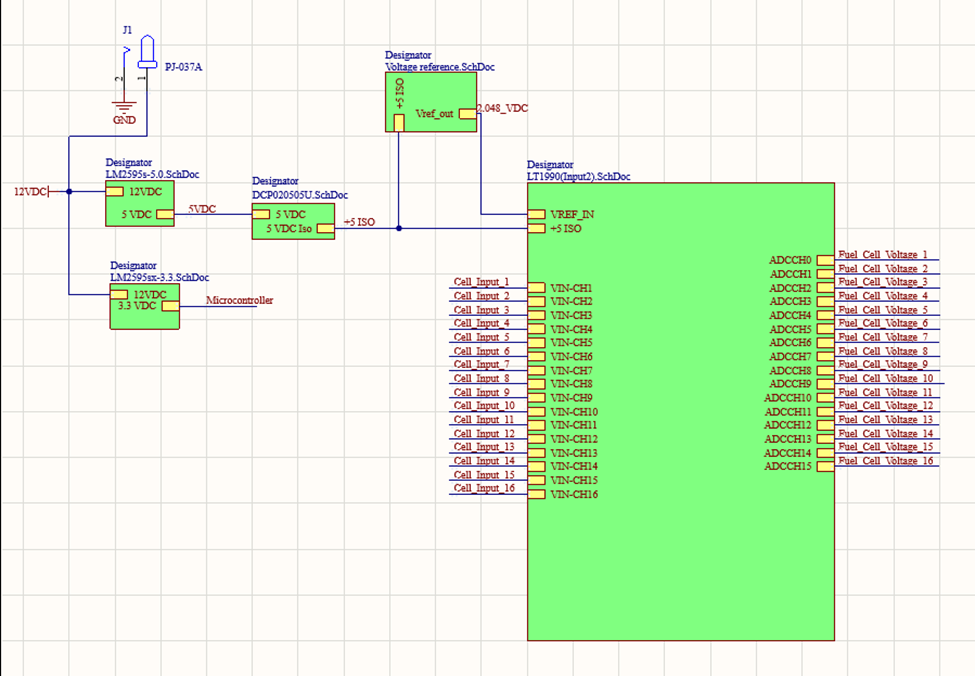


Figure . Power Subsystem High Level Circuit Schematic

## Subsystem Validation:

**2*.3.1. Electronic Load:***

Both DC/DC converters (LM2595s-5.0 and LM2595sx-3.3) were functioning correctly and outputted roughly 5V and 3.3V. The max load current on both converters is 1 A. After applying an electronic load where the current was varied from 0 to 1 A, the output voltage was recorded as seen in Table 1.

Table . Power Subsystem Electronic Loads

|  |  |  |  |
| --- | --- | --- | --- |
| LM2595s-5.0 | | LM2595sx-3.3 | |
| Current (A) | Output Voltage (V) | Current (A) | Output Voltage (V) |
| 0 | 4.983 | 0 | 3.294 |
| .995 | 4.24 | .99 | 3.13 |

**2*.3.2. Varying Input Voltage:***

The input voltage range for the converters is 4.5V – 40V. The output voltage was calculated after varying the input voltage across the input voltage range. The 5 different input voltage values that were tested for were 5V, 12V (nominal voltage), 16V, 20V, and 25V. The highest input voltage applied was 25V despite the maximum input voltage being 40V. This was because the power supply equipment didn’t provide more than 25V to the system. The findings of the output voltage from the converters can be seen in Figure 2 and Figure 3.

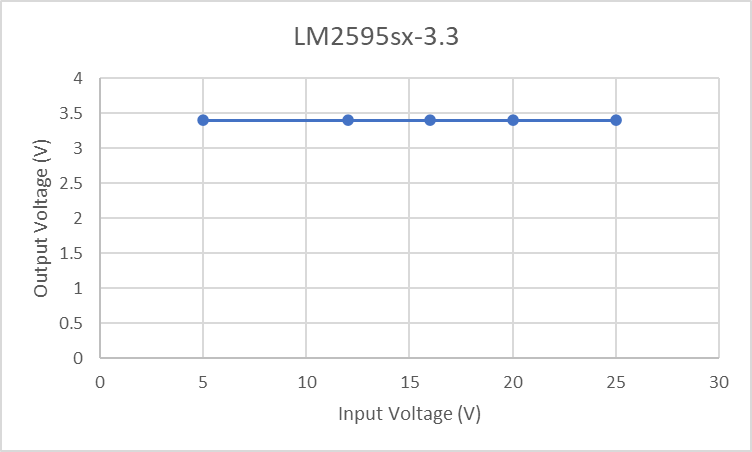


Figure Power Subsystem Output to Output Voltage, LM2595sx-3.3

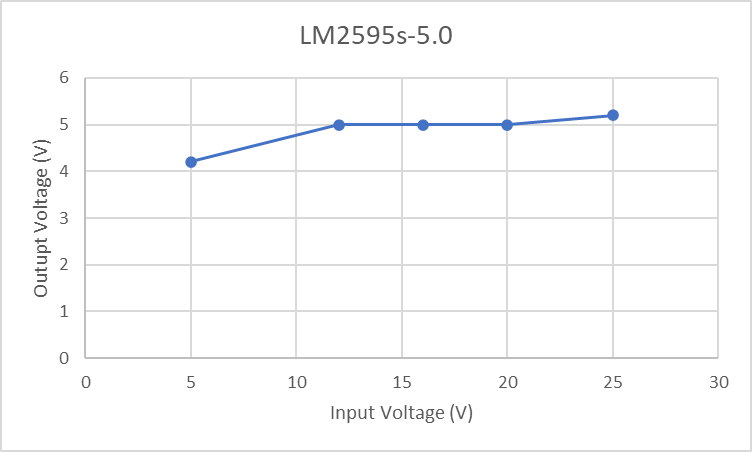


Figure .Power Subsystem Output to Output Voltage, LM2595s-5.0

As seen in Figure 2, the LM2595sx-3.3 has a straight line and the output voltage was at 3.4V for every input voltage. From Figure 3, the output voltage varied somewhat, especially when the input voltage was very low. For an input voltage of 5V, the output voltage was 4.2V. All the other output voltage readings were relatively close to 5V.

**2*.3.3. Noise on Output Voltage:***

The LM2595sx-3.3 had a 71mV ripple voltage on the 3.3V output as shown in Figure 4.

U,{d98fc01c-f93e-446d-839f-ef5572662243}{91},10,6.666666666666667

Figure . Power Subsystem DC/DC Converter Noise Measurement LM2595sx-3.3

The LM2595s-5.0 had a 91mV ripple voltage on the 5V output as shown in Figure 5.

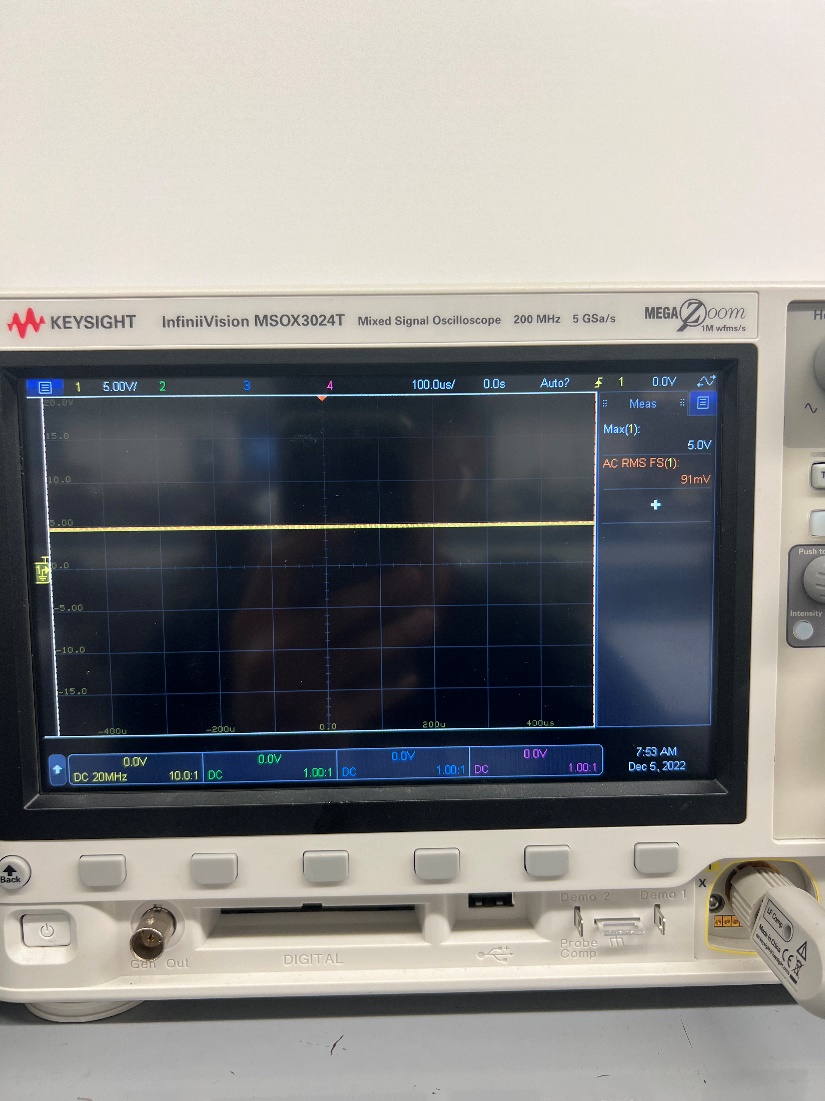


Figure .Power Subsystem DC/DC Converter Noise Measurement LM2595s-5.0

**2*.4. Diagnostic:***

The 5V output from the DC/DC converter goes to the input of the isolated DC/DC converter. The output of the isolated DC/DC converter came out to be 5.3V. After this 5.3V passes through the resistor, it doesn’t give a voltage reading anymore. Due to this reason, the functionality of the voltage reference and the differential amplifiers was not able to be measured since they require the 5V from the isolated DC/DC converter to operate. There could be some issue during the soldering process where a component could have possibly been burnt or was not soldered on properly. It could also be possible that the diode was placed in the wrong direction which is causing the current to not flow properly.

**2*.5. Mitigation:***

My solution to this is to unsolder my components that are at the output of my isolated DC/DC converter and to solder it again and make sure to do it properly. Another thing I am going to try is making sure the diode is placed in the correct direction and reversing it if need be. I will also go look at the datasheet of the resistor to confirm that it can handle the output voltage. If it can’t, then I will order a new resistor that can meet these requirements.

**2*.6. Conclusion:***

Overall, the DC/DC converters in the subsystem worked as expected. They both outputted voltages that were extremely accurate to the simulated values. The isolated DC/DC converter outputted the correct voltage as well; however, there was no voltage reading after it went through the first component. Due to this, the voltage reference didn’t receive any power and the differential amplifiers were not able to be tested out. Before the start of ECEN 404, corrections will be made to the circuit so that the voltage can be passed to every component on the board and run as expected.

# Internal Signal Transfer and Data Manipulation Subsystem:

**3.1. Subsystem Introduction:**

The internal signal and data manipulation subsystem (ISDMS) is designed to pass multiple digital signals across an isolation barrier from a 5-volt power side to a 3.3 volt power side. The 5volt side is also the side which connects to the fuel cells

**3.2. Subsystem Details:**

The ISDMS of two ADS8344N/1k analog to digital converters, two ACSL6400-50TE opto-isolators, and one MAX6037\_41 Voltage reference. The analog to digital converters receive the analog signal from the differential op-amps and sends the digital value back to the microcontroller through the opt Isolators. See Figure One below for high level circuit diagram.

Diagram, schematic

Description automatically generated

Figure 6. Internal Signal Transfer and Data Manipulation High Level Circuit Schematic.

**3.3. Subsystem Validation:**

Validation of the ISDMS was done on a bread board using an Arduino as the microcontroller and an oscilloscope to confirm the signal transfer on the isolated side of the circuit.

**3.3.1 Signal Transfer Validation:**

The signal from the micro processor was successfully passed from the microcontroller to the ADC as depicted in figure 7. The maximum achievable frequency was using the current design is one megahertz before the signal appears undefinable.

A screen shot of a computer

Description automatically generated with low confidence

Figure 7. ISDMS Oscilliscope Signal Verification.

**3.3.2. ADC Control and Feedback:**

Control of the ADC as well as receiving a signal from the ADC was not validated. See section 3.4. for further explanation and discussion.

**3.4. Diagnostic and Mitigation:**

The following sections outline possible contributing factors in the failure of the ADC validation

**3.4.1. Output Resistor Values:**

PROBLEM: The resistors on the output of the opto-isolators have current values of ten kilo Ohms. During the initial testing of the circuit, the resistor for the clock frequency was changed to one kilo Ohm and produced a signal which more closely resembled that of a square wave than the clock signal depicted in purple in figure 2. Further testing is necessary to determine if distortion in the signal is the cause of the malfunction.

SOLUTION: Change each output resistor in the circuit to one kilo Ohm and retest for ADC controllability.

**3.4.2. Arduino Serial Peripheral Interface (SPI) Set Up:**

PROBLEM: The Arduino microcontroller has a SPI system which requires the user to choose the parameters such as bit read direction, clock frequency, and edge trigger. The parameters must match the expected parameters of the device for the device to function properly. These parameters can be determined from the device data sheet but are not in terms of the Arduino IDE syntax.

SOLUTION: The current Arduino SPI parameters must be changed one at a time and the circuit retested for each change. Differences in each test must be annotated and tracked until the necessary configuration is achieved.

**3.5. Subsystem Conclusion:**

The ability of the circuit to transfer a digital signal was validated for frequencies of 250 Kilohertz to 1 Megahertz. The ability of the circuit to control the ADC was unsuccessful and requires more testing and circuit design adjustments. The possible problems and solution from section 3.4 will be the first to be tested for design validation. If neither of these solutions proves effective, consultation with experienced engineers will be pursued for possible design solutions. There is more testing and validation to be completed prior to integration in 404.

# ESP32 and PIC32 Subsystem Report

## Subsystem Introduction

The ESP32 and PIC32 subsystem is designed to take voltage readings from the AD converter and sends it to an android application through Wi-Fi. The PIC32 will receive voltage readings from the AD converter through SPI communication. The PIC32 will then store these readings will be stored into an array, then sent to the ESP32 through UART communication. Then, the ESP32 will communicate to the database through Wi-Fi to send these voltages to the database.

## Subsystem Details

Shape

Description automatically generated with low confidence

Database/ Application

ESP32

PIC32

UART

SPI

A/D Converter

***Figure8: Block diagram of the ESP32 and PIC32***

Some of the challenges with this subsystem is sending and receiving data from the database. The wrong URL was used to communicate with the ESP32 causing errors and causing the ESP32 not communicate with the database properly.

## Subsystem Validation

The ESP32 was first tested for connection to Wi-Fi. This was done using the Espressif IDE to code the ESP-WROOM-32 using C language to show the ESP32 can successfully connect to Wi-Fi. The ESP32 was successfully connected to Wi-Fi, and it displays the SSID and the password of the network that it is connected to. In the event where the ESP32 disconnects with the Wi-Fi or it can’t find the network, it will display that.

A picture containing text

Description automatically generated***Figure 9: ESP32 connection to Wi-Fi***

The connection of the ESP32 to Wi-Fi, by using Espressif IDE and the ESP-WROOM, was confirmed. This confirmed the ability to communicate with the database. In conjunction with the characterization of the connection to Wi-Fi above, this test confirms that as long as there is Wi-Fi where the SSID and password is given, ESP32 can connect to Wi-Fi, thereby validating its performance.

The communication of UART of the ESP32 was then tested to make sure that the ESP32 can communicate to the PIC32. The read function of the ESP32 worked correctly reading the data that is sent through a terminal, realterm.

Graphical user interface, text, application

Description automatically generated

***Figure 10: ESP32 UART communication***

The communication through UART will ensure that the ESP32 will be able to communicate to the PIC32. This will confirm that the data that was sent from the PIC32 will be able to transfer to the ESP32 that will then be sent to the database through Wi-Fi. In conjunction with the characterization of the UART communication above, this test confirms that ESP32 can communicate through UART, thereby validating its performance.

The connection of the ESP32 to the database was then tested. The wrong URL for the database was used to test the connection to the database through Wi-Fi.

Scatter chart

Description automatically generated with medium confidence

***Figure 11: ESP32 database connection***

The ability to connect ESP32 to database, by using the HTTP get and post functions, was not properly working due to the wrong usage of the URL. This error was also there because there was an error was the access with the actual database website.

For the PIC32, the UART communication was implemented to ensure that the PIC32 would be able to communicate with the ESP32. This is done using MPLABX and the PIC32MX230f256b using C language to show that the PIC32 can send and receive data from the ESP32.

Graphical user interface, text, application, email

Description automatically generated

***Figure 12: UART code for PIC32***

The communication through UART will ensure that the ESP32 will be able to communicate to the PIC32. This will confirm that the data that was sent from the PIC32 will be able to transfer the voltages from the AD converter to the ESP32. Due to a fried PIC32. Due to a fried PIC32 chip, this code was not able to be validated.

For the PIC32, the array was created to show that the PIC32 was able to store the voltage readings inside an array. This is done using MPLABX and the PIC32MX230f256b using C language to show that the PIC32.

Graphical user interface, text, application

Description automatically generated

***Figure 13: Array code for PIC32***

The code above should display the array created one by one using a for loop. The will confirm that the PIC32 should be able to store the values in an array and display them when needed. In conjunction with the characterization of the array code above, this test confirms that PIC32 can create and display an array, thereby validating its performance.

For the PIC32, the SPI communication was implemented for the PIC32 to be able to communicate with the A/D converter. This is done using MPLABX and the PIC32MX230f256b using C language to show that the PIC32 can send and receive data from the ESP32.

Graphical user interface, text, application, email

Description automatically generated

***Figure 14: SPI code for PIC32***

The communication through SPI will ensure that PIC32 would be able to communicate with the A/D converter. This will confirm that the data from the fuel cell that was sent through the A/D converter will be stored in the PIC32 and then sent to the ESP32. Since an A/D converter was not present, the SPI code was not able to be tested.

## Subsystem Diagnostic and Mitigation

### ESP32 connection to the database

ESP32 is supposed to connect to the database through Wi-Fi to send data receive from the PIC32. The ESP32 was not able to connect to the database due to the usage of the wrong URL, and not having access to the database. In order to solve this issue, the right URL needs to be used and the HTTP get and post functions need to be functional to get the ESP32 to connect to the database.

### UART and SPI communication for PIC32

Due to the fried chip for the PIC32, the test and validation was not completed for the PIC32. In addition, the UART code that the first written was for a button on a bread for to send signal to the PIC32 in order to display the UART communication, however the code didn’t work due to the fried PIC32. For the UART communication, once the curiosity board arrives, the code will be tested on the curiosity board and realterm terminal to ensure that the PIC32 will be to send and receive data from the ESP32. The code that was first written for the A/D converter was reading the values using the A/D converter that was inside the PIC32 as shown in the figure below.

Graphical user interface, text

Description automatically generated

***Figure 15: Wrong ADC code for PIC32***

However, the PIC32 is supposed to read the A/D converter that is designed by the internal signal transfer and data manipulation subsystem through SPI communication. Once the A/D converter is available, the SPI communication code will be able to be tested. The A/D converter will send the data received from the fuel cell through SPI, which will be able to store the data in PIC32.

## Subsystem Conclusion

The ability for the ESP32 to connect to Wi-Fi was validated using the Espressif IDE and ESP-WROOM-32. The UART communication for the ESP32 was validated using the realterm terminal and Espressif IDE. Before the start of ECEN 404, the ESP32 would connect and communicate with the database. In addition, the PIC32 would be able to send and receive from the A/D converter using SPI communication, and the PIC32 would be able to communicate to the ESP32 using UART.

# Android Application Subsystem Report

## Subsystem Introduction

The android application is designed to display the data for each fuel in the stack to the user. This application was tested to confirm its ability to operate on an android device, to properly display accurate data to the user and be fully functional, connect to a database containing data related to the fuel cells, and update the user of any errors detected in the fuel cell stack.

## App running on Android Device

### Operation

For development purposes, the app was continuously tested on the emulator within Android Studio. The emulator was the virtual version of the Google Pixel 4a API 30. For demo and validation purposes, the app was run and tested on an actual hardware android device.

### Validation

It was validated through running the app on an actual hardware android device. The app was able to show up on the device and perform as expected.



Figure 16: App running on a hardware device

## App Connecting to Database

### Operation

Firebase was used to create the NoSQL database and store data in the database. The initial use of AWS to create a PostgreSQL database proved to be unsuccessful and moving to Firebase resulted in better compatibility with the app. The sample data was structured as desired and was to be used to send the data to the app to display using graphs.

### Validation

It was validated through the connection status being displayed in Android Studio. The database was successfully able to connect to the app, but unfortunately retrieving the necessary data from the database was not successful by the time of this report.

Text

Description automatically generated

Figure 17: Connectivity between the app and database

## Properly Display Accurate Data to Users

### Operation

The app was able to display accurate voltage levels for the fuel cells in the stack to the user. The MPAndroidChart package was used to create and display the necessary graphs in the app. For further understanding and readability, the user is also able to access the voltage levels of the fuel cells in the form of a table.

### Validation

It was validated through the app properly displaying the data in the corresponding sections of the app. All buttons in the app work as expected and the app does not crash upon use.

  Graphical user interface, table

Description automatically generated

Figure 18: Status Graph Figure 19: Fuel Cell Table

## Alerts to Users Regarding Errors

### Operation

There is a section in the app on the homepage for the user to check and see if any errors have been detected within the fuel cell stack. The message will update if any errors are detected.

### Validation

It was validated through the correct message being displayed in the corresponding section of the app.

Graphical user interface, application

Description automatically generated

Figure 20: Error Detection

## Subsystem Diagnostic and Mitigation

### Diagnostic

The only thing not fully working in my subsystem is pulling data from the database and accurately showing the data in the respective figures in my app. When attempting to read data from the database, it returns null values. As of now most things are hardcoded to display the accurate results. Connectivity to the database has been validated, so the problem is certainly occurring in the reading data process of my code. The GUI could also be formatted in a nicer way, particularly when it comes to the graphs included in the app.

### Mitigation

To solve the issues currently present in the app, I plan to change the way I am reading data from the database to properly display data. I will restructure the data in the database, if necessary, to be able to access the desired nodes. I will use a different format for displaying historical data rather than a regular bar graph, to better display the data for the users.

## Subsystem Conclusion

The subsystem overall worked as expected, with a few minor setbacks. The user can use the app on an android device and view accurate data and alerts regarding the fuel cells. The proper adjustments and corrections will be made before the start of ECEN 404, and the subsystem will be ready for integration.