Fuel Cell Monitor

Jessica Odutola
Russell Wells
Rana Kortam
Sameer Osama

CONCEPT OF OPERATIONS

CONCEPT OF OPERATIONS
FOR
Fuel Cell Monitor

CONCEPT OF OPERATIONS
FOR
Fuel Cell Monitor

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	[09/15/2022]	Fuel Cell Monitor System		Draft Release
2	10/02/2022	Russell Wells		FSR, ICD attachment
3	12/04/2022	Russell Wells		403 Final Report Attachment

Table of Contents

List of Tables

List of Figures

1. Executive Summary

A single fuel cell is an easy power source to monitor, but to achieve any level of real usable power they must be connected in a stack which provides a usable amount of power. Our fuel cell monitor will be focused on the integrity of individual cells. The proposed device will monitor individual voltages on a real time basis, transmit the data via Wi-Fi to a mobile app which will have a graphical display of the individual cell voltages.

2. Introduction

Fuel cell monitor is used to measure the efficiency and electrical characteristics of fuel cells [2]. A Fuel cell is an electrochemical energy conversion device where it utilizes hydrogen and oxygen to generate electricity, heat, and water [3]. The fuel cell monitor that will be created will provide real-time monitoring of every cell in each fuel stack. This monitor will be able to communicate via Wi-Fi to transfer the data from the cell to an application. The purpose of this monitor is to track the fuel cells to see if they fail at some point.

2.1. Background

Fuel cells are not new technology, and neither are fuel cell monitors. However, with the more recent push for clean and renewable energy, different types of fuel cells have become much more prevalent in research as well as personal use of recreational equipment. Some of the fuel cells take hydrogen and use it to generate electricity. The fuel cell monitor will be used to track these voltages. The monitor will have a microcontroller unit which will be used to store the data from the fuel cell.

2.2. Overview

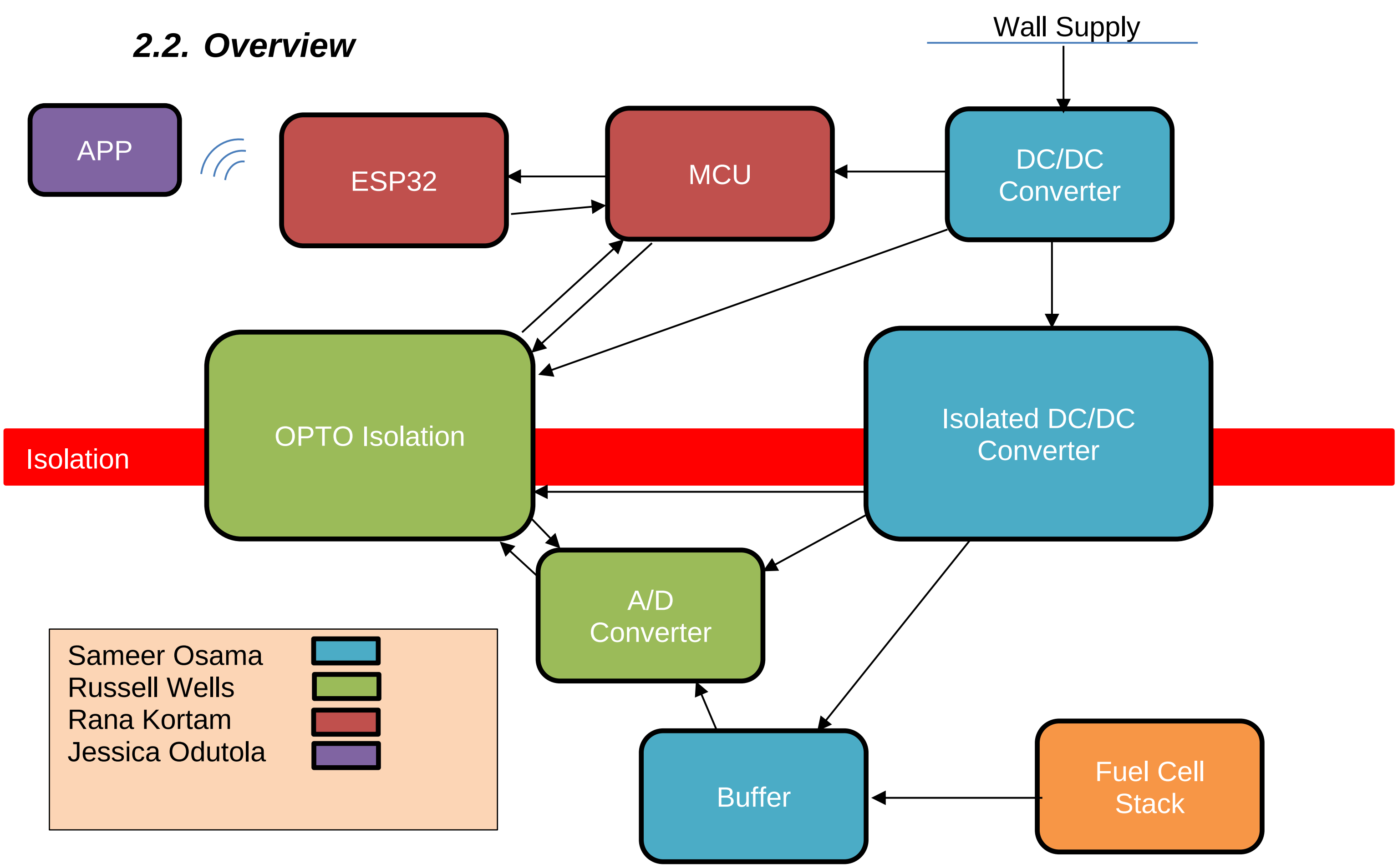

Figure 1: Fuel Cell Monitor Block Diagram

The proposed device is not intended as a replacement for a more robust control system such as SCADA but is intended as an affordable, low maintenance, mobile, voltage monitoring and alarm system. The system will display real-time voltages on the app and push notifications to the user via the app when a cell is not operating within expected ranges. The monitor will not have a control aspect but will send a warning to the owner/operator when action is required to preserve fuel cell integrity.

The app is going to allow real-time monitoring as well as provide historical trend data that can be used to diagnose current issues and anticipate future failures based on past experiences.

2.3. Referenced Documents and Standards

[1] Marsh, Jane. "5 Hydrogen Fuel Cell Environmental Impacts." *Environment Co*, 19 Nov. 2020, https://environment.co/hydrogen-fuel-cells-environmental-impacts/.

[2] "Fuel Cell Monitor Pro 4.0." *Fuel Cell Store*, https://www.fuelcellstore.com/fuel-cell-monitor-pro-3-u103.

[3] "Fuel Cell Basics." *Fuel Cell & Hydrogen Energy Association*, https://www.fchea.org/fuelcells.

3. Operating Concept

3.1. Scope

The scope of this project is to provide a monitor that will be able to transfer all relevant parameters to track fuel cells. The exact deliverables for the scope of this project are as follows:

- Power system development
- Internal signal transfer and manipulation development
- Microcontroller development
- Application development

Documentation for the design, construction, and programming of the units will be provided for all parts of the project.

3.2. Operational Description and Constraints

The fuel cell monitoring system is intended for use in a controlled laboratory or secure location. The system will include Wi-Fi-enabled data transfer from the device to a user's cell phone.

Constraints:

- The device is not intended for use outdoors or in harsh environments
- The device requires a 110Vac wall outlet for power

3.3. System Description

- Power System: A wall wart power supply will be used to convert the AC voltage to DC voltage. This DC voltage will be passed through multiple DC/DC converters to bring the voltage down to different levels. The new lower voltage levels will be supplied to the microcontroller, the opto-isolator, and the isolated DC/DC converter. The isolated DC/DC converter is used to pass power over the isolation line and provide power to the op-amps and the ADC.

- Internal Signal manipulation and transfer: The voltages from the fuel cells will be passed through an op-amp buffer to an analog to digital converter. The converter will send the digital signal to an opto-isolator which will transfer the signal as a scaled current that is representative of the differential voltage.

- Signal Processing: The signal from the opto-isolator will be sent to the microcontroller unit where the data will be stored. Then, the microcontroller will be connected to another microcontroller, ESP32, with a UART where the data will be transferred to the application.

- Application and graphical display: The application will display the voltages in the form of a graph of each fuel cell in the stack to the user. The user will be able to navigate to the fuel cell of their choice and/or the fuel cell stack to view the voltages.

3.4. Modes of Operations

- Normal Operating Mode: All cells are within a preset range and the data is displayed graphically on the App.
- Cell Alarm Mode: One or more of the cells have fallen below threshold voltage or spiked above. App will display the faulty or over loaded cell as defective.
- System Alarm Mode: The app no longer receives information from the system outside of normal disconnect.

3.5. Users

- Initial Installer: An electrician or fuel cell technician should conduct the initial installation and test.
- General Operator: The normal operator of this device will not require training. Anyone with access to an android smartphone will be able to connect to the device through the app and monitor the system.
- Maintenance Technician: Maintenance on the device, beyond.

3.6. Support

User will be given a parts list with all replaceable parts outlined for purchase. These parts would include the wall power supply, type and quantity of external signal wires, and a component list for all major PCB components. The program for the microcontroller and App will not be supplied

4. Scenario(s)

4.1. Experimental Lab Equipment

Our project, although not all inclusive, will be relatively inexpensive and less concerning when connected to experimental fuel cell systems. The use of the application as a monitoring system will also allow the user to monitor the fuel cells and maintain historical data even if left alone for an extended period.

4.2. Personal or Home Power Generation

The system could be used by a homeowner or non-commercial entity where an individual would like to be able to monitor a fuel cell stack.

5. Analysis

5.1. Summary of Proposed Improvements

An improvement that the proposed system will provide is the mobile app connectivity. Another improvement is cost efficiency.

5.2. Disadvantages and Limitations

The limitations of the Fuel Cell Monitor include:

- The app used to monitor and display the voltages for each individual fuel cell is only available on Android devices.
- To use our monitor, it is necessary to be near an outlet.
- The monitor cannot be used outdoors.

5.3. Alternatives

- Full monitor and control system would be an alternative. A PLC cabinet with SCADA control nodes for back pulse and temperature regulation as well as full power monitor would be an alternative and would do the job of our proposed device but at a much higher cost to the client in both initial implementation as well as annual maintenance.

5.4. Impact

[1] Fuel cells, depending on where the hydrogen is extracted, have a positive impact on the environment. Hydrogen can be found in most things in nature; therefore, hydrogen fuel cells are a renewable resource. The ubiquity of hydrogen in nature means that the fuel cells are also a sustainable resource. The use of hydrogen fuel cells significantly reduces carbon emissions. In comparison to their alternatives, fuel cells are the best option for the environment. Being able to monitor the levels of the fuel cells ensures that the negative effects on the environment are limited.

Fuel Cell Monitor

Rana Kortam
Russell Wells
Sameer Osama
Jessica Odutola

INTERFACE CONTROL DOCUMENT

REVISION – Draft
03 October, 2022

Interface Control Document
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 II, P.E. Date

____
Dalton Cyr Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	[10/01/2022]	Fuel Cell Monitor System		Draft Release
2	04/12/2022	Russell Wells		403 Final Report Attachment

Table of Contents

List of Tables

No table entries found.

List of Figures

No figure entries found.

1. Overview

The following sections outline the physical, electrical, and communication characteristics of the fuel cell monitor System. Some of the fields outlined below are subject to change depending on availability of parts and part characteristics

2. References and Definitions

Provide any references (i.e., standards documents) and definitions. Examples are shown below.

2.1. References

Not Applicable at this time.

2.2. Definitions

ADC	Analog to digital converter
mA	Milliamp
mW	Milliwatt
MHz	Megahertz (1,000,000 Hz)
TBD	To Be Determined
TTL	Transistor-Transistor Logic
VME	VERSA-Module Europe

3. Physical Interface

3.1. Weight

The weight of the Fuel Cell Monitor shall be less than or equal to 0.25 kilograms.

3.2. Dimensions

Dimensions are unknown at this time, but the volume is expected not to exceed 0.125 cubic feet. FSR will be updated when dimensions are known

3.3. Mounting Locations

Specific mounting options will not be considered for this system.

4. Electrical Interface

4.1. Power

4.1.1 Primary Input Power

Primary input power shall be from a standard wall outlet through a wall wart AC to DC converter which will bring down the voltage to 12 VDC.

4.1.2 Internal Power

Internal power shall be regulated through DC-DC converters and stepped down to voltages ranging from +2.7 to +5.5 VDC.

4.2. Signal Interfaces

4.2.1 Raw Data Signal Interface

The voltages from the fuel cell will be passed to the monitor system via signal wire to PCB mounted terminal block.

4.2.2 Internal Data Signal Interface

The internal signal shall be transferred to the microcontroller via Op-amp filter, ADC, and opto-isolator.

4.3. User Control Interface

4.3.1 User Graphical Interface

The android application shall display voltages for both the fuel cell stack and the individual fuel cells. The user shall receive alerts when errors occur with the fuel cells.

4.3.2 User Control Interface

The app shall allow the user to set both low and high point alarms.

s

5. Communications / Device Interface Protocols

5.1 Wireless Communications (Wi-Fi)

The microcontroller has a built-in Wi-Fi module using IEEE 802.11 b/g/n standards. This connection will be used to send a user android application for review.

5.2 Device Peripheral Interface

The MCU will connect to the ESP-32 microcontroller through a UART port. This allow to transfer the signal from the MCU to the ESP-32 microcontroller to send the signal to the application.

Fuel Cell Monitor

Jessica Odutola
Russell Wells
Rana Kortam
Sameer Osama

FUNCTIONAL SYSTEM REQUIREMENTS

FUNCTIONAL SYSTEM REQUIREMENTS
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	10/03/2022	Fuel Cell Monitor System		Draft Release
2	12/04/2022	Russell Wells		403 Final Report Attachment

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Purpose and Scope

Monitoring a fuel cell to make sure it doesn't fail or give a different voltage than expected is important to the fuel cell stack. Our aim is to provide a fuel cell monitoring system that will take the readings from each fuel cell in the fuel cell stack and to monitor these readings through an application. With our system, the voltage will be taken from the fuel cell stack and then passed through a differential amplifier to filter the noise. The signal will then go to the microcontroller via ADC and opto-isolator which provide a signal the processor can read and protect in the event of a short or cell malfunction. Then, the microcontroller will send the data to the ESP32 which will communicate with the app to display the voltage for each fuel cell in the stack.

Figure 1. Fuel Cell Monitor Conceptual Image

1.2. Responsibility and Change Authority

Subsystem	Responsibility
Power System Development	Sameer Osama
Internal Signal Transfer and Manipulation Development	Russell Wells
Microcontroller Development	Rana Kortam
Mobile Application Development	Jessica Odutola

Table 1: Subsystem Leads

2. Requirements

2.1. System Definition

The Fuel Cell Monitor is a practical and reliable system. It allows users to observe voltage variations on each individual fuel cell via mobile app. The Fuel Cell Monitor has four sub-systems: Power System, Internal Signal Transfer and Manipulation, Microcontroller, Application Development.

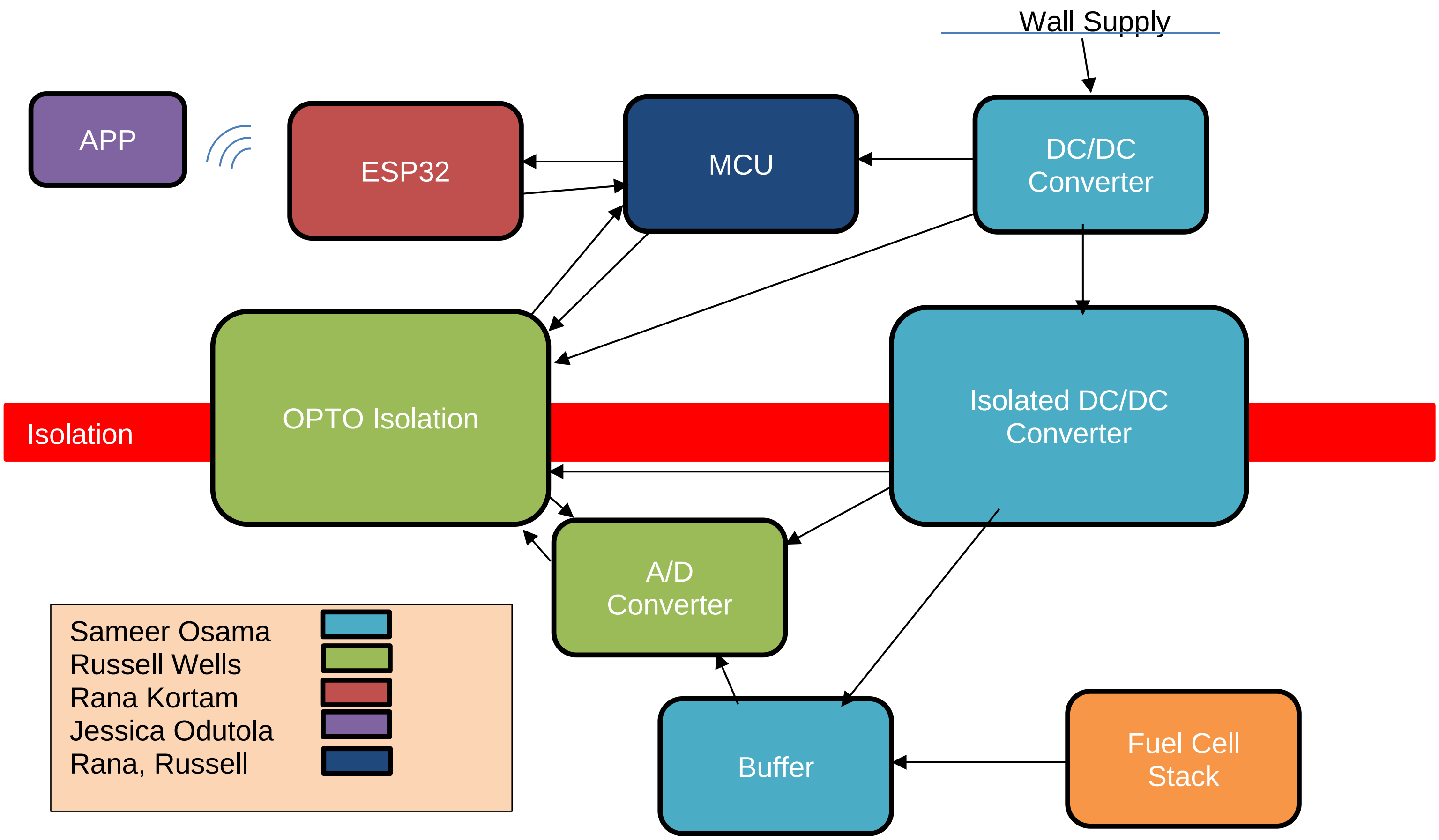

Figure 2. Block Diagram of System

In figure 2, the block diagram above shows the different subsystems within the Fuel Cell Monitor. The power for the whole system will be provided by a wall supply, which will be then converted to DC voltage using a DC/DC converter. The voltage reading will be taken from the fuel cell, and it will pass through the buffer to reduce noise. The signal will then be passed to the microcontroller via ADC and opto-isolator which provide a signal that processor can read as well as protect in the event of a short or cell malfunction. Then, the microcontroller will send the data to the ESP32 which will communicate with the app to display the voltage for each fuel cell stack.

2.2. Characteristics

2.2.1. Functional / Performance Requirements

2.2.1.1. Fuel Cell Monitor Voltage Range

The fuel cell monitor system shall be capable of monitoring voltages ranging from 0 VDC to +4 VDC.

Rationale: The requirement specified by the client was roughly 0.7 VDC. However, the capabilities of the parts used in the system allow for a higher range of voltage inputs.

2.2.2. Operating Environment

The fuel cell monitor shall be capable of operating within a controlled laboratory or under non-environmentally harsh conditions.

Rationale: The client specified a laboratory environment in which fuel cells are being used for experimentation.

2.2.3. Physical Characteristics

2.2.3.1. Mass

The mass of the Fuel Cell Monitor shall be less than or equal to 0.25 kilograms.

Rationale: This is a requirement specified by our customer due to constraints of their system in which the Fuel Cell Monitor is operating.

2.2.3.2. Volume Envelope

The volume envelope of the Fuel Cell Monitor system shall be less than or equal 0.125 cubic feet.

Rationale: The monitoring system is intended to be simple and capable of monitoring multiple fuel cell types in multiple stack configurations. The monitor must be small enough to sit inside the stack enclosure or near it without interfering with stack operation and maintenance.

2.2.3.3. Mounting

The fuel cell monitor will be designed to sit on a shelf or the floor inside the fuel cell housing

Rationale: This is a requirement of the client. The system is also compatible with any fuel cell stack producing voltages within the specified range and universal mounting is outside the project's scope.

2.2.4. Electrical Characteristics

2.2.4.1. Signal Inputs

The inputs of the system will be voltages transferred directly from the fuel cells to the monitor system via signal wire.

Rationale: This is a necessity given the nature of the monitor system

3.2.4.2. Power Consumption

The maximum peak power of the system shall not exceed 3 Watts.

Rationale: Although the major components have been determined, the minor components have not. % Watts is expected to be much higher then is necessary.

3.2.4.3. System Voltage Input

The input voltage level for the Fuel Cell Monitor System shall be +2.7 VDC to +5.5 VDC.

Rationale: This is a requirement outlined by the client and constrained by the usable parts.

3.2.4.4. Data Output

The Fuel Cell Monitoring System shall include an android application for users to view alerts if a fuel cell fails. The android application shall display the voltage levels of both the stack and the individual fuel cells.

Rationale: The Search and Rescue information passes directly to the customer's system.

3.2.4.5. Diagnostic Output

The Fuel Cell Monitoring System shall include a diagnostic interface for control and data logging.

Rationale: Provides the ability to control things for debugging manually and a way to view/download the node map with associated potential targets.

2.2.5. Environmental Requirements

The Fuel Cell Monitoring System shall be designed to withstand and operate in the environments specified in the following section.

2.2.5.1. Thermal

The Fuel Cell Monitoring System shall be able to function in environments ranging from -40°C to 85°C.

Rationale: The range of temperatures is provided in the datasheets of the system components.

2.2.5.2. Water Damage

The Fuel Cell Monitoring System will be capable of withstanding small non continuous splashes or misting with water.

Rationale: Given the nature of fuel cells such as hydrogen fuel cells. It is necessary to ensure the device is not susceptible to minor exposure to water.

3. Support Requirements

The Fuel Cell Monitoring system requires an internet connection to interact with the applications to give notifications. User must provide WI-FI and a wall supply to supply power to the rest of the system.

Appendix A: Acronyms and Abbreviations

BIT	Built-In Test
CCA	Circuit Card Assembly
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
EO/IR	Electro-optical Infrared
FOR	Field of Regard
FOV	Field of View
GPS	Global Positioning System
GUI	Graphical User Interface
Hz	Hertz
ICD	Interface Control Document
kHz	Kilohertz (1,000 Hz)
LCD	Liquid Crystal Display
LED	Light-emitting Diode
mA	Milliamp
MHz	Megahertz (1,000,000 Hz)
mW	Milliwatt
PCB	Printed Circuit Board
RMS	Root Mean Square
TBD	To Be Determined
TTL	Transistor-Transistor Logic
USB	Universal Serial Bus
VME	VERSA-Module Europe

Fuel Cell Monitor System

Rana Kortam, Russell Wells, Sameer Osama, Jessica Odutola

SCHEDULE

Revision – 2

4 December 2022

Fuel Cell Monitor Execution Plan Fall 2022

	9/5/2022	9/12/2022	9/19/2022	9/26/2022	10/3/2022	10/10/2022	10/17/2022	10/24/2022	10/31/2022	11/7/2022	11/14/2022	11/21/2022	11/28/2022	DATE
TEAM DELIVERABLES														
Understand Project Problem														
Project design Overview														Completed
Divide Into Subsystems														In Progress
ConOps Report														Not Started
Create Major Parts List														Behind Schedule
FSR, ICD Report														
Midterm Presentation														

Order Major Parts														
Status Update Presentation														
Final Presentation														
Final Demo														
Final Report														
POWER SUBSYSTEM														
Determine IC Components														
Design Schematics														
Order IC components														
Create PCB footprints in Altium														
Create PCB design in Altium														
Make Gerber files and send to FEDC														

Test components on circuit board														
INTERNAL SIGNAL SUBSYSTEM														
Determine IC Components														
Design System														
Order Components														
Create PCB Schematic														
Assemble and Test Demo														
Create PCB Design														
Order PCB														
MICRO CONTROLLER SUBSYSTEM														
Determine Microcontrollers in use														

Learn IDE to code microcontroller														
Implement "Hello World" on ESP32														
WiFi connection on ESP32														
UART on ESP32														
UART on PIC32														
Array code for PIC32														
SPI AD Converter code														
Connect ESP32 to database														
APP SUBSYSTEM														
App Displays "Hello World"														
App Displays Home Page														

App Displays all pages needed														
AWS Database Created														
Tables Populated in Database														
Connect Database to App														
App Sends Alerts to Users														
App Works with Test Data														

Fuel Cell Monitor System

Rana Kortam, Russell Wells, Sameer Osama, Jessica Odutola

VALIDATION PLAN

Revision – 2

4 December 2022

Fuel Cell Monitor Validation Plan

Paragraph #	Test Name	Success Criteria	Methodology	Status	Responsible Engineer(s)
3.2.4.2	Power Devices On PCB	PCB transfers power without overheating or burnout	Power Board and watch, smell, listen	Untested	Russell, Sameer
3.2.1.1	Internal signal voltage range	System can properly handle the specified voltages with minimal difference between tests.	Introduce voltages of 0-4V and measure output signals	FAIL	Russell
3.2.1.1	Differential voltage tests	Pass a differential voltage through the Opamp buffer and receive the proper digital signal from the optoisolator	Introduce a range of voltages including edge cases and ensure proper output	Untested	Russell, Sameer
3.2.4.4	Android application graphical functionality	Application can properly display accurate voltage levels to user.	Use application on android device and verify volatages are accurately displayed	Passed	Jessica
3.2.4.4	Android application database read and write data functionality	Application can properly read and write data from Firebase Database	Graph uses data pulled from the database as values	Fail	Jessica
3.2.4.4	Android application database connectivity	Application can connect to Firebase Database	Verify connection status within application	Passed	Jessica
3.2.4.4	Android application alarm functionality	Application send alarm to user when voltage goes above or below ranges	Add set points to app and introduce alarm level voltages	Untested	Jessica
3.2.4.2	Power system functionality test	Power is applied from wall outlet and proper power transfer is read at outputs	Apply power to system and read voltage output at device trace	Untested	Sameer

3.2.4.1	Opamp system functionality test	Differential voltages are passed to the opamp and expected voltage is seen on the output	Power opamps and apply varrying differential voltages and read output voltage	Untested	Sameer
N/A	PIC32 Microcontroller functionality test	The code for recieving the voltage signal for data acquisition	PCB board and coding on IDE	Untested	Rana
N/A	ESP32 Microcontroller functionality test	The code for communicating with the application	PCB board and coding on IDE	Tested	Rana

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

List of Tables

List of Figures

1. 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.

2. Power Subsystem

2.1. 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.

2.2. 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.

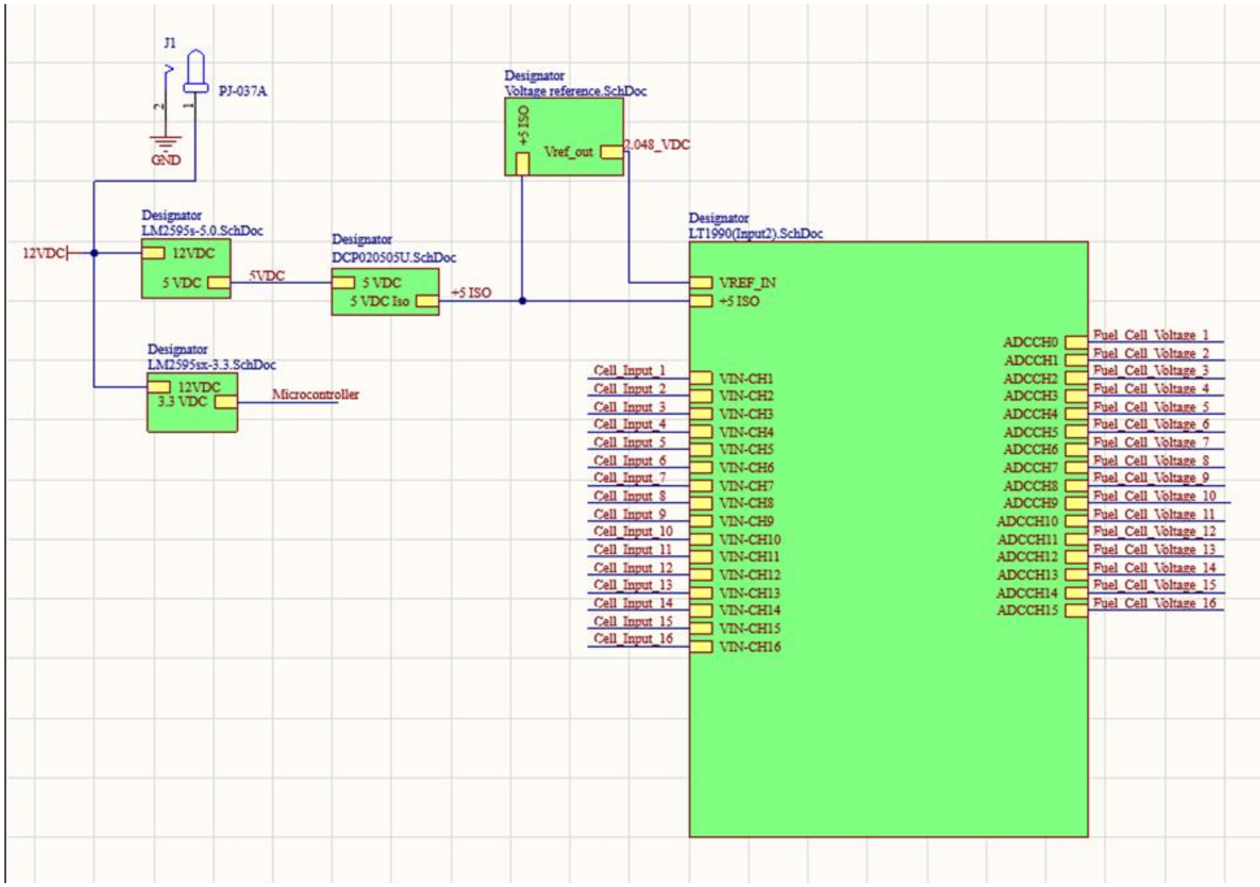

Figure 1. Power Subsystem High Level Circuit Schematic

2.3. 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 1. 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.

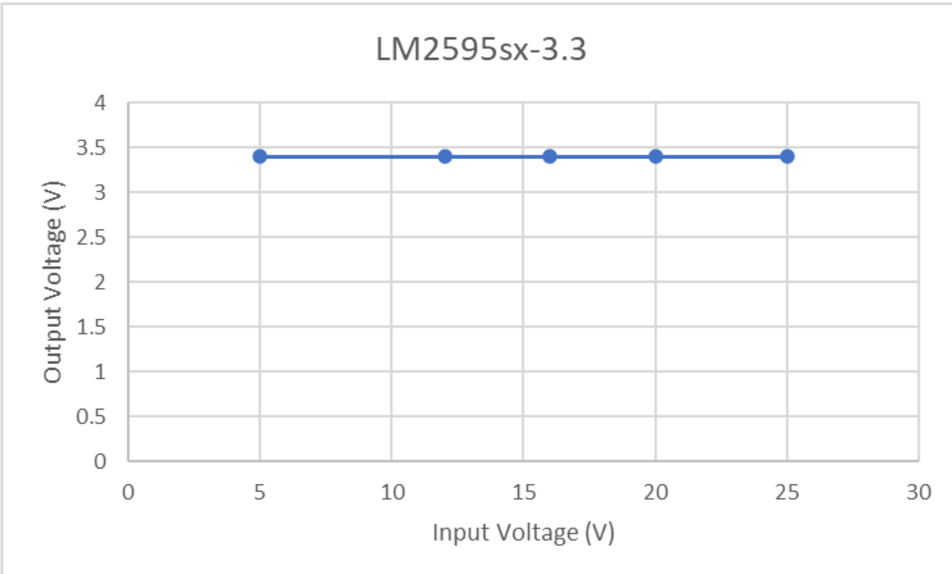

Figure 2 Power Subsystem Output to Output Voltage, LM2595sx-3.3

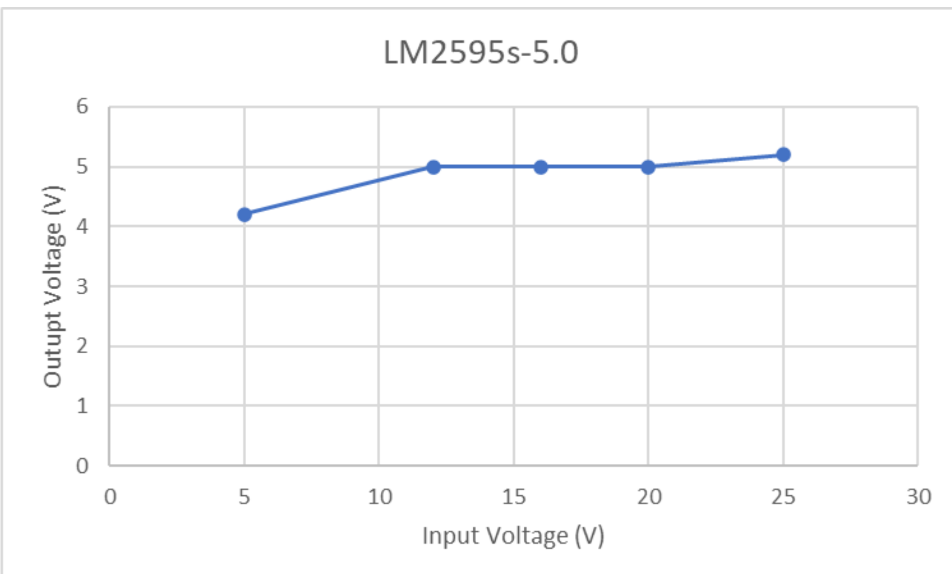

Figure 3.Power Subsystem Output to Output Voltage, LM2595s-5.0

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.

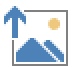

Figure 4. Power Subsystem DC/DC Converter Noise Measurement LM2595sx-3.3

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

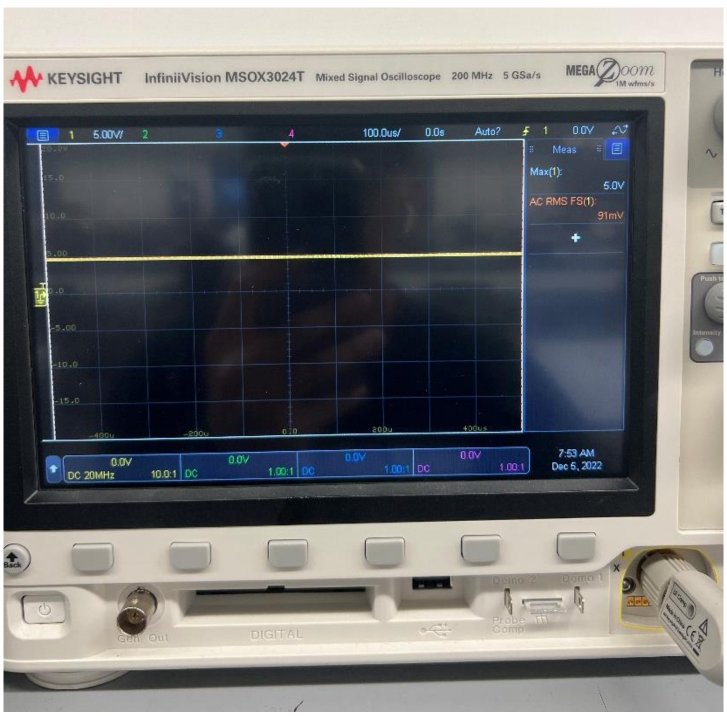

Figure 5.Power Subsystem DC/DC Converter Noise Measurement LM2595s-5.0

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.

3. 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.

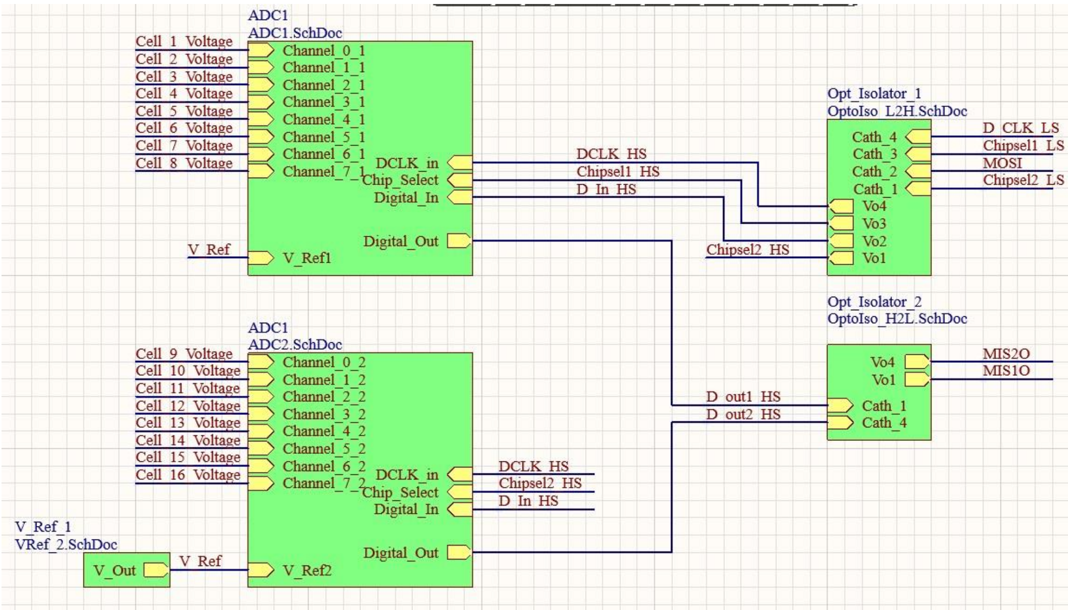

Figure 6. Internal Signal Transfer and Data Manipulation High Level Circuit Schematic.

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.

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.

4. ESP32 and PIC32 Subsystem Report

4.1. 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.

4.2. Subsystem Details

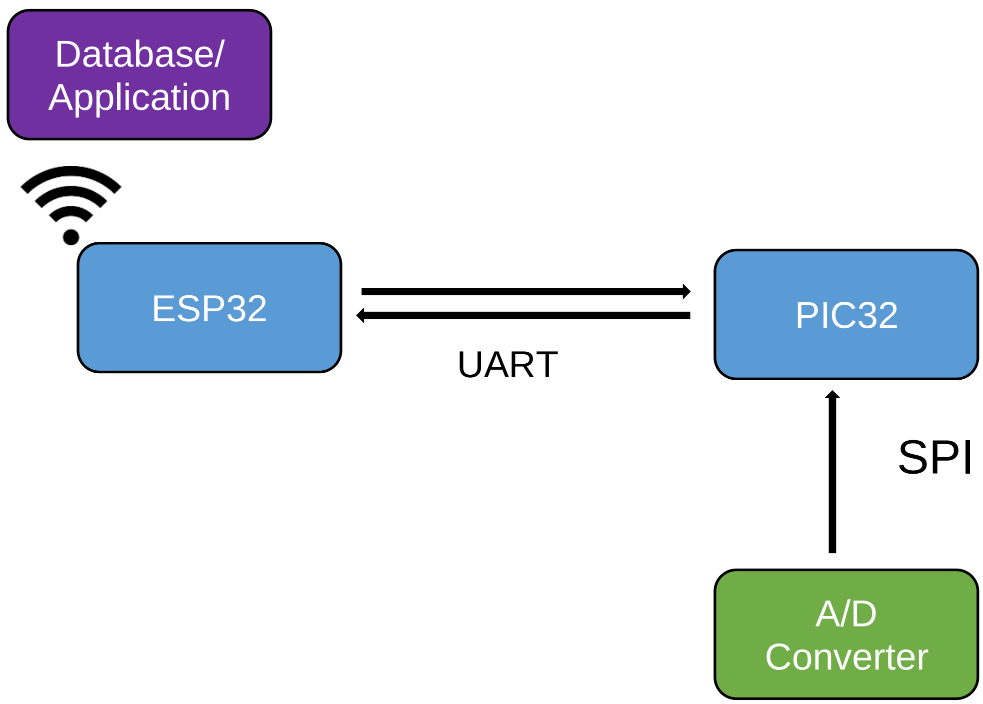

Figure8: Block diagram of the ESP32 and PIC32

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.

4.3. 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.

```
COM5
I (788) wifi:state: init -> auth (b0)
I (788) wifi:state: auth -> assoc (0)
I (798) wifi:state: assoc -> run (10)
I (1148) wifi:connected with NETGEAR82, aid = 7, channel 11, BW20, bssid = b0:7f:b9:13:37:58
I (1148) wifi:security: WPA2-PSK, phy: bgn, rssi: -59
I (1158) wifi:pm start, type: 1

I (1238) wifi:AP's beacon interval = 102400 us, DTIM period = 1
W (1768) wifi:<ba-add>idx:0 (ifx:0, b0:7f:b9:13:37:58), tid:0, ssn:7, winSize:64
I (4558) esp_netif_handlers: sta ip: 192.168.1.15, mask: 255.255.255.0, gw: 192.168.1.1
I (4558) wifi station: got ip:192.168.1.15
I (4558) wifi station: connected to ap SSID:NETGEAR82 password:sillybug998
```

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.

```
<Closed> COM5
I (193) boot: Loaded app from partition at offset 0x10000
I (193) boot: Disabling RNG early entropy source...
I (207) cpu_start: Pro cpu up.
I (207) cpu_start: Starting app cpu, entry point is 0x400810f4
0x400810f4: call_start_cpu1 at C:/Users/queen/Desktop/esp-idf-v4.4.2/components/esp_system/port/cpu_start.c:160

I (194) cpu_start: App cpu up.
I (222) cpu_start: Pro cpu start user code
I (222) cpu_start: cpu freq: 160000000
I (222) cpu_start: Application information:
I (226) cpu_start: Project name:     uart_simple
I (231) cpu_start: App version:      v4.4.2-dirty
I (237) cpu_start: Compile time:     Oct 23 2022 13:24:06
I (243) cpu_start: ELF file SHA256:  e00fba47c0b80872...
I (249) cpu_start: ESP-IDF:          v4.4.2-dirty
I (255) heap_init: Initializing. RAM available for dynamic allocation:
I (262) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (268) heap_init: At 3FFB2C60 len 0002D3A0 (180 KiB): DRAM
I (274) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (280) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (287) heap_init: At 4008C270 len 00013D90 (79 KiB): IRAM
I (294) spi_flash: detected chip: generic
I (298) spi_flash: flash io: dio
W (301) spi_flash: Detected size(4096k) larger than the size in the binary image header(2048k). Using the size in the binary image header.
ï¿½Read 4 bytes: '1000'
```

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.

```
COM5
I (1125) wifi:security: WPA2-PSK, phy: bgn, rssi: -70
I (1135) wifi:pm start, type: 1

I (1215) wifi:AP's beacon interval = 102400 us, DTIM period = 1
W (1735) wifi:<ba-add>idx:0 (ifx:0, b0:7f:b9:13:37:58), tid:0, ssn:10, winSize:64
I (3115) esp_netif_handlers: sta ip: 192.168.1.14, mask: 255.255.255.0, gw: 192.168.1.1
I (3115) wifi station: got ip:192.168.1.14
I (3115) wifi station: connected to ap SSID:NETGEAR82 password:sillybug998
E (3645) esp-tls-mbedtls: No server verification option set in esp_tls_cfg_t structure. Check esp_tls API reference
E (3645) esp-tls-mbedtls: Failed to set client configurations, returned [0x8017] (ESP_ERR_MBEDTLS_SSL_SETUP_FAILED)
W (3645) wifi:<ba-add>idx:1 (ifx:0, b0:7f:b9:13:37:58), tid:4, ssn:4, winSize:64
E (3665) esp-tls: create_ssl_handle failed
E (3665) esp-tls: Failed to open new connection
E (3675) TRANSPORT_BASE: Failed to open a new connection
E (3685) HTTP_CLIENT: Connection failed, sock < 0
```

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.

```c
void SendDataBuffer(const char *buffer, UINT32 size)
{
    while(size)
    {
        while(!UARTTransmitterIsReady(UART2))
            ;

        UARTSendDataByte(UART2, *buffer);

        buffer++;
        size--;
    }

    while(!UARTTransmissionHasCompleted(UART2))
        ;
}
// *************************************************************************
// UINT32 GetDataBuffer(char *buffer, UINT32 max_size)
// *************************************************************************
UINT32 GetDataBuffer(char *buffer, UINT32 max_size)
{
    UINT32 num_char;

    num_char = 0;

    while(num_char < max_size)
    {
        UINT8 character;

        while(!UARTReceivedDataIsAvailable(UART2))
            ;

        character = UARTGetDataByte(UART2);

        if(character == '\r')
            break;

        *buffer = character;
```

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.

```c
/*
 * File:   array.c
 * Author: queen
 *
 * Created on November 25, 2022, 1:28 PM
 */

#include <stdio.h>
#include <stdlib.h>
#include "C:\Program Files\Microchip\MPLABX\v6.00\packs\Microchip\PIC32MX_DFP\1.5.259\include\proc\p32mx230f256b.h"

/*
 *
 */
int main(void) {

    int i;
    int voltages[] = { -2000,-1000,0,1000,2000}; //array
    TRISB = 0;                              //initialize Port_B as output
    for(i=0; i<=4;i++){                      //indexing start from 0 to 4
        PORTB = voltages[i];                          //from sum to PORT_B
    }
}
```

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.


```
while(nCycles-- && !fail)
{
    int     txferSize;
    SpiPkt* pTxPkt;
    SpiPkt* pRxPkt;

    txferSize=MIN_SPI_TXFER_SIZE+rand()%(MAX_SPI_TXFER_SIZE-MIN_SPI_TXFER_SIZE+1);  // get a random transfer size

    pTxPkt=(SpiPkt*)malloc(txferSize*sizeof(short)+2);
    pRxPkt=(SpiPkt*)malloc(txferSize*sizeof(short)+2);      // we'll transfer 16 bits words; extra room for nChars

    if(pTxPkt && pRxPkt)
    {
        int             ix;

        for(ix=0; ix<txferSize; ix++)
        {
            pTxPkt->data[ix]=(unsigned short)rand();    // fill buffer with some random data
        }
        pTxPkt->nChars=txferSize;


        SpiChnChangeMode(chn, 1, 1, 1);     // set master mode
        SpiChnPutS(chn, (unsigned int*)pTxPkt, txferSize+2);    // write packet

        SpiChnChangeMode(chn, 0, 0, 1);     // set slave mode
        SpiChnGetS(chn, (unsigned int*)pRxPkt, txferSize);

        // now verify the received buffer contents
        for(ix=0; ix<txferSize; ix++)
        {
            if(pTxPkt->data[ix]!=pRxPkt->data[ix])
            {
                fail=1;     // data mismatch
                break;
            }
        }
```

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.

4.4. Subsystem Diagnostic and Mitigation

4.4.1. 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.

4.4.2. 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.

```
// enable prefetch cache but will not change the PBDIV. The PBDIV value
// is already set via the pragma FPBDIV option above..
SYSTEMConfig(SYS_FREQ, SYS_CFG_WAIT_STATES | SYS_CFG_PCACHE);

// configure and enable the ADC
CloseADC10();   // ensure the ADC is off before setting the configuration

// define setup parameters for OpenADC10
//              Turn module on | ouput in integer | trigger mode auto | enable autosample
#define PARAM1  ADC_FORMAT_INTG | ADC_CLK_AUTO | ADC_AUTO_SAMPLING_ON

// define setup parameters for OpenADC10
//              ADC ref external    | disable offset test    | disable scan mode | perform 2 samples | use dual buffers | use alternate mode
#define PARAM2  ADC_VREF_AVDD_AVSS | ADC_OFFSET_CAL_DISABLE | ADC_SCAN_OFF | ADC_SAMPLES_PER_INT_2 | ADC_ALT_BUF_ON | ADC_ALT_INPUT_ON

// define setup parameters for OpenADC10
//                use ADC internal clock | set sample time
#define PARAM3  ADC_CONV_CLK_INTERNAL_RC | ADC_SAMPLE_TIME_15


// define setup parameters for OpenADC10
//               set AN4 and AN5 as analog inputs
#define PARAM4  ENABLE_AN4_ANA | ENABLE_AN5_ANA


// define setup parameters for OpenADC10
// do not assign channels to scan
#define PARAM5  SKIP_SCAN_ALL

// use ground as neg ref for A | use AN4 for input A      |  use ground as neg ref for A | use AN5 for input B

// configure to sample AN4 & AN5
SetChanADC10( ADC_CH0_NEG_SAMPLEA_NVREF | ADC_CH0_POS_SAMPLEA_AN4 |  ADC_CH0_NEG_SAMPLEB_NVREF | ADC_CH0_POS_SAMPLEB_AN5); // configure to sample AN4
OpenADC10( PARAM1, PARAM2, PARAM3, PARAM4, PARAM5 ); // configure ADC using the parameters defined above

EnableADC10(); // Enable the ADC

while (1)
```

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.

4.5. 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.

5. Android Application Subsystem Report

5.1. 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.

5.2. App running on Android Device

5.2.1. 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.

5.2.2. 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.

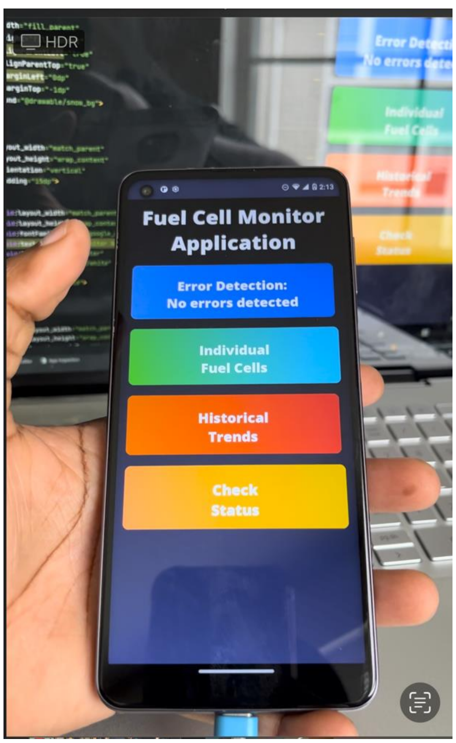

Figure 16: App running on a hardware device

5.3. App Connecting to Database

5.3.1. 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.

5.3.2. 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.

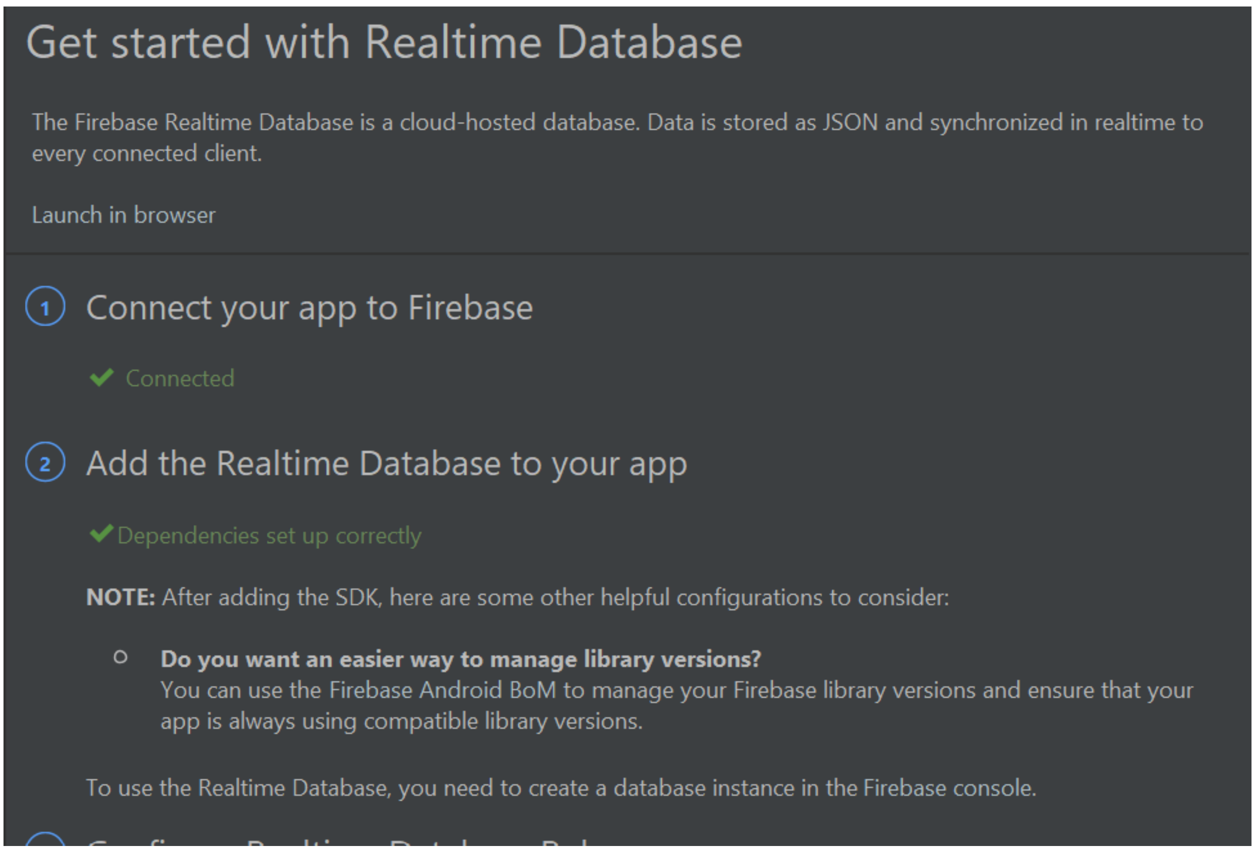

Figure 17: Connectivity between the app and database

5.4. Properly Display Accurate Data to Users

5.4.1. 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.

5.4.2. 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.

Figure 18: Status Graph

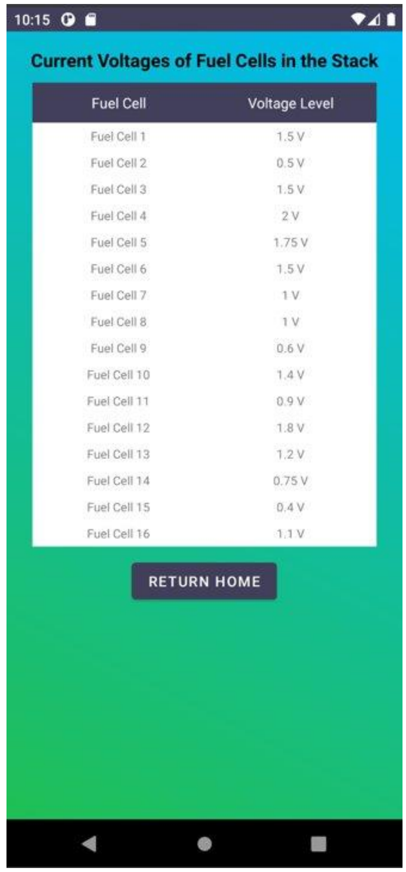

Figure 19: Fuel Cell Table

5.5. Alerts to Users Regarding Errors

5.5.1. 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.

5.5.2. Validation

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

Figure 20: Error Detection

5.6. Subsystem Diagnostic and Mitigation

5.6.1. 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.

5.6.2. 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.

5.7. 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.