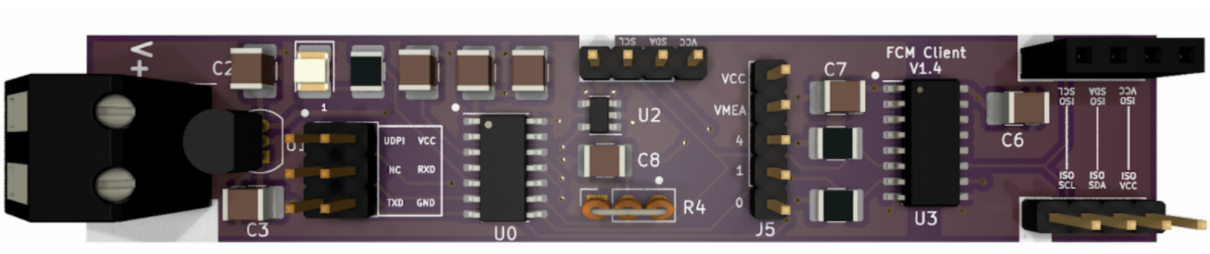
A multi-channel voltage monitoring system for solid oxide fuel cell stacks

Andrew Chen; Catherine Huang; Michelle Li; Dominic Shillingford

April 10, 2022



Project Sponsor: Casey Brown, P.Eng., FuelCell Energy

Project Number: 2213

Engineering Physics Project Laboratory

The University of British Columbia

List of Contents

List of Figures

List of Tables

1 Executive Summary

The purpose of this project is to design, test and implement a multi-channel cell voltage monitoring system for solid oxide fuel cells. Our sponsor is Casey Brown, Principal Engineer at FuelCell Energy in Calgary, AB working on solid oxide fuel cell R&D. Cell voltage monitoring is an integral part of designing and testing batteries because they allow an engineer or technician to monitor the health of the battery and ensure optimal working conditions over its lifetime. Our solution must be modular, provide voltage readings with ± 4 milliVolts of accuracy and be capable of handling high voltages of around 500 Volts.

Our team has designed and tested a solution that meets all of the above requirements. The sponsor is interested in channels which monitor a small sub-stack of fuel cells (fuel cell block). Each channel is measured with an individual PCB which we have designed and can be easily connected together and mounted. The individual PCBs are then connected to a host PCB which collects, processes and reports the data with Modbus via RS-485. Initially, the task was to measure a steady DC voltage from the fuel cell but we increased the scope to accommodate the fuel cell switching on and off load which presents as a square wave.

If power draw becomes an issue, we recommend using a switching voltage regulator in place of a linear regulator because switching regulators typically draw significantly less power. One important consideration in using a switching regulator is doing lab testing of the power supply rejection ratio which characterizes how well ripple at the input of the regulator is attenuated. This is because we power many components using the same fuel cells we are measuring and a steady DC power supply is important. This parameter is often not stated on the data sheet because it depends too much on the external passive components. Future directions for this project include adding temperature sensing capabilities, and improving the packaging, mounting and wiring of the project to make it look like a commercial product.

2 Introduction

Battery management systems (BMS) are an integral part of designing and testing batteries. In particular, the voltages of individual cells or groupings of cells are of great interest because they allow an engineer to monitor the health of individual cells and ensure optimal working conditions over its lifetime. The sponsor for this project is Casey Brown, Principal Engineer at FuelCell Energy working at their Calgary location where they focus on the R&D of solid oxide fuel cells (SOFCs). The rapid development of battery technology has partially been enabled by these battery management systems that can monitor individual cells in a large pack. State-of-the-art battery management systems typically monitor individual cells and report data to a common bus and most importantly, they can monitor very high voltages using low voltage devices. These solutions are not suitable for fuel cells because of the differences in individual and stack voltages compared to typical batteries. The solutions currently available for fuel cells are expensive, relatively inaccurate and the design is not optimized.

2.1 Terminology

- ADC - Analog-to-Digital Converter
- BMS - Battery Management Systems
- Fuel Cell - Single SOFC cell that operates at 1 - 1.5V
- Fuel Cell Block - A group of ten fuel cells that operates at 10 - 15V, and is the system of interest for measurement
- Fuel Cell Stack - A group of 350 fuel cells that operates at 350 - 525V.
- IC - Integrated Circuit

2.2 The Problem

Our sponsor has tasked our team with creating a modular fuel cell voltage monitoring system that mimics the architecture of state-of-the-art BMS, but adapts the specific functions to the operating conditions of the fuel cells that are being developed. Our goal is to design, test and implement a stack-able, modular, and cost effective voltage monitoring system for fuel cell blocks for an entire fuel cell stack. The design must have:

- The ability to handle 35 channels each measuring the voltage of a single fuel cell block (10V - 15V), totalling 525 V for the entire stack
- 12-bit resolution which translates to ± 4 mV of accuracy
- Data reported via a common bus
- At least a 1 Hz update rate
- The ability to accommodate a square wave voltage of 200-2000 Hz, with a minimum of 9.5 V and a maximum of 15 V

- Tested hardware
- A total cost of \$500 per 35 channel device, in quantities of 10

There is also a desire for the design to be flexible enough accommodate different needs, such as switching to five cells per channel instead of ten.

3 Discussion

The major technical challenges of this project can be classified into the following categories, measurement, data processing, isolation and data transfer. A subsection of this discussion will be devoted to each of these categories.

3.1 System Architecture

Before any detail about implementation can be discussed, the architecture of our approach must be understood. The key design parameter for this are the voltages ranges of the fuel cell stack, and the blocks that compose it. Each cell block has a maximum voltage of around 15 V, but as you put more and more of these in series, they begin to approach dangerous and difficult to measure voltages without having a floating ground.

3.1.1 Single-Ended ADCs

The most straightforward approach to measure our 35 blocks would have been to simply attach an ADC to each fuel cell block. This is inefficient in practice as each measurement module would need to contain the voltage attenuation hardware to allow for a conventional ADC to measure the entire range of 525V, unless various modules were designed for different voltage ranges. Either approach to this configuration would result in increases in cost and complexity, and overall was deemed to be inefficient.

3.1.2 Double-Ended ADCs

The easiest way to improve on the single-ended configuration is to take the ground reference of every ADC and connect it to the positive terminal of the fuel cell block below it in terms of voltage. This allows all the measurement modules to have the same design, and only need to be able to measure a range of 0-15V. The issue now is that, relative to earth, the measurement modules are at dangerously high voltages.

3.1.3 Isolated Double-Ended ADCs

To solve the issue of high voltages, some form of electrical isolation must be introduced. Either the analog signal between the fuel cell and ADC must be stepped down to a lower range (signal-side isolation), or the digital signal that the ADC outputs must be shifted to a safer range (data-side isolation). This isolation will allow the numerous measurement modules to all be connected to a control system that is safe for an engineer or technician to interface with.

3.1.4 Data Side vs. Signal Side Isolation

Both approaches were researched, and ways to achieve both were identified. Signal-side isolation was ultimately rejected despite having found an IC that meets all the physical design requirements. The ISO122U is an isolation amplifier which would have let us feed in our high voltage analog signals, and get the exact same signal back but with a lower ground reference. The issue is that each chip costs $48 even when purchasing multiples of ten. Although its possible to come up with a analog multiplexing circuit to attempt to feed multiple cell block readings through one of these chips, the added complexity to manage all those different voltage ranges while switching between them was decided to be more effort than it was worth. Without multiplexing this solution would have greatly exceeded our cost.

The final chosen configuration is a Data-Side Isolated Double-Ended ADC architecture. Further discussion about design describe how we implemented this configuration.

3.2 Measurement

As a voltage monitoring system, the fundamental goal of the system is to provide voltage measurements to the user. There are many ways to collect voltage measurements with trade-offs in terms of cost, bandwidth and sampling rate. The system proposed in this report uses an ATtiny microcontroller as an analog-to-digital converter which provides acceptable accuracy at a low cost while allowing for a large amount of flexibility in how voltage readings are packaged and processed. As a trade-off, it is difficult to achieve a very large sampling rate from the analog-to-digital converter on a microcontroller. In order to demonstrate these trade-offs in full context, it is important to present the theory behind the measurement of voltages.

3.2.1 Theory

The goal of any data acquisition system is to translate a time varying quantity from the real (analog) world into a stream of discretized, digital readings stored on a digital device. The overall quality of a data acquisition system is measured by a number of figures of merit including accuracy, resolution, bandwidth, sampling rate, and range. In the context of this voltage monitoring system, minimum specifications were provided by the project sponsor using these figures of merit and are shown in Table 1. A component of the project was therefore devoted to understanding these figures of merit themselves.

Range	0 volts to 20 volts
Accuracy	4 millivolts
Resolution	12 bits (4096 values)

Table 1: Project specifications.

The most straightforward of these figures of merits is resolution, which describes the total number of available readings for a data acquisition system. In general, resolution is required to ensure accuracy; for example, with only 20 possible readings in a range from 0 V to 20 V, it is impossible to achieve an accuracy of 0.25 V because the possible readings are spaced 1 V apart, but resolution is not a guarantee of accuracy.

The accuracy of a data acquisition system is a measure of how close the readings from a data acquisition system are to the true values being measured. There is some subtlety to this definition since a data acquisition system may more faithfully reproduce signals that vary slowly compared to those that vary quickly.

A more precise measure is the steady-state accuracy. When the input to the system is rapidly changed from one value to another (a step input) the running average of readings will converge on a value that is close to the true value of the input. The difference between this value and the true value is the steady-state accuracy.

After a long time, the average variation of readings from the running average of readings will also converge to a particular value which is the steady-state noise of the data acquisition system.

These steady-state measures of accuracy do not account for time. If it takes a very long time for the readings of a data acquisition system to approach the true value after a change, the data acquisition system will have no hope of accurately measuring a fast varying signal regardless of the steady-state accuracy. A useful measure of bandwidth of a data acquisition system is the time it takes for the readings from the system to get close (perhaps within 10%) of the steady-state accuracy. This is shown in Figure 1.

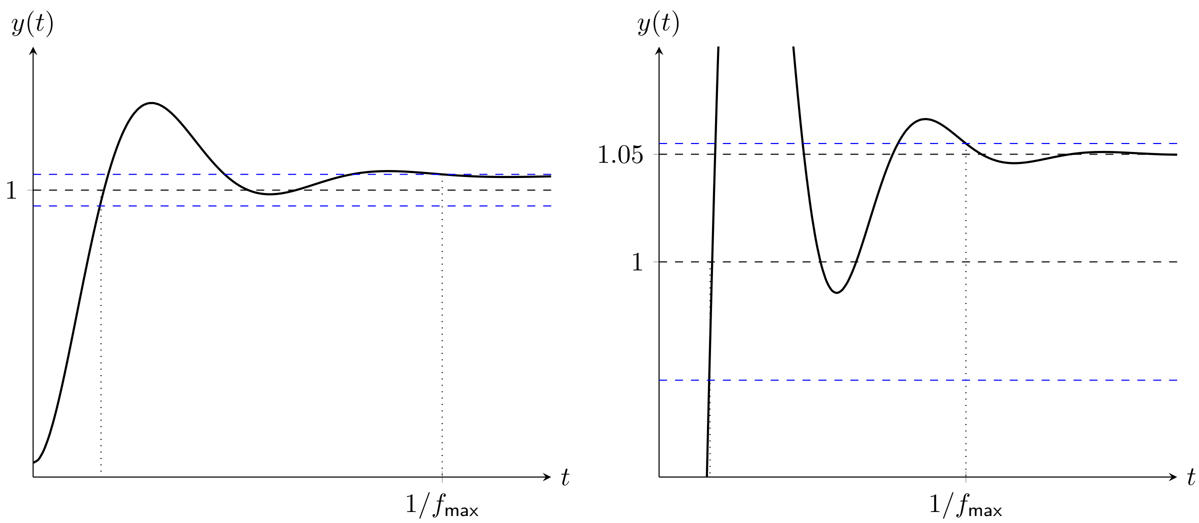


Figure 1: Settling time of a data acquisition system.

If the input to the data acquisition system has frequency components that vary quickly within this settling time, they will not be accurately reproduced and therefore the reciprocal of the settling time sets the bandwidth of the data acquisition system.

This bandwidth is also limited by the sampling rate of the data acquisition system. From the Nyquist Sampling Theorem, a data acquisition system cannot distinguish signal components with a frequency greater than half the sampling theorem, therefore the sampling rate of the data acquisition system must be sufficiently large so as not to limit the usable bandwidth of the system.

For the specific case of an analog-to-digital converter where the input quantity is a voltage difference, the precision is fixed by the architecture of the device and is measured in bits, e.g. an analog-to-digital converter may advertise twelve bits of precision corresponding to $2^{12} = 4096$ possible readings.

There are many sources of inaccuracy in an analog-to-digital converter as well some of which can be eliminated through calibration.

The most obvious source of error is the quantization error which is caused by the fact that the true voltage measurement must be quantized into one of the finitely many allowed readings for the device. If the input has sufficiently high noise however, the quantization error can be interpreted as additional noise since the input noise will cause readings to jump between multiple bins[1]. In this model, each addition bit of precision will cause the quantization noise to drop by 6 dB.

There are other sources of errors caused by variations in manufacturing of analog-to-digital converters. Two of these sources of error can be calibrated out, these are offset error and gain error. Offset error causes the reading for an analog-to-digital converter to have a constant offset compared the true voltage input across the entire range of possible inputs while gain error causes the reading of an analog-to-digital converter to vary from the true error in a way that is proportional to the input.

The last source of error that cannot be calibrated out is non-linearity which causes the reading of an analog-to-digital converter to vary in a way that is neither constant nor proportional to the input. Although it is not possible to calibrate this error out, analog-to-digital converter manufacturers will often provide a bound on how large these errors can be.

The bandwidth of an analog-to-digital converter is primarily limited by its internal capacitances and inductances, which increase the required settling time of the converter, and the sampling rate. For example, a large class of analog-to-digital converter will use a "sample and hold" circuit at their input to hold the input voltage steady over a single measurement period. These circuits use a capacitor to hold the input constant and therefore the settling time will be at least as long as the charging time of this capacitor.

3.2.2 Design

The above information about the characterization of an analog-to-digital converter was used to select a suitable analog-to-digital converter for this project. An analog-to-digital converter with at least twelve bits of precision must be used to meet the precision and accuracy requirements from the project sponsor.

From a bandwidth point-of-view, the expected input to the system is a square wave with a peak-to-peak amplitude of 5.5 V that is offset from ground by 12 V with a period greater than 500 μs and a duty cycle between 5 % and 95 %.

To accurately characterize this input in the worst case, a settling time much smaller than 25 μs is needed along with a sampling rate much greater than 40 kHz.

To meet these requirements while minimizing cost, the on-board analog-to-digital converter of a microcontroller from the tinyAVR 2 family is used in this project. With evolving analog-to-digital converter technologies, microcontrollers can offer a huge amount of flexibility and competitive accuracy compared to discrete analog-to-digital converters. As a trade-off, it is often difficult to get a large bandwidth from these devices. In this application where a high accuracy and a low cost are required with only a modest bandwidth required an on-board analog-to-digital converter is an obvious choice.

The microcontrollers in the tinyAVR 2 family are capable of sampling at a rate of 700 kHz and the bandwidth is limited by the sample-and-hold circuit which has a settling time of about 750 ns. Combining these measures, the

[1] In some applications, noise is added to the input to ensure that the output will in fact jump between multiple quantized readings even with a constant input, this is known as dithering.

microcontroller in this family have an effective maximum (theoretical) sampling rate of roughly 400 kHz.

These samples have twelve bits of precision which immediately satisfies the precision requirements for the project. Without calibration, the readings may have up to 0.75 % error however by using a calibration procedure to eliminate the offset and gain errors we can achieve 0.04 % error which corresponds to an 8 mV absolute error on a 20.48 V input.

3.2.3 Testing

Two main tests were done to verify ADC accuracy. To verify the 12 bits of resolution, which was interpreted as being accurate to 4 mV, was done by comparing the readings of a calibrated client module to a precision multimeter.

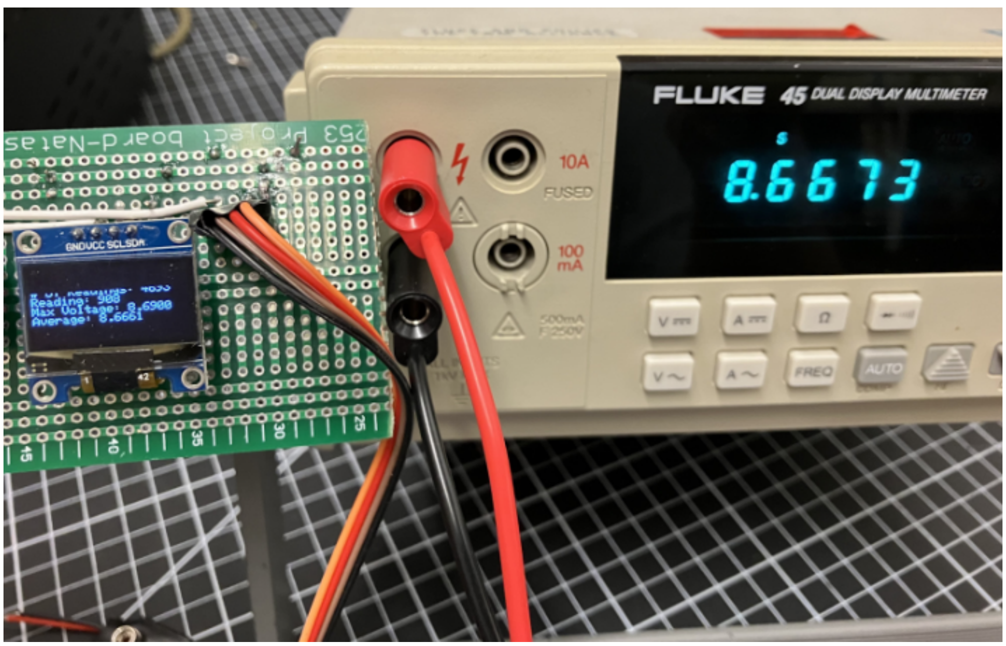


Figure 2: Left: Output of Client PCB readings Right: 5-Digit Multimeter

The tests clearly showed that our calibrated average readings were within 1 mV of the expected values. To improve on this test, data from both devices should be collected and an error value can be graphed to see any long time trends.

To better understand what the ADC was actually reading, a white-box test was done where the raw ADC readings were collected and graphed. The experimental setup used a high side switch and a voltage divider to generate a clean square-wave input at 2.046 kHz, with a peak-to-peak voltage of about 4V. Initial tests showed that we were seeing overshoot in our values, which would have affected our reported maximum values.

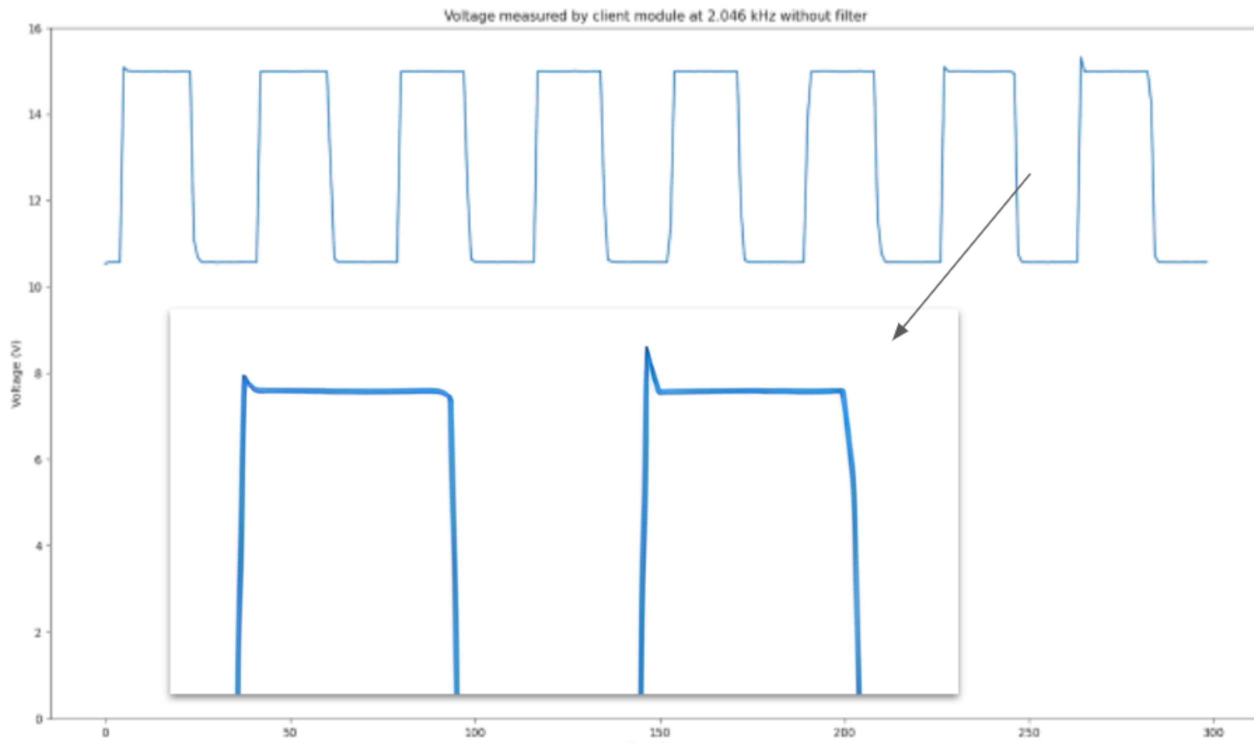


Figure 3: Raw ADC readings with 2.046 kHz input

To filter out this overshoot, a 11 nF capacitor was added between the output of the voltage divider and ground, which was found to be a good trade off between added slew rate and smoothing

3.3 Data Processing

3.4 Isolation

Isolation is required in our system because the host is at a different ground reference from each of our signals. Thus isolation is incorporated to bring all the signals to a common ground reference.

3.4.1 Theory

Data-side isolation was previously chosen over signal-side isolation due to cost constraints. The isolation method requires the system to be able to send data readings from the 35 ADC's to a host microcontroller. Three standards of communication were considered for implementing this solution: UART (Universal Asynchronous Receiver/Transmitter), SPI (Serial Peripheral Interface), and I^2C (Inter-Integrated Circuit). Other more specialized bus-communication protocols like CAN were not considered due to the additional hardware needed to support them, whereas the former three are all integrated into most micro controllers.

I^2C was chosen as the data communication protocol due to its inherent modularity which lets us add up to 128 clients to our data bus without much issue. UART is only able to accommodate 2 devices and while SPI is able to accommodate multiple devices, the wiring or firmware becomes more complex as all clients would need a Chip Select pin, or all data would need to be shifted through each client in order to reach the host.

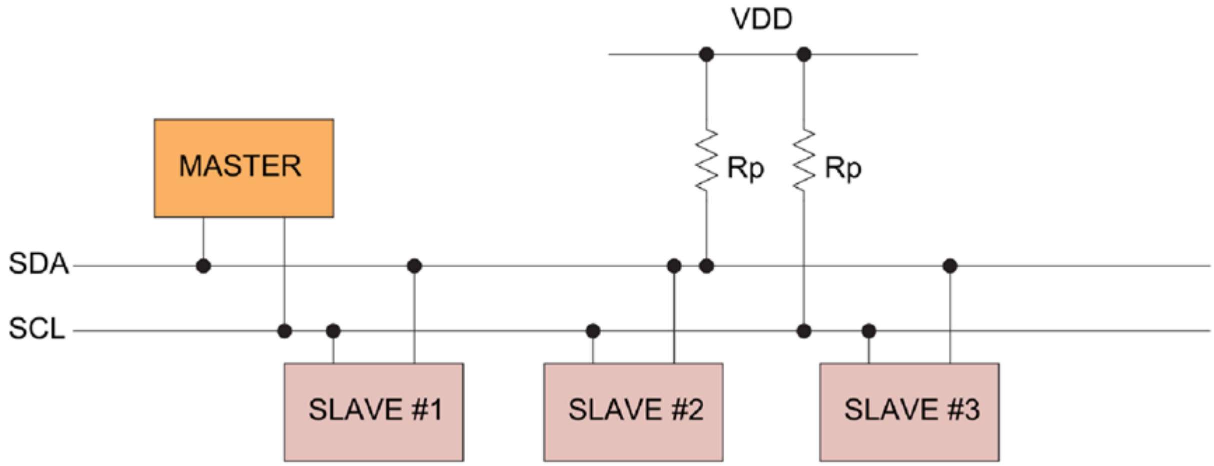


Figure 4: Basic I2C Block Wiring Diagram

At a high level, I^2C works in a one-to-many configuration, where one master device will communicate with many slave devices. This communication is done through two bi-directional lines, SDA (Data) and SCL (Clock) which are both pulled high by a pair of resistors. By pulling down these lines to ground, packets of serial data can be transmitted. Directed communication is achieved through the use of addresses assigned to all devices, and is used to prefix every message in order to ensure the correct device gets the correct message, despite all being on the same node. This feature is what lets us simply connect clients together, as long as they are each programmed with unique addresses.

3.4.2 Design

The MAX14850 chip was selected as the I^2C isolator chip. This was mainly due to chip availability as well as its simple implementation, as some other digital isolation chips require multiple to be used in a specific arrangement, while the MAX14850 abstracts that all away internally. Any IC that meets the voltage isolation requirements and can support I^2C communication at at least 100 kbit/s is expected to be suitable as well.

As I^2C communication needs unique addresses, each client needs to be programmed with a unique address. This is can be done during the initial programming steps, and can also be updated later.

3.4.3 Testing

To ensure that we could send data through our selected isolation chip. We connected the SDA and SCL pins of two Arduinos to the opposite sides of the chip and powered one side with a battery and the other side with a laptop.

Additionally, a high voltage test was conducted on our client board where the high side of the client board was powered by a 9V battery where the battery's ground was floated to 500 Volts, while the microcontroller on the low side was powered by a separate battery. An OLED display on was placed beside the microcontroller to allow us to display the data transmitted from our client board.

One potential issue that this solution was not tested for the full 35 client setup due to various constraints. On paper, it is expected that the capacitance of the 35 digital isolators will not exceed the ratings for the I2C protocol, but it is unlikely but possible that the leakage current exceeds that of which the SDA and SCL pins on the host Arduino Nano can source. If this is the case, a I2C repeater circuit can be introduced to one or all of the client PCBs. An potential IC for this is the P82B715. This would drive up the overall cost of the solution.

3.5 Data Transfer

Once the measurements are processed and sent to our host microcontroller, we need to report this data to a common bus to be displayed to a user. The objective is seamless integration, thus picking industry standard communication protocols and a simple microcontroller are priorities. As detailed in Section 3.2, we do not have the bandwidth to simply send every reading from the ADC, so we have chosen to compress the data into a data frame (or packet) containing what we and the sponsor consider to be the most important information. Namely,

- Channel ID (the address of the client)
- Minimum and maximum readings
- Number of readings
- Sum of readings
- A histogram of how many readings fall into bins of a specified voltage range

Details of this data frame and how to process them are found in Appendix C. Using these pieces of information, the technician or engineer will be able to better characterize the voltage of the fuel cell block.

3.5.1 Approach

We will be using an Arduino Nano as the host microcontroller as it is cheap, easy to program and has strong community support for troubleshooting. The code for the Arduino can be found in this github repository: `https://github.com/fcefcms/host-firmware`

Data Type	Absolute Addresses	Data Address	Access
Discrete Coil	0-9999	0000-270E	Read-Write
Discrete Input	10001-19999	0000-270E	Read Only
Analog Input Registers	30001-39999	0000-270E	Read Only
Holding Registers	40001-49999	0000-270E	Read-Write

Table 2: Modbus Registers

We will be using the Modbus protocol as it is open source and used commonly for industrial settings. Modbus is a data communication protocol that is often transmitted over serial lines, and follows a master-slave architecture. Data is stored in the slave device in four different tables and the master device will poll the slave device to read and write from its registers. The first two tables store 1's and 0's and the input and holding registers are 16-bits. More details about the Modbus tables are shown in Table 2. In general, the Modbus registers are accessed not by the absolute address but by the data type and then the data address of the register. For the physical layer, we will be using RS-485 which is very commonly used in conjunction with Modbus. It transmits data over a twisted pair cable.

In this project, our Arduino acts as the Modbus slave and writes the data packet from the clients to its holding registers. The Modbus master is an external device which will poll the Arduino, like a computer running LabVIEW which an engineer or technician will interface with. LabVIEW has an easy to use Modbus library to allow easy data acquisition. An example LabVIEW test bench we set up and used for testing can be found in the github repository: `https://github.com/fcefcms/modbus-labview-test/blob/main/MODBUS%20Test/ModbusLabviewTest.vi`.

3.6 Client PCB Design

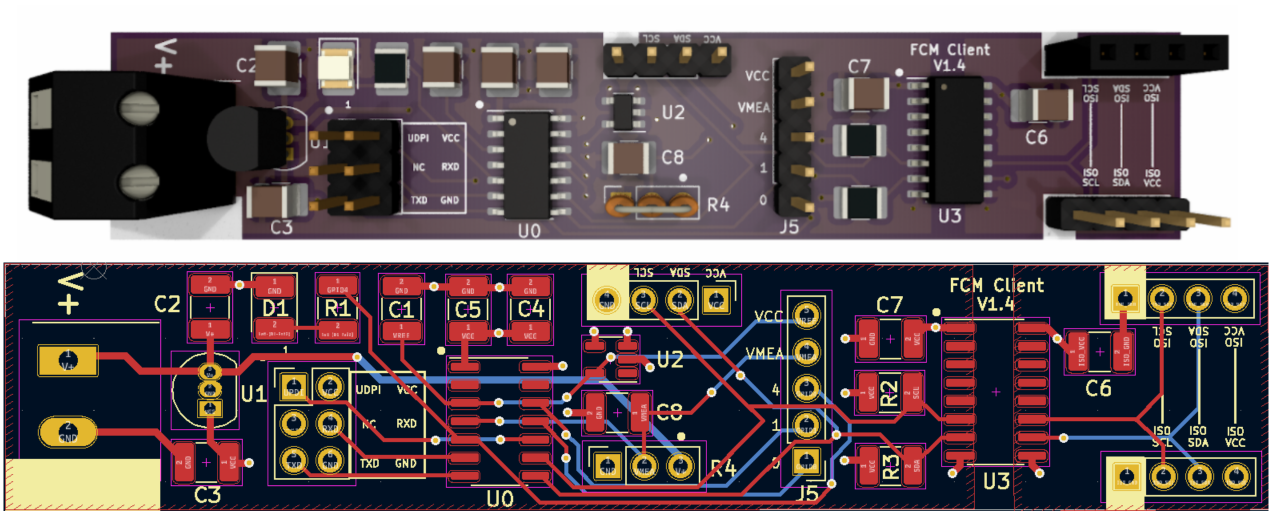


Figure 5: Top: KiCAD Client PCB 3D Preview Bottom: KiCAD PCB Layout

A 4-layer PCB was designed to interface with the fuel cell blocks, perform the necessary attenuation to accurately measure the voltages, process the raw values, and transmit the data to a common bus (I2C). The KiCAD project files can be found in this github repository: https://github.com/fcefcms/client-pcb/tree/main. The components used and links to purchase them can be found here: [add link to google sheet]

As there will be many client PCBs, the physical layout was minimized to save space and to match the existing solution, which made routing more challenging than usual. Four layers were used to allow for easier routing of signal lines by using the inner two layers as a power and ground plane.

A common screw terminal was chosen as an interface to allow FCE to plug in their probes. This can be easily replaced in the future to suit whatever connectors will be used. The voltage input must then be attenuated.

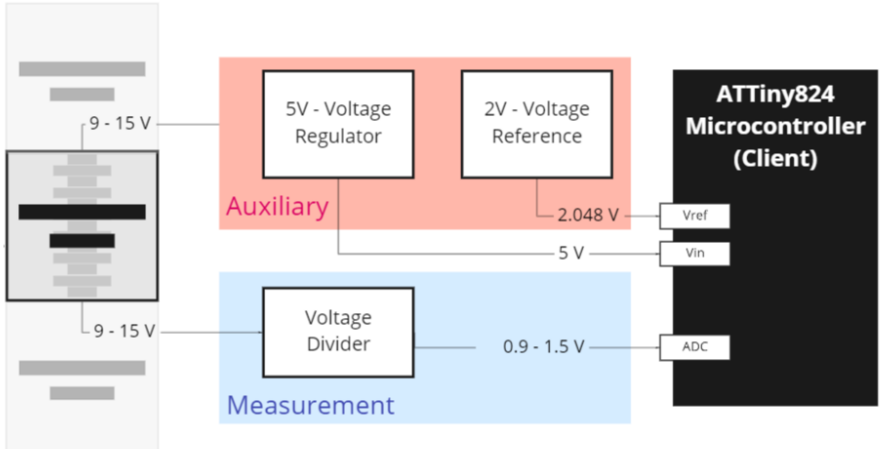


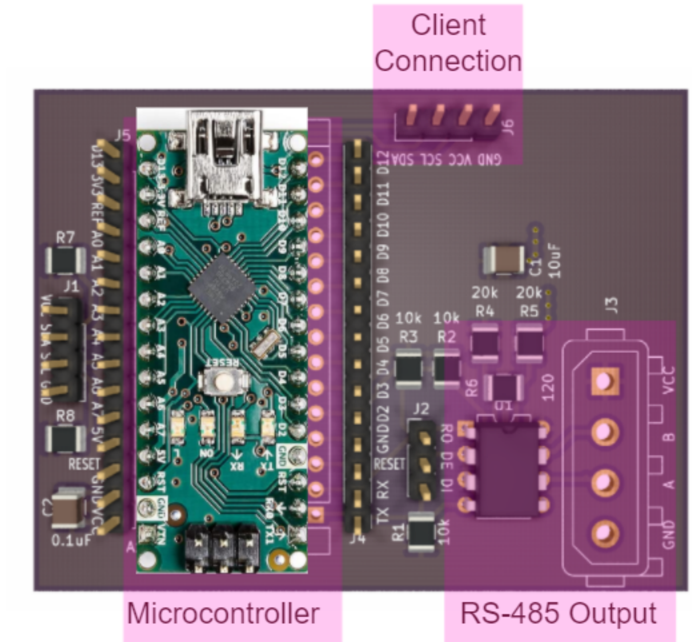
Figure 6: Block Diagram of PCB Voltage Measurement Circuit.

We can easily drop our 15 V input to a 1.5V value measurable by our ADC by using a voltage divider. The 15 V input is also fed into a 5 V linear regulator to be used as power for the ATtiny 824 microcontroller, the isolator, and the voltage reference. This voltage reference is read by the microcontroller is used to improve the accuracy of the integrated ADC. After the fuel cell voltage is read and processed by the microcontroller, the data can then be passed onto an I2C bus through the digital isolator, which also acts as a way to manage creepage and clearance of the PCB.

As the linear regulator is fairly inefficient and produces excess heat, the more temperature sensitive components like the voltage divider and reference were moved away from the regulator.

Debug pins are provided for most outputs, with only GPIO 2 and 3 inaccessible by normal means. The silkscreen provides labels for all pins as well as orientation for certain chips. The white squares indicate ground.

3.7 Host PCB Design



A 2-layer PCB was designed to collect the data from all the client PCB's and send the information over to LabVIEW via Modbus protocol. The KiCAD project files can be found in this GitHub repository: `https://github.com/fcefcms/HostPCB`

The board consists of an Arduino Nano as the main controller as explained in section 3.5.1

Surface mount resistors and capacitors with package of 1210 are used. The packaging was chosen to allow us to assemble the board more easily but they can be changed in future iterations to save space on the board.

A common screw terminal is used to connect the input power to the microcontroller and the RS-485 output. Header pins are used for connecting the I2C lines to the client PCBs. Both of these connectors can be easily changed in the future iterations to suit whichever connectors will be used.

3.7.1 Mounting and Wiring

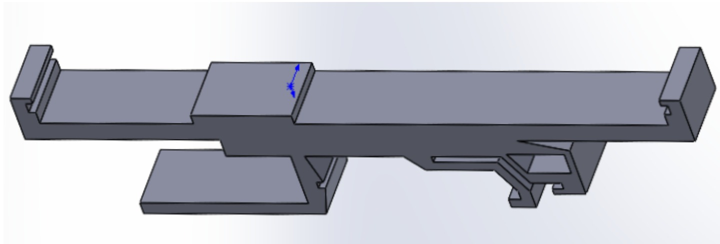


Figure 7: SolidWorks model of basic DIN rail mount for client PCB.

A basic mount was designed to allow for each client PCB to be mounted to a 35x7.5mm DIN rail. It allows the PCBs to be snapped in and then for the entire assembly to be snapped onto the rail. The current mount exposes all the electronics, and it is recommended that the final version would cover the high side components. A commercial solution from OKW RAILTEC was initially found, but integration was left as future work. It is expected that the existing PCB would need modification to size, terminal type, and location to use the commercial solution. The specific product can be found here: `https://www.okwenclosures.com/en/Railtec-C/B6700106.htm?var=633e67e0-d9ad-11e2-99c1-0050568225d7`

To wire all the Client PCBs to the host, insulation-displacement connectors (IDCs) were used with a 22-AWG flat ribbon cable with four conductors. The IDCs listed in the component breakdown are not the preferred solution, as they have a closed end and require slight modification to allow the cable to pass through. The intention is that using open-ended IDCs with a long ribbon cable would be the easiest way to connect many of these together in order for communication.

An example of a fully integrated system is shown below.

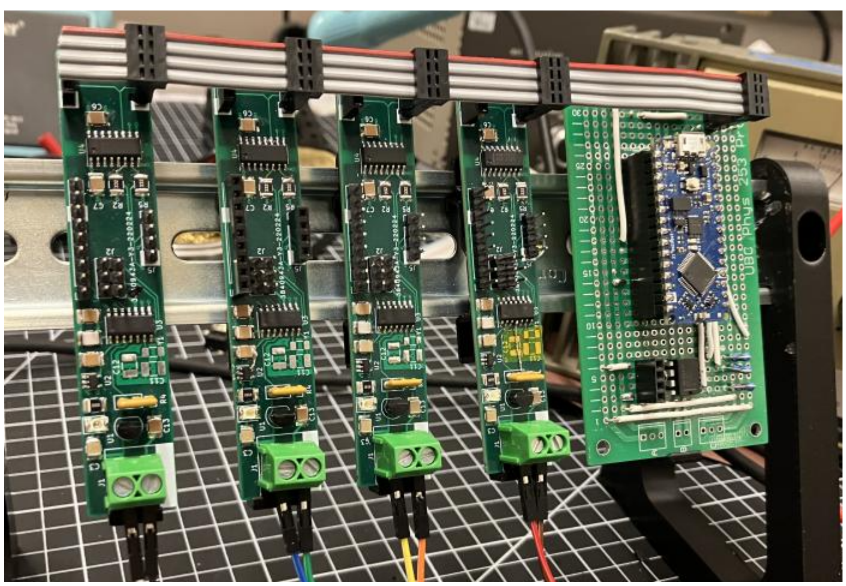


Figure 8: Test rig of four clients connected to a host and mounted onto a 35x7.5mm DIN Rail

4 Conclusions

Our team set out to design, test and build a cell voltage monitoring system for use in the R&D of solid oxide fuel cells. We created a series of discrete client modules that each measure the voltage of a single fuel cell block which has an upper voltage of 15V. By using a combination of calibration and on-board references, we are able to achieve 12-bit accuracy using the on-board ADC of an ATtiny824 microcontroller. Our solution over-samples and processes these values into a single informative data frame and is able to report these values over a common I^2C bus to a single host module, capable of accommodating up to 128 clients. The host module is finally able to present the readings to a LabVIEW user interface using the Modbus protocol over RS-485. In quantities of ten devices (350 clients and 10 hosts), our solution achieves a cost of $607.03 per device.

4.1 Recommendations

A potential problem remains with the amount of power drawn from the fuel cell. If it becomes a significant issue, our recommendation is to use a switching voltage regulator in place of the linear voltage regulator on each client PCB as that appears to be a main source of power draw. Future improvements may include a temperature sensor, current measuring and better packaging of the product to make it look more like a commercial product.

5 Deliverables

1. Gerber files and BOM for client and host PCBs
2. Detailed cost breakdown spreadsheet
3. Code base for client (ATtiny) and host (Arduino Nano) microcontrollers
4. User manuals for debugging, calibration and setup (See Appendix A and Appendix B)
5. Sample LabVIEW test bench

Appendices

Appendix A: Serial Debug Commands

- `ram`: Returns the available dynamic memory on the device
- `read`: Return the calibrated voltage at the device's input (multiply by 5 millivolts for a physical reading)
- `rread`: Return the uncalibrated voltage at the device's input (multiply by 10 millivolts for a physical reading)
- `addr [address]`: Set the I2C address of the device (default 0x55 when flashed)
- `getaddr`: Returns the current I2C address of the device
- `setl [current voltage in millivolts]`: Sets the lower calibration point of the device
- `seth [current voltage in millivolts]`: Sets the upper calibration point of the device
- `getl`: Gets the lower calibration point of the device
- `geth`: Gets the upper calibration point of the device

Appendix B: Calibration Procedure

1. Gather a voltage supply capable of outputting voltage both above and below the desired measurement voltages, e.g. the on and off bus voltages of a solid oxide fuel cell sub-stack, a serial terminal interface (such as a computer and a TTY-to-USB device) and a millivolt accurate multimeter.
2. Access the serial terminal of the device (9600 baud per second) through the UPDI header (see Figure 9). For orientation, the UPDI pin is labeled by a black mark on the PCB. Ensure the lines are connected RX to TX and TX to RX. Verify serial connectivity by running a `ram` command.
3. Connect a voltage into the screw terminals of the device that is below (but nearby) the minimum desired measurement voltage (perhaps 8 volts if measuring a range between 9.5 volts and 15 volts).
4. Measure the input voltage to millivolt accuracy using the multimeter.
5. Execute a `setl` command with the measured input voltage (in millivolts).
6. Verify the calibration using a `getl` command.
7. Connect a voltage into the screw terminals of the device that is above (but nearby) the maximum desired measurement voltage (perhaps 17 volts if measuring a range between 9.5 volts and 15 volts).
8. Measure the input voltage to millivolt accuracy using the multimeter.
9. Execute a `seth` command with the measured input voltage (in millivolts).
10. Verify the calibration using a `geth` command.
11. The calibration values are stored in EEPROM and will persist even after a power cycle.

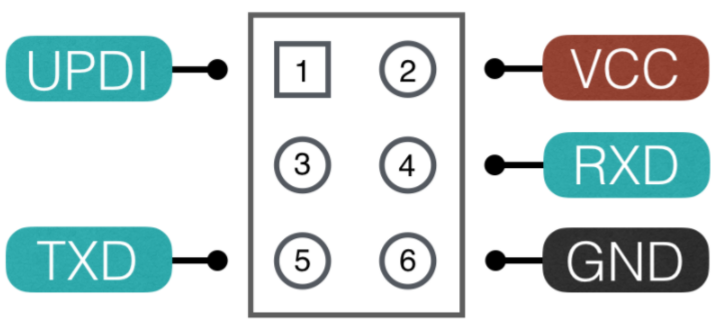


Figure 9: Orientation of debug header. The screw terminal input of the device will provide power to the device and therefore if a voltage is put across the input screw terminals, the VCC pin must be left unconnected.

Appendix C: Processing the data frame

The data frame that will be sent via Modbus is shown in Figure 10. Modbus holding registers are 16 bits so there are a total of 30 registers used. Holding registers at data address 2, 3 and 6-9 inclusive are the minimum reading, maximum reading and sum of readings, respectively. They are given in the registers in units of 5 mV, therefore they all need to be divided by 200 to get back to volt units. The 10 histogram bins each have a 2 V range (ie. First bin: 0V - 2V, Second bin: 2V - 4V and so on) up to 20 V.

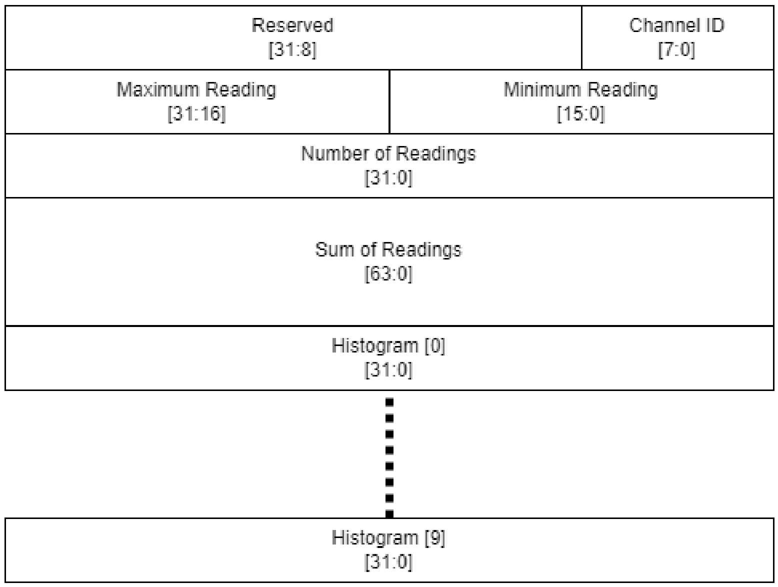


Figure 10: Data frame