

# Mini-DAQ Final Report

Bret Bosma, Ryan Coe, Giorgio Bacelli, Ted Brekken, Budi Gunawan

6/29/2021

## ABSTRACT

Testing of model-scale marine hydrokinetic (MHK) devices present a challenge on many fronts including data acquisition and control. Available data acquisition systems are often expensive, heavy, and difficult to integrate into a small-scale device. Especially at the proof-of-concept stage, an easily configurable, open source, lightweight, inexpensive data acquisition solution would accelerate development time to testing and provide a valuable resource for the industry. To this end, a Mini-DAQ data acquisition is being developed and is outlined in this document. Two functioning systems have been developed and tested and detailed comparison with an industry standard high quality data acquisition system has been completed. Results show a competent system with favorable characteristics considering the cost and footprint. The Mini-DAQ shows promise as a low-cost very capable system which could favorably impact barriers to small scale device testing for the industry.

# CONTENTS

<b>ABSTRACT .....</b>	<b>2</b>
1. Introduction .....	4
2. System Architecture.....	5
3. Hardware .....	7
3.1 Single board computer.....	7
3.2 EtherCAT Shield.....	7
3.3 EtherCAT Primary Device .....	8
3.4 Analog to Digital Converter.....	9
3.5 RS232 Communication.....	9
3.6 Inertial Measurement Unit .....	10
3.7 4-20mA Current Sensor.....	10
3.8 Pressure Transducer .....	11
3.9 Encoder Transducer .....	12
4. Software.....	13
4.1 Arduino IDE .....	13
4.2 Easy Configurator .....	14
4.3 TwinCAT .....	15
4.4 MATLAB/Simulink .....	16
5. MiniDAQ-V1 .....	16
5.1 System Requirements .....	17
5.2 Bill of Materials .....	17
5.3 Mini-DAQ VS HWRL-DAQ Comparison.....	18
5.3.1 HWRL-DAQ specification.....	19
5.3.2 Voltage Standard Input .....	19
5.3.3 Precision Time Protocol .....	21
5.3.4 LED signal .....	21
5.3.5 Sine wave .....	22
5.3.6 White Noise.....	24
6. Demo WEC .....	27
7. Future Work.....	28
References .....	29

# 1. INTRODUCTION

Model-scale testing of marine hydrokinetic (MHK) devices presents a challenging problem. Firstly, test-devices must accurately capture the intended rigid-body mass and moments of inertia. Available data acquisition systems are often expensive, heavy, and challenging to locate on a small model-scale device. These systems can also be challenging to waterproof, as they often dissipate large amounts of heat which will be trapped inside a waterproof enclosure. Alternatively, the data acquisition (DAQ) system can be located at some nearby platform and signals may be run over a tether. This solution is problematic due to the high electro- magnetic interference (EMI) environment created by most wave energy converters (WECs). The motors used to represent power take-offs (PTO) emit large amounts of EMI, which can interfere with and corrupt measurement signals.

This project will design and construct a small data-acquisition system suitable for small-scale MHK device testing. This mini-DAQ is based on an EtherCAT networking architecture, which, unlike Ethernet (e.g., UDP) networks, is deterministic thus enabling high-performance real-time feedback and control. This type of architecture has already been utilized for testing conducted by the Advanced WEC Dynamics and Controls project, where it was successfully providing the signal quality and sampling rates necessary (in sufficiently small form factors) to execute feedback control. By leveraging this existing experience and hardware for the Advanced WEC Dynamics and Controls project, the proposed effort can be extended to not only develop a design for a Mini-DAQ, but also assemble a prototype to validate performance. The mini-DAQ designed in this project will serve as a reference design/template for end-users to replicate and/or apply variations specific to their own systems.

In addition to utilizing knowledge from DOE-funded MHK testing, Sandia and national laboratory partners will leverage extensive advanced research from wind energy and defense programs on DAQ and EMI to provide best practices guidance and case-studies. Available off the shelf (OTS) components will be leveraged to design an example mini-DAQ system, which utilizes best practices to minimize EMI. This will result in a DAQ system suited to both small-scale WEC testing (i.e., small, lightweight, limited cabling to interfere with dynamics) that can also be adapted to support large scale testing (distribution acquisition, industrial components).

Current DAQ systems often utilize custom hardware and software, requiring its user to have high skills in coding and system setup. With the increasing interest in renewable energy, there are also small corporations, university research departments, or independent companies that additionally require DAQ systems, but on a smaller scale. The purpose of this project is to produce an easy-to-use EtherCAT slave DAQ system alternative, aimed towards new developers in the wave energy field. This report serves to outline the progress made thus far in the Mini-Data Acquisition (mini-DAQ) project funded by Sandia National Laboratories.

## 2. SYSTEM ARCHITECTURE

Fundamentally, the Mini-DAQ consists of a small form factor real-time capable computing system that is capable of a range of inputs and outputs and is compatible with an EtherCAT [1] network. This allows for a small form-factor package that can be integrated into an MHK device, while minimal cabling is routed to shore. Most MHK concepts have unique requirements for data acquisition, employing a range of sensors useful in both operation and monitoring of the system. In this study, commonly used sensors in the industry are demonstrated, however each system will likely be unique and need to be customized.

The heart of the Mini-DAQ system is the Arduino Mega 2560 [2]. While many other small board computers could be used for this task, the Arduino Mega 2560 has a specification that allows for a broad range of input and output capabilities and is relatively easy to use. Arduino provides an open-source hardware and software solution with a rich user community and a short learning curve to effective use. See section X for a more detailed look at the Arduino and its use in the Mini-DAQ.

Paired with the Arduino Mega 2560 is the EasyCAT Shield for Arduino [3] by AB&T. This allows the Arduino board to become an EtherCAT device. The EasyCAT Shield conveniently plugs directly onto the Arduino creating a compact package while providing a pass-through for all of the Arduino pins for use as necessary. The EasyCAT communicates with the Arduino through a serial peripheral interface (SPI) and with the EtherCAT network through two RJ45 connections. The EtherCAT network allows for communication with the primary EtherCAT device where control and data recording can be implemented. See 3.2 EtherCAT Shield for a more detailed look at the EasyCAT module.

Input and output peripheral devices are numerous and specific to each device however several commonly used peripherals are detailed in this study. An analog to digital converter communicating via SPI used to interface with a load cell will be detailed. A quadrature encoder communicating through serial communication and an Arduino Nano will be shown. An IMU device communicating via RS-232 will be demonstrated.

A high-level look at the Mini-DAQ system shows an EtherCAT primary system sitting on shore. An ethernet cable and power cable then run from shore to the Mini-DAQ system on the MHK device consisting of the Arduino Mega 2560 and EasyCAT shield. Other EtherCAT enabled items on the network can be attached, such as a motor/generator drive. The Mini-DAQ then communicates with the desired peripherals via the appropriate communications protocol. This outline is shown in graphical form in Figure 1.

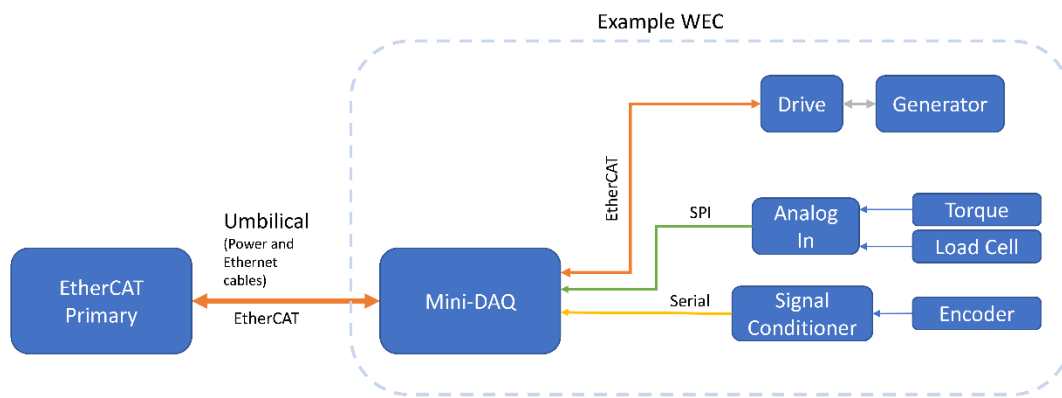


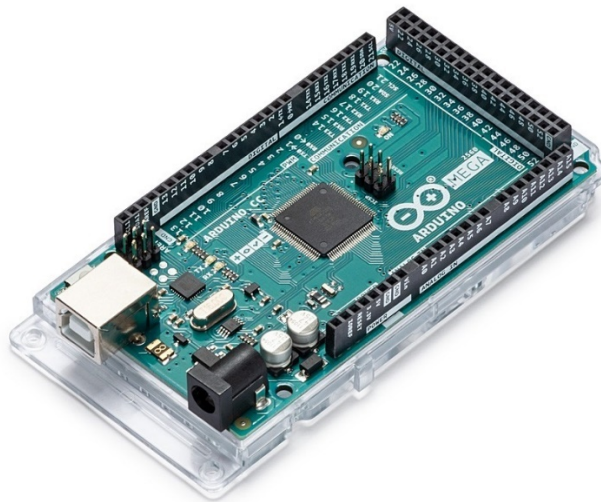
Figure 1: Example system architecture showing major components and their relation.

## 3. HARDWARE

### 3.1 SINGLE BOARD COMPUTER

Single board computers have become extremely popular in recent years due to their drop in cost, relative ease of use and common availability. Prior to their availability, a user would need to buy individual components including a microprocessor, memory, and interface units and either use a prototype board or generate a printed circuit board custom for their project.

The Arduino Mega 2560 Rev3 [2] – Development Board, shown in Figure 2 is popular among hobbyists, offering a large resource community and relatively easy setup and coding language. Based on the ATmega2560 microcontroller, it has 54 digital input/output pins, 16 analog inputs, and 4 hardware serial ports. It can be powered and programmed via a USB cable, or powered by an external power source. It is recommended for the Mini-DAQ that the system is programmed via USB, then run via power supply once program is established. The input voltage recommended to supply the Arduino is 7-12V DC. The Arduino provides protection such that both USB and external power source can be connected at the same time. The Arduino can be programmed with the Arduino IDE [4].



*Figure 2: Arduino Mega 2560 Rev3 single board computer*

### 3.2 EASYCAT SHIELD

To make the Arduino single board computer EtherCAT compatible a shield is necessary. The EasyCAT Shield for Arduino [3] is shown in Figure 3. This board handles the EtherCAT communication between the Arduino board and the rest of the EtherCAT network. It interfaces with the Arduino via SPI communication and provides two ethernet connections for input and output of the EtherCAT network.

By default the EasyCAT is configured with 32 bytes input and 32 bytes output. It is possible to use the EasyCAT in this manner, adjusting all signals to comply with this data type. However, if different data types or number of data are desired, programming of the EasyCAT is required. This is done using the Easy Configurator software tool [3].



*Figure 3: EasyCAT Shield for Arduino*

### 3.3 ETHERCAT PRIMARY DEVICE

All EtherCAT networks need to have a primary device in the network. For this project a Speedgoat real-time target machine [5] fills the role of this primary device shown in . This has several advantages including access to the tools possible within the MATLAB/Simulink environment. Data visualization, data logging, and control implementation are a few of the advantages to working in this environment. Beyond the basic MATLAB/Simulink software, the Simulink Real-Time toolbox [6] is needed. The performance line machine was used however a baseline line machine would be sufficient.





Figure 4: Speedgoat real-time target machines

### 3.4 ANALOG TO DIGITAL CONVERTER

The Arduino Mega 2560 does have 16 analog inputs however they only provide 10 bits of resolution and a range of 0-5 V which is too restrictive for most applications. For MHK devices a specification of 16 bit and  $\pm 10$  V was chosen. The DC682 development board by Linear Technology [7] was chosen to fit this need as show in Figure 5. This is a demo board for the LTC1859 8-Channel, 16-Bit, 100ksps A/D converter [8]. The board can be configured for single ended or differential inputs with voltage ranges of 0-5 V, 0-10 V,  $\pm 5$  V, or  $\pm 10$  V.

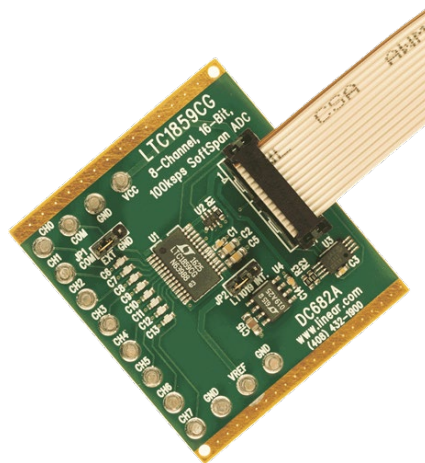


Figure 5: DC682 16 bit A/D converter

### 3.5 RS232 COMMUNICATION

The MAX232 from Maxim Integrated Products [9] is used for RS232 communication is shown in Figure 6. This allows communication with any RS232 compatible device and in this project was use for the inertial measurement unit (IMU) communication as shown in the next section.

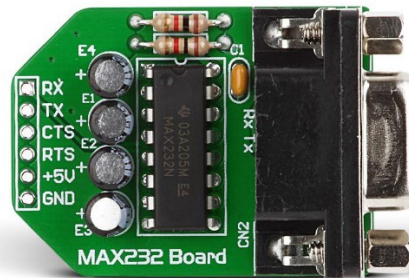


Figure 6: RS232 communications using the MAX232 board

### 3.6 INERTIAL MEASUREMENT UNIT

The inertial measurement unit used in this project is the MTi-20 VRU by xsens [10] as shown in Figure 7. This communicates via RS232 through the MAX232 board as outlined in the previous section. Output is translational accelerations and rotational euler angles used to estimate six degrees of freedom motion.

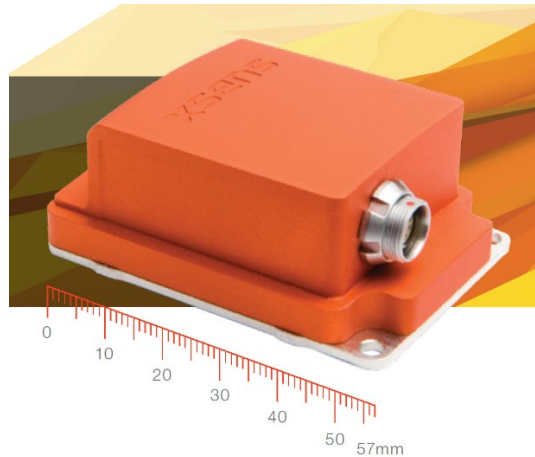


Figure 7: MTi-20 VRU by xsens

### 3.7 4-20MA CURRENT SENSOR

Typical instrumentation in industrial applications operate on a 4-20 mA current output due to its relative noise immunity. The NCD 2-Channel 4-20 mA Current Loop Receiver 16-Bit ADS1115 I2C Mini Module [11] was used for measuring sensors with this output and is shown in Figure 8. The pressure sensor chosen for the Mini-DAQ uses the 4-20 mA output. Details of the pressure sensor used will be shown in the next section.

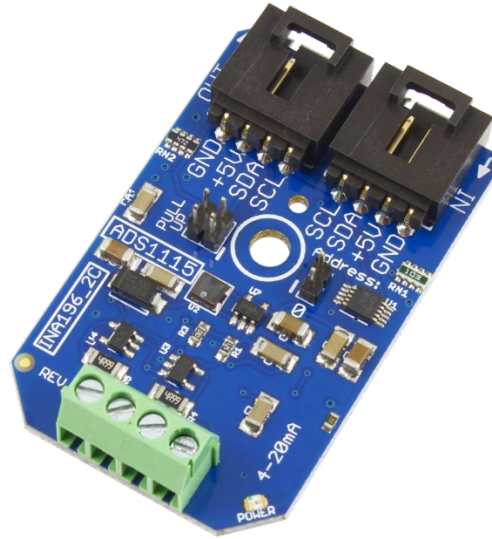


Figure 8: 2-Channel 4-20 mA Current Loop Receiver 16-Bit ADS1115 I2C Mini Module

### 3.8 PRESSURE TRANSDUCER

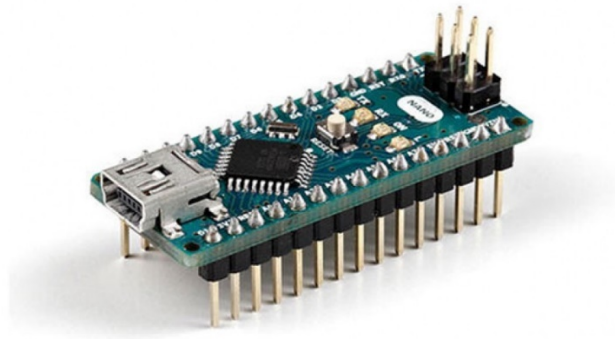
The TD1200 from TRANSDUCERS direct [12] was used in this project as shown in Figure 9. With a 4-20 mA output signal the current transducer from the previous section was used.



*Figure 9: Transducer Direct TD1200 pressure sensor*

### 3.9 ENCODER TRANSDUCER

Quadrature Encoders are typical low cost devices for measuring position in a MRE system. These typically output in two digital channels where interpretation of the results is needed. Although the main Mini-DAQ Arduino system could handle this task, the computational requirements could interfere with the operation of the system. Therefore, a second Arduino Nano [13], seen in Figure 10, can be employed to read the encoder signals, interpret them and communicate via serial communication with the main Arduino system.

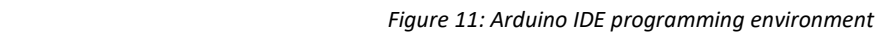


*Figure 10: Arduino Nano used for reading quadrature encoder signals*

## 4. SOFTWARE

### 4.1 ARDUINO IDE

The Arduino IDE can be used to program both the Arduino MEGA 2560 and the Arduino Nano and provides an intuitive, well supported, development environment [4]. Figure 11 shows a screenshot of the development environment for the Arduino. Example code is given in Appendix A.



The Easy Configurator [3] allows for customization of the EasyCAT Arduino with the specific channels intended for communication on the EtherCAT network. By default, the EasyCAT system comes with 32 bytes input and 32 bytes output. There is no problem with using it in this configuration if all data can be transformed into a form to fit this data type. However, if it is desired to transmit on the EtherCAT network different data types and customize the names of the data the Easy Configurator is needed. An example of Easy Configurator is shown in Figure 12.

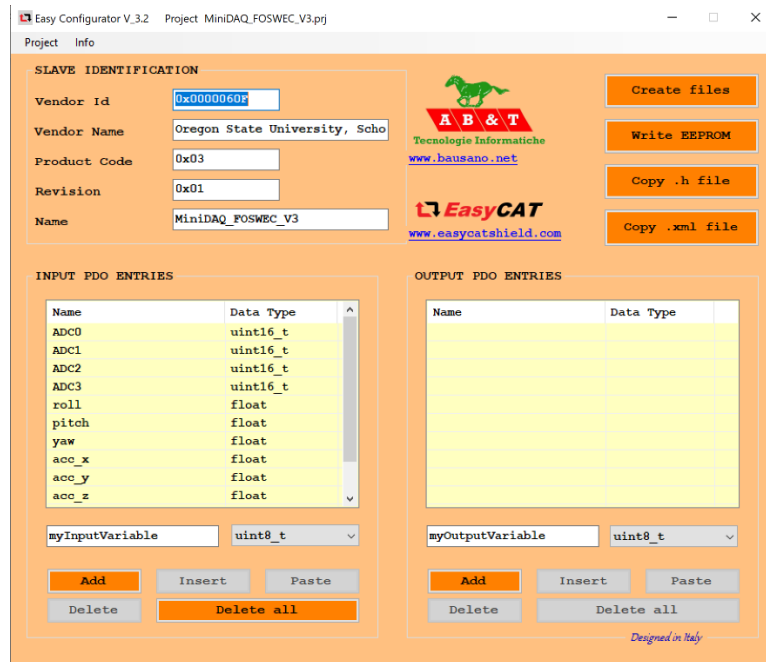


Figure 12: EasyCAT Configurator used for customizing data types and channels

## 4.3 TWINCAT

When establishing an EtherCAT network it is necessary to generate xml files that define the network and allow the main EtherCAT device to manage the network. This is a one time operation for each configuration of a EtherCAT network and is done using the TwinCAT automation software [14]. TwinCAT is a plugin for Microsoft Visual Studio [15] and allows for definition and debugging of the EtherCAT network. A screenshot from the TwinCAT environment is shown in Figure 13.

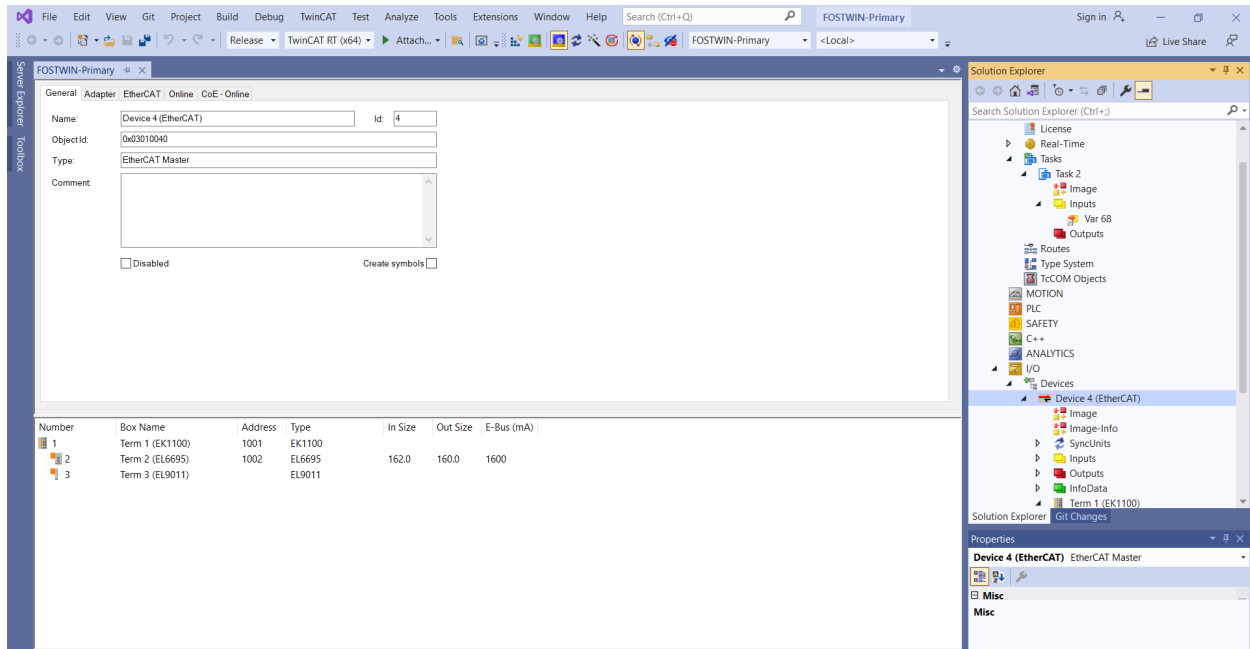


Figure 13: TwinCAT3 environment for setting up EtherCAT network

## 4.4 MATLAB/SIMULINK

In the current Mini-DAQ concept, a Speedgoat system performs the task as the main EtherCAT device. The Speedgoat system uses the MATLAB/Simulink environment and manages the visualization and recording of the data collected by the Mini-DAQ.

## 5. MINIDAQ-V1

The MiniDAQ-V1 is the first iteration of the Mini-DAQ created. Figure 14 shows the system with no sensors attached.





Figure 14: MiniDAQ-V1

## 5.1 SYSTEM REQUIREMENTS

The Mini-DAQ-V1 has the following as system requirements

- 100 Hz sampling period
- Analog to Digital Converter
  - 16 bit
  - Differential input
- 4-20 mA input
  - 16 bit
  - Used for pressure sensor input
- RS-232 capability
  - Used for IMU input

## 5.2 BILL OF MATERIALS

Bill of materials for Mini-DAQ\_V1 is in Figure 15.

## Mini-DAQ\_V1

Assembly Name :	MINI-DAQ FOSWEC
Assembly Number :	1
Assembly Revision :	1
Approval Date :	April 16, 2021
Part Count :	14
Total Cost :	\$249.20



Part #	Part Name	Description	Qty	Mass (g)	Picture	Supplier	Unit Cost	Cost
1	<a href="#">Arduino Mega</a>	ARDUINO MEGA2560 ATMEGA2560	1	51		Digi-Key	\$ 38.50	\$ 38.50
2	<a href="#">EasyCAT</a>	EasyCAT Shield for Arduino	1	32.5		bausano.net	\$ 56.84	\$ 56.84
3	<a href="#">Analog to Digital Converter</a>	Bit 100k Samples per Second Analog to Digital	1	9		Digi-Key	\$ 50.00	\$ 50.00
4	<a href="#">4-20 mA input</a>	Current Loop Receiver 16- Bit ADS1115 I2C Mini	1	10		NCD.io	\$ 45.95	\$ 45.95
5	<a href="#">MAX232 Board</a>	232 Interface Evaluation Board	1	14.3		Digi-Key	\$ 8.06	\$ 8.06
6	<a href="#">Enclosure</a>	CHASSIS ALUM UNPAINTED 6"L X 5"W	1	270		Digi-Key	\$ 16.38	\$ 16.38
7	<a href="#">Perma-Proto board</a>	Quarter-sized Breadboard PCB - Single	1	6		Digi-Key	\$ 2.95	\$ 2.95
8	<a href="#">DC/DC power converter</a>	Replacement DC DC Converter 1 Output 9V	1	2		Digi-Key	\$ 11.70	\$ 11.70
9	<a href="#">RJ45 Connector</a>	8p8c (RJ45, Ethernet) Straight Shielded Cat5e	1	11.4		Digi-Key	\$ 10.07	\$ 10.07
10	<a href="#">Rocker Switch</a>	(AC) 125V Panel Mount, Snap-In	1	3.8		Digi-Key	\$ 0.98	\$ 0.98
11	<a href="#">Panel mount jack</a>	Jack 2.50mm ID (0.098"), 5.50mm OD (0.217")	1	5.3		Digi-Key	\$ 4.47	\$ 4.47
12	<a href="#">Power plug Arduino</a>	Plug 2.10mm ID (0.083"), 5.50mm OD (0.217") Free	1	2.7		Digi-Key	\$ 1.65	\$ 1.65
13	<a href="#">Power plug panel jack</a>	Plug 2.50mm ID (0.098"), 5.50mm OD (0.217") Free	1	2.5		Digi-Key	\$ 1.65	\$ 1.65
14	<a href="#">Pressure sensor</a>	Resolution Digital Measurement, General	1	138.8		Transducers direc	\$ -	\$ -
								\$ -
Total				444				\$ -
Measured				444				\$ -
Total			14	559.3				\$ 249.20

Figure 15: Mini-DAQ\_V1 Bill of Materials

## 5.3 MINI-DAQ VS HWRL-DAQ COMPARISON

To quantify the quality and usefulness of the Mini-DAQ a comparison with a known professional working DAQ system was completed. Primarily, the analog to digital conversion was analyzed with four input signals and various metrics for analysis as described in this section. Each signal was split at the source and routed to the Mini-DAQ and HWRL-DAQ systems.

### 5.3.1 HWRL-DAQ SPECIFICATION

The HWRL-DAQ runs on National Instruments PXI architecture computers. They use a real-time version of the LabVIEW programming environment. Analog data acquisition is controlled by a NI PXI-6295 M-series 16-bit multifunction DAQ module. The DAQ module communicates via the PXI/SCXI backplane to SCXI-1143 Butterworth anti-aliasing filter modules in its containing PXI-1052 chassis. Each filter module is fronted with SCXI-1305 terminal blocks that take +/-5V differential inputs from analog channels via 50 ohm coaxial cable with BNC connectors. The SCXI-1143 Butterworth anti-aliasing filters are set with cutoff frequency at  $\frac{1}{4}$  the sampling rate. The HWRL DAQ is shown in Figure 16.

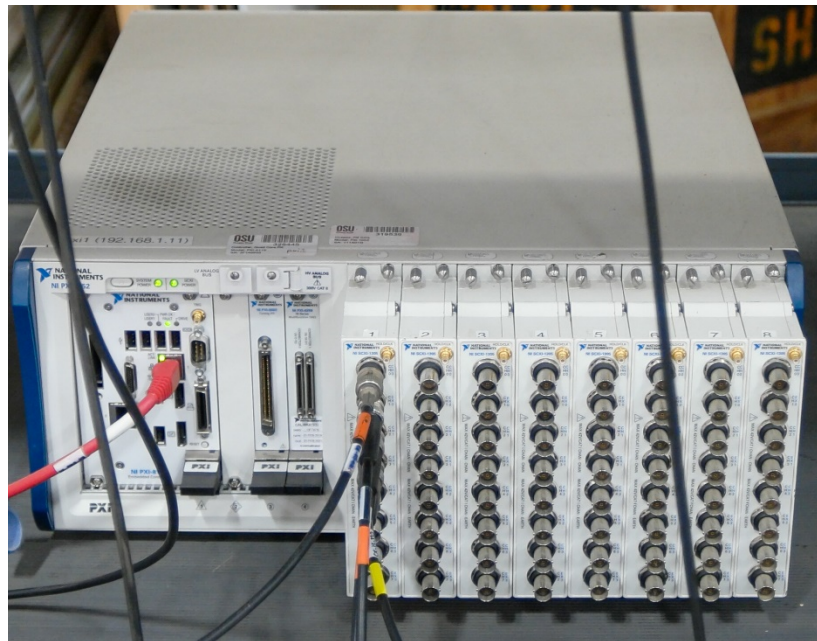


Figure 16: HWRL-DAQ system

### 5.3.2 VOLTAGE STANDARD INPUT

The first comparison between the Mini-DAQ and HWRL-DAQ was with an Analog AN3100 voltage standard input as shown in Figure 17. The voltage standard has an up to date NIST traceable calibration. The DAQ systems were run for 60 minutes with the output shown in Figure 18: Recorded voltage standard signals on Mini-DAQ and HWRL-DAQ. The resulting statistics are summarized in Table 1. Although the HWRL-DAQ results are better, the results from the Mini-DAQ are likely suitable for most testing applications.



Figure 17: Voltage standard used in DAQ-DAQ test

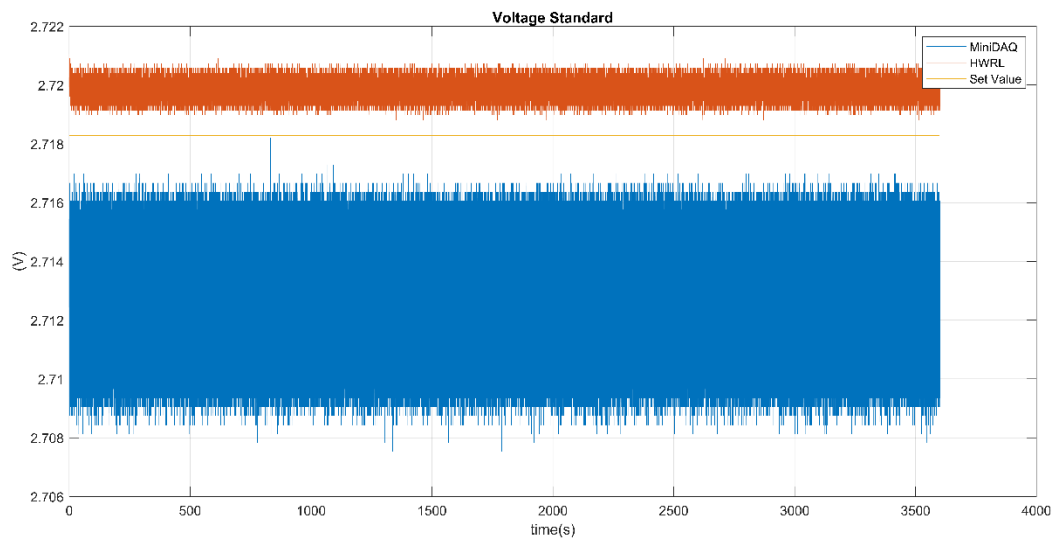


Figure 18: Recorded voltage standard signals on Mini-DAQ and HWRL-DAQ

Table 1: MiniDAQ-HWRLDAQ voltage standard comparison

	MiniDAQ	HWRL-DAQ
Mean	2.7128	2.7199
% error	0.2033	0.0580
Standard deviation	0.0011	0.0002

### 5.3.3 PRECISION TIME PROTOCOL

To align the signals between the two DAQ systems a precision time protocol (PTP) standard was used. HWRL has a GPS based time system which outputs PTP. The HWRL-DAQ is already set up to read and record time based on the PTP signal. The Speedgoat system was also set up to record time based on the PTP. This allowed for alignment of the signals to quantify the time it took each system to acquire the signal.

### 5.3.4 LED SIGNAL

The next signal recorded was what the lab calls the LED channel. This is a channel with a signal that turns on and off with a random delay between 0.25 s and 5 s. This channel is often recorded on multiple DAQ systems to synchronize data between them. Figure 19 shows the first 6 seconds of the record of this signal. The signals have been aligned using the PTP protocol. The ringing shown on the HWRL DAQ signal is due to the anti-aliasing filters.

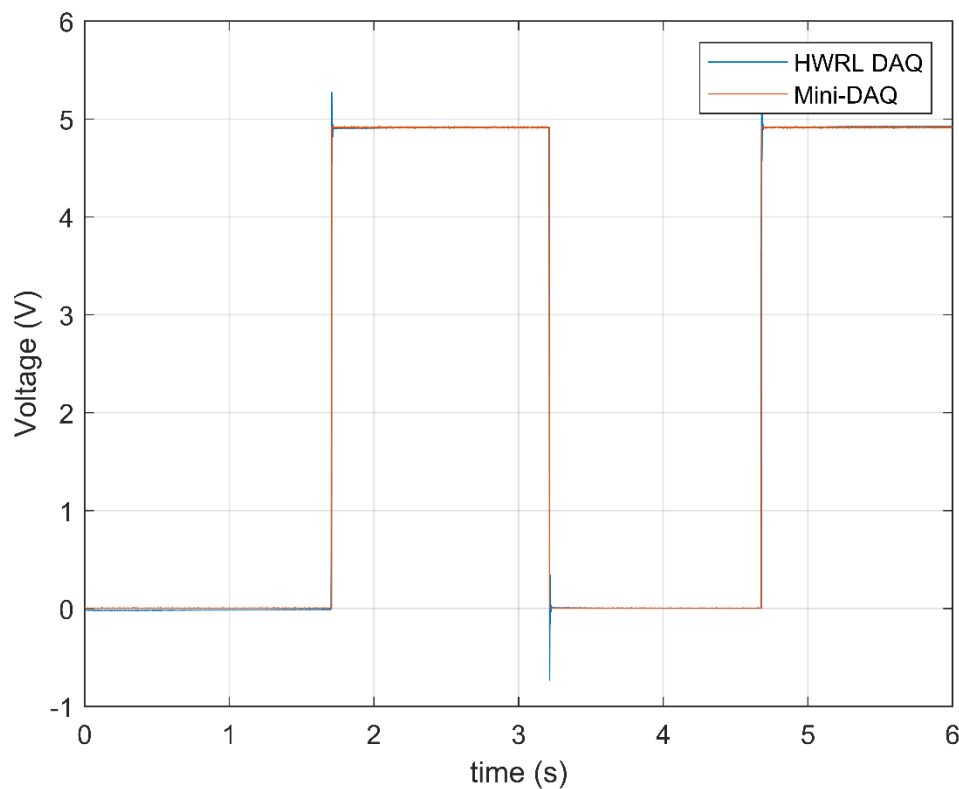


Figure 19: LED signal with random delay

If we then look near the end of a 60 minute recording we see that the clock skew shows a delay of about 0.5 seconds as shown in Figure 20. This will be investigated in future work. First a real-time clock module will be used for the Arduino system as this is suspected as the source of the discrepancy.

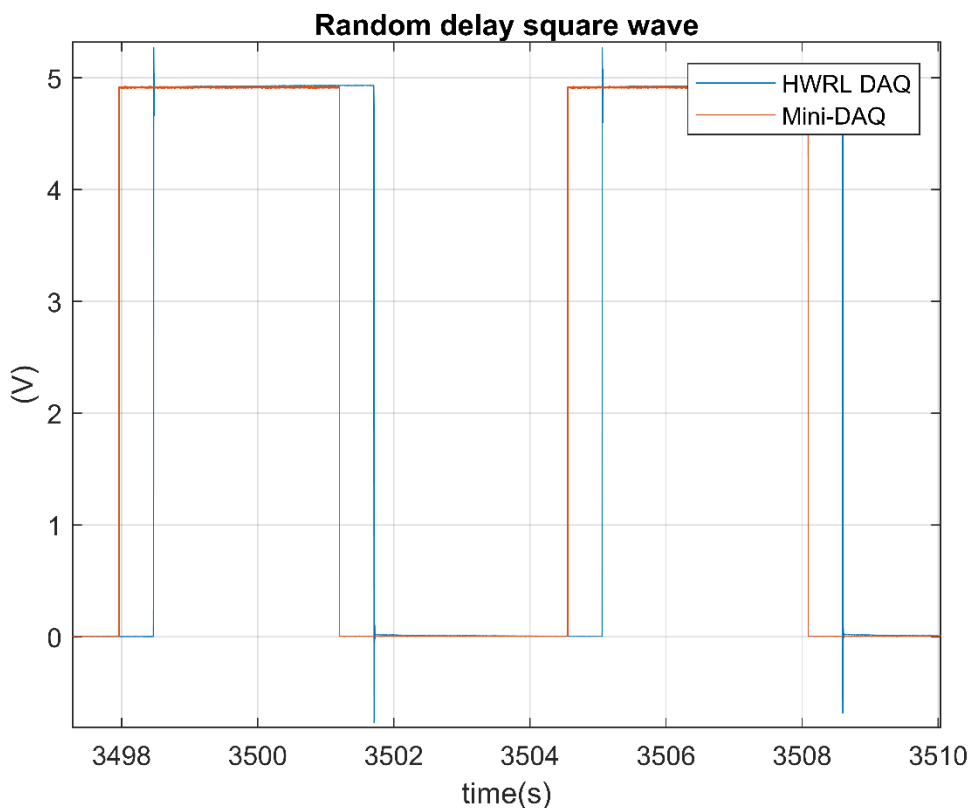


Figure 20: LED signal at end of 60 minute record. Notice delay of approximately 0.5 seconds showing the drift in the Arduino clock relative to the National Instruments clock

### 5.3.5 SINE WAVE

The next input was a sine wave with frequency of  $\sim 1$  Hz. These signals were aligned with the PTP at the start of the record. Figure 21 shows the full record, start of record, and end of record showing the  $\sim 0.5$  s shift which will be investigated in future work. Next investigating of the clock skew and how it changes over time was done. Analyzing the zero crossings of the waveform and calculating the difference between the HWRL DAQ and the Mini-DAQ gives the results shown in Figure 22. The top plot shows the period of each wave as a function of time. Notice the very small drift in both of the upper plots and the larger band in the Mini-DAQ data. In the lower plot notice the skew getting larger with time linearly and see the line indicating the HWRL DAQ specification.

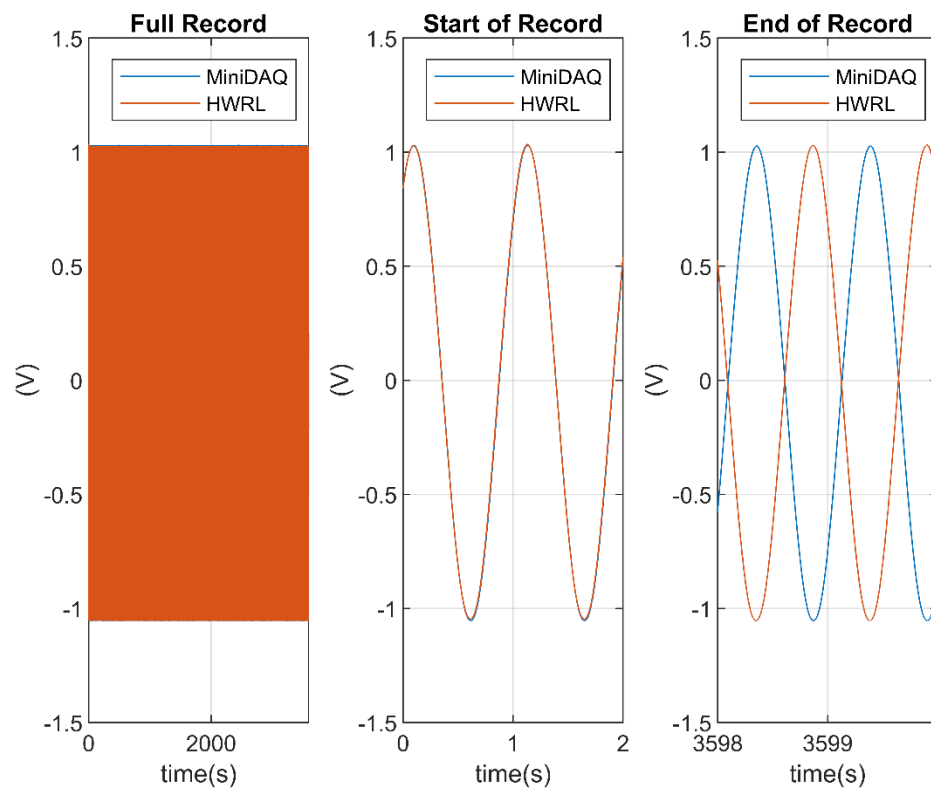


Figure 21: Sine wave signal results

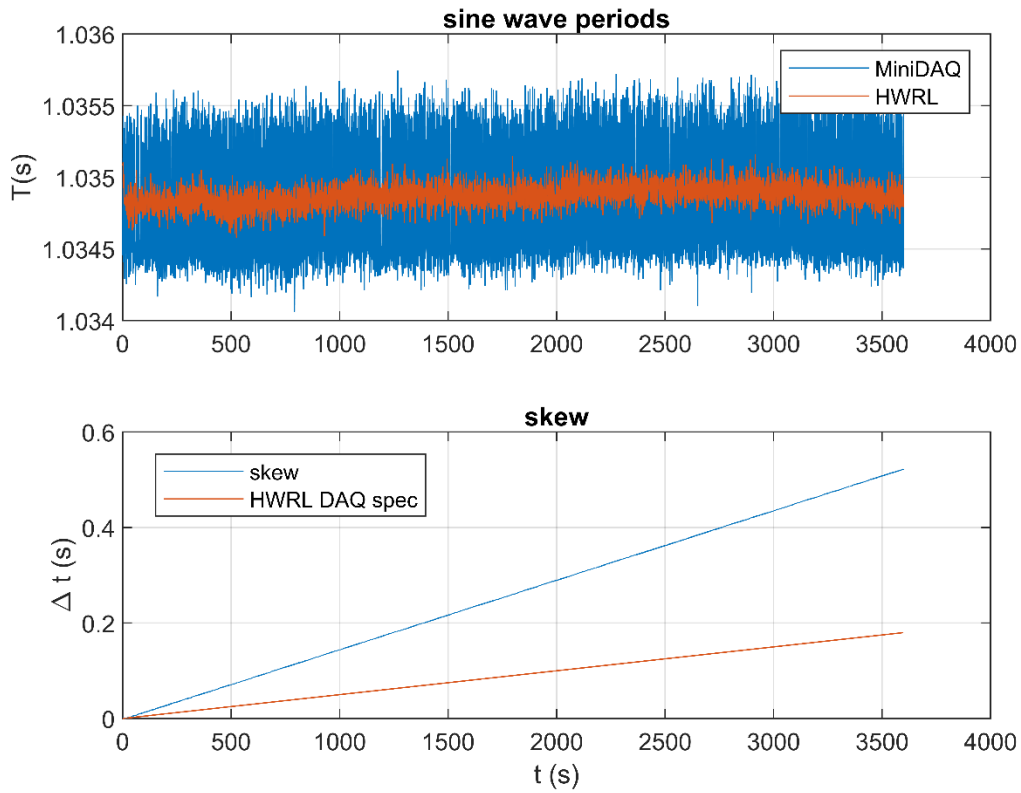


Figure 22: clock periods and skew

### 5.3.6 WHITE NOISE

The final signal recorded on both DAQs was a white noise signal. This signal was created using Simulink Desktop Real-Time and was output on a National Instruments PCIe6363 data acquisition card. The signal was band-limited from 0.5 Hz to 25 Hz and was captured on the Mini-DAQ and HWRL-DAQ systems. Figure 23 shows the power spectral density for the input, recorded Mini-DAQ, and recorded HWRL-DAQ data. System identification was performed on the two input signals to estimate the relationship between the signal captured on the Mini-DAQ and the signal captured on the HWRL-DAQ. The MATLAB command `iddata` was used with the `spa` command to estimate frequency response with fixed frequency resolution using spectral analysis with the confidence interval option on. The resulting bode plot is shown in Figure 24.



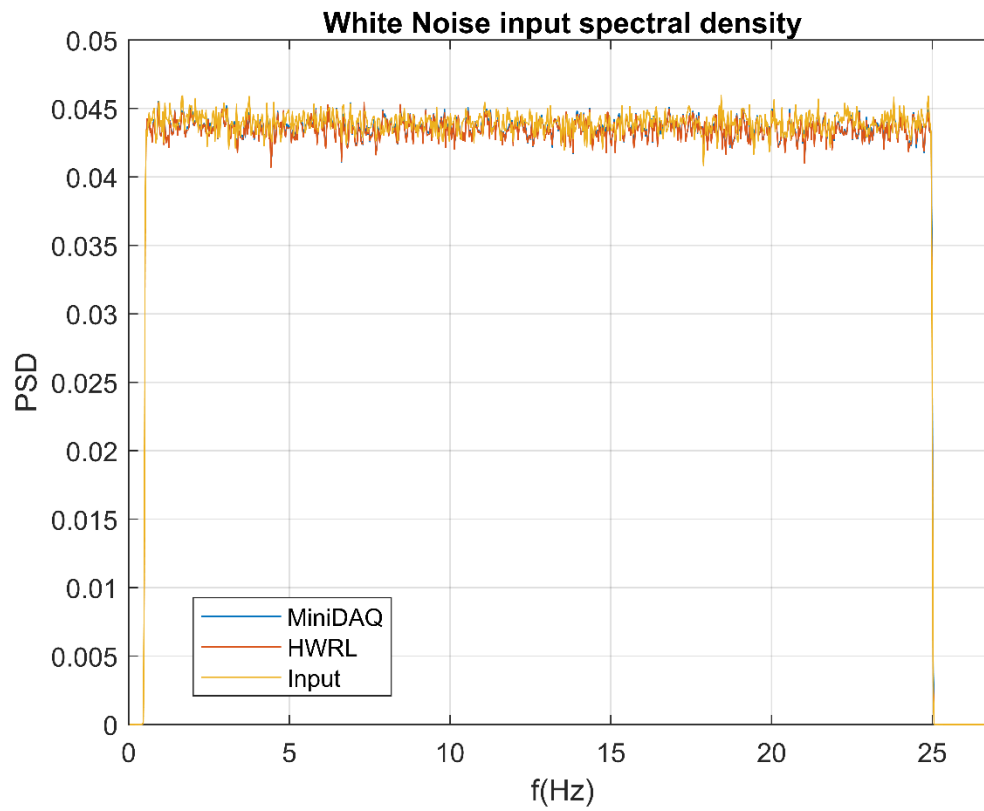
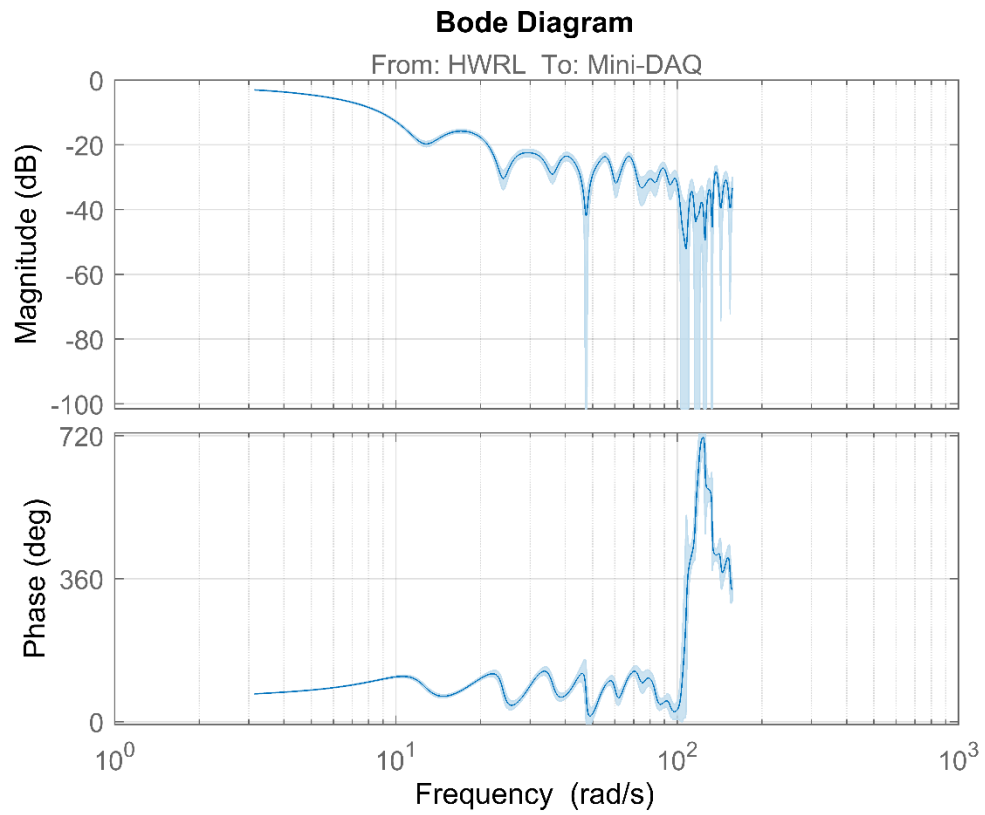


Figure 23: White noise power spectral density showing input signal, Mini-DAQ recording, and HWRL-DAQ recording.



*Figure 24: System Identification for white noise*

Also identified in the white noise capture, when sampled at 1 kHz, was a small periodic glitch as shown in Figure 25. This will be investigated in future work.

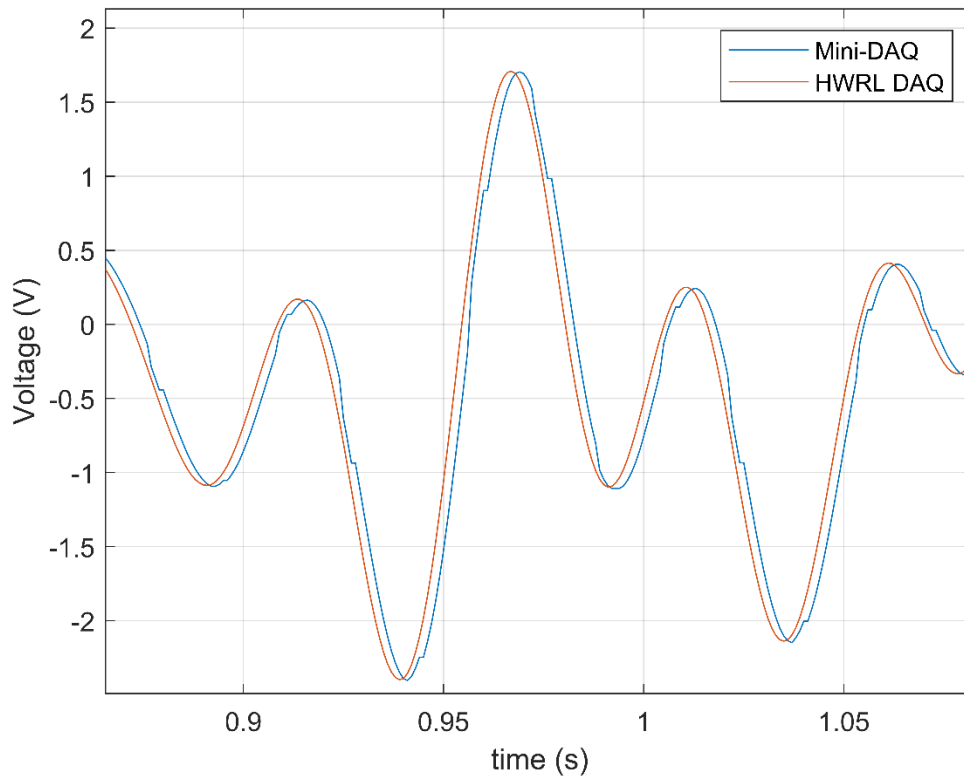


Figure 25: Close-up of white noise capture with the HWRL DAQ and Mini-DAQ. Notice the small periodic glitch on the Mini-DAQ signal

## 6. DEMO WEC

The DemoWEC is a bench top demonstration system designed to demonstrate the concept of the Mini-DAQ and show some of its potential. The system represents a dry version of a WEC and provides a complete feedback-controlled system as shown in Figure 25. A brushed DC motor and quadrature encoder attached to a gearbox and spindle drive allow for linear motion and control. An optical absolute position sensor is used for homing. A motion profile can be chosen, with provisions for regular sinusoidal input or JONSWAP spectrum-based time series. After a homing sequence that centers the “float”, the desired motion profile will be followed. An Arduino Nano is used to capture encoder counts and send them via serial to the Arduino Mega. The Demo WEC system connects to a Speedgoat computer via EtherCAT to command the motion and provide visualization and logging capabilities.

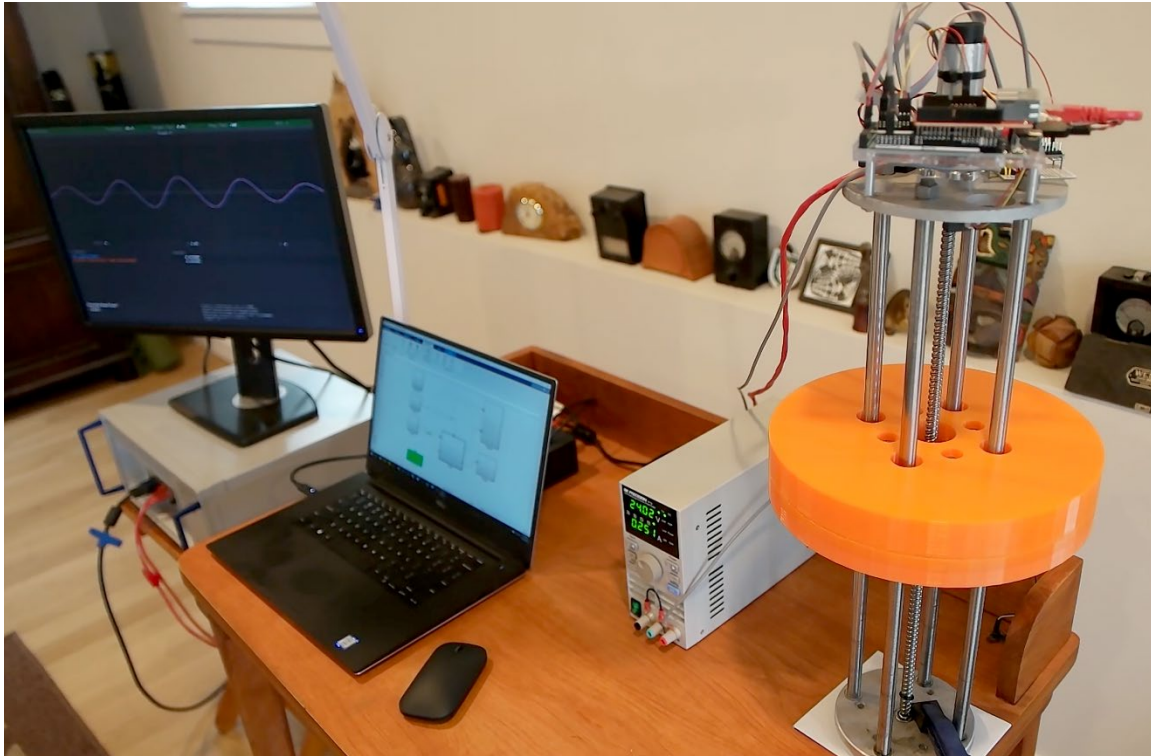


Figure 26: Demo WEC bench top testing

## 7. FUTURE WORK

- Investigate and attempt to remedy clock skew
  - Add real-time clock to Arduino
  - Investigate EtherCAT distributed clock options
- Investigate and attempt to remedy glitch in analog in data
  - Look at task execution times and Arduino loop timing
- Investigate packaging options and creating a standard set of inputs and outputs

## REFERENCES

- [1] G. Prytz, "A performance analysis of EtherCAT and PROFINET IRT," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, Sep. 2008, pp. 408–415, doi: 10.1109/ETFA.2008.4638425.
- [2] "Getting Started with Arduino MEGA2560." <https://www.arduino.cc/en/Guide/ArduinoMega2560> (accessed Apr. 12, 2021).
- [3] "EtherCAT® and Arduino." <https://www.bausano.net/en/hardware/ethercat-e-arduino/easycat.html> (accessed Apr. 12, 2021).
- [4] "Arduino IDE Software." <https://www.arduino.cc/en/software> (accessed Apr. 12, 2021).
- [5] "Simulink real-time-target machines | Speedgoat." <https://www.speedgoat.com/products-services/real-time-target-machines> (accessed Apr. 12, 2021).
- [6] "Simulink Real-Time." <https://www.mathworks.com/products/simulink-real-time.html> (accessed Apr. 12, 2021).
- [7] "DC682A Evaluation Board | Analog Devices." <https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/dc682a.html> (accessed Apr. 12, 2021).
- [8] "LTC1859 Datasheet and Product Info | Analog Devices." <https://www.analog.com/en/products/ltc1859.html#product-overview> (accessed Apr. 12, 2021).
- [9] "MAX232 Board - | Mikroe," *MikroElektronika*. <http://www.mikroe.com/max232-board> (accessed Apr. 12, 2021).
- [10] Xsens, "MTi 10-series." <https://www.xsens.com/products/mti-10-series> (accessed Apr. 12, 2021).
- [11] "2-Channel 4-20 mA Current Loop Receiver 16-Bit ADS1115 I2C Mini Module," *store.ncd.io*. <https://store.ncd.io/product/2-channel-4-20-ma-current-loop-receiver-16-bit-ads1115-i2c-mini-module/> (accessed Apr. 12, 2021).
- [12] "TD1000 Pressure Transducer - Vacuum & Compound Pressure Ranges," *Pressure Transducers, Pressure Sensors: Transducers Direct*. <https://transducersdirect.com/products/pressure-transducers/vacuum-pressure-transducers/td1000-pressure-transducer/> (accessed Apr. 12, 2021).
- [13] "Arduino Nano | Arduino Official Store." <https://store.arduino.cc/usa/arduino-nano> (accessed Apr. 16, 2021).
- [14] B. A. G. & C. K. Germany Hülshorstweg 20, 33415 Verl, "TwinCAT | Automation software," *Beckhoff Automation*. <https://www.beckhoff.com/en-us/products/automation/twincat/> (accessed Apr. 16, 2021).
- [15] "Visual Studio IDE, Code Editor, Azure DevOps, & App Center - Visual Studio." <https://visualstudio.microsoft.com/> (accessed Apr. 16, 2021).

## APPENDIX A. ARDUINO CODE EXAMPLE

[illegible]



// —▲  
}

```
void loop()
```

 $\{$ 

```

t-micros(); // Monitor Controller internal time at loop start
heartbeat(); // Blink LED to prove it is alive!
EASYSAT.MainTask(); // UDF for EasyCAT Trigger
read_ADC(); // UDF for ADC reading
//adc0 = ads.Measure_SingleEnded(0); // current sensor read value
read_IMU(); // UDF for IMU reading
EtherCAT_Frame_Update(); // UDF for EtherCAT data Frame Updating
IMU_Packet_DeFrame(); // UDF for IMU Data frame decoding

```

```
Serial.print(N) ;Serial.print("\t");
```

```
Serial.println((micros()-t)/1000.); // Loop execution time
```


```
while(!myTimer.isTimeReached()); // Wait here Till 1 msec sample time Tick
}
```

//  User Defined Functions 

//

```
// Bin To Float
float binToFloat(byte D3, byte D2, byte D1, byte D0)
```

```
{
    byte bin[4];
    bin[3] = D3; bin[2] = D2; bin[1] = D1; bin[0] = D0;
    return *( (float*) bin );
}
```

//  ADC Read 

```
void read_ADC()
```

```
{
  for(word add=0;add<=3;add++)
    digitalWrite(CS_ADC, LOW);
    word x= add<12 | 0x4000;
    M1ADC[add]=SPI.transfer16( x );
    digitalWrite(CS_ADC, HIGH);
    digitalWrite(CS_ADC, LOW);
    M1ADC[add]=SPI.transfer16( x );
    digitalWrite(CS_ADC, HIGH);
  }
}
```

```
// ===== Initialize current sensor =====
```

```
void Initialize_currentSens(void)
```

```
ads.getAddr_ADS1115(ADS1115_DEFAULT_ADDRESS); // 0x48, 1001 000 (ADDR = GND)
ads.setGain(GAIN_ONE); // 1x gain +/- 4.096V 1 bit = 0.125mV
ads.setMode(MODE_CONTIN); // Continuous conversion mode
ads.setRate(RATE_128); // 128SPS (default)
ads.setOSMode(OSMODE_SINGLE); // Set to start a single-conversion
ads.begin();
}
```

```
// ===== Initialize IMU =====
```

```
void Initialize_IMU(void)
```

```
{
  byte temp[5]={0,0,0,0,0};
  int i=0;
  while(i<=10) // Ten attempts to put IMU in Measure Mode
  {
    delay(500); //Needed to let the IMU wake up before serial communication
    Serial1.write(Put2_Config_Mode,5);
    //delay(5);
    while(Serial1.available()>0) Serial1.read();
    delay(5);
  }
}
```

```
Serial1.write(SetOutputConfiguration,17);
delay(100);           // This delay is needed to prevent fail
while(Serial1.available()>0) Serial1.read();
```

```

Serial1.write(Put2_Measure_Mode,5);
Serial1.readBytes(temp, 5);

if(temp[2]==17)
{
  Serial.println("\n IMU Initilised Successfully");
  delay(1000);
  break;
}
i=i+1;
Serial.print(" ");
}
if(i==11)
{
  Serial.println("IMU Initilisation Failed 10 Attempts");
  //
  while(1); // Code stop here.
}
}

//===== IMU Read =====
void read_IMU(void)
{
  Serial1.write(Request_IMU_Packet,5);
  N=0;
  while(Serial1.available()>0 && N<Len) IMUPacket[N++]=Serial1.read();
  while(Serial1.available()>0)Serial1.read(); //flush
  //Serial.println(IMUPacket[8]);
}

//===== IMU Data Deframe =====
void IMU_Packet_Deframe(void)
{
  if(N==Len)
  {
    r = binToFloat(IMUPacket[7],IMUPacket[8],IMUPacket[9],IMUPacket[10]); //deg
    p = binToFloat(IMUPacket[11],IMUPacket[12],IMUPacket[13],IMUPacket[14]);
    y = binToFloat(IMUPacket[15],IMUPacket[16],IMUPacket[17],IMUPacket[18]);

    ax=binToFloat(IMUPacket[22],IMUPacket[23],IMUPacket[24],IMUPacket[25]); //m per second^2
    ay=binToFloat(IMUPacket[26],IMUPacket[27],IMUPacket[28],IMUPacket[29]);
    az=binToFloat(IMUPacket[30],IMUPacket[31],IMUPacket[32],IMUPacket[33]);

    T=binToFloat(IMUPacket[37],IMUPacket[38],IMUPacket[39],IMUPacket[40]); // deg C
  }
}

//===== EtherCAT Data Update =====
void EtherCAT_Frame_Update()
{
  EASYCAT.BufferIn.Cust.ADC0 = M1ADC[0]; // analog to digital channel 0
  EASYCAT.BufferIn.Cust.ADC1 = M1ADC[1];
  EASYCAT.BufferIn.Cust.ADC2 = M1ADC[2];
  EASYCAT.BufferIn.Cust.ADC3 = M1ADC[3];

  EASYCAT.BufferIn.Cust.roll = r; //roll
  EASYCAT.BufferIn.Cust.pitch = p; //pitch
  EASYCAT.BufferIn.Cust.yaw = y; //yaw

  EASYCAT.BufferIn.Cust.acc_x = ax; //acceleration in x
  EASYCAT.BufferIn.Cust.acc_y = ay; //acceleration in y
  EASYCAT.BufferIn.Cust.acc_z = az; //acceleration in z

  EASYCAT.BufferIn.Cust.current0 = adc0;

  EASYCAT.BufferIn.Cust.temperature = T;
}

//===== Toggle LED to prove alive =====
void heartbeat()
{
  count++;
  if (count == 1){
    digitalWrite(Debug_LED, HIGH); //Turn LED on
  }
}

```



```

    }
    else if (count == 100){
        digitalWrite(Debug_LED, LOW); //Turn LED off
    }
    else if (count == 200){
        count = 0;
    }
}

```

[illegible]