

Vibration Energy Harvester for Powering Field Instruments

ECE 480
Spring 2016
Team 11

Sponsor

Instrumented Sensor Technology, INC., Okemos, MI

Facilitator

Dr. Xiaobo Tan

Team Members

Andrew Schroeder
Daniel Fresneda-Rojas
Daniel “Tom” Dochoda
Jovanni Ivanaj
Stephen Pliska

MICHIGAN STATE
UNIVERSITY



Executive Summary

The demand for means of sustainable-portable energy is increasing indefinitely. However, the technological advancement currently addressing this is not sufficient. Accordingly, we have been working towards a product to circumvent the demand for prolonged energy where viable sources are scarce. With economic expense in mind, our product is effective for harnessing vibrational energy for conversion into usable and storable electric charge.

Instrumented Sensor Technology, INC, has provided us the challenge to develop a system in which to extend the operational period of their field device. This device is designed to operate within the trailer of a commercial shipping vehicle, which provides substantial vibration at rates of low frequency. Most available vibration energy harvesting methods involve mechanisms that function poorly in such low frequency environments. Our resourcefulness has allowed us to bridge the gap where most available products lack compatibility.

We have implemented the concept of electromagnetic induction for its ability to operate at low frequency to generate charge. For prolonging the operation of IST's device, a single rechargeable lithium-ion battery is installed. This alternative, to disposable batteries, grants the previously nonexistent ability to efficiently provide adequate-energizing power all while storing energy collected from the environment.

These two systems integrated with a third powertrain-system have proven to supply enough power for the field device over, theoretically, an indefinite length of time. Further evaluations have yielded promising evidence confirming this product meets the demand for sustainable and portable energy for applications in shipping trucks.

Acknowledgements

The team would like to thank:

- Our sponsor: Instrumented Sensor Technology, INC. We deeply appreciate the opportunity you have provided for us, the welcoming and assistance of the faculty, and especially Greg Hoshal for his guidance and support of our creativity towards tackling one of the industry's ongoing projects from the beginning.
- Our faculty facilitator, Dr. Xiaobo Tan, for his willingness to assist us in our product's quality development and his analytic capability that provided additional guidance throughout the design process.
- Aaron Walworth, Patrick McDavid, and Alyssa Harben at the MSU School of Packaging. Each one of you were very receptive to our team's interest in the lab and we really appreciate your ability to find time to help us to test our device, those tests made all the difference. Aaron you get a special second mention for not charging us to use the lab, Alyssa you get a second mention for playing hangman with us, and Patrick you get a second mention for helping find that tiny resistor.

We appreciate the support from everyone mentioned and, also, want to thank anyone who indirectly assisted throughout the four months. The help from everyone made a measureable difference in our experience working on this project and the project's success.

Table of Contents

<u>Executive Summary</u>	2
<u>Acknowledgements</u>	3
<u>Table of Contents</u>	4
<u>Chapter 1 – Introduction and Background</u>	6
<u>1.1 Introduction</u>	6
<u>1.2 Background</u>	6
<u>Chapter 2 – Exploring The Solution and Approach</u>	8
<u>2.1 Critical Customer Requirements</u>	8
<u>2.2 Design Choices</u>	8
<u>2.2.1 Viable Subsystem Concepts</u>	8
<u>2.2.2 Vibration Harvesting Subsystem</u>	9
<u>2.2.3 Powertrain Subsystem</u>	10
<u>2.2.4 Power Supply Subsystem</u>	10
<u>2.2.5 Monitoring Subsystem</u>	11
<u>2.3 Budget</u>	13
<u>2.4 Schedule</u>	14
<u>Chapter 3 – Technical Description of Work Performed</u>	15
<u>3.1 Energy Harvesting Apparatus</u>	15
<u>3.2 Powertrain and Supply</u>	19
<u>3.3 Monitoring Subsystem</u>	21
<u>3.4 Energy Storage and Supply Subsystem</u>	23
<u>Chapter 4 – Test Data With Proof of Functional Design</u>	25

<u>4.1 - Vibrational Energy Harvester</u>	25
<u>4.1.1 - Test 1</u>	26
<u>4.1.2 - Test 2</u>	27
<u>4.1.3 - Test 3</u>	28
<u>4.2 - Powertrain System</u>	30
<u>4.2.1 - Rectification</u>	30
<u>4.2.2 - Voltage Conversion</u>	31
<u>4.3 - Microcontroller</u>	35
<u>4.4 - Entire Device</u>	36
<u>4.5 - Proposal Design Specs</u>	40
<u>Chapter 5 – Design Issues</u>	42
<u>5.1 Product Liability</u>	42
<u>5.2 Universal Design Principles</u>	43
<u>Chapter 6 – Final Cost, Schedule, Summary and Conclusion</u>	44
<u>Appendix</u>	45
<u>Appendix 1 – Technical roles, responsibilities, and work accomplished</u>	45
<u>Stephen Pliska</u>	45
<u>Andrew Schroeder</u>	46
<u>Daniel “Tom” Dochoda</u>	46
<u>Daniel Fresneda-Rojas</u>	47
<u>Jovanni Ivanaj</u>	48
<u>Appendix 2 – Literature and website references</u>	49
<u>Appendix 3 – Detailed technical attachments</u>	51

Chapter 1 – Introduction and Background

1.1 Introduction

Over the past several decades there has been a growing demand for electronic devices that require a portable battery. This demand exists in nearly every electronic market and it unlikely to decline in the near future. As a result of this there is an increasing interest in research for battery technology, and battery-life improvement. However, the demand for portable energy is growing much more rapidly than the developing-battery technology available in the markets today.

A field of research that shows promise, to help with extending battery life, is vibrational energy harvesting. This is a conversion of mechanical energy into useful electrical energy. The idea is that the mechanical energy, due to vibration, can be harnessed in order to develop an electric charge differential that can be transferred into usable and storable electrical energy.

What our team is trying to working to accomplish is to find a inventive, new way to extend the operation length of a field instrument designed and provided to us by our sponsor, IST. Currently, vibrational energy harvesting products exist that are mainly effective only in higher-frequency vibration environments. These are mainly piezoelectric devices. Therefore, there are very few products that can develop electrical energy in low-frequency environments, such as on a shipping truck (semi-truck). We want to design and produce a device that can extend the life of the field instrument, and potentially replace the internal, and temporary, battery system.

1.2 Background

The instrument is an acceleration recording device, VTP-3Dv2, which is designed to measure and record shock vibration (and temperature/humidity) data. It currently can operate for a period of three months equipped with its built-in, non-rechargeable battery system. Roughly 5mA is drawn by the device and supplied by the current battery system consisting of four AA 3.6v lithium batteries, each providing a capacity of approximately 2600mAh.

Our approach, which reflects many of the engineering principles developed in our collegiate career, has been to develop a system in which low-frequency vibrations can be used to create a changing magnetic field to induce a manageable electrical potential. Specifically, a powerful neodymium magnet is suspended by springs, and encapsulated

within a conductive coil in which the electrical energy will be generated. We have strong belief in the functionality of this approach due to having experience with the physics concepts in our collegiate career, and understanding the evidence that there are functional products in the market today that harness low-frequency mechanical energy into electrical energy, but for very different applications.

Under the assumption that we are successful, we will be able to effectively lengthen the operation life of the VTP-3Dv2 and, or replace the currently equipped battery system. Our product can potentially be further engineered to be able to improve size, cost, energy-transfer efficiency, and the spectrum of application. With further research, and increased resources, other groups will be able to take the technology we are developing and expand the foundation for battery technology that currently exists.

Chapter 2 – Exploring The Solution and Approach

2.1 Critical Customer Requirements

Using the quality function deployment technique, our team identified the Critical Customer Requirements of our sponsor, Instrumented Sensor Technology. We found that the most important requirement of our project was that our design be able to harvest vibrations in order to extend the operation length of IST's instruments. This is the basic function of our project. In order to successfully complete this function, our requirements are:

- To convert and store energy harvested from low-frequency vibrations present in shipping truck environments
 - The device from IST is commonly found in shipping truck beds and inside packages transported by trucks. These vibrations tend to be of low frequency and high force.
- To provide the stored energy as primary power supply for the VTP-3Dv2
 - The device from IST is typically powered by four 3.6 volt batteries. Our design must be able to power the device with a similarly appropriate voltage.
- Develop mathematical relationships between vibrational energy and stored electrical energy as a function of time.

2.2 Design Choices

Our approach for effectively tackling the demands of our product led to a separation of distinct subsystems, each with their own respective purpose in mind. By separating into subsystems, that together handle all the requirements, we can analyze potential subsystem designs in better detail. The most crucial subsystem is the vibration harvesting apparatus. Secondly, the powertrain subsystem used to transform raw electric potential into usable energy. Next, a power supply subsystem for storing and supplying energy usable by the VTP-3Dv2. Finally, a monitoring subsystem which records data values for product efficiency.

2.2.1 Viable Subsystem Concepts

Vibration Harvesting Apparatus:

- Electromagnetic induction
 - Generates varying electrical potential at low frequencies

- Piezoelectric material
 - Generates varying electrical potential at high frequencies
 - Option promptly eliminated after basic understanding and evaluation
- Steel Spring Suspension
 - Allows for wide range of vertical motion
 - Cheap and wide varieties available
- Magnetic Suspension
 - Frictionless
 - More expensive
 - Creates additional magnetic fields that may decrease efficiency of coil
 - Option eliminated due to risk to efficiency

Generate an AC Voltage:

- Coil to make readily available electrical energy
 - Provided electromagnetic induction occurs

Convert AC to DC Voltage:

- Voltage rectification circuit to maintain a DC voltage output
 - Voltage doubler, tripler, etc.
 - Bridge Rectifier

Develop Efficient Useable Power:

- Add circuit system to provide desired DC voltage levels
 - Buck-Boost Converter, Linear Regulators
- Include microcontroller to monitor values and provide analytical data
 - Must be low power to provide most efficiency

Store and Supply Power:

- System that allows continuous operation of VTP-3Dv2 and can store extra charge to avoid loss of valuable energy
 - Supercapacitor, rechargeable battery
- Proper supply of power for the VTP-3Dv2
 - Supercapacitor or rechargeable battery with outputs rated at values near device power demands.

2.2.2 Vibration Harvesting Subsystem

The vibration harvesting subsystem will produce an AC voltage caused by the nature of electromagnetic induction requiring a change in magnetic field over time. Data obtained from IST specifies that the typical frequencies of vibration present in a shipping truck trailer are below 10 Hz. In order to maximize the amount of energy generated, our design should resonate at a frequency less than 10 Hz.

A method, suggested by IST, involves using piezoelectric materials which produce AC voltage due to vibrations. There are many piezoelectric energy harvesters available in the market. A second method considered uses magnetic induction using an oscillating magnet within an inductive coil. Faraday's law states that this changing magnetic field induces a current in the coil which produces an AC voltage at the coil terminals.

Piezoelectric devices can be very efficient, however they are only operational at much higher frequencies. Therefore an electromagnetic-apparatus and coil design was unanimously decided as it effectively generates AC power while operating well for low-frequency applications. By building a spring-mass system with a magnet between two springs, we could calculate the appropriate spring constants which would lead to the system resonating at a specific frequency below 10 Hz thereby increasing the chances of our design succeeding.

2.2.3 Powertrain Subsystem

The powertrain subsystem must be able to take the AC voltage produced by the vibration harvester and convert it into a DC voltage that is large enough for the succeeding subsystems to use. This requires a voltage rectifier of variable gain increase. The first iteration of our design used a voltage tripler for this purpose. However, we quickly noticed that the power loss incurred by the numerous diodes in the circuit was too large. It was difficult for the input from the vibration harvester to reach the minimum voltage required to get a stable output of the powertrain. Our final design is a bridge rectifier which is passed through a boost converter. We decided on this approach due to the nominal efficiency of boost converters.

2.2.4 Power Supply Subsystem

The supply subsystem involves regulating the voltage from the powertrain to a suitable level for battery charge and allowing the battery itself to supply power to the VTP-3Dv2. In order to convert the potentially high voltages produced by the powertrain, we decided to use a buck converter which are known to have very high efficiency, meaning low power losses.

Our feasible choices for an energy storage cell are either rechargeable batteries or supercapacitors. Our team decided to move forward with installing a rechargeable lithium-ion battery rather than a supercapacitor. The supercapacitor is capable of accepting and delivering charge faster than a rechargeable battery, however they have a considerably-low energy density and have a self-discharge rate much quicker than that of a lithium-ion battery. We are capable of purchasing from a large variety of rechargeable batteries in the market at nearly the exact capacity and voltage rating that

the VTP-3Dv2 requires. Datasheets developed by the battery manufacturer provided detailed test results that show that, for our application, we would very likely be able to properly charge the battery and supply sufficient power. Therefore, a rechargeable lithium-ion battery is unquestionably the most effective power storage device for our supply subsystem system.

2.2.5 Monitoring Subsystem

The monitoring subsystem involves measuring the power output and efficiency of the system as a whole. The method that we decided to use for this subsystem was to utilize a microcontroller which measures the current and voltage coming out of the powertrain and the supply subsystems and compare them. Using basic calculations the microcontroller can calculate the efficiency and power output of the whole system using these measurements.

The main factors for choosing an effective microcontroller are power usage, the resolution of its analog-to-digital converter (ADC), and the availability of an LCD module. Various popular microcontrollers such as Arduino and Raspberry Pi models draw too much power for our application. The microcontroller that we decided to use was the Texas Instruments MSP430FR6989, contained in a Launchpad development board which also integrates it with a 6-digit LCD. This microcontroller was chosen due to its very low power usage, approximately 100 μ A per MHz in active mode, and its 12-bit ADC.

Shown in Figure 2.1 is the feasibility matrix that we used in order to decide on our final design.

		Feasibility of Potential Design Concepts (Effectiveness 1-low, 5-high, X-N/A)					
		Energy Harvesting		Energy Storage		Suspension	
Design Requirements	Importance (1-low, 5-high)	Magnetic Induction	Piezoelectric Material	Supercapacitor	Rechargeable Battery	Steel Springs	Magnetic Suspension
Low Frequency Application	5	5	1	2	4	4	3
Efficient Energy Conversion	5	4	1	X	X	5	3
Sufficient DC Voltage Output	4	1	1	2	5	4	4
Easy Operation	3	4	3	4	5	3	4
Meet VTP-3Dv2 Power Demands	5	X	X	2	4	X	X
Portable	3	3	5	4	5	5	5
Extends Operation Period	5	4	1	1	5	5	4
Stores Sufficient Power	5	X	X	1	5	X	X
Feasibility Rating Total	Importance x Effectiveness	90	43	62	140	110	93

Figure 2.1 - Selection Matrix for Energy Harvesting Subsystems

2.3 Budget

Table 2.1 shows the approximate cost that would be incurred to make another unit of our prototype.

Table 2.1

Qty.	Item	Line Total
1	N52 Neodymium Magnet	\$68.99
4	Compression Spring	\$19.90
1	PVC Pipe	\$3.95
2	PVC Cap	\$3.60
1	Foam	\$1.50
1	Magnet Wire Spool	\$28.40
1	Bridge Rectifier	\$0.65
3	Buck-Boost Converter	\$16.59
3	Inductor	\$1.14
9	Capacitor	\$3.74
4	Diode	\$1.72
10	Resistor	\$1.10
1	MSP430 Launchpad	\$17.99
1	Lithium Ion Battery	\$5.99
2	Current Sense Amplifier	\$1.76
Total		\$177.02

2.4 Schedule

At the beginning of the semester we developed a GANTT chart which we followed. This is shown in Figures 2.2 and 2.3.

TASK	START DATE	DURATION	END DATE
Discuss possible vibration harvesting techniques	1/11/2016	14	1/25/2016
Retrieve more detailed customer specs	2/4/2016	6	2/10/2016
Research power-point tracking solutions	1/15/2016	26	2/12/2016
Research power storage solutions	1/15/2016	18	2/2/2016
Finalize initial prototype design	2/10/2016	14	2/24/2016
Order parts for initial design	2/12/2016	12	2/24/2016
Assemble and test initial prototype	2/24/2016	21	3/16/2016
Meet with IST to discuss results	3/2/2016	14	3/16/2016
Review results and implement improvements	3/15/2016	28	4/11/2016
Finalize project design	4/12/2016	10	4/22/2016
Build final project	4/22/2016	7	4/29/2016

Figure 2.2 - Schedule

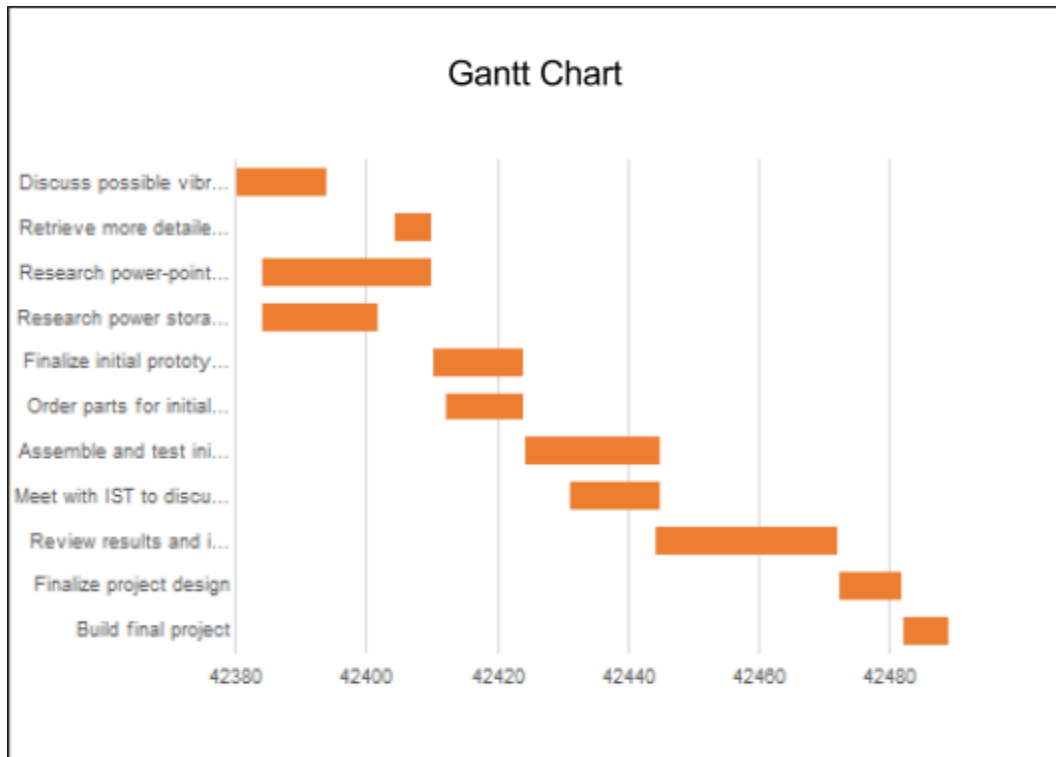


Figure 2.3 - GANTT Chart

Chapter 3 – Technical Description of Work Performed

There are four subsystems to this whole project. There is the energy harvesting apparatus, powertrain, performance monitoring, and power supply and storage.

3.1 Energy Harvesting Apparatus

N52 Neodymium magnet (778 grams) [2" (diameter) x 2" (height)]

This magnet will be sitting vertically between the wrappings of the coil. The reason why our team invested in using a neodymium rare earth material based magnet versus a common ceramic, strontium cobalt, or perhaps a barium ferrite is due to the strength of the magnet. When it comes to producing appropriate amounts of current for charging a battery, we want a strong magnet to create a strong inductive field between the magnet and the coil of wire(s). A neodymium magnet will produce more current, unlike barium ferrite, or ceramic magnets; they will not produce very much current that will be required for the circuit, which will help recharge a battery or extend battery life. The magnet we used in our design is shown in Figure 3.1.



Fig. 3.1 - Neodymium magnet used for inducing fields

(x2) Compressional Springs (210 N/m) [5" (height {each})]

(x2) Compressional Springs (210 N/m) [4.94" (height {each})]

When vibration is present the springs will oscillate and the magnet will move within the height of the magnet wire wrappings. The magnet will sit on top of the lower

compressional spring. In Fig. 3.2, you might notice that there are two strings, and that is to suspend the springs to the cap of the electromagnetic apparatus. If the springs are not suspended, the springs will move freely and oscillate without solid mounting to any structure and this could cause bottom-outs, and the mass of the spring can chip and crack the magnet, or the magnetic field of the magnet could collapse the springs, and that can halt the oscillation. Zero oscillation will stop current production, and the electromagnetic coil potentially could get damaged with the springs not being suspended.

The other thing that is important about the spring is its spring constant. To determine the appropriate constant (N/m), we use the desired resonant frequency of our system to solve for the needed spring constant for our springs. This formula can be found in Figure 3.2.

$$f = \frac{1}{2\pi} * \sqrt{\frac{k_1 + k_2}{m}}$$

Figure 3.2

Assuming $k_1 = k_2$, we can obtain the resulting spring constant of 245 N/m. However, given that custom springs are expensive and take a long time to make, we decided to select springs with a 210 N/m spring constant instead. This saved us time and money.

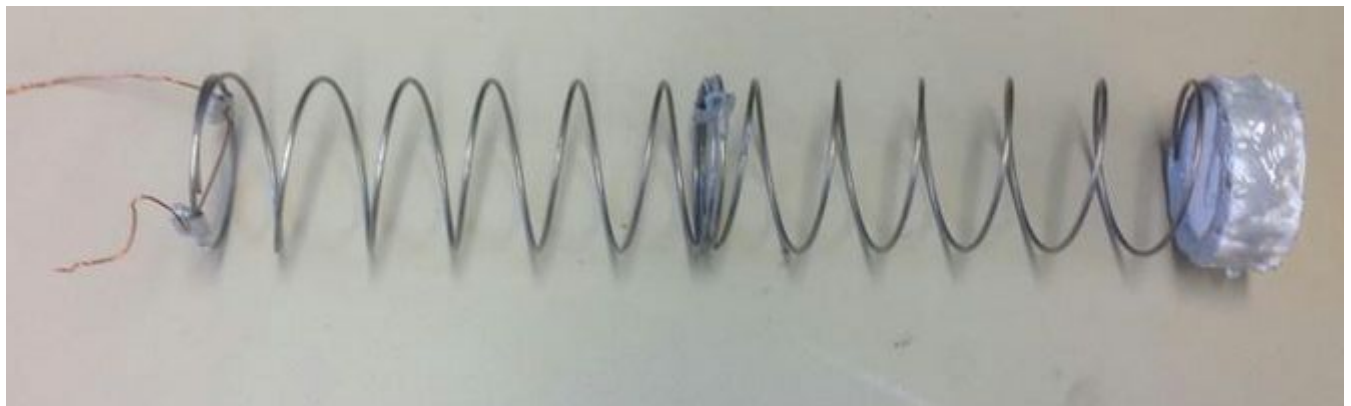


Figure 3.3 - Spring suspension system with pad

(2x) PVC caps (holes drilled to suspend the compression springs) [2.75" {OD}, 2.5" {ID}]

(2x) Impact absorbing foam pads

The foam is inside the PVC cap to help absorb any sudden shock when the spring is oscillating with the magnet. There is foam on the end of the spring that is supposedly in contact with the magnet. The foam on Fig. 3.2 helps keep the metal (from spring) away from the magnet. If the spring magnetizes to the spring, it will create friction between the spring, and the PVC pipe, which will reduce the efficiency and could damage the spring. When the springs are suspended to the cap, friction between the spring, PVC pipe and magnet is hardly noticeable.



Figure 3.4 - PVC Caps



Figure 3.5 - Foam pads

PVC Pipe with coil height of 3.5" [2"(diameter) x 21" (height)]

We cut the pipe to this length, to ensure that the compression springs and the magnet are placed appropriately within the coil height. It is important where the magnet is

placed, especially if you want good RMS current to be produced. Misplacing the magnet out of the coil can dramatically reduce the efficiency. Determining the coil height was determined by the spring length, and magnet height. If the coil height is too high, it is harder to produce current at efficient rate. The magnetic field and inductance works much better when there is a short coil height with a high concentration of windings (multi-layered), and the magnet is able to produce positive and negative polarity when it is moving vertically when vibration is present.



Figure 3.6 - Inductor coil for harvesting

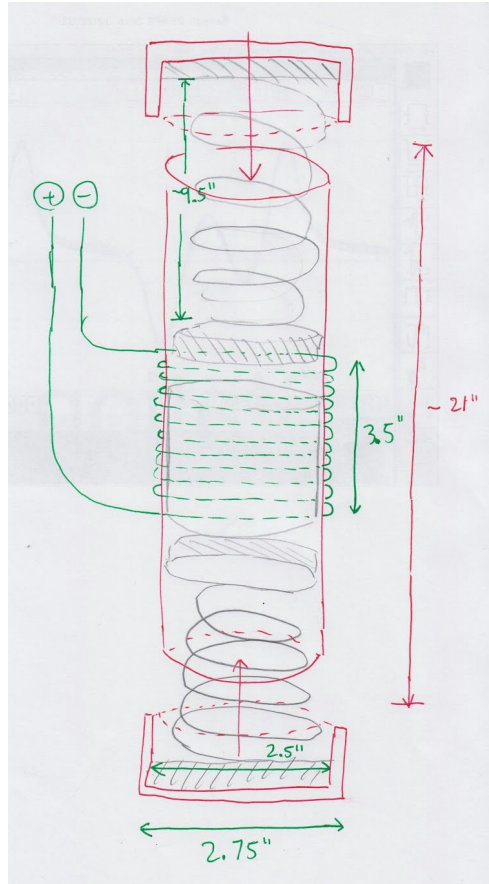


Figure 3.7 - The design layout of the electromagnetic apparatus

In the whole design layout, as seen in Figure 3.7, the magnet is enclosed and sits between the two compressional springs. The displacement of the magnet when the spring is oscillating is about $0.5'' \rightarrow >1''$. Now, this specific design only works effectively at lower frequencies (1 to 10 Hz). After 10 Hz, the amount of RMS voltage will begin to decrease, and less current will be produced. Ultimately, the total power will be reduced to unusable levels. At higher frequencies, the electromagnetic apparatus becomes an inefficient design. In the intended application of this design project, this electromagnetic induction apparatus is suitable.

3.2 Powertrain and Supply

The powertrain subsystem's purpose is to transmit the irregular power delivered from the harvesting apparatus into the battery. The output of the powertrain subsystem is required to output a rated DC voltage so that charging of the battery is consistent and not damaging to the battery.

To accomplish this, the powertrain rectifies the incoming AC voltage into a DC voltage and stores this DC charge in a capacitor as shown in Figure 3.8.

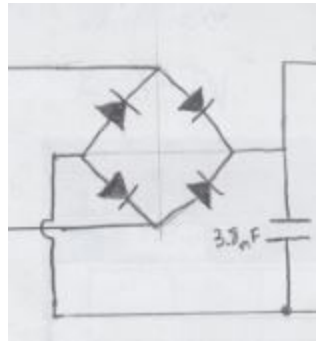


Figure 3.8 - The voltage rectifying circuit

From there the collected DC voltage is run into a boost converter which adjusts the DC voltage to a rated output value of 5 Vdc. This portion of the buck boost converter steps up input voltages ranging from 1.5 Vdc to 5 Vdc to a rated 5 Vdc. Incoming voltages greater than 5 Vdc are passed with little alteration (e.g. 6 Vdc into the boost converter will result in approximately 6 Vdc out). This is shown in Figure 3.9.

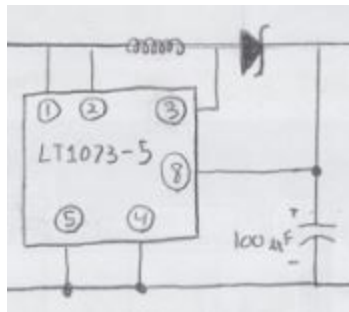


Figure 3.9 - The boost converter circuit

The buck portion of the circuit then steps down the higher voltages produced by the boost converter down to approximately 4 Vdc. The purpose of this buck boost converter is to convert whatever DC voltage is produced, after the rectifier, to a fixed value of 4 Vdc. This fixed value is ideal for charging our battery because it is both effective at charging our particular battery while also not being able to overcharge the battery, potentially damaging it.

A diode is placed in series with the output of the buck converter before the battery to prevent current from the battery leaking backwards into the powertrain while the apparatus is not generating power. This will prevent extra current loss in the battery while our device is inactive. The diode used has a small forward-voltage to reduce

power loss as charge is being stored from the powertrain into the battery. This part of the circuit is shown in Figure 3.10.

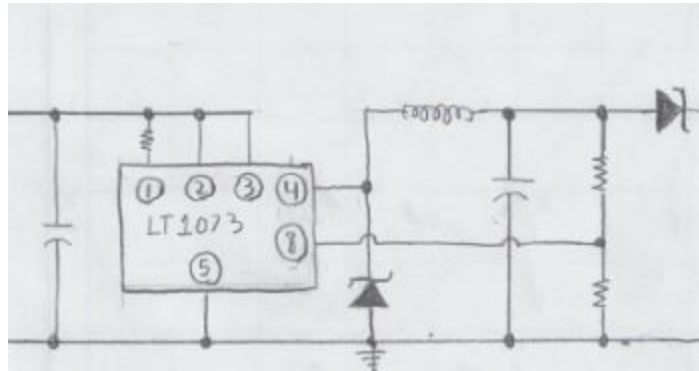


Figure 3.10 - The buck converter that charges the battery

The powertrain subsystem will operate with a microcontroller. This microcontroller requires power to operate. Another buck converter was designed to convert the 3.6 Vdc from the battery to approximately 3.3 Vdc. This buck converter works the same as the last buck converter.

3.3 Monitoring Subsystem

The monitoring subsystem required programming of the microcontroller chosen, the TI MSP430FR6989, in order to measure the power output of the system as a whole. The microcontroller is packaged in a TI Launchpad pictured in Figure 3.11. In order to calculate the power output, the microcontroller needs to measure both voltage and current at a point in the circuit. The most important point to make these measurements at is at the output of the supply subsystem which charges the battery. Measuring the voltage is quite trivial as all that is necessary is to use a voltage divider with a suitable ratio for the microcontroller to measure.

A 2.5 volt reference was chosen as the reference for the analog-to-digital converter (ADC) to use in order to maximize accuracy and reduce the amount of noise measured by the microcontroller. Voltage dividers were placed at the output of the rectifier and at the positive terminal of the battery with ratios of approximately 5.6 and 1.4 respectively. High resistance values were chosen in order to minimize the power loss incurred. The ratio of the voltage divider at the output of the rectifier was chosen in order to minimize the chances that very hard shaking of the vibration harvester would cause voltages that are too high to be input into the microcontroller and possibly cause damage. This is not

a problem at the positive terminal of the battery because it has a fixed maximum output voltage.

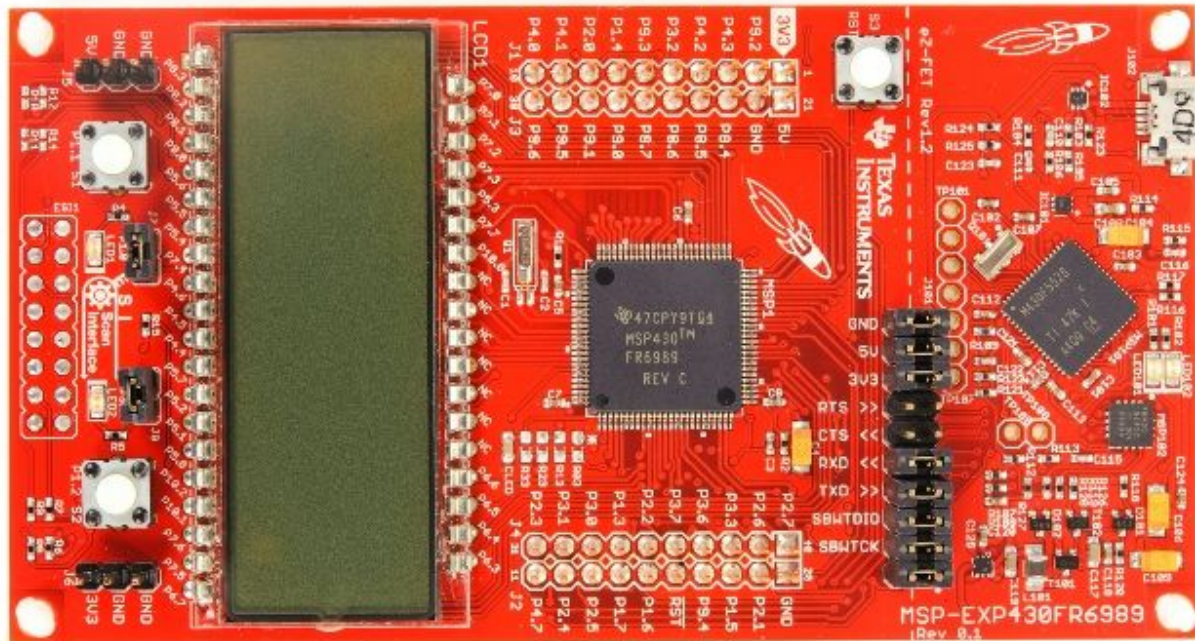


Figure 3.11 - The TI Launchpad containing the TI MSP430FR6989 microcontroller

In order to measure the current at these places in the circuit, we chose to implement a common solution using a current sense resistance. A current sense resistance is typically in the range of a few milliohms to tens of milliohms. When placed in series at the output of a circuit, this resistance produces a voltage drop in proportion to the current passing through it by Ohm's law. However, this voltage drop tends to be very small, on the order of a few millivolts or even less. This kind of measurement range is not feasible for commonly available hardware such as what is found inside of a microcontroller to measure. Another problem with using a current sense resistor is that both ends of the resistor are still potentially at a voltage that is too high for the microcontroller to measure or that could potentially damage it.

Thus, a solution was needed in order to amplify the voltage drop across the resistor to a level high enough for the microcontroller to measure. For this purpose there already exist current sense amplifiers, which are what we decided to implement. After research on the power usage of these amplifiers we chose the Silicon Labs TS1100 amplifier with a gain multiplier of 100 which has what we found to have the lowest power usage on the market.

Since we found that the power delivered by the energy harvester is delivered in bursts and not in a fashion typical of DC power, the microcontroller was programmed to measure the voltage and current using the aforementioned method 32 times per second and calculate the RMS voltage and current using those data points. The microcontroller has a built-in Real Time Clock (RTC) with the ability to use an external 32.768kHz crystal, included in the TI Launchpad, which allows it to take very accurate measurements with respect to time. A photo of the microcontroller measuring the power output of our design is shown in Figure 3.12.

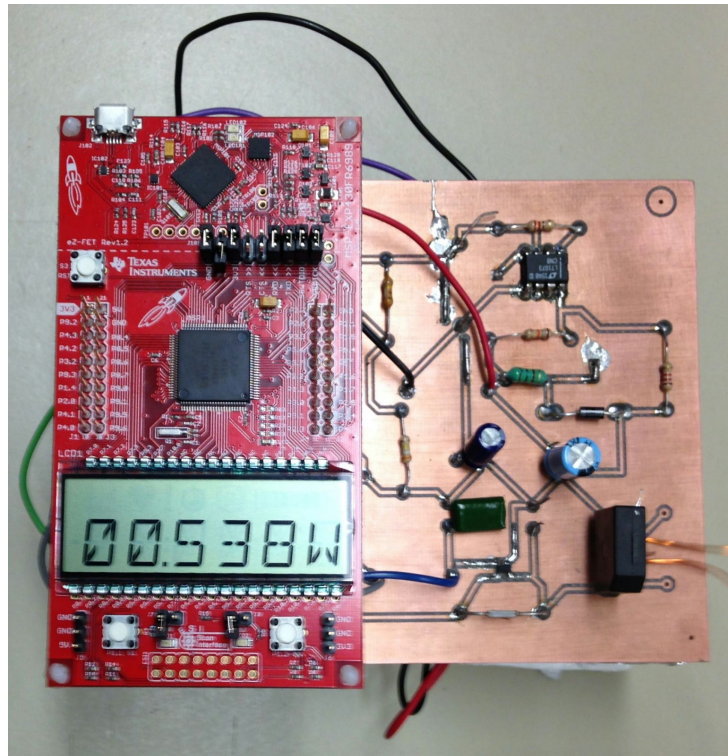


Figure 3.12 - The microcontroller displaying the output generated by the design

For the purposes of testing our design and collecting data, a separate program was developed that would record data points 128 times per second and transmit it to a connected computer via the debug interface on the TI Launchpad. This allowed us to get more granular data when power usage was not a concern which could be interpreted using a spreadsheet program such as Excel.

3.4 Energy Storage and Supply Subsystem

The actions required to develop a functional subsystem, to store electric charge and supply necessary power for the device, revolve around obtaining a rechargeable lithium-ion battery with proper voltage and current ratings. Immediately after verifying

the power demands of the VTP-3Dv2, we purchased batteries that would operate according to our design, provided that ample amounts of energy were charging this system.

The battery itself has been soldered to provide usable terminals for installation into the powertrain circuit and the field device. In order to replace the VTP-3Dv2's power system, the device was opened and the four AA rated batteries were removed. These four batteries are all connected in parallel and so soldering a single terminal for both the positive and negative joints within the device would suffice to allow for external connections. This is shown in Figure 3.13.



Figure 3.13 - Replaced power system for VTP-3Dv2.

Chapter 4 – Test Data With Proof of Functional Design

Our product has been tested throughout several stages of its development and we have documented the results in various formats. Throughout the product's developmental stages we organized testing in a manner in which each of the product's systems could be analyzed individually. This breakdown allowed us to pinpoint issues and tune each system without having to mistakenly alter the functionality of an adjacent system, and allowing said system to alter the results and functionality of the system in question. We worked with the MSU School of Packaging to use their shock and vibration testing equipment to accurately obtain data similar to what we would find in a truck-borne environment. Their laboratory equipment and software allowed us to simulate different frequencies, G-forces, and vertical displacements that have been recorded directly as measurements from environments on actual shipping truck trailers.

4.1 - Vibrational Energy Harvester

The vibration harvesting apparatus underwent testing three times, individually, while various control parameters were implemented to provide broader information regarding the system functionality. The goal of testing for this system was to generate data to determine whether enough power was harvested from a low-frequency vibrating environment to supply the demand for the powertrain, microcontroller, the rechargeable battery, and VTP-3Dv2. As mentioned throughout the report, foam pads are added within the shaft of the vibration harvesting apparatus. These pads have two functions, to negate forces that can damage the device and to alter the vertical equilibrium of the magnet in regards to the height of the exterior coils. Following are detailed test procedures and data evaluations regarding the individual system and integration with the product as a whole.

Testing Control Parameters

- Apparatus vertically fixed to testing hardware
- Mechanical motion is involved only with the magnet and springs
- A resistor of value 676 ohms was used to determine the RMS voltage generated

4.1.1 - Test 1

Control: The apparatus was subjected to a frequency sweep in the range of 3Hz to 12Hz at a constant g-force of 0.5g, specifically (3.817 Newtons on the magnet). The apparatus contained a single foam pad on the lower spring between the spring and the lower PVC cap.

Results: At 3Hz, the lowest the testing equipment can reach, V_{rms} was highest due to 3Hz being most near the resonant frequency of the spring-mass system. The values decreased as the frequency was increased. The V_{rms} waveforms appear to be jagged because, with one foam pad, the vertical equilibrium of the magnet is below or at the bottom edge of the coil. Therefore, a minimum of one-half of the energy developed by the displacement of the magnet is not obtainable through the coil and resistor and therefore disappearing from the signal. This data is shown in Figure 4.1 and Table 4.1.

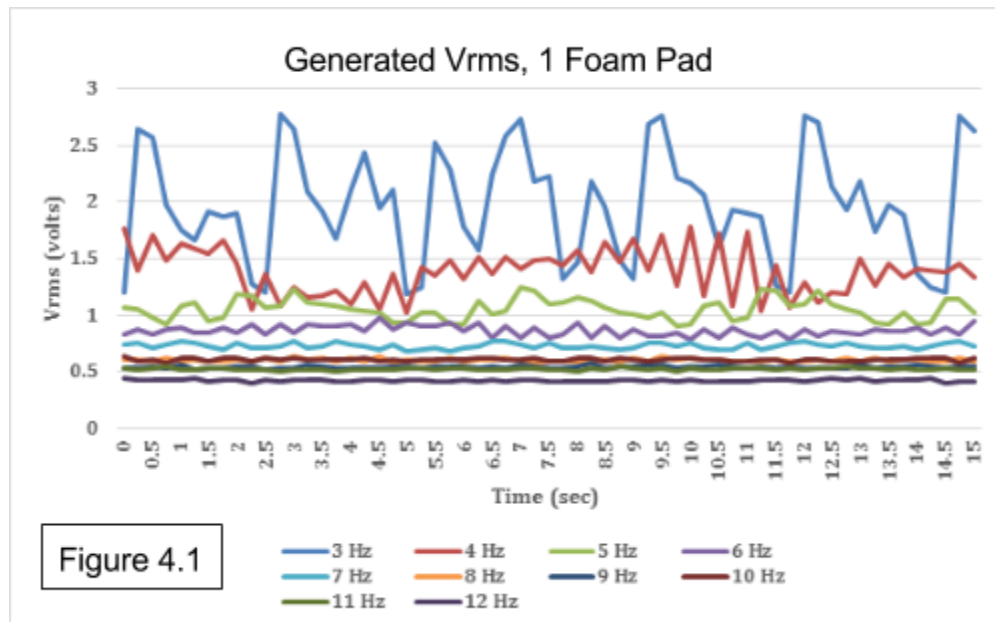


	Table 4.1 - Frequency Sweep (1 foam pad, force=0.5g)									
	3 Hz	4 Hz	5 Hz	6 Hz	7 Hz	8 Hz	9 Hz	10 Hz	11 Hz	12 Hz
Average Vrms (Volts)	1.975	1.389	1.068	0.866	0.729	0.601	0.542	0.605	0.524	0.421
Average Power (mW)	5.769	2.856	1.689	1.110	0.786	0.534	0.435	0.542	0.406	0.262

4.1.2 - Test 2

Including more foam pads to raise the equilibrium height of the magnet, in regards to the coil, would allow the harvesting system to generate higher RMS voltage values. The reason for this is that if the equilibrium point is nearer to the vertical-center point of the coils, the displacement extrema of the magnet will be within the height range of the coils and will allow a bigger induced electric field to be generated. Therefore, more RMS voltage will be developed.

Control: The apparatus was subjected to a frequency sweep in the range of 3Hz to 12Hz at a constant g-force of 0.5g, specifically (3.817 Newtons on the magnet). The apparatus contained three foam pads on the lower spring between the spring and the lower PVC cap.

Results: As shown in Figure 4.2 and Table 4.2, the Vrms generated by the apparatus is noticeably greater across the frequency spectrum, as expected. We can conclude that setting the equilibrium height of the magnet nearer to the vertical-center of the coil will allow the most efficient harvesting of mechanical energy to electrical potential with an electromagnetic induction device. The graph also highlights a major difference between the 3, 4, and 5 Hz waveforms. The 3 and 5 Hz waveforms oscillate sinusoidally, while the 4 Hz waveform values appear to remain relatively constant. The driving factor behind this is our apparatus' resonant frequency. By selecting certain springs based on the weight of the magnet, we were able to tune the resonant frequency to 4 Hz, which is the most common vibrational frequency in our shipping environment. This maximizes our power output in the truck trailer environment.

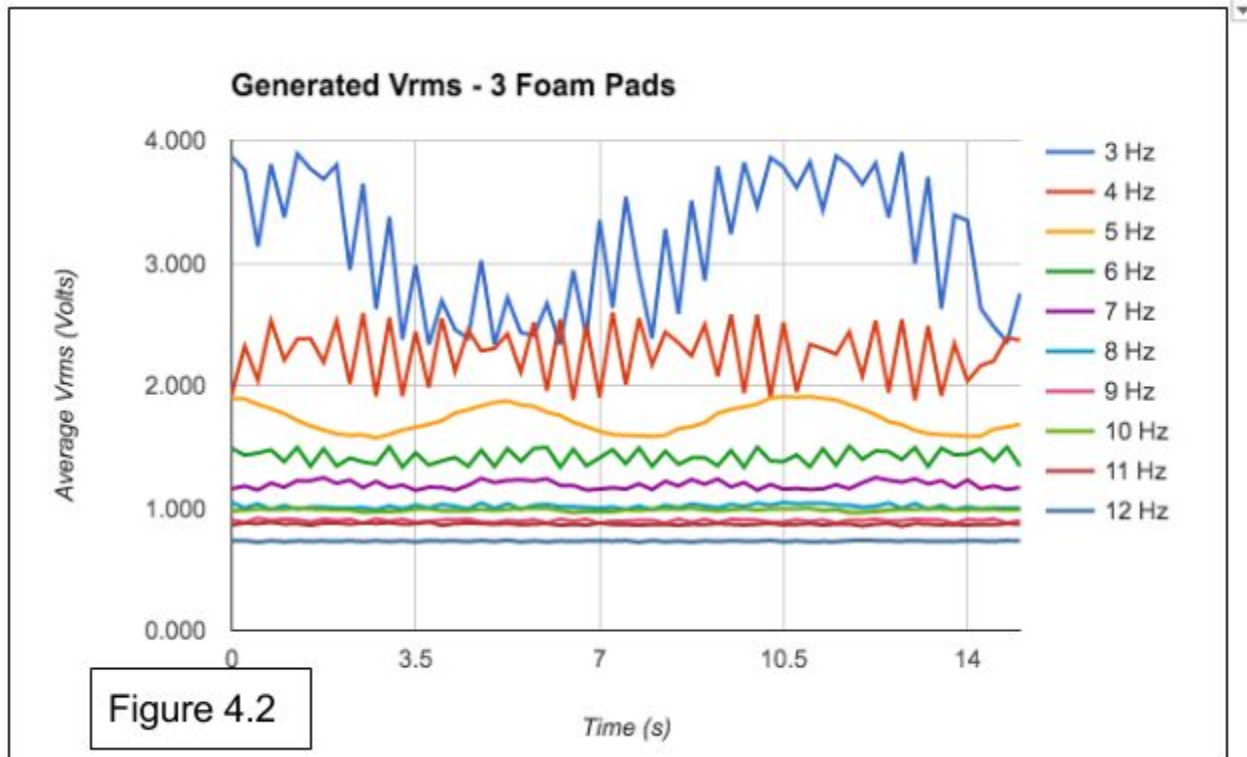


	Table 4.2 - Frequency Sweep (3 foam pads, force=0.5g)									
	3 Hz	4 Hz	5 Hz	6 Hz	7 Hz	8 Hz	9 Hz	10 Hz	11 Hz	12 Hz
Average Vrms (Volts)	3.159	2.262	1.725	1.417	1.189	1.013	0.896	0.984	0.868	0.730
Average Power (mW)	14.761	7.572	4.403	2.971	2.092	1.518	1.188	1.431	1.114	0.788

4.1.3 - Test 3

Control: The apparatus contained three foam pads on the lower spring between the spring and the lower PVC cap. The system was also subjected to random, and varying, frequency and g-force parameters determined by a template developed by ISTA (International Safe Transit Association) to effectively simulate true vibrations found in commercial shipping trucks used today. The simulation was a length of 15 minutes, with an overall grms value of 0.53. Although the simulation length was 15 minutes, this test was made to represent 70 miles of highway driving, or approximately 1 hour of total drive time.

Result: Over the length of this condensed 15 minute simulation, what we can conclude from these results is that the vibration harvesting apparatus is capable of generating enough power to supply the rectification circuit, the microcontroller, charging the battery, and powering the VTP-3Dv2. For approximately 59.2% of the time, the product is able to power the VTP-3Dv2 and sustain its operation with complete dependence on the rechargeable battery. It is required that 1.4Vrms is generated to have successful operation of our design. It is also promising to see that the average value is well enough above the threshold voltage for operation. These results are illustrated in Figure 4.3 and Table 4.3.

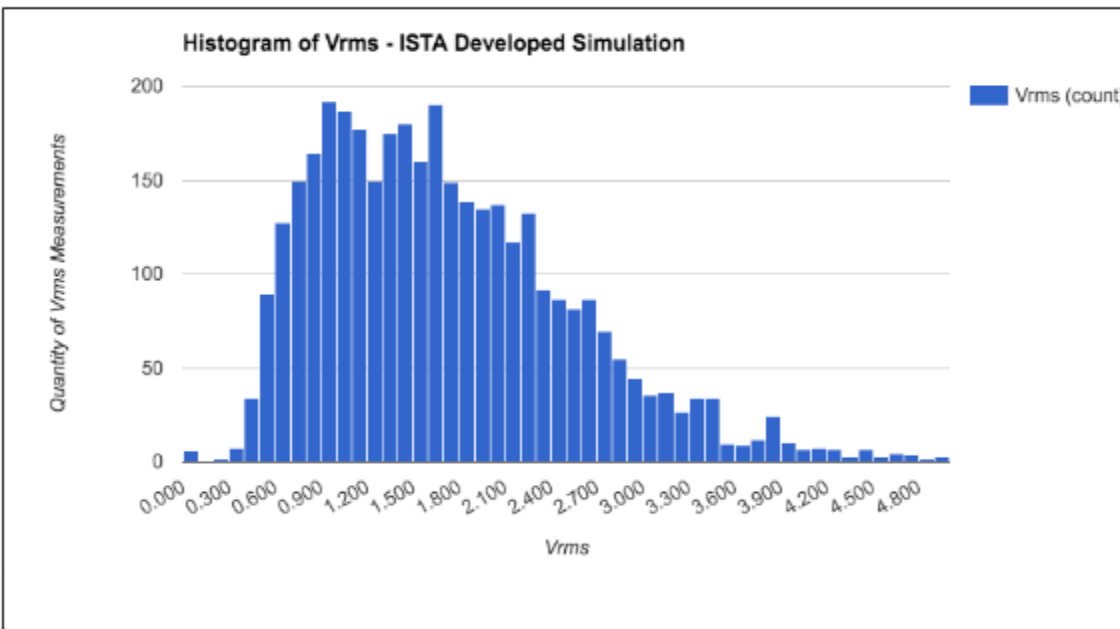


Figure 4.3

Table 4.3 - Specific Data Points of Histogram (3 foam pads, ISTA Developed Simulation)		
Average Vrms	Mode Vrms	Time Beyond Threshold Voltage of 1.4Vrms (%)
1.71	1.63	59.2

4.2 - Powertrain System

The powertrain subsystem is responsible for regulating the incoming power to specific output values to power the microcontroller and charge the battery. The goal of various tests were to analyze to verify reliability of the powertrain output voltage.

4.2.1 - Rectification

First, the incoming alternating current is rectified to eliminate the negative voltages. This was done using a FW100 Bridge Rectifier. We then ran a controlled signal from the function generator into the rectifier and measured the DC voltage output using a DMM. The information was then put into a graph used the data to calculate a line of best fit.

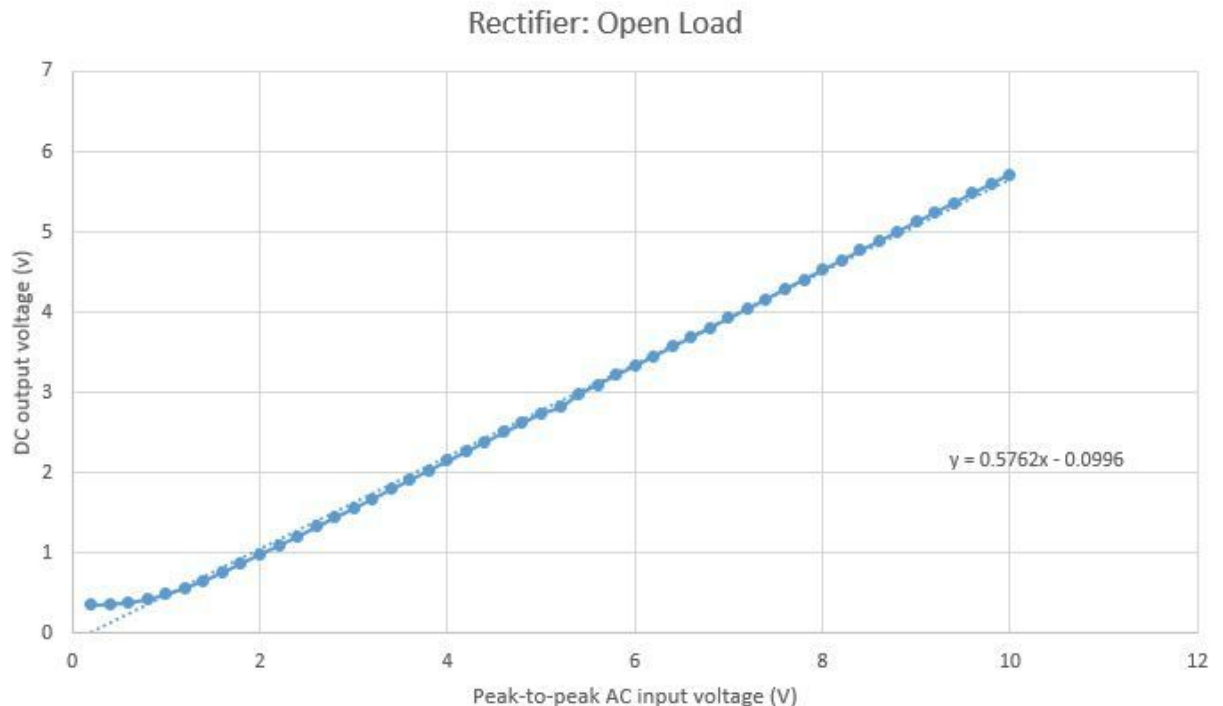


Figure 4.4 - AC Input Open Load Rectifier Voltage Plot

Figure 4.4 illustrates how the measured V_{in} vs. V_{out} values of the rectifier compared to the values generated by the line of best fit. The output voltage values from the rectifier are roughly half of the peak-to-peak AC input voltage values.

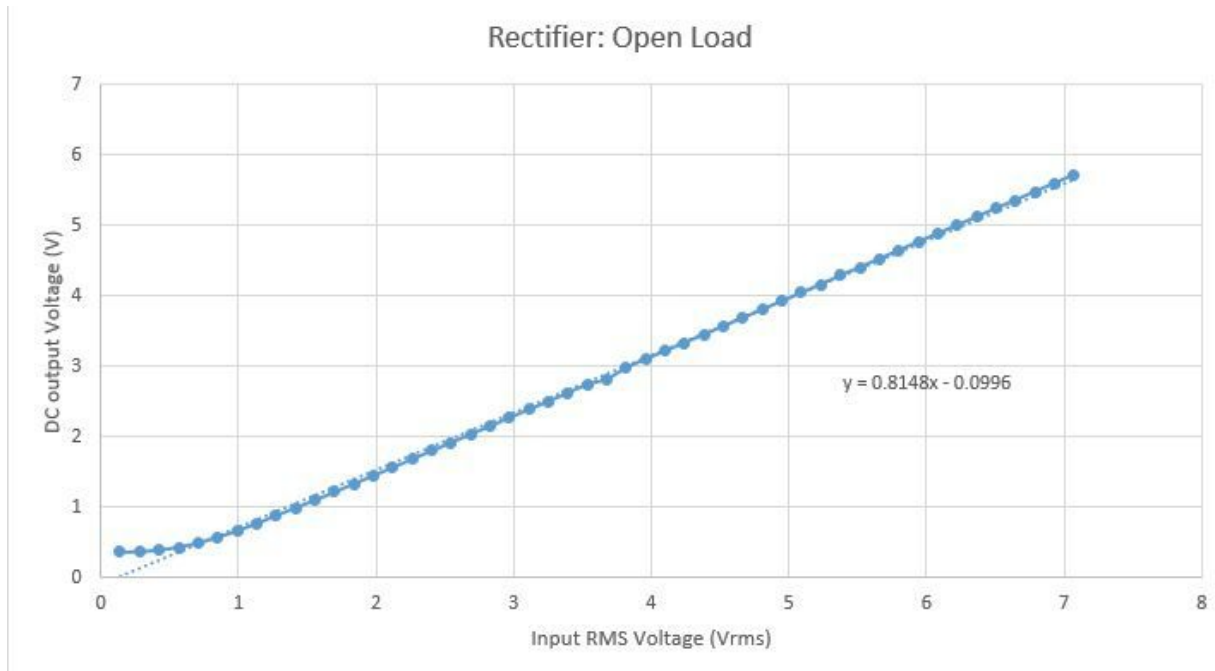


Figure 4.5 - Vrms Input Open Load Rectifier Voltage Plot

We also measured the input Vrms vs output voltage, and plotted a line of best fit. The output voltage seemed to consistently be approximately 81% of the input Vrms voltage. This is shown in Figure 4.5.

4.2.2 - Voltage Conversion

After the voltage is rectified, it needs to be converted into voltages that we can use. Even though the voltage has been rectified, it still has a wide range of values. Referring back to Figure 4.3, we can see that our input Vrms values range anywhere from 0.3 to 5 volts. Using Figure 4.5, we can calculate an approximate rectified voltage range of 0 to 4 volts. Not only does this voltage need to be converted into our needed values, it also needs to be held constant, to protect our circuit components. In order to charge our battery, we need to have 4 volts to overcome the 3.6 volts that the battery outputs. Also, we need to have 3.3 volts to power the microcontroller. Due to our wide range of rectified voltages, we decided to use a combination of boost and buck converters to obtain these necessary voltages. The first boost converter raises the rectified voltage to 5 volts. As shown in Figure 4.6, the converter only activates if the incoming voltage is greater than 1 volt. If it is less than 1 volt, the converter acts as a short circuit.

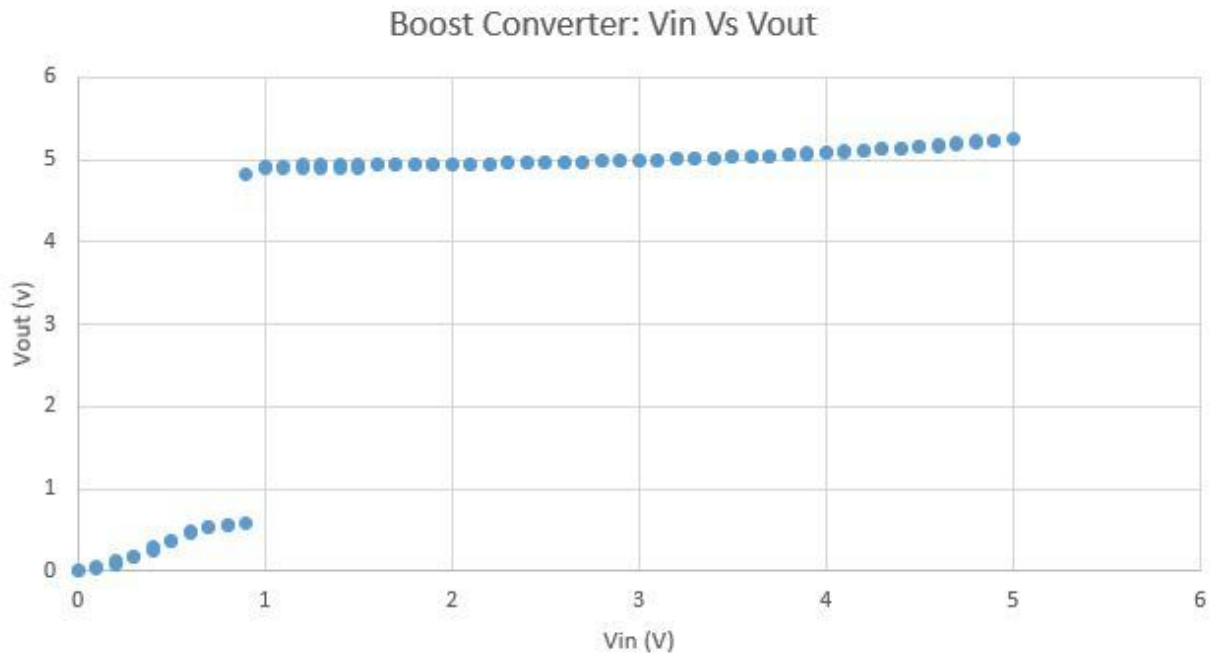


Figure 4.6 - Boost Converter Voltage Plot

After the boost converter, a buck converter lowers the 5 volts to 4 volts. If the input voltage to the converter is lower than 4 volts, the converter becomes a short circuit, and the voltage is passed through. Figure 4.7 shows a graph of the input voltage versus output voltage for the converter. As shown, until the input voltage reaches 0.5 volts, no voltage is passed through, due to the converter diode's forward voltage. When the voltage is between 0.5 and 4 volts, the output voltage is passed through relatively unchanged. After the voltage increases past 4 volts, the converter lowers the voltage down to 4 volts.

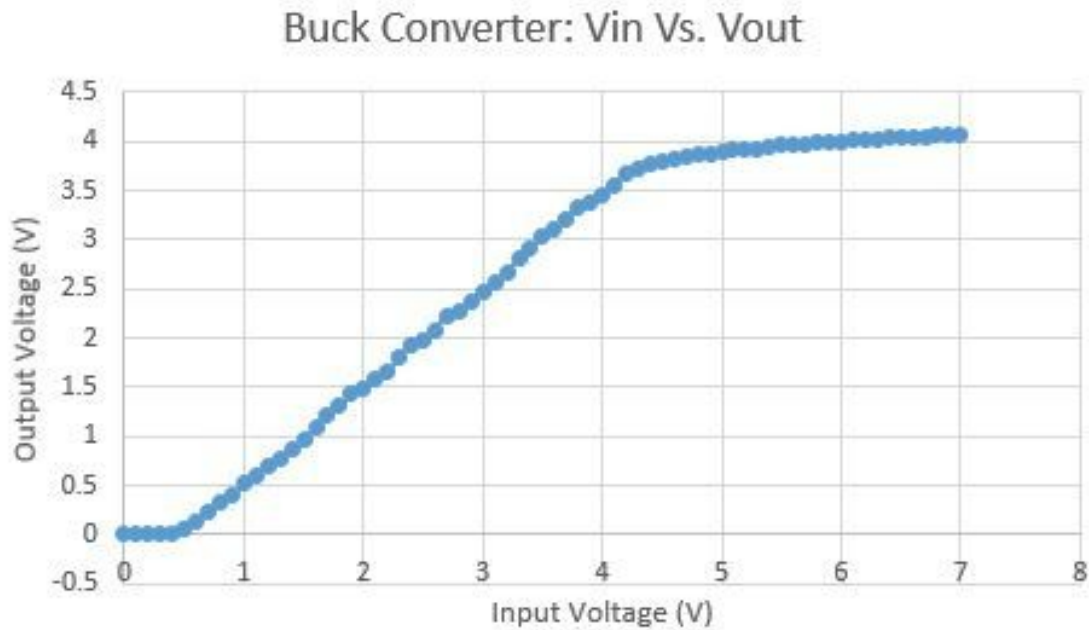


Figure 4.7 - Buck Converter Voltage Plot

In order to stop current from traveling backwards from the battery and into our circuit, a Schottky diode was added in series with the output of the 4 volt buck converter. This diode has a forward biasing voltage of approximately 450 mV. The combination of the Schottky diode and the battery's voltage of 3.6 prevents our buck converter from drawing any current until the incoming voltage reaches 4.3 volts. This is illustrated in Figure 4.8.

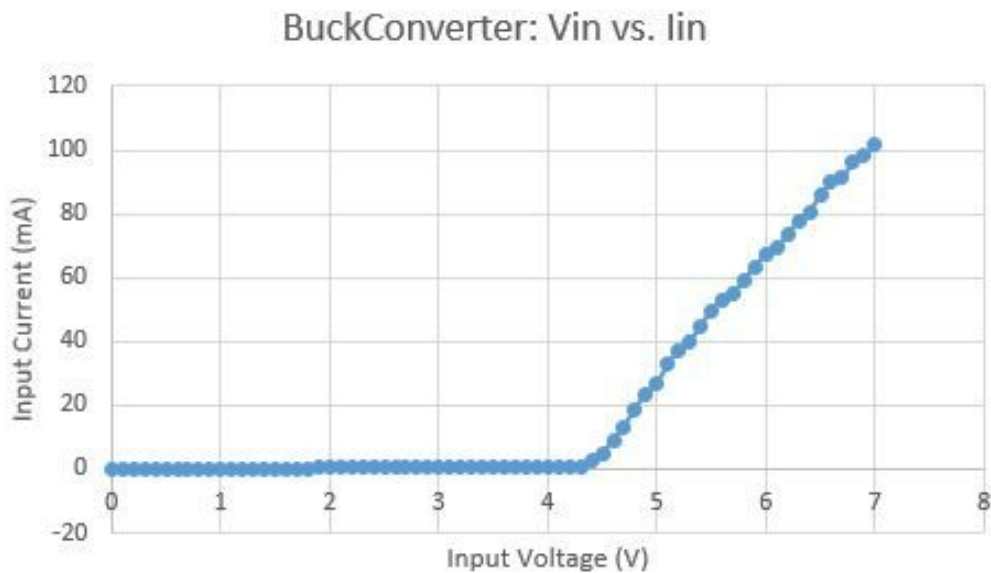


Figure 4.8 - Buck Converter Current/Voltage Plot

After the Schottky diode, there is another buck converter that lowers the voltage down to 3.3 volts to power the microcontroller. This converter's location on our circuit results in a constant source of power for the microcontroller. If our rectified voltage is high enough to recharge the battery, then the 3.3 volt buck converter will draw from the 4 volt buck converter. However, if the rectified voltage is not high enough to activate the 4 volt buck converter, then the 3.3 volt buck converter will draw from the rechargeable battery. This means that the microcontroller will always be running, which is not viable for long term use. That is the main reason why the microcontroller is used as a short-term data gathering device, and can be detached for long-term trips.



Figure 4.9 - Full Device Voltage Plot

Figure 4.9 shows all the circuit component's voltages plotted versus the rectified voltage. Looking at the graph, we can see that our overall rectified biasing voltage is around 1.4 volts. Above this value, the boost and buck converters control the output voltage to supply adequate values to both the microcontroller (3.3v) and the battery (4v). The 3.3 volt source is used to run the monitoring microcontroller, while the 3.9 volt source is used to charge the lithium-ion battery.

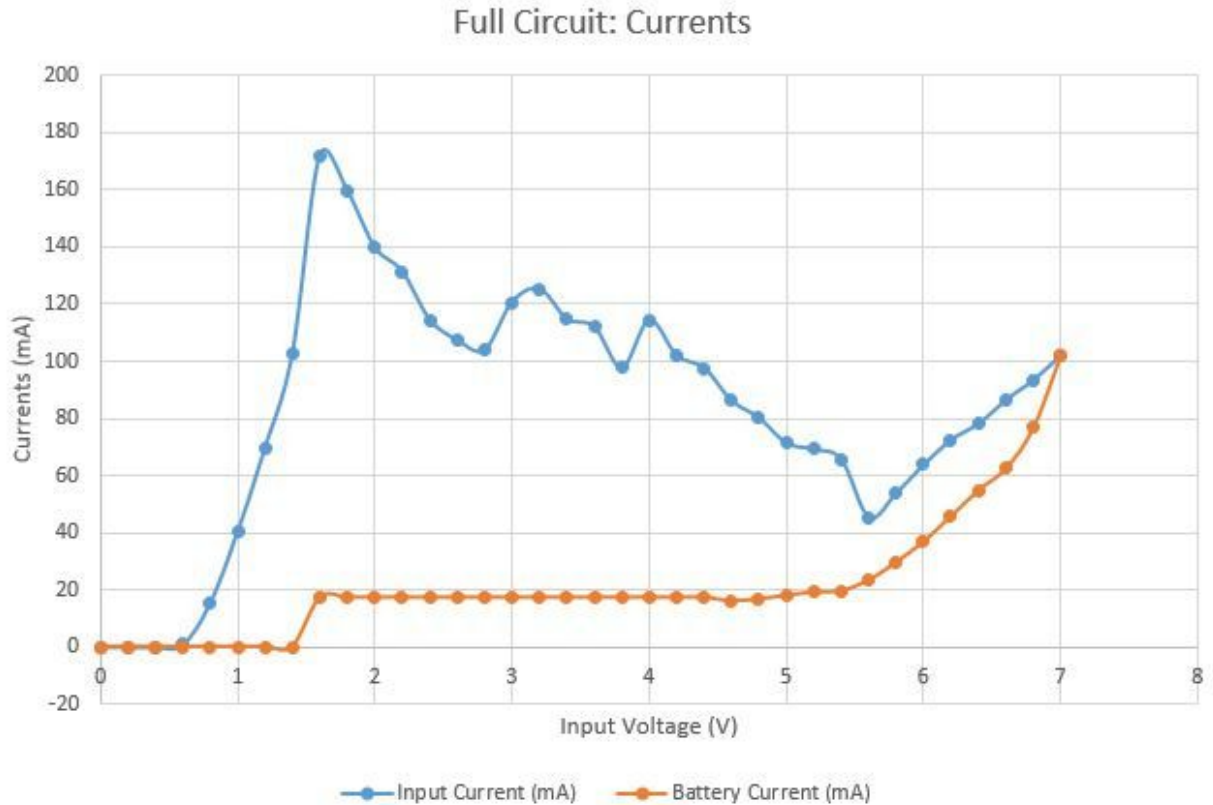


Figure 4.10 - Battery Voltage vs. Input Current Plot

In Figure 4.10, we measured how much current was being charged into the battery, based on the input voltage and current of our powertrain circuit. As stated earlier, we do not charge the battery at all until the input V_{rms} value increases past 1.4 volts. This can be reiterated by tracing the battery's current, and seeing that it remains at 0 until the input voltage rises above 1.4 volts. As shown in Figure 4.10, the charging current remains constant at 20 mA until the input voltage reaches 5.3 volts. Then, the charging current increases parabolically. We do not need to worry about this happening, because as shown in Figures 4.1, 4.2, and 4.3, our generated input voltage never increases above 5 volts.

4.3 - Microcontroller

Testing the microcontroller was quite simple. An input pin was configured to measure the incoming voltage. After connecting the function generator to a 47 nF capacitor, we tested the RMS voltage calculated by the microcontroller versus the RMS voltage measured by an oscilloscope. The frequency we selected was 5 Hz. We connected the

microcontroller's input pin to one end of the capacitor and connected the microcontroller's ground pin to the other end. After connecting the microcontroller to the capacitor, we also connected the oscilloscope to both ends of the capacitor. As the sine wave traveled through the capacitor, we recorded both the microcontroller and the oscilloscope's measurements, and calculated the microcontroller's percent error. This is graphed in Figure 4.11.

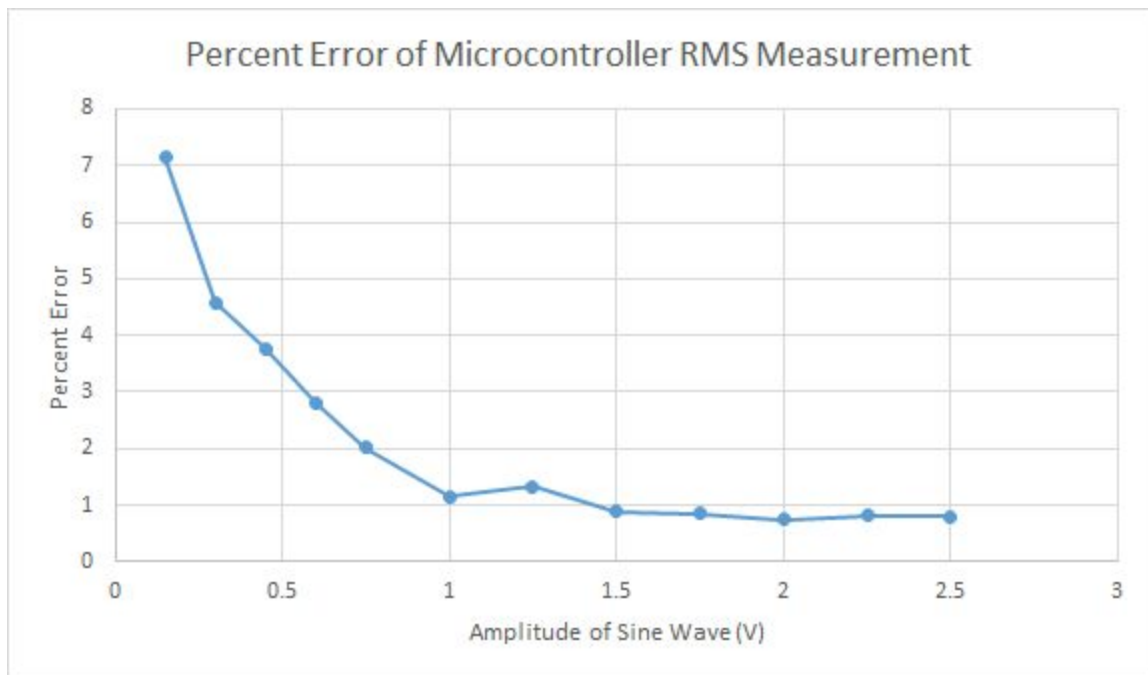


Figure 4.11 - Microcontroller Percent Error Plot

As the measured RMS voltages increase past 1 volt, the percent error of the MCU levels off at approximately 0.8%.

4.4 - Entire Device

4.4.1 - Test 1

Control: The entire device was subjected to a frequency sweep in the range of 4Hz to 5Hz at a constant g-force of 1g, specifically 7.632 Newtons on the magnet. The apparatus contained three foam pads on the lower spring between the spring and the lower PVC cap. The microcontroller was used to measure the current flowing into the battery, and the battery's voltage.

Results: The results of this frequency sweep are shown in Figures 4.12-14. Using the measured current value flowing into the battery, and the battery's current voltage, we

were able to calculate the power flowing into the battery during the entirety of this test. Figure 4.12 shows the power flowing into the battery during 2 seconds of the 4 Hz section of the frequency sweep.

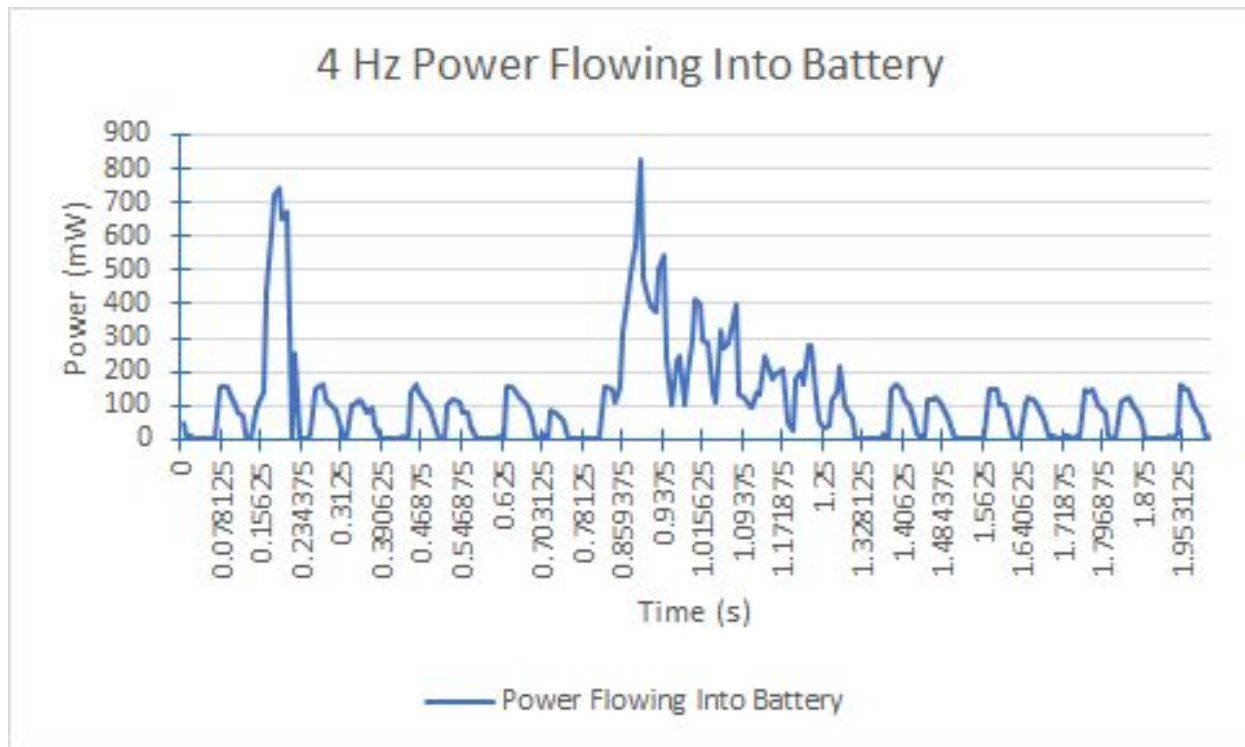


Figure 4.12 - Complete Device 4 Hz Power Output Plot

Figure 4.13 shows the power flowing into the battery during 2 seconds of the 4.5 Hz section of the frequency sweep.

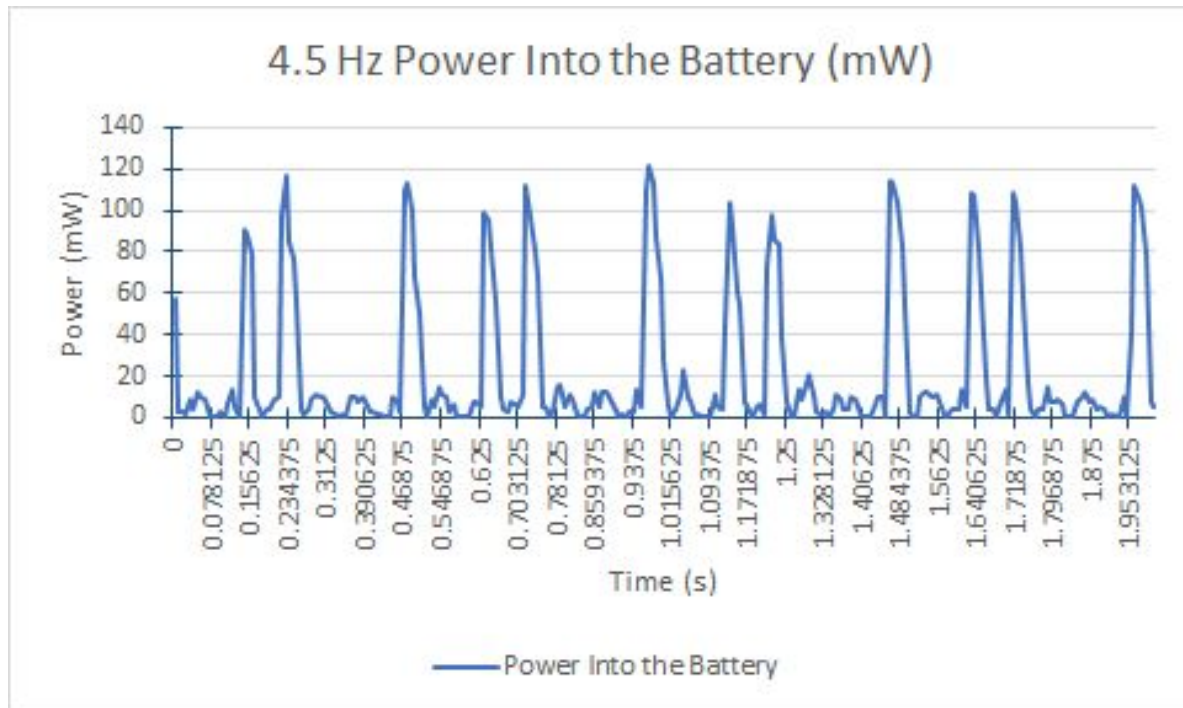


Figure 4.13 - Complete Device 4.5 Hz Power Output Plot

Figure 4.14 shows the power flowing into the battery for a two-second portion of the 5 Hz section of the frequency sweep.

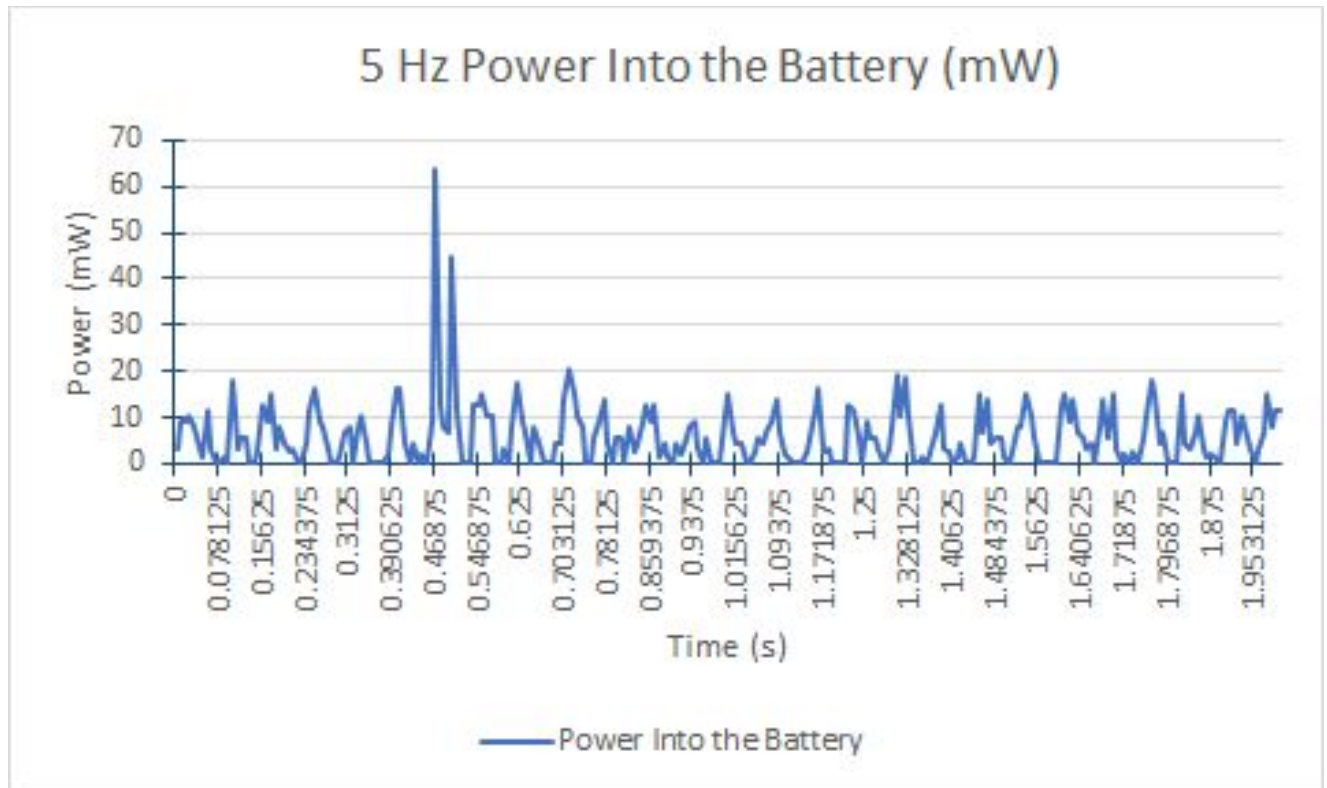


Figure 4.14 - Complete Device 5 Hz Power Output Plot

4.4.1 - Test 2

Control: The entire device was subjected to a 5 minute ISTA 3A over-the-road truck trailer test. The apparatus contained three foam pads on the lower spring between the spring and the lower PVC cap. The microcontroller was used to measure the battery current and the battery voltage.

Results: Our device performed well throughout the test. The device remained above the threshold voltage 56.343% of the time, which confirms the data measured in Test 3. At the end of the test, we observed that the battery's voltage had surpassed its initial level by 2 millivolts, which is indicative of the success of our project. Using our battery current and voltage values, we calculated the power flowing into the battery. Figure 4.15 graphs a two-second excerpt from our test, along with some data shown in Table 4.4.

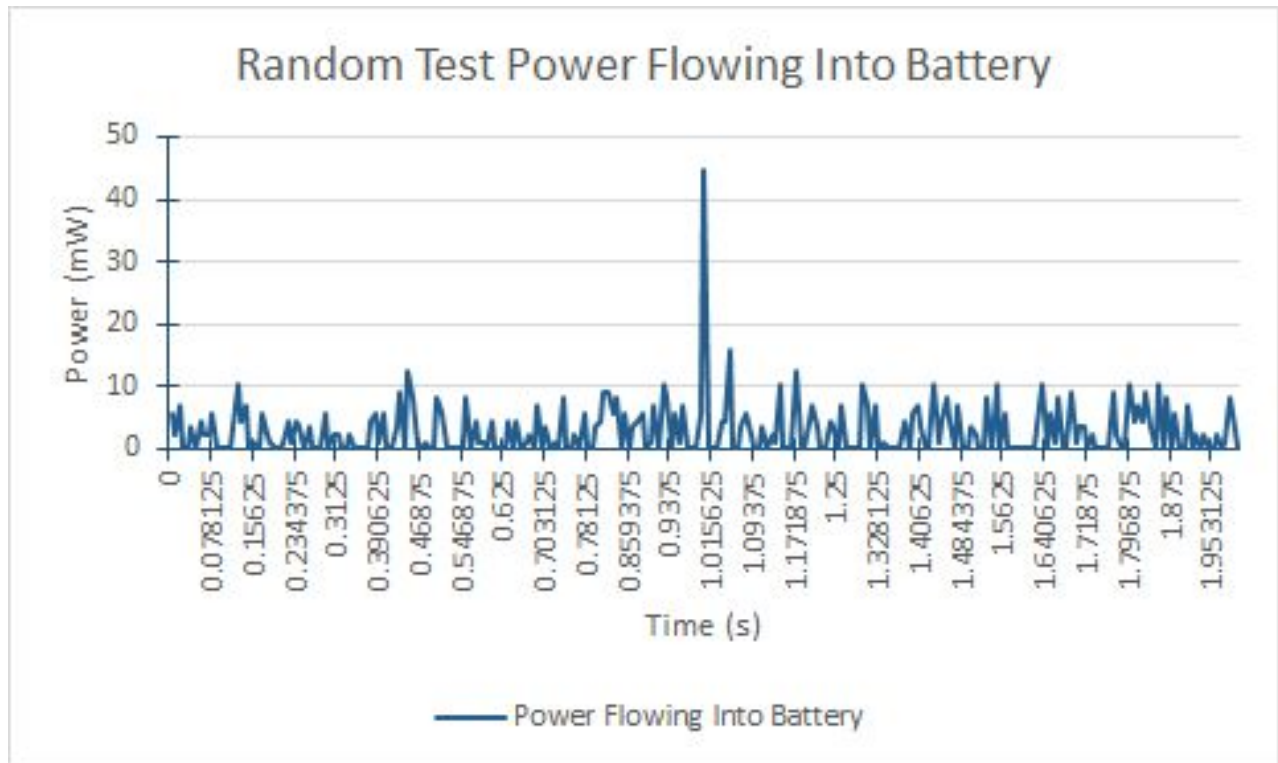


Figure 4.15 - Complete Device ISTA Power Plot

Table 4.4 - Line Plot of Charging Power (5 minute ISTA Developed Simulation)	
Overall Change in Battery Voltage	Percentage of Total Time Charging the Battery
+2.81 mV	56.342

4.5 - Proposal Design Specs

- Device should be able to supply power to the VTP-3Dv2 device:

Our device can adequately power the VTP-3Dv2 device. It converts the incoming AC into a stable and usable voltage that powers the VTP-3Dv2, with the excess being stored in a rechargeable battery. Our device is portable and can exist safely in an automotive environment.

- Device should provide a rated output voltage of 3.6v:

The power output of our device is 3.6 volts when in operation. When dormant, the device outputs 0 volts, and the VTP-3Dv2 runs off the rechargeable battery, along with its own personal battery power.

- Output a current of 5 mA:

In accordance with the data in Table 4.4, our device is able to surpass its threshold voltage approximately 56% of the time, which means it can power the VTP-3Dv2 for roughly half of the drive time. However, while the truck is at rest, our device will generate no power, which will result in the draining of the rechargeable battery.

- Operate at low frequencies (under 10 Hz):

Using data provided by IST, we deduced that the most powerful vibrational frequency for truck-trailer vibrations exists at 4 Hz. Knowing the mass of our magnet allowed us to calculate the optimum spring constant values in order for our device to resonate at 4 Hz. The success of this resonance is clearly evident in Figure 4.2, which accentuates the difference between 3, 4, and 5 Hz oscillations.

- Recharging device (vibration harvester) should be able to assist in recharging VTP-3Dv2 and help extend the time of use beyond 3 months, or even reduce the need of batteries in the VTP-3Dv2.

As stated earlier, our device will effectively be able to completely power the VTP-3Dv2 while the device is in transit for approximately half the drive time. This will extend the time of use past 3 months, and it will definitely reduce the need for batteries in the VTP-3Dv2.

- Re-charging generator is attachable/detachable:

Our device is easily attachable and detachable to/from the VTP-3Dv2. The only limitation our device faces is its large size in comparison to the VTP-3Dv2.

- Operational in an automotive environment:

Our device's parts are all rated to operate at temperatures from -40C to 125C. It can also operate under most humidity ranges.

Chapter 5 – Design Issues

5.1 Product Liability

We are trying to extend the operational life of a wireless sensor device called the VTP-3Dv2. This device runs off of four AA lithium batteries rated at 3.6v and has an estimated lifespan of approximately 3 months. To extend this lifespan of the VTP-3Dv2 we are harvesting vibrational energy from the environment around it and converting this energy into electrical power to be stored for later in a rechargeable battery. However, there are a few things which can compromise our vibration harvester and the system on which it is implemented.

All batteries have a charge capacity, or a limit to how much power can be stored within them. When too much charge is stored within the battery there is a possibility of damaging the battery itself and potentially anything that is connected to the battery's terminals, in our case the VTP-3Dv2. The battery which we are using in our design has a possible risk of being overcharged. While operating accordingly, our vibration harvester will typically output around 10 mA of current into the battery while the VTP-3Dv2 draws only 5 mA. This could potentially cause the battery to store charge until it finally reaches capacity and begins breaking down.

To prevent this from happening we have rated the output of our powertrain circuit to provide a safe DC voltage level that will greatly reduce the potential of such issues from ever occurring. The battery which we are charging with our vibration harvester is rated for 3.6 volts, but it can vary depending on the amount of charge it has stored within it. As more charge is stored the voltage value of the battery increases, reaching a maximum voltage value of 4.2 Vdc. Using this to our advantage, we rate the output value of our powertrain to be approximately 4 Vdc. At this level, we can still charge the battery at its rated voltage level of 3.6 Vdc but the circuit is unable to keep providing charge once the battery exceeds the 4 Vdc limit.

However, our vibration harvester is capable of varying the rated output depending on the vibration input. Strong enough vibrations can produce spikes of power which can push the rated output voltage of our powertrain higher than 4 Vdc and supply output currents greater than 100 mA into the battery. These spikes have the potential to push our battery's storage capacity beyond the safety limit and potentially damage the device. There is not much we can do to prevent this problem simply, but lower the

likelihood of its happening. With the way our design is set up the chances of this happening are so small it is basically nonexistent.

5.2 Universal Design Principles

Our device is currently designed for a specific application, to supply power to a device called the VTP-3Dv2, developed by Instrumented Sensor Technology. The VTP-3Dv2 is meant to operate in a shipping truck and our vibration harvester is designed to supply power to the VTP-3Dv2 with power by collecting mechanical energy from the vibration environment characteristic to that of a shipping truck and converting that energy into electrical power. These circumstances have made our device limited in its application and accessibility. However, this does not mean that our design can not be modified to meet the needs of wider range of issues.

Wireless sensor devices are everywhere and anywhere there is a battery powered device in the presence of mechanical motion, or vibration, our design can be simply altered to better fit that specific application. The concepts behind our project are very simple and easily repeatable. It would not be difficult to diversify the potential of our design to meet a more widely recognized group of application.

Chapter 6 – Final Cost, Schedule, Summary and Conclusion

Overall, this project was very successful. Initially, the beginning of this project was mainly all research and brainstorming. The original design of the vibrational apparatus and powertrain circuit were completely different from what we are using now. The idea of using electromagnetic induction as a method of energy harvesting was still going through trial and error, and the powertrain circuit was originally a buck-boost converter. The energy harvesting apparatus was designed and wound to see if it was working, but it had to be re-designed in order to harvest more energy, and later on the consideration of moving parts and risks came into the full development of the electromagnetic apparatus.

The apparatus resonates at our target frequency of around 4 Hz, which maximizes the power output at the most common vibrational frequency in the shipping truck environment. Our initial vibration harvesting apparatus did not supply adequate power to be feasible for our project goals, so we upgraded the strength of the magnet along with increasing the size of the device.

The powertrain circuit converts the raw current from the apparatus into controlled voltage and current levels, which in turn power the VTP-3Dv2, with the excess energy being stored in a rechargeable battery for future use. We are able to effectively charge the battery over long periods of time and have implemented a high degree of safety into our design to prevent overcharging of the battery.

Initially, we were planning to use the microcontroller to control the charging of the battery. The microcontroller would use PWM to supply the necessary charge to the battery, based on the current charge level of the battery. This idea was quickly abandoned after testing revealed how difficult it would be to implement. Instead, we used a combination of boost and buck converters to convert the voltages into our needed values. This proved to be much more efficient.

Our product effectively prolongs the operational period of IST's field device. As a result, we believe this product is testimony towards the viability of harnessing vibrational energy for sustainable-portable energy to help meet the demand for these reliable energy sources.

Appendix

Appendix 1 – Technical roles, responsibilities, and work accomplished



Team members left to right: Stephen Pliska, Giovanni Ivanaj, Andrew Schroeder, Daniel “Tom” Dochoda, Daniel Fresneda-Rojas

Stephen Pliska

I was responsible for developing the powertrain circuitry which connects the harvesting apparatus to the power cell where power is stored. I designed the original macro design and schematic for the powertrain circuit presented in our proposal towards the beginning of the semester. I researched the necessary parts and calculated the appropriate values for each part in order for the circuit to work effectively. I then composed a bill of materials and order all the parts required.

I built the first prototype of our powertrain circuit and verified that it worked but not as effectively as I had expected. The circuit only worked with a select range of inputs and was not efficient with power. I calculated that the circuit itself require far more power to operate than what we were trying to supply. This prototype was only able to output a very small fraction of what we had put in thus making it impractical for our application. I then did some research to find an alternative solution with a similar function over a wider range of function with greater power efficiency.

I developed a new schematic with the improvements which I calculated and ordered the required parts. Once I had collected all the necessary parts I built the new prototype. This time the prototype worked over the entire range of function which we were trying to

apply to our circuit. I collected data on the function of our circuit to verify that it worked properly and effectively. I made a few corrections to optimize the efficiency of the circuit.

Once I had finished optimizing the powertrain circuit I implemented it into the rest of the project to verify that was capable of working appropriately in the entire design. I helped perform tests on the entire project and verified that our vibration harvester can get charge into a battery.

Andrew Schroeder

My technical responsibilities included the physics calculations for the electromagnetic induction apparatus. I examined the vibrational data given to us by IST, and calculated the nominal spring constants for our desired resonant frequency. I also calculated estimations for our peak-to-peak voltages and power outputs. After receiving the parts for the apparatus, I assisted Tom with its construction. I developed the spring attachment assembly, along with the idea of using 2 springs on each side of the magnet to increase the total amount of force our device can safely handle. I also created foam inserts to raise and lower the magnet's equilibrium position, allowing us to determine which position generates the most power. When our magnet's overabundance of strength prevented the springs from fully extending, I came up with the plan to use foam inserts to act as a barrier between the springs and the magnet.

After completing the construction of the electromagnetic induction apparatus, I designed and produced testing plates which allowed our apparatus to be bolted onto the testing shaker located at IST. Also, I painted the metal housing for the PCB.

Daniel "Tom" Dochoda

My technical role to this design team was the research and development of the mechanical → electrical harvesting apparatus (electromagnetic induction apparatus). I came up with the design and idea of using the magnet and coil method of energy harvesting, due to the application of low frequency. I did some research on various methods of vibration energy harvesting prior to designing the electromagnetic apparatus, to see which method is appropriate. I also did research on Faraday's Law of induction, and assisted in rough calculation with Stephen Pliska, to see if the method of using electromagnetic induction was possible. At the end of the day, electromagnetic induction apparatus was the final decision.

I worked closely with Stephen for improving and optimizing this apparatus, and making any design changes to it. The original electromagnetic apparatus was built to ensure that the method of choice was feasible and it worked, and over time, simple changes

have been implemented. The original electromagnetic apparatus had very low efficiency due to the placement of the magnet, the erroneous coil length, and the lack of stiffness in the spring(s). As improvements were being made, many of the flawed designs have been fixed and rectified. Andrew Schroeder helped supply new parts to the electromagnetic apparatus, and assisted in assembly and purchases of new parts. Currently, the improvements to the device have been implemented and have been tested.

I also designed the printed circuit board (PCB) for Stephen's powertrain circuit design. The computer aid software, EAGLE was used to design the Gerber files for the printing of the circuit. Designing the PCB was not complicated, but getting a grasp with EAGLE software took some time, but at the end I was able to get a circuit board printed. Stephen soldered on the components to the PCB, and initial tests were performed. The PCB has been tested for results and functionality and the PCB is fully functioning. The goal for the rest of the project is to assemble every portion of the circuit that has been developed, tested and improved.

Daniel Fresneda-Rojas

My role in the project mostly dealt with the battery charging circuitry and the monitoring subsystem. The initial design for charging the battery in our circuit consisted of methods that through testing I found to not be suitable for our purposes. I then contributed to the designing and testing of a new method for charging the battery and picking the appropriate components. We found this new method of using buck-boost converters to be sufficiently appropriate for us to use in our design. The initial idea was to have a microcontroller control the duty cycle of a PWM signal that would drive a switch in a buck-boost converter. However, after testing this idea using the microcontroller we found that it would be simpler to use a commercially-available buck-boost converter chip instead.

In dealing with the monitoring subsystem, I researched various ultra-low-power microcontrollers and chose one that would be appropriate for our design, the TI MSP430FR6989. I then designed and coded software that accurately reports the power generated by our design. During this process, I needed to read the user guide and the datasheet of the microcontroller extensively to configure the various modules of the microcontroller properly.

In order to allow the microcontroller to measure the current produced from the vibration harvester, I researched various ways to measure current passing through a circuit. This culminated in choosing an appropriate current sense amplifier which uses very little power in combination with a very small resistance which also incurs very little power

loss. I also tested these parts and did the calculations in order to make the readings accurate. At the end of the semester, I was responsible for designing a way to collect data from our circuit using the microcontroller.

Jovanni Ivanaj

I am primarily focused on the functions of the project that relate to the charging of the battery, and also the power discharge, or supply, for the VTP-3Dv2. The rechargeable battery design was chosen by myself in order to ensure that the nominal voltage, current, and capacity ratings were able to energize the mentioned device. I did research to find a data sheet relating to the specific battery model, and continued to test the charge and discharge rates in order to check the validity of the datasheet provided by the manufacturer. Keeping above the nominal discharge voltage of the battery was the goal to reach for the powertrain. I replaced the VTP-3Dv2's original battery system to be compatible with our own. This task included soldering and design that allowed for connection and disconnection of the device and rechargeable battery.

Also, I worked with Steve on the buck-boost converter circuits that were necessary to provide a DC voltage in order for the battery to charge. Through the implementation of this converter, there was too much power loss in specific circuit diodes; therefore, I suggested use of a Schottky diode and provided evidence that this circuit element would suit our needs far better. I was also our team's lead for establishing meetings to have our product qualitatively and quantitatively evaluated. I have been working with my team and provide assistance for various tasks.

Appendix 2 – Literature and website references

“Introduction to Vibration Energy Harvesting”

by. Francesco Cottone

<http://www.nipslab.org/files/file/nips%20summer%20school%202011/Cottone%20Introduction%20to%20vibration%20harvesting.pdf>

<https://www.boundless.com/physics/textbooks/boundless-physics-textbook/induction-ac-circuits-and-electrical-technologies-22/magnetic-flux-induction-and-faraday-s-law-161/faraday-s-law-of-induction-and-lenz-law-569-6693/>

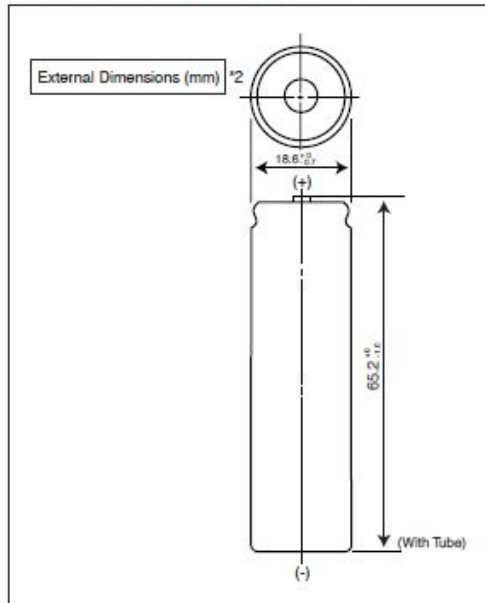
<http://hyperphysics.phy-astr.gsu.edu/hbase/electric/farlaw.html>

<http://web.mit.edu/viz/EM/visualizations/coursenotes/modules/guide10.pdf>

Nise, Norman. *Control Systems Engineering, Sixth Edition*. Hoboken NJ: John Wiley & Sons, 2011.

Data Sheet - Rechargeable Lithium-Ion Battery

CGR18650CG: Cylindrical Model



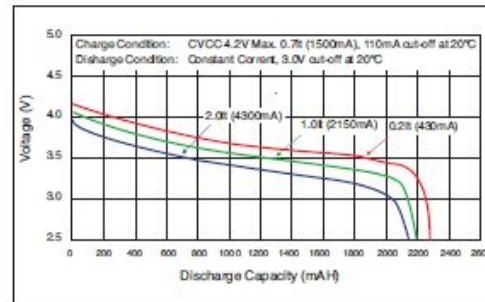
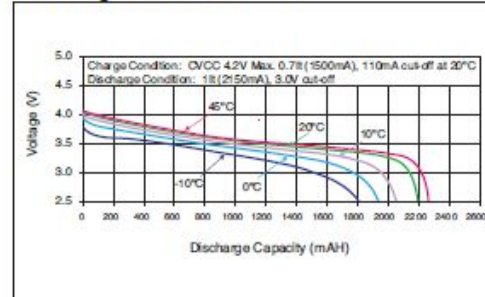
To ensure safety, the referenced Li-ion cell is not sold as a bare cell. Li-ion cells must be integrated with the appropriate safety circuitry via an authorized Panasonic Li-ion pack assembler.

Specifications

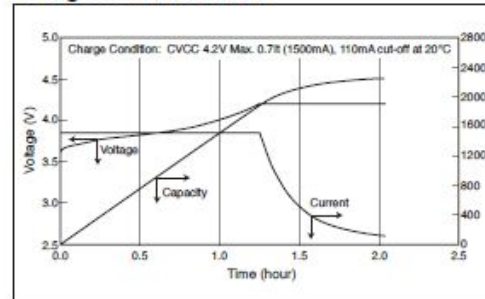
Nominal Voltage		3.6 V
Standard Capacity ^{*1}		2250mAh
Dimensions ^{*2}	Diameter	18.6 ± 0.7mm
	Height	65.2 ± 0.1mm
	Weight	Approx. 45g

- *1 After a fresh battery has been charged at constant voltage/constant current (4.2 V, 1500mA (max), 2 hours, 20°C), the average of the capacity (ending voltage of 3 V at 20°C) that is discharged at a standard current (430mA).
- *2 Dimensions of a fresh battery

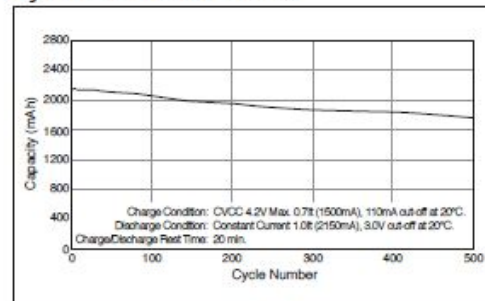
Discharge Characteristics



Charge Characteristics



Cycle Life Characteristics



Panasonic

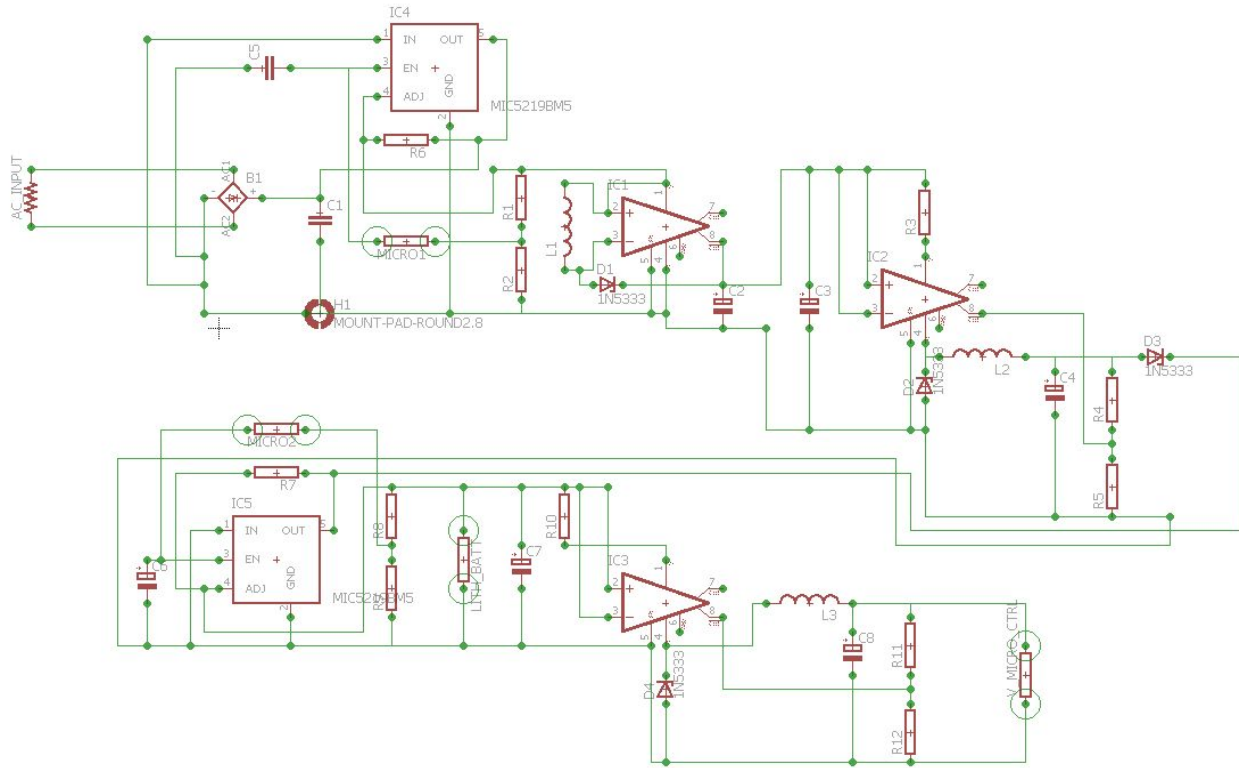
LITHIUM ION

DECEMBER 2008

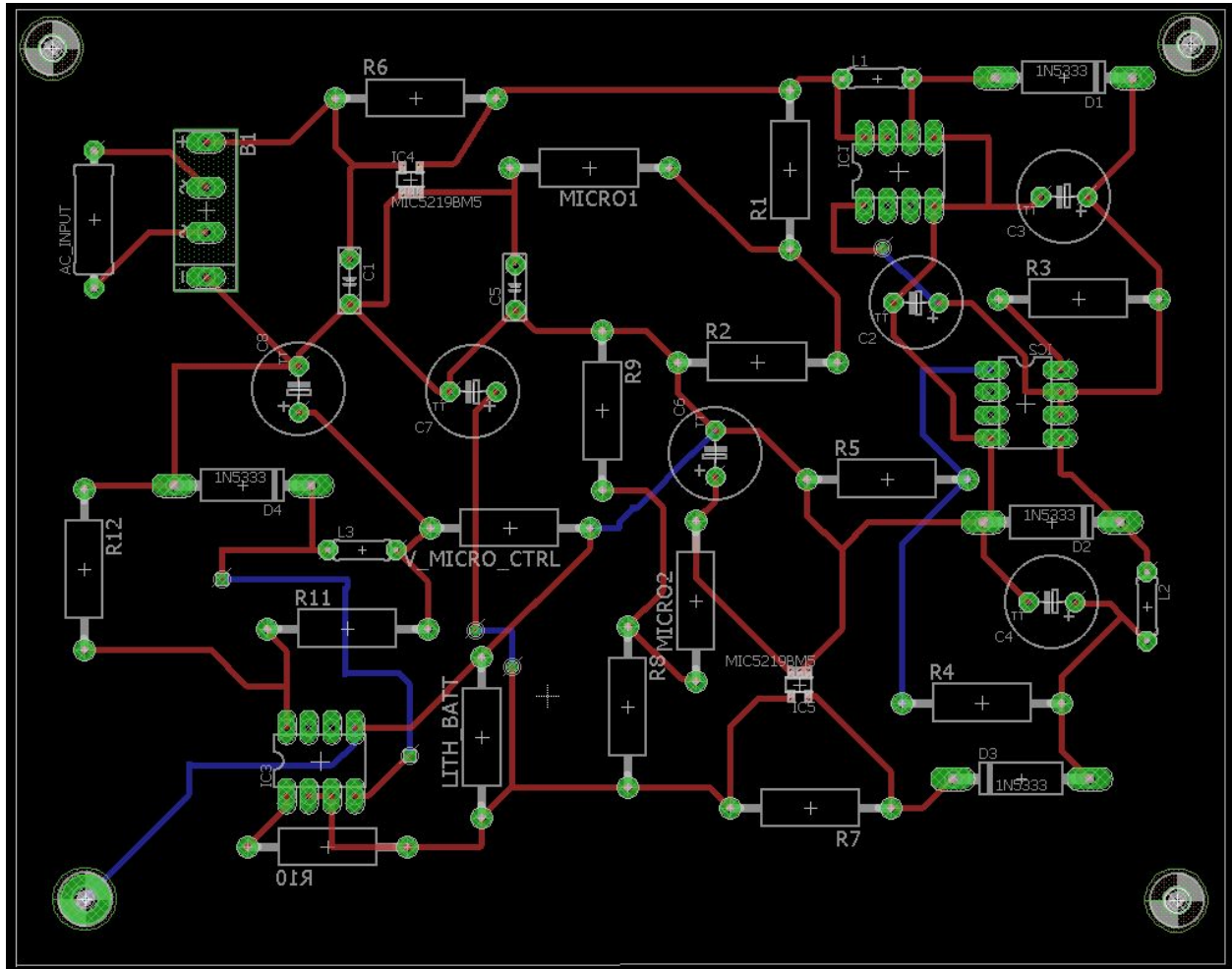
This information is generally descriptive and is not intended to make or imply any representation, guarantee or warranty with respect to any cells and batteries. Cell and battery design/specifications are subject to modification without notice. Contact Panasonic for the latest information.

Appendix 3 – Detailed technical attachments

Schematic of the powertrain circuit:



Powertrain PCB Artwork:



Source code of all programs:

MSP430 program for collecting test data:

```
#include <msp430fr6989.h>
#include <stdint.h>
#include <stdbool.h>

void Init_RTC();
void Init_ADC();

bool running = false;

unsigned int dataTransmit[4] = {0};

int main(void) {
    // Stop watchdog timer
    WDTCTL = WDTPW | WDTHOLD;

    // Set all pins to output low
    PAOUT = PBOUT = PCOUT = PDOUT = PEOUT = 0x0;
    PADIR = PBDIR = PCDIR = PDDIR = PEDIR = 0xFFFF;

    // Unlock high-impedance mode when resetting from LPMx.5
    PM5CTL0 &= ~LOCKLPM5;

    /*
    * Clock System configuration
    */

    // Unlock CS
    CSCTL0 = CSKEY;

    // Set DCO to 8 MHz
    CSCTL1 = DCOFSEL_6;

    // Set ACLK to LFXT
    // Set SMCLK to DCO
    // Set MCLK to DCO
    CSCTL2 = SELA__LFXTCLK | SELS__DCOCLK | SELM__DCOCLK;

    // Set ACLK, SMCLK, MCLK dividers to 1, 1, 2
    CSCTL3 = DIVA__1 | DIVS__1 | DIVM__2;

    // Lock CS
    CSCTL0_H = 0;

    /*
    * UART configuration - UCA1 connected to Launchpad J101 jumpers
    */

    // USCI_A1 UART operation
    P3SEL0 |= BIT4 | BIT5;
```

```

P3SEL1 &= ~(BIT4 | BIT5);

// Configure USCI_A1 for UART mode
UCA1CTLW0 = UCSWRST;           // Put eUSCI in reset
UCA1CTLW0 |= UCSSEL__ACLK;      // CLK = ACLK
UCA1BR0 = 3;                   // 9600 baud
UCA1MCTLW |= 0x5300;           // 32768/9600 - INT(32768/9600)=0.41
                                   // UCBRSx

value = 0x53 (See UG)
UCA1BR1 = 0;
UCA1CTL1 &= ~UCSWRST;          // Initialize eUSCI

/*
 * DMA configuration - transfer the array of data through UART
 */

// Set DMA0 to trigger on UART transmit buffer empty
DMACTL0 = DMA0TSEL__UCA1TXIFG;

// Set source and destination addresses for DMA
data16 write_addr((unsigned short) &DMA0SA,(unsigned long) &dataTransmit[0]);
data16 write_addr((unsigned short) &DMA0DA,(unsigned long) &UCA1TXBUF);

// Set number of bytes of the source
DMA0SZ = 8;

// Set DMA to transmit byte at a time
DMA0CTL = DMASRCINCR_3 | DMASRCBYTE | DMADSTBYTE;

/*
 * GPIO configuration - set up buttons to trigger interrupt
 */

// Set P1.1/2 (S1/S2 on Launchpad) to trigger interrupt on button press (high to
low transition)
P1DIR &= ~(BIT1 | BIT2);
P1IES |= BIT1 | BIT2;
P1IE |= BIT1 | BIT2;

// Set P1.1/2 pull-up resistor
P1REN |= BIT1 | BIT2;
P1OUT |= BIT1 | BIT2;

// Clear P1.1/2 interrupts
P1IFG &= ~(BIT1 | BIT2);

// Set VREF to 2.5V
REFCTL0 = REFVSEL_2;

Init_RTC();
Init_ADC();

__enable_interrupt();

```

```

while (1) {
    if (running) {
        // Start an ADC conversion
        ADC12CTL0 |= ADC12ENC | ADC12SC;

        // Wait for ADC conversion result
        __bis_SR_register(LPM0_bits | GIE);
        __no_operation();

        dataTransmit[0] = ADC12MEM0;
        dataTransmit[1] = ADC12MEM1;
        dataTransmit[2] = ADC12MEM2;
        dataTransmit[3] = ADC12MEM3;

        // Start transfer to UART
        DMA0CTL |= DMAEN;
    }

    // Wait for RTC interrupt (1/32 s)
    __bis_SR_register(LPM3_bits | GIE);
    __no_operation();
}

/*
 * Set up RTC to interrupt 128 times every second
 */
void Init_RTC() {
    // Unlock CS
    CSCTL0 = CSKEY;

    // Enable LFXT pins
    PJSEL0 |= BIT4 | BIT5;

    // Enable LFXT for RTC
    CSCTL4 &= ~LFXTOFF;

    do
    {
        // Clear LFXT fault flag
        CSCTL5 &= ~LFXTOFFG;
        SFRIFG1 &= ~OFIFG;
    } while (SFRIFG1 & OFIFG); // Test oscillator fault flag

    // Lock CS
    CSCTL0_H = 0;

    // Unlock RTC
    RTCCTL0_H = RTCKEY_H;

    // Enable RTC time event interrupt
    RTCCTL0_L = RTCTEVIE;

    // Stop the RTC, set to counter mode

```

```

    // Select clock input to LFXT (32kHz)
    // Set the time event to 8-bit overflow of the counter
    RTCCTL1 = RTCHOLD | RTCSEL_0 | RTCIEV_0;

    // Start the RTC
    RTCCTL1 &= ~(RTCHOLD);
}

/*
 * Set up ADC
 */
void Init_ADC() {
    // Enable ADC
    // Set to automatically sample all channels
    // Sample for 512 cycles
    ADC12CTL0 = ADC12ON | ADC12MSC | ADC12SHT0_6;

    // Sample multiple channels in sequence
    // Set ADC to pulse sample mode (sample only when setting the ADC12SC bit)
    ADC12CTL1 = ADC12CONSEQ_1 | ADC12SHP;

    // Enable A4-A7 inputs on P8.7-P8.4
    P8SEL1 |= BIT7 | BIT6 | BIT5 | BIT4;
    P8SEL0 |= BIT7 | BIT6 | BIT5 | BIT4;

    // Setup ADC12MEM0 with A4 input
    // Set VR+ = VREF (2.5V)
    ADC12MCTL0 = ADC12INCH_4 | ADC12VRSEL_1;

    // Setup ADC12MEM1 with A5 input
    // Set VR+ = VREF (2.5V)
    ADC12MCTL1 = ADC12INCH_5 | ADC12VRSEL_1;

    // Setup ADC12MEM2 with A6 input
    // Set VR+ = VREF (2.5V)
    ADC12MCTL2 = ADC12INCH_6 | ADC12VRSEL_1;

    // Setup ADC12MEM3 with A7 input
    // Set VR+ = VREF (2.5V)
    // Mark as end of sequence
    ADC12MCTL3 = ADC12INCH_7 | ADC12VRSEL_1 | ADC12EOS;

    // Enable ADC conversion complete interrupt for MEM3
    ADC12IER0 |= ADC12IE3;
}

/*
 * RTC Interrupt Service Routine
 */
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=RTC_VECTOR
__interrupt
#elif defined(__GNUC__)
__attribute__((interrupt(RTC_VECTOR)))

```

```

#endif
void RTC_ISR(void)
{
    switch(__even_in_range(RTCIV, 16))
    {
        case RTCIV_NONE: break;           //No interrupts
        case RTCIV_RTCOIFG: break;        //RTCIFG
        case RTCIV_RTCRDYIFG: break;      //RTCRDYIFG
        case RTCIV_RTCEVIFG: break;       //RTCEVIFG
        // Interrupts every 1/32 second
        // Exit LPM
        __bic_SR_register_on_exit(LPM3_bits);
        __no_operation();
        break;
        case RTCIV_RTCAIFG: break;        //RTCAIFG
        case RTCIV_RT0PSIFG: break;       //RT0PSIFG
        case RTCIV_RT1PSIFG: break;       //RT1PSIFG

        default: break;
    }
}

/*
 * ADC12 Interrupt Service Routine
 */
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector = ADC12_VECTOR
__interrupt void ADC12_ISR(void)
#elif defined(__GNUC__)
void __attribute__((interrupt(ADC12_VECTOR))) ADC12_ISR (void)
#else
#error Compiler not supported!
#endif
{
    switch(__even_in_range(ADC12IV, ADC12IV_ADC12RDYIFG))
    {
        case ADC12IV_NONE: break;        // Vector 0: No interrupt
        case ADC12IV_ADC12OVIFG: break;   // Vector 2: ADC12MEMx Overflow
        case ADC12IV_ADC12TOVIFG: break;   // Vector 4: Conversion time overflow
        case ADC12IV_ADC12HIFG: break;     // Vector 6: ADC12BHI
        case ADC12IV_ADC12LOIFG: break;    // Vector 8: ADC12BLO
        case ADC12IV_ADC12INIFG: break;    // Vector 10: ADC12BIN
        case ADC12IV_ADC12IFG0: break;     // Vector 12: ADC12MEM0
        case ADC12IV_ADC12IFG1: break;     // Vector 14: ADC12MEM1
        case ADC12IV_ADC12IFG2: break;     // Vector 16: ADC12MEM2
        case ADC12IV_ADC12IFG3: break;     // Vector 18: ADC12MEM3
        // Clear interrupt flag
        ADC12IV &= ~ADC12IV_ADC12IFG3;
        // Exit LPM
        __bic_SR_register_on_exit(LPM0_bits);
        break;
        case ADC12IV_ADC12IFG4: break;     // Vector 20: ADC12MEM4
    }
}

```

```

    case ADC12IV_ADC12IFG5:    break;           // Vector 22:  ADC12MEM5
    case ADC12IV_ADC12IFG6:    break;           // Vector 24:  ADC12MEM6
    case ADC12IV_ADC12IFG7:    break;           // Vector 26:  ADC12MEM7
    case ADC12IV_ADC12IFG8:    break;           // Vector 28:  ADC12MEM8
    case ADC12IV_ADC12IFG9:    break;           // Vector 30:  ADC12MEM9
    case ADC12IV_ADC12IFG10:   break;           // Vector 32:  ADC12MEM10
    case ADC12IV_ADC12IFG11:   break;           // Vector 34:  ADC12MEM11
    case ADC12IV_ADC12IFG12:   break;           // Vector 36:  ADC12MEM12
    case ADC12IV_ADC12IFG13:   break;           // Vector 38:  ADC12MEM13
    case ADC12IV_ADC12IFG14:   break;           // Vector 40:  ADC12MEM14
    case ADC12IV_ADC12IFG15:   break;           // Vector 42:  ADC12MEM15
    case ADC12IV_ADC12IFG16:   break;           // Vector 44:  ADC12MEM16
    case ADC12IV_ADC12IFG17:   break;           // Vector 46:  ADC12MEM17
    case ADC12IV_ADC12IFG18:   break;           // Vector 48:  ADC12MEM18
    case ADC12IV_ADC12IFG19:   break;           // Vector 50:  ADC12MEM19
    case ADC12IV_ADC12IFG20:   break;           // Vector 52:  ADC12MEM20
    case ADC12IV_ADC12IFG21:   break;           // Vector 54:  ADC12MEM21
    case ADC12IV_ADC12IFG22:   break;           // Vector 56:  ADC12MEM22
    case ADC12IV_ADC12IFG23:   break;           // Vector 58:  ADC12MEM23
    case ADC12IV_ADC12IFG24:   break;           // Vector 60:  ADC12MEM24
    case ADC12IV_ADC12IFG25:   break;           // Vector 62:  ADC12MEM25
    case ADC12IV_ADC12IFG26:   break;           // Vector 64:  ADC12MEM26
    case ADC12IV_ADC12IFG27:   break;           // Vector 66:  ADC12MEM27
    case ADC12IV_ADC12IFG28:   break;           // Vector 68:  ADC12MEM28
    case ADC12IV_ADC12IFG29:   break;           // Vector 70:  ADC12MEM29
    case ADC12IV_ADC12IFG30:   break;           // Vector 72:  ADC12MEM30
    case ADC12IV_ADC12IFG31:   break;           // Vector 74:  ADC12MEM31
    case ADC12IV_ADC12RDYIFG:  break;           // Vector 76:  ADC12RDY
    default: break;
}
}

/*
 * PORT1 Interrupt Service Routine
 * Handles S1 and S2 button press interrupts
 */
#pragma vector = PORT1_VECTOR
__interrupt void PORT1_ISR(void)
{
    switch(__even_in_range(P1IV, P1IV_P1IFG7))
    {
        case P1IV_NONE : break;
        case P1IV_P1IFG0 : break;
        case P1IV_P1IFG1 : // Button S1 pressed
            running = !running;
            break;
        case P1IV_P1IFG2 : // Button S2 pressed
            running = !running;
            break;
        case P1IV_P1IFG3 : break;
        case P1IV_P1IFG4 : break;
        case P1IV_P1IFG5 : break;
        case P1IV_P1IFG6 : break;
        case P1IV_P1IFG7 : break;
    }
}

```

} }

Windows program for collecting test data:

```
#include <iostream>
#include <string>
#include <Windows.h>
#include <fstream>
#include <stdint.h>

using namespace std;

#define bytes 4 * 2 // 4 16-bit integers

void writeOutput(ofstream& file, uint8_t data[8]);

int main(int argc, char* argv[]) {
    if (argc < 3) {
        cout << "Usage: " << argv[0] << " <COMX> <dest.csv>" << endl;
        return 1;
    }

    // Open serial port specified
    HANDLE comPort = CreateFile(("\\\\.\\\\" + string(argv[1])).c_str(),
        GENERIC_READ,
        0,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL,
        NULL);

    if (comPort == NULL) {
        cout << "Error opening COM port " << argv[1] << endl;
        return 1;
    }

    // Configure serial port settings to match the settings on the microcontroller
    DCB Dcb;
    Dcb.DCBlength = sizeof(Dcb);
    Dcb.BaudRate = CBR_9600;
    Dcb.StopBits = ONESTOPBIT;
    Dcb.ByteSize = 8;
    Dcb.Parity = NOPARITY;
    Dcb.fParity = 0;
    Dcb.fOutxCtsFlow = 0;
    Dcb.fOutxDsrFlow = 0;
    Dcb.fDsrSensitivity = 0;
    Dcb.fTXContinueOnXoff = TRUE;
    Dcb.fOutX = 0;
    Dcb.fInX = 0;
    Dcb.fNull = 0;
    Dcb.fErrorChar = 0;
    Dcb.fAbortOnError = 0;
    Dcb.fRtsControl = RTS_CONTROL_DISABLE;
    Dcb.fDtrControl = DTR_CONTROL_DISABLE;
```

```

    if (!SetCommState(comPort, &Dcb)) {
        cout << "Error: Failed to configure COM port" << endl;
        return 2;
    }

    // Open output file
    ofstream outFile(argv[2]);

    if (!outFile) {
        cout << "Error: opening output file" << endl;
        return 3;
    }

    // Write CSV header
    outFile << "VSenseRect,VRect,VSenseBat,VBat" << endl;

    uint8_t data[bytes] = { 0 };
    int dataIndex = 0;

    while (true) {
        // Repeatedly read from serial port until we get the right number of
bytes
        DWORD read = 0;
        uint8_t buffer[bytes];

        ReadFile(comPort, &buffer, bytes, &read, NULL);

        if (read > 0) {
            for (int i = 0; i < read; i++) {
                if (dataIndex == bytes) {
                    writeOutput(outFile, data);
                    dataIndex = 0;
                }

                data[dataIndex++] = buffer[i];
            }

            if (dataIndex == bytes) {
                writeOutput(outFile, data);
                dataIndex = 0;
            }
        }
    }

    void writeOutput(ofstream& file, uint8_t data[8]) {
        for (int i = 0; i < 4; i++) {
            uint16_t number = (uint16_t)(data[i * 2] | (data[i * 2 + 1] << 8));
            cout << number << " ";
            file << number;

            if (i < 3) {

```

```
        file << ",";
    }
}

cout << endl;
file << endl;
file.flush();
}
```

MSP430 program for displaying power output on LCD:

```
#include <msp430fr6989.h>
#include <driverlib.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <cmath>
#include "../common/hal_LCD.h"

extern char lcdPos[6];

#define filter 0.5
#define rate 32 // times per second readings will occur
#define readings 4 // channels that will be read by ADC

#define POWER_MODE 0
#define EFFICIENCY_MODE 1

unsigned char mode = POWER_MODE;

void Init_RTC();
void Init_ADC();

int main(void) {
    // Stop watchdog timer
    WDTCTL = WDTPW | WDTHOLD;

    // Set all pins to output low
    PAOUT = PBOUT = PCOUT = PDOUT = PEOUT = 0x0;
    PADIR = PBDIR = PCDIR = PDDIR = PEDIR = 0xFFFF;

    // Unlock high-impedance mode when resetting from LPMx.5
    PM5CTL0 &= ~LOCKLPM5;

    /*
    * Clock System configuration
    */

    // Unlock CS
    CSCTL0 = CSKEY;

    // Set DCO to 1 MHz
    CSCTL1 = DCOFSEL_0;

    // Set ACLK to VLO
    // Set SMCLK to DCO
    // Set MCLK to DCO
    CSCTL2 = SELA__VLOCLK | SELS__DCOCLK | SELM__DCOCLK;

    // Set ACLK, SMCLK, MCLK dividers to 1, 1, 2
    CSCTL3 = DIVA__1 | DIVS__1 | DIVM__1;
```

```

// Lock CS
CSCTL0_H = 0;

// Set VREF to 2.5V
REFCTL0 = REFVSEL_2;

/*
 * GPIO configuration - set buttons to trigger interrupt
 */

// Set P1.1/2 (S1/S2) to trigger interrupt on button press (high to low
transition)
P1DIR &= ~(BIT1 | BIT2);
P1IES |= BIT1 | BIT2;
P1IE |= BIT1 | BIT2;

// Set P1.1/2 pull-up resistor
P1REN |= BIT1 | BIT2;
P1OUT |= BIT1 | BIT2;

// Clear P1.1/2 interrupts
P1IFG &= ~(BIT1 | BIT2);

Init_RTC();
Init_ADC();
Init_LCD();

__enable_interrupt();

unsigned int iteration = 1;

uint16_t data[readings][rate];

while (1) {
    // Start an ADC conversion
    ADC12CTL0 |= ADC12ENC | ADC12SC;

    // Wait for ADC conversion result
    __bis_SR_register(LPM0_bits | GIE);
    __no_operation();

    unsigned int i = iteration - 1;
    data[0][i] = ADC12MEM0;
    data[1][i] = ADC12MEM1;
    data[2][i] = ADC12MEM2;
    data[3][i] = ADC12MEM3;

    // Display on LCD every second
    if (iteration == rate) {
        iteration = 0;
        int8_t pos = 4;

        // Calculate RMS values for readings
        uint32_t rmsValue[4] = {0};

```

```

unsigned int r;
for (r = 0; r < 4; r++) {
    for (i = 0; i < rate; i++) {
        rmsValue[r] += (uint32_t)data[r][i] * data[r][i];
    }

    rmsValue[r] = sqrt(rmsValue[r] / (double)rate);
}

double currentRect = (rmsValue[0] / 4095.0 * 2.5) * 1.125;
double voltageRect = (rmsValue[1] / 4095.0 * 2.5) * 6.57051802;
double currentBat = (rmsValue[2] / 4095.0 * 2.5) * 0.513;
double voltageBat = (rmsValue[3] / 4095.0 * 2.5) * 2.30162703;

double powerRect = -1;

if (voltageRect >= 2) {
    powerRect = currentRect * voltageRect;
}

double powerBat = currentBat * voltageBat;

unsigned long displayNum = 0;
char suffix;

if (mode == POWER_MODE) {
    displayNum = powerBat * 1000;
    suffix = 'W';
} else {
    if (powerRect >= 0) {
        displayNum = powerBat / powerRect * 100 * 1000;
        suffix = '%';
    } else {
        suffix = 'U';
    }
}

unsigned char decimalPointPos = 1;
while (displayNum > (uint32_t)(pow(10, pos + 1)) - 1) {
    displayNum /= 10;
    decimalPointPos++;
}

clearLCD();

showChar(suffix, lcdPos[5]);

while (pos >= 0) {
    showChar(displayNum % 10 + '0', lcdPos[pos--]);
    displayNum /= 10;
}

if (decimalPointPos < 5) {

```

```

        LCDMEM[lcdPos[decimalPointPos] + 1] |= BIT0;
    }
}

iteration++;

// Wait for RTC interrupt
__bis_SR_register(LPM3_bits | GIE);
__no_operation();
}
}

/*
 * Set up RTC to interrupt 128 times every second
 */
void Init_RTC() {
    // Unlock CS
    CSCTL0 = CSKEY;

    // Enable LFXT pins
    PJSEL0 |= BIT4 | BIT5;

    // Enable LFXT for RTC
    CSCTL4 &= ~LFXTOFF;

    do
    {
        // Clear LFXT fault flag
        CSCTL5 &= ~LFXTOFFG;
        SFRIFG1 &= ~OFIFG;
    } while (SFRIFG1 & OFIFG); // Test oscillator fault flag

    // Lock CS
    CSCTL0_H = 0;

    // Unlock RTC
    RTCCTL0_H = RTCKEY_H;

    // Enable RTC time event interrupt
    RTCCTL0_L = RTCTEVIE;

    // Stop the RTC, set to counter mode
    // Select counter clock input to LFXT
    // Set the time event to 8-bit overflow of the counter
    RTCCTL1 = RTCHOLD | RTCSEL_2 | RTCTEV_0;

    // Set prescale 1 clock source to LFXT
    // Set clock divider to 2 (32kHz/4 = 8kHz)
    RTCPS1CTL = RT1SSEL_0 | RT1PSDIV_1;

    // Start the RTC
    RTCCTL1 &= ~(RTCHOLD);
}

```

```

/*
 * Set up ADC
 */
void Init_ADC() {
    // Enable ADC
    // Set to automatically sample all channels
    // Sample for 512 cycles
    ADC12CTL0 = ADC12ON | ADC12MSC | ADC12SHT0_6;

    // Sample multiple channels in sequence
    // Set ADC to pulse sample mode (sample only when setting the ADC12SC bit)
    ADC12CTL1 = ADC12CONSEQ_1 | ADC12SHP;

    // Enable A4-A7 inputs on P8.7-P8.4
    P8SEL1 |= BIT7 | BIT6 | BIT5 | BIT4;
    P8SEL0 |= BIT7 | BIT6 | BIT5 | BIT4;

    // Setup ADC12MEM0 with A4 input
    // Set VR+ = VREF (2.5V)
    ADC12MCTL0 = ADC12INCH_4 | ADC12VRSEL_1;

    // Setup ADC12MEM1 with A5 input
    // Set VR+ = VREF (2.5V)
    ADC12MCTL1 = ADC12INCH_5 | ADC12VRSEL_1;

    // Setup ADC12MEM2 with A6 input
    // Set VR+ = VREF (2.5V)
    ADC12MCTL2 = ADC12INCH_6 | ADC12VRSEL_1;

    // Setup ADC12MEM3 with A7 input
    // Set VR+ = VREF (2.5V)
    // Mark as end of sequence
    ADC12MCTL3 = ADC12INCH_7 | ADC12VRSEL_1 | ADC12EOS;

    // Enable ADC conversion complete interrupt for MEM3
    ADC12IER0 |= ADC12IE3;
}

/*
 * RTC Interrupt Service Routine
 */
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=RTC_VECTOR
__interrupt
#elif defined(__GNUC__)
__attribute__((interrupt(RTC_VECTOR)))
#endif
void RTC_ISR(void)
{
    switch(__even_in_range(RTCIV, 16))
    {
        case RTCIV_NONE: break;           //No interrupts
        case RTCIV_RTCOIFG: break;        //RTCIFG
        case RTCIV_RTCRDYIFG: break;      //RTCRDYIFG
    }
}

```

```

    case RTCIV_RTCTEVIFG:           //RTCEVIFG
    // Interrupts every 1/32 second
    // Exit LPM
    __bic_SR_register_on_exit(LPM3_bits);
    __no_operation();
    break;
    case RTCIV_RTCAIFG: break;       //RTCAIFG
    case RTCIV_RT0PSIFG: break;      //RT0PSIFG
    case RTCIV_RT1PSIFG: break;      //RT1PSIFG

    default: break;
}
}

#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector = ADC12_VECTOR
__interrupt void ADC12_ISR(void)
#elif defined(__GNUC__)
void __attribute__((interrupt(ADC12_VECTOR))) ADC12_ISR (void)
#else
#error Compiler not supported!
#endif
{
    switch(__even_in_range(ADC12IV, ADC12IV_ADC12RDYIFG))
    {
        case ADC12IV_NONE:          break;           // Vector 0:  No interrupt
        case ADC12IV_ADC12OVIFG:    break;           // Vector 2:  ADC12MEMx Overflow
        case ADC12IV_ADC12TOVIFG:    break;           // Vector 4:  Conversion time overflow
        case ADC12IV_ADC12HIIFG:     break;           // Vector 6:  ADC12BHI
        case ADC12IV_ADC12LOIFG:     break;           // Vector 8:  ADC12BLO
        case ADC12IV_ADC12INIFG:     break;           // Vector 10: ADC12BIN
        case ADC12IV_ADC12IFG0:      break;           // Vector 12: ADC12MEM0
        break;
        case ADC12IV_ADC12IFG1:      break;           // Vector 14: ADC12MEM1
        break;
        case ADC12IV_ADC12IFG2:      break;           // Vector 16: ADC12MEM2
        case ADC12IV_ADC12IFG3:      break;           // Vector 18: ADC12MEM3
        // Clear interrupt flag
        ADC12IV &= ~ADC12IV_ADC12IFG3;
        // Exit LPM
        __bic_SR_register_on_exit(LPM0_bits);
        break;
        case ADC12IV_ADC12IFG4:      break;           // Vector 20: ADC12MEM4
        case ADC12IV_ADC12IFG5:      break;           // Vector 22: ADC12MEM5
        case ADC12IV_ADC12IFG6:      break;           // Vector 24: ADC12MEM6
        case ADC12IV_ADC12IFG7:      break;           // Vector 26: ADC12MEM7
        case ADC12IV_ADC12IFG8:      break;           // Vector 28: ADC12MEM8
        case ADC12IV_ADC12IFG9:      break;           // Vector 30: ADC12MEM9
        case ADC12IV_ADC12IFG10:     break;           // Vector 32: ADC12MEM10
        case ADC12IV_ADC12IFG11:     break;           // Vector 34: ADC12MEM11
        case ADC12IV_ADC12IFG12:     break;           // Vector 36: ADC12MEM12
        case ADC12IV_ADC12IFG13:     break;           // Vector 38: ADC12MEM13
        case ADC12IV_ADC12IFG14:     break;           // Vector 40: ADC12MEM14
        case ADC12IV_ADC12IFG15:     break;           // Vector 42: ADC12MEM15
    }
}

```

```

        case ADC12IV_ADC12IFG16: break;           // Vector 44: ADC12MEM16
        case ADC12IV_ADC12IFG17: break;           // Vector 46: ADC12MEM17
        case ADC12IV_ADC12IFG18: break;           // Vector 48: ADC12MEM18
        case ADC12IV_ADC12IFG19: break;           // Vector 50: ADC12MEM19
        case ADC12IV_ADC12IFG20: break;           // Vector 52: ADC12MEM20
        case ADC12IV_ADC12IFG21: break;           // Vector 54: ADC12MEM21
        case ADC12IV_ADC12IFG22: break;           // Vector 56: ADC12MEM22
        case ADC12IV_ADC12IFG23: break;           // Vector 58: ADC12MEM23
        case ADC12IV_ADC12IFG24: break;           // Vector 60: ADC12MEM24
        case ADC12IV_ADC12IFG25: break;           // Vector 62: ADC12MEM25
        case ADC12IV_ADC12IFG26: break;           // Vector 64: ADC12MEM26
        case ADC12IV_ADC12IFG27: break;           // Vector 66: ADC12MEM27
        case ADC12IV_ADC12IFG28: break;           // Vector 68: ADC12MEM28
        case ADC12IV_ADC12IFG29: break;           // Vector 70: ADC12MEM29
        case ADC12IV_ADC12IFG30: break;           // Vector 72: ADC12MEM30
        case ADC12IV_ADC12IFG31: break;           // Vector 74: ADC12MEM31
        case ADC12IV_ADC12RDYIFG: break;          // Vector 76: ADC12RDY
        default: break;
    }
}

/*
 * PORT1 Interrupt Service Routine
 * Handles S1 and S2 button press interrupts
 */
#pragma vector = PORT1_VECTOR
__interrupt void PORT1_ISR(void)
{
    switch(__even_in_range(P1IV, P1IV_P1IFG7))
    {
        case P1IV_NONE : break;
        case P1IV_P1IFG0 : break;
        case P1IV_P1IFG1 : // Button S1 pressed
        case P1IV_P1IFG2 : // Button S2 pressed
            if (mode == POWER_MODE) {
                mode = EFFICIENCY_MODE;
            } else {
                mode = POWER_MODE;
            }
            break;
        case P1IV_P1IFG3 : break;
        case P1IV_P1IFG4 : break;
        case P1IV_P1IFG5 : break;
        case P1IV_P1IFG6 : break;
        case P1IV_P1IFG7 : break;
    }
}

```

MSP430 LCD library:

```
/* --COPYRIGHT--,BSD
 * Copyright (c) 2015, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * * Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 *
 * * Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 *
 * * Neither the name of Texas Instruments Incorporated nor the names of
 *   its contributors may be used to endorse or promote products derived
 *   from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/
/*****
 *
 * hal_LCD.c
 *
 * Hardware abstraction layer for the FH-1138P Segmented LCD
 * on MSP-EXP430FR6989
 *
 * February 2015
 * E. Chen
 *
 *****/

#include <driverlib.h>
#include "hal_LCD.h"
#include "string.h"

const char lcdPos[6] = {
    pos1, pos2, pos3, pos4, pos5, pos6
};
```

```
// LCD memory map for numeric digits
const char digit[10][2] =
{
    {0xFC, 0x28}, /* "0" LCD segments a+b+c+d+e+f+k+q */
    {0x60, 0x20}, /* "1" */
    {0xDB, 0x00}, /* "2" */
    {0xF3, 0x00}, /* "3" */
    {0x67, 0x00}, /* "4" */
    {0xB7, 0x00}, /* "5" */
    {0xBF, 0x00}, /* "6" */
    {0xE4, 0x00}, /* "7" */
    {0xFF, 0x00}, /* "8" */
    {0xF7, 0x00}  /* "9" */
};
```

```
// LCD memory map for uppercase letters
const char alphabetBig[26][2] =
{
    {0xEF, 0x00}, /* "A" LCD segments a+b+c+e+f+g+m */
    {0xF1, 0x50}, /* "B" */
    {0x9C, 0x00}, /* "C" */
    {0xF0, 0x50}, /* "D" */
    {0x9F, 0x00}, /* "E" */
    {0x8F, 0x00}, /* "F" */
    {0xBD, 0x00}, /* "G" */
    {0x6F, 0x00}, /* "H" */
    {0x90, 0x50}, /* "I" */
    {0x78, 0x00}, /* "J" */
    {0x0E, 0x22}, /* "K" */
    {0x1C, 0x00}, /* "L" */
    {0x6C, 0xA0}, /* "M" */
    {0x6C, 0x82}, /* "N" */
    {0xFC, 0x00}, /* "O" */
    {0xCF, 0x00}, /* "P" */
    {0xFC, 0x02}, /* "Q" */
    {0xCF, 0x02}, /* "R" */
    {0xB7, 0x00}, /* "S" */
    {0x80, 0x50}, /* "T" */
    {0x7C, 0x00}, /* "U" */
    {0x0C, 0x28}, /* "V" */
    {0x6C, 0x0A}, /* "W" */
    {0x00, 0xAA}, /* "X" */
    {0x00, 0xB0}, /* "Y" */
    {0x90, 0x28}  /* "Z" */
};
```

```
// LCD memory map for uppercase letters
const char alphabetSmall[26][2] =
{
    {0xEF, 0x00}, /* "A" LCD segments a+b+c+e+f+g+m */
    {0xF1, 0x50}, /* "B" */
    {0x9C, 0x00}, /* "C" */
    {0xF0, 0x50}, /* "D" */

```

```

        {0x9F, 0x00}, /* "E" */
        {0x8F, 0x00}, /* "F" */
        {0xBD, 0x00}, /* "G" */
        {0x6F, 0x00}, /* "H" */
        {0x90, 0x50}, /* "I" */
        {0x78, 0x00}, /* "J" */
        {0x0E, 0x22}, /* "K" */
        {0x1C, 0x00}, /* "L" */
        {0x2B, 0x10}, /* "m" */
        {0x6C, 0x82}, /* "N" */
        {0xFC, 0x00}, /* "O" */
        {0xCF, 0x00}, /* "P" */
        {0xFC, 0x02}, /* "Q" */
        {0xCF, 0x02}, /* "R" */
        {0xB7, 0x00}, /* "S" */
        {0x80, 0x50}, /* "T" */
        {0x38, 0x00}, /* "u" */
        {0x0C, 0x28}, /* "V" */
        {0x6C, 0x0A}, /* "W" */
        {0x00, 0xAA}, /* "X" */
        {0x00, 0xB0}, /* "Y" */
        {0x90, 0x28} /* "Z" */
};

void Init_LCD()
{
    LCD_C_initParam initParams = {0};
    initParams.clockSource = LCD_C_CLOCKSOURCE_VLOCLK;
    initParams.clockDivider = LCD_C_CLOCKDIVIDER_1;
    initParams.clockPrescaler = LCD_C_CLOCKPRESCALAR_8;
    initParams.muxRate = LCD_C_4_MUX;
    initParams.waveforms = LCD_C_LOW_POWER_WAVEFORMS;
    initParams.segments = LCD_C_SEGMENTS_ENABLED;

    LCD_C_init(LCD_C_BASE, &initParams);
    // LCD Operation - VLCD generated internally, V2-V4 generated internally, v5 to
ground
    LCD_C_setPinAsLCDFunction(LCD_C_BASE, LCD_C_SEGMENT_LINE_4);
    LCD_C_setPinAsLCDFunctionEx(LCD_C_BASE, LCD_C_SEGMENT_LINE_6,
                                LCD_C_SEGMENT_LINE_21);
    LCD_C_setPinAsLCDFunctionEx(LCD_C_BASE, LCD_C_SEGMENT_LINE_27,
                                LCD_C_SEGMENT_LINE_31);
    LCD_C_setPinAsLCDFunctionEx(LCD_C_BASE, LCD_C_SEGMENT_LINE_35,
                                LCD_C_SEGMENT_LINE_39);

    LCD_C_setVLCDSource(LCD_C_BASE, LCD_C_VLCD_GENERATED INTERNALLY,
                        LCD_C_V2V3V4_GENERATED INTERNALLY_NOT_SWITCHED_TO_PINS,
                        LCD_C_V5_VSS);

    // Set VLCD voltage to 3.20v
    LCD_C_setVLCDVoltage(LCD_C_BASE,
                        LCD_C_CHARGE PUMP_VOLTAGE_3_02V_OR_2_52VREF);

```

```

    // Enable charge pump and select internal reference for it
    LCD_C_enableChargePump(LCD_C_BASE);
    LCD_C_selectChargePumpReference(LCD_C_BASE,
                                    LCD_C_INTERNAL_REFERENCE_VOLTAGE);

    LCD_C_configChargePump(LCD_C_BASE, LCD_C_SYNCHRONIZATION_ENABLED, 0);

    // Clear LCD memory
    LCD_C_clearMemory(LCD_C_BASE);

    //Turn LCD on
    LCD_C_on(LCD_C_BASE);
}

/*
 * Scrolls input string across LCD screen from left to right
 */
void displayScrollText(char *msg)
{
    int length = strlen(msg);
    int i;
    int s = 5;
    char buffer[6] = " ";
    for (i=0; i<length+7; i++)
    {
        int t;
        for (t=0; t<6; t++)
            buffer[t] = ' ';
        int j;
        for (j=0; j<length; j++)
        {
            if (((s+j) >= 0) && ((s+j) < 6))
                buffer[s+j] = msg[j];
        }
        s--;

        showChar(buffer[0], pos1);
        showChar(buffer[1], pos2);
        showChar(buffer[2], pos3);
        showChar(buffer[3], pos4);
        showChar(buffer[4], pos5);
        showChar(buffer[5], pos6);

        __delay_cycles(200000);
    }
}

/*
 * Displays input character at given LCD digit/position
 * Only spaces, numeric digits, and uppercase letters are accepted characters
 */
void showChar(char c, int position)
{
    if (c == ' ')

```

```

{
    // Display space
    LCDMEM[position] = 0;
    LCDMEM[position+1] = 0;
}
else if (c >= '0' && c <= '9')
{
    // Display digit
    LCDMEM[position] = digit[c-48][0];
    LCDMEM[position+1] = digit[c-48][1];
}
else if (c >= 'A' && c <= 'Z')
{
    // Display alphabet
    LCDMEM[position] = alphabetBig[c-65][0];
    LCDMEM[position+1] = alphabetBig[c-65][1];
}
else if (c >= 'a' && c <= 'z')
{
    // Display alphabet
    LCDMEM[position] = alphabetSmall[c-97][0];
    LCDMEM[position+1] = alphabetSmall[c-97][1];
}
else if (c == '=') {
    LCDMEM[position] = 0x12;
    LCDMEM[position+1] = 0;
}
else if (c == '%') {
    LCDMEM[position] = 0;
    LCDMEM[position+1] = 0xaa;
}
else
{
    // Turn all segments on if character is not a space, digit, or uppercase letter
    LCDMEM[position] = 0xFF;
    LCDMEM[position+1] = 0xFF;
}
}

/*
 * Clears memories to all 6 digits on the LCD
 */
void clearLCD()
{
    LCDMEM[pos1] = LCDBMEM[pos1] = 0;
    LCDMEM[pos1+1] = LCDBMEM[pos1+1] = 0;
    LCDMEM[pos2] = LCDBMEM[pos2] = 0;
    LCDMEM[pos2+1] = LCDBMEM[pos2+1] = 0;
    LCDMEM[pos3] = LCDBMEM[pos3] = 0;
    LCDMEM[pos3+1] = LCDBMEM[pos3+1] = 0;
    LCDMEM[pos4] = LCDBMEM[pos4] = 0;
    LCDMEM[pos4+1] = LCDBMEM[pos4+1] = 0;
    LCDMEM[pos5] = LCDBMEM[pos5] = 0;
    LCDMEM[pos5+1] = LCDBMEM[pos5+1] = 0;
}

```

```
    LCDMEM[pos6] = LCDBMEM[pos6] = 0;  
    LCDMEM[pos6+1] = LCDBMEM[pos6+1] = 0;  
  
    LCDM14 = LCDBM14 = 0x00;  
    LCDM18 = LCDBM18 = 0x00;  
    LCDM3 = LCDBM3 = 0x00;  
}
```

MSP430 LCD Library Header:

```
/* --COPYRIGHT--,BSD
 * Copyright (c) 2015, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * * Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 *
 * * Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 *
 * * Neither the name of Texas Instruments Incorporated nor the names of
 *   its contributors may be used to endorse or promote products derived
 *   from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/
/*****
 *
 * hal_LCD.h
 *
 * Hardware abstraction layer for the FH-1138P Segmented LCD
 * on MSP-EXP430FR6989
 *
 * February 2015
 * E. Chen
 *
 *****/

#ifndef OUTOFBOX_MSP430FR6989_HAL_LCD_H_
#define OUTOFBOX_MSP430FR6989_HAL_LCD_H_

//Change based on LCD Memory locations
#define pos1 9 /* Digit A1 begins at S18 */
#define pos2 5 /* Digit A2 begins at S10 */
#define pos3 3 /* Digit A3 begins at S6 */
#define pos4 18 /* Digit A4 begins at S36 */
```

```

#define pos5 14  /* Digit A5 begins at S28 */
#define pos6 7   /* Digit A6 begins at S14 */

// Define word access definitions to LCD memories
#ifndef LCDMEMW
#define LCDMEMW      ((int*) LCDMEM) /* LCD Memory (for C) */
#endif

extern const char digit[10][2];
extern const char alphabetBig[26][2];

void Init_LCD(void);
void displayScrollText(char*);
void showChar(char, int);
void clearLCD(void);

#endif /* OUTOFBOX_MSP430FR6989_HAL_LCD_H_ */

```