

Incoming Power Grid Monitoring

Michigan State University
Senior Design – ECE 480 – Team 3

Sponsors

Great Lakes Controls & Engineering

Facilitator

Dean Aslam

Team Members

James McCormick

Alex Lange

Victor Tobenna Ezenwoko

Jacob Jebb

Zhihoug Qian



Executive Summary

The purpose of this project is to develop a three-phase four-wire AC power monitoring system capable of data logging voltage phase-phase, voltage phase-ground, current, and power. The system will alert a user at the occurrence of any transient event, as well as permit the user the ability to change the sampling rate, and possess enough memory to log data for a week with a sampling rate of 100 milliseconds. The system will also have a user interface to allow data viewing, changes for monitoring purposes; and must also allow the user to remove data with a USB memory stick.

Acknowledgements

Team 3 of the Electrical and Computer Engineering Senior Capstone Design class of the College of Engineering, at Michigan State University, would like to thank the following people who assisted us in achieving our goals for this project:

Our sponsors from Great Lakes Controls & Engineering: Special thanks to Justin Walz and Bill Fanson, for their guidance and support in the development of our project. We appreciate all the advice, feedback and resources provided to us at different stages of our project.

Our facilitator: Special thanks to Dr. Dean Aslam, for the continuous insight and guidance he provided as we carried out the project. He ensured we were able to complete our project in due time.

Mike Varney: Special thanks also to Mike Varney, a doctoral student of Dr. Aslam. He provided us with assistance and advice when we came across issues programming our device.

MSU ECE Technical Services: Mr. Greg Mulder and Mrs. Roxanne Peacock, for aiding the team through printing the PCB, as well as, supplying and ordering needed parts.

Division of Engineering Computing Services (DECS): We would like to thank the staff of DECS, most especially Ryan, for aiding us register our device to the MSU engineering network to enable us test some functions of our device in school.

Dr. Timothy Grotjohn, Dr. Lalita Udpa, and all guest speakers: We would also like to thank the above individuals for all the knowledge shared throughout the semester. This information helped us throughout the semester and will continue to be utilized within our professional careers.

Table of Contents

➤ Chapter 1 – Project Introduction	5
Chapter 1.1 – Introduction	5
Chapter 1.2 – Background	5
Chapter 1.3 – Great Lakes Controls & Engineering	7
Chapter 1.4 – Current Designs	8
➤ Chapter 2 - Solution Space and Approach	9
Chapter 2.1 – FAST Diagram	9
Chapter 2.2 – Selecting a Solution	9
Chapter 2.3 – Budget and Expenses	12
Chapter 2.4 – Development Timeline	13
➤ Chapter 3 - Technical Descriptions	15
Chapter 3.1 – Hardware Descriptions	15
Chapter 3.1.1 – Prototype 1	15
Chapter 3.1.2 – Prototype 2	18
Chapter 3.1.3 – Final Prototype	19
Chapter 3.2 – Software Descriptions	22
Chapter 3.2.1 – Stage 1	22
Chapter 3.2.2 – Stage 2	26
Chapter 3.2.3 – Final Stage	29
➤ Chapter 4 - Data Tests	31
➤ Chapter 5 – Design Issues	37
➤ Chapter 6 - Incoming Power Grid Monitor: Final Summary	38
Chapter 6.1 – Summary	38
Chapter 6.2 – Bonus Features	39
Chapter 6.3 – Final Cost	40
Chapter 6.4 – Future Designs	43
➤ Appendix 1 – Personnel	44
➤ Appendix 2 – Literature Review	49
➤ Appendix 3 – Technical Documentation	50

Chapter 1: Project Introduction

1.1 –Introduction

Great Lakes Controls & Engineering were recently contracted with the task of developing a power monitoring system that would be applied in factories which deal with 3 phase power that could possess voltage imbalances and a poor power factor. They had a factory which lost many motors and drives due to bad power caused by voltage imbalances. Therefore, there arose the need for a device able to detect the occurrence of voltage imbalances. Voltage imbalances damage delicate machinery which could be expensive to repair or replace. Poor power factors result in additional charges from power utilities. The developed system should be able to not only detect voltage imbalance occurrences, but must also be able to send out alert notifications about the occurred event.

1.2 – Background

Voltage imbalance is a condition when the voltage to the major 3-phase components is significantly different between the 3 phases. Three phase power consists of 3 “hot” wires each having full line voltage with respect to the other two. These three voltages should be nearly, if not exactly, equal in voltage to each other. If the voltages are too far out of balance, components (like motors and compressors) will start to overheat. Motors and compressors operating with too high a voltage imbalance may continue to run but at elevated motor winding temperatures. This, in turn, will reduce the life of that component. ^[1] Voltage Imbalance is a very common in a 3-phase system with unequal loads. This Imbalance leads to many issues, especially with loads that are attached to all three phases. One such load is a 3-phase induction motor. Many factories utilize these lower-cost ruggedly designed machines to power different parts of their assembly lines.

Unbalanced voltages at motor terminals can cause current imbalance ranging from 6-10 times the percentage of the voltage imbalance. These, in turn, can cause an increase in winding temperature and an overheating problem that can be harmful to and shorten the lifespan of the motor. ^[2] In addition, many solid-state motor controllers and inverters include components that are especially sensitive to voltage imbalances. ^[2] The winding losses due to current imbalance can be calculated using I^2R ; which would show a current imbalance of 10% would result in a 21% increase in the winding losses, as can be seen in the figure below.

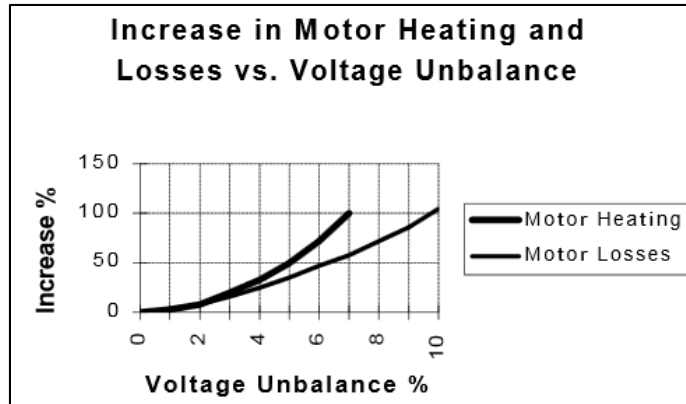


Figure 1. Effects of voltage imbalance on motor heating and losses.^[3]

Voltage imbalance also brings about a reduction in the efficiency of the motor, as the power output remains the same, although there is an increase in the electrical power input due to the increase in current as well as increased resistance due to heating. The increase in resistance and current stack to also contribute to an exponential increase in motor heating. The overheating caused by the excessive power also leads to the thermal degradation of the insulation. The excessive heat leads to winding insulation degradation which causes the windings to short together. Shorted turns in stator windings often do not affect the normal operation of the motor in the early stages, however, depending on the type of short that occurs, they could cause more overheating, instantaneous motor failure, and also lead to complete motor failure.

The adverse effect of unbalanced voltages on induction motors stem from the fact that the unbalanced voltage breaks down into two opposing components that can be described using a method of symmetrical components. Through the use of symmetrical components, we can describe an unbalanced three-phase system of voltage or current phasors using three balanced systems of phasors termed positive, negative and zero sequence.^[4]

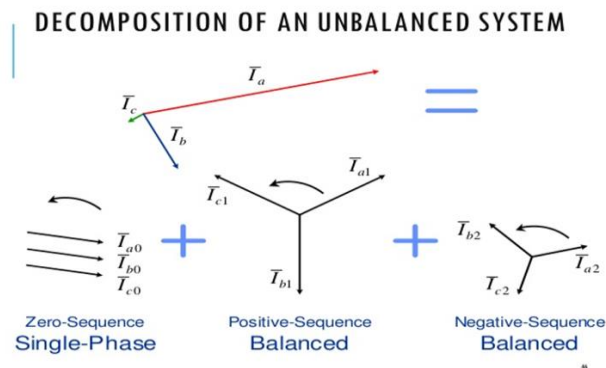


Figure 2. Breakdown of unbalanced three-phase systems

In the case of machines, motors are typically connected delta or ungrounded wye. Therefore, there would be no path to neutral for the zero components to flow. Which

would result in the zero sequence components resulting to zero. Leaving us with just the positive and negative sequence. The positive sequence set consists of the balanced three-phase currents and line-to-neutral voltage supplied by the system. They are all equal in magnitude and phase, rotating at the system frequency with a phase sequence of normally a, b, c. The negative sequence set consists of the balanced three-phase currents and line-to-neutral voltage supplied by the system. They are all equal in magnitude and phase, but with opposite phase rotation. Positive sequence voltage produces the desired positive torque, but the negative sequence voltage produces an air gap flux rotating against the rotation of the rotor producing an unwanted negative torque. [4]

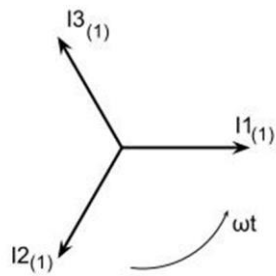


Figure 3a. Positive sequence system

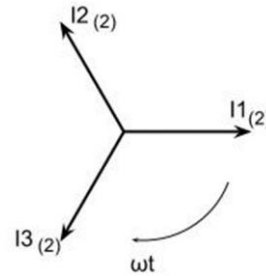


Figure 3b. Negative sequence system

The negative sequence voltage also increase the possibility of torque pulsations and motor noise, due to the effect the negative sequence voltage has on the output torque of the motor. Torque pulsations increase fatigue load on mechanical components, and increase motor noise. The negative voltage sequence components also generate large negative sequence currents due to the low negative sequence impedance. This large negative current increases the machine copper losses and temperatures.

Poor power factors can be problematic for several reasons, particularly at industrial or manufacturing sites where you have motors running heavy equipment. Poor power factor serves as a sign of inefficiency that can result in additional cost associated with the maintenance of the machinery. A poor power factor, in some special cases, could be a sign of three phase power imbalance, which would mean the equipment is working against itself, creating a lot of wear and tear, and generating a lot of heat, which could be a safety hazard. Also, because the reactive power still requires capacity on the system even when it is not doing any useful work, utility/energy companies will charge you more for it.

1.3 – Great Lakes Controls & Engineering

Great Lakes Controls & Engineering (GLC&E) was founded in 2014, with the aim of maximizing profit and productivity through quality engineering and service for

customers. GLC&E provides electrical control system solutions for all aspects of manufacturing. Using the latest in electrical control technology, they provide our customers a quality machine control system while maintaining a safe environment. GLC&E also designs electrical control systems for all types of manufacturing and industrial needs. These include Automatic Screw Machines, Centerless Grinders, Broaching Systems, Parts Washers, Parts Measurement Systems, Robotics and Plant OEE Systems. GLC&E builds complete control systems to any customer specifications. From start to finish, their engineers work with the client to produce a complete control system that exceeds current production needs. GLC&E also deal with troubleshooting and repairing any electrical problems, within electrical control systems already in production, detrimental to production or safety.

1.4 – Current Designs

There are currently multiple power monitoring devices in the market today; for example, the SEL-735 Outdoor Meter, Fluke 435 Series II Three-Phase Power Quality Analyzer, Lutron DW-6092 Power Analyzer, etc. Our sponsor, Great Lakes Control & Engineering, presented us with the EXTECH 3-Phase Power Analyzer/Datalogger as a case study for the project. Although the EXTECH device possesses vast functions for power grid monitoring, like most of the current power monitoring devices, it lack adequate memory for data storage, internet connectivity and the ability to send out alerts at the detection of a voltage or current imbalance. We are expected to create a new power monitoring device which would replicate the primary functions of the Extech device, with additional support features solving some of its shortcomings.



Figure 4. Extech 3-Phase Power Analyzer/Datalogger ^[5]

Chapter 2: Solution Space and Approach

2.1 – FAST Diagram

We began the project by closely examining the problem statement and end goals of our project. Afterwards, we developed a Function Analysis System Technique (FAST) Diagram, which can be seen in Figure 5, to establish and discuss the basic primary and secondary functions of our desired power monitoring system.

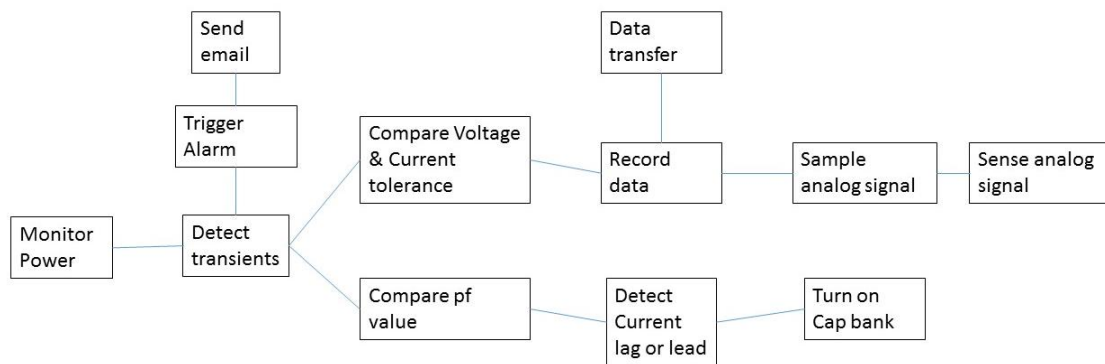


Figure 5. FAST Diagram of system goals

This diagram provided the team with a high level view of the overall functionality and purpose of our desired device. The primary function/goal of our project, seen on the far left of our FAST Diagram, is to monitor power. The secondary functions leading up to the primary goal can be seen, on our FAST Diagram, to the right of the primary function. The FAST Diagram was created and organized in such a manner asking “How?” or “Why?” depending on what direction you look at it. As you move left, it shows the different aspects involved in achieving the particular function. As you move right, it shows you the purpose of a particular function. Analyzing each successive subsystem of operation in both the how and why direction, provided clarity of design requirements and enabled concise formulation of development plans.

2.2 – Selecting a Solution

In order to properly select and develop the best solution, we developed a Voice of Customer interview guide and had a meeting with our sponsors, which enabled us to get a better understanding of our sponsor’s desired outcome of the project. After discussing with our sponsor we developed a flow diagram, which can be seen in Figure 6, to give a basic outline of the project and its different components. The diagram also included some bonus features the team decided to include to the system.

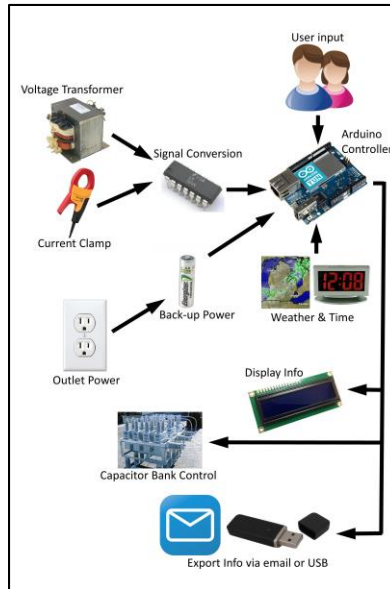


Figure 6. Developed flow diagram for the project

A vital part of our project involved selecting a microcontroller we could program to carry out the necessary functions desired by our sponsor. We were initially torn between the Arduino Uno and Arduino Yun. However, the Arduino Yun proved to be the obvious choice, as it offered better solutions. Although Arduino is seen as more of a hobbyist controller, given our short timeframe for the project and the inexperience of the team in terms of computer programming, this microcontroller serves as the best choice for multiple reasons. This selected microcontroller also has 6 analog inputs for our signal sampling, an Ethernet port, microSD port, a USB port, and enough digital input/output ports to write to the LCD screens and our pushbutton inputs. With this controller, we can sample the signals, program any settings with the pushbuttons, save data to the microSD, send emails, and export data to a USB drive.

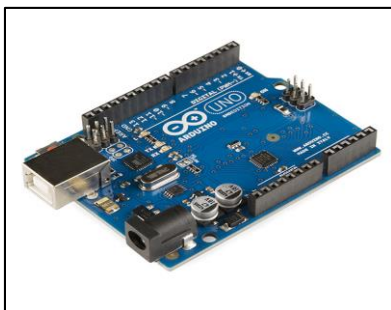


Figure 7a. Arduino Uno



Figure 7b. Arduino Yun

A limiting factor of our selected microcontroller was the number of inputs/outputs, as we would need a total of nine digital pins on the microcontroller in order to control the four LCD screens. Seeing as we would need pins for various applications, there was a need to reduce the number of pins used for the LCDs without

reducing the number of LCD's. In order to do this we would have to use I²C communication, therefore, we would need to buy modules that allow the LCD's to use this form of communication by giving the LCD's an address, where no LCD shares the same address.

The next part of the design we pondered upon was decided how many units our project would consist of. We came up with an initial design concept for the overall unit, which can be seen in Figure 8, however we were initially unsure if we would have the power transformation section within the overall unit or if we would have the power stepped down in a separate box before being led into a separate unit with the circuitry and microcontroller. Due to health and safety concerns, we decided to go two separate units. One would be the main hub of the project, while the latter would consist of the transformers to break down the high level voltages, as such high rated voltages would be hazardous for both the project parts and personal health.

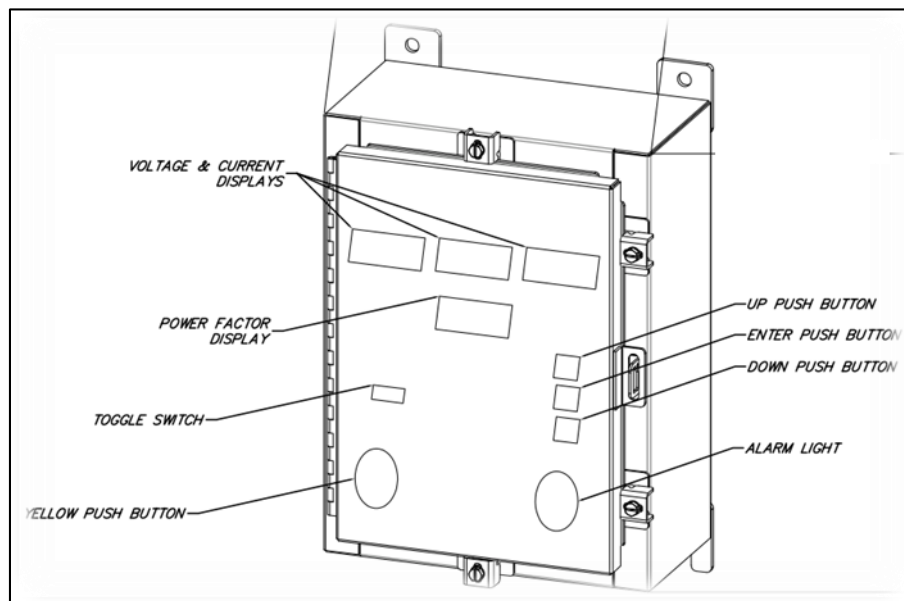


Figure 8. Proposed unit design

To ensure we made the best decisions while exploring possible solutions for our project, we employed the use of a selection matrix. The selection matrix rated the importance of the different engineering criteria's on a scale of 1 to 5, with 5 being the highest rating; and rated the different possible solution combinations based on how well they met the engineering design criteria, these were rated 1, 3, or 9, which represented ratings of weak, moderate or strong respectively. The developed selection matrix, as well as its legend key, can be seen in Table 1 and 2.

Engineering Criteria	Importance	Possible solutions			
		AU, 1 Bx	AU, 2 Bx	AY, 1Bx	AY, 2Bx
Internet capability	5	3	3	9	9
User friendly	3	9	9	9	9
Cost	1	9	9	3	3
Ability to safely measure voltages up to 600 VAC	5	3	9	3	9
Data removal	5	3	3	9	9
LCD Data Display	5	1	1	9	9
Total		86	116	180	210

Table 1. Selection Matrix

AU	Arduino Uno
AY	Arduino Yun
Bx	Enclosed box(s)

Table 2. Selection Matrix Legend

2.3 – Budget and Expenses

The team was initially assigned a budget of \$500 by the University to spend on any parts or resources needed for our project; however, after meeting and discussing with our sponsor, Great Lakes Controls & Engineering, we were instructed not to lay much emphasis on the budget as they would provide any needed necessary components they had or would order any component which would go over the University given budget. Based on this knowledge, components provided by the sponsor have been kept out of the budget; components used in the circuit which were free and provided by the ECE shop were also excluded from the budget.

The expected budget breakdown for the project can be seen in table 3.

Quantity	Item	Cost
2	Arduino YÚN Microcontroller	\$138.54
4	Basic 16x2 Character LCD - White on Black 5V	\$39.80
4	I2C Serial Interface Board Module LCD	\$8.36
1	64GB MicroSD Card	\$35.99
	Total:	\$222.69

Table 3. Breakdown of expected expenses

Minus some of the more expensive components ordered by our sponsor, we believe the project to be cost-friendly enough to be replicated in a large scale setting. The

microcontroller proved the highest expected expense, however, this is because we ordered a spare microcontroller we would use in implementing and testing the programming of our project. If the produced were to be replicated or reproduced on a larger scale, only one microcontroller would be needed, which would see our total expected expenses drop down to \$153.42.

2.4 –Development Timeline

In order to successfully complete our project in due time and maximize efficiency, the team designed a time schedule for this project in the form of a GANTT chart. The GANTT chart lists expected dates to start and complete different aspects of the project. The GANTT chart can be found below broken down into Figure 9a, 9b, and 9c.

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessor	Resource Names
1		Project Definition	6 days	Wed 9/9/15	Wed 9/16/15		
2		group meet, outline questions of topic	2 days	Wed 9/9/15	Thu 9/10/15		Everyone
3		meet with sponsor, address questions,	2 days	Fri 9/11/15	Mon 9/14/15	2	Everyone
4		group meet, define project in words	1 day	Tue 9/15/15	Tue 9/15/15	3	Everyone
5		meet with facilitator, present topic	1 day	Wed 9/16/15	Wed 9/16/15	4	Everyone
6		Research Topic	3 days	Thu 9/17/15	Mon 9/21/15		
7		Research professional monitoring systems	2 days	Thu 9/17/15	Fri 9/18/15	5	Everyone
8		Research Hobby monitoring systems	2 days	Thu 9/17/15	Fri 9/18/15	5	Everyone
9		Brainstorm individual ideas	1 day	Mon 9/21/15	Mon 9/21/15	8	Everyone
10		Combine rough Ideas	1 day	Tue 9/22/15	Tue 9/22/15		
11		meet with group, discuss ideas	1 day	Tue 9/22/15	Tue 9/22/15	9	Everyone
12		define feasible ideas	1 day	Tue 9/22/15	Tue 9/22/15	9	Everyone
13		choose one direction to head in	1 day	Tue 9/22/15	Tue 9/22/15	9	Everyone
14		Ordering Parts	26 days	Thu 9/24/15	Thu 10/29/15		
15		Order microcontroller	7 days	Thu 9/24/15	Fri 10/2/15	29	Alex
16		Order ethernet shield	7 days	Thu 9/24/15	Fri 10/2/15	29	Alex
17		order USB shield	7 days	Thu 9/24/15	Fri 10/2/15	29	Alex
18		Order LCD screens	5 days	Mon 10/5/15	Fri 10/9/15	17	Alex
19		Order push buttons/knob	5 days	Mon 10/5/15	Fri 10/9/15	48	James
20		Order AC loss alarm relay	5 days	Mon 10/5/15	Fri 10/9/15	47	James
21		Order signal conversion supplies	5 days	Fri 10/9/15	Thu 10/15/15	51	James
22		Order AC current clamps	5 days	Fri 10/23/15	Thu 10/29/15	55	James
23		Order cap control relay	5 days	Thu 10/1/15	Wed 10/7/15	37	James
24		Order plexiglass	5 days	Fri 10/23/15	Thu 10/29/15	55	James
25		Order step down transformers	5 days	Fri 10/23/15	Thu 10/29/15	55	James
26		Order wire/terminals/rails	5 days	Fri 10/23/15	Thu 10/29/15	55	James
27		Order final box	5 days	Fri 10/23/15	Thu 10/29/15	55	James
28		Micro-controller	11 days	Wed 9/23/15	Wed 10/7/15		
29		Research microcontrollers to use	1 day	Wed 9/23/15	Wed 9/23/15	13	Jake/Alex
30		Develop code to write to LCD screen	1 day	Wed 9/23/15	Wed 9/23/15	13	Alex

Figure 9a. GANTT chart breakdown

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessor	Resource Names
31		Develop code to read RMS values	2 days	Wed 9/23/15	Thu 9/24/15	13	Alex
32		Develop sample values to test RMS code	1 day	Fri 9/25/15	Fri 9/25/15	31	Victor
33		Develop code to send emails	3 days	Wed 9/23/15	Fri 9/25/15	13	Jake
34		Develop code to read weather from	3 days	Wed 9/23/15	Fri 9/25/15	13	Victor
35		Develop code to write to USB	3 days	Wed 9/23/15	Fri 9/25/15	13	Alex
36		Develop code to calculate PF	2 days	Mon 9/28/15	Tue 9/29/15	32	Alex
37		Develop code for cap control output	1 day	Wed 9/30/15	Wed 9/30/15	36	Alex
38		develop code to calculate True power	1 day	Wed 9/30/15	Wed 9/30/15	36	Alex
39		Develop code for pushbutton/alarm inputs	1 day	Thu 10/1/15	Thu 10/1/15	38	Alex
40		Develop code for processing inputs	4 days	Fri 10/2/15	Wed 10/7/15	39	Alex/Jake/Victor
41		Hardware	12 days	Wed 9/23/15	Thu 10/8/15		
42		Research step down transformers	3 days	Wed 9/23/15	Fri 9/25/15	13	James
43		research push buttons	3 days	Wed 9/23/15	Fri 9/25/15	13	James
44		research battery packs	3 days	Wed 9/23/15	Fri 9/25/15	13	James
45		research power supplies	2 days	Mon 9/28/15	Tue 9/29/15	44	James
46		design power circuit	2 days	Wed 9/30/15	Thu 10/1/15	45	James
47		design loops of AC circuit	1 day	Fri 10/2/15	Fri 10/2/15	39	James
48		design knob/push button inputs	1 day	Fri 10/2/15	Fri 10/2/15	39	James
49		design USB port interface	3 days	Mon 9/28/15	Wed 9/30/15	35	Victor
50		research current clamps	3 days	Thu 9/24/15	Mon 9/28/15	29	James
51		design signal conversion to	10 days	Fri 9/25/15	Thu 10/8/15	31	James/Jake/Victor
52		Prototype 1	6 days	Thu 10/15/15	Thu 10/22/15		
53		Build sample enclosure	1 day	Thu 10/15/15	Thu 10/15/15	73	James
54		brainstorm layout	1 day	Fri 10/16/15	Fri 10/16/15	53	Everyone
55		assemble parts	4 days	Mon 10/19/15	Thu 10/22/15	54	Everyone
56		Final design	12 days	Fri 10/23/15	Mon 11/9/15		
57		Brainstorm layout	2 days	Fri 10/23/15	Mon 10/26/15	55	Everyone
58		Assemble parts	7 days	Fri 10/30/15	Mon 11/9/15	27	Everyone
59		Web Design	33 days	Wed 9/23/15	Fri 11/6/15		
60		research old websites for ideas	2 days	Wed 9/23/15	Thu 9/24/15	13	Everyone

Figure 9b. GANTT chart breakdown

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessor	Resource Names
61		meet with group, create outline	1 day	Fri 9/25/15	Fri 9/25/15	60	Everyone
62		start skeleton of website	5 days	Mon 9/28/15	Fri 10/2/15	61	Zhihong
63		fill in body	10 days	Mon 10/5/15	Fri 10/16/15	62	Everyone
64		update	10 days	Mon 10/19/15	Fri 10/30/15	63	Zhihong
65		finish website	5 days	Mon 11/2/15	Fri 11/6/15	64	Zhihong
66		Testing	36 days	Mon 9/28/15	Mon 11/16/15		
67		Complete ethernet interface	3 days	Mon 10/5/15	Wed 10/7/15	16	Everyone
68		Complete sending emails	3 days	Mon 10/5/15	Wed 10/7/15	16	Everyone
69		Complete writing to USB	3 days	Mon 10/5/15	Wed 10/7/15	17	Everyone
70		Complete reading sample RMS values	3 days	Mon 9/28/15	Wed 9/30/15	32	Everyone
71		Complete Reading analog RMS values	3 days	Mon 10/5/15	Wed 10/7/15	15	Everyone
72		Complete display on LCDs	3 days	Mon 10/12/15	Wed 10/14/15	18	Everyone
73		Complete pushbutton/alarm inputs	3 days	Mon 10/12/15	Wed 10/14/15	20	Everyone
74		Complete Cap control	3 days	Thu 10/8/15	Mon 10/12/15	23	Everyone
75		Complete prototype	5 days	Fri 10/23/15	Thu 10/29/15	55	Everyone
76		Complete final design	5 days	Tue 11/10/15	Mon 11/16/15	58	Everyone
77							

Figure 9c. GANTT chart breakdown

Chapter 3: Technical Descriptions

3.1 – Hardware Descriptions

3.1.1 – Prototype 1

We began our project by creating our initial prototype, as seen in Figure 10.

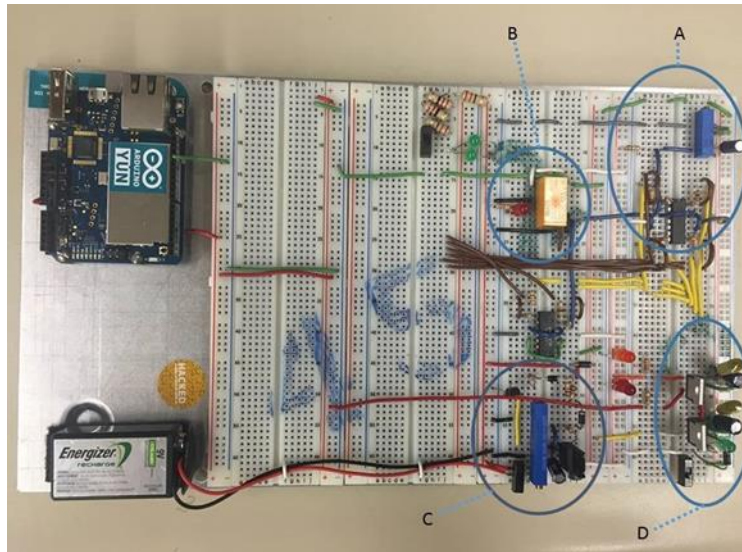


Figure 10. Initial prototype

The most vital part of our design is the circuit's ability to take an input voltage or current signal and convert it to a format our microcontroller can sample. Our microcontroller can only read voltages ranging from 0 to 5 V. To solve this issue, we designed a summing amplifier that would scale down the input signal and also add a 2.5 V DC offset, which would enable us to properly see the sine wave of the voltage or current input signal.

We plan to use transformers that will step down the signals to a 5 Vrms voltage; therefore, we will have an input of roughly 15 Vpp. If we build an inverting summer, the signal will be inverted which would not distort any information as long as each signal was inverted. We can also choose resistors such that the input signal is scaled down by a value of 4.7. If we are able to get a 5 Vpp input to the microcontroller, it would mean our input to the signal conversion circuit would be, $4.7 \times 5 = 23.5$ Vpp, well suited to handle the expected 15 Vpp signal. We used to quad operational amplifiers to realize this design as can be seen in part A of Figure 10.

To create the DC offset, we used a voltage divider with a variable pot to grant us the ability to control the output offset through calibration. The input terminals of the

operational amplifier have a high impedance so there is little concern of loading effects while using a voltage divider and to create a stiff voltage, we used capacitors tied to ground. The signal conversion circuit schematic can be found in Figure 11 below:

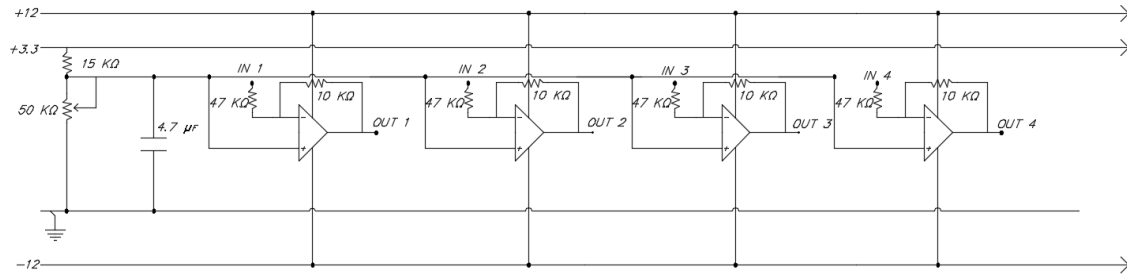


Figure 11. Signal conversion unit

To power our circuit, we plan to use a 120AC/12VDC converter. However, our micro controller does not have an on board regulator and requires a regulated 5 VDC power supply. To accommodate this requirement, we built a circuit to provide a regulated 5 V and 3.3 V source incase if needed. The design comes from research and availability of parts from the ECE 480 class lab at Michigan State University. We added LEDs with current limiting resistors to give an indication of a voltage presence at the output of each voltage regulator. The circuit can be seen in part D of Figure 10 and the schematic for the circuit can be found in Figure 12 below:

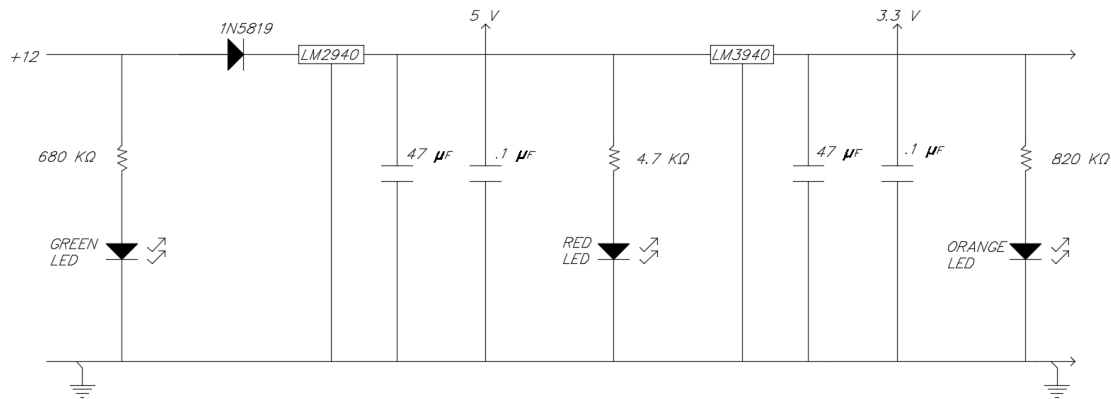


Figure 12. Input voltage circuit

Another major issue we wanted to overcome, not included in the design specifications, was the concern for a loss of power. The product is intended for use in a factory setting, where after it is installed, there could possibly be no human interaction other than when an alarm rings out. Therefore, if the unit undetectably loses power, any voltage or current transients would go unnoticed and no email notification will be sent. If no transients are suspected, no one will think to have any interaction with the product. As a resolution, we designed a backup battery with a charging circuit.

To build a battery charging circuit that would have an overcharge protection, we referred to a conceptual design in Dr. Wierzba's ECE 402 class notes ^[6]. The circuit can

be seen in Figure 13 below. We used an LM317 voltage regulator so that we could vary the output voltage using a potentiometer. If the voltage charging the battery gets too high, the zener diode will conduct and forward bias the transistor. When the transistor is forward biased, the output of the voltage regulator will drop and the diode at the top of the circuit will prevent the battery from backfeeding the regulator. We chose a 9 V rechargeable battery that has a nominal voltage of 8.4 V. The components and settings of the circuit were chosen such that the battery will charge to 8.6 V with a maximum current of 20 mA. These parameters came from research of the specifications of the battery charger designed for this battery.

We also included a discharge design in this circuit, which utilizes a switch to add a load to the battery for testing purposes. This is made of a series of four 22- Ω resistors.

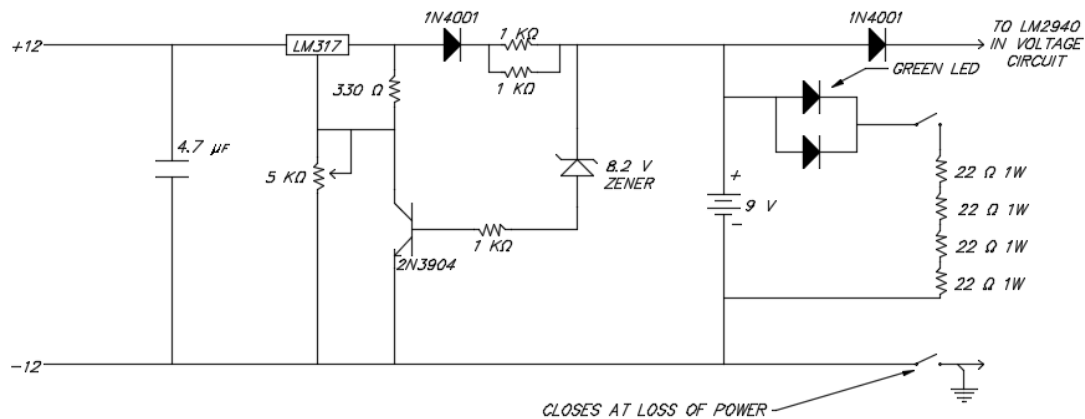


Figure 13. Battery charger circuit

In order for the battery charger to work, a 24 V supply was needed. However, when the battery is required to power the circuit, the negative terminal must be connected to the circuit ground. To solve this issue, we used an undervoltage relay to close a contact that shorts the negative 12 V supply to ground when there is a loss of power. We also used a contact on the relay to turn on a red LED to signal the loss of input power. We plan to use inverted logic as an input to the micro controller to signal a loss of power. If the input is shorted to ground, external power is present, and if it is floating, there is a loss of power. The schematic for the circuit can be found in Figure 14 below and a picture can be seen in part B of Figure 10.

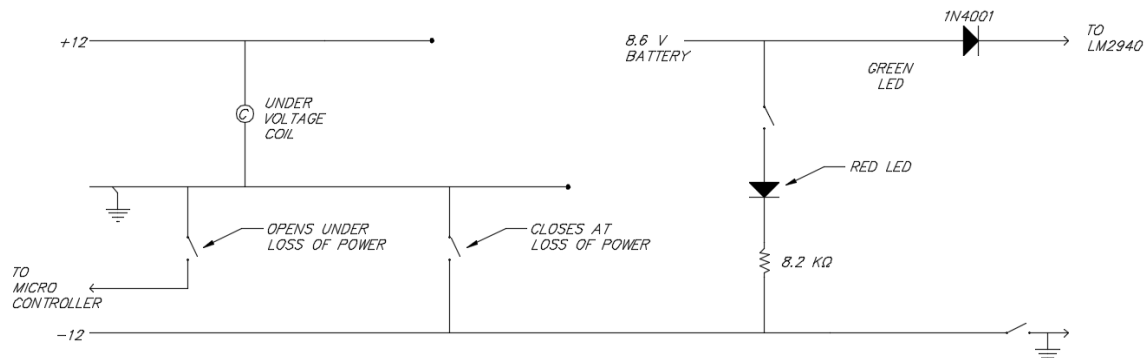


Figure 14. Loss of AC switchover circuit

3.1.2 – Prototype 2



Figure 15. Prototype 2

Building upon our initial prototype, we built prototype 2, which can be seen in Figure 15, we went further in the development of our unit. We built a wooden enclosure to represent the overall unit our circuitry would be placed. Next:

- We added transformers to step down voltage from 120V to 13.5V, and added fuses for our transformers as a precaution in case we had any wiring errors. The transformers took down 120V, fused it and put it to a power supply to bring it down to positive and negative 12 volts DC, then connected it through a switch to turn the system on or off for testing purposes. We then connected the positive and negative 12V to the board to replace the power supplies we had been using for testing, before adding a fan for the heat sink that went from 12V to 5V. Finally, we also connected a transformer to 120VAC, ran it through a fuse and connected that to the board where we would read the voltage using our differential op-amp.
- We also design and implemented a capacitor bank control circuit. The schematics for this circuit can be seen in Appendix 3 of this report. The 12V relay turns off when the transistor gets cut off, because there would be no current going through the coil. When we forward bias the transistor, V_{CE} goes to 0, and there would be

12V across the relay coil, which contacts then switches states, the 12V then goes through the limiting resistor, and the yellow LED comes on. When the 120V contact closes and brings 120V to the large relay, the coil picks up, then closes the large relay cap contact. We designed it this way because small relays can only land 500mA on contacts, which is enough to pick up large relay coil but not enough for external connections. The large contact can handle up to 250V and 10A, and the extra relay provides us with backfeed isolation from our PCB.

- We also connected our LCD screens to the device. To avoid using all of our microcontroller pins in displaying, we opted to use I²C communication with adapters on the screens.
- We also added 4 momentary push buttons to the right of the LCD. The microcontroller samples a digital pin that is automatically set to “pull-up”, but gets grounded when pushed. We also used this same logic in designing the knob/switch control that would be used in switching from line-to-line voltage to line-to-neutral voltage. When ungrounded the program is set to display V_{L-L} , and display V_{L-N} when grounded. For this prototype, we used push buttons in place of a switch.

3.1.3 – Final Prototype

Building upon our prototype 2, we began the development of our final prototype/unit. To do this:

- We designed and implemented the alarm light circuit. The circuit works the same way with the transistor forward bias to turn relays on. When the relay is turned on, 120VAC goes to the alarm light to turn it on, and the bottom of the relay circuit “locks itself in”. When the contacts change state, the bottom side of the coil is grounded through the push button. Even if the transistor is not forward biased again, the relay stays locked in, until the button is pushed, after which the circuit breaks and the coil would no longer be connected to the ground. The relay turns off, and the 120VAC would not be connected to the light again, turning it off.
- We designed and implemented the kill circuit. At the event of a loss of power to the system, the backup battery would kick in to enable the microcontroller send out an email notifying the user about the loss of power, after the email has been sent this circuit would then “kill” the power to the microcontroller.

- We then designed and printing the PCB for our final unit, then transferred and soldered our circuitry unto the PCB. The PCB with the circuitry can be seen in Figure 16.

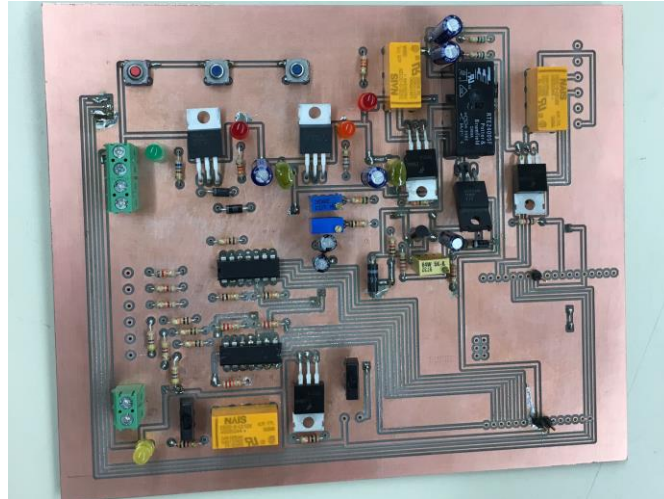


Figure 16. PCB

- We also added a second power supply to our design. Although our unit can operate solely with the 12V power supply, we added this additional power supply because we did not have enough time to build a switching regulator that could more efficiently step down 12V DC to 5V DC. This eliminated the need for the fan we had in prototype 2, as the fan was used to cool down the linear regulator which would get too hot.
- At this point, the enclosure for our final unit arrived, so we proceeded to sawing and drilling the enclosure to accommodate the LCD screens, alarm light, V_{L-L} and V_{L-N} switch, alarm light push button, and wires and cables into the unit. This can be seen in Figure 17a and 17b.



Figure 17a. Top view of the enclosure

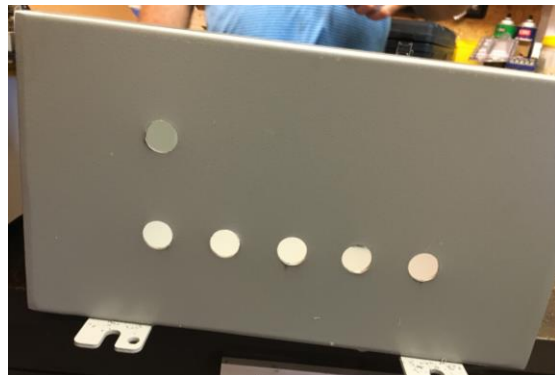


Figure 17b. Bottom view of enclosure

- Finally, we added a window/screen in front of the unit, then put the whole unit together, by inputting the LCD screens, PCB, power supplies, relays, etc. We used either self-tapping screws or taps to connect everything within our box to avoid the necessity for tightening objects from the back side. We were especially careful in the selection of wire color for each component and used a wire code similar to how our sponsor builds their panels to avoid confusion once the design is implemented. Much care was also taken to ensure the wires were labeled and organized in a comprehensible and neat manor. Industry standards were also followed when landing any outgoing signal on the exterior of the terminal block, and interior functions on the other side. Because our design is intended for direct implementation into a professional environment and is not just a simple prototype concept design, we put in extra effort to ensure our device met professional standards. The goal of this final step was to deliver the final product as any other professional engineering firm would. The final unit and its internal components can be seen in Figure 18 and Figure 19 respectively.



Figure 18.

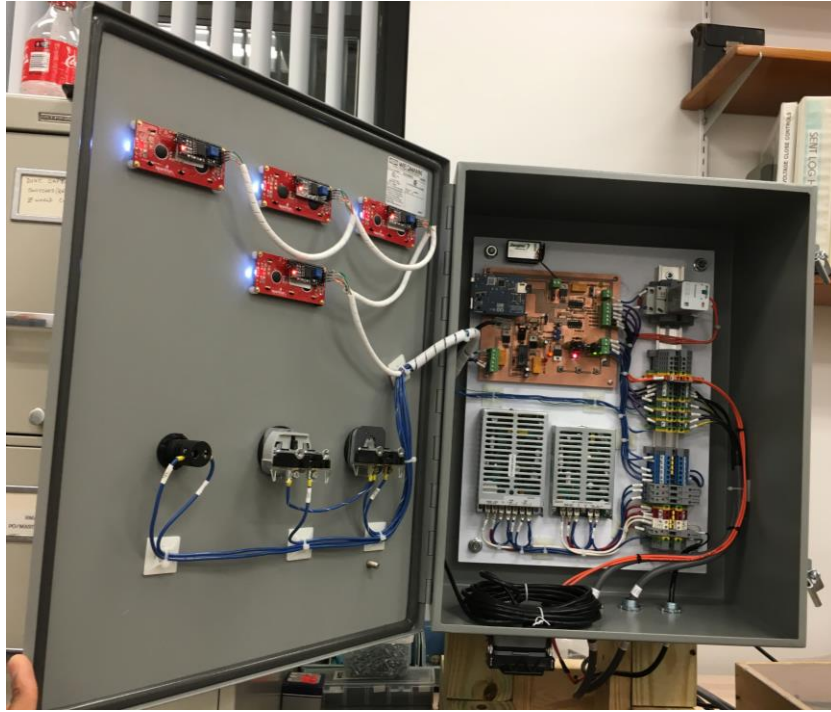


Figure 19. Inside our unit

3.2 – Software Descriptions

For our software design, there were three main stages utilizing two different programming and testing methods. The first stage involved using an Arduino Uno to start testing RMS calculations before the arrival of final microcontroller. The second stage occurred upon the arrival of the Arduino Yun and completion of sufficient amount of the hardware. The final stage began upon the total completion of the hardware and took place with the final design of all circuitry.

3.2.1 – Stage 1

The first stage of prototyping consisted of using an Arduino Uno, belonging to a member of the team, as well as an LCD to calculate and display the RMS value of a 5V peak-to-peak sine wave. With the Arduino Uno we were able to create an array of length 16 that when read continuously would create a sine wave. This was very useful in determining if the Arduino was accurately calculating the RMS value. The Uno could create its own sine wave to read, which proved beneficial as hardware was not needed. It was also used to test the button functionality because at the time we still planned on utilizing push buttons to change different settings. A picture of the test setup can be seen

in Figure 20. To illustrate the accuracy, the RMS of a 5V peak-to-peak sine wave can be calculated using:

$$Amplitude = \frac{V_{peak-peak}}{2} = 2.5$$
$$RMS = \frac{Amplitude}{\sqrt{2}} = 1.7678$$

Again, looking at Figure 20, the Uno is calculating an RMS value of 1.76. This gives an error of .439%; this minimal error can be attributed to the program rounding down rather than up.



Figure 20. Stage 1 Prototype – Arduino Uno, LCD, and buttons

The next two stages of prototyping utilized two separate Arduino Yun microcontrollers for development. The first Arduino Yun interfaced with the hardware, while the latter Arduino Yun was used independently to test network capabilities and data transfer. The independent Arduino Yun will be discussed first as it was used concurrently with the final two stages that will be discussed. With the independent Yun, the following programming and testing was completed: sending an email, logging data into a MicroSD card, wirelessly downloading the logged data from the MicroSD card to any computer connected to the same network, downloading real time from the internet to be logged with the data, and finally, allowing an operator to change settings, such as the email address, wirelessly. We developed each of those functions separately, before then implementing them into the Arduino Yun that was attached to the hardware, which provided us with the benefit of identifying bugs within each code. If we attempted to implement a few functions at a time an error could be attributed to any of those functions.

One of the biggest improvements we implemented to our device was the enabling a user change settings wirelessly. There is a txt file on the MicroSD card that the Arduino Yun reads a variety of settings off of. Using a software called FileZilla, an operator can connect to the Yun and upload a new Settings.txt file to the MicroSD. To transfer data wirelessly from the Yun to a computer or vice versa, we used a FTP (File Transfer Protocol) software. FileZilla offered a simple user friendly interface to achieve our goal of data transfer between an operator and the Yun. Using the Yun's MAC address and associated password, FileZilla can be synced up to the Arduino Yun, seen in Figure 21.

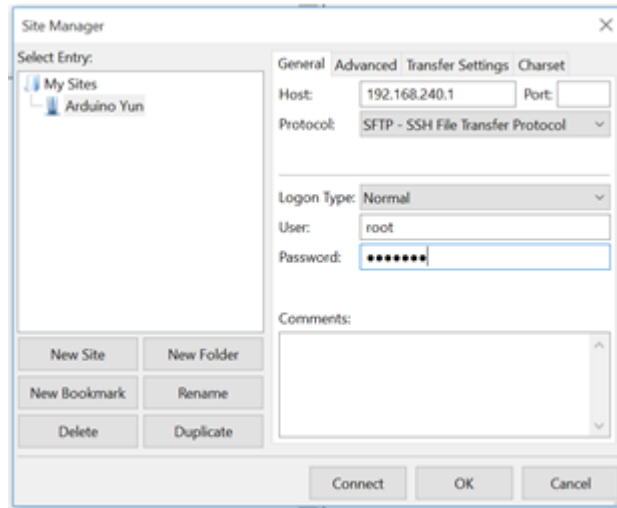


Figure 21. Connecting to the Yun

Figure 22 shows the folders located on the MicroSD card on the Yun. To change the settings or pull the data, it is as simple as treating the Yun's MicroSD card as a folder on your computer so the operator can just copy the data logging file and paste it to their computer.

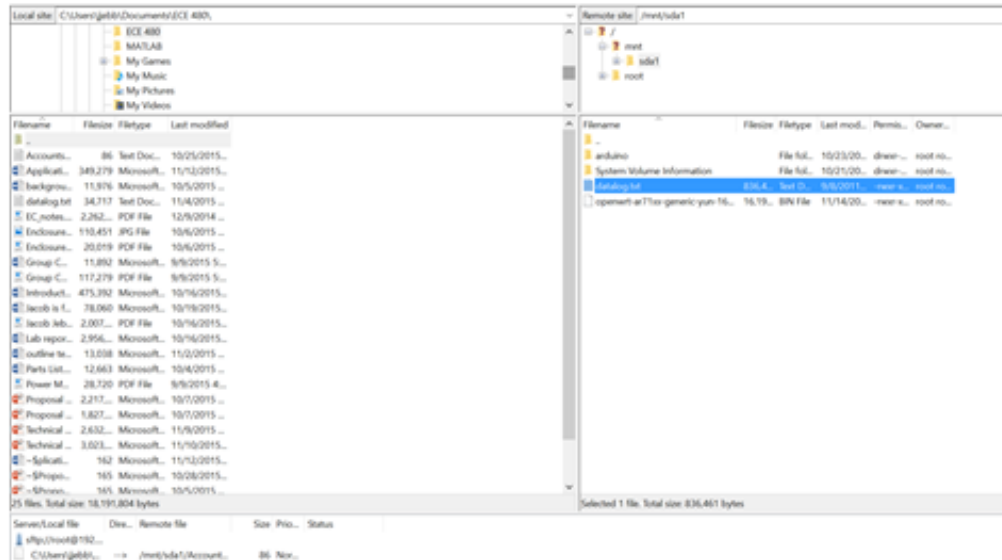


Figure 22. Transferring the Data from the Yun to Computer

The same procedure goes for uploading new settings, the operator simply saves the new settings to the Arduino Yun's MicroSD. The Arduino Yun will then automatically update the settings based on this new txt file. An example of the possible settings can be seen in Figure 23. By simply changing the value on the right side of the '=' sign, the Yun will incorporate any updates

```
[NumSamples=1000] *Number of points used to calculate RMS
[voltageCutoff=128] *Define low voltage for sending an alert - Int/1024 = % of max
[currentCutoff=61] *Define low current for sending an alert - Int/1024 = % of max
[voltageScale=340] *Maximum value the voltage can be
[currentScale=50] *Maximum value the current can be
[TO_EMAIL_ADDRESS=Example@msu.edu] *Email address for sending alerts
```

Figure 23: Settings.txt

In order to allow comments in the text file for the operator to reference, the code in Figure 24 was used. The code reads through each character but only records the values between the two brackets. Between the first bracket and the equal sign of Figure 23, we can find the variable name that can be changed. After the equal sign is the integer or string for which that variable should be set to. This method enables the operator to make changes without having to physically interact with the unit.

```

String settingName; // variable name of the setting to change
String settingValue; // value of the setting
File Settings = FileSystem.open("/mnt/sd/Settings.txt"); //opens the txt file
if(Settings){
  while(Settings.available()){
    character = Settings.read();
    while((Settings.available()) && (character != '[')) //read until first '['
      character = Settings.read();
  }
  character = Settings.read();
  while((Settings.available()) && (character != '=')){ //reads setting name stops
    settingName = settingName + character; //when it reaches '='
    character = Settings.read();
  }
  character = Settings.read();
  while((Settings.available()) && (character != ']')){ //reads value, stops
    settingValue = settingValue + character; //when it reaches ']'
    character = Settings.read();
  }
  if(character == ' '){
    applySettings(settingName, settingValue); //used to convert string into
    settingName = ""; //appropriate format
    settingValue = "";
  }
}
Settings.close();
}

```

Figure 24: Reading Settings.txt

The unit's ability to log a time stamp when sending an email, was of the utmost importance, as it would be difficult to diagnose the cause of voltage imbalance without accurate real-time data. If voltage imbalance is detected, an email will be sent to the operator alerting them of it. The operator could then wirelessly download the logged data from the Arduino Yun's MicroSD card. The RMS value versus time can then be plotted in Excel to see what time the voltage transient occurred at. An example of how this can be useful, would be if the voltage imbalance was as a result of one of the phases becoming overloaded. The logged time can be utilized to determine the equipment in use when imbalance was detected, as well as which phase is experiencing the overload. The operator can then determine how to better distribute the power consumption.

3.2.2 – Stage 2

The second stage of the software prototyping involved interfacing an Arduino Yun with the prototype hardware. The first step was changing the code to read the real sine for current and voltage through two analog pins, rather than the simulated sine wave we had used. At this point, the LCDs with I²C communication modules arrived. I²C is a

type of communication that only requires two lines that act as buses. Devices can be connected in parallel, with each possessing its own unique address. The Arduino Yun can then address each LCD individually to tell it what to print. The benefit of using I²C communication is that, while the standard method would require more than 10 digital pins, only two pins are needed. Now that the Arduino Yun was receiving a signal for current and voltage, power and power factor could be calculated and displayed on one of the screens. Power was calculated by simply multiplying the instantaneous current by the instantaneous voltage. Power factor was determined by dividing the real power by apparent power.

The second stage also saw us implement a few bonus features. The first being a backup battery, which involved programming the Arduino Yun to send out an email notification at the event of a loss of power to the system, and then turning itself off. We programmed the Arduino Yun to kill itself as a precautionary measure, because if the battery voltage goes too low the Arduino Yun will be undervoltaged and could get damaged. As soon as power is restored to the unit, the Arduino Yun will turn back on and resume normal operation. The second bonus feature implemented during this stage was the ability to control a capacitor bank. If power factor drops below a certain value, it is sometimes desired to close a capacitor bank. We utilized hysteresis loop in the code logic utilized for the capacitor bank control. The power factor value for opening/closing a capacitor bank can be designated by the operator so the following numbers are used just as an example. The hysteresis loop logic was set up to have the capacitor bank close if the power factor drops below 0.81, and remain closed until the power factor reaches a value of 0.90. If the power factor is between 0.81 and 0.90 the capacitor bank will simply hold its current state. The code used for control of the capacitor bank can be seen in Figure 25. A timer must also be implemented to keep the capacitor bank from opening/closing rapidly. The value this timer counts to can be changed very easily so that the operator can determine how quickly he wants the capacitor bank to be able to switch states. The capacitor bank is controlled simply by writing 5V to a digital pin that is connected to a relay. If the Arduino Yun stops outputting a voltage to that digital pin, the relay will open.

```

// *****
// *****Cap Bank Control*****
// *****
if(loopTime - taskTime[3] > taskPeriod[3]){
    taskTime[3] = loopTime-1;
    if(CapBankTimer < CapBankDelay){
        CapBankTimer = CapBankTimer+NumSamples;
    }
    else if(PowerFactor<.81 && CapBankState == 0){
        digitalWrite(CapBankPin, HIGH);
        CapBankTimer = 0;
        CapBankState = 1;
    }
    else if(PowerFactor>.9 && CapBankState == 1){
        digitalWrite(CapBankPin, LOW);
        CapBankTimer = 0;
        CapBankState = 0;
    }
}
}

```

Figure 25: Capacitor Bank Control

The final part of code implemented during this stage was the control of an alarm light. The operator can designate a percentage of voltage imbalance that is tolerable, we will use 5% as an example. A range of acceptable values is then determined to be 95% to 105% of the average voltage. If any of the three voltages does not fall within this range, the alarm light is tripped and an email is sent. Similar to the capacitor bank, the alarm light is tripped simply by writing 5V to one of the Arduino Yun's digital pins. The code for checking voltage imbalance can be seen in Figure 26. InputReal is an array of length six where the first three values are the three phase voltages and the last three values are their corresponding phase currents. The 'for' loop steps through the first three values in the array and compares them to the range of acceptable voltage values.

```

// *****
// **Voltage Imbalance and Alarm**
// *****
VoltageReal_Avg = (InputReal[1]+InputReal[2]+InputReal[3])/3;
ImbalanceOver = (1+ImbalanceTolerance)*VoltageReal_Avg;
ImbalanceUnder = (1-ImbalanceTolerance)*VoltageReal_Avg;
for(x = 1; x<4; x++){
    if(InputReal[x] < ImbalanceUnder || InputReal[x] > ImbalanceOver){
        digitalWrite(AlarmLight, HIGH);
    }
    else{
        digitalWrite(AlarmLight, LOW);
    }
}
}

```

Figure 26: Checking for Voltage Imbalance

3.2.3 – Final Stage

The final design consisted of having a printed circuit board for all the electronics as well as having three phase power, meaning instead of one voltage and one current there are three of each. As explained in the earlier section, arrays were utilized for the inputs to cut down on repeating similar code. RMS is calculated the same way as current and voltage so creating 'for' loops to do the same calculation six times saves flash memory. An example of this can be seen in Figure 27. In order for the Arduino Yun to get values it can read, a 2.5 volt DC offset was added to the sine wave so that a sine wave between 0 and 5 volts is created. The Arduino Yun reads the sine wave as a 10 bit number where the maximum voltage it can read is 5 volts, relating this to the largest 10 bit number that is possible gives 1023. This means if 5 volts is applied to the analog pin, the Yun will read 1023. In order to calculate the correct RMS value, the 2.5V offset must be subtracted from this number. The scale between voltage and its corresponding number in binary is linear so if 5 volts correlates to 1023, 2.5 volts correlates to half of 1023, or 512. The software then squares this number and creates a sum of all the samples until the designated number of samples required to calculate RMS voltage is obtained. The RMS value is calculated by taking InputSum[x] from Figure 27 and dividing it by the number of samples. As the number of samples increases, the accuracy of the reading increases as well.

```
for(int x = 0; x<7; x++){  
    InputAC[x] = analogRead(Input[x])-512;  
    InputSum[x] = InputSum[x] + sq((float)InputAC[x]);  
}
```

Figure 27: For Loop to Reduce Repetitive Code

The final step required for a complete design enabling a user to switch between displaying line-to-neutral and line-to-line voltage. In order to determine if the switch on the front of the box is in the line-to-line position or the line-to-neutral position, one side of the switch is connected to ground and the other is connected to a digital input. The Arduino Yun is set to treat this digital pin as an input and writing a HIGH to an input enables the internal pullup resistors on the microcontroller. The Arduino Yun then samples this pin and will read either a HIGH when the switch is open and LOW when the switch is closed. The switch was connected so that if the Arduino Yun reads a LOW on the corresponding digital pin it relates to the switch being in the line-to-line position. The final step was to calculate the line-to-line voltage. This was done by taking one line voltage and subtracting another from it which will create a line-to-line sine wave. The same method as above was then implemented to calculate the RMS value of the line-to-line voltage. The three line-to-line voltages are: voltage phase x to phase y, voltage phase y to phase z, and voltage phase z to phase x. This looks like:

$$\begin{aligned}
Voltage_{XY} &= Voltage_X - Voltage_Y \\
Voltage_{YZ} &= Voltage_Y - Voltage_Z \\
Voltage_{ZX} &= Voltage_Z - Voltage_X
\end{aligned}$$

The software that is new compared to what is currently available is the ability to change settings wirelessly, being able to log data for a longer period of time at a quicker sampling rate, transmitting the data wirelessly, as well as sending email notifications if a problem arises. Having an email alert system is especially important because it allows the operator to know within a couple minutes of when a voltage transient occurs. This will give the operator a reference point when looking at the data to find where the event occurred. If an email was not sent the operator might have to search through weeks or months of data to locate the one event. Assuming a sampling rate of 1 second, there would be 604,800 data points to sort through after just one week.

Chapter 4: Data Tests

Due to our inability test the voltage and current ratings desired by our sponsor, we designed and developed a prototype transformer unit to simulate the transformer unit which be any factory our device is installed. This device steps down 120 VAC, and can be seen in Figure 28.

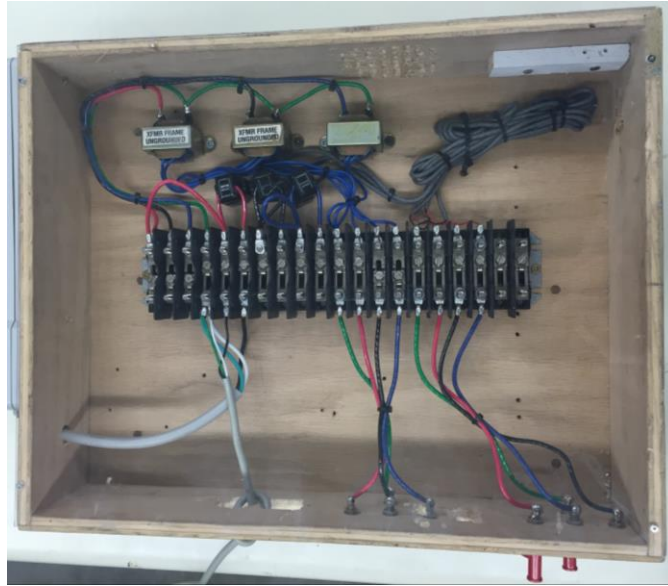


Figure 28. Prototype transformer unit

We employed the use of the Manta MTS-5100 Protective Relay Test System, as it allowed us to test each functionality of our unit, as well as test some features on a phase by phase basis. The entire setup for our testing can be seen in Figure 29.



Figure 29. Entire setup for testing our final prototype

The different functionalities we testes are as follows:

Power Factor Measurement

To test this, we used the Manta MTS-5100 to send different power factor values to our device, and recorded the values measured by our device. We also calculated the percentage error for our measured values and found the average percentage error to be 3.15%. The results and visual representation can be seen in Table 4 and Figure 30 respectively.

Expected	Measured	% Error	
0.5	0.47	6.00%	
0.6	0.57	5.00%	Average Error
0.7	0.67	4.29%	3.15%
0.8	0.78	2.50%	
0.9	0.89	1.11%	
1	1	0.00%	

Table 4. Results from power factor testing

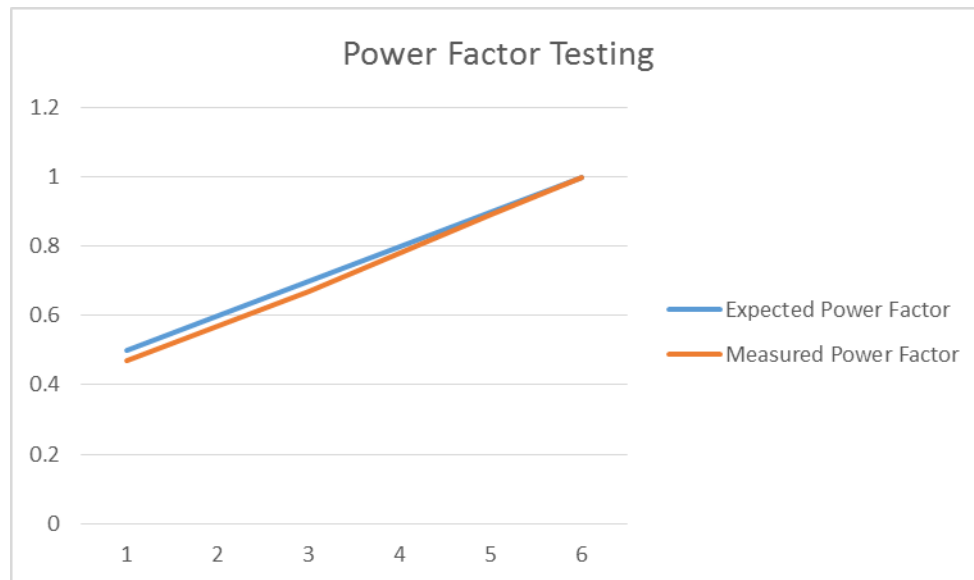


Figure 30. Visual representation of results from power factor testing

Voltage Measurement

To test this, we used the Manta MTS-5100 to send different voltage values to our device, and recorded the values measured by our device. We also calculated the percentage error for each phase of our measured values and found the average percentage error to be 4.26%, 2.69% and 3.24% for the X, Y and Z phases respectively. The results and visual representation can be seen in Table 5 and Figure 31 respectively.

Expected Voltage	X	Y	Z	X % Error	Y % Error	Z % Error
40	43.162	41.846	42.125	7.91%	4.61%	5.31%
50	49.472	51.806	52.133	1.06%	3.61%	4.27%
60	63.422	61.436	62.093	5.70%	2.39%	3.49%
70	73.712	71.726	72.383	5.30%	2.47%	3.40%
80	84.002	81.356	82.343	5.00%	1.69%	2.93%
90	94.302	91.646	92.303	4.78%	1.83%	2.56%
100	104.59	101.6	102.6	4.59%	1.60%	2.60%
110	114.55	111.56	112.56	4.14%	1.42%	2.33%
120	122.19	119.86	120.53	1.83%	0.12%	0.44%
130	128.5	126.5	127.17	1.15%	2.69%	2.18%
140	132.48	130.02	131.48	5.37%	7.13%	6.09%
Average % Error				4.26%	2.69%	3.24%

Table 5. Results from voltage measurement testing

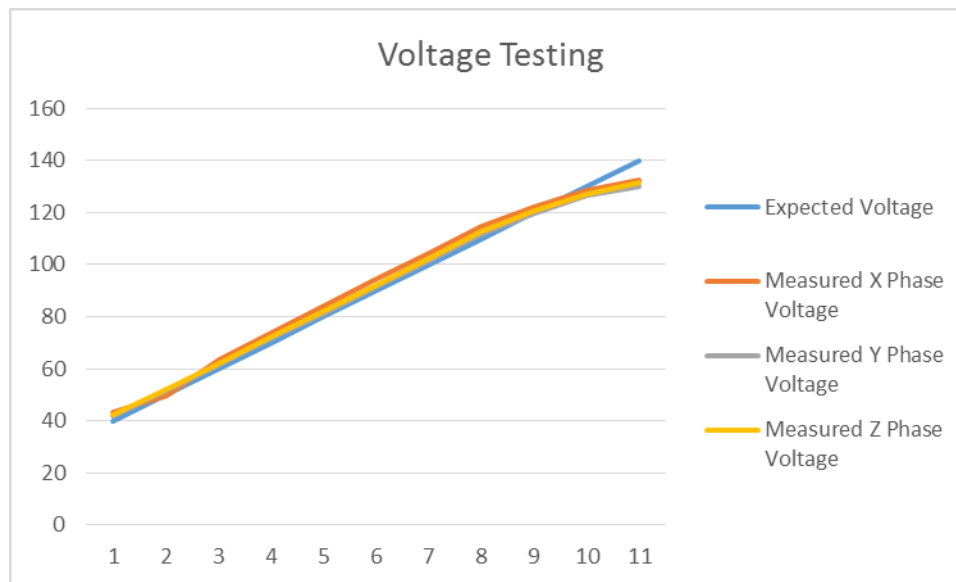


Figure 31. Visual representation of result from voltage measurement testing

Current Measurement

To test this, we used the Manta MTS-5100 to send different current values to our device, and recorded the values measured by our device. We also calculated the percentage error for each phase of our measured values and found the average percentage error to be 6.43%, 8.71% and 6.67% for the X, Y and Z phases respectively. We had a split core CT that accidentally came apart, and tried rebuilding as best as we could, however, this could have caused some errors on the Y phase currents. The results and visual representation can be seen in Table 6 and Figure 32 respectively.

Expected Current	X	Y	Z	X % Error	Y % Error	Z % Error
0.5	0.63	0.64	0.63	26.00%	28.00%	26.00%
1	1.04	1.02	1.05	4.00%	2.00%	5.00%
1.5	1.49	1.45	1.49	0.67%	3.33%	0.67%
2	1.96	1.89	1.96	2.00%	5.50%	2.00%
2.5	2.43	2.35	2.43	2.80%	6.00%	2.80%
3	2.91	2.79	2.9	3.00%	7.00%	3.33%
3.5	3.27	3.18	3.26	6.57%	9.14%	6.86%
Average % Error				6.43%	8.71%	6.67%

Table 6. Results from current measurement testing

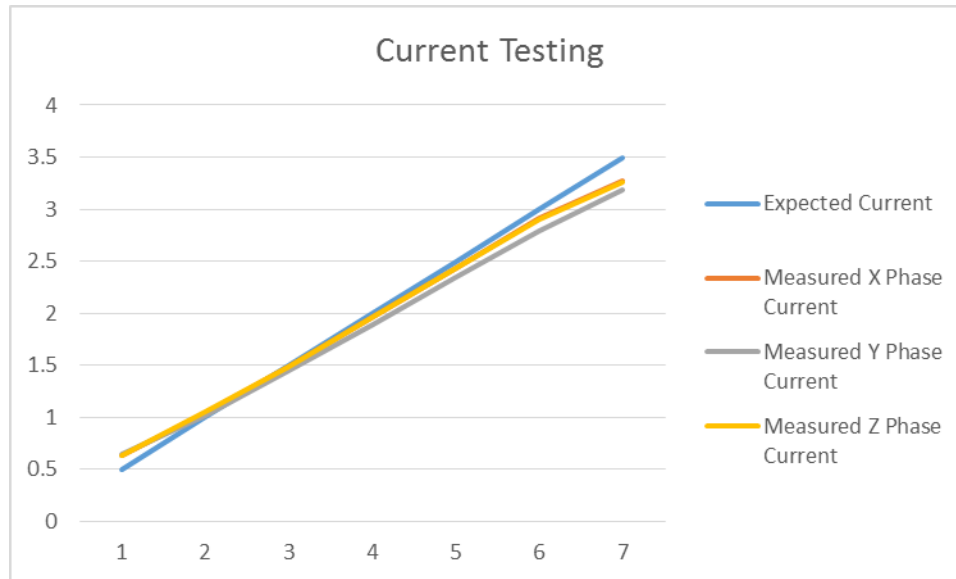


Figure 32. Visual representation of result from current measurement testing

Power Measurement

To test this, we used the Manta MTS-5100 to send different power values to our device, and recorded the values measured by our device. We also calculated the percentage error for our measured values and found the average percentage error to be 3.79%. The results and visual representation can be seen in Table 7 and Figure 33 respectively.

Expected	Measured	% Error	
150	154.87	3.25%	
300	294.59	1.80%	Average % Error
450	434.24	3.50%	
600	574	4.33%	
750	714.27	4.76%	
900	854.45	5.06%	

Table 7. Results from power measurement testing

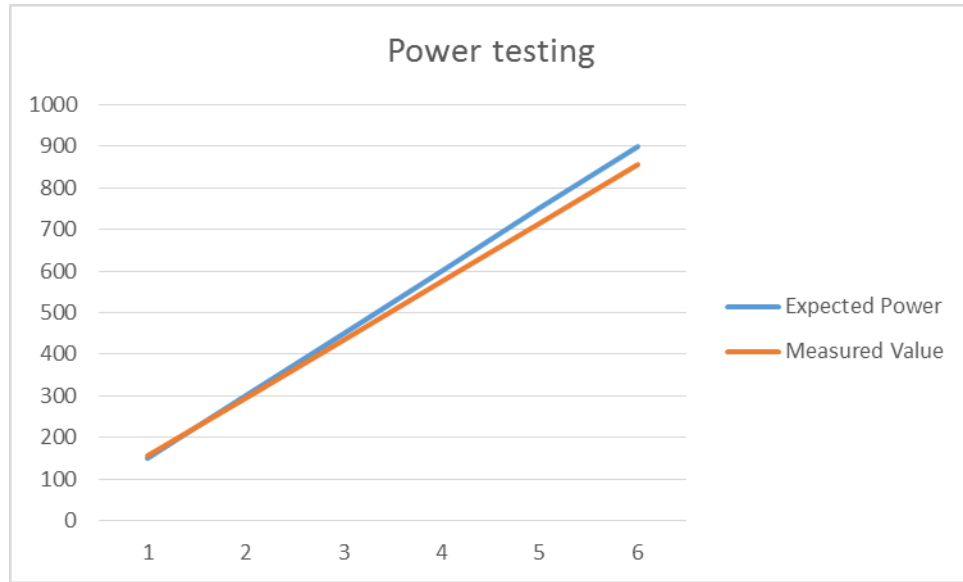


Figure 33. Visual representation of results from power measurement testing

Line-to-Neutral Voltage Measurement

To test this, we used the Manta MTS-5100 to send different voltage values to our device, we then multiplied the sent values by $\sqrt{3}$ to calculate the expected theoretical Line-to-Neutral voltage, and recorded the values measured by our device. We also calculated the percentage error for each phase of our measured values and found the average percentage error to be 1.06%, 0.98% and 5.79% for the X, Y and Z phases respectively. The results and visual representation can be seen in Table 8 and Figure 34 respectively.

Theoretical L-N	X	Y	Z	X % Error	Y % Error	Z % Error
173.2050808	174.32	172.66	184.28	0.64%	0.31%	6.39%
190.5255888	192.91	191.91	203.54	1.25%	0.73%	6.83%
207.8460969	205.2	203.87	216.48	1.27%	1.91%	4.15%
Average % Error				1.06%	0.98%	5.79%

Table 8. Results from line-to-neutral voltage measurement testing

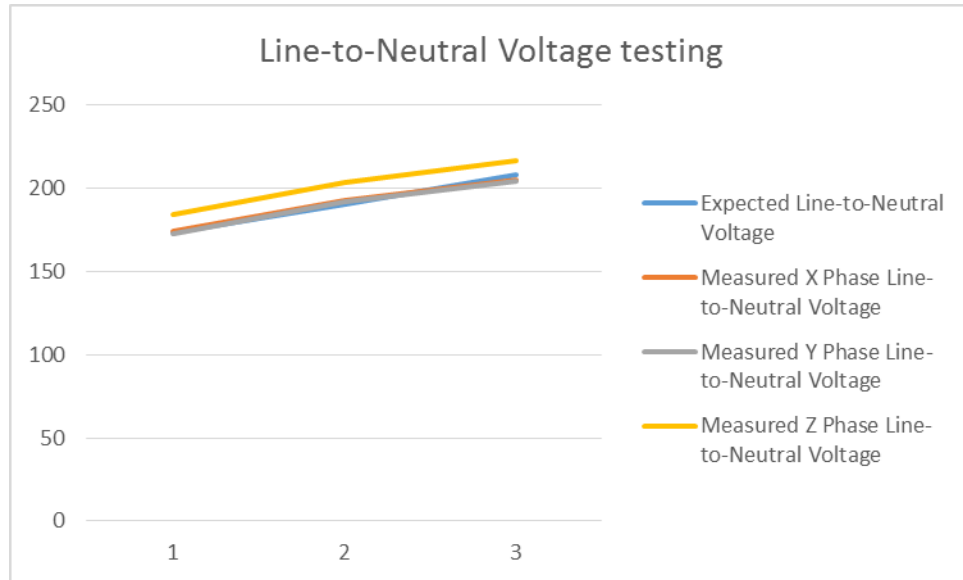


Figure 34. Visual representation of results from line-to-neutral voltage measurement testing

Line-to-Line Voltage Imbalance per phase Measurement

This was a vital feature to test, because with 3 phase systems, you typically have a single phase imbalanced, therefore it is beyond important to test our devices ability to detect voltage imbalance on a phase by phase basis. To test this, we used the Manta MTS-5100 to send different voltage values to our device, with an imbalance of one of the phases, as seen in Table 9, and recorded the values measured by our device, as soon in Table 10.

Expected		
X	Y	Z
120	110.9	120
106.6	120	120
120	120	109.5

Table 9. Different phase voltage values sent to the system

Measured		
x	Y	Z
122.52	112.56	120.53
111.23	119.86	120.53
122.52	119.86	112.23

Table 10. Resulting values from test

Chapter 5: Design Issues

Non-technical Design Issues

The first design issue arose after our first meeting with our facilitator, as our device was initially required to measure up to 600VAC and 100 Amps. However, those are really high voltage and current ratings and are not allowed within our design space laboratory. So we had to adjust the requirement, to have our device read up to 120VAC and 3 Amps. The second design issue came due to our lack of an accessible three phase function generator to test our design. In order to make up for this, we built our prototype transformer unit to not only step down voltages but also simulate a three phase system.

We built our device to meet industry standards in terms of colored wiring and arrangement within the unit, and ensured we created a product that could be directly implemented into a factory as desired. We also ensured our device was safe enough to be used by anyone and reduced the use of large voltages within the unit to a minimal.

With regards to future improvements that could be made to further improve the safety of our device, we would recommend the use of a shorting switch for CT circuit, thus preventing the possibility for open CT.

Technical Design Issues

We faced a minimal amount of technical design issues in the development of our project; mostly thanks to the amount of prior research we did before partaking in each step. The only design mishap occurred when a member of the team accidentally applied a half wave rectified ac voltage to the microcontroller, while trying to create a voltage clipper.

The only major issue we encountered came as a result of the limited flash memory of the Arduino Yun. Due to the, complexity of our codes and programming, we had to use and call a lot of libraries which took up most of the flash memory. One of the ways to save on flash memory was to implement the use of 'for' loops as mentioned in Chapter 3.2 of this report. In the beginning each calculation was done with its own line of code, to allow for better readability of the code and allow anyone look at the sketch and easily diagnose which line was affecting the phase x voltage. After switching to the arrays it is not as intuitive as all the inputs are labeled Input[x] where x is an integer from 1 to 6 which relates to a location in the array, a location of x=1 relates to phase x voltage, x=2 is phase y voltage, x=3 is phase z voltage, x=4 is phase x current, x=5 is phase y current, and x=6 is phase z current. The benefit however is that it cuts down on the amount of flash memory used which ended up being more vital than how easily the code could be read.

Chapter 6: Power Monitoring System

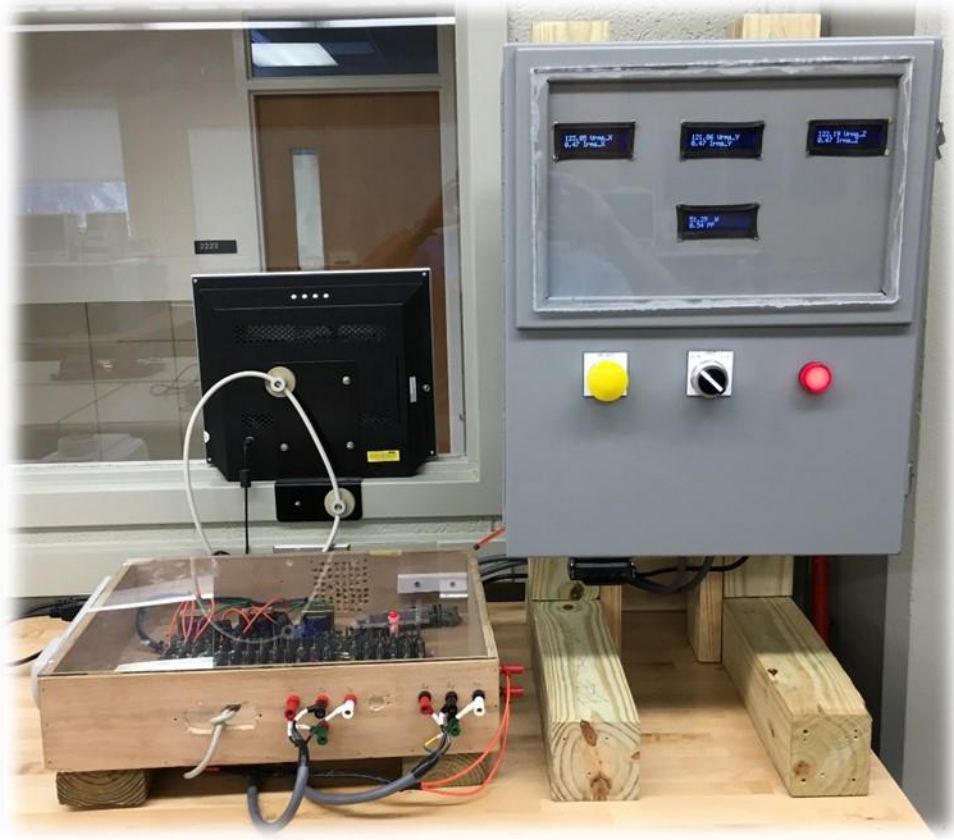


Figure 35. Final unit design (right) and prototype transformer unit (left)

6.1 – Final Summary

With only a few iterations made, the development of our project went almost exactly according to plan. The flow diagram for our completed project can be seen in Figure 36. In comparison to the initial flow diagram seen in Chapter 2.2 of this report, there are not many notable changes. More details of the comparison can be seen in later sections of this chapter.

In the completion of our project, we faced some notable obstacles. Our project proved to be both hardware and software intensive, which posed the first obstacle as we lacked sufficient programming background and did not anticipate how software intensive it would be, as our team consisted solely of Electrical Engineering majors and no Computer Science/Engineering majors. For a more comprehensive review of the design issues the team faced, like the lack of flash memory within the microcontroller and our inability to test higher voltages, please refer to Chapter 5 of this report.

Overall, we were successful in achieving our goal in developing a three-phase four-wire AC power monitoring system capable of data logging voltage phase-phase, voltage phase-ground, current, and power. Our system is also capable of alerting users at the occurrence of transient events, and possesses options to enable a user transfer data or change any desired settings. We also designed a prototype transformer unit to simulate the transformer unit to be used in the factory. The final design for our unit and the prototype transformer unit can be seen in Figure 35.

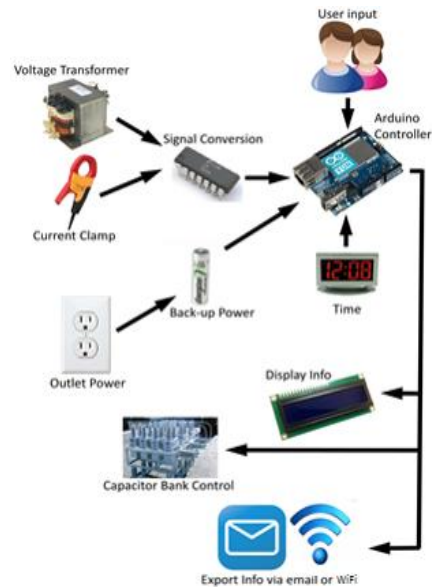


Figure 36. Flow diagram of project outcome

6.2 – Bonus Features

In the development of our product, we equipped our product with some additional features which were not specified as part of our product requirements. More details of some of these features can be seen within this report or in the Operator Manual we created for our product. Some of the notable bonus features we implemented include:

- Capacitor bank control: We designed our product with capacitor bank control, which would open or close depending on the set power factor limits.
- Backup battery and kill circuit: In the event of a loss of power to the device, we designed a backup battery and kill circuits, which would kick in whenever power is lost, powering the microcontroller long enough to send an email to alert users of the loss of power, before “killing” itself.
- Wireless data and settings transfer: We were requested to develop a unit which would possess a USB port for transferring the data logged by the unit. With our

- developed unit, we've eliminated that need. Push buttons were also desired to change certain settings like the email address for alert notifications or the tolerance levels. Our microcontroller currently emits its personal WiFi network, which you could connect to with a laptop and wirelessly download the logged data from the device or change the desired settings.
- Internet connectivity: We were requested to develop a unit which could connect to the internet via Ethernet connection. Our developed product can connect to the internet either via Ethernet or WiFi connection. Connecting via WiFi enables a user with a computer connected to the same network, transfer data from or change settings of the device anywhere within the factory.
 - Extra data-logging memory: Although our device was required to log data for a week, our device can log data for a month.

6.3 – Final Cost

The overall expenses incurred in completing our project totaled up to \$702.88. The breakdown of total expenses can be seen in Table 11. The breakdown consists of all part acquired by our team, and parts acquired by our sponsor.

Components	Cost	Quantity	Total
*Arduino YUN Micro-controller	\$69.27	3	\$207.81
Basic 16x2 Character LCD – White on Black 5V	\$9.95	4	\$39.80
I2C Serial Interface Board Module LCD	\$2.09	4	\$8.36
16GB MicroSD Card	\$12.00	1	\$12.00
24V KPR Relay	\$7.55	1	\$7.55
RJ45 Port	\$47.5	1	\$47.5
12V Power Supply	\$24.00	1	\$24.00
5V Power Supply	\$24.00	1	\$24.00
24V Alarm Light	\$7.00	1	\$7.00
Reset Button	\$6.45	1	\$6.45
2 Position Switch	\$5.96	1	\$5.96
Terminal Block	\$1.50	26	\$39.00
Fuse Block	\$2.99	4	\$11.96
Enclosure	\$244.00	1	\$244.00
PCB Board	\$0	1	\$0
LM324	\$0.44	2	\$0.88
LM317	\$0.35	1	\$0.35

LM3940	\$0.55	1	\$0.55
LM2940	\$0.60	1	\$0.60
DS2E-S-DC 12V Relay	\$2.45	3	\$7.35
DS2E-S-DC 5V Relay	\$3.59	1	\$3.59
TIP31A	\$0.39	3	\$1.17
Misc Caps, Resistors, LEDs	-	-	\$3.00
Total:			\$702.88

Table 11. Breakdown of total expenses incurred

However, the total amount consisted of additional expenses that would not re-occur if the device were to be reproduced. For example, as part of our expenses, we bought 3 microcontrollers which totaled \$207.81, whereas we would typically need a single microcontroller which would cost \$69.27. We originally bought 2 microcontrollers, with one intended for testing and the other for the final prototype, however, we damaged a microcontroller during testing and therefore had to acquire a third one. After taking the extra microcontroller costs into account, the total cost to reproduce the product will amount to \$564.34.

In comparison with the initial cost listed in Chapter 2 of this report, we changed 64GB micro SD card into 16GB micro SD card which saves some money, because we did not need that much memory to save data. Besides that, and the extra microcontroller costs, the only other difference would be the extra components has been added to the initial cost. The total costs to reproduce the device can be seen in Table 12.

Components	Cost	Quantity	Total
Arduino YUN Micro-controller	\$69.27	1	\$69.27
Basic 16x2 Character LCD – White on Black 5V	\$9.95	4	\$39.80
I2C Serial Interface Board Module LCD	\$2.09	4	\$8.36
16GB MicroSD Card	\$12.00	1	\$12.00
24V KPR Relay	\$7.55	1	\$7.55
RJ45 Port	\$47.5	1	\$47.5
12V Power Supply	\$24.00	1	\$24.00
5V Power Supply	\$24.00	1	\$24.00
24V Alarm Light	\$7.00	1	\$7.00
Reset Button	\$6.45	1	\$6.45
2 Position Switch	\$5.96	1	\$5.96
Terminal Block	\$1.50	26	\$39.00
Fuse Block	\$2.99	4	\$11.96
Enclosure	\$244.00	1	\$244.00
PCB Board	\$0	1	\$0

LM324	\$0.44	2	\$0.88
LM317	\$0.35	1	\$0.35
LM3940	\$0.55	1	\$0.55
LM2940	\$0.60	1	\$0.60
DS2E-S-DC 12V Relay	\$2.45	3	\$7.35
DS2E-S-DC 5V Relay	\$3.59	1	\$3.59
TIP31A	\$0.39	3	\$1.17
Misc Caps, Resistors, LEDs	-	-	\$3.00
Total:			\$564.34

Table 12. Breakdown of cost to reproduce our device

To fully assess the total cost of our project, we selected the SEL-735 Outdoor Meter to use as a comparable case study. The SEL-735 meter shall be 32-bit microprocessor-based with a combination of recording, automation, and communications capabilities. The meter shall include self-diagnostic functions to alarm upon detected failure. Major specifications include:

- Accuracy.
- Power Quality.
- Load Profile Recording.
- Transformer/Line-Loss Compensation.
- Human-Machine Interface.
- Instrument Transformer Compensation.
- Sequential Events Recorder (SER).
- Voltage Sag/Swell/Interruption Recording (VSSI).
- Waveform Capture.
- Harmonic Metering.

In comparison, our project possesses most of functions of SEL-735, except the harmonic measuring feature. However, the cost of our project amounts to less than one third of the price of SEL-735, as can be seen in Table 13.

Cost of Reproducing our Project	\$564.34
Price of SEL-735	\$2095.00

Table 13. Comparison of our product against the SEL-735

6.4 – Future Designs

In the development of our project, we took notes of future implementations and improvements that could be made to future improve the functionality and efficiency of our product. Some possible improvements that could be made include, but are not limited to:

- Improving the modularity of the product. Our overall unit box was designed according to the specifications provided by our sponsor. However, redesigning the product with a smaller and more portable enclosure would be something to look towards.
- An additional feature we thought about implementing but never got the opportunity to, was to enable our device calculate and measure harmonics. This can be achieved through the use/implementation of FFT's within our programming.
- Adding a switching regulator to eliminate the need for the second (5V) power supply. Switching regulators are also more efficient than linear voltage regulators.
- Averaging the DC offset to attain more accurate measurements.
- Enabling the device with the ability to pull weather information from the internet to include in the email notification sent out. This was the only deliverable we were not able to achieve.

Appendix 1 – Personnel

In order to successfully meet the sponsor's requirements and in due time, the team divided the project tasks between its members. The duties for each team member are as follows:

McCormick, James - General Management

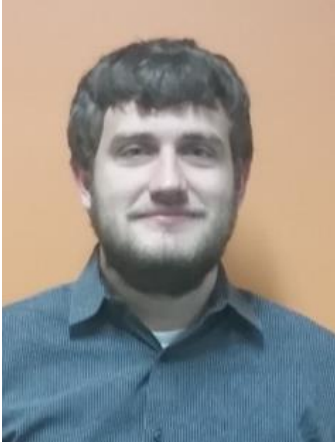


James, the General Manager of the team, was in charge of the overall coordination and supervision of the project. He played a part in the supervision of every aspect of the project and work with all other coordinators in making sure all tasks are completed promptly and efficiently. He developed the team's timeline schedule in the form of a GANTT chart, and was also responsible for maintaining regular communications with our facilitator and sponsor.

Being the only member with prior experience with power, and having the most experience with analog circuits, he was in charge of researching, developing and testing all the necessary circuits needed for proper functioning of the device. He first designed the concept for a power supply to power a protoboard with a 120V supply. He worked with Victor in developing the circuit for signal conversion for sampling a sinewave with the microcontroller for our project. He came up with the idea and developed the circuit to switch the device to backup battery power and send a signal to the microcontroller when external power was lost, before then "killing" itself. He also designed the PCB, and developed the capacitor bank circuit, as well as the test box used for prototype 2. He then worked with Alex and Victor in brainstorming the code logic behind integrating the capacitor bank circuit within the programming.

Finally, he worked with Victor in developing the final box for our product, where we went to the sponsor's workstation and sawed and drilled holes into the metal box for the cables and LCD screens to go through, then screwed and set up the screens to the box.

Lange, Alex - Lab Coordinator



Alex was primarily in charge of the supervision of all laboratory equipment, experiments and procedures. He was tasked with placing orders for any needed parts, as well as maintenance of all the equipment and the team's assigned locker. He researched and identified most of the necessary parts needed for our project, which include but are not limited to our selected microcontroller, LCD screen and I²C modules to use.

Being the only member of the team with prior experience with the team's chosen microcontroller, Arduino, he was in charge of the supervision, development and testing of all programming and software related tasks, as well as, ensuring the software side of our project met the deliverables as requested by our sponsor. Amongst other things, he developed/implemented the coding/programming behind the:

- I²C communication for writing to the LCDs
- Calculations for RMS current, RMS voltage line-to-neutral, RMS voltage line-to-line, total power, power factor, voltage imbalance
- Switch for displaying line-to-line or line-to-neutral voltage
- Triggering an alarm light at the occurrence of a transient event
- Capacitor bank control and change of state delay
- Auto shutoff after email notification has been sent when on backup battery

Finally, he was also in charge of the overall compilation of the programming after each member of the team had developed and tested their assigned sections of programming tasks.

Ezenwoko, Victor Tobenna - Documentation Manager



Victor was primarily in charge of coordinating all forms of documentation regarding the team. His main tasks included taking notes during each team meeting, to the preparation, review and supervision of all needed documents for submission; where he prepared the skeleton and majority of the documentation, and assign minimal parts to the rest of the team members. Afterwards, he reviewed all documentation ensuring they meet all requirements and the team's quality standards. He also worked closely with the presentation manager in the preparation of written, visual and oral presentations.

Apart from his documentation responsibilities, he worked in developing the circuit for signal conversion for sampling a sinewave with the microcontroller for our project; and developed the final operator manual for the device. He was also in charge of acquiring and developing the codes/programming to enable our microcontroller:

- Connect to the internet via Ethernet. Which also included working with DECS to make sure the device could connect to the MSU Engineering network for design day.
- Download real time stamp data from the internet
- Send out email notifications, at the occurrence of a transient event.

He also worked with James and Alex in brainstorming the code logic we used in programming the capacitor bank circuit; and with Jacob in developing the code that would enable the microcontroller download data from and into a text file.

Finally, he worked with James in developing the final box for our product, where we went to the sponsors workstation and sawed and drilled holes into the metal box for the cables and LCD screens to go through, then screwed and set up the screens to the box. And was heavily involved in testing the final programming of the microcontroller.

.

Jebb, Jacob - Presentation Manager



Jacob was primarily in charge of the coordination of all forms of presentations regarding the project. This includes, but is not limited to, the preparation, review and supervision of all needed visual and oral presentations; where he prepared the skeleton and majority of the documentation, and assign minimal parts to the rest of the team members. He was also in charge of capturing images and videos of all concepts, ideas, progress and prototypes.

He also assisted the documentation manager in the preparation of all necessary documentation.

Apart from his presentation responsibilities, he worked actively in the coding/programming of our project. He was in charge of acquiring and developing the codes/programming to enable our microcontroller:

- Connect to the internet via WiFi.
- Data logging into the microcontrollers MicroSD
- Exporting data wirelessly
- Wireless transfer and update of the microcontroller settings

At the early stages of our project, due to our inability to connect to the engineering network, he was solely in charge of testing the internet connectivity and email functionality of our project, as he had a free network at his home to test with. He also worked with Victor in developing the operator manual for our device. Finally, he created all the digital representation of the PCB layout, as well as all circuits, schematics, and designs.

Qian, Zhihoug – Webmaster



Zhihoug is primarily in charge of coordinating the team's online image and platform. This not only includes the creation, design and maintenance of the website, but also involves ensuring all the team's documents and media are available on the website, and creating a respectable brand for the team. As the webmaster, he spent a lot of his time developing the group's website. He learned to use HTML and CSS language, and to create website with Adobe Dreamweaver. He did his best to create every single page of website, including inserting pictures, filling content, designing picture slider and dropdown menu, to make our website visual pleasing; and ensure to keep the website

updated on a daily basis.

His primary technical responsibilities involved testing the different segments/sections of code within our entire program, and brainstorming, researching and coming up with future ideas and implementations for our device. He began programming our microcontroller to calculate and measure harmonics with the use of FFT within our coding. Although he was unable to fully implement it, he made great strides. He also worked with James in reviewing the PCB layout. Finally, he was in charge of designing the design day showcase poster.

Appendix 2 – Literature Review

- [1] Voltage Imbalance in 3-Phase Systems, York Central Tech Talk.
<https://yorkcentraltechtalk.wordpress.com/2012/07/03/voltage-imbalance-in-3-phase-systems/> 07/03/2012.
- [2] Tsai-Hsiang Chen, Chwnng-Han Yang, and Ting-Yen Hsieh, Case Studies of the Impact of Voltage Imbalance on Power Distribution Systems and Equipment, National Taiwan University of Science and Technology, Department of Electrical Engineering.
<http://www.wseas.us/e-library/conferences/2009/hangzhou/ACACOS/ACACOS78.pdf>
- [3] Unbalanced Voltages and Electric Motors: Causes and Consequences
<http://www.elongo.com/pdfs/voltages.PDF>
- [4] A. von Jouanne and B. B. Banerjee, Assessment of voltage unbalance, IEEE Transactions on Power Delivery, Volume: 16, Issue: 4, page 782-790, 2001.
- [5] 1200A 3-Phase Power Analyzer/Datalogger, Extech Instruments Product Datasheet, 2009. Authorship by Extech Instruments Corporation.
- [6] ECE 402 Applications of Analog Integrated Circuits Course e-Notes, 2015. Authorship by Gregory M. Wierzba.
- [7] ECE 302 Applications of Analog Integrated Circuits Course e-Notes, 2015. Authorship by Gregory M. Wierzba.
- [8] Woll, R.F., "Effect of Unbalanced Voltage on the Operation of Polyphase Induction Motors," in *Industry Applications, IEEE Transactions on* , vol.IA-11, no.1, pp.38-42, Jan. 1975
- [9] de Abreu, J.P.G.; Emanuel, A., "Induction motor thermal aging caused by voltage distortion and imbalance: loss of useful life and its estimated cost," in *Industrial and Commercial Power Systems Technical Conference, 2001. Conference Record. Papers Presented at the 2001 Annual Meeting. 2001 IEEE* , vol., no., pp.105-114, May 2001.
- [10] Faiz, J.; Ebrahimpour, H.; Pillay, P., "Influence of unbalanced voltage on the steady-state performance of a three-phase squirrel-cage induction motor," in *Energy Conversion, IEEE Transactions on* , vol.19, no.4, pp.657-662, Dec. 2004

Appendix 3

Software/Programming

```
#include <Wire.h>
#include <FileIO.h>
#include <Process.h>
// #include <Bridge.h>
// #include <Temboo.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
// #include "TembooAccount.h"

LiquidCrystal_I2C lcd1(0x25, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); //Defining the I2C pins
LiquidCrystal_I2C lcd2(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); //Defining the I2C pins
LiquidCrystal_I2C lcd3(0x23, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); //Defining the I2C pins
LiquidCrystal_I2C lcd4(0x26, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); //Defining the I2C pins

unsigned long int taskTime[10]; //array used for creating time based events
unsigned long int taskPeriod[10]; //array used for creating time based events
int Input[8]; //array for reading input
int InputAC[8]; //array of input after 2.5V shift
int VoltageLL[5]; //line to line
unsigned long int VoltageLLSum[5]; //sum of l-l squared
int VoltageLLRMS[5]; //voltage rms in binary
float VoltageLLReal[5]; //Voltage RMS converted to decimal
unsigned int LowVoltageLL[5]; //Low voltage states
unsigned long int InputSum[8]; //Sum of all squared values for calc RMS
unsigned long int PowerSum[8]; // 1=X, 2=Y, 3=Z
int InputRMS[8]; //RMS values of the inputs
float PowerRealAverage[8]; //real power
unsigned long int PowerApparent[5]; //apparent power
float InputReal[8]; //RMS values of the inputs in decimal
int LowVoltage[5]; //low voltage state
int LowCurrent[8]; //low current state
float PowerFactor, PowerFactor_X, PowerFactor_Y, PowerFactor_Z; //power factor
float VoltageReal_Avg; //VoltageRMS converted to real voltage
float PowerReal, PowerReal_X, PowerReal_Y, PowerReal_Z; //power of each line
int ArrayLocation = 0; //counting number of samples
int CapBankPin = 11; //pin for capacitor bank control
int CapBankDelay = 10000; //10 seconds
int CapBankTimer = 0; //used as counter
int CapBankState = 0; //state of cap bank
int AlarmLight = 10; //digital pin for alarm light
```

```

/**** Changeable Settings ****/
int NumSamples = 1000; //Number of samples to calculate RMS
int VoltageCutoff = 128; //definition of low voltage (in binary)
int CurrentCutoff = 61; //definition of low current
int VoltageScale = 340; //voltage peak value
int CurrentScale = 50; //peak current value
String TO_EMAIL_ADDRESS; //where to send an alert email
char character; //used for converting
/****End Changable Settings****/
int Switch_LL_LN = 7; //reading pin for state of switch
int Switch_LL_LN_State; //state of pin
int PowerSense = 12; //sensing if battery power or not
int GoodPowerStatus = 1; //good power vs battery
int KillPower = 9; //pin for turning itself off
float ImbalanceTolerance = .05; //percent of acceptable imbalance
float ImbalanceOver;
float ImbalanceUnder;
const String GMAIL_USER_NAME = "MSUPowerMonitor@gmail.com";
const String GMAIL_APP_PASSWORD = "blhtlbizhfhdtdyb";

void setup()
{
  Bridge.begin(); //initializing bridge
  FileSystem.begin(); //initializing file transfer
  lcd1.begin(16,2); //initializing lcds.
  lcd2.begin(16,2);
  lcd3.begin(16,2);
  lcd4.begin(16,2);
  Input[1] = A0; //Vx
  Input[2] = A1; //Vy
  Input[3] = A2; //Vz
  Input[4] = A4; //Ix
  Input[5] = A5; //Iy
  Input[6] = A3; //Iz
  taskPeriod[0] = NumSamples; //how often it prints VoltageRMS to LCD
  taskPeriod[1] = 1; //read a new value of array Sin5V every 1ms
  taskPeriod[2] = 1000; //gets new time every second
  taskPeriod[3] = NumSamples;
  taskPeriod[4] = 50; //sample buttons
  taskPeriod[5] = 1000;
  taskPeriod[6] = 15000;
  pinMode(Input[1], INPUT); //setting analog pins in inputs for reading data
  pinMode(Input[2], INPUT);
  pinMode(Input[3], INPUT);
  pinMode(Input[4], INPUT);

```

```

pinMode(Input[5], INPUT);
pinMode(Input[6], INPUT);
pinMode(CapBankPin, OUTPUT); //allowing control of cap bank
digitalWrite(CapBankPin, LOW);
pinMode(AlarmLight, OUTPUT);
pinMode(PowerSense, INPUT);
pinMode(KillPower, OUTPUT);
pinMode(Switch_LL_LN, INPUT);
digitalWrite(Switch_LL_LN, HIGH); //enable pullup resistor
}

void loop(){
  unsigned long int loopTime = millis();
  String dataString; //Used to format data for MicroSD

  if(loopTime - taskTime[4] > taskPeriod[4]){
    taskTime[4] = loopTime-1;
    Switch_LL_LN_State = digitalRead(Switch_LL_LN); //reads state of L-L/L-N switch
    GoodPowerStatus = digitalRead(PowerSense); //read state of power
    if(GoodPowerStatus == LOW){
      //NOT ENOUGH MEMORY TO USE
      //  TembooChoreo SendEmailChoreo;
      //  SendEmailChoreo.begin();
      //  SendEmailChoreo.setAccountName(TEMBOO_ACCOUNT);
      //  SendEmailChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
      //  SendEmailChoreo.setAppKey(TEMBOO_APP_KEY);
      //  SendEmailChoreo.setChoreo("/Library/Google/Gmail/SendEmail");
      //  SendEmailChoreo.addInput("Username", GMAIL_USER_NAME);
      //  SendEmailChoreo.addInput("Password", GMAIL_APP_PASSWORD);
      //  SendEmailChoreo.addInput("ToAddress", TO_EMAIL_ADDRESS);
      //  SendEmailChoreo.addInput("Subject", "ALERT");
      //  SendEmailChoreo.addInput("MessageBody", "Loss of Power");
      //  unsigned int returnCode = SendEmailChoreo.run();
      digitalWrite(KillPower, HIGH); //Turn off
    }
  }

  if(loopTime - taskTime[2] > taskPeriod[2]){
    taskTime[2] = loopTime-1;
    //NOT ENOUGH MEMORY TO USE
    //  date.begin("/bin/date");
    //  date.addParameter("/%m+%d/%Y %T");
    //  date.run();
    //  if(date.available()>0){
    //    String timeString = date.readString();

```

```

// }
}

// *****
// *****Read Voltage*****
// *****
if(loopTime - taskTime[1] > taskPeriod[1]){
    taskTime[1] = loopTime-1;
    for(int x = 0; x<7; x++){
        InputAC[x] = analogRead(Input[x])-512; //read pin and subtract 2.5V
        InputSum[x] = InputSum[x] + sq((float)InputAC[x]); //summation of inputs
squared
        InputAC[5] = InputAC[5]*1.025; //Used to scale hardware
    }
    VoltageLL[1] = InputAC[1]-InputAC[2]; //calculating line-to-line
    VoltageLL[2] = InputAC[2]-InputAC[3];
    VoltageLL[3] = InputAC[3]-InputAC[1];
    for(int x = 0; x<4; x++){
        VoltageLLSum[x] = VoltageLLSum[x] + sq((float)VoltageLL[x]); //summation of
l-l
    }
    for(int x = 0; x<5; x++){
        PowerSum[x] = PowerSum[x] + (float)InputAC[x]*(float)InputAC[x+3]; //calc
real power
    }
    ArrayLocation = ArrayLocation + 1; //index array

    if(ArrayLocation == NumSamples){
        for(int x = 0; x<8; x++){
            InputRMS[x] = sqrt(InputSum[x]/NumSamples); //calc RMS in binary
        }
        for(int x = 0; x<4; x++){
            PowerRealAverage[x] = PowerSum[x]/NumSamples; //calc real power
            PowerApparent[x] = (float)InputRMS[x]*(float)InputRMS[x+3]; //calc apparent
power
            VoltageLLRMS[x] = sqrt(VoltageLLSum[x]/NumSamples); //calc L-L in binary
        }
        PowerFactor_X =
cos(acos((float)PowerRealAverage[1]/(float)PowerApparent[1])-.209); //calc and
calibrating power factor
        PowerFactor_Y =
cos(acos((float)PowerRealAverage[2]/(float)PowerApparent[2])-.3665); //calc and
calibrating power factor
        PowerFactor_Z =
cos(acos((float)PowerRealAverage[3]/(float)PowerApparent[3])-.2269); //calc and
calibrating power factor

```

```

    for(int x = 0; x<4; x++){
        InputReal[x] = VoltageScale*((float)InputRMS[x]/1024); //calc L-N in decimal
        VoltageLLReal[x] = VoltageScale*((float)VoltageLLRMS[x]/1024); //calc L-L in
decimal
    }
    for(int x = 4; x<7; x++){
        InputReal[x] = CurrentScale*((float)InputRMS[x]/1024); //calc current in
decimal
    }
    PowerReal_X =
VoltageScale*CurrentScale*((float)PowerRealAverage[1]/1048576); //calc real
power
    PowerReal_Y =
VoltageScale*CurrentScale*((float)PowerRealAverage[2]/1048576); //calc real
power
    PowerReal_Z =
VoltageScale*CurrentScale*((float)PowerRealAverage[3]/1048576); //calc real
power

//  dataString += String(timeString);
//  dataString += " = ";

//***** Checking for low voltage or current state*****//
    for(int x = 0; x<4; x++){
        if(InputRMS[x] < VoltageCutoff){
            LowVoltage[x] = 1;
            dataString += "0, ";
        }
        else{
            LowVoltage[x] = 0;
            dataString += String(InputReal[x]);
            dataString += ",";
        }
    }
    for(int x = 0; x<4; x++){
        if(VoltageLLRMS[x] < VoltageCutoff){
            LowVoltageLL[x] = 1;
        }
        else{
            LowVoltageLL[x] = 0;
        }
    }
}

    for(int x = 4; x<7; x++){
        if(InputRMS[x] < CurrentCutoff){
            LowCurrent[x] = 1;

```

```

    dataString += "0, ";
}
else{
    LowCurrent[x] = 0;
    dataString += String(InputReal[x]);
    dataString += ",";
}
}

if(LowVoltage[1] == 1 || LowVoltage[2] == 1 || LowVoltage[3] == 1 ||
LowCurrent[4] == 1 || LowCurrent[5] == 1 || LowCurrent[6] == 1){
    digitalWrite(AlarmLight, HIGH); //low of voltage
}
else{
    digitalWrite(AlarmLight, LOW);
}
//***** ADJUST POWER FACTOR AND POWER BASED ON OUTAGE*****//
if(LowVoltage[1] == 1 || LowCurrent[4] == 1){
    PowerReal_X = 0;
    PowerFactor_X = 0;
}
if(LowVoltage[2] == 1 || LowCurrent[5] == 1){
    PowerReal_Y = 0;
    PowerFactor_Y = 0;
}
if(LowVoltage[3] == 1 || LowCurrent[6] == 1){
    PowerReal_Z = 0;
    PowerFactor_Z = 0;
}

PowerFactor = (PowerFactor_X + PowerFactor_Y + PowerFactor_Z)/3; //calc
power factor
PowerReal = PowerReal_X + PowerReal_Y + PowerReal_Z; //calc real power
dataString += String(PowerReal); //add to data string
dataString += ",";
dataString += String(PowerFactor); //add to data string
File dataFile = FileSystem.open("/mnt/sd/datalog.txt", FILE_APPEND);
dataFile.println(dataString);
dataFile.close();
ArrayLocation = 0;
for(int x = 0; x<8; x++){ //avoid overflow
    InputSum[x] = 0;
}
for(int x = 0; x<8; x++){ //avoid overflow
    PowerSum[x] = 0;
}

```

```

    for(int x = 0; x<4; x++){ //avoid overflow
        VoltageLLSum[x] = 0;
    }

    // *****
    // **Voltage Imbalance and Alarm**
    // *****
    VoltageReal_Avg = (InputReal[1]+InputReal[2]+InputReal[3])/3;
    ImbalanceOver = (1+ImbalanceTolerance)*VoltageReal_Avg;
    ImbalanceUnder = (1-ImbalanceTolerance)*VoltageReal_Avg;
    if(InputReal[1] < ImbalanceUnder || InputReal[1] > ImbalanceOver ||
InputReal[2] < ImbalanceUnder || InputReal[2] > ImbalanceOver || InputReal[3] <
ImbalanceUnder || InputReal[3] > ImbalanceOver ){
        digitalWrite(AlarmLight, HIGH);
        //NOT ENOUGH MEMORY TO USE
    //    TembooChoreo SendEmailChoreo;
    //    SendEmailChoreo.begin();
    //    SendEmailChoreo.setAccountName(TEMBOO_ACCOUNT);
    //    SendEmailChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
    //    SendEmailChoreo.setAppKey(TEMBOO_APP_KEY);
    //    SendEmailChoreo.setChoreo("/Library/Google/Gmail/SendEmail");
    //    SendEmailChoreo.addInput("Username", GMAIL_USER_NAME);
    //    SendEmailChoreo.addInput("Password", GMAIL_APP_PASSWORD);
    //    SendEmailChoreo.addInput("ToAddress", TO_EMAIL_ADDRESS);
    //    SendEmailChoreo.addInput("Subject", "ALERT");
    //    SendEmailChoreo.addInput("MessageBody", "Voltage Imbalance");
    //    SendEmailChoreo.run();
    }
    else{
        digitalWrite(AlarmLight, LOW);
    }
}
}

// *****
// *****Print VoltageRMS to LCDs*****
// *****
if(loopTime - taskTime[0] > taskPeriod[0]){
    taskTime[0] = loopTime-1;
    if(Switch_LL_LN_State == HIGH){
        if(LowVoltage[1] == 1){
            lcd1.setCursor(0,0);
            lcd1.print("Low Voltage! ");
        }
        else{
            lcd1.setCursor(0,0);

```

```

    lcd1.print(InputReal[1]);
    lcd1.setCursor(6,0);
    lcd1.print(" Vrms_X ");
}
if(LowVoltage[2] == 1){
    lcd2.setCursor(0,0);
    lcd2.print("Low Voltage! ");
}
else{
    lcd2.setCursor(0,0);
    lcd2.print(InputReal[2]);
    lcd2.setCursor(6,0);
    lcd2.print(" Vrms_Y ");
}
if(LowVoltage[3] == 1){
    lcd3.setCursor(0,0);
    lcd3.print("Low Voltage! ");
}
else{
    lcd3.setCursor(0,0);
    lcd3.print(InputReal[3]);
    lcd3.setCursor(6,0);
    lcd3.print(" Vrms_Z ");
}
}

else{
    if(LowVoltageLL[1] == 1){
        lcd1.setCursor(0,0);
        lcd1.print("Low Voltage! ");
    }
    else{
        lcd1.setCursor(0,0);
        lcd1.print(VoltageLLReal[1]);
        lcd1.setCursor(6,0);
        lcd1.print(" Vrms_XY ");
    }
    if(LowVoltageLL[2] == 1){
        lcd2.setCursor(0,0);
        lcd2.print("Low Voltage! ");
    }
    else{
        lcd2.setCursor(0,0);
        lcd2.print(VoltageLLReal[2]);
        lcd2.setCursor(6,0);
        lcd2.print(" Vrms_YZ ");
    }
}

```

```

}
if(LowVoltageLL[3] == 1){
    lcd3.setCursor(0,0);
    lcd3.print("Low Voltage! ");
}
else{
    lcd3.setCursor(0,0);
    lcd3.print(VoltageLLReal[3]);
    lcd3.setCursor(6,0);
    lcd3.print(" Vrms_ZX ");
}
}

if(LowCurrent[4] == 1){
    lcd1.setCursor(0,1);
    lcd1.print("Low Current! ");
}
else{
    lcd1.setCursor(0,1);
    lcd1.print(InputReal[4]);
    lcd1.setCursor(4,1);
    lcd1.print(" Irms_X ");
}
if(LowCurrent[5] == 1){
    lcd2.setCursor(0,1);
    lcd2.print("Low Current! ");
}
else{
    lcd2.setCursor(0,1);
    lcd2.print(InputReal[5]);
    lcd2.setCursor(4,1);
    lcd2.print(" Irms_Y ");
}
if(LowCurrent[6] == 1){
    lcd3.setCursor(0,1);
    lcd3.print("Low Current! ");
}
else{
    lcd3.setCursor(0,1);
    lcd3.print(InputReal[6]);
    lcd3.setCursor(4,1);
    lcd3.print(" Irms_Z ");
}
lcd4.clear();
lcd4.setCursor(0,0);
lcd4.print(PowerReal);

```

```

    lcd4.setCursor(6,0);
    lcd4.print(" W ");
    lcd4.setCursor(0,1);
    lcd4.print(PowerFactor);
    lcd4.setCursor(5,1);
    lcd4.print("PF");
}
// *****END PrintVoltageRMS*****

// *****
// *****Cap Bank Control*****
// *****
if(loopTime - taskTime[3] > taskPeriod[3]){
    taskTime[3] = loopTime-1;
    if(CapBankTimer < CapBankDelay){ //want to avoid turning cap bank on/off
quickly
        CapBankTimer = CapBankTimer+NumSamples;
    }
    else if(PowerFactor<.81 && CapBankState == 0){ //turn cap bank on
        digitalWrite(CapBankPin, HIGH);
        CapBankTimer = 0;
        CapBankState = 1;
    }
    else if(PowerFactor>.9 && CapBankState == 1){ //turn cap bank off
        digitalWrite(CapBankPin, LOW);
        CapBankTimer = 0;
        CapBankState = 0;
    }
}

// *****
// *****Update Settings*****
// *****
if(loopTime - taskTime[5] > taskPeriod[5]){
    taskTime[5] = loopTime-1;
    String settingName;
    String settingValue;
    File Settings = FileSystem.open("/mnt/sd/Settings.txt"); //open the settings.txt
file
    if(Settings){
        while(Settings.available()){
            character = Settings.read();
            while((Settings.available() && (character != '['))){ //scrolls until bracket
                character = Settings.read();
            }
            character = Settings.read();

```

```

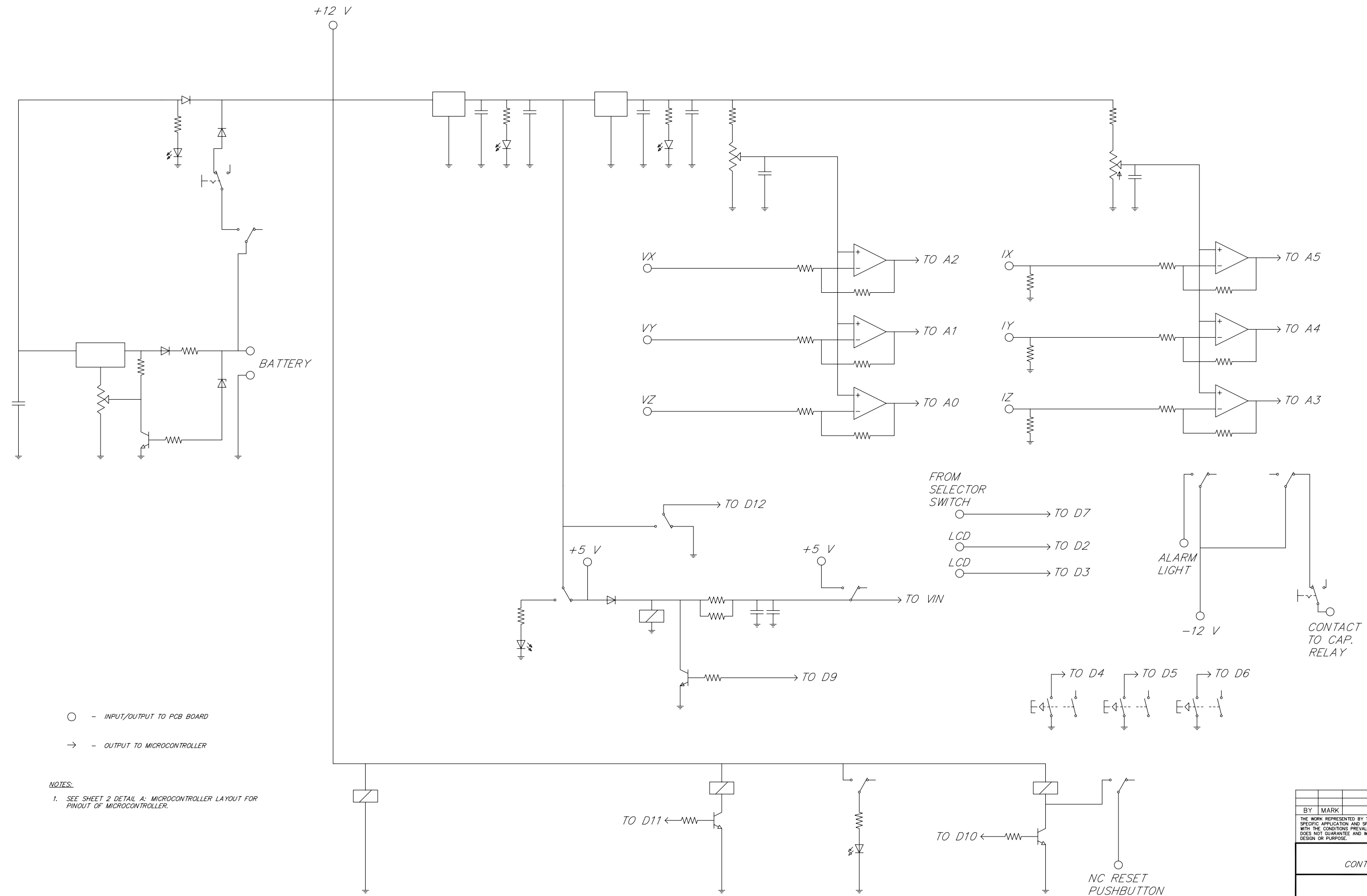
        while((Settings.available() && (character != '='))){ //reads values and sets to
setting name from [ to =
            settingName = settingName + character;
            character = Settings.read();
        }
        character = Settings.read();
        while((Settings.available() && (character != ']'))){ //reads values and sets to
setting value from = to ]
            settingValue = settingValue + character;
            character = Settings.read();
        }
        if(character == '']){
            applySettings(settingName, settingValue); //used to convert strings to usable
definitions
            settingName = "";
            settingValue = "";
        }
    }
    Settings.close();
}
}

}

/**Update Settings Function***/
void applySettings(String settingName, String settingValue){
    if(settingName == "NumSamples"){
        NumSamples=settingValue.toInt(); //converts string to int
    }
    if(settingName == "VoltageCutoff"){
        VoltageCutoff=settingValue.toInt(); //converts string to int
    }
    if(settingName == "CurrentCutoff"){
        CurrentCutoff=settingValue.toInt(); //converts string to int
    }
    if(settingName == "VoltageScale"){
        VoltageScale=settingValue.toInt(); //converts string to int
    }
    if(settingName == "CurrentScale"){
        CurrentScale=settingValue.toInt(); //converts string to int
    }
    if(settingName == "TO_EMAIL_ADDRESS"){
        TO_EMAIL_ADDRESS=settingValue;
    }
}
}

```

Schematics

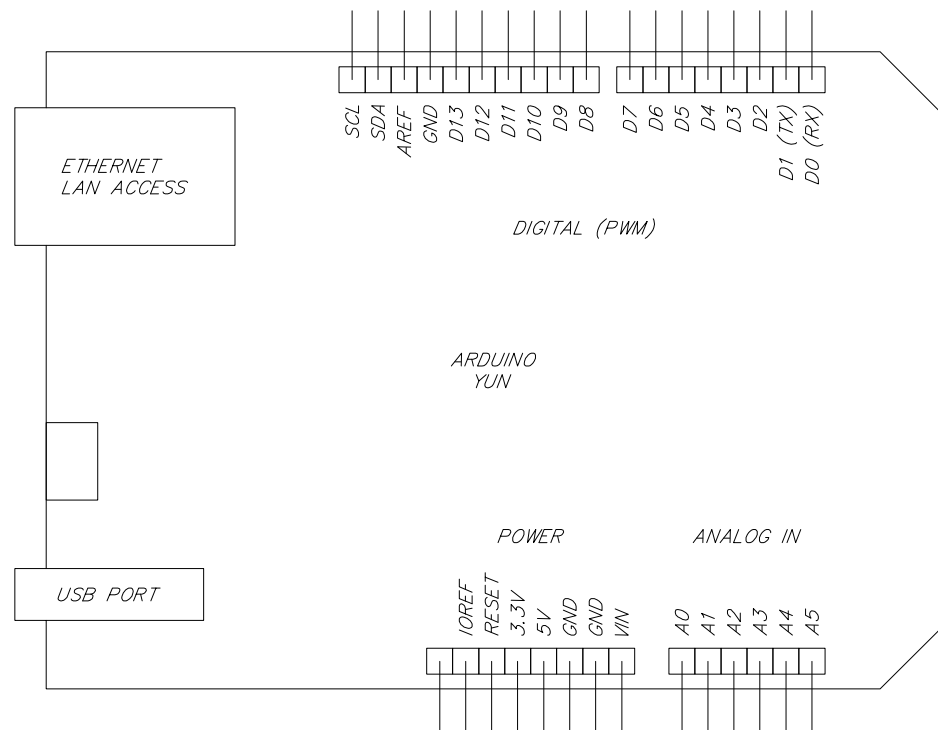


○ - INPUT/OUTPUT TO PCB BOARD
→ - OUTPUT TO MICROCONTROLLER

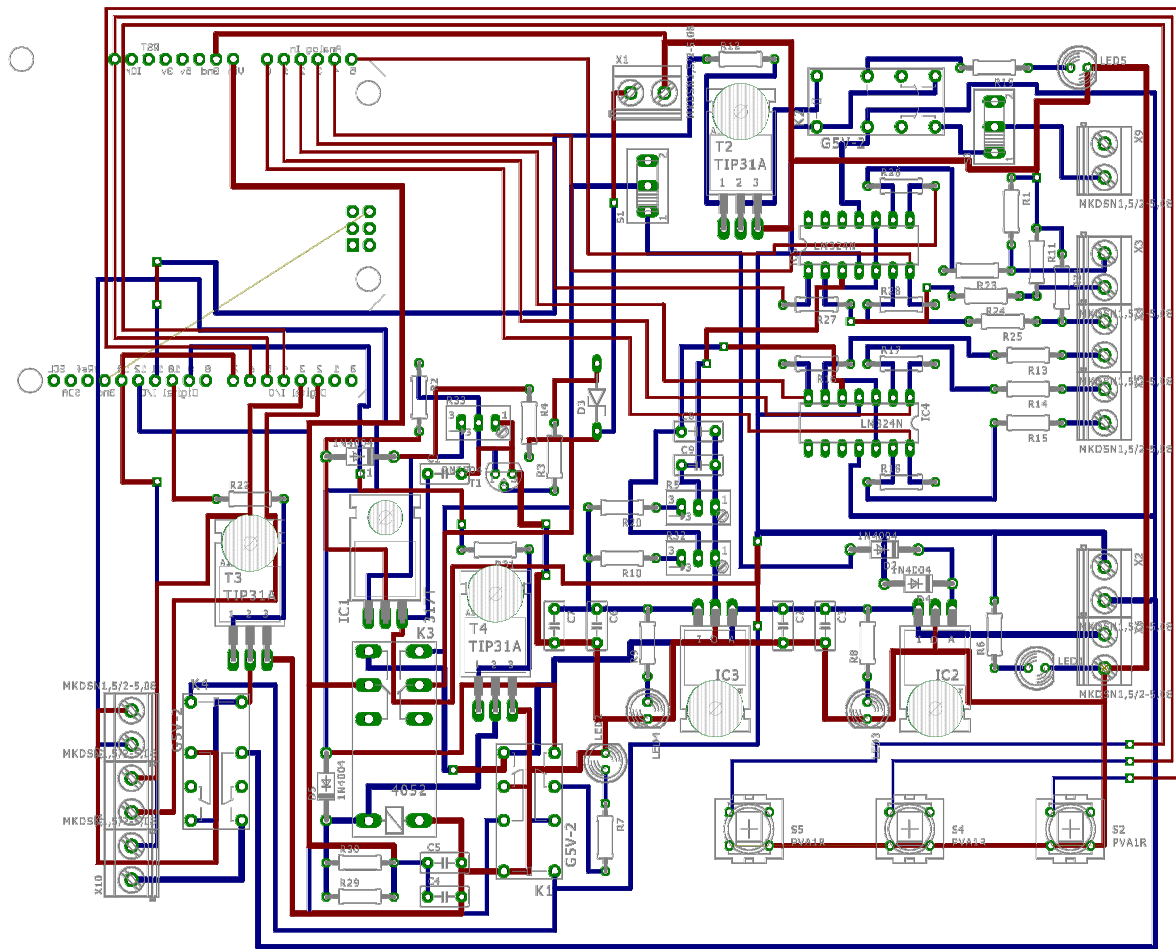
NOTES:
1. SEE SHEET 2 DETAIL A: MICROCONTROLLER LAYOUT FOR PINOUT OF MICROCONTROLLER.

PCB SCHEMATIC

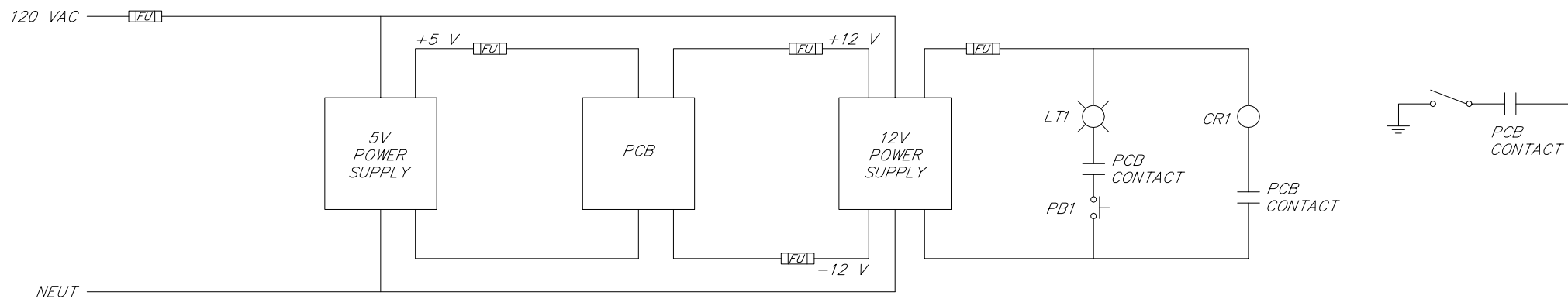
BY		REVISIONS	DATE
THE WORK REPRESENTED BY THIS DRAWING WAS DESIGNED BY THE ENGINEER FOR THIS SPECIFIC APPLICATION AND SPECIFIC LOCATION DESCRIBED HEREIN IN ACCORDANCE WITH THE CONDITIONS PREVALENT AT THE TIME THE DESIGN WAS DONE. THE ENGINEER DOES NOT GUARANTEE AND WILL NOT BE LIABLE FOR ANY OTHER LOCATION, CONDITION, DESIGN OR PURPOSE.			
GREAT LAKES CONTROLS & ENGINEERING			
PCB SCHEMATIC			
		MICHIGAN STATE UNIVERSITY Design Team 3 Incoming Grid Power Monitoring	
DE. BY: JRM	CH. BY:	PROJECT NO.	
DR. BY: JAJ	APP. BY:	SENIOR DESIGN	
STDS.	SHEET 1 OF 3		E
DATE: DECEMBER, 2015	FILE NO.		1
SCALE: AS SHOWN			



DETAL A: MICROCONTROLLER LAYOUT



DETAIL B: PCB LAYOUT



POWER SCHEMATIC

BY	MARK	REVISIONS	DATE
THE WORK REPRESENTED BY THIS DRAWING WAS DESIGNED BY THE ENGINEER FOR THIS SPECIFIC APPLICATION AND SPECIFIC LOCATION DESCRIBED HEREIN IN ACCORDANCE WITH THE CONDITIONS PREVALENT AT THE TIME THE DESIGN WAS DONE. THE ENGINEER DOES NOT GUARANTEE AND WILL NOT BE LIABLE FOR ANY OTHER LOCATION, CONDITION, DESIGN OR PURPOSE.			
GREAT LAKES CONTROLS & ENGINEERING			
POWER SCHEMATIC & DETAILS			
		MICHIGAN STATE UNIVERSITY Design Team 3 Incoming Grid Power Monitoring	
DE. BY: JRM	CH. BY:	PROJECT NO.	
DR. BY: JAJ	APP. BY:	SENIOR DESIGN	
STD.	SHEET 2 OF 3		E
DATE: DECEMBER, 2015	FILE NO.		2
SCALE: AS SHOWN			

