Mesh-Network Power Analyzer

Austin Cline
Alex Garcia
Ryker Elkins
Derek Janak

CONCEPT OF OPERATIONS

REVISION – Final Revision
7 March 2017

CONCEPT OF OPERATIONS
FOR
Mesh-Network Power Analyzer

TEAM ℧

APPROVED BY:

__Alexander Garcia_____3/7/17_
Project Leader Date

_____
Prof. S. Kalafatis Date

_____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	9/12/2016	Derek Janak		Draft Release
2	10/1/2016	Alex Garcia		Report Revision
3	12/4/2016	Alex Garcia		Semester Draft Revision
4	3/7/2017	Alex Garcia		Midterm Revision
4	5/4/2017	Derek Janak		Final Release

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

1. Executive Summary

The concept for this project is to provide a mesh network focused on diagnosing faults within the electrical systems of homes and businesses that uses relays to remove circuits from damage in the event of a hazardous condition. The system will utilize a network of wireless nodes communicating via a wireless protocol to relay information to a central database software on a desktop computer. Sensor nodes will be interfaced to electrical circuitry within the outlets, and they will sense a variety of electrical faults. In the event of a hazardous condition, the nodes will operate a relay to safely disconnect the circuit from power.

The nodes are to be battery powered and will consist of a power chain, sensors, and a wireless device. These devices will be operated by an on-board microprocessor. The device will relay the data to a USB device which will interface with a software database on the host device. The software will perform necessary calculations to determine the state of the circuit, and will broadcast commands back to the node devices. The software will also contain a User Interface for presenting data to users.

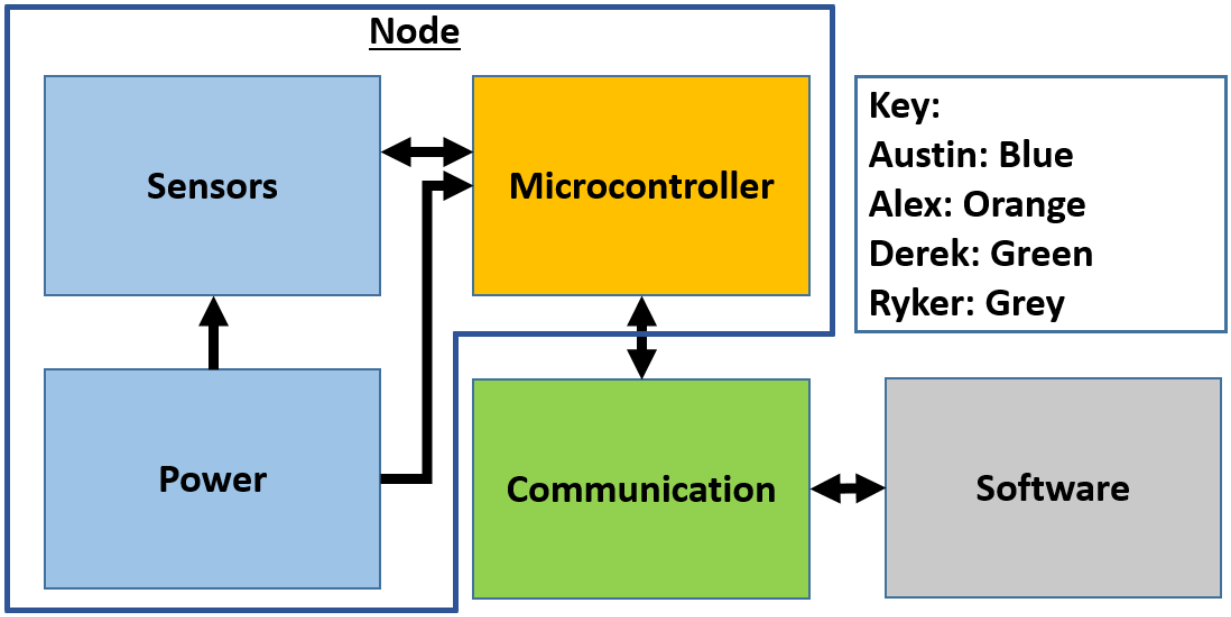


Figure 1: System Block Diagram

2. Introduction

Home automation is a growing field within the technical industry as the internet of things makes monitoring and controlling your home remotely into a much easier task. In fact, according to ABI Research, 1.5 million home automation systems were installed in 2012 in the U.S. alone, and it predicted that 8 million would be installed in 2017 (“1.5 million”). Additionally, Statista reports that Home Automation market has grown from a 770 million dollar industry in 2014 to a 3.6 billion dollar industry in 2016, and projects that it will be about an 8 billion dollar industry by 2020 (“Home Automation”). More people are becoming interested in being efficient and knowledgeable about their energy usage. This project addresses the issues of monitoring energy consumption, diagnosing electrical failures, and intervening in certain cases of electrical faults.

2.1 Background

There are a few similar devices on the market: most notably the variety of wireless smart plugs. These devices are easier to install than this project, but do not particularly address the issue of diagnosing electrical failures. Also, the majority of these products do not offer any data on power consumption, or diagnostic information for power surges, open circuits, or closed circuits. These devices can tend to be bulky, often covering both of the outlets of a two outlet socket, and aesthetically displeasing. This project won’t be visible after installation, and will be installed within the outlet.

The Wireless Power Manager (WPM) is another very similar device, which was a previous Capstone Project here at A&M. The WPM measured necessary circuit parameters for power consumption data, and focused on using this data to more directly control power consumption via scheduling circuit activity. Unfortunately, it was an incredibly bulky device, and wasn’t really practical for actual use in a home or business. This project will be small enough to fit within the outlet, so that it is out of sight. Also, our future goal is to implement a user scheduling system.

In conclusion, there is not an easy to find device on the market that addresses the issues of monitoring energy consumption, diagnosing electrical failures, and intervening in certain cases of electrical faults. This project could be very useful if it is kept to a reasonable size, and can be easily scaled up to manage enough nodes to fully cover a home or business.

2.2 Overview

This project is comprised of a number of sensor nodes which monitor and collect data on necessary circuit parameters for power consumption at an individual outlet. These nodes will

also detect electrical faults after they occur such as power surges, open and closed circuits. They will wirelessly communicate to a central hub which will connect to the user's computer to store and analyze data. A UI will report electrical problems in the user's home, display power consumption data, and basic data analytics. More specifics will be provided throughout this document.

2.3 Referenced Documents and Standards

"Home Automation - United States | Statista Market Forecast." *Statista*. Statista, n.d. Web. 14 Sept. 2016. <https://www.statista.com/outlook/280/109/home-automation/united-states #market-smarthomes>.

"1.5 Million Home Automation Systems Installed in the US This Year." *Arch.com*. ABI Research, 19 Nov. 2012. Web. 14 Sept. 2016. <https://www.abiresearch.com/press/ 15-million-home-automation-systems-installed-in-th/>.

This project will use UART TIA RS-232 standard for on board communication.
This project will use IEEE 802.15.4 wireless devices and protocols for wireless communication.
This project will need to follow any applicable home electronic standards and will adhere to the United States standard AC power at 120V and 60Hz.

3. Operating Concept

3.1 Scope

The objective of this project is to implement a mesh network designed to diagnose electrical faults and shut down active circuits under emergency conditions. These network nodes will be low power and capable of being installed in an outlet.

The current goals of this project include:

- Designing a wireless mesh network capable of supporting up to 10 end nodes
- Implementing sensing of relevant circuit parameters necessary for diagnosis
- Facilitating wireless transfer of collected data to and from a hub device
- Compiling data from the network within a database for analysis
- Analyzing and displaying pertinent data via a desktop UI
- Implementing safety shutdown of dangerous circuits
- Handling fault conditions such as power loss and node failure

The sub-phases of this project will include, but are not limited to, the following:

- Power source selection and power chain design
- Sensor selection and implementation
- Node-local hardware interfacing and firmware
- Achievement of Node to hub communication
- Software handling of data received from nodes

The resources that we have access to are the PCB machines in the fabrication shop, the soldering stations, 3-D printers, and tools provided by the EIC. Collectively, we also have access to personal logic analyzer, oscilloscope, and soldering equipment. The estimated budget for the project is the $500 provided by the department. The design process shall be completed before December 2016 and then the project will be finished before the showcase in May 2017.

3.2 Operational Description

Our project will be used by homeowners and businesses to monitor energy consumption and facilitate electrical repairs. The project would be typically installed by electricians to the existing home circuitry. The device will be used inside the house and will be concealed within the wall socket. Since it is installed within the wall, the size of device is constrained to the size of the socket.

3.3 Constraints

An ethical concern is the privacy of the user's power data. Will the users power data be private or will we be able to access their data for our own purposes? What information will the user need to provide during system initialization and setup? We are also concerned with how secure the IEEE Standard 802.15.4 RF communication line will be. Could someone other than the user receives the data if the are close enough to the user's home. Additionally, how could someone hijack the system, and what harm could actually be done? Economically, we need to determine how much will the product be sold for. One of our highest concerns is that the home electrical codes must be followed, and the system must be safe to the user's health and safe for the environment.

3.4 System Description

Power Chain Design

This system will be responsible for powering the sensing nodes. The design of this system will initially include battery management hardware and regulators to provide an efficient and stable low-power supply to the on-board devices. Our future goal is and additional LDO with AC-DC converter drawing power from the host circuit.

Node Hardware Design

The node hardware subsystem design will require selection of appropriate sensors, wireless device, and a low power microcontroller. Relevant design parameters will include selection of communication protocol, schematic design, and board layout. This subsystem must also interface closely with the power chain and node firmware subsystems.

Node Firmware Design

The node firmware subsystem design will require development of robust software capable of implementing on board communication protocols and capable of initiating data transmission to the host via the wireless device. This system is also responsible for handling reception of transmissions from the host when applicable.

Database & UI Design

The database and UI subsystem will require development of software necessary to transmit and receive data from individual nodes. The software will be responsible for compiling this data within a database, running basic analyses of the data, and displaying relevant parameters via a User Interface. This software should also implement basic error handling/diagnosis for instances such as electrical failure of a node or data transmission failure.

3.5 Modes of Operation

There are four major modes of operation for this project:

- The passive mode collects data and routes it to the hub for analysis.
- In the event that a potentially hazardous fault is detected, the system will enter the active mode of operation, in which it will disable affected circuits until the fault is handled.
- The manual mode of operation will allow a user to request data and diagnostics or to shut down circuits at will via the user interface software.
- Finally, the sleep mode will allow the device to idle in a low power state between periods of activity.

3.6 Users

Home and business owners will use the project in monitoring the condition of their electrical circuitry. Aspects monitored will include but are not limited to fault conditions, power consumption, electrical outages, and power surges. The goal of this project is to make users more responsible and informed energy consumers. Users monitoring their homes or businesses through the system will need basic computer skills and training to operate the interface; this training is available through an instruction manual. Other potential users include electricians who might install the system into the electrical sockets for use in maintenance and repair work. For the qualified electricians, installation will not require any special training and will be no more difficult than making routine electrical repairs.

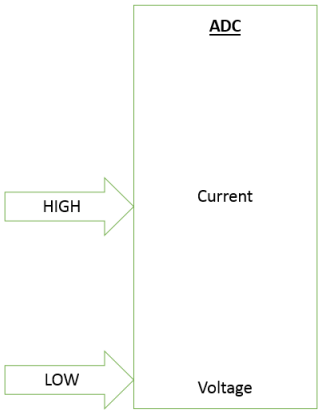
3.7 Support

Users would receive support via the project manual, which should catalog the necessary setup and operation of the mesh network. Applicable sections of the project manual should include hardware description and installation instructions, as well as complete documentation of the software. Raw software files will be provided in a read only format as well. Additionally, the user should seek hardware installation assistance from the building contractor or from a certified electrician.

4. Scenarios

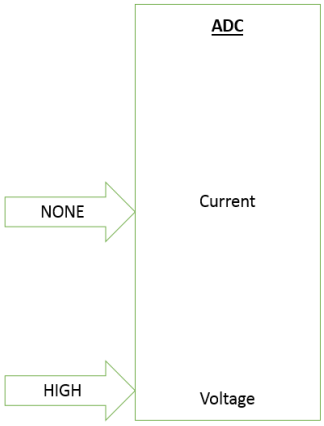
4.1 A short circuit occurs in the wiring of a home or business

In a home or business a short circuit could occur at anytime inside the walls. When this occurs the sensors in network shall detect that the short circuit has occurred, and the MNPA shall send a notification to the user's personal device. This information will help users detect, locate, and repair the short circuits.



4.2 Open circuit occurs in the wiring of a home or business

In a home or business an open circuit could occur inside the walls. When this occurs the sensors in network shall detect that the short circuit has occurred, and the MNPA shall send a notification to the user's personal device. This information will help users detect, locate, and repair the open circuits.



4.3 Power surge protection

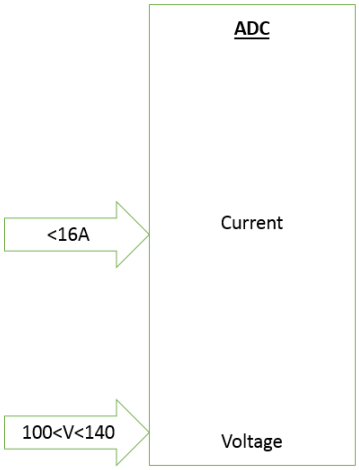
During a thunderstorm it is not uncommon for a power surge to occur. When this happens, the MNPA shall be able to detect the surge with a future goal of shutting off the circuits before it damages the electronics. In addition, a notification shall be sent to the user's personal device so that they can check their electronics.

4.4 Power loss detection and analytics

Power consumption is a hot topic whether it is being used in a home or business due to the fact that it relates directly to saving money. The MNPA could provide assistance in this area and give users the opportunity to run analytics on their power consumption.

4.5 Live circuit detection and notification

The MNPA provides users with a lot of information when it comes to locating necessary repairs such as shorts and open circuits. However, the MNPA will provide additional tools to ensure safety such as the the live circuit detection and notification. This feature will help users identify what breakers need to be switched off in order to repair the wiring safely and efficiently.



5. Analysis

5.1 Summary of Proposed Improvements

Summary of Improvements provided by the MNPA

- Hidden automation for aesthetic appeal
- Fault diagnostic information
- Detection of open and short circuits
- Control of live or off circuits from a computer

5.2 Disadvantages and Limitations

Disadvantages and Limitations include:

- Ease of installation will vary depending on users
- Size limitations due to installation behind an outlet

- Decreasing size reduces power efficiency
- Installation could take a long time depending on ease of installation and number of nodes
- Only guaranteed to work indoors
- Not guaranteed to protect outlets from excess current

5.3 Alternatives

Wireless Smart Plugs

- Easier installation than the MNPA
- No diagnostic information for power surges, open circuits, or short circuits
- No power analytics depending on how expensive the smart plug is

Wireless Power Manager

- Has the ability to work outdoors

Governor (cutoff switch)

- Easier installation

Mesh-Network Power Analyzer

Austin Cline
Ryker Elkins
Alex Garcia
Derek Janak

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – Semester Draft
4 May 2017

FUNCTIONAL SYSTEM REQUIREMENTS
FOR
Mesh Network Power Analyzer

PREPARED BY:

_____
Author Date

APPROVED BY:

_Alexander Garcia_____12/4/16
Project Leader Date

_____
Dr. Sam Villareal Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	9/22/2016	Austin Cline	N/A	Draft Release
2	10/1/2016	Alex Garcia		Presentation Revision
3	12/4/2016	Alex Garcia		Semester Draft
4	5/4/2017	Derek Janak		Final Release

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

1. Introduction

1. Purpose and Scope

The Mesh-Network Power Analyzer is a network of sensor nodes that work together to provide fault detection, control, and provide the user with information on energy consumption with regards to a user's home circuitry. The Mesh-Network will wirelessly communicate raw data to a central hub which will process this data and provide control, as well as provide the user with information via the user interface. This specification defines the technical requirements for the development items and support subsystems delivered to the client for the project. Figure 1 shows a representative integration of the project in the proposed CONOPS. The verification requirements for the project are contained in a separate Verification and Validation Plan.

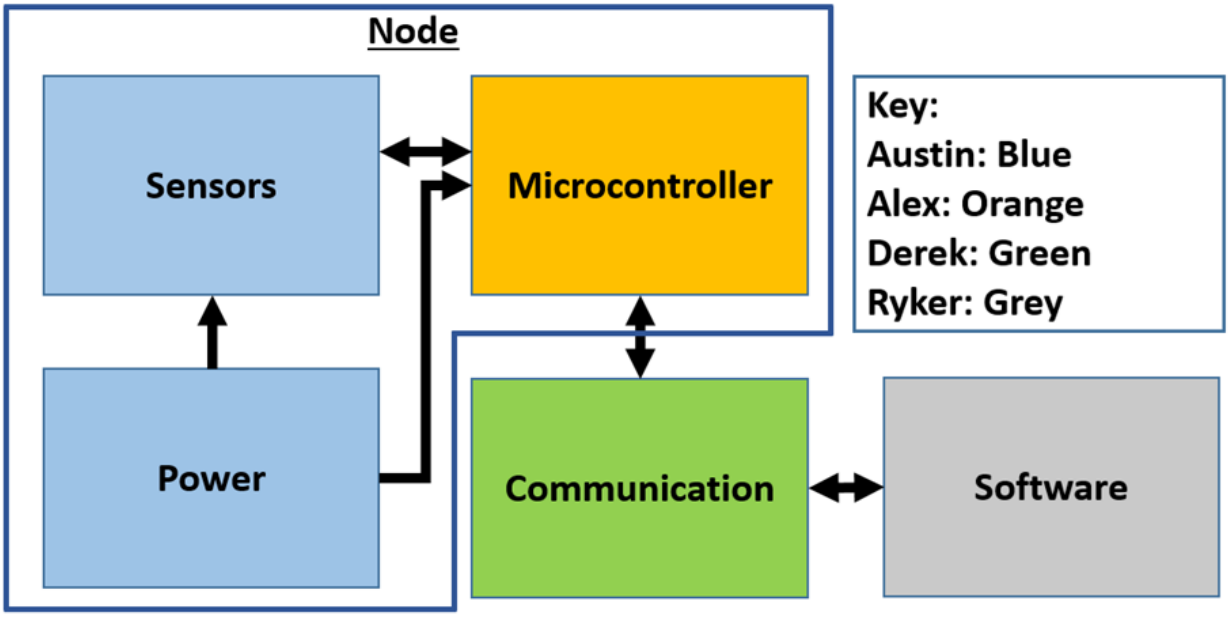


Figure 1. Project Conceptual Image

2. Responsibility and Change Authority

Although each team member will be in charge of their primary subsystem, there will be a lot of overlap and cooperation between members on all of the subsystems. Since this project was proposed by us as a team, we are the clients. Each team member has the responsibility of making sure requirements are met and the authority to make changes with a majority of the team's approval.

Team Member	Subsystem	Responsibilities
Austin Cline	Power/ Sensors	Provide power for all of the nodes and any boards/controllers. Implementation of sensors and other hardware.
Ryker Elkins	Software	Construct Graphical User Interface and implement data analysis.
Alex	Microcontroller	Write microcontroller code that reads signal, triggers relay,

Garcia		and uses UART(serial communication) to send receive data.
Derek Janak	Communication	Provide wireless communication between the processor, nodes, and database. Compose the packet structure for data.

2. Applicable and Reference Documents

1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	Revision/Release Date	Document Title
NFPA 70	2014	National Electrical Code

2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification, and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
IEEE 802.15.4	2013	Standard for Low Rate WPAN

3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable document are considered to be for guidance and information only,

with the exception of ICDs that have their applicable documents considered to be incorporated as cited.

3. Requirements

1. System Definition

The Mesh-Network Power Analyzer is a system of distributed sensing nodes which monitor conditions within the major electrical circuits found in residential homes and businesses. Each node will measure current and voltage conditions present in the host circuit and will transmit this information back to a host device for storage and processing. The user will have access to the data through a user interface located on the host device. Each node will be battery powered. In the event that dangerous faults are detected within the circuit, the nodes will deactivate the circuit by triggering a relay. The user may also manually control various circuits through the user interface software.

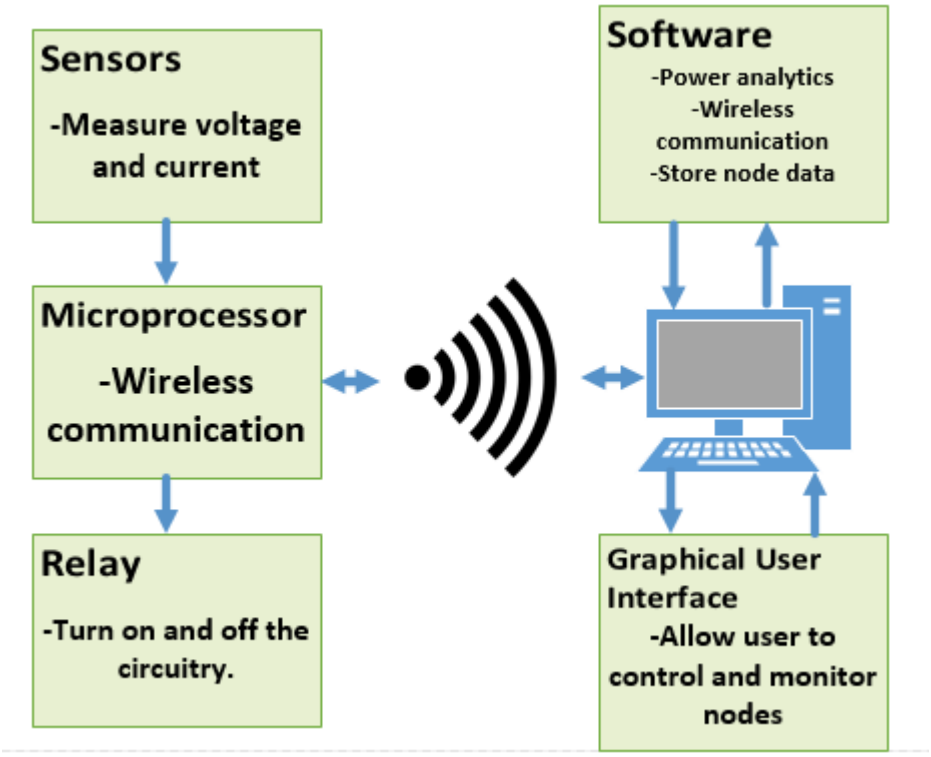


Figure 2. Block Diagram of System

The entire system may be divided into several smaller subsystems. These subsystems include node hardware, node firmware, host device software and user interface, and node power. Each subsystem carries out its own particular role in device operation. Node hardware consists of the physical layout, component selection, and routing of traces for node devices. Node firmware drives the node hardware by implementing error handling, data collection, and communication protocols such as UART and IEEE Standard 802.15.4 transmission, as well as controlling the state of the relays used in circuit shutdown operations. Host device software consists of the database, analysis, and user interface which collect and process data from the nodes and send signals back to the nodes when necessary. Node power is responsible for powering each remote node with the stable, efficient DC supplies necessary for the various chips to operate.

2. Characteristics

1. Functional / Performance Requirements

1. Battery Life

The node devices shall exhibit a minimum functional battery life of 1 hour.

> Rationale: The node devices are currently meant to be a proof of concept and therefore do not need to have a long battery life. In the event that this device is moved out of the proof of concept stage this requirement will need to be revisited.

2. Transmission Range

The maximum functional transmission range from router to central hub will be 20m. The full network, may be capable of additional range via multi-node routing.

> Rationale: The node devices must be able to support transmission across a reasonable distance to allow them to be spread throughout the circuits present in a typical home or business. The range must be sufficient for the devices to transmit from distributed locations to the central hub.

3. Network Capacity

The mesh network must be capable of support for a minimum of 10 distinct end nodes.

> Rationale: The maximum number of breakers allowed per panel is 42, but for proof of concept if we can implement this for 10 nodes, we can extend it to 42. This is the minimum capacity needed to guarantee that a single network can service a full panel.

4. Database Capacity

The database will be sufficiently large to contain all relevant data generated over the course of one month at minimum.

> Rationale: The typical billing cycle for utilities is one month. The database shall hold a minimum of one billing cycle's worth of data in order for it to be useful in analyzing energy usage trends.

5. Fault Detection

The device shall have a 99% fault detection rate for all anticipated faults in the design.

Rationale: The device will have a very high detection rate for all anticipated faults monitored by the design. The one percent allowance corresponds to faults occurring in the presence of external or unanticipated factors.

2. Physical Characteristics

1. Board Area

The node pcb will require maximum board area of 9 square inches.

Rationale: The devices must remain small enough to fit inside of a typical junction box, but must be large enough to afford sufficient room for device placement.

2. Volume Envelope

The devices shall have critical dimensions of of 4½ x 2¾ x ½ inches.

Rationale: The device will be able to fit inside of the typical wall outlet space.

3. Mounting

The devices will be wired such that the host circuit passes through the terminals of the breaker and so that the sensors are able to detect the voltage and current present in the circuit.

Rationale: The electrical connection is necessary to allow sensing of parameters and control of the circuit in the event of a shutdown condition.

4. Thermal Characteristics

The devices will exhibit low thermal dissipation. Dissipated heat will not exceed 100 mW/cm^2.

Rationale: The devices must fit into an enclosed space with minimal airflow and little room for a heat sink. Additionally, the devices may be close to wire insulation and will not dissipate enough heat to risk development of exposed wiring sections, as this could increase the risk of short circuit conditions.

5. Device Enclosure

The devices will be enclosed within a casing constructed from a durable material.

Rationale: The casing is needed to protect the devices from strain and compressive forces, as well as to decrease risk of shorting between the PCB and potentially exposed wiring.

3. Electrical Characteristics

1. Inputs

The nodes will have inputs in the form of electrical signals flowing through the relay, data measured by the sensors, and commands received via wireless communications.

Rationale: These are the signals necessary for the node to control power to the circuit, detect fault conditions and ambient power usage, and receive information from the central hub.

1. Power Consumption

The power consumption of the device will not exceed 100 mW.

Rationale: The device must support an extended battery life. Therefore, it shall operate with relatively low power requirements.

2. Supply Voltage Level

The devices will be supplied with dc-dc regulated power carrying minimal ripple. The supply voltage levels will comply to established 3.3V or 5V logic levels.

Rationale: The device requires low noise from the power supply in order to avoid disrupting the measurement of the host circuit. The device also requires standardized logic levels to minimize need for level shifting and related complexities.

3. Communication Protocols

The device will utilize established serial and wireless communication protocols (such as UART or IEEE Standard 802.15.4). The same communication protocol will be used for all communications of similar type (i.e. wireless versus on-board). Wireless communications will be carried out in noncommercial, unlicensed frequency bands.

Rationale: The standardization of communications protocols is necessary to establish stable intra-system communication of data and commands.

2. Outputs

1. Data Output

Data will be output from nodes via wireless communication. It will be accessible to the user via the user interface.

> Rationale: The data is transmitted back to a central location where the user may access comprehensive information regarding the system.

2. User Interface

The user will have access to the system via a graphical interface that can be launched as a computer application.

> Rationale: Users must be able to view and access data via the host computer. The data will be presented in such a way that users of low to moderate technical sophistication may easily access it.

3. Connectors

The nodes and system hardware will make use of standard commercial connectors, such as USB connectors and pin headers.

> Rationale: The device connections must be easily serviceable and accessible using established market devices for programming and connection to a computer or other device.

4. Wiring

Standard wiring practices conforming to the National Electrical Code as well as to local standards will be used in connection and mounting of nodes within wall sockets and junction boxes.

> Rationale: The device is intended to be integrated into home and business wiring, and therefore it must comply with applicable safety regulations and best practices.

4. Environmental Requirements

The Mesh-Network Power Analyzer is designed to withstand and operate in the environments specified in the following section.

1. Thermal

The device operates in temperature ranges between -40 degrees and 125 degrees Celsius.

> Rationale: This temperature range corresponds to the acceptable temperature rating for most standard silicon devices.

2. Humidity

The device operates in low to moderate humidity conditions characteristic of a climate-controlled indoor environment.

> Rationale: The system is intended for use in indoor and sheltered environments within homes and businesses.

4. Support Requirements

The Mesh-Network Power Analyzer will come with an installation/setup manual to get users started.The customer will need a computer with a Windows 8 operating system(or later) for the software, and Wireless Hub. The customer may need the help of an electrician for installation of the nodes. However, the Hub and software can be installed with the aid of the user manual.

Appendix A Acronyms and Abbreviations

EMI	Electromagnetic Interference
GUI	Graphical User Interface
ICD	Interface Control Document
mA	Milliamp
mW	Milliwatt
PCB	Printed Circuit Board
RMS	Root Mean Square
TBD	To Be Determined
TTL	Transistor-Transistor Logic
USB	Universal Serial Bus
SPI	Serial Peripheral Interface

Appendix B Definition of Terms

No terms to be defined at this time

Mesh Network Power Analyzer

Austin Cline
Alex Garcia
Ryker Elkins
Derek Janak

INTERFACE CONTROL DOCUMENT

REVISION – Final Revision
3 March 2017

INTERFACE CONTROL DOCUMENT
FOR
Mesh Network Power Analyzer

PREPARED BY:

_____
Author Date

APPROVED BY:

_Alexander Garcia_____3/7/17_
Project Leader Date

_____
Prof. Stavros Kalafatis Date

_____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	9/25/2016	Austin Cline		Draft Release
2	12/4/2016	Alex Garcia		Semester Draft
3	3/7/2017	Alex Garcia		Midterm Revision
4	5/4/2017	Derek Janak		Final Release

Table of Contents

List of Tables

List of Figures

1. Overview

This ICD will detail the entire system of the Mesh Network Power Analyzer and how it will accomplish our requirements specified in the FSR. Included are detailed descriptions of our systems physical interface which will include information on size and location of use. Our electrical interface section will discuss all of our electrical aspects such as our current sensors, power supply, and much more. Finally our communications interface will discuss what we plan to use for communication as well as how it will work together.

2. References and Definitions

1. References

NPFA 70
National Electrical Code
2017

IEEE 802.15.4
Standard for Low Rate WPAN
2013

2. Definitions

EMI	Electromagnetic Interference
GUI	Graphical User Interface
ICD	Interface Control Document
FSR	Functional System Requirements
mm	Millimeter
mA	Milliamp
mW	Milliwatt
PCB	Printed Circuit Board
RMS	Root Mean Square
TBD	To Be Determined
TTL	Transistor-Transistor Logic
MNPA	Mesh Network Power Analyzer
USB	Universal Serial Bus
LDO	Low Drop Out
MCU	Microcontroller Unit

3. Physical Interface

1. Dimensions

1. Dimensions of MNPA node

The MNPA node volume shall have a total volume of 4.5 x 2.75 x 0.5inches,
The PCB board shall have an area of 5806 square mm and will include:

Component	Size (mm^2)
Isolation Amplifier (ACPL-7900-500E)	62

TI CD4007 OP AMP	84
Current Transformer (B82801C565A100)	288
TI CD4007 OP AMP	84
16A 2-Coil Latching Relay (RT314F05)	368
3.3V Coin Cell Battery Holder	617
3.3V LDO (LP2985-33DBVR)	9
5V LDO (TPS76350DBVR)	9
TI MSP430	178
XBEE S2C 802.15.4	747
Total: 2,446 mm^2	

Table 1 - PCB component dimensions

The table above shows all of the components that will need to be placed on the PCB board and the dimensions they would require. Based upon these values our PCB will be under the 5805 square mm requirement.

2. Dimensions of MNPA usb dongle (Xbee stick)

The MNPA will utilize the XBEE XStick. The XStick has dimensions of 5.83cm x 2.03cm x 1.08cm.

2. Location of Use

1. Location of MNPA node

MNPA nodes shall be placed in the users in wall circuitry. This location will be accessed from power outlets where the MNPA node will be installed and then covered by the outlets.

2. Location of MNPA dongle

The MNPA dongle will be placed in the USB port of a user's computer. This could pose a issue of communication range depending on node location.

4. Electrical Interface

1. Primary Input Power

Two Duracell 2032 coin cell batteries will be used in series for the MNPA. These batteries were chosen due to our need for a 6V and the combination of two batteries in series fulfills that requirement. In addition, if the batteries ever needed to be changed we wanted to use one of the most common brands on the market so that it would be easy for users. Duracell is one of the two large competitors in the market fulfilled this need. Finally, Duracells battery have a good battery life as can be seen in the figure below.

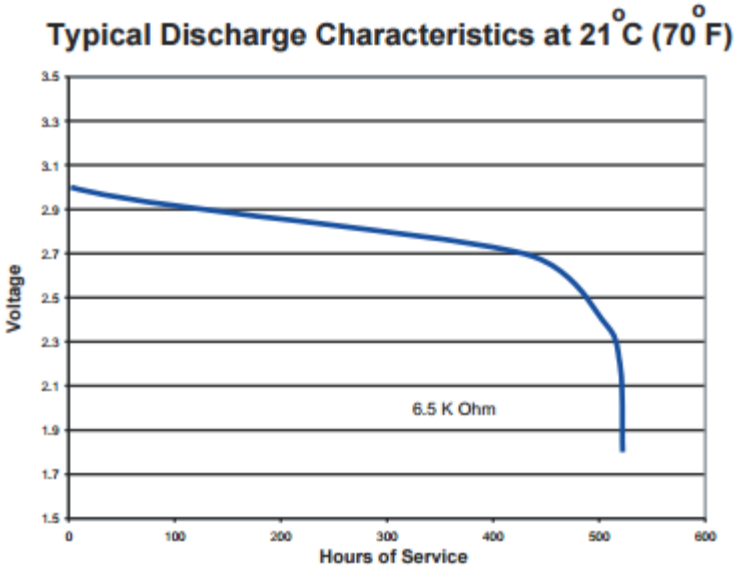


Figure 1 - Duracell battery life

The batteries will be held by the MPD BH800S - Coin Cell Holder. This coin cell battery holder picked due to its small PCB footprint, PCB through hole, and the ability to hold two 2032 coin cell batteries

2. Relay Control

The RT314F05 by TE was selected to be our relay. Reasons for this include the low turn on voltage (5V), 16A current rating, and two coil latching system. This relay give us the ability to easily control the relay's state by using two MSP430 output's and two 2N7000 transistors.

3. Current Sensor

We will be using the TDK B82801C current sense transformer to measure the current that is flowing through the circuit being monitored by each node. It has a wide operating temp that falls perfectly within our specifications(-40 to 125°C).

Following the current transformer the current measurement sub-circuit will utilize the TI OP07CDR to create a differential amplifier to increase our voltage so that the MSP430 will have a large enough signal to accurately measure.

4. Voltage Sensor

For voltage sensing, we have designed a non inverting amplifier to sample the voltage of the power line into and decrease the voltage into a measurable value by the micro controller. Then since the resistors and equations will be known we can relate this to the actual voltage on the power line.

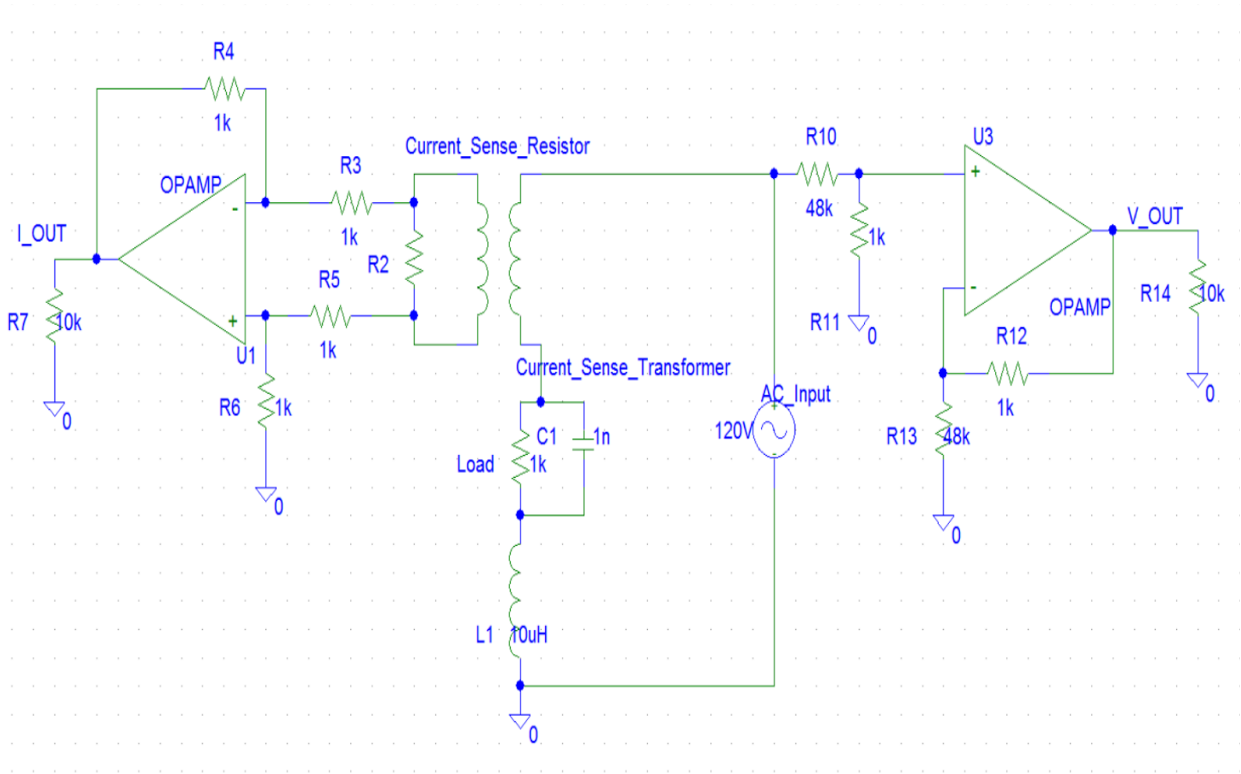


Figure 2- Voltage Sensor

5. Microcontroller

We will be using the Texas Instruments MSP430G2553 for this project. This device was chosen for its Ultra-Low Power consumption; around 250uA while active and 0.1 uA on standby. The MSP430 also have high precision 10-bit ADC for sampling the analog voltage and current signals for processing. Initially we will be using the launchpad to get a working prototype. The end result will be a custom PCB that uses this microprocessor programmed from the launchpad.

6. User Control Interface

The User Control Interface will be a software GUI that allows the user to directly monitor and view power consumption data, error and command logs, and system health. It will also allow the user to input commands to individual nodes, and will include a system setup and configuration wizard.

5. Communications / Device Interface Protocols

1. Wireless Communications

The XBEE S2C will be placed on all PCB boards in order to allow wireless communication to and from the nodes. All information from the nodes will be sent by the XBEE S2C and received by the DIGI XStick discussed in the next section.

2. Host Device

We will be using the DIGI XStick to serve as the hub between the computer database and the routers and nodes. The Xstick has an indoor range of 20m which will be sufficient for most of the applications. However, with the use of intermediate routers, we can extend the range of the mesh further. Additionally, the host device will have software designed to store and analyze data, process the errors after they occur and then to send commands to the nodes.

3. Device Peripheral Interface

We will be using the serial peripheral interface for the communication between microcontrollers. The S peripherals are the sensors of the board.

Execution Plan

Tasks	1/23/2017	2/6/2017	2/20/2017	3/6/2017	3/27/2017	4/3/2017	4/10/2017	4/17/2017	4/24/2017	5/1/2017
Verify Current Subsystem Capabilities										
Brainstorm Necessary Improvements/Modifications										
Order Parts/ PCB Design										
Add user input to Data Display GUI										
Add functionality to Settings and Setup GUI										
Verify API Communications Packets										
Add functionality to the Log GUIs										
Complete Basic Communications Network										
Add functionality to the Input Commands GUI										
Complete Communications Revisions										
PCB Submission										
Optimize software block interfaces and code										
Troubleshooting Phase										
Software testing and fixing										
Demo System Integration/Video										
Final Project Demonstration										
Final Report										

Key:	
Completed	
Significant Progress	
To Be Completed	

All pending tasks were completed. Any further development would focus on further troubleshooting and addition of new feature sets.

Hardware Subsystem Writeup
05/03/2017
Austin Cline

This subsystem consists of all hardware and PCB design for the MNPA. This includes the power system, voltage measurement system, current measurement system, and relay system. For the measurement systems, I will be responsible for stepping down the voltage values to an acceptable values so that the MSP430 microcontroller can measure them without failure. Currently, our device plans to use two 3V CR2032 coin cell batteries as the power supply. This voltage will be fed to two different LDOs. One will step the voltage down to 3.3V, for the XBee, MSP430, and op-amps, and the other will step the voltage down to 5V in order to power the relay. In addition, our system requires a relay to disconnect the home circuitry that can be controlled by the microcontroller. The largest challenge for this subsystem are the size constraints which per the FSR require that our PCB have an area of 9 in^2 or 5806.44 mm^2. Below is a diagram for the hardware detailing the connecting pieces as well as how it interacts with the other subsystems.

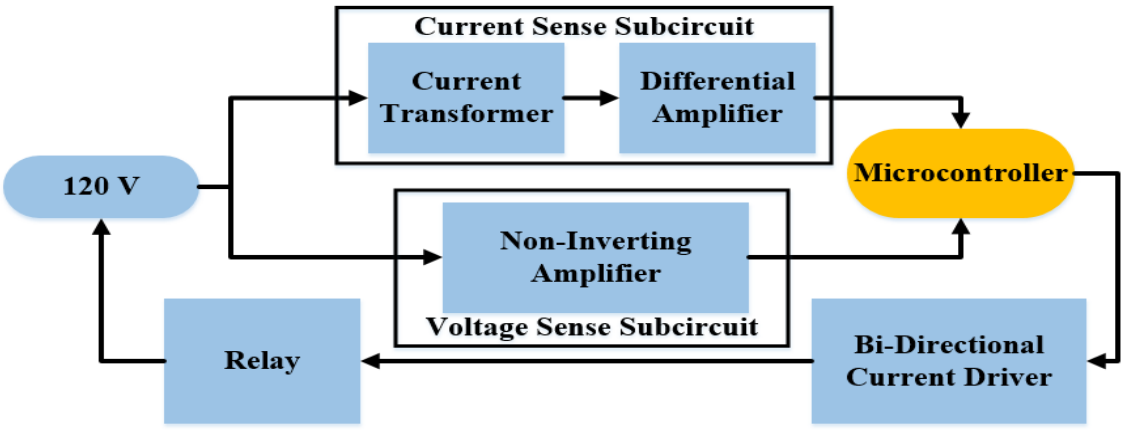


Figure 1: Hardware Diagram

Voltage Measurement System:

Simulation Results**:**

Below in figures one and two are the results from circuit simulations. Figure one is the interactive simulation for the voltage measurement circuit drawn out on Multisim. Key takeaways from this simulation are that based on the resistors used our gain should equal:

Theoretical Gain of a Non-Inverting Amplifier = R2/R1 = 1.1k/100k = 0.011

Based on our simulated input and output voltages seen in figure 1 we calculate our gain to equal:

Simulated Gain of a Non-Inverting Amplifier = Vout/Vin =0.22 V/20 V = 0.011

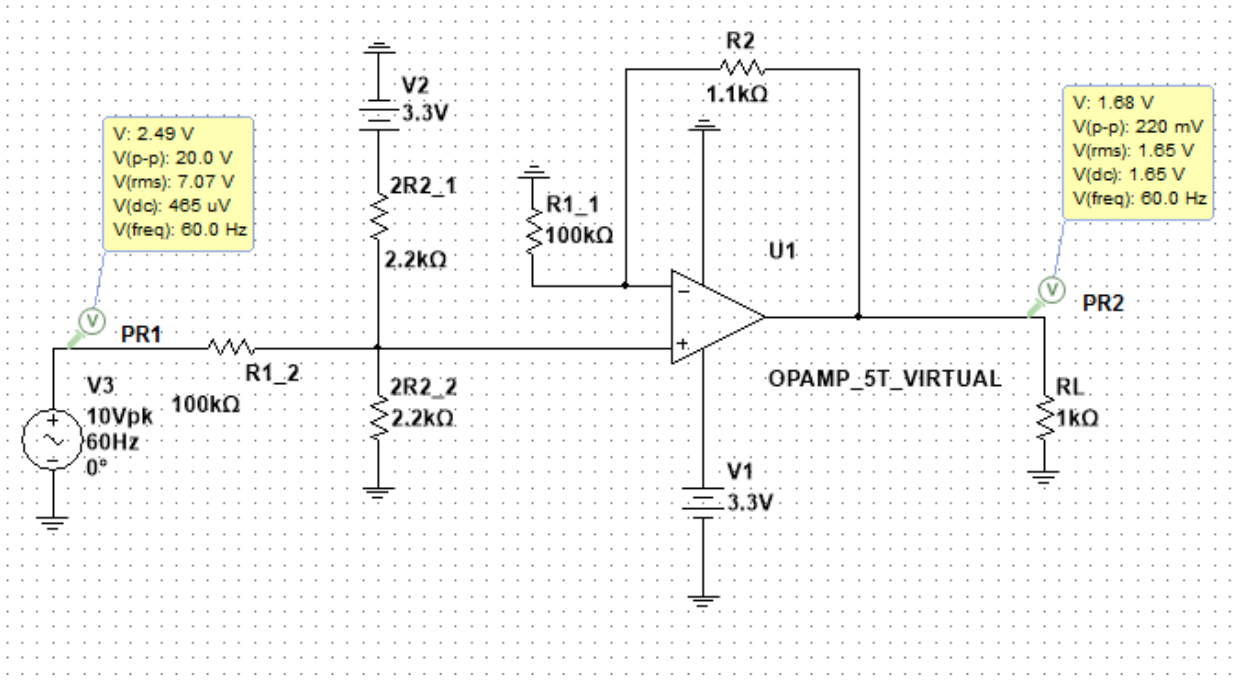


Figure 1: Voltage Measurement Sub-circuit Schematic

In addition to the correct gain on our simulation we are able to see that our DC voltage shift is working as expected by the V(dc) = 1.65 V seen in figure 1.

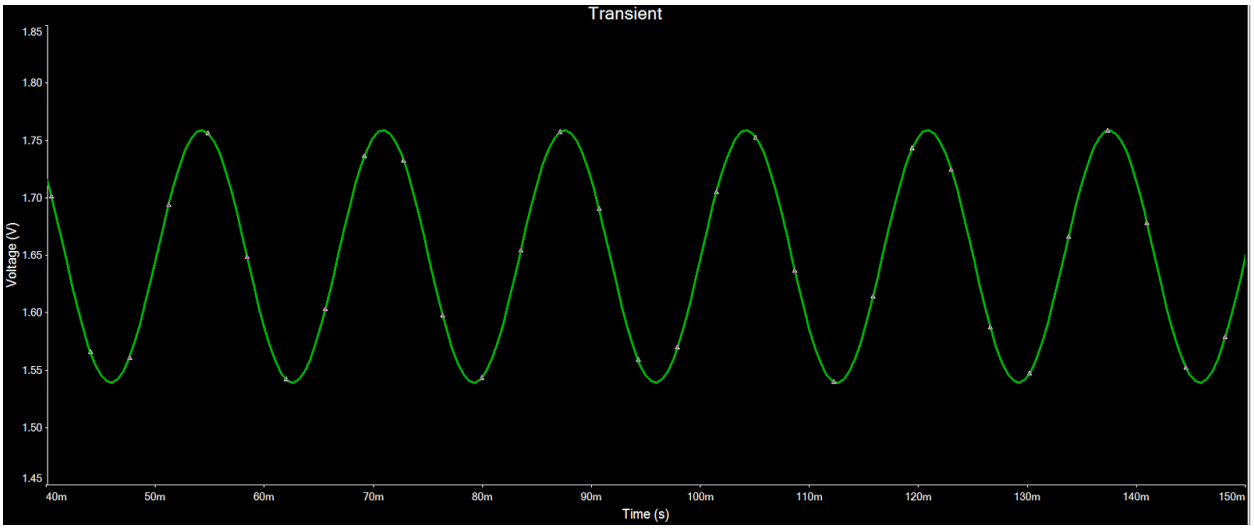


Figure 2: Transient Response of the Simulated Circuit

Testing Results

The remainder of this report will be the physical circuit and results measured from the oscilloscope.

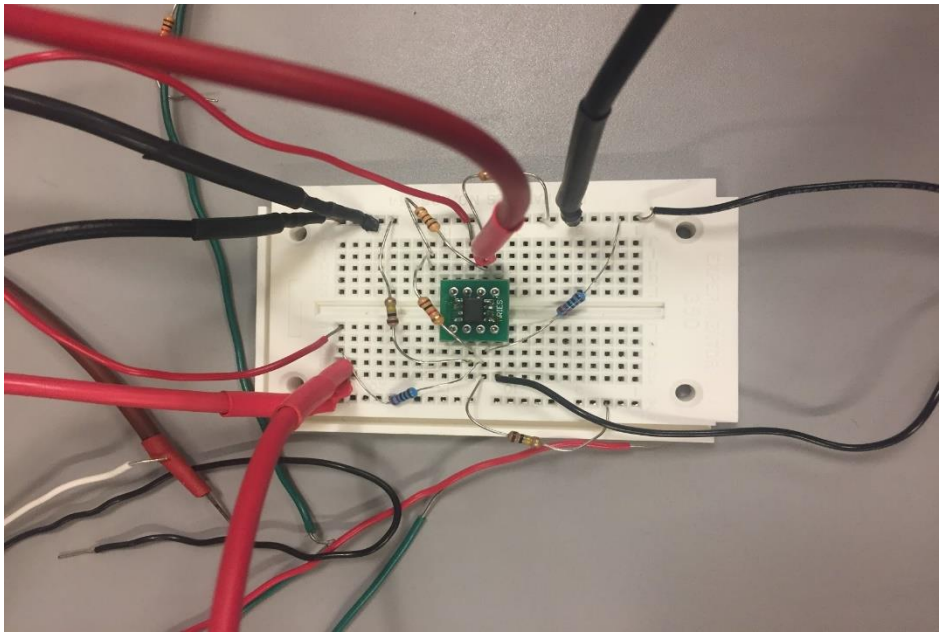


Figure 3: Voltage Measurement Sub-circuit

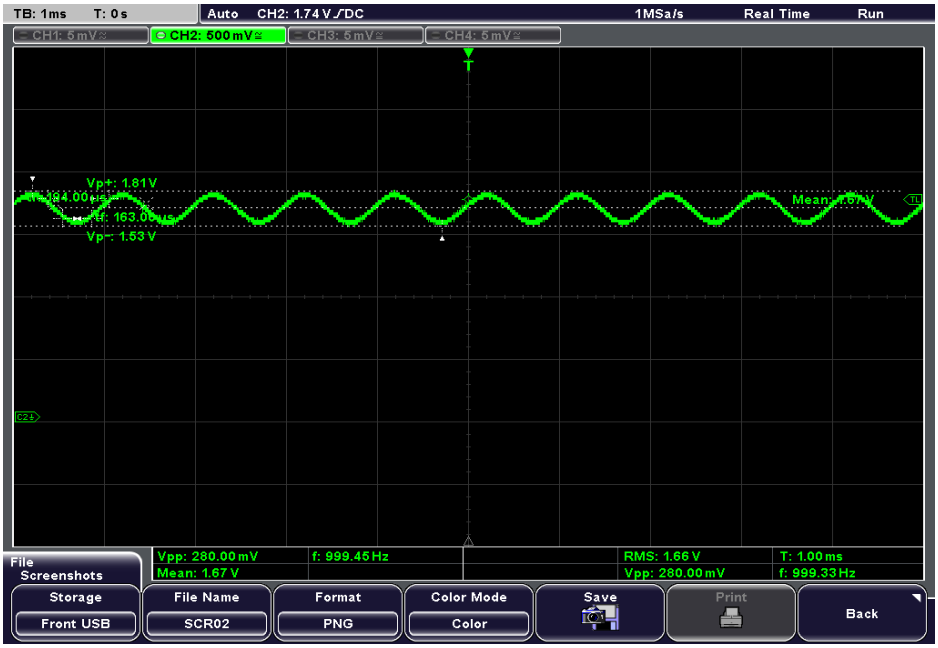


Figure 4: DC Coupled Output Waveform

From this waveform we can see that the mean value is 1.67 V this is only 0.02 V off from our simulated mean voltage. In addition, this verifies that our dc shifting solution for our clipping problem can reliably solve our problem.

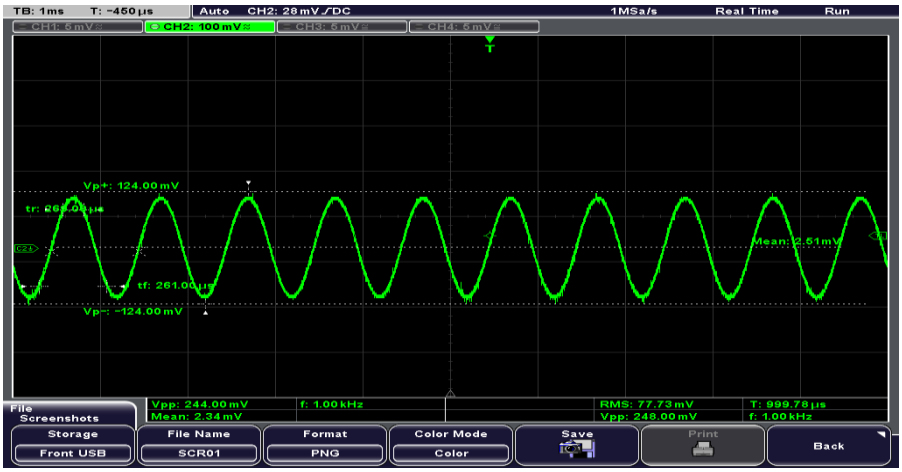


Figure 5: AC Coupled Output Waveform

Figure 4 shows us the AC coupled waveform. I choose to include this waveform in the report in order to have a better measurement for our peak to peak voltage. From the waveform we see a Vpp of 0.244 V. Comparing this to our simulation we see that our output is off by

0.024V and this difference is acceptable given the tolerances of resistors and the needs of our system.

Current Measurement System:

Introduction:

The idea behind this subsystem is that it will utilize a current transformer and current sense transformer to convert the current in a user's outlet into a voltage that we can measure and relate back to the original current.

Simulation:

Below is the differential op-amp portion of the current measurement sub-circuit with simulation results

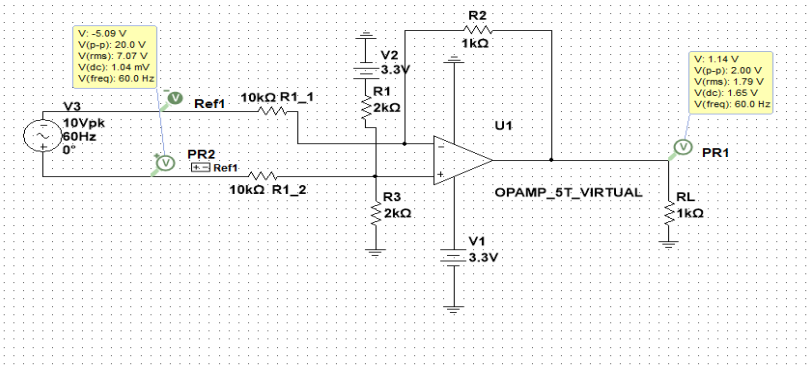


Figure 10: Schematic of Current OPAMP Sub-circuit

This schematic shows a the current OPAMP sub-circuit with a 10V pk-pk input resulting in a 2.00V pk-pk output. The input and resistor values are subject to change. In addition, this schematic has a DC voltage of 1.65 V at the output due to the DC shifting at the positive input terminal. The waveforms from this sub-circuit can be seen below.

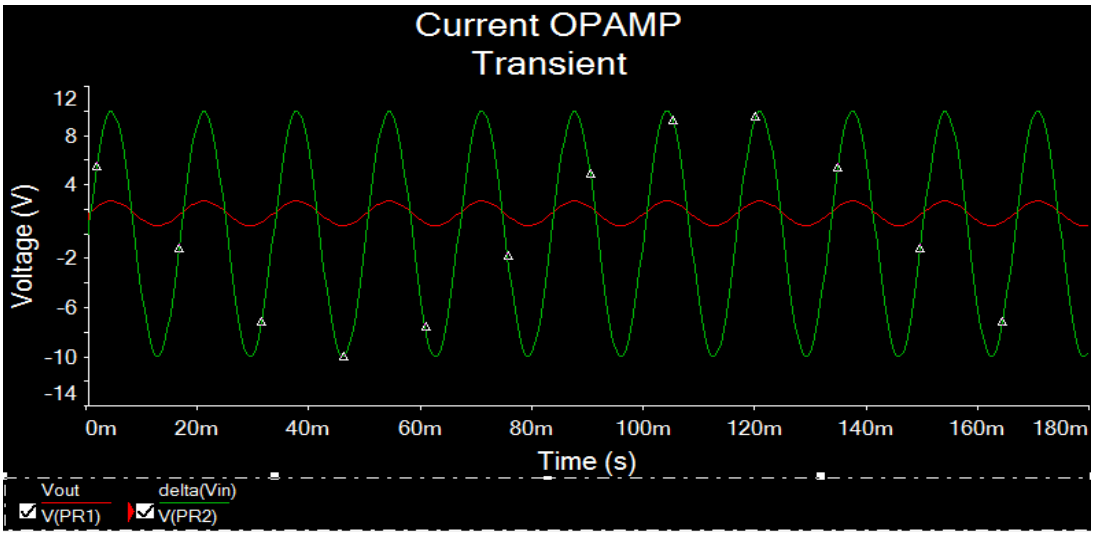


Figure 2: Transient Waveform of the Sub-circuit

Testing:

Testing for this subsystem included running a function generator through the current transformer and then placing a load at the end of the circuit. Once this was created we were able to establish a known current running through our transformer. This caused a current to run through the other side of the transformer and then a 1 Ω resistor was used to turn the current into voltage for the differential amplifer tested above. No screencaptures were taken during testing, but data points were in order to show the linearity of the system. This is important because if the system is linear it would allow us to create a resistor relationship at low voltage that would only require a multiplication to scale to high voltage testing. Below is the plot showing the linearity.

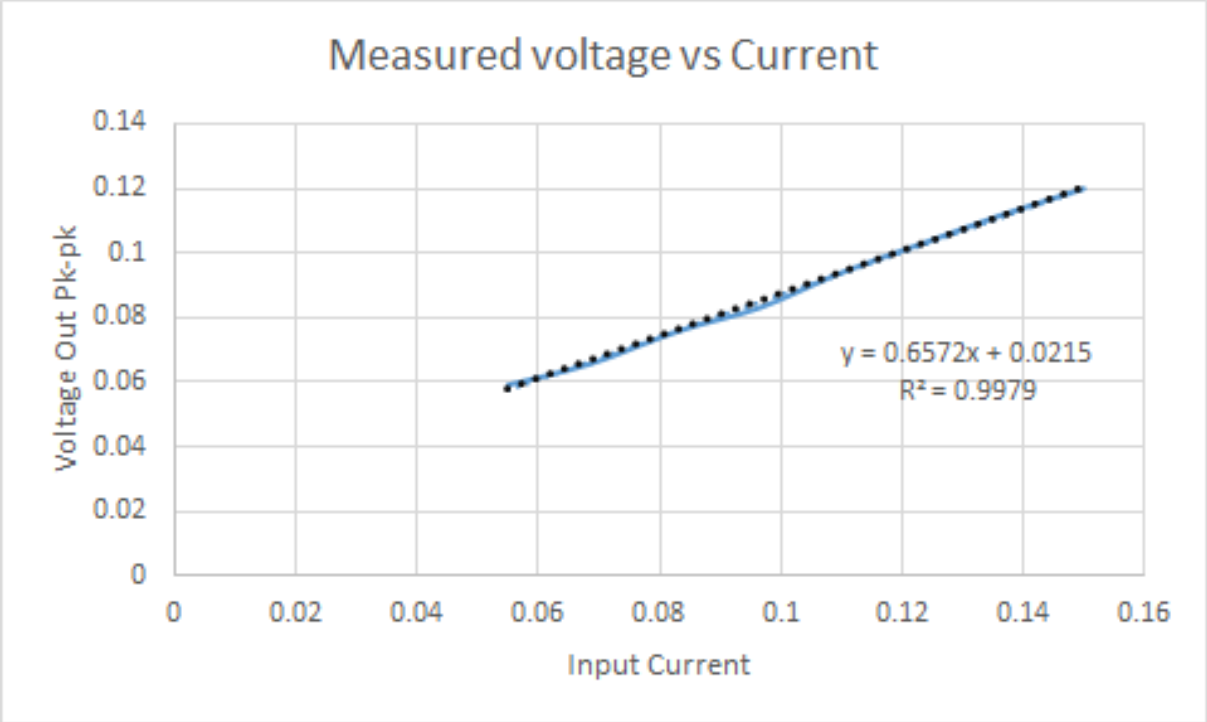


Figure : Linearity of Current Measurement Circuit

Relay System

The Mesh Network Power Analyzer requires a safety relay in order to disconnect a user's outlet from current when a fault or command is detected. This is done by using a dual coil latching relay. This relay was chosen due to its 16 A current rating (the max current a outlet can supply) and 250 Vrms voltage rating. The schematic is shown below.

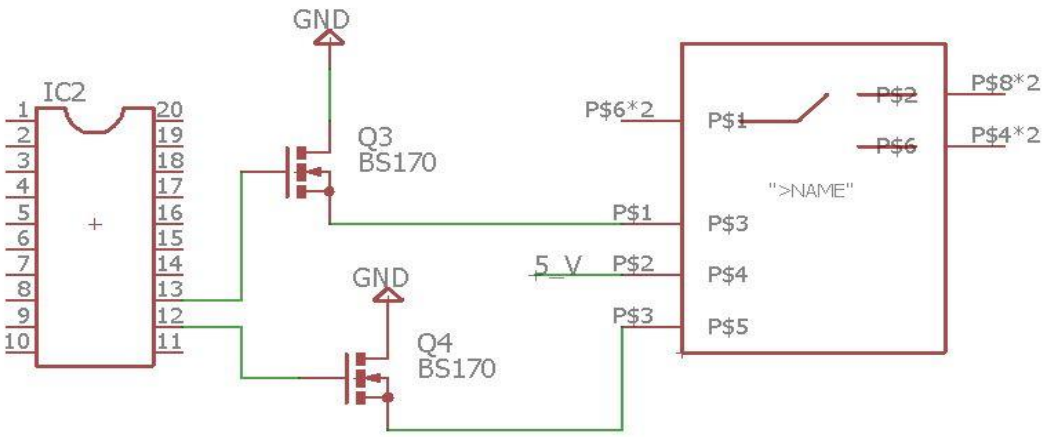


Figure 1: Relay Schematic

This circuit would take inputs from the MSP430 into the gate of the mosfet. This would allow the microcontroller to determine whether the P$3 or P$5 are low on the relay and when one of them goes low the relay latches to the according side and is "flipped".

Hardware Components:

In the next section of this report I will be outlining the components that are being used in each portion of the subsystem, what specifications they met, and how they connect to provide the necessary requirements for the subsystem. Below is a table outlining the components, what circuitry they will be apart of, as well as the area since that is a major constraint for the subsystem.

Voltage Measurement Sub-circuit	
Component	Size (mm^2)
Isolation Amplifier (ACPL-7900-500E)	62
TI OP07CDR OP AMP	84
Resistors	N/A
Current Measurement Sub-circuit	

Current Transformer (B82801C565A100)	288
TI OP07CDR OP AMP	84
Resistors	N/A
Relay Sub-circuit	
16A 2-Coil Latching Relay (RT314F05)	368
2 x 2N4007 Relay	N/A
Power System	
3.3V Coin Cell Battery Holder	617
2 x 3.3 V Coin Cell Battery	N/A (accounted for in the battery holder)
3.3V LDO (LP2985-33DBVR)	9
5V LDO (TPS76350DBVR)	9
Micro Controller	
TI MSP430	178
XBEE S2C 802.15.4	747
Total: 2,446 mm^2	

Figure 2: Components Table with Area

As you can see above the total area for most of the components is about 3,400 mm^2 less than the required PCB area. While this is definitely not the area of the PCB it still leaves about 58% of the total area to be utilized by the PCB.

The final PCB that was created can be seen below. The figure also includes the length and width of the board.

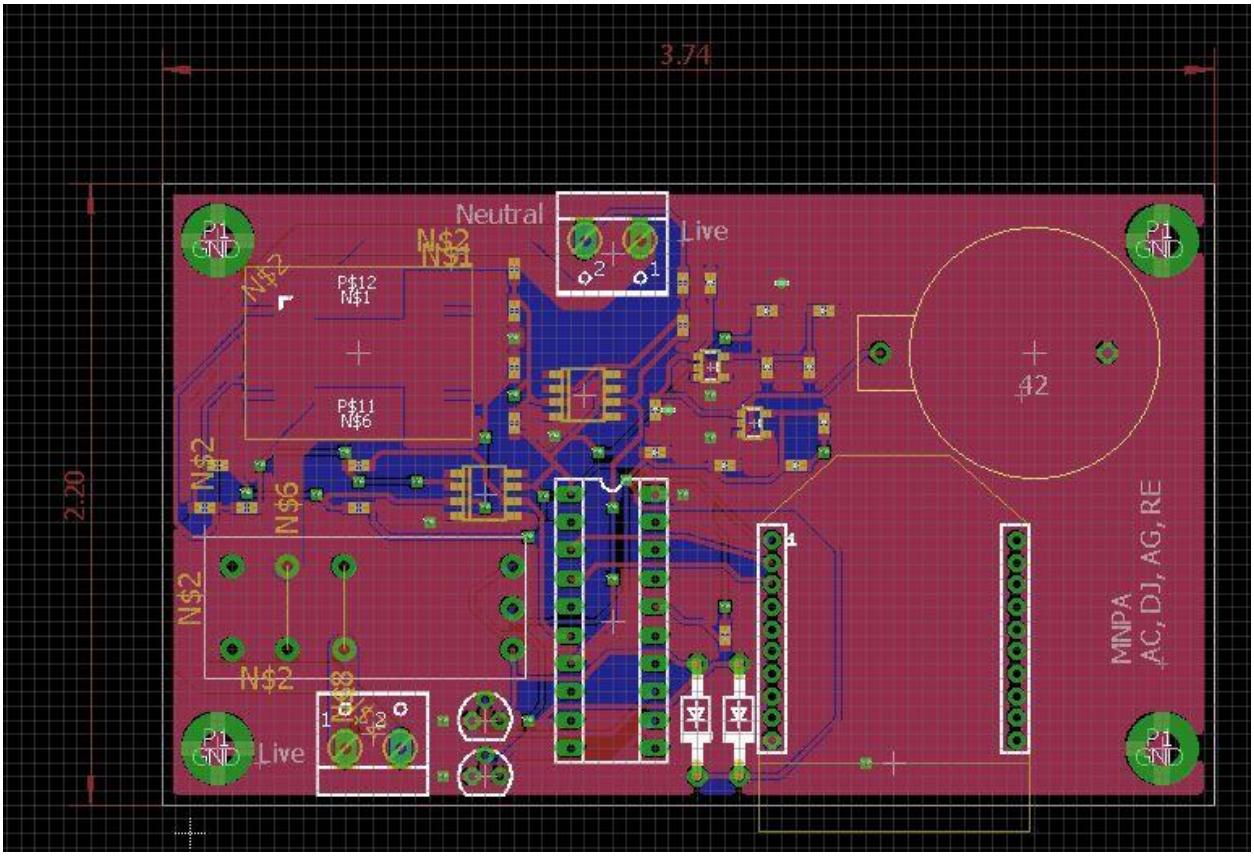


Figure 3: Board Layout

This PCB was very successful when completed and tested. The PCB was able to supply power, transmit data, and flip the relay from the microcontroller all on the PCB.

Physical testing of the PCB was difficult due to problems with both the PCB and supplier. A total of 3 PCBs were given to me. The first had routing and design errors that were discovered and initially expected. The second PCB was printed flipped and this lead to problems that were not obvious at first glance such as incorrect pad placements and routes when soldering components normally. The final and third PCB was from a manufacture and arrived one two days before the engineering showcase. My teammate Derek and I took turns soldering the PCB

board and then corrected any errors together. A final image of the subsystem can be seen below.

Figure 4: Completed Subsystem

Microcontroller Subsystem Writeup

05/03/2017

Alex Garcia

The microcontroller (MCU) subsystem is the interface between the hardware and the host computer software, the microcontroller is responsible from sampling the data from the stepped-down signals from the sensor subsystem and then reading these values, doing low-level calculations for RMS values, and making immediate emergency decisions to trigger the relay in the event of electrical failure. The MCU is responsible for using its serial hardware to transmit packed data to the host computer.

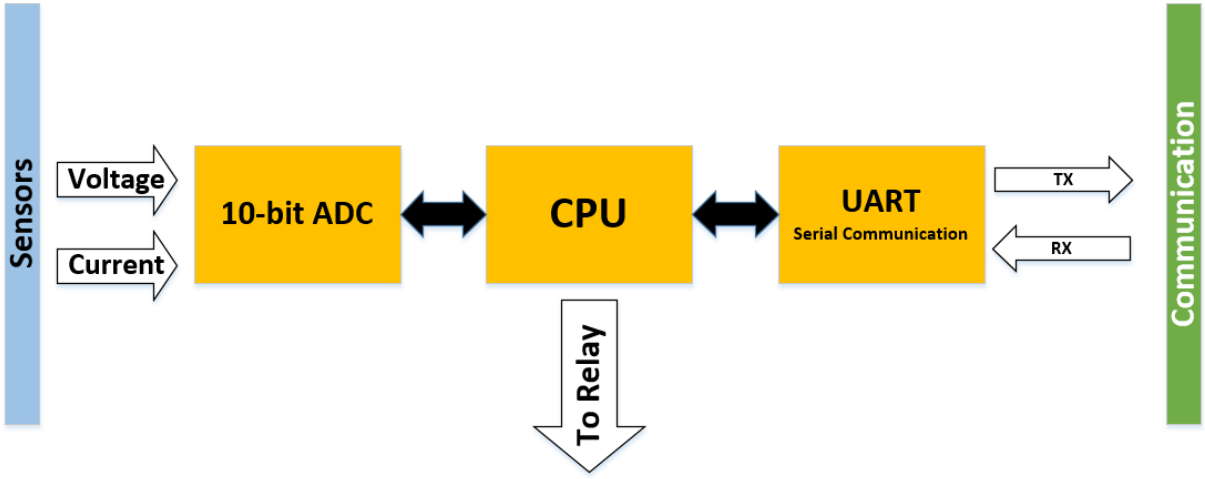


Figure1: MCU Subsystem Block Diagram

The modes of operation defined in the report generally are dependent on the mode of operation the MCU is in. The MSP430 allows for low-power mode of operation.

The fault scenarios the microcontroller needs to detect are:

- Short; Current is huge, voltage is low
- Over-signal; current or voltage signals are above threshold
- No signal; current and voltage signals are both zero
- Live circuit detection; If the ADC is seeing a voltage/current in the acceptable normal range (i.e. I<16A and V>140V) then return that the circuit is active

The user control also requires that the MSP430 must receive data via UART and transmit the data sampled. The system will have a 99% fault detection rate, and with the MSP430's ADC the error to less than 0.1% with a combination of code and analog sensor modifications. While the MCU is connected to the relay, there a current driver amp to protect the GPIO inputs of the microcontroller. The device will not exceed power consumption of 100 mW, and the MSP430 has power consumption rated at 230uA in active mode, and 0.5uA in standby mode. The device will use UART/SPI/I2C communication protocols, the MSP430 has UART hardware and the

capability to use SPI protocols. An ethical concern is the privacy of the user's data. This has been taken into account with the ZigBee communication protocol using encryption, and since there is a wired connection between the MCU and the XBee chip, there isn't a risk of data interception. The MSP430G2553 MCU was chosen because of its availability, extensive documentation/online resources, and low-power applications. The MSP430 satisfies all of the requirements, has bountiful support on the TI e2e community, and facilitates the formulation of the PCB because the dip-package can be easily mounted; instead of the MSP432's package and the required debugging hardware to get the chip programmed. This way we can program and debug the chip on the Launchpad and then transfer that same package to our own PCB. The 10-bit ADC was the main aspect of determining whether the MSP430 would be sufficient for the project. Generally, we are concerned with the average power consumption over the course of a month, and the extreme emergency cases, where the inherent error of the ADC wouldn't interfere with either of these uses. The full analysis of the error and plots compared to the ideal received signal are given in the following pages.

The firmware (code running on the MCU) performs the following tasks:

- Sample the voltage/current signals (system clock running at 1MHz)
- Transmit 40 samples (20 current and 20 voltage) for RMS every 30seconds
- Constantly monitor the signal for fault conditions
- Receive command from user

The final program was implemented using a combination of interrupt functions and polling. This way the MCU can conserve energy and be in a standby Low-power mode of operation when not transmitting data. Also since we are looking at average power consumption over the course of a month, then it is unnecessary to constantly transmit data to the host device, but it will be necessary to constantly monitor for fault conditions.

The validation plan was something that needs more thorough definition in order to adequately test and verify that the MCU subsystem works. While we listed the different tasks we would accomplish this semester in our execution plan, the concrete systematic testing of the components was never fully defined. This semester, basic functionality of the following features was demonstrated:

- UART data packet transmission of polled input signals
- Multi-pin ADC value comparison to threshold
- GPIO pin output to relay
- Receive interrupt that acts whenever anything is received

The issues last semester with printing data to a terminal was fixed through better implementation of the code and use of a different hyper-terminal. Also after system updates the MCU is able to successfully connect to the laptop a majority of the time. The testing for the received data will be implemented by transferring whatever is in the RX-buffer to the TX-buffer where it will transmit received data to the terminal. For reception, the interrupt would trigger if anything was received, so we decided that the only command would be "toggle", since the MSP430 couldn't determine

the difference between commands. The MCU sampled the 60Hz signals at 1205 Hz. This was well above the Nyquist frequency, but since there was still power information in as far as the 5th harmonic, this sampling frequency was necessary. With this frequency, it also allowed for 20 samples per period, which increased the resolution of the RMS values.

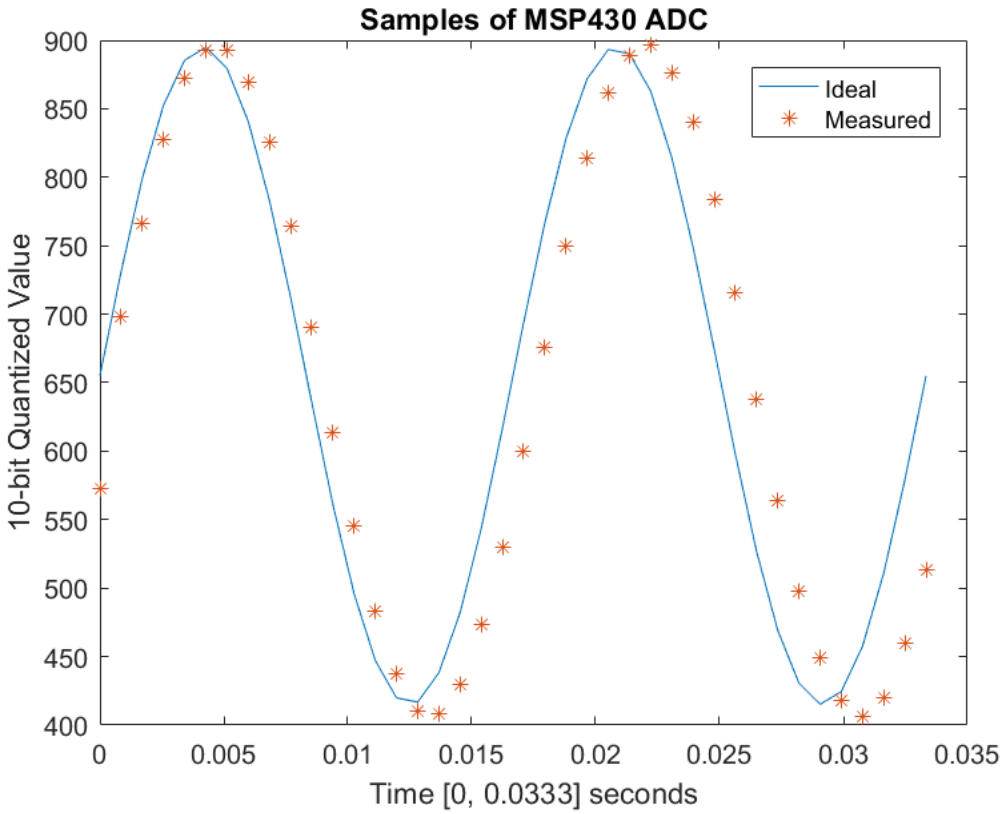
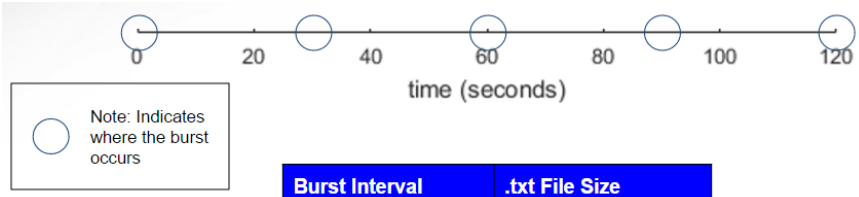


Figure 2: Sampled Signal

In order to reduce the storage necessary for a large system of nodes, the node was set to transmit every 30 seconds rather than continuously. While the MCU was constantly monitoring the signal for faults, it transmitted data for RMS calculations on a much longer interval. Again, since the system is concerned with the power consumption over time, 30 seconds is sufficient, since the power consumption change within 30 seconds wouldn't be significant.



Burst Interval	.txt File Size (single node/month)
1 second	100.2 MB
10 seconds	10.02 MB
30 seconds	3.312 MB
60 seconds	1.635 MB

Figure 3: Burst-Sample Interval

Overall, the microcontroller system was able to packet and transmit current and voltage values that were transmitted every 30 seconds. Additionally, the software would set the GPIO outputs high when anything was present at the receive buffer. There were issues with the relay hardware and software, so the fault prevention capabilities weren't working for the final demo and first showcase. However, the fault cases of triggering of the relay with the driver hardware were corrected in the lab over the weekend, but these were never fully integrated in the main firmware.

Communications Subsystem Writeup

05/03/2017

Derek Janak

The communications subsystem is comprised of the wireless mesh network and the serial communication links with the microprocessor and with the computer. The purpose of this subsystem is to allow data transmission from distributed nodes back to the hub computer, and transmission of various command instructions from the hub computer to individual nodes. The network must be fast enough that data samples being relayed by the various nodes do not saturate it, and reliable enough that it can continue to operate in the event of failures at the level of isolated nodes. The FSR defines the following subsystem requirements:

- Minimum communication range of at least 20 meters
- Network capacity of at least 10 nodes
- Use of an established wireless standard

The communications subsystem diagram is shown below.

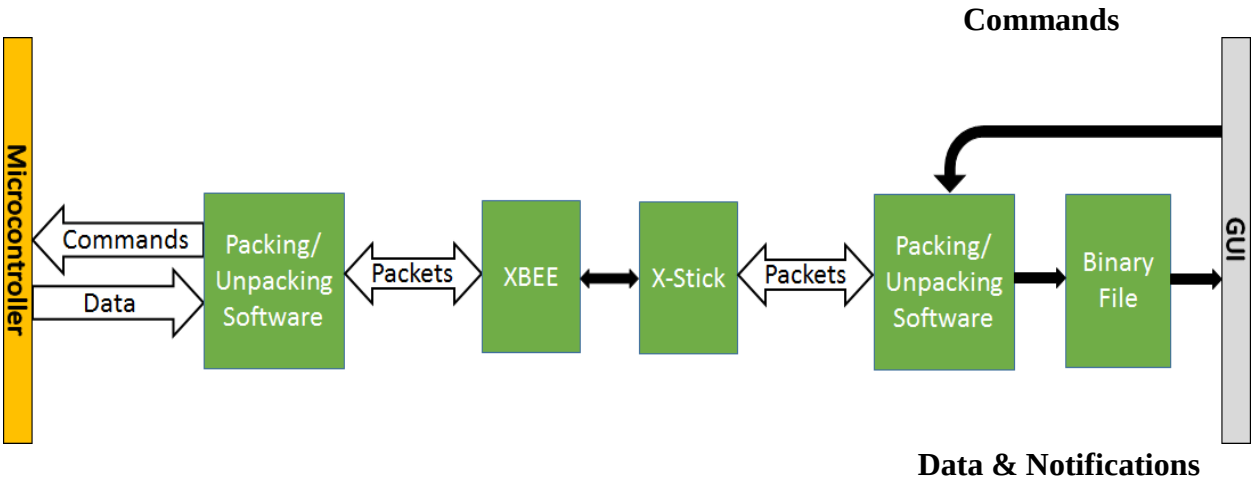


Figure 1: Communications Subsystem Block Diagram

The work done in the design and development of this subsystem can be grouped into several distinct categories, namely hardware selection, wireless troubleshooting and configuration, and software development.

The first portion of the communications design process required the evaluation of various wireless communication protocols and technologies. Standard protocols such as Bluetooth, Wi-Fi, and ZigBee were reviewed and compared. Ultimately, the decision was made to utilize the IEEE Standard 802.15.4 wireless communication protocol. This standard, which operates at 2.4 GHz and is similar to the commercial ZigBee standard, was selected based on ease of use and level of support. The Digi XBEE S2C device was chosen for use in the distributed nodes, while the Digi X-Stick was selected to interface with the hub computer via the local USB ports. Following

review of the pertinent features of each device, the devices were purchased and research was done to identify key signal pins on the XBEE for use in hardware integration. A list of potentially relevant pins was identified, as shown below.

Pin #	Name	Direction	Default state	Description
1	VCC	-	-	Power supply
2	DOUT/DIO13	Both	Output	UART data out
3	DIN/CONFIG / DIO14	Both	Input	UART data in
5	RESET	Input	Input	Module reset
6	RSSI PWM/PWMO DIO10	Both	Output	RX signal strength indicator/GPIO
9	DTR/SLEEP_RQ/ DIO8	Both	Input	Pin sleep control line/GPIO
10	GND	-	-	Ground
12	CTS/DIO7	Both	Output	Clear-to-send flow control/GPIO
13	ON_SLEEP/DIO9	Both	Output	Device status indicator/GPIO
15	ASSOCIATE/DIO5	Both	Output	Associate indicator/GPIO
16	RTS/DIO6	Both	Input	Request to send flow control/ GPIO

Table 1: XBEE Pinout

In evaluating the ability of this hardware to meet the demands set forth in the FSR, it was verified that the devices were able to support a distributed network of more than 10 devices. It was determined that the line of sight range of the devices extended well beyond the required 70 meters, but that this range might drop below the required threshold in the absence of a clear line of sight. Therefore, it was determined that routing between nodes might be employed to extend the signal range.

Following receipt of the devices, experimentation began on configuring the two devices to communicate with each other. The proprietary Digi XCTU software was installed on a test computer, and the various characteristics of the software were examined. The software was used to detect the X-Stick, and the firmware of the X-Stick was updated to the newest version. In the

absence of the debugger typically used to program the XBEE chip, an Arduino Uno was repurposed to interface the device to the computer. This was done by wiring the power and ground pins of the XBEE device to the power and ground pins of the Uno and tying the UART TX and RX pins of the XBEE to the communication pins of the Uno. A dummy program consisting of an empty infinite loop was programmed to the Uno in order to prevent it from responding to serial communications from the USB port. This allowed the Uno to act as an adapter for the XBEE by passing through the required signals from the USB port of the computer. The XCTU software successfully recognized the presence of the XBEE, and the firmware of the XBEE was updated. The pan ID's of both devices were set equal to each other to facilitate their communication, and troubleshooting began. Both the X-Stick and the XBEE were connected to separate serial ports of the computer. Using the XCTU software, it was verified that a packet could be sent through one device and received via the other device. Using Python 3 code and an online serial library, a temporary program was developed and used to verify that the communication link was accessible by reading and writing data to and from the computer's serial ports.

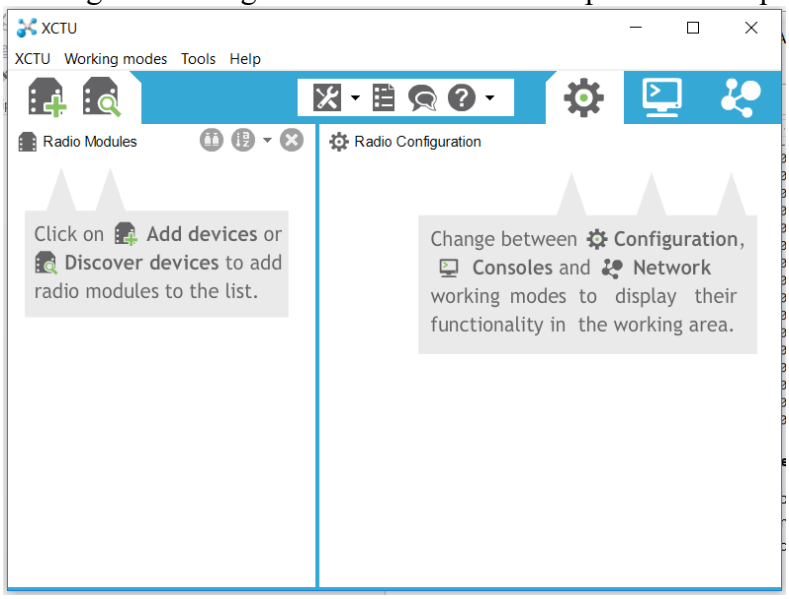


Figure 2: XBee XCTU Software

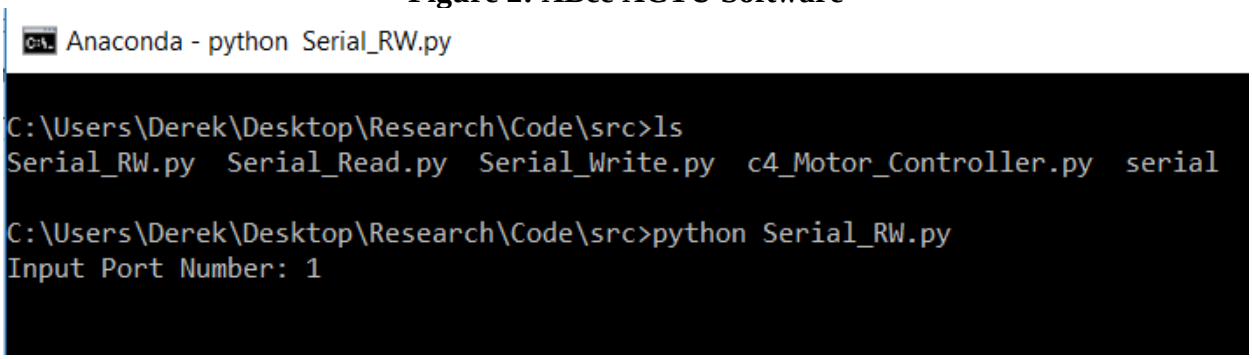


Figure 3: Python Serial Sending Program

With the method of interface between the computer and the wireless mesh verified, the subsystem design moved into the software development phase. In planning the structure of the communication software, it was necessary to both accommodate current design parameters and to plan for flexibility in the face of potential unforeseen challenges. The decision was made to utilize the XBEE AT mode for the proof of concept; however, it was recognized that this mode might not be capable of supporting router nodes and might be inefficient for a network with a large number of end nodes. It was determined that the more robust but less transparent API mode might be required in the event that AT mode proved insufficient. This fact was taken into account early into the software design process.

Initial software design consisted of designing a packet structure for use in communication. The packet structure chosen was intentionally chosen to be similar to the standard API mode packet structure to facilitate transition to API mode if necessary. The initial packet structure chosen was as follows:

- Start Delimiter (0xFF) – 1 byte
- Length of packet – 2 bytes
- Node ID – 1 byte
- Packet type (Command, Notification, or Data) – 1 byte
- Payload – typically 4 bytes
- Checksum – 1 byte

Additionally, provisions were made to incorporate escaping of reserved bytes using the escape delimiter 0xEE.

In translating this packet structure to code, the C language was chosen to facilitate compatibility with both the C code used by the microcontroller and the C++ code used on the hub computer. First, a library was written with a wrapper struct, Array, and various accompanying functions; this struct was used to contain the arrays of bytes used to send information. Next, a C file and accompanying header were written with all functions necessary to pack and unpack data. A simple test program was used to verify the functionality of these libraries by packing and unpacking various sets of test data. This code was provided for use in data handling by the microcontroller.

```
    Directory: C:\Users\Derek\Desktop\Fall_2016\Senior_Design\C_Code\all_files


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/8/2017   9:41 PM          59266 a.exe
-a----        11/6/2016  11:59 PM           1828 arrays.c
-a----        11/6/2016  11:58 PM            956 arrays.h
-a----       11/21/2016   6:06 PM        1188284 arrays.h.gch
-a----       12/29/2016   1:29 PM           3849 AT_packet.c
-a----       12/29/2016   1:14 PM           1469 AT_packet.h
-a----       11/21/2016   6:06 PM        1196500 AT_packet.h.gch
-a----        11/6/2016   7:23 PM            376 TestArray.c


PS C:\Users\Derek\Desktop\Fall_2016\Senior_Design\C_Code\all_files> gcc -std=c11 *.c
PS C:\Users\Derek\Desktop\Fall_2016\Senior_Design\C_Code\all_files> ./a.exe
Array Values: 0x01, 0x0A, 0xFF
Bool: 1
Array Values: 0x01, 0x0A, 0xFF, 0x10
Bool: 1
Array Values: 0x01, 0x0A, 0x10
PS C:\Users\Derek\Desktop\Fall_2016\Senior_Design\C_Code\all_files>
```

Figure 4: Array Library Test

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----       12/29/2016   1:29 PM          59266 a.exe
-a----        11/6/2016  11:59 PM           1828 arrays.c
-a----        11/6/2016  11:58 PM            956 arrays.h
-a----       11/21/2016   6:06 PM        1188284 arrays.h.gch
-a----       12/29/2016   1:29 PM           3849 AT_packet.c
-a----       12/29/2016   1:14 PM           1469 AT_packet.h
-a----       11/21/2016   6:06 PM        1196500 AT_packet.h.gch
-a----        11/6/2016  11:48 PM            720 TestATPacket.c


PS C:\Users\Derek\Desktop\Fall_2016\Senior_Design\C_Code\all_files> gcc -std=c11 *.c
PS C:\Users\Derek\Desktop\Fall_2016\Senior_Design\C_Code\all_files> ./a.exe

Test Primitive Array Length Macro
Size of array: 6

Test Pack Functions

Array Values: 0x10, 0x01, 0xFF, 0x13, 0xEE, 0x11
Packet Output from PackData: Array Values: 0x7E, 0x00, 0x08, 0x01, 0x01, 0x10, 0x01, 0xFF, 0x7D, 0x33, 0xEE, 0x7D, 0x31,
 0xDB

Test Unpack Functions

Payload from UnpackData:
Array Values: 0x10, 0x01, 0xFF, 0x13, 0xEE, 0x11

GetNodeID and GetPacketType:
NodeID: 0x01  Packet Type: 0x01

PS C:\Users\Derek\Desktop\Fall_2016\Senior_Design\C_Code\all_files>
```

Figure 5: AT Packet Library Test

The next phase of development consisted of developing C++ code to replace the Python code previously written for serial communications on the hub computer. C++ was chosen over Python due to the ability to precompile C++ code into an executable. The C++ code necessary to communicate with the network was developed using Microsoft Visual Studios and the CSerial open source library, and the code was compiled into executable files. Finally, sample code was written to monitor the serial port and store any data received from the network within a binary file for retrieval and processing by the GUI. This code was written to include the data, node ID, and the timestamp of when the data was received. The code was also structured to sort data into

separate log files based on the month in which the data was taken. This code allowed the interfacing of the MCU with the Communications network and the GUI.

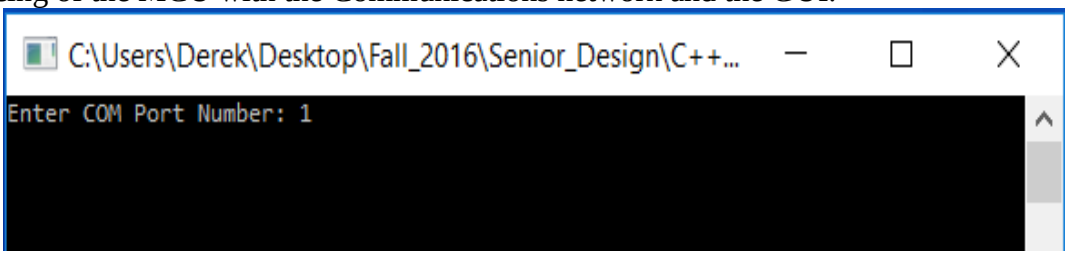


Figure 6: C++ Serial Send/Receive Program

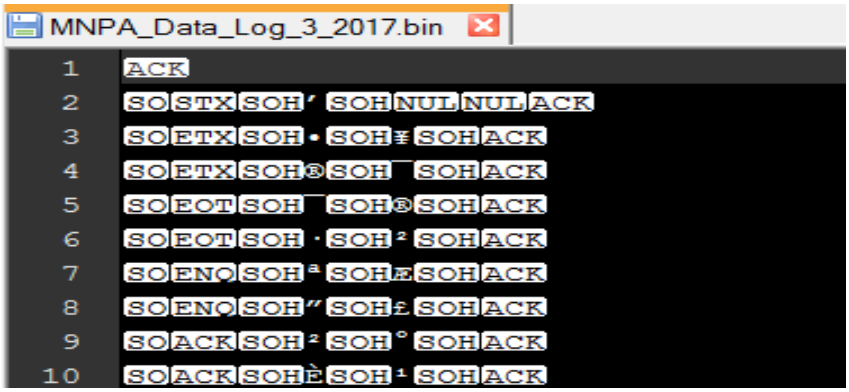


Figure 7: Serial Data Log File

The serial logging program was used to create sample log files for use in testing the GUI. Data for the log files was generated by the microcontroller and was transmitted via the wireless network. Testing of the GUI using these data files is ongoing.

Investigation of the API mode of XBee communications was undertaken. In order to more easily understand API based communications, the XCTU software was used to produce a log file detailing the API serial communication packets necessary for remote node discovery and configuration of nodal parameters. Various types of API packets were identified which corresponded to querying/setting local device parameters, querying/setting remote device parameters, and communicating with remote devices. The sequence of packets necessary to establish a wireless network was investigated and stored in various log files. One of the log files generated is shown below.

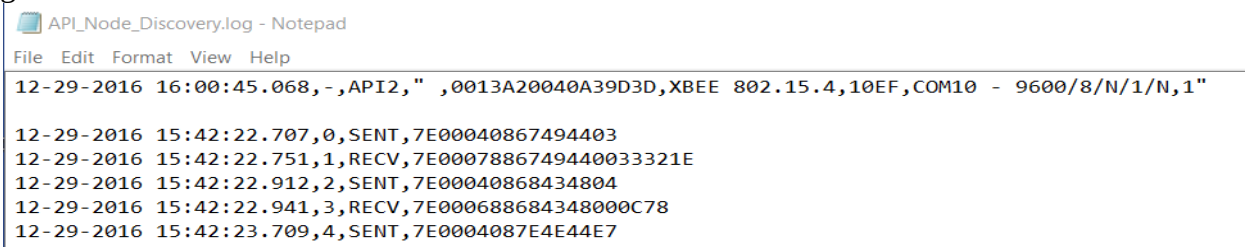


Figure 8: XCTU Node Discovery Log File

The AT packet based library was converted to an API based library in order to increase the robustness of the network and to mitigate crosstalk between various devices. The main packet structure utilized was the XBee API 64-bit addressing format, which consists of the following:

To Send from the Node to the Hub

- Start Delimiter - 1 byte (0xFE)
- Length - 2 bytes
- API Command Code - 1 byte (TX Request: 0x10)
- Frame ID - 1 byte (0x00)
- Node MAC - 8 bytes (0x00 00 00 00 00 00 00 00)
- Generic Local Address - 2 bytes (0xFF FE
- Congiguration Information – 2 bytes (0x00 00)
- Packet Type - 1 byte
- Data - 4 bytes
- Checksum - 1 byte
-

To Send from the Hub to the Node

- Start Delimiter - 1 byte (0xFE)
- Length - 2 bytes
- API Command Code - 1 byte (TX Request: 0x00)
- Frame ID - 1 byte (0x00)
- Node MAC - 8 bytes
- Congiguration Information – 1 byte (0x00)
- Packet Type - 1 byte
- Command - 1 byte
- Checksum - 1 byte

Additionally, the XBee API standard escaping sequence was implemented. This required escaping of bytes 0x11, 0x13, 0xFE, and 0x7D by XOR'ing them with 0x20 and subtracting the result from 0xFF, then prefixing the output with 0x7D. An example calculation to escape 0x11 is shown below:

0xFF-0x11^0x20=0xFF-0x31=0xCE

Result:0x7D 0xCE

Utilizing this packet structure, robust dataflow was achieved from the nodes to the central hub. Testing with two distinct nodes showed only a single packet (19 bytes) dropped out of over 17 KB. Additionally, tests confirmed that the communications code was successfully able to separate error notifications from the nodes from data for storage in a distinct binary file, as evidenced by the creation of two separate log files.

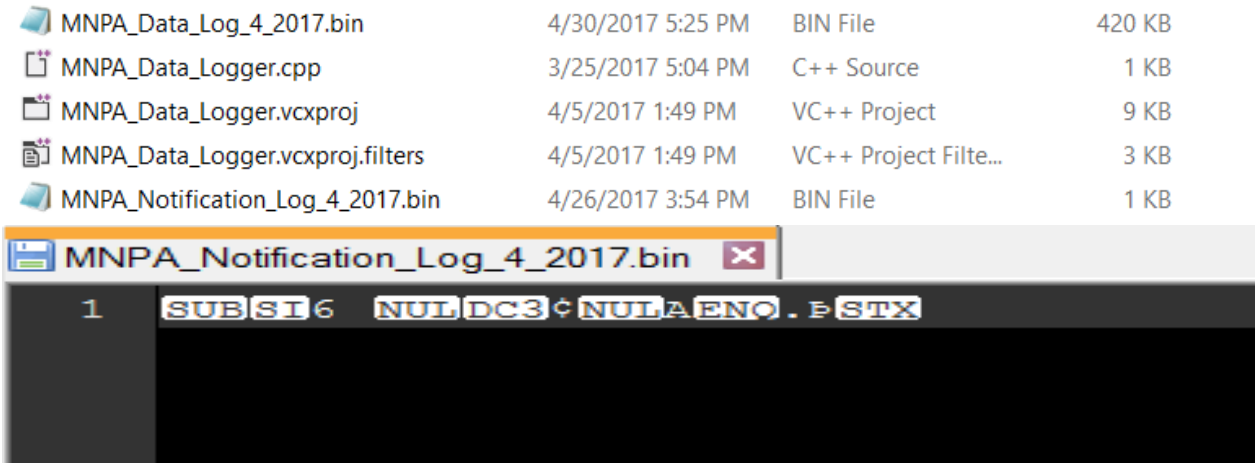


Figure 8: Notification Log File

Commands were also implemented in which signals from the GUI were sent out from the hub to individual nodes in order to switch the node relays on or off manually. Testing showed the commands subroutine to be functioning correctly, with audio/visual confirmation of the switching of the relay upon issuance of a command.

Full system integration was achieved with both the microcontroller and GUI subsystems, allowing for low-latency, bidirectional transfer of data, commands, and notifications. All verifiable subsystem requirements were met as outlined in the FSR.

Flow Charts of Communications Software

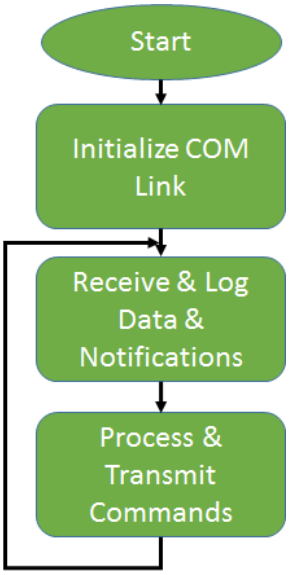


Figure 9: Overall System Flow Chart

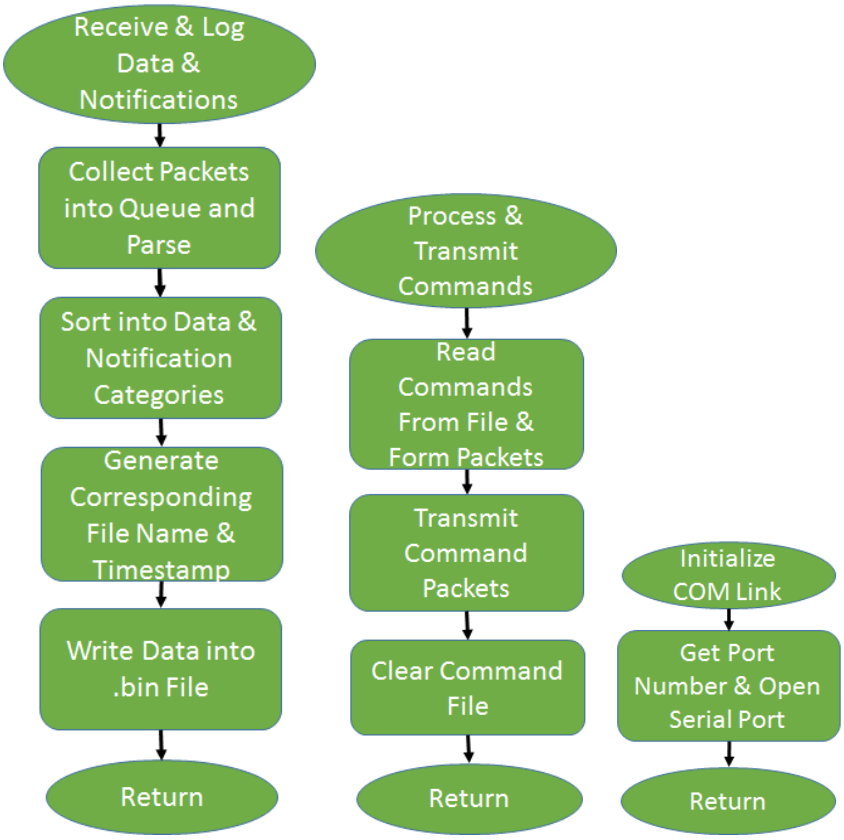


Figure 10: Flowcharts of Subroutine Logic

Further Contributions

Beyond the original scope of the communications subsystem, additional tasks were completed in cooperation with other teammates. For the hardware subsystem, a casing was designed for the node via Autodesk and was produced via 3D printing. Additionally, soldering of surface mount components and extensive troubleshooting/repairs of the PCB (printed circuit board) were performed throughout development of the PCB, both individually and collaboratively. Both PCB and casing are shown below.

Figure 11: PCB and Casing

A display was also developed to showcase the final node demonstration. The display board is pictured below.

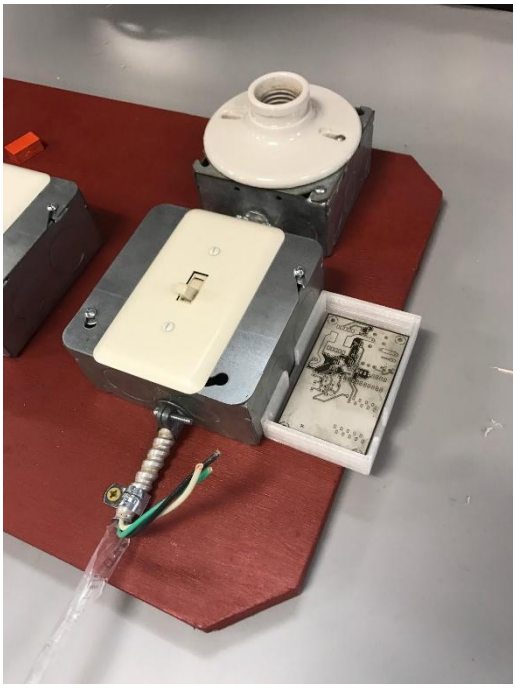


Figure 12: Test Fixture with Dummy Node

Working in conjunction with the microcontroller subsystem, code was developed to implement RMS processing of data within the communications code on the hub computer. This code was validated and was fully incorporated into the communications subsystem.

```
PS C:\Users\Derek> cd Desktop/Fall_2016/Senior_Design/C++_Code/API
PS C:\Users\Derek\Desktop\Fall_2016\Senior_Design\C++_Code\API> g++ -std=c++11 progress.cpp
PS C:\Users\Derek\Desktop\Fall_2016\Senior_Design\C++_Code\API> ./a.exe
Vector Values: 0x00, 0x13, 0xA2, 0x00, 0x40, 0xD7, 0xB7, 0xBB, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x0
1, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x02, 0x03, 0x02, 0x03, 0x02, 0x03, 0x02, 0x03, 0x02, 0x0
3, 0x02, 0x03, 0x02, 0x03, 0x02, 0x03, 0x02, 0x03, 0x02, 0x03

RMS: 257, 515
Vector Values: 0x00, 0x13, 0xA2, 0x00, 0x40, 0xD7, 0xB7, 0xBB, 0x01, 0x01, 0x02, 0x03
PS C:\Users\Derek\Desktop\Fall_2016\Senior_Design\C++_Code\API>
```

Figure 13: Console Output of RMS Function Validation

Additional work was done on the GUI subsystem. In order to aid in troubleshooting and to provide backup code in the event of difficulty, a mock GUI was implemented. The GUI software reached a level of basic functionality with the ability to plot three data types (current, voltage, and power) over four timespan options (month, week, day, and hour) for multiple nodes. The functionality of the plotting code was verified via sample data files.

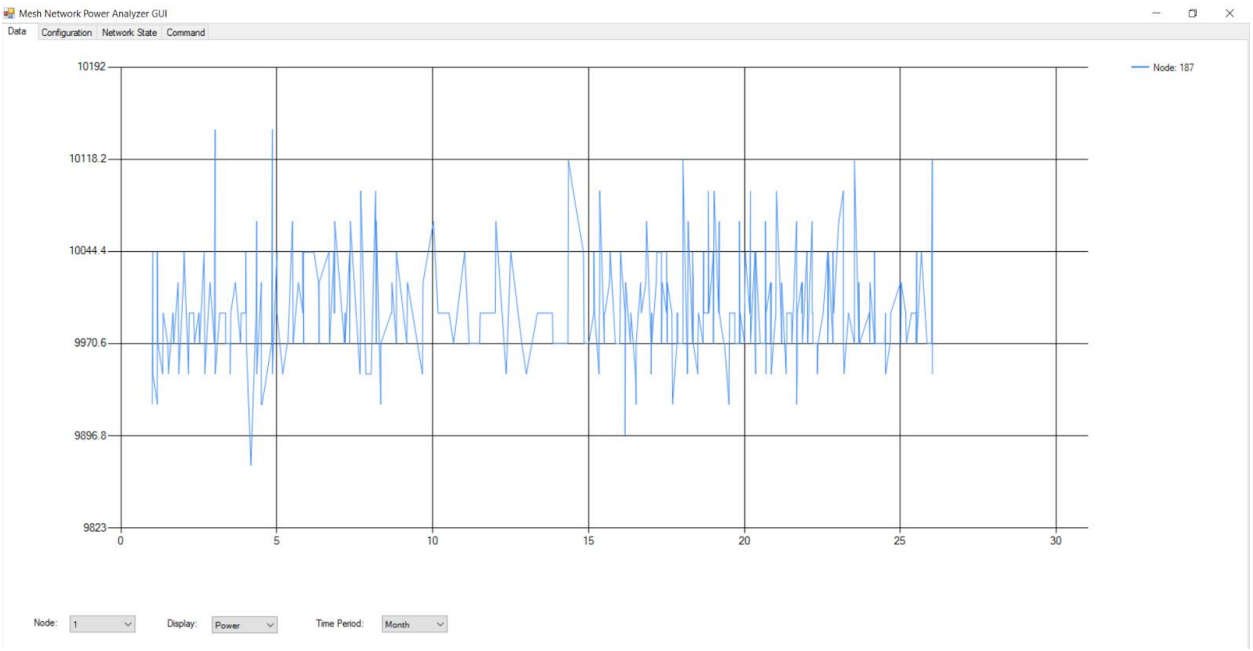


Figure 14: Sample GUI Displaying Power Consumption

Other contributions included coordination of team meetings, strategic planning, and algorithmic development.

Software Subsystem Writeup
05/03/2017
Ryker Elkins

The software subsystem is responsible for configuring the nodes, reading the binary files from the communications subsystem, translating the file to database, and presenting the database in a GUI. This is the 1000-mile view, but creating a good GUI has proven to be a challenge. The software is required to store at least one month of data, and be able to run on at least a Windows 8 system. The software will be completely stored on the user's computer of choice, which will be the same computer that the Xigby Hub is connected too. The only other software installed on the user's computer, which is not encompassed by this subsystem, is the communication software. The communication software will be reading the data packets from the nodes, and creating the binary data files that this subsystem interprets. The Software Subsystem will only be running when the GUI is open, and will not display real-time data.

The software is comprised of several key components: The Database and Measurement classes, the Main Navigation page of the GUI, the Node Configuration page, the Error and Command Log pages, the Input Commands page, and all the functions that comprise and tie together the previous components.

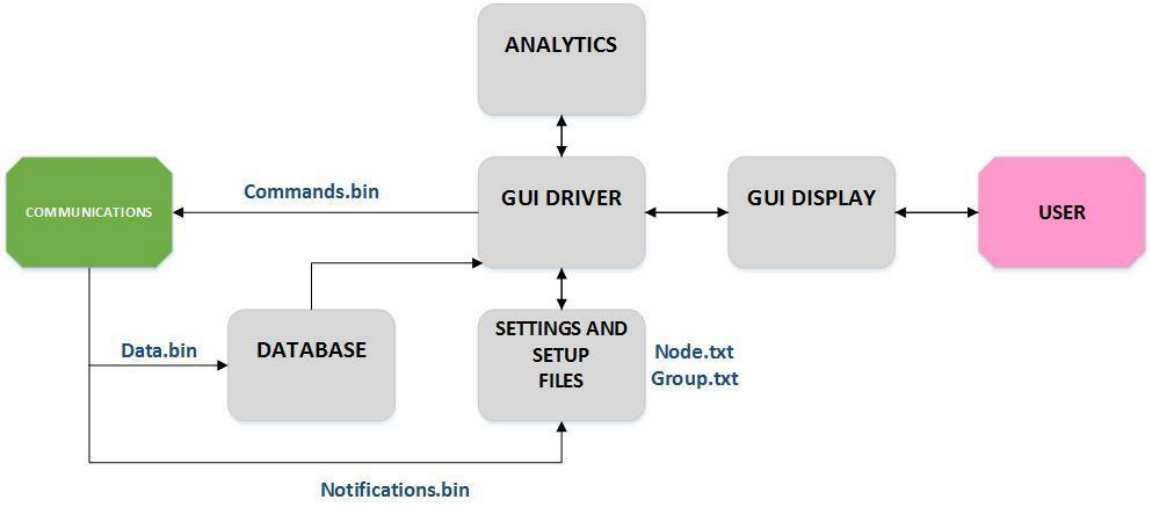


Figure 1: Software Subsystem Diagram

Start-Up:

When the software is booted up, it goes through a startup sequence, which uses the current local time to generate the binary file name from the communication block. The file name format contains the current year and month. This formats are as follows:

MNPA_Data_Log_MM_YYYY.bin (ex. MNPA_Data_Log_12_2016.txt, for December 2016) for current and voltage data, and MNPA_Notifications_MM_YYYY.bin for errors and commands. A database of measurements, and notifications are then created using the binary file and a binary parsing function.

Parsing the binary file relies on the following measurement and notification packet formats:

Variable	Day	Hour	Minute	Second	Node Address	Current	Voltage
Bit Size	8	8	8	8	64	16	16

Variable	Day	Hour	Minute	Second	Node Address	Notification Type
Bit Size	8	8	8	8	64	8

Where the time variables denote the time the packet was received by the communications software, and the current and voltage are RMS values. The node address is the MAC Address of the source node. Notification type determines what kind of error or command the node has experienced. The binary parser reads a packet at a time and creates new measurement objects with the data. The new measurement object is then put into the database.

The database is a vector of measurements. It holds all of the measurements for the current month. Because each database only covers one month, the database class can print the database to a .txt file which can be accessed outside of its inherent time frame. This .txt file is also much easier to read than the binary file, and contains all the information needed to convert it back to a database.

```
1       23:0:0          2       3.1596          121.8           384.85
1       23:0:0          3       8.8242          120.34          1061.9
1       23:30:0         1       6.9764          119.39          832.89
1       23:30:0         2       4.8076          117             562.47
1       23:30:0         3       6.106           116.38          710.62
2       0:0:0           1       5.2145          118.82          619.6
2       0:0:0           2       2.6077          122.62          319.77
2       0:0:0           3       3.9193          115.4           452.31
```

Figure 2: Database .txt file (day, hour:min:sec, node ID, RMS current, voltage and power)

After initializing and filling the database from the binary file, the main navigation page will display. At this point the program is ready for user input. The other windows, will only display one of themselves at a time, no matter how many times their respective buttons are clicked, and can be re-opened multiple times after closing.

The Main Navigation Page:

The main navigation page is the main window of the GUI, and therefore of the entire software subsystem. It is tied to the main() function of the software, so that if the main navigation page is closed, then all of the software closes as well. This page contains buttons that open the other windows of the GUI in separate new forms, but remains open when the other windows are displayed.

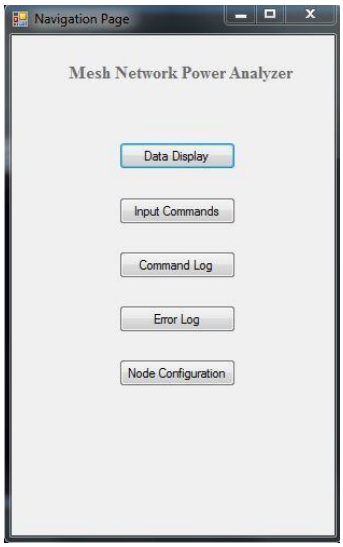


Figure 3: Main Navigation Page

Node Configuration Page:

The Node Configuration page allows the user to name and group nodes to their liking, and in the process, it will gather the MAC addresses of the nodes in the system for communication purposes. The page consists of a few groups of list boxes, text boxes and buttons, which allow the user add and remove nodes and groups by filling their respective text boxes, and pressing the respective button. Upon saving, it will write to two .txt files; node.txt, with the format (on/off)(1/0)Status, MAC Address, Node Name, new line, and group.txt with the format Group Name, Node ID List, new line. These files are tied to the Data Display node setting for ease of node selection, and are also used throughout the software when needed.

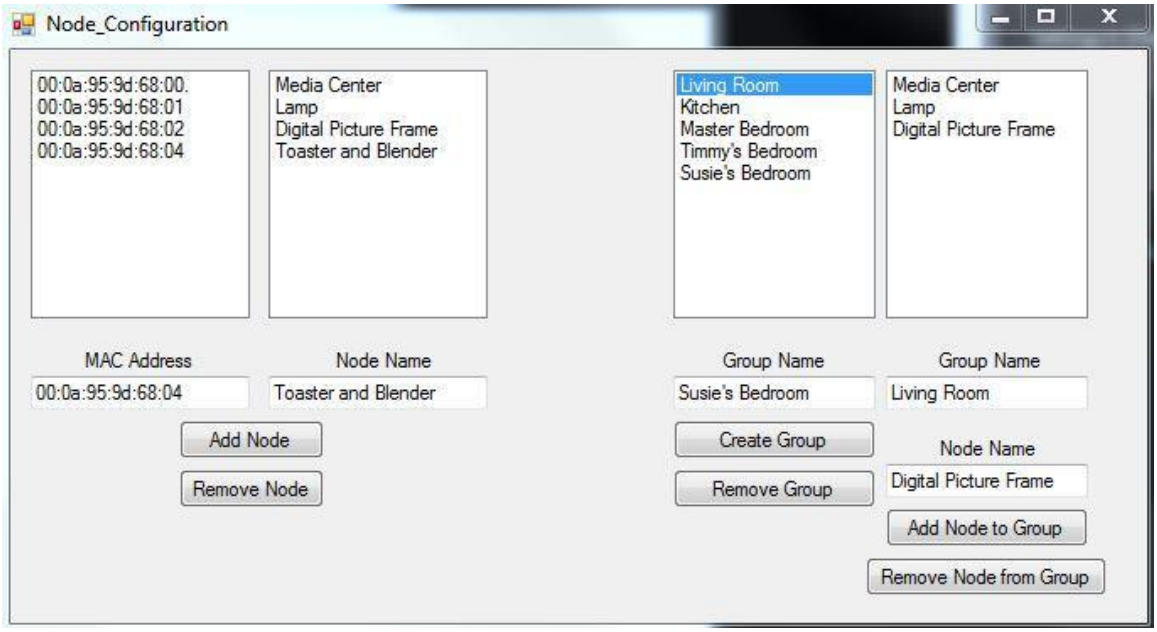


Figure 4: Node Configuration Page

Data Display Page:

The Data Display page currently has three variable settings for choosing a time frame (today, this week, and this month), a node (any individual node, group of nodes, or the aggregate of all nodes), and a data type (RMS Power, RMS Current, RMS Voltage, or Power Consumption). With these settings, it displays a line plot of the chosen data type, from the chosen node, over the chosen time interval. The time axis unit changes depending on the interval selected, and the data axis scale changes based on the max and min of the series being plotted. Although this is not a real-time plot, the update button will reload the database with the most current values. In addition, if RMS Power or Power consumption are chosen, the total power consumption is computed and displayed to the screen. Lastly the node variable pulls the node and group names for the node.txt and group.txt files.

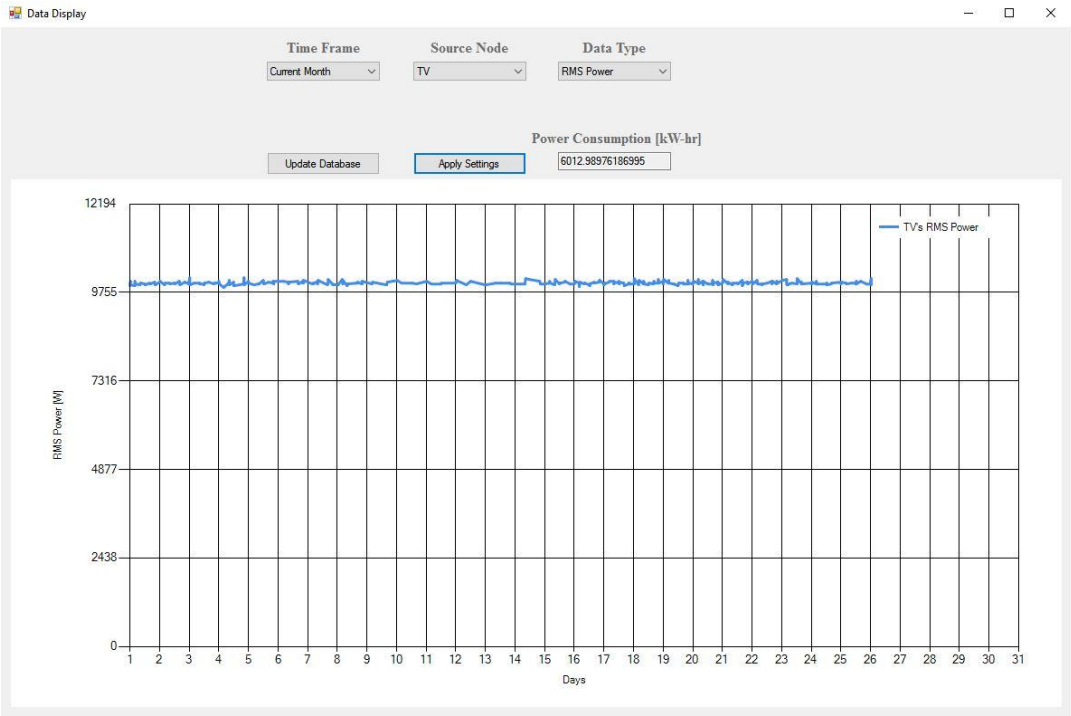


Figure 5: Data Display Page

Input Commands:

The Input Commands page simply displays all the nodes in the system and their current on/off status. In addition, it allows the user to toggle the nodes on and off, by typing in the node name, and pressing the toggle button. Upon saving, it writes the MAC Address of all toggled nodes, and a 1/0 (to denote being turned on/off) to a binary file for communications.

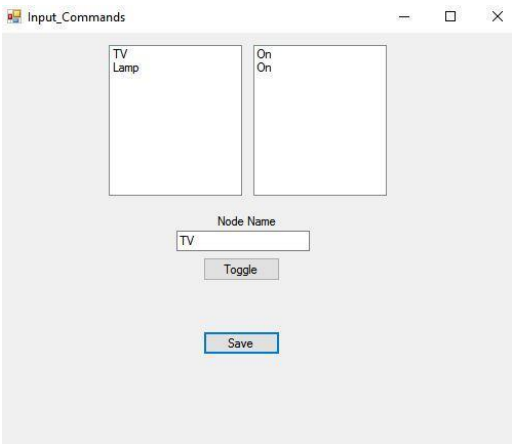


Figure 6: Input Commands Page

Notification Logs:

The notification logs consist of the command and error log pages. These pages are identical in their purpose and implementation. They simply display their respective notification type from most recent to least recent with a time stamp, and a small message with the source node. An update button is included to get the most recent notifications.

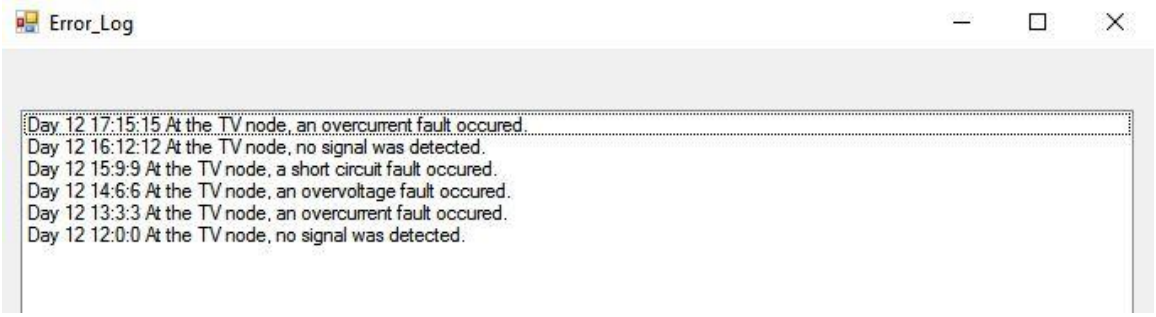


Figure 7: Error Log Page

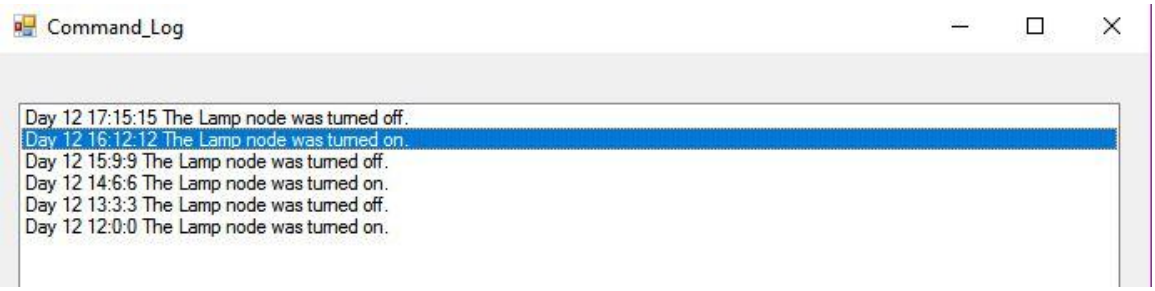


Figure 8: Command Log Page

There isn't much data to report. The only data that has influenced the software subsystem is data about the C++ programming language and how to use it in Visual Studios. When this project began, I thought I had a good foundation in C++, but I was very wrong. I do have a sense for programming, but this project has been a crash course in C++ and how to learn C++.

The validation plan for this subsystem was to make sure that all of the files exchanged between software and communications were being read and written correctly by the software. Notifications and data binary files were created by a node, were successfully displayed correctly in the GUI. The commands binary file is written by the software, and has been successfully received by communications.