Wireless Dimmer Control Using Power Line Communication

by
Bernard Grégoire
Dustin Morscheck
George Wong

Final report submitted in partial satisfaction of the requirements for the degree of
Bachelor of Science
in
Electrical and Computer Engineering
in the
Faculty of Engineering
of the
University of Manitoba

Faculty Supervisor:
Behzad Kordi, Ph.D., P.Eng.

Spring 2013

© 2013 Copyright by Bernard Grégoire, Dustin Morscheck, and George Wong
Abstract

Power Line Communication (PLC) in a home environment is used to control a network of units which can dim their loads powered by the mains voltage while monitoring the power usage of those loads and send it to a user. An off the shelf PLC modem chip is used for communication between one to 16 nodes over a home power wiring network operating on a client-server architecture. Each node has the ability to supply 120VAC power to a 1A load while dimming it, turning the load off, or running the load without impeding the source voltage. Wi-Fi is used to connect a user with a PC to control the dimmer settings and monitor the power usage of each individual node, by a Wi-Fi connection to a single master node. Multisim and MATLAB were used to design and simulate our circuit designs for the power supply, dimmer, power monitor, and PLC modem analog coupling to the mains. The designs were constructed and tested with standard lab equipment. The project was not completed due to problems in constructing the units, and lack of documentation for the software side of some of the components selected.
Contributions

Power Line Communication allows the use of pre-existing power lines as a communication network instead of installing new lines. The ability of controlling and monitoring power going into loads around a household can allow homeowners to reduce power consumption and save money on their electric bills. The following table shows a breakdown of the division of tasks between members for the project.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Individual(s) in Charge</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Literature Review</strong></td>
<td>Filter Design</td>
</tr>
<tr>
<td></td>
<td>Power Supply Design</td>
</tr>
<tr>
<td></td>
<td>USB Protocol</td>
</tr>
<tr>
<td></td>
<td>Power Monitor Review</td>
</tr>
<tr>
<td><strong>Hardware Design</strong></td>
<td>High Level Design</td>
</tr>
<tr>
<td></td>
<td>Power Supply Design</td>
</tr>
<tr>
<td></td>
<td>Power Monitor Design</td>
</tr>
<tr>
<td></td>
<td>Dimmer Design</td>
</tr>
<tr>
<td></td>
<td>MCU Circuitry</td>
</tr>
<tr>
<td></td>
<td>PLC Modem Circuitry</td>
</tr>
<tr>
<td><strong>Software Design</strong></td>
<td>GUI</td>
</tr>
<tr>
<td></td>
<td>Unit to GUI Protocol</td>
</tr>
<tr>
<td></td>
<td>Unit Control Code</td>
</tr>
<tr>
<td><strong>Software Programming</strong></td>
<td>GUI</td>
</tr>
<tr>
<td></td>
<td>MCU code</td>
</tr>
<tr>
<td><strong>Building &amp; Testing</strong></td>
<td>Order Parts</td>
</tr>
<tr>
<td></td>
<td>Build Subunits</td>
</tr>
<tr>
<td></td>
<td>Test Subunits</td>
</tr>
<tr>
<td></td>
<td>Build MCU &amp; Modem</td>
</tr>
<tr>
<td></td>
<td>Test MCU &amp; Modem</td>
</tr>
<tr>
<td></td>
<td>Full Prototype Build</td>
</tr>
<tr>
<td></td>
<td>Full Test</td>
</tr>
<tr>
<td><strong>Final Report</strong></td>
<td></td>
</tr>
</tbody>
</table>
Acknowledgement

We would like to thank our advisor, Dr. Behzad Kordi, for introducing the topic to us, and for his guidance during the course of the project. We would also like to thank the following people who were instrumental in the success of this project; Aidan Topping for her useful feedback on our reports, Sinisa Janjic and Allan Mckay for ordering of parts and general technical help, and Zoran Trajkoski for the manufacture of the printed circuit boards.
# Table of Contents

Abstract .............................................................................................................................................. i  
Contributions .................................................................................................................................... ii  
Acknowledgement .............................................................................................................................. iii  
Table of Contents ............................................................................................................................... iv  
List of Figures ..................................................................................................................................... v  
List of Tables ..................................................................................................................................... viii  
Glossary of Terms .............................................................................................................................. ix  
Nomenclature .................................................................................................................................... xi  

Chapter 1 – Introduction ...................................................................................................................... 1  
1.1 Motivation ..................................................................................................................................... 1  
1.2 Power Line Communication ......................................................................................................... 1  
1.3 Design Details ............................................................................................................................... 2  
1.4 Requirements & Specifications .................................................................................................... 3  
1.5 Report Organization .................................................................................................................... 5  

Chapter 2 – Theory .............................................................................................................................. 6  
2.1 Power Line Communication ......................................................................................................... 6  
2.2 Dimmer ....................................................................................................................................... 6  
2.3 AC Mains ..................................................................................................................................... 9  
2.4 Power Monitor ............................................................................................................................ 9  

Chapter 3 – Hardware Design & Simulation ...................................................................................... 11  
3.1 Topology ..................................................................................................................................... 11  
3.2 Power Supply ............................................................................................................................. 12  
3.3 Dimmer ...................................................................................................................................... 17  
3.4 Microcontroller ......................................................................................................................... 22  
3.5 PLC Modem .............................................................................................................................. 22  

List of Figures

FIGURE 1-1: TYPICAL HOME NETWORK SETUP ................................................................. 3

FIGURE 2-1: PWM EXAMPLE ......................................................................................... 7

FIGURE 2-2: MAINS PWM SPECTRUM ........................................................................... 9

FIGURE 3-1: SYSTEM TOPOLOGY .................................................................................. 11

FIGURE 3-2: UNIT ARCHITECTURE .............................................................................. 12

FIGURE 3-3: POWER SUPPLY SCHEMATIC ................................................................. 13

FIGURE 3-4: POWER SUPPLY SIMULATION ................................................................. 17

FIGURE 3-5: PWM CONTROLLER SCHEMATIC ............................................................ 19

FIGURE 3-6: SWITCH SCHEMATIC ............................................................................... 20

FIGURE 3-7: PWM CONTROLLER TRANSIENT SIMULATION ....................................... 22

FIGURE 3-8: PLC MODEM CHIP .................................................................................. 23

FIGURE 3-9: SUPERDIODE CIRCUIT DIAGRAM ......................................................... 26

FIGURE 3-10: POWER MONITOR SCHEMATIC ............................................................ 27

FIGURE 3-11: POWER MONITOR TRANSIENT SIMULATION ...................................... 32

FIGURE 3-12: BURNT TRACE ....................................................................................... 33

FIGURE 4-1: SIMPLIFIED UML CLASS DIAGRAM ....................................................... 34

FIGURE 4-2: GRAPHICAL USER INTERFACE WINDOW ............................................. 35

FIGURE 4-3: PLC PROTOCOL FLOWCHARTS ............................................................... 37

FIGURE 4-4: PLC PROTOCOL PACKET FORMATS ....................................................... 37

FIGURE 4-5: WIRELESS PROTOCOL MASTER NODE FLOWCHART ............................. 38
FIGURE 4-6: WIRELESS PROTOCOL PC FLOWCHART ................................................................. 39
FIGURE 4-7: WIRELESS PROTOCOL PACKET FORMATS ...................................................... 39
FIGURE 5-1: MODEM LINE COUPLING CIRCUIT ................................................................. 41
FIGURE 5-2: PLOT OF DAC INPUT VS. MEAN OF PWM OUTPUT AND BEST FIT LINE ............ 44
FIGURE 5-3: PWM CONTROLLER OUTPUTS ............................................................................ 45
FIGURE 5-4: PWM CONTROLLER TESTING CIRCUIT ............................................................ 45
FIGURE 5-5: SWITCH TESTING SETUP .................................................................................... 47
FIGURE 5-6: LOAD VOLTAGE EXAMPLE DURING SWITCH TEST ........................................... 48
FIGURE 5-7: VOLTAGE MONITOR OUTPUT VS. ACTUAL MEAN AND BEST FIT LINE .......... 50
FIGURE 5-8: VOLTAGE MONITOR OUTPUT VS. ACTUAL MEAN AND BEST FIT LINE ............ 52
FIGURE 5-9: ANALOG FULL SIDE TEST OUTPUT – DAC CONTROLLING LOAD VOLTAGE WITH PWM AT VARYING DUTY CYCLES .......................................................................................... 53
FIGURE 5-10: ANALOG FULL SIDE TEST SETUP .................................................................... 54
FIGURE 5-11: INCOMPLETE PCB BOARD .............................................................................. 55
FIGURE 5-12: PAD LAYOUT FOR MODEM SOCKET ............................................................... 56
FIGURE 5-13: MICROCONTROLLER AND WI-FI SETUP ....................................................... 56
List of Tables

TABLE 1-1: SPECIFICATIONS ........................................................................................................ 5

TABLE 5-1: PWM LINEARITY RAW RESULTS .............................................................................. 43

TABLE 5-2: PWM LINEARITY ANALYSIS RESULTS ................................................................. 43

TABLE 5-3: VOLTAGE MONITOR LINEARITY RAW RESULTS ................................................. 49

TABLE 5-4: VOLTAGE MONITOR LINEARITY ANALYSIS RESULTS ....................................... 49

TABLE 5-5: CURRENT MONITOR LINEARITY RAW RESULTS ................................................ 51

TABLE 5-6: CURRENT MONITOR LINEARITY ANALYSIS RESULTS ........................................ 51
Glossary of Terms

**Power Line Communication (PLC):** A mode of communications using pre-installed power lines for the transmission medium.

**Wi-Fi:** Any “wireless local area network (WLAN) products that are based on the Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standards.” [1]

**National Instruments Multisim (Multisim):** Software used for simulating electric circuits.

**MathWorks MATLAB (MATLAB):** Numerical analysis software used to calculate functions and plot data.

**Pulse Width Modulation (PWM):** A method of controlling the magnitude and frequency of the output waveform by modifying the duty cycle of a switching signal.

**Electromagnetic Interference (EMI):** A disturbance to a circuit from external source of electromagnetic radiation, inductive coupling, or capacitive coupling.

**Perfboard:** Material useful for prototyping circuits using through-hole components made up of a grid of holes each one surrounded by a copper pad.

**Serial Peripheral Interface (SPI):** A three-wire full duplex synchronous point-to-point serial communication protocol.

**Digital-to-analog Converter (DAC):** A circuit that takes a digital word value and converts it to an analog voltage signal.

**Analog-to-digital Converter (ADC):** A circuit that takes an analog voltage signal and converts it to a digital word value.

**Graphical User Interface (GUI):** A PC application or component which uses the mouse and controls such as buttons, checkboxes, and windows for primary user interaction.

**Root Mean Square (RMS):** A measure of a varying quantity which is calculated by taking the square root of the mean of the square of a function being measured such as voltage or current.

**Spread Frequency Shift Keying (SFSK):** An FSK technique, with frequencies spread further apart than regular FSK, which is used to modulate the PLC signal on the mains.
Frequency Shift Keying (FSK): A modulation scheme using shifts in frequency of a carrier waveform to encode information.

Signal-to-Noise Ratio (SNR): The ratio of the desired signal level to the unwanted noise level.

Rectifier: A power electronic device that converts alternating current into a direct current.

Low-Pass Filter (LPF): An electronic filter, which filters out the low frequency component.

Equivalent Series Resistance (ESR): Representation of the resistance in series in a circuit model which can be caused by different factors (such as the energy required orienting the dielectric in a capacitor).

High-Pass Filter (HPF): An electronic filter, which filters out the high frequency component.

Linear Technology LTSpice (LTSpice): SPICE-based software used for simulating electric circuits, in many ways similar to Multisim, containing LT components with the ability to import SPICE models for other components.

National Instruments Ultiboard (Ultiboard): Software used to create layouts for PCB traces.

Eclipse (Eclipse): Free and open-source programming environment mostly geared for Java applications, although it can be used with many languages using plug-ins.


Microchip MPLAB X (MPLAB X): Environment for embedded applications development using Microchip PIC controllers.

Arithmetic Logic Unit (ALU): Microcontroller component used to perform mathematical and logical operations.
Nomenclature

For symbols, this report uses the standard practice wherein capitalized letter represents a constant value while a non-capitalized letter represents a variable value.

### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Alternating Current</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog-to-Digital Converter</td>
</tr>
<tr>
<td>ALU</td>
<td>Arithmetic Logic Unit</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital-to-Analog Converter</td>
</tr>
<tr>
<td>EMI</td>
<td>Electromagnetic Interference</td>
</tr>
<tr>
<td>ESR</td>
<td>Equivalent Series Resistance</td>
</tr>
<tr>
<td>HPF</td>
<td>High-Pass Filter</td>
</tr>
<tr>
<td>IC</td>
<td>Integrated Circuit</td>
</tr>
<tr>
<td>LPF</td>
<td>Low-Pass Filter</td>
</tr>
<tr>
<td>MCU</td>
<td>Microcontroller</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PLC</td>
<td>Power Line Communication</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
</tbody>
</table>

### Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Ampère (unit of current)</td>
</tr>
<tr>
<td>C</td>
<td>Capacitor</td>
</tr>
<tr>
<td>f</td>
<td>frequency</td>
</tr>
<tr>
<td>F</td>
<td>Farad (unit of capacitance)</td>
</tr>
<tr>
<td>H</td>
<td>Henry (unit of inductance)</td>
</tr>
<tr>
<td>Hz</td>
<td>Hertz (unit of frequency)</td>
</tr>
<tr>
<td>i</td>
<td>current</td>
</tr>
<tr>
<td>L</td>
<td>Inductor</td>
</tr>
<tr>
<td>R</td>
<td>Resistor</td>
</tr>
<tr>
<td>V</td>
<td>Voltage</td>
</tr>
<tr>
<td>W</td>
<td>Watt (unit of power)</td>
</tr>
<tr>
<td>Z</td>
<td>Impedance</td>
</tr>
<tr>
<td>τ</td>
<td>Time Constant</td>
</tr>
<tr>
<td>Ω</td>
<td>Ohms (unit of resistance)</td>
</tr>
</tbody>
</table>
Chapter 1 – Introduction

1.1 Motivation

With increasing amounts of electrical appliances in residential homes, power usage is becoming a larger concern. Home owners require a more convenient system to monitor power usage so they can make decisions to use power more efficiently. Since there is already a wired network in most homes, the power line mains, it is unnecessary to add additional wiring for a power monitoring system. Therefore, the mains should be used for such a power monitoring and control system, increasing the convenience for the home owner.

The design of a Wireless Dimmer Control Using Power Line Communication was undertaken because of a need for users to efficiently manage the consumption of power in their homes. Costs will be kept to a minimum by the use of existing home wiring.

1.2 Power Line Communication

A power line communication (PLC) system superimposes a signal on the mains. It is the usage of power lines as a transmission channel for the exchange of data [2]. Each PLC unit can send or receive commands and data using this communication channel. The power usage of a load is monitored by a PLC unit, and the resulting data is sent back to the home owner over the power lines. The user can then reduce the power going into the load with a dimmer circuit on the PLC unit.

Electrical companies are increasingly using smart meter technology to monitor their client’s power usage. These meters track and record power usage and can deliver statistic such as peak usage hours and total energy used. The information is relayed over the already laid power line networks with nodes at each transformer box. This information is then transferred to the main office resulting in a low cost information collecting system. This system is designed in mind of the electric company’s interest, and not the home owner. The system does not allow a home owner to track his or her own usage per appliance and reduce consumption accordingly. The system we propose will open up that opportunity.
1.3 Design Details

We proposed a design which can control multiple loads using multiple nodes, with a home owner able to control the system through a PC application using a wireless connection over Wi-Fi. A node has connected to it a single load such as a lamp or fan. The load can be turned off, turned on, or dimmed and can be inductive, capacitive, or resistive. The power going through each load is monitored and sent back to the user. Each node communicates with the others through PLC on the household mains. The user ultimately sees the power consumption information, and controls the dimmers through a Wi-Fi connection to one of the nodes through a custom designed application on a PC.

We chose to implement PLC using an existing off-the-shelf modem. The data which is to be communicated using PLC contains information about the power usage of any load connected to our device and a control signal which will allow the user to regulate the amount of power being sent to the load. This data is used to pulse-width modulate (PWM) the load to regulate its power. The power is monitored by measuring the average voltage and average current separately and multiplied, giving the average power.

The PLC system was designed and simulated using Multisim and MATLAB software. Hardware testing was done on breadboards, before the final printed circuit boards (PCBs) and perfboard layouts were designed. Three units were to be produced in hardware on a PCB with case, and tested with standard lab equipment such as function generators and oscilloscopes.

The wireless connection the user’s PC could have been done with either of the two most common standards; Bluetooth or Wi-Fi. Wi-Fi was chosen over Bluetooth because it has a larger range, is more commonly used in households than other communication system, and is compatible with more PC’s, since most PC’s have a Wi-Fi adapter, but less have a Bluetooth adapter.

There are several options for dimming of a load. A resistive voltage divider is one method that works well, but the drawbacks are that power consumption is high and it is more difficult to design. Pulse-width modulation has lower power consumption. It has been used by previous groups, and we have the advantage of learning from their method so that problems they experience would not hinder our progress. It does produce high electromagnetic interference (EMI) due to high switching frequencies, and may interfere with the PLC modem’s operation if not designed correctly.
A typical setup can be seen in Figure 1-1; the green boxes represent the device that we designed. Its dimensions are approximately the same as a wall outlet. A load is connected to each unit; we will be using a lamp as the load, but it could be any household electrical device such as a cellphone charger or a fan. The nodes are arranged in a client-server architecture, there is only one master node for each network and 0 to 15 slave nodes. In the image, the master node is located on the bottom. All slave nodes can communicate with the master, while only the master has the additional ability to communicate wirelessly with the PC. An application on the PC allows the user, through the master node, to observe the power usage of each node and control the average voltage level sent to each node.

![Figure 1-1: Typical home network setup](image)

1.4 Requirements & Specifications

We are allowed to have a budget of $100 CAD for each group member; therefore the budget for this project was $300 ± 10%. We were allowed to use any off-the-shelf hardware to reduce the amount of design work required. The requirements we set for this project are outlined below.

For the functional requirements, our design should have a method of controlling the voltage level sent to the load from fully off, fully on, and at least 10 dimming levels in between. It should also be capable of tracking power usage to 5% accuracy as well as act as a form of circuit protection or power regulation. It needed to be a multi-nodal system with minimally 3 nodes but
ideally at least capable of 15, and wirelessly controlled via PC. It should be user friendly (simple to install and use) making it marketable. It should be capable of handling any expected interference include EMI, line transient and interference from between network. The system should contain one major access point that has Wi-Fi for controlling the other minor nodes. It should follow FCC regulations in terms of minimizing interference.

Components determined that were required for the design are a GUI on PC, a power supply, a dimmer which includes a PWM controller and power switch, a power measurement device consisting of a differential voltage monitor and a current monitor, a fuse for short circuit protection, a microcontroller, a Wi-Fi board, and the PLC modem system. The microcontroller should have a small board, operate at 10Mhz clock or higher, compatible C compiler, it should be inexpensive, have interrupt support, one serial peripheral interface (SPI) port for Wi-Fi communication, a digital-to-analog converter (DAC) for dimmer control, two analog-to-digital converter (ADC) inputs for power monitoring, and a UART for communication to the modem. The Wi-Fi card should allow SPI communication, include a built-in antenna, and be easy to integrate into the MCU not requiring significant programming. Alongside the hardware design components, the communication protocol between nodes must be designed and coded onto the microcontrollers, and the master node to PC protocol and control application must be designed and written.

We limited the scope of the project to be able to control 100W lamp and similar loads. This results in a current limit of 1A for the load of our system, for a total output power rating of 125W. In a network there needs to be a master node and there can be a very large number of slave nodes, limited to the modem addressing (2¹⁶ possible addresses) and the ID codes allowable on our microcontroller design (limited to 16 unique ID’s). Due to budget limitations and since we only wish to prove the concept, we have three nodes: a master node and two slave nodes. More nodes could easily be added to our existing system. Dimming of the load must be varied from completely off to fully on and anything in between. Varying the duty cycle of the PWM system allows us to do this. We want to control the network wirelessly via PC and we chose to use Wi-Fi to achieve this. Finally, since our budget is limited to approximately $300, with three nodes the cost per node must be kept under $100.

The resulting requirements are listed in Table 1-1.
Table 1-1: Specifications

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Monitor</td>
<td>Power usage by household item plugged into the device</td>
<td>0-125W and 1s/frame</td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nodes</td>
<td>Number of nodes (devices), scalable</td>
<td>≥3</td>
</tr>
<tr>
<td>Dimmer Voltage</td>
<td></td>
<td>Off, 0-120V AC (RMS)</td>
</tr>
<tr>
<td>Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input Power</td>
<td>Run on ordinary household power socket</td>
<td>120V AC (RMS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60Hz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15A</td>
</tr>
<tr>
<td>Power accuracy</td>
<td>Accuracy of load power dissipation measurement</td>
<td>95%</td>
</tr>
<tr>
<td>Current Output</td>
<td>Maximum current at the device output</td>
<td>1A</td>
</tr>
<tr>
<td>Size</td>
<td>Maximum size of individual unit</td>
<td>15 cm x 7 cm x 10 cm</td>
</tr>
</tbody>
</table>

1.5 Report Organization

Chapter 1 provides an introduction to both the topic and to the layout of the report.

In Chapter 2, the theory of power line communication is presented. Modulation, common methods of dimming, and MOSFETs are also discussed to give the reader a background on the topics involved.

In Chapter 3, the hardware design process is analyzed by discussing the topology of the unit and each one of its components (power supply, dimmer, microcontroller, PLC modem, Wi-Fi controller, power monitor). For each component, the design is covered in detail and simulation results are presented. Finally, the logic behind the board layout is explained.

In Chapter 4, the software design and implementation is presented. It discusses the graphical user interface (GUI), the communication protocols, and the microcontroller.

In Chapter 5, all of the testing and results are disclosed in detail. It begins with testing of individual modules before proceeding with a full analog-side test.

Finally, in Chapter 6, we provide some recommendations for future work on this topic and conclude the report.
Chapter 2 – Theory

2.1 Power Line Communication

Power line communication is a form of communication which uses power lines as the communications channel. It can be used for a variety of purposes from Internet access to home automation. Also, it can be used at the different potential levels in a power utility network; the high voltage transmission lines, the medium voltage distribution lines, or the low voltages present inside of homes and buildings. Exchange of data is done by modulating a high-frequency signal onto the lower frequency mains power. The power lines serve a dual purpose: transmission of power and communication of information.

Modulation of the data signal is done using spread frequency shift keying (SFSK). In regular frequency shift keying (FSK), information is transmitted by changing the frequency of a carrier wave. The simplest type is binary FSK which only uses two symbols: zero and one. The frequency chosen to represent the ‘0’ symbol is called the space frequency while the one chosen to represent the ‘1’ symbol is called the mark frequency. These frequencies are relatively close to each other and a filter is required to differentiate between the two.

In SFSK, there is a larger gap between the frequencies. It is a slight modification which results in major benefits. By separating the frequencies (or tones) the bandwidth is increased and the noise immunity is improved. Tones are usually chosen between 20kHz and 100kHz which keeps it far away from the low frequency of the mains and therefore results in a high signal-to-noise ratio (SNR). Small packets of tones are sent at the zero-crossings of the power line’s voltage which makes this method very reliable. Baud rate (also known as modulation rate) is the rate at which the frequency tones occur.

2.2 Dimmer

Dimming is used to control the power going to the load. This can be done using pulse-width modulation. PWM controls the width of the pulses and therefore controls the average voltage. Duty cycle is the percentage of time that voltage is applied to the load. By changing the on/off duty cycle, we can determines this width of the pulses. As can be seen in Figure 2-1, the average voltage is directly proportional to the duty cycle. A shorter duty cycle results in a smaller
average voltage sent to the load, whereas a longer duty cycle results in a larger average voltage sent to the load. The examples in the figure use a constant DC value but this concept can be applied to other signals as well, such as a sinusoidal wave.

Switching a high voltage, high current load on and off must be done with components that can handle the line ratings. Several common switching solutions are thyristors, relays, power MOSFETs and IGBTs. Thyristors are latching semiconductors often used in applications of under 10kHz. The thyristor latches on when applying a gate to anode current, and stays on until the cathode to anode current reduces to near zero, even if the gate current goes to zero. Relays are electromechanical devices which use a solenoid coil to toggle an internal switch. They are unreliable and operate very slowly. Power MOSFET’s are MOSFET’s designed with a very low $R_{on}$ and often have a reverse voltage protection diode in parallel with the drain and source. The gate capacitance is often much larger than in other MOSFET’s and require a significant amount of charge to switch on the MOSFET. They can switch with very high frequencies given that the gate can be driven with enough current. IGBT’s are similar to MOSFET’s but have intrinsic BJTs at the drain and source which amplify the current handling capability. They have a smaller voltage drop than a similar sized power MOSFET at larger currents, but switch slower. The

![Figure 2-1: PWM Example](image-url)
longer switching times correspond to higher switching losses. IGBT’s, power MOSFET’s, and thyristors are unipolar devices, in that current can only be passed in one direction. The control voltage must also be close to one of the terminal’s voltages, so direct isolation is not possible. The relay has neither of these issues, and the control voltage is galvanically isolated since the switch is magnetically operated.

MOSFETS are usually three terminal devices with a gate, drain, and source. They are nearly ideal semiconductor switches when operated in the linear region of their transfer characteristic, and can be modelled as a small resistance when conducting (often near 10mΩ for power MOSFETS). They can be modelled as an open circuit (resistances in the 100MΩ to 10GΩ range) when not conducting. The MOSFET conducts between the drain and source when the voltage between the gate and source terminals minus the threshold voltage device parameter is above zero. This voltage difference must be raised to allow more current to pass, otherwise the MOSFET saturates and behaves approximately as a current source. The transition region between these two modes of operation is approximately quadratic with the drain-source voltage. The gate is essentially insulated from the conducting channel between the drain and source and can be modelled as a capacitance in the pF range. It requires little current to switch compared to a BJT, but enough to charge the gate capacitance to switch at the required speed. The gate to source junction is very sensitive to high voltages, and is usually limited to 30V before breakdown, even if the transistor can handle over 400V from drain to source.

Using PWM to dim a load with AC power, major impulses in the spectrum exist at every odd harmonic of the PWM signal’s fundamental frequency plus/minus the fundamental frequency of the AC signal. Figure 2-2 shows the positive side of the spectrum of switching a 1V, 60Hz signal at 35kHz, obtained with a Hamming window to improve the side lobes resulting from the truncation required by the DFT. This spectrum is of a high magnitude due to the mains voltage. The PLC modem communicates on the same lines that power the load, and the mains are not ideal and can be modelled as an RLC network. This switching will put some signal onto the lines since the load impedance is rapidly changing, which affects the loading of the lines, and hence the line voltage. At low duty cycles, the signal is essentially a train of impulses, and gives a frequency spectrum which is also a series of impulses with a very slow attenuation with frequency. The modem would have to operate at frequencies in the MHz range to avoid interference from the PWM circuit, which is not possible with many modems. To reduce interference between the modem and the PWM circuit, the fundamental frequency of the PWM circuit can be placed higher than the modem’s highest frequency so the modem circuit can reliably filter out all
interference. In the graph below, the modem would operate between the PWM circuit’s zeroth and first harmonics, allowing space for lobes present in an amplitude shift keying (ASK) signal, since SFSK is essentially two simultaneous ASK signals.

![Figure 2-2: Mains PWM Spectrum](image)

### 2.3 AC Mains

When connecting to the power lines it is important to understand the difference between ground and neutral wires. The neutral wire is the return path for the current and can be at a different potential than the ground wire. The neutral wire is not necessarily the same potential as ground potential at all points since it has a non-zero resistance and carries a large amount of current, creating a non-zero voltage across it. The ground wire, with ground level potential, is a form of protection and usually does not carry current. It is in place to prevent high voltages being present on the case of equipment with which people may come in contact. For this reason, the ground should never be connected at the neutral except at a breaker panel or fuse box.

### 2.4 Power Monitor

The average power across a load can be found by multiplying the average of the magnitude of the voltage and the average of the magnitude of the current. Finding the average value of either component as a sine wave can be done by first taking the absolute value of the signal. This can be achieved with diodes or bridge rectifiers. Diodes and bridge rectifiers exhibit a voltage drop which must be taken into account in design, if the drop is significant relative to the amplitude of the signal. The average value of a signal can be found by obtaining the DC component of the signal. For any periodic signal \( x(t) \) on the interval \( T_0 \), the DC component can be determined with the Fourier series as

\[
a_0 = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} x(t) \, dt
\]

The mean value of a continuous signal is defined as
As we increase the period of the Fourier DC component to infinity, we achieve equivalent equations

\[ \bar{x} = \lim_{T \to \infty} \frac{1}{T} \int_{-\infty}^{\infty} x(t) \, dt \]

Since all signals which are periodic on a finite period are also periodic on an infinite period, the DC component is synonymous with the mean value of a signal for all realizable signals. To extract the DC component, a low-pass filter (LPF) which significantly rejects all harmonics except the 0th harmonic can be used, leaving a DC value at the output which can be buffered and manipulated as needed. The harmonics cannot be completely rejected in an analog filter, so the output will contain some ripple whose magnitude is dependent on the rejection of the major harmonics present.
Chapter 3 – Hardware Design & Simulation

3.1 Topology

Using the specifications outlined in the previous section, we came up with a design. After giving a general overview of the topology of the system the various components will be described in more detail.

We are using the client-server architecture between nodes. There are three or more units connected to the power line for both power and communication. Each node is capable of turning on, turning off, or dimming a 120V, 1A load. One of these nodes is a master node which is capable of communicating with a PC running a custom-designed application over Wi-Fi for control. The master node is identical to the slave node in hardware other than the Wi-Fi connection. Figure 3-1 shows the topology of the system as described.

Figure 3-1: System topology

Figure 3-2 shows the architecture of an individual unit. Each unit uses a microcontroller for communications and control. The Microchip PIC18F45K22 [3] was chosen as the microcontroller due to low cost, sufficient output modules, and prior experience with the PIC series chips. A custom designed power monitor circuit relays back the power used by the load to the microcontroller for display to the control application. A dimmer circuit, also custom designed, uses PWM to regulate the power going through the load, and to turn the load completely on or off. A custom power supply, running on the mains and used to power the unit, is designed to output the various required voltages at sufficient currents. No commercially available circuits for the required tasks met both functional and financial requirements, requiring custom design. A power line modem chip, the ON Semiconductor AMIS-49587 [4], was chosen for power line communication due to ease of use and at the time it was the most affordable fully featured modem on the market. The Wi-Fi card chosen for the master unit is the Microchip
MRF24WB0MA [5] since it is designed to work with PIC series chips, simplifying programming. The slave nodes do not include the Wi-Fi module.

![Figure 3-2: Unit Architecture](image)

A power supply is needed to supply each unit (microcontroller, power monitor, dimmer, Wi-Fi card, PLC modem) with the correct supply voltage. We require relatively stable (i.e. DC) voltage at three levels: 3.3V, 5V, and 6V.

### 3.2 Power Supply

The linear power supply is a device consisting of a single transformer, a rectifier, a filter capacitor, and some regulators. Starting from the AC voltage input from the mains, we step it down to a smaller value using a Signal Transformer 14A-5.0R-12 [6] in series with a fuse [7] to protect the circuit in case of short-circuit. The lower amplitude AC is half-wave rectified using a Vishay 1N4001-E3/54 diode [8] and the output is filtered using a low-pass filter (an 18mF capacitor) to remove the 60Hz and obtain an approximately DC source. The reason we used a half-wave rectifier is that with a full-wave rectifier, such as a bridge rectifier, the voltage drop would be over two diodes, or approximately 2.4V, which wouldn't allow us to obtain the 6V supply without a larger transformer and more power loss across each regulator. Therefore, to keep
the same transformer, we perform half-wave rectification which means that the LPF capacitor needs to be larger. Another advantage is that the heat dissipation for the rectifiers is at a sufficiently low level that no external heat sink is required. At higher frequencies, the low-pass filtering electrolytic capacitor begins to act like an inductor due to its equivalent series inductance and cannot block the frequencies as effectively. We have 330nF ceramic capacitors before the regulators to filter out the high frequencies from noise and EMI. Figure 3-3 shows a functional schematic for the power supply.

![Power Supply Schematic](image)

The approximately DC source is regulated to remove any ripples by sending it to three separate regulators which adjusts it to the three required voltage levels. These regulators are off-the-shelf components. Due to the non-idealities of the first low-pass filtering capacitor, another capacitor is needed at the input of each voltage regulator to acts as a low-pass filter to remove the higher frequencies. For the 6V fixed voltage, used for the PLC modem line driver, we used a ROHM Semiconductor BA60BC0T regulator [9]. For the 5V fixed voltage, used for analog control and some digital circuits, we used a Microchip MCP1702-5002E/TO regulator [10]. Finally, for the 3.3V fixed voltage, used for digital circuits, we used a Fairchild Semiconductor KA278R33CTU regulator [11]. The total maximum current draw of the supply is 687mA, which is in the range that the transformer can supply. Due to a high voltage drop (~4V, or 3.7V to be exact) across the 3.3V fixed voltage regulator, a heat sink is required for the regulator to operate for a long period of time. Using the Assmann V7235-T heat sink [12] the regulator can operate at
a power dissipation of 15W which is more than enough for our needs because we only require the chip function at a level which would have a power dissipation of approximately 7W.

What follows are the calculations for the component values. The power supply needs to supply power to the MCU (5V; 20 mA), the Dimmer (5V; 55 mA), the power monitor (5V; 2 mA), the PLC modem (3.3V; 160 mA), the PLC driver (6V; 300 mA), as well as to the Wi-Fi card (3.3V; 154 mA). Therefore, it must be capable of supplying voltages of 3.3V, 5V, and 6V and a total current of 687 mA.

The rectifier diode has a drop of 1.1V. From the 6V regulator datasheet the maximum input voltage must be less than 15V. The 6V regulator datasheet states that the 6V regulator has a minimum voltage drop of 500 mV across the regulator; therefore the voltage at the output of the rectifier must be less than 15V but more than the sum of 6V, 500 mV, and 1.1V which is 7.6V. To be safe, we choose a voltage of 8.05V.

The mains power supply varies from 105V to 125V due to maximum loading. Looking at the worst-case, when the supply is lowest and we need to supply the highest value, we calculate the maximum transformer turns ratio.

$$\frac{V_1}{V_2} = \frac{l_2}{l_1} = \frac{n_1}{n_2} \leq \frac{105\sqrt{2}}{8.05V} = 18.446$$

We also need a transformer that will supply 8.05V peak or 5.69V RMS. We chose the Signal Transformer mentioned above which supplies 6.3V RMS at 800 mA with a turns ratio of 115V/6.3V or 18.25.

For the rectifier we use a single diode whose voltage drop is 1.1V. The voltage at the rectifier input is

$$V_{Diode,input} = 105\sqrt{2} \frac{6.3V}{115V} = 8.1348V$$

which makes the voltage at the rectifier output

$$V_{Diode,output} = 8.1348V - 1.1V = 7.0348V$$

Therefore, we chose voltage regulator components that are able to handle approximately 7V at their input node. From the datasheet of the 6V fixed output regulator, the voltage drop is 0.25V at an output current, $I_{out}$, of 500 mA. Thus, the voltage at input of the regulator must be
\[ V_{6V, \text{input}} = 6V + 0.25V = 6.25V \]

The allowed voltage ripple is 7.0348V - 6.25V = 0.7848V. This allows us to find the capacitor value

\[ C_{PS1} = \frac{I_{\text{total}}}{fV_{pp}} = \frac{77mA + 314mA + 300mA}{60Hz(0.7848V)} = \frac{0.691A}{60Hz(0.7848V)} = 14.67mF \]

Taking into account the ±20% tolerance of the capacitor

\[ 0.8C \geq 14.67mF \]

\[ C \geq \frac{14.67mF}{0.8} = 17.6mF \]

Taking the next largest standard value, we chose an 18mF capacitor. Now, we look at the heat dissipation. The voltage drop across the regulator is

\[ 7.0348V - 6V = 1.0348V \]

The current through the regulator is 300mA, consequently the power across the regulator is

\[ 1.0348V(0.3A) = 0.31044W \]

Using thermal resistivity value from the datasheet, the operating temperature for the given power dissipation is

\[ 0.31044W(62.5\degree C/W) = 19.4\degree C \]

Secondly, we calculate the 5V fixed output voltage regulator component values. The voltage drop across the regulator is

\[ 7.0348V - 5V = 2.0348V \]

The current through the regulator is 77mA, consequently the power across the regulator is

\[ 2.0348V(0.077A) = 0.15668W \]

Using thermal resistivity value from the datasheet, the operating temperature for the given power dissipation is

\[ 0.15668W(73\degree C/W) = 11.44\degree C \]
Thirdly, we calculate the 3.3V fixed output voltage regulator component values. The voltage drop across the regulator is

$$7.0348V - 3.3V = 3.71348V$$

The current through the regulator is 314mA, consequently the power across the regulator is

$$3.71348V(0.314A) = 1.1660W$$

From the datasheet of the 3.3V regulator, the absolute maximum power dissipation operation with heat sink is 15W, but without heatsink is 1.5W and the maximum operating temperature is 80°C. To cool the chip, we added a heat sink with a thermal resistance of 24°C/W. With an ambient air temperature of 25°C, the total operating temperature is a reasonable 56°C.

Finally, we have the input and output capacitors for the regulators. The input capacitors eliminate transients coming from the mains. The output capacitors act as a voltage storage device which will provide current to the load if the regulator cannot supply it fast enough. From the operating current, the datasheet gives an equivalent series resistance (ESR) for these external capacitors. Then, also from the datasheet, we find the capacitor values must be 0.33μF for the input and 100μF for the output. Using these two factors we chose capacitors that satisfy both capacitance and ESR requirements. These values apply to the 6V regulator which has the highest amplitude transients which need to be smoothed as well as close to the largest output current required. Therefore, the same capacitor values will be sufficient for the 5V and 3.3V regulators.

The simulation in Figure 3-4 shows the worst-case scenario with an input voltage of 105V RMS, the red graph on the left is the voltage taken at the output of the half-wave rectifier, the yellow graph on the right is the voltage taken at the output of the 5V regulator and the green graph on the right is the voltage taken at the output of the 3.3V regulator. As it is shown in the graph, the regulators are still capable of providing a constant voltage of 5V and 3.3V to its load under the worst case.
3.3 Dimmer

As stated in Section 2.2, it is ideal if the PWM fundamental frequency is significantly higher than the modem’s highest frequency. The modem requires spacing between the mark and space frequency to be at least 10kHz. Since our data rates are expected to be low, we can set the modem frequency to the minimum supported by the chip. The selected modem frequencies are 10kHz and 20kHz, and the dimmer must operate at or above 30kHz. Higher frequencies impose higher switching losses, since the power switch’s pull up and pull down times given a gate current is approximately constant with switching frequency, and the switch is in a transition state and dissipating power for more time per cycle. We should therefore set the dimmer frequency as low as practical while still ensuring correct operation. Because component tolerances can shift the actual frequency of the dimmer, the frequency was set to slightly higher than 30kHz, and components set so that the minimum possible frequency given component tolerances is just above 30kHz.

The dimmer’s PWM driver is controlled from the microcontroller using the microcontroller’s DAC output. The PWM driver uses an approximate triangle wave generator driving a threshold detector, which outputs a fully on, fully off, or a PWM dimming signal. This signal is passed through a voltage buffering inverter and switches a power MOSFET, the IRL640A [13], which cuts off current to the load through a GBUSK-BP bridge rectifier [14], since the MOSFET works correctly as a switch only with unipolar current. The switching
circuitry and the load are totally isolated from the mains with an isolation transformer to prevent grounding issues.

The dimmer uses PWM to control the average magnitude of the voltage (and therefore the power) sent to the load. It can be broken down into two circuits; the PWM driver and the switcher.

A relaxation oscillator is constructed with a single comparator operating as a Schmitt trigger charging and discharging $C_{D1}$. This relaxation oscillator outputs an approximate triangle wave with identical positive and negative slopes. Another comparator is used to create the PWM signal by outputting 5V when the triangle wave is above a certain input threshold and 0V otherwise. This allows for total 0% and 100% operation by placing the threshold completely below or completely above the triangle wave, respectively. The triangle wave goes from approximately 2V to 3V, and the DAC goes from 0V to 5V, so the DAC voltage is scaled with a linear resistor network at the input of the threshold comparator. The PWM driver has a BJT voltage buffer at its input. The output impedance of the DAC on the microcontroller is significant and it is stated in its datasheet that a voltage buffer is recommended at its output. The output impedance of this buffer is near 4.7kΩ driving an approximately 50kΩ load. The voltage here does not have to be accurate, but must be linear and must deliver a full enough range to control the threshold detector. The output of the PWM driver is buffered with a CMOS inverter to allow the power MOSFET to have enough gate drive current to switch sufficiently fast. Figure 3-5 shows a schematic of the PWM controller.
Several design equations were derived to assist in selecting components for the PWM controller. \( V_s \) is the supply voltage. \( V_c \) is the voltage at the output of the voltage buffer coming from the NPN voltage buffer. \( V_{r+} \) and \( V_{r-} \) are the maximum and minimum amplitudes of the triangle wave. \( V_{t+} \) and \( V_{t-} \) are the maximum and minimum inputs to the threshold detect comparator after the voltage scaling resistor network of \( R_{D5} \), \( R_{D6} \), and \( R_{D7} \). The triangle wave was selected to be between 2.0V and 3.0V, to operate with a mean that is half the voltage of the supply for easy biasing. The peak to peak is large enough to not be significantly affected by noise, but is small enough to be scaled with similar valued resistors. The following equations show required ratios between resistors used to select them.

\[
\frac{R_{D3}}{R_{D1}} = \frac{V_c - \Delta V_r}{2\Delta V_r}
\]

For the average \( V_r \) to equal half of \( V_s \), \( R_{D1} \) is the same value as \( R_{D2} \).

\[
For \lim_{a \to \infty} \frac{1}{a} \int_0^a V_r \, dt = \frac{V_s}{2}, R_{D1} = R_{D2}
\]

The triangle wave voltage amplitudes are found using
The voltage scaling network for the threshold voltage is determined by

\[
V_{r+} = V_S \frac{R_{D_2}R_{D_3} + R_{D_1}R_{D_2}}{R_{D_1}R_{D_2} + R_{D_1}R_{D_3} + R_{D_2}R_{D_3}}
\]

\[
V_{r-} = V_S \frac{R_{D_1}R_{D_2}}{R_{D_1}R_{D_2} + R_{D_1}R_{D_3} + R_{D_2}R_{D_3}}
\]

The fundamental frequency of the triangle wave can be found from the time the capacitor takes to charge and discharge between the triangle wave rails. The resulting equation is

\[
f = \frac{1}{-R_{D_4}C_{D_1} \ln\left(\frac{V_{r+}}{V_{r-}} : \frac{V_{r+} - V_{r-}}{V_{r+} - V_S}\right)}
\]

The load is switched on and off with an n-channel MOSFET. The MOSFET source pin is placed at near ground potential so the 0 to 5V control signal can correctly switch the MOSFET. The source is not exactly at ground since the current monitor’s sense resistor is between the MOSFET source and ground, and can have up to 700mV across it. The MOSFET as well as the power monitor circuit require unipolar current. The switching circuit interfaces the AC side and the unipolar side with a bridge rectifier. This bridge rectifier also is used in rectifying the signals for the power monitor. Figure 3-6 shows a functional schematic of the switch circuit.

![Figure 3-6: Switch Schematic](image-url)
An isolation transformer [15] bridges the gap between the dimmer and the mains to prevent grounding issues. As stated in Section 2.3, the current carrying neutral wire is not necessarily at ground potential and may have a DC voltage on it. The isolation transformer is used to remove this DC component and make it safe to attach the circuit to ground.

Several dimmer designs were attempted. Some designs using a 555-timer chip as a PWM generator were attempted but could not be simulated by us due to errors in the model provided by Multisim, so we tried other options. Several PWM drivers using discrete components which charged a capacitor with a current source and discharged the same by shorting it with a transistor was attempted, but proved not to allow a full 100% duty cycle, since discharge time of a capacitor is finite, but we required a near infinite discharge time. A discrete component triangle wave generator was created by charging and discharging a capacitor with separate current sources. This design required perfect matching of currents for the two current sources in order to obtain duty cycles approaching 0 with threshold detection, which was not possible.

A harmonic filter was designed to prevent interference and EMI caused by the switching on the mains, but was found to be prohibitively expensive due to large components needed for the low mains frequency and high current, so this was abandoned. We also designed several methods of isolating sensitive circuitry from the power lines, but all solutions cost nearly the same as an isolation transformer due to high circuit complexity. The designs included opto-isolators for MOSFET gate drives and complimentary unregulated and isolated low current DC supplies, and signal transformers on the inputs of the power monitor, which could be made to pass 60Hz since impedance was very high. These ideas were abandoned for a simple isolation transformer solution, since cost and performance were comparable.

The final circuit was simulated in Multisim and proved to allow 0% to 100% inclusive PWM. The frequency centered at about 35kHz, and does not go below 30kHz in the worst case situation. The dimmer was implemented in hardware as a prototype and tested successfully.

Figure 3-7 shows start-up of the circuit in a transient simulation using Multisim. Note the time before the oscillator starts from t=0 to 650μs. The simulation provides an overly ideal situation with very low noise and offset errors. The Schmitt trigger oscillator requires a small offset between its terminals to be present to start oscillating. This only occurred after the simulation accumulated enough quantization error to start the oscillation. In practice, the physical device starts oscillating immediately after power on.
3.4 Microcontroller

The chip we have chosen is the Microchip PIC18F45K22 [3]. It is an 8-bit chip. It was chosen because it has necessary ports without requiring complicated port switching circuits external to the MCU. It has good processing speed, is cheap enough to be allowed by our budget and has the necessary units (SPI, UART, ADC, PWM) we require. The final issue is our time constraint – we can’t look into every existing MCU to find the optimal one for our purposes.

Each node is assigned a unique ID number using a 4-bit DIP switch on each unit. This ID is used by the network and the PC control GUI to identify nodes. The master node is assumed to have ID 1, and the slaves can take any of the remaining 15 ID numbers.

3.5 PLC Modem

We were originally going to go with the LinkSprite SPYDER which is a complete board, but it has insufficient documentation and is expensive. Another modem we considered was the Texas Instruments AFE030 [16] but it did not have a complete collision detection or receiver function (expected to be implemented by user of MCU). The chip we chose was the ON Semiconductor AMIS-49587 [4] which can be seen in Figure 3-8. The PLC modem we have chosen is an off-the-shelf component. It did require some external circuitry (such as capacitors,
resistors, and OpAmp) which we had to order and build. The tonal frequencies for this particular chip can be chosen between 9kHz and 95kHz. The baud rate is 360Bd to 2880Bd. The AMIS-49587 [4] has an advanced easy-to-use multinodal communication, is reasonably priced, and the external circuitry is straightforward. It is structured in the OSI model and implements the physical layer (partially) and the link layer. The reason this particular chip was selected is that it comes with a pre-built low-level protocol and was the most inexpensive chip with the capability we required which was available at the time it was chosen.

Figure 3-8: PLC Modem Chip

We looked at the PLC modem datasheet to see what it needs as supporting circuitry (filters, oscillator, etc.) using Figure 1 on p. 2 of the PLC Modem datasheet [4]. For the transmission filter, we are using a 95kHz low-pass fourth order active filter using the ON Semiconductor NCS5650 [17] as specified in the datasheets to remove aliasing from the digital synthesis signal from the modem. This is to allow the zero and mark frequencies to pass, and eliminate aliasing. At the receiver, the resistors and capacitors form (along with the internal chip circuitry) a 9kHz, second order active high-pass filter (HPF) which blocks the 60Hz mains frequency and passes the higher frequency communication. The Zener diodes are there to prevent voltage spikes from causing damage to the chip.

The modem transmits and receives data on the mains through a modem coupling circuit consisting of a transformer for isolation and a capacitor to reduce the voltage at the modem. See Appendix B for a schematic. The capacitor in series with the transformer inductance produces a reactive voltage divider to reduce the voltage on the modem inputs from the mains at the low frequencies of the mains, while still passing the higher-frequency, 10kHz and 20kHz signals unimpeded. We chose a signal transformer with the correct inductance between the terminals and the correct high voltage polymer capacitor by calculating the transfer function, and optimizing the 60Hz blocking and 10kHz passing abilities of the resulting high-pass filter.
The crystal oscillator is expected to operate in parallel resonant mode. The frequency required by the chip is 24MHz ± 100ppm. We chose a crystal [18] which has that frequency and it needs to have capacitors $C_x$ of 33pF. On p.20 of the oscillator’s datasheet it said that the parallel resistor $R_x$ should be 1MΩ, which falls in the typical range of 500kΩ to 2MΩ.

The NCS5650 line driver chip and the supporting circuitry form a 4th order low-pass filter for the output. Instead of calculating the values ourselves to match the frequency we wanted (which could be very difficult to do) we used the values already calculated in the AND8466/D application note called “NCS5650 PLC Filter Design” [19] from ON Semiconductor, the manufacturers of the NCS5650. For the receiver, we will use the resistor and capacitor values given in this application note’s Table 26 on p. 26. For the pull-up resistor on the digital modem output lines, we used the typical resistor value of 10kΩ. The NCS5650 has a digital current limit warning flag. This was used to implement a high current cut-off with an NMOS transistor, in case of faults in the circuit.

### 3.6 Wi-Fi Controller

Wi-Fi is used to communicate between the PLC network and a PC for user control. The chip we have chosen is the Microchip MRF24WB0MA [5]. It is an off-the-shelf component and was chosen because it is inexpensive, easy to use over SPI, and it is interrupt and debug compatible. This chip follows FCC regulations, satisfying our requirements.

### 3.7 Power Monitor

We required the power monitor to measure the average power consumption of the load. We came up with the following specifications: measure the current in series through the 120V AC output line, measure the voltage in parallel through the 120V AC line, send the voltage levels to the MCU in a format suitable for the ADC or put together an analog multiplier to get the power. We resolved that the best option was to detect average levels of current and voltage.

The first option we considered was using off-the-shelf components. We found a chip, the LT2940 Power and Current Monitor [20], that would take as inputs the current and voltage and output the power. We were going to use a typical application circuit that was included in the datasheet (“Fully Isolated AC Power and Current Monitor” on p. 21) which provided full
electrical isolation using two transformers. This seemed like the ideal solution, however there was a problem. Linear Technology provided proprietary software, called LTSpice, for simulating the chip but did not have a SPICE model that we could import into Multisim. We needed this model in order to simulate the chip along with the rest of our PLC device. We began to recreate parts of it using only LTSpice, but the user interface was very poor and we were even unable to obtain output waveforms. In the end, using the software proved to be too difficult. This combined with the fact that we wanted to minimize the use of transformers contributed to the idea of using this chip being finally abandoned.

Then, we thought about another solution. The voltage would be measured using an amplifier, to attenuate the signal, and a buffer. The current could be measured separately using one of several options. The first option would be to use a 1:1 transformer to sense the current on the line between the dimmer and the load. The output of this transformer could be passed to a current mirror before being amplified and sent to the MCU. This idea was rejected when we decided to reduce the amount of transformers we were using due to cost and size. Another method for measuring the current could be using a closed loop Hall Effect sensor or other current monitor using a coil winding such as a Rogowski coil [21]. Yet another way would be to use a current transformer to measure the current. The reason this idea was not pursued is that we did not have a pre-made SPICE model for these components for use in Multisim, and creation of an accurate model would take too much time.

Consequently, we decided to modify the second solution. Rather than using a current transformer, a far simpler and more economical method is to use a small resistor in series with the load. Knowing the resistance and by measuring the voltage across it, we are able to calculate the current using Ohm’s law.

We came up with a new Power Monitor circuit idea using a “superdiode” which can be seen in Figure 3-9 is a diode with an OpAmp. Together, we elaborated on this idea and designed a new circuit.
The transfer function of this circuit can be found as follows:

\[ V_A = A(V_I - V_O) \]

\[ V_O = V_A - V_d \quad \text{or} \quad V_A = V_O + V_d \]

\[ V_O + V_d = A(V_I - V_O) = AV_I - AV_O \]

\[ V_O(1 + A) + V_d = AV_I \]

\[ V_O \frac{(1 + A)}{A} + \frac{V_d}{A} = V_I \]

which as \( A \) approaches infinity becomes

\[ \lim_{A \to \infty} V_I = V_O + 0 \]

\[ V_O = V_I \]

The “superdiode” only provides half-wave rectification, so the period of the resulting waveform will be twice that of a full wave rectified waveform. This idea was abandoned since the input to the OpAmp would be at or above 170V peak, which most OpAmps are not capable of handling.

Here is description of how the final design operates. The average power is read as separate average voltage and current signals. The voltage monitor takes the rectified voltage across the bridge rectifier, filters it to take the average using an RC low-pass circuit, and buffers it using an ON Semiconductor OpAmp [22] to output an accurate voltage to the microcontroller.
The input voltage is reduced by a resistor voltage divider before filtering to prevent high voltages being present across the micro-circuitry. The bridge rectifier is placed on the high voltage side of this voltage divider so that its constant voltage drop of about 2.4V is of minimal impact to the voltage reading. The current is first measured as a voltage over a 100mΩ resistor, and then manipulated in the same manner as the voltage reading to obtain a measurement. Then, both voltage and current inputs are passed to the microcontroller through two of its analog-to-digital converter (ADC) inputs, and multiplied in software to obtain the average power output. Figure 3-10 shows a functional schematic of the power monitor section.

![Power Monitor Schematic](image.png)

**Figure 3-10: Power monitor schematic**

Depending on the load, the current and voltage may not be in phase and this phase could even change over time. To eliminate this issue, the RMS values of voltage and current are measured separately in the Power Monitor and each value is sent to a separate ADC port on the MCU. The MCU multiplies the values in software to obtain the average power.

The Power Monitor has two subsections: one each for voltage and current measurement. They are almost the same in terms of operation. We will discuss both of these beginning with the voltage measurement subsection. The output of the dimmer, which is sent to the load, is a rectified voltage. This voltage is lowered using a resistive voltage divider. This lowered voltage is
measured by filtering out all the AC component of the signal leaving the DC component, and is
then buffered with an OpAmp and sent to the MCU ADC.

What follows are the calculations for the Power Monitor components starting with the
voltage measurement section. The maximum input voltage to the MCU ADC is 3.3V. The
operational amplifier has been setup to provide unity gain therefore its output voltage will follow
the non-inverting terminal voltage. Because of the resistive divider, by voltage division, we have

\[ V_{out} = V^+ = \frac{R_{p3}}{R_{p2} + R_{p3}} V_s \]

Using resistors with 5% tolerance, the output voltage will be at maximum when \( R_3 \) and \( V_s \) are
largest and \( R_1 \) is smallest and this value must not be larger than 3.3V.

\[ V_{out,max} = \frac{1.05R_{p3}}{0.95R_{p2} + 1.05R_{p3}} (120V + 5V) \leq 3.3V \]

Therefore

\[ \frac{0.95R_{p2}}{1.05R_{p3}} \geq \frac{125V}{3.3V} - 1 \]

\[ R_{p3} \leq \frac{0.95R_{p2}}{125V \over 3.3V - 1} \]

If we choose

\[ R_{p2} = 10M\Omega \]

Then

\[ R_{p3} \leq \frac{0.95(10M\Omega)}{125V \over 3.3V - 1} = 257.6k\Omega \]

The next smaller standard resistor value is

\[ R_{p3} = 240k\Omega \]

With these values the maximum and minimum output voltages will be
Using resistors with a 1% tolerance yields

\[ V_{\text{out, max}} = \frac{1.05(240\,\Omega)}{0.95(10\,\Omega) + 1.05(240\,\Omega)}(120 + 5\,V) = 3.23\,V \]

\[ V_{\text{out, min}} = \frac{0.95(240\,\Omega)}{1.05(10\,\Omega) + 0.95(240\,\Omega)}(120 - 5\,V) = 2.44\,V \]

Even with 1% tolerance, the error in the voltage measurement is

\[ \text{error} = \frac{\Delta V_{\text{in}}/2}{V_{\text{in}}} \times 100\% = \frac{(2.99\,V - 2.64\,V)/2}{(2.99\,V + 2.64\,V)/2} \times 100\% = 6.22\% \]

which is too high. A variable potentiometer in series with R could compensate for the error in R. This error can also be accounted for on the microcontroller by multiplying the reading by a correction constant, unique to each node. Finding this constant is experimental since it is dependent on the actual resistor values. This is the option we chose.

A first-order RC low-pass filter is made between the resistor divider and a capacitor with one terminal to ground. This is used to reject all AC components and leave the DC average voltage. The AC is not completely rejected due to limitations in passive analog filters, so the output exhibits a small ripple. This ripple contributes to the total error. The low-pass filter does not completely eliminate the 60Hz causing this ripple. This error cannot be accounted for with the microcontroller since it is time dependent. The total error for the power reading was chosen to be a maximum of 5%. The errors of the voltage and current monitor contribute to the total power error as

\[ (1 + \varepsilon_V)(1 + \varepsilon_I) - 1 = \varepsilon_P \]

where \( \varepsilon_I \), \( \varepsilon_V \) and \( \varepsilon_P \) are the errors in current, voltage, and power respectively. If the current and voltage error are assumed to be equal, the maximum total error for either component is \( \sqrt{(1 + \varepsilon_P)} - 1 = 2.5\% \). The filter was designed to attenuate the 60Hz component by a factor of at least 2.5%, since that it the lowest frequency and primary cause of any output ripple. With a low-
pass filter, any higher frequencies will be significantly more attenuated than the 60Hz component and can be neglected. We solve for

\[
\left| \frac{1}{1 + 2\pi \cdot 60\text{Hz} \cdot 220\text{k}\Omega \cdot C_{p1}} \right| \leq 0.025
\]

to find the required capacitance, which has a nearest standard value of 470nF, although a larger capacitor will give a better result. The minimum total ripple error is 2.35%.

Now, we calculate the current measurement part of the Power Monitor. The current is measured as a voltage across a small resistance. To prevent saturation of the MOSFET in the dimmer, we set this sensing resistor value to

\[
r_{\text{sense}} = R_{p1} = 100\text{m}\Omega
\]

As per our requirements, there will be, at most, 1A going through the sensing resistor

\[
i_{\text{peak}} = 1\text{A}\sqrt{2} = 1.414\text{A}
\]

The maximum \( r_{\text{sense}} \) voltage is

\[
V_{\text{peak}} = i_{\text{peak}} r_{\text{sense}} = 1.414\text{A}(100\text{m}\Omega) = 0.1414\text{V}
\]

The resistor will need to be able to handle at least

\[
P = i_{\text{rms}}^2 r_{\text{sense}} = (1\text{A})^2 (100\text{m}\Omega) = 100\text{mW}
\]

Using these requirements, we chose a Vishay cemented wirewound precision resistor [23] with a 1W power dissipation and 1% tolerance value.

Because of the low-pass filtering capacitor, the OpAmp input is the RMS value of 1A. From simulation, the OpAmp input voltage is

\[
V_1 = 505\text{mV}
\]

The series resistor in the current monitoring part of the circuit (\( R_4 \)) does not require precision since the low-pass filter only requires a maximum cut-off frequency, and larger values will simply increase the accuracy of the circuit.
To find the required gain of the OpAmp in the current monitor, we attempt to scale the output voltage to the full range of the ADC while accounting for tolerance error. The largest possible gain is

\[ A_{max} = 1 + \frac{1.01R_{p7}}{0.99R_{p6}} \]

The maximum input voltage multiplied by the gain must be less than the full range of the ADC, therefore \(0.505V(A_{max}) \leq 3.3V\)

Simplifying

\[ \frac{R_{p7}}{R_{p6}} \leq 5.4251 \]

If we choose

\[ R_{p6} = 10k\Omega \]

Then

\[ R_{2} \leq 5.4251(10k\Omega) = 54.251k\Omega \]

The next smaller standard resistor value is

\[ R_{p7} = 53.6k\Omega \]

The current monitor uses the same filter as the voltage monitor since it results in the same ripple error. The total uncorrectable error (nonlinear error) is calculated only with the ripple errors and is 4.76%. The linear errors are accounted for as scaling factors on the microcontroller. The transient analysis graph in Figure 3-11 shows results of the voltage and current readings (which happen to be very close in this case) at the bottom and the power reading at the top. The voltage and current are in phase which results in maximum power ripple error. The mean of the power reads correctly as 1.95V. The ripple is peak 35mV, for an error of 1.8%. This ensures correct operation.
3.8 Board Layout

Originally, we planned to build all of the components except the modem circuitry on perfboard because it can handle high voltage, but does not support surface mount chips. Perfboard was chosen over breadboard since it provides more reliable unit where wires will not fall out, and breadboards are usually only rated for 15V. While working on building the perfboard modules, we found out that if we made a mistake on soldering the components it is hard for us to fix the problem. Therefore we decided to build all the low-voltage component on the breadboard and high-voltage component on the perfboard. Doing so reduced our perfboard design by more than half of the time it took to complete one board. We can easily move or replace the components (resistors, capacitors, etc.) around on breadboard in case of errors.

For the layout of the components on the perfboard, we intended to keep all the heat-producing elements (e.g. regulator, high value capacitor) together on one side of the perfboard and keep all the sensitive elements (e.g. microcontroller, Power Monitor) away from the heat-producing elements which will reduce the effects of the undesirable noise.

The reason we need to build a PCB is because our PLC line driver and Wi-Fi chip are surface mount components. We managed to find a socket for the modem chip to facilitate the soldering process. For the line driver chip there is no socket for it. Since our group members have no experience making PCB’s, we needed to do research to learn the basics of making a PCB
using Ultiboard. Due to the small size (4.3mm x 4.3mm) of the line driver, we were unable to solder it onto the PCB. The CNC router used to manufacture the PCB was unable to accurately cut traces small enough for the chip. As shown in Figure 3-12, we burnt off the trace while trying to solder line driver chip onto the PCB.

![Burnt trace image](image)

*Figure 3-12: Burnt trace*
4.1 Graphical User Interface

The GUI uses a model-view-controller design and has been written using the Java programming language in Eclipse. The view component is comprised of a list-box to select the current node, a power display, and a dimmer setting slider control. The model component stores the most up-to-date data about each connected node including name, ID number, power usage, and dimmer setting. The controller consists of the interface to the master node which updates the node data as messages are sent and received over Wi-Fi. Figure 4-1 shows a simplified class diagram of the GUI.

![Simplified UML class diagram](image)

The GUI is an important part of our design; it serves as the interface between the user and our device. It provides a way for the user to control the power sent to the load and monitor the power usage. A view of the GUI can be seen in Figure 4-2. Section 1 in the figure contains a list of all the active nodes in the system. As nodes are added to the system, they are automatically added to this list. The data presented in Sections 2 and 3 of the figure are those of the node selected in this list. Section 2 contains a graphical display of the power level of the currently selected node as well as its numerical value below it. Section 3 contains a slider that can be used by the user to control the voltage level sent to the currently selected node. At the top, a menu contains additional options for further control. It allows the user to rename nodes to a name that is more meaningful to them. This is also where one can find the “Connect” button. When the
application is run for the first time, the user must type in his or her IP address, as described in Section 3.6 (Wi-Fi modem). If a disconnection occurs during operation and data is no longer refreshed, the button can be clicked to reconnect to the modem. We included an “About” section giving some details about the software and why it was created.

![Graphical user interface window](image)

**Figure 4-2: Graphical user interface window**

The GUI class is where the main application is run. It starts by creating the modem interface ModemIF, proceeds to instantiate the Window class, before finally updating the power in an infinite loop.

The Window class is what creates the window on the screen containing all the components seen in the Figure 4-2 above. It uses the Java Swing, a GUI widget toolkit, to create the list of nodes and the slider and it contains custom classes to create and animate the power gauge. The Window contains listeners to react to event-driven components. These listeners detect and handle menu options, react to slider changes by sending the updated dimmer level to the selected node through PLC, and react to a new node selection by updating the display to show its power and dimmer levels.
The ModemIF class is the interface between the GUI which is seen by the user and the Wi-Fi modem which connects to the network through the master node. It permits the GUI to send dimmer data to the nodes and to receive power updates. It runs a parallel thread to handle networking to reduce latency and avoid unnecessary packet loss. NodeStatusCallback is a callback for the GUI which is called by ModemIF when nodes are detected to have lost connection or have been added to the network.

The NodeData class simply contains the variables describing a node in the network such as power level, dimmer level, ID number, and node name as well as some simple methods to allow the Window to modify these values.

The full code for the various GUI classes is included as part of Appendix C.

4.2 Protocols

The slave and master node send regular dimmer and power updates to each other through a client-server architecture. The master node sends dimmer updates to all active slaves only and the slaves send its current power reading to the master node. The master node communicates to the PC to relay the PLC network information to the user using user datagram protocol (UDP).

When a slave node powers on, it attempts to send connection request messages (CRQST) until the master node responds with and acknowledge message (ACK). This way, the slave nodes and master node can be plugged in and activated in any order. Once the ACK message is received from the master node, the slave waits for dimmer messages. When a dimmer message is received, it sends its power data to the master. The master node sends regular dimmer messages to all nodes that it acknowledges as active. Figure 4-3 shows the operation of the slave and master node in reference to the PLC protocol.
Figure 4-3: PLC protocol flowcharts

The PLC packet formats are shown in Figure 4-4. All packets are four bytes long. The first byte is the node ID which the following data corresponds to. The second byte identifies the packet type. A connection request (CRQST) packet does not use the following two bytes. The DIMMER packet uses one byte to identify one of 21 dimmer levels. The POWER packet uses two bytes to identify the direct multiplication of the current and voltage monitor ADC readings.

PLC Packet Formats

<table>
<thead>
<tr>
<th>Op Code</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node ID</td>
<td>CRQST</td>
</tr>
<tr>
<td>Node ID</td>
<td>DIMMER</td>
</tr>
<tr>
<td>Node ID</td>
<td>POWER</td>
</tr>
</tbody>
</table>

Figure 4-4: PLC protocol packet formats
The master node assumes that there has been a disconnection with a slave node when it does not receive a power update after a timeout period. It then removes that particular node from its active nodes list, and no longer communicates with that node until it attempts a new connection. The slave nodes also expect regular dimmer updates from the master node. Upon a timeout condition, the node reverts back to the beginning of its procedure in attempting a reconnect of the master node.

The PC initiates a PC to master node connection over Wi-Fi by sending an initialization (INIT) packet. Once the master node receives an INIT packet, it begins sending ACK packets. The ACK packet contains a two byte bit field identifying which nodes are active. Once the PC receives an ACK packet, it begins sending dimmer updates and a connection is established. Whenever the master node receives a dimmer packet, it responds with a power packet. The PC sends dimmer updates twice a second so to not overload the master in processing the Wi-Fi data. Figure 4-5 and Figure 4-6 show the operation of the protocol in more detail.

Master Node Wi-Fi Protocol

![Master Node Wi-Fi Protocol Flowchart]

Figure 4-5: Wireless protocol master node flowchart
PC Wi-Fi Protocol

![Flowchart Diagram]

**Figure 4-6: Wireless protocol PC flowchart**

The UDP packet formats are shown in Figure 4-7. PC Packets are 17 Bytes long and include one byte to identify the packet type, and 16 bytes of data. A DIMMER packet contains one byte specifying one of 21 dimmer levels for each possible node and referenced by the node IDs. The master node packets are 33 bytes long, starting with a one byte packet type. In an ACK packet, the first two data bytes are bit field specifying which nodes are active. In a POWER packet, each pair of bytes specifies the power reading from directly multiplying the current and voltage ADC readings, which is scaled before display to the user.

**PC UDP Packet Formats**

<table>
<thead>
<tr>
<th>Op Code</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>16 Bytes Unused</td>
</tr>
<tr>
<td>DIMMER</td>
<td>Dimmer 0 Dimmer 1 ... Dimmer 15</td>
</tr>
</tbody>
</table>

**Master Node UDP Packet Formats**

<table>
<thead>
<tr>
<th>Op Code</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Active 0:7 Active 8:16 30 Bytes Unused</td>
</tr>
<tr>
<td>POWER</td>
<td>Power 0 L Power 0 H ... Power 15 L Power 15 H</td>
</tr>
</tbody>
</table>

**Figure 4-7: Wireless protocol packet formats**
4.3 Microcontroller

Aside from the PLC and Wi-Fi communications, the microcontroller must set the DAC and read the power from two ADCs. The ADC’s read 10-bit values across two bytes for the voltage and current. These values are multiplied directly on the microcontroller and scaled on the PC. The one byte dimmer value received from the PC is directly set to the DAC.

The code for the power reading and DAC setting has been completed and tested using **MPLAB X**. The master node’s code for UDP communication and the PLC protocol is complete but untested and requires debugging. The slave node code for UDP and PLC, and UART initialization is incomplete. The Wi-Fi initialization is not complete due to poor documentation provided for the required proprietary libraries. See Appendix C for code.
Chapter 5 – Testing and Results

5.1 Modular Testing

Hardware was tested separately in sub units to ensure correct operation and ease of debugging, and later combined to show correct operation of all circuits together. Output analysis of each component was done only to the individual sub units separately. The power supply, modem, and Wi-Fi were not tested for reasons discussed later. The analog modem line coupling was tested as this was self-designed.

5.1.1. Graphical User Interface

The modem coupling must pass 10kHz and 20kHz signals between nodes over the power lines, but impede the high voltage on the lines from causing an over-volt condition on the low voltage side of the circuit. The circuit used to test the design is shown in Figure 5-1.

![Figure 5-1: Modem line coupling circuit](image)

A function generator was used to supply both the signal (voltage source on the left) and the mains signal (voltage source in the middle). This generator has an output impedance of 50Ω, and an output voltage limit of ±10V, so the results must be scaled to get the actual expected results when testing the mains blocking.
Passing both 10kHz and 20kHz obtained the same responses. The signal on the mains is distorted due to the signal transformer, but still retains a strong 10kHz or 20kHz component, and is easily differentiable from each other in the frequency domain. Input filters on the modem should filter out most of this distortion, and the signal should still be detectable. From modem to modem, the signal is attenuated by a factor of 0.24 resulting in a 10kHz amplitude of 1.44V peak to peak with a 6V peak to peak input. This is assumed to be a sufficient signal strength since it is stated that the modem can receive 100mV peak signals reliably. However, it is not explicitly stated what the minimum signal to noise ratio for a receivable signal in any documentation related to the modem.

The input to the receiving side of the modem must always be less than 3.3V and above 0V including both the 1.44V peak to peak signal and the 170V peak, 60Hz mains source since safety clamping diodes are present on the input circuit, otherwise the signal will be strongly distorted. 60Hz suppression tests resulted in a 60Hz attenuation of 0.001. This scales to a maximum of 170mV peak 60Hz signal on the modem side of the coupling. This should pose no issue since the maximum total peak to peak signal on the modem side will be 1.44V + 170mV + 170mV = 1.78V, which is less than the 3.3V supply, and the modem receiver has a high-pass filter to remove the 60Hz signal.

5.1.2. PWM Controller Testing

The PWM controller was tested using a voltage level input from a bench top lab power supply to simulate the microcontroller DAC. The output was measured at the position of the gate of QD3, the high current load switching power MOSFET. A possible inaccuracy with this setup are the lower input impedance of the power supply compared to the DAC. This should not affect the operation due to the voltage buffer at the input to the dimmer circuit having an input impedance of greater than 200kΩ, and the output impedance of the DAC being less than 5kΩ. The switch was tested later connected to the PWM controller, so problems with the output of the PWM controller was not considered in this test. The PWM controller was implemented on breadboard since it is an exclusively low voltage, low current circuit.

The power supply output in all test cases 5.1V peak. We first measured the 0% threshold and 100% threshold levels. We then took 10 intermediate dimmer level points between the two thresholds. An oscilloscope was used to measure the mean output voltage. This mean corresponds to the dimming level of the PWM module. Table 5-1 lists the data collected for these tests.
Table 5-1: PWM Linearity Raw Results

<table>
<thead>
<tr>
<th>DAC (V)</th>
<th>Mean (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 1.74V</td>
<td>35mV</td>
</tr>
<tr>
<td>1.74</td>
<td>35mV</td>
</tr>
<tr>
<td>2.03</td>
<td>535mV</td>
</tr>
<tr>
<td>2.32</td>
<td>1.10</td>
</tr>
<tr>
<td>2.61</td>
<td>1.67</td>
</tr>
<tr>
<td>2.90</td>
<td>2.22</td>
</tr>
<tr>
<td>3.20</td>
<td>2.72</td>
</tr>
<tr>
<td>3.49</td>
<td>3.20</td>
</tr>
<tr>
<td>3.78</td>
<td>3.77</td>
</tr>
<tr>
<td>4.07</td>
<td>4.2</td>
</tr>
<tr>
<td>4.36</td>
<td>4.61</td>
</tr>
<tr>
<td>4.65</td>
<td>5.1</td>
</tr>
<tr>
<td>&gt; 4.65</td>
<td>5.1</td>
</tr>
</tbody>
</table>

The dimmer accuracy is measured as a factor of linearity since, once a linear transfer function is achieved, accurate dimmer control through the microcontroller is a matter of multiplication by a constant and adding of an offset. A linear system also allows maximum range through the DAC since the DAC is also linear. To measure the linearity, a best fit line was calculated through MATLAB and maximum and RMS error was calculated. Table 5-2 lists resulting data from this analysis, and Figure 5-2 shows a plot of the data.

Table 5-2: PWM Linearity Analysis Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Fit Line Slope</td>
<td>1.7506</td>
</tr>
<tr>
<td>Best Fit Line Offset</td>
<td>-2.943</td>
</tr>
<tr>
<td>Maximum Absolute Error</td>
<td>97.3mV</td>
</tr>
<tr>
<td>RMS Error</td>
<td>67.6mV</td>
</tr>
<tr>
<td>0% Duty Cycle Input Threshold</td>
<td>1.74V</td>
</tr>
<tr>
<td>100% Duty Cycle Input Threshold</td>
<td>4.65V</td>
</tr>
<tr>
<td>Average Percent Linearity</td>
<td>2.7%</td>
</tr>
</tbody>
</table>
With this data, we calculated the maximum number of dimmer levels, by calculating the number of steps the DAC provides between the two threshold voltages. The DAC provides 32 levels between 0 and 5V, and 21 different dimmer levels between 0% and 100% duty cycle. This satisfies our requirement of a minimum of 10 levels between 0% and 100% duty cycle. The linearity error added with the DAC 1/2LSB error is less than 5%, which is acceptable since an exact quantized dimmer level is not required by a user dimming a lamp for home use.

The user of the control application inputs dimmer levels as values between 0.0 and 1.0. This value is used to calculate the DAC value on the microcontroller from 0 to 31 using the test result data. The DAC value is calculated as

\[ DAC = 21 \ast \text{dimmer} + 10 \]

Figure 5-3 shows oscilloscope outputs displaying correct operation of the PWM controller. Figure 5-4 shows the setup used to test the circuit.
(a) 0% duty cycle  (b) Small duty cycle  
(c) Medium duty cycle  (d) Large duty cycle  
(e) 100% duty cycle  

Figure 5-3: PWM controller outputs

Figure 5-4: PWM controller testing circuit
5.1.3. Switch Testing

Due to safety concerns, we could not connect the system to the 120VAC mains. Instead, we tested the system powering the load with a 20Vpp, 50Ω AC function generator, and 40VDC from a bench top power supply. This proves correct operation with AC voltage and voltages much higher than the circuit control voltage (5V). When powering the circuit with the function generator, the load voltage was smaller than 20Vpp due to significant loading between the 50Ω signal source and the 82Ω load, and voltage drop across the bridge rectifier (about 2.4V).

There were significant problems in testing the switch due to common grounding of the test equipment. The function generator ground and oscilloscope ground in the equipment used are connected to earth, and therefore share a common ground. Connecting both pieces of equipment to the circuit unintentionally placed two ground points on the bridge rectifier, allowing current to pass through it on only half a cycle. The common ground between the oscilloscope channels posed a similar problem when trying to measure a differential voltage across the load.

Figure 5-5 shows the resulting test setup for the circuitry, allowing correct measurements to be made. The function generator DC component was removed from the rest of the circuit through an isolation transformer. The power transformer ordered for stepping the voltage down for the power supply was used as an isolation transformer. The transformer is compatible with breadboard, making easier to wire for a test than the isolation transformers bought for mains isolation. It has two electrically isolated windings on the primary side of an equal turns ratio, and therefore does not step down the voltage, when hooked up to isolate. Both oscilloscope channels are connected to either side of the load, and the oscilloscope math functions are used to get a differential reading of the voltage across the load.
Figure 5-5: Switch testing setup

With this setup, correct operation is observed. The smoothing of the PWM signal should be noted, and is due to the low-pass filtering effects of the gate capacitance of the switching transistor, and leakage inductance of the isolation transformer. Figure 5-6 shows correct operation of the switching circuit at different timescales with an AC input, showing correct operation with AC input at a large timescale and correct switching at a small timescale.
5.1.4. Power Monitor Testing

While testing the voltage monitor, resistors $R_{P1}$ and $R_{P2}$ were replaced with 10Ω and 220kΩ resistors, respectively. Since we were not connecting to the mains, these resistors were replaced to raise the voltage and current so that meaningful results could be obtained. Replacing these resistors does not have any effect on the circuit except for scaling the voltage levels. The readings must be scaled after analysis back to original value resistors to get the expected results in a case where it is connected to the mains. The load used to test with is a 150Ω resistor and input is a 20V$_{pp}$, 50Ω function generator. In both the current monitor and voltage monitor, the output is expected to be linear with the actual mean input, and performance is measured against a best fit line. Scaling and offset factors are then computed to be used in software to compute the actual power dissipation in the load.

5.1.5. Voltage Monitor Testing

The input voltage is divided by half and input into the voltage monitor circuit. Input peak voltage is 5.36V, and the voltage input to the filter is 2.8V peak. The actual mean voltage is measured using an oscilloscope’s mean function and compared to the output voltage of the voltage monitor. Table 5-3 shows data points collected from the tests.
Table 5-3: Voltage Monitor Linearity Raw Results

<table>
<thead>
<tr>
<th>Duty Cycle</th>
<th>Mean (Osc)</th>
<th>Vout</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.287</td>
<td>0.325</td>
</tr>
<tr>
<td>25</td>
<td>0.355</td>
<td>0.407</td>
</tr>
<tr>
<td>30</td>
<td>0.43</td>
<td>0.481</td>
</tr>
<tr>
<td>40</td>
<td>0.57</td>
<td>0.624</td>
</tr>
<tr>
<td>50</td>
<td>0.708</td>
<td>0.79</td>
</tr>
<tr>
<td>60</td>
<td>0.849</td>
<td>0.941</td>
</tr>
<tr>
<td>70</td>
<td>0.985</td>
<td>1.1</td>
</tr>
<tr>
<td>75</td>
<td>1.05</td>
<td>1.17</td>
</tr>
<tr>
<td>80</td>
<td>1.12</td>
<td>1.24</td>
</tr>
<tr>
<td>85</td>
<td>1.42</td>
<td>1.58</td>
</tr>
</tbody>
</table>

The maximum ripple is less than 74mV with the highest mean value signal. This is a 2.3% error. This data is compared to a best fit line and maximum error is calculated. Table 5-4 shows analysis data, and Figure 5-7 plots the best fit line and actual data for visual comparison.

Table 5-4: Voltage Monitor Linearity Analysis Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Fit Line Slope</td>
<td>1.1075</td>
</tr>
<tr>
<td>Best Fit Line Offset</td>
<td>0.00504</td>
</tr>
<tr>
<td>Maximum Error</td>
<td>12.3mV</td>
</tr>
<tr>
<td>RMS Error</td>
<td>5.6mV</td>
</tr>
<tr>
<td>Average Linearity Error</td>
<td>0.75%</td>
</tr>
</tbody>
</table>
The linearity error added with the ripple error gives a total error for the voltage monitor is 3.0%. To scale the results of this test to the actual case, a scaling factor of 1.3631 multiplied by the actual readings is needed. To compute the actual voltage given the ADC reading, only a scaling factor is needed because the offset is very small. The ADC gives values from 0 to 1023 with voltage input from 0V to 5.0V. This data results in a required voltage scaling factor of 0.0040 to give the voltage mean from the ADC input.

5.1.6. Current Monitor Testing

The peak to input voltage on the current sense resistor is 120mV. The actual mean voltage is measured using an oscilloscope’s mean function. This is compared to the current monitor output. Table 5-5 shows data collected during testing.
Table 5-5: Current Monitor Linearity Raw Results

<table>
<thead>
<tr>
<th>DC</th>
<th>Osc Mean (mV)</th>
<th>Vo (mV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>27</td>
<td>211</td>
</tr>
<tr>
<td>30</td>
<td>40.7</td>
<td>324</td>
</tr>
<tr>
<td>40</td>
<td>54.0</td>
<td>429</td>
</tr>
<tr>
<td>50</td>
<td>68.0</td>
<td>535</td>
</tr>
<tr>
<td>60</td>
<td>81.9</td>
<td>643</td>
</tr>
<tr>
<td>70</td>
<td>95</td>
<td>748</td>
</tr>
<tr>
<td>80</td>
<td>110</td>
<td>860</td>
</tr>
</tbody>
</table>

The maximum output ripple at 80% duty cycle is 24.4mV peak to peak and the minimum ripple at 20% duty cycle is 6.16mV peak to peak. This results in a maximum ripple error of 2.9%. This was compared to a best fit line to test for linearity and compute scaling factors. Table 5-6 shows the analysis results and Figure 5-8 plots the best fit line and the measurement data for visual comparison.

Table 5-6: Current Monitor Linearity Analysis Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Fit Line Slope</td>
<td>7.8046</td>
</tr>
<tr>
<td>Best Fit Line Offset</td>
<td>4.3357</td>
</tr>
<tr>
<td>Maximum Error</td>
<td>4.06mV</td>
</tr>
<tr>
<td>RMS Error</td>
<td>2.51mV</td>
</tr>
<tr>
<td>Average Linearity Error</td>
<td>0.58%</td>
</tr>
</tbody>
</table>
5.1.7. Power Monitor Results

The total power error is 6.40%. We expected a 5% error. The majority of the error is from ripple error, which can be easily reduced by increasing the capacitance of the power monitor filters. The ADC values will be multiplied on the microcontroller without scaling since the microcontroller lacks a floating point arithmetic logic unit (ALU). The scaling factor will be applied before display to the user on the PC. This power scaling factor is the voltage scaling factor multiplied by the current scaling factor for a value of $0.0102 \times 0.004 = 40.8 \times 10^{-6}$. 

Figure 5-8: Voltage monitor output vs. actual mean and best fit line

The linearity error added with the ripple error result in a total error of 3.4%. The calculated scaling factor to compute expected values in actual application is 0.833. The scaling factor required to convert the ADC value into a mean current value is 0.0102.
5.2 Analog Side Full Test

The microcontroller, PWM controller, load switch, and power monitor circuits were combined and tested together. The Wi-Fi, modem, and power supply were not tested in this final test. The microcontroller read its ID number set by the DIP switch. The DAC on the microcontroller was programmed to output a triangular wave from 0V to 5V to show that all dimmer levels are able to be output without having to control it through the PC, since the Wi-Fi interface was incomplete. The load was simulated with an 82Ω resistor. The power supply was replaced with a bench-top lab power supply outputting 5V for the system components. The load was tested first with 20Vpp AC from a 50Ω function generator to show correct operation with an AC input voltage. The voltage was reduced at the load due to the voltage division effect of the function generator’s 50Ω source impedance and the 82Ω load. To prove that the system can switch higher voltages, a bench top lab power supply was used to input 40VDC. The microcontroller DAC drove the PWM controller input. The PWM controller drove the switch input. The power monitor outputted into the microcontroller ADC’s. Figure 5-9 shows output of a small part of the load waveform being supplied with the function generator and the PWM being controlled by the microcontroller using a triangular wave. The output shows a full range of duty cycles from and including 0% and 100% on.

![Figure 5-9: Analog full side test output – DAC controlling load voltage with PWM at varying duty cycles](image)

Figure 5-10 shows the setup for the full test, including three bread boards and two perfboards.

![Figure 5-10: Analog full side test setup](image)

This testing showed correct operation of the analog side circuitry and interface to the microcontroller.

### 5.3 Incomplete Testing

The power supply was not tested at the time of writing the report due to safety concerns. The power supply requires a direct connection to the 120VAC mains. A grounded enclosure is required to safeguard against touching the high voltage lines and accidental shorting. We were not able to include a grounded enclosure due to time constraints. The lack of grounded enclosure also limited our ability in testing the dimmer and power monitor circuits, so approximations were used as was discussed earlier.

The modem board was not able to be manufactured due to problems with printed circuit board (PCB) manufacture. Three PCB’s were manufactured with a CNC router at the University of Manitoba. The pads for IC7 (NCS5650) in a QFN-20 package could not be routed correctly, preventing correct soldering. Attempting to solder the chip damaged the board. The completed board is shown in Figure 5-11. Some incorrectly routed pads can be seen at the bottom center in a row of nine pads, two are connected to the ground plane. These could not be fixed.
This chip is required for use with the AMIS49587 modem and only comes in the QFN-20 package. The only alternative to this problem is getting the PCB manufactured at a professional company. The costs at such companies are prohibitive for this project, and would total near $500. The pads and traces for the socket for IC6 (AMIS49587) were auto-routed in the PCB design software, Ultiboard. The software placed pads on both sides of the PCB which made soldering pads underneath very difficult to solder. Figure 5-12 shows the pads and the socket for IC6. The top pads must be connected to the bottom. To correct this issue, the PCB would need to be redesigned with all pads on one layer and vias would be used to connect the pins to the other layer outside the perimeter of the socket. The only practical solution to correct these problems for our project is to replace the modem with another which comes in easier to use packages. We did not have the time or budget to do this.
The Wi-Fi card was not able to be tested, but was manufactured. All interfacing between the microcontroller and the Wi-Fi is only possible with the Microchip Application Libraries (MAL). We found the documentation provided by Microchip for MAL is incomplete in regards to the Wi-Fi interface. However, documentation for the UDP stack is acceptable and we implemented untested internet master to PC interface code for the master node. Figure 5-13 shows with Wi-Fi board attached to the microcontroller board. The PC to master node wireless protocol was tested and verified to work correctly using a Java simulation of the master node’s side of the UDP communication on a second PC over a wireless LAN.
Chapter 6 – Recommendations & Conclusion

6.1 Recommendations

This project entailed much more work than we had originally anticipated. The amount of work is more suitable for a larger team. A large amount of time was spent being unsure of how to exactly approach many of the design problems encountered, and having to modify requirements of components after a large amount of design work was done. This could have been circumvented by a more thorough literature review and obtaining advise from people more experienced in power electronic design. The power supply design could have been excluded by purchasing readily available power supply modules, saving some time.

A large portion of time was spent in producing a PCB which is manufacturable by the equipment available at the university, while having no prior experience in PCB design. Not having PCB’s to mount the PLC modem and Wi-Fi card on, a large part of the code base could not be tested or debugged, which is an integral part of software development. We could therefore not complete the software as we would have liked. The PCB design could have been avoided completely by choosing a PLC modem which comes in a DIP or similar package.

The Wi-Fi card we chose from Microchip requires the proprietary Microchip Application Libraries. The documentation for the MAL is poor in regards to the Wi-Fi connection and seemingly lacking in required files as of the versions used. The library could not be worked around since the actual SPI commands to the Wi-Fi card are hidden and are not documented. Choosing another Wi-Fi card option with a more open development paradigm may be easier to work with.

6.2 Future Work

An analog side prototype has been built and tested and operates correctly. This is not ready for use, and requires being put into a single module along with the digital side component, inside a protected, grounded case for safety concerns. It is suggested to design a full PCB to place all components on, but this PCB must be manufactured with an etching process rather than a CNC router due to tight tolerances. A test using the actual 120VAC mains is recommended, but correct operation is very likely as completed tests suggest. The modem and Wi-Fi code are
currently in an incomplete state and must be tested once a working modem and Wi-Fi module can be connected to the microcontroller.

The project can be expanded in several methods. The GUI software on PC could be ported to the Android or IPhone platform to give the user easier and more mobile control of their home power usage. If an internet server is available, a user can control and monitor their home power usage from miles away, as long as a Wi-Fi or GSM internet connection is available. If the modem is replaced with another with a higher data rate, the PLC system can be used for a distributed serial communication system, such as Ethernet or Wi-Fi. The current PLC modem used is not capable of such data rates. The accuracy of the power monitor could be easily improved with a better input filter, and even further improved with a non-linear line segment approximation compensation on the microcontroller for both current and voltage readings. Using a different method of isolation from the mains, such as opto-isolators, the isolation transformer in the dimmer/power monitor section can be eliminated. The total current output can then be raised above 10A at about the same price point if the mains switching MOSFET is replaced with a larger one (or another in parallel).

6.3 Conclusion

The project was not completed, but the hardware design was designed and tested to show correct functionality. Part of the microcontroller software was completed, and the PC control application was completed and tested. The project ran into several large obstacles, limiting our success. With more time, the project is still feasible.
## Appendix A – Parts List

### PLC Modem

<table>
<thead>
<tr>
<th>Name</th>
<th>Component</th>
<th>Qty</th>
<th>Supplier / Part No</th>
<th>Manufacture / Part No</th>
<th>Unit Cost ($CAD)</th>
<th>Total Cost ($CAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMD1</td>
<td>47uF, 20%, Capacitor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CMD2</td>
<td>470pF, 10%, Capacitor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CMD3</td>
<td>1nF, 10%, Capacitor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CMD4</td>
<td>1uF, 20%, Capacitor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CMD5</td>
<td>33pF, 10%, Capacitor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CMD6</td>
<td>33pF, 10%, Capacitor</td>
<td>6</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CMD7</td>
<td>100nF, 20%, Capacitor</td>
<td>6</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CMD8</td>
<td>68pF, 10%, Capacitor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CMD9</td>
<td>1uF, 20%, Capacitor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CMD10</td>
<td>1.5nF, 20%, Capacitor</td>
<td>6</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CMD11</td>
<td>470nF 20%, polypropylene metallized, Capacitor</td>
<td>3</td>
<td>Digikey 399-5869-ND</td>
<td>KEMET Electronics Corporation</td>
<td>0.69</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>Schottky Diode</td>
<td>6</td>
<td>Digikey SD101CVSCT-ND</td>
<td>Vishay Semiconductor SD101C-TR</td>
<td>0.70</td>
<td>4.20</td>
</tr>
<tr>
<td>D3</td>
<td>Zener Diode</td>
<td>6</td>
<td>Digikey 1N5221BCT-ND</td>
<td>Vishay Semiconductor 1N5221B-TR</td>
<td>0.53</td>
<td>0</td>
</tr>
<tr>
<td>D4</td>
<td>Zener Diode</td>
<td>6</td>
<td>Digikey 1N5221BCT-ND</td>
<td>Vishay Semiconductor 1N5221B-TR</td>
<td>0.53</td>
<td>0</td>
</tr>
<tr>
<td>Name</td>
<td>Component</td>
<td>Qty</td>
<td>Supplier / Part No</td>
<td>Manufacture / Part No</td>
<td>Unit Cost (C$)</td>
<td>Total Cost (C$)</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------</td>
<td>-----</td>
<td>---------------------</td>
<td>-------------------------------</td>
<td>----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>D6 D7</td>
<td>Schottky Diode</td>
<td>6</td>
<td>Digikev SD101CVSCT-ND</td>
<td>Vishay Semiconductor SD101C-TR</td>
<td>0.70</td>
<td>4.20</td>
</tr>
<tr>
<td>D8</td>
<td>Trisil</td>
<td>3</td>
<td>Digikev 497-12723-1-ND</td>
<td>STMicroelectronics TPA220</td>
<td>1.07</td>
<td>0</td>
</tr>
<tr>
<td>IC6</td>
<td>PLC Modem</td>
<td>3</td>
<td>Mouser 863-AMIS49587C5871G</td>
<td>ON Semiconductor AMIS49587C5871G</td>
<td>9.64</td>
<td>0</td>
</tr>
<tr>
<td>IC7</td>
<td>PLC Line Driver</td>
<td>3</td>
<td>Digikev NCS5650MNTXG OSCT-ND</td>
<td>ON Semiconductor NCS5650MNTXG</td>
<td>4.29</td>
<td>0</td>
</tr>
<tr>
<td>QMD1</td>
<td>N-Channel MOSFET</td>
<td>3</td>
<td>Digikev NTD4906N-35GOS-ND</td>
<td>ON Semiconductor TD4906N-35G</td>
<td>0.61</td>
<td>0</td>
</tr>
<tr>
<td>RMD1</td>
<td>3.32kΩ, 1%, 1/4W, Resistor</td>
<td>3</td>
<td>Digikev RNF14FTD3K32CT-ND</td>
<td>Stackpole Electronics Inc RNF14FTD3K32</td>
<td>0.16</td>
<td>0</td>
</tr>
<tr>
<td>RMD2</td>
<td>10.7kΩ, 1%, 1/4W, Resistor</td>
<td>3</td>
<td>Digikev RNF14FTD10K7CT-ND</td>
<td>Stackpole Electronics Inc RNF14FTD10K7</td>
<td>0.16</td>
<td>0</td>
</tr>
<tr>
<td>RMD3</td>
<td>8.25kΩ, 1%, 1/4W, Resistor</td>
<td>3</td>
<td>Digikev RNF14FTD8K25CT-ND</td>
<td>Stackpole Electronics Inc RNF14FTD8K25</td>
<td>0.16</td>
<td>0.0</td>
</tr>
<tr>
<td>RMD4</td>
<td>4.22kΩ, 1%, 1/4W, Resistor</td>
<td>3</td>
<td>Digikev RNF14FTD4K22CT-ND</td>
<td>Stackpole Electronics Inc RNF14FTD4K22</td>
<td>0.16</td>
<td>0</td>
</tr>
<tr>
<td>RMD5</td>
<td>10kΩ, 1%, 1/4W, Resistor</td>
<td>3</td>
<td>Digikev S10KCACT-ND</td>
<td>Stackpole RNMF14FTC10K0</td>
<td>0.14</td>
<td>0.0</td>
</tr>
<tr>
<td>RMD6</td>
<td>8.45kΩ, 1%, 1/4W, Resistor</td>
<td>3</td>
<td>Digikev RNF14FTD8K45CT-ND</td>
<td>Stackpole Electronics Inc RNF14FTD8K45</td>
<td>0.16</td>
<td>0</td>
</tr>
<tr>
<td>RMD7</td>
<td>10Ω (5W), 5% Resistor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RMD8</td>
<td>33kΩ, 5% Resistor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RMD9</td>
<td>10kΩ, 5% Resistor</td>
<td>12</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RMD10</td>
<td>1MΩ, 5% Resistor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Name</td>
<td>Component</td>
<td>Qty</td>
<td>Supplier / Part No</td>
<td>Manufacture / Part No</td>
<td>Unit Cost ($CAD)</td>
<td>Total Cost ($CAD)</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------</td>
<td>-----</td>
<td>-----------------------------------------</td>
<td>-------------------------------</td>
<td>-----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>RMD14</td>
<td>22kΩ, 5% Resistor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RMD15</td>
<td>11kΩ, 5% Resistor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RMD16</td>
<td>1MΩ, 5% Resistor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RMD17</td>
<td>10kΩ, 5% Resistor</td>
<td>12</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>S2</td>
<td>Modem Socket</td>
<td>3</td>
<td>Digikey AE1055-ND</td>
<td>Assmann WSW A-CCS28-Z-R</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>T3</td>
<td>Signal Transformer</td>
<td>3</td>
<td>Digikey 811-2655-ND</td>
<td>Murata Power Solutions Inc 78615/2C</td>
<td>1.67</td>
<td>0</td>
</tr>
<tr>
<td>XTAL</td>
<td>Crystal Oscillator 24MHz</td>
<td>3</td>
<td>Digikey X986-ND</td>
<td>ECS Inc. International ECS-240-20-1X</td>
<td>0.75</td>
<td>0</td>
</tr>
<tr>
<td><strong>Dimmer</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD1</td>
<td>2.2nF, 5% Ceramic Capacitor</td>
<td>3</td>
<td>Digikey 445-4742-ND</td>
<td>TDK Corporation FK28C0G1H222J</td>
<td>0.36</td>
<td>0</td>
</tr>
<tr>
<td>CD2</td>
<td>10µF, 20%, 300V Capacitor</td>
<td>3</td>
<td>Digikey BC2752-ND</td>
<td>Vishay VY2102M29Y5UG63V7</td>
<td>0.30</td>
<td>0</td>
</tr>
<tr>
<td>CD3</td>
<td>6.8nF, 20% Ceramic Capacitor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HS1</td>
<td>Heat Sink</td>
<td>3</td>
<td>DigiKey A10764-ND</td>
<td>Assmann WSW V7235-T</td>
<td>0.36</td>
<td>0</td>
</tr>
<tr>
<td>IC1A</td>
<td>Dual Comparator</td>
<td>3</td>
<td>Digikey 296-1841-5-ND</td>
<td>Texas Instruments TLC3702CP</td>
<td>1.04</td>
<td>0</td>
</tr>
<tr>
<td>IC1B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Component</td>
<td>Qty</td>
<td>Supplier / Part No</td>
<td>Manufacture / Part No</td>
<td>Unit Cost ($CAD)</td>
<td>Total Cost ($CAD)</td>
</tr>
<tr>
<td>------</td>
<td>----------------</td>
<td>-----</td>
<td>---------------------------------</td>
<td>---------------------------------</td>
<td>-----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>IC2</td>
<td>Bridge Rectifier</td>
<td>3</td>
<td>Digikey GBU8K-BPMS-ND</td>
<td>Micro Commercial Co. GBU8K-BP</td>
<td>1.04</td>
<td>0</td>
</tr>
<tr>
<td>QD1</td>
<td>P-Channel MOSFET</td>
<td>3</td>
<td>Digikey ZVP3306A-ND</td>
<td>Diodes Inc. ZVP3306A</td>
<td>0.68</td>
<td>0</td>
</tr>
<tr>
<td>QD2</td>
<td>N-Channel MOSFET</td>
<td>3</td>
<td>Digikey 2N7000TACT-ND</td>
<td>Fairchild 2N7000TA</td>
<td>0.42</td>
<td>0</td>
</tr>
<tr>
<td>QD3</td>
<td>Power NMOS</td>
<td>3</td>
<td>Digikey IRL640A-ND</td>
<td>Fairchild IRL640A</td>
<td>1.62</td>
<td>0</td>
</tr>
<tr>
<td>QD4</td>
<td>NPN BJT</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>Fairchild 2N3904</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RD1</td>
<td>10kΩ, 1%, 1/4W</td>
<td>6</td>
<td>Digikey S10KCACT-ND</td>
<td>Stackpole RNMF14FTC10K0</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>RD2</td>
<td>20kΩ, 1%, 1/4W</td>
<td>3</td>
<td>Digikey S20KCACT-ND</td>
<td>Stackpole RNMF14FTC20K0</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>RD3</td>
<td>14.0kΩ, 1%, 1/4W</td>
<td>3</td>
<td>Digikey RNF14FTD14K0CT-ND</td>
<td>Stackpole RNF14FTD14K0</td>
<td>0.16</td>
<td>0</td>
</tr>
<tr>
<td>RD5</td>
<td>33kΩ, 5%</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RD6</td>
<td>33kΩ, 5%</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T1</td>
<td>Isolation Transformer</td>
<td>3</td>
<td>Digikey 237-1272-ND</td>
<td>Triad Magnetics VPS230-570</td>
<td>19.15</td>
<td>0</td>
</tr>
<tr>
<td>RD7</td>
<td>27kΩ, 5%</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RD8</td>
<td>82Ω, 5%</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RD9</td>
<td>100kΩ, 5%</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RD10</td>
<td>4.7kΩ, 5%</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RD7</td>
<td>27kΩ, 5%</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Name</td>
<td>Component</td>
<td>Qty</td>
<td>Supplier / Part No</td>
<td>Manufacture / Part No</td>
<td>Unit Cost ($CAD)</td>
<td>Total Cost ($CAD)</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>-----</td>
<td>--------------------</td>
<td>-----------------------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>RD8</td>
<td>82Ω, 5% Resistor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RD9</td>
<td>100kΩ, 5% Resistor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Power Monitor**

<table>
<thead>
<tr>
<th>Name</th>
<th>Component</th>
<th>Qty</th>
<th>Supplier / Part No</th>
<th>Manufacture / Part No</th>
<th>Unit Cost ($CAD)</th>
<th>Total Cost ($CAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP1</td>
<td>470nF, 10% ceramic Capacitor</td>
<td>6</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CP2</td>
<td></td>
<td></td>
<td></td>
<td>ON Semiconductor LM358NG</td>
<td>0.48</td>
<td>0</td>
</tr>
<tr>
<td>IC3A</td>
<td>Dual Op Amp</td>
<td>3</td>
<td>Digikey LM358NGOS-ND</td>
<td>Vishay BC Components PAC100001007 FA1000</td>
<td>1.03</td>
<td>0</td>
</tr>
<tr>
<td>IC3B</td>
<td></td>
<td></td>
<td></td>
<td>Stackpole Electronics Inc RNF14FTD10M0C T-ND</td>
<td>0.31</td>
<td>0</td>
</tr>
<tr>
<td>RP1</td>
<td>100mΩ (1W), 1% Current Sense Resistor</td>
<td>3</td>
<td>Digikey PPC1D.10CT-ND</td>
<td>Stackpole Electronics Inc RNF14FTD10M0C T-ND</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>RP2</td>
<td>10MΩ, 1%, 1/4W, Resistor</td>
<td>3</td>
<td>Digikey RNF14FTD10M0C T-ND</td>
<td>Stackpole Electronics Inc RNF14FTD10M0C T-ND</td>
<td>0.16</td>
<td>0</td>
</tr>
<tr>
<td>RP3</td>
<td>220kΩ, 1%, 1/4W, Resistor</td>
<td>3</td>
<td>Digikey S220KC1ACT-ND</td>
<td>Stackpole Electronics Inc RNF14FTD53K6C T-ND</td>
<td>0.16</td>
<td>0</td>
</tr>
<tr>
<td>RP4</td>
<td>1MΩ, 5%, Resistor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RP5</td>
<td>220kΩ, 5% Resistor</td>
<td>3</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RP6</td>
<td>10kΩ, 1%, 1/4W, Resistor</td>
<td>3</td>
<td>Digikey RNF14FTD10K0C T-ND</td>
<td>Stackpole Electronics Inc RNF14FTD10K0C T-ND</td>
<td>0.16</td>
<td>0</td>
</tr>
<tr>
<td>RP7</td>
<td>53.6kΩ, 1%, 1/4W, Resistor</td>
<td>3</td>
<td>Digikey RNF14FTD53K6C T-ND</td>
<td>Stackpole Electronics Inc RNF14FTD53K6C T-ND</td>
<td>0.16</td>
<td>0</td>
</tr>
</tbody>
</table>
## Power Supply

<table>
<thead>
<tr>
<th>Name</th>
<th>Component</th>
<th>Qty</th>
<th>Supplier / Part No</th>
<th>Manufacture / Part No</th>
<th>Unit Cost (CAD)</th>
<th>Total Cost (CAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPS1</td>
<td>18mF, 20% Aluminum Electrolytic Capacitor</td>
<td>3</td>
<td>Digikey   P6867-ND</td>
<td>Panasonic Electronic Components ECO-S1CP183CA</td>
<td>2.82</td>
<td>0</td>
</tr>
<tr>
<td>CPS2</td>
<td>330nF, 5% Ceramic Capacitor</td>
<td>9</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CPS3</td>
<td>100uF, 20% Aluminum Electrolytic Capacitor</td>
<td>9</td>
<td>Digikey  493-3716-ND</td>
<td>Nichicon RR71C101MDN1</td>
<td>0.68</td>
<td>0</td>
</tr>
<tr>
<td>CPS4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.46</td>
<td>0</td>
</tr>
<tr>
<td>D1</td>
<td>Rectifier Diode</td>
<td>3</td>
<td>Digikey 1N4001-E3/54GICT-ND</td>
<td>Vishay General Semiconductor 1N4001-E3/54</td>
<td>0.36</td>
<td>0</td>
</tr>
<tr>
<td>HS2</td>
<td>Heat Sink</td>
<td>3</td>
<td>Digikey A10764-ND</td>
<td>Assmann WSW Components V7235-T</td>
<td>1.09</td>
<td>0</td>
</tr>
<tr>
<td>ICPS1</td>
<td>3.3V Voltage Regulator</td>
<td>3</td>
<td>Digikey KA278R33CTUFS-ND</td>
<td>Fairchild Semiconductor  KA278R33CTU</td>
<td>0.55</td>
<td>0</td>
</tr>
<tr>
<td>ICPS2</td>
<td>5V Voltage Regulator</td>
<td>3</td>
<td>Digikey MCP1702-5002E/TO-ND</td>
<td>Microchip Technology MCP1702-5002E/TO</td>
<td>1.83</td>
<td>0</td>
</tr>
<tr>
<td>ICPS3</td>
<td>6V Voltage Regulator</td>
<td>3</td>
<td>Digikey BA60BC0T-ND</td>
<td>Rohm Semiconductor BA60BC0T</td>
<td>11.95</td>
<td>0</td>
</tr>
<tr>
<td>T2</td>
<td>Power Transformer</td>
<td>3</td>
<td>Digikey 595-1001-ND</td>
<td>Signal Transformer 14A-5.0R-12</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Microcontroller

<table>
<thead>
<tr>
<th>Name</th>
<th>Component</th>
<th>Qty</th>
<th>Supplier / Part No</th>
<th>Manufacture / Part No</th>
<th>Unit Cost ($CAD)</th>
<th>Total Cost ($CAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM1</td>
<td>10nF, 10% Ceramic Capacitor</td>
<td>6</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CM2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IC4</td>
<td>DIP Switch</td>
<td>3</td>
<td>Digikey CT2104MS-ND</td>
<td>CTS Electrocomponents 210-4MS</td>
<td>0.59</td>
<td>0</td>
</tr>
<tr>
<td>IC5</td>
<td>Microcontroller</td>
<td>3</td>
<td>Digikey PIC18F45K22-I/P-ND</td>
<td>Microchip Technology PIC18F45K22-I/P</td>
<td>3.79</td>
<td>0</td>
</tr>
<tr>
<td>RM1</td>
<td>4.7kΩ, 5% Resistor</td>
<td>15</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RM2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RM3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RM4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RM5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>S1</td>
<td>40 Pin PDIP Socket</td>
<td>3</td>
<td>Digikey AE10008-ND</td>
<td>Assmann WSW A40-LC-TT-R</td>
<td>0.54</td>
<td>0</td>
</tr>
</tbody>
</table>

### Wi-Fi Card

<table>
<thead>
<tr>
<th>Name</th>
<th>Component</th>
<th>Qty</th>
<th>Supplier / Part No</th>
<th>Manufacture / Part No</th>
<th>Unit Cost ($CAD)</th>
<th>Total Cost ($CAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CW1</td>
<td>10uF, 10% Ceramic Capacitor</td>
<td>2</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CW2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IC8</td>
<td>Wi-Fi Card</td>
<td>1</td>
<td>Digikey MRF24WB0MA/RM-ND</td>
<td>Microchip Technology MRF24WB0MA/RM</td>
<td>27.80</td>
<td>0.00</td>
</tr>
<tr>
<td>RW1</td>
<td>4.7kΩ, 5% Resistor</td>
<td>5</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RW2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RW3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RW4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RW5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Other

<table>
<thead>
<tr>
<th>Name</th>
<th>Component</th>
<th>Qty</th>
<th>Supplier / Part No</th>
<th>Manufacture / Part No</th>
<th>Unit Cost ($CAD)</th>
<th>Total Cost ($CAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICD 3</td>
<td></td>
<td>1</td>
<td>Supplied by University of Manitoba</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>F1</td>
<td>1A Fuse</td>
<td>3</td>
<td>Digikey 507-1261-ND</td>
<td>Bel Fuse Inc 5MF 1-R</td>
<td>0.20</td>
<td>00</td>
</tr>
<tr>
<td>FH1/FH2</td>
<td>Fuse Holder</td>
<td>6</td>
<td>Digikey WK6242-ND</td>
<td>Littelfuse Inc 52000001009</td>
<td>0.18</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Copper Clad Board</td>
<td>1</td>
<td>Digikey PC76-T-ND</td>
<td>Injectorall Electronics PC76-T-ND</td>
<td>35.17</td>
<td>35.17</td>
</tr>
</tbody>
</table>

**Totals**

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost ($CAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component Total</td>
<td>295.35</td>
</tr>
<tr>
<td>S &amp; H Digikey</td>
<td>Free</td>
</tr>
<tr>
<td>S &amp; H Mouser</td>
<td>20.00</td>
</tr>
<tr>
<td>GST &amp; PST</td>
<td>35.44</td>
</tr>
<tr>
<td>Total Estimated Cost</td>
<td>330.79</td>
</tr>
<tr>
<td><strong>Total Final Cost</strong></td>
<td><strong>310.72</strong></td>
</tr>
</tbody>
</table>
Appendix B – Schematics
Dimmer/Power Monitor
PLC Modem
Microcontroller
Wi-Fi Card
Appendix C – Software

Master Node Code – mastermain.c

/* mastermain.c
This is the source code file for the Master Node microcontroller.
It is incomplete, but DAC and ADC functionality is complete. Modem and UDP code
is complete but untested, and deactivated with #define USE_UDP and
* #define USE_MODEM. Wi-Fi initialization is not complete. */

//#define USE_UDP
//#define USE_MODEM

#include <pic18.h>
#include <htc.h>

#pragma config IESO=OFF FOSC=INTIO7 PRICLKEN=ON FCMEN=ON PLLCFG=OFF BOREN=OFF
#pragma config WDTEN=OFF

#define BYTE unsigned char
#define HWORD unsigned short
#define WORD unsigned int
#define NUM_NODES 16

#define AD_GO 0x02
#define AD_ON 0x01
#define AD_AN0 0x00
#define AD_AN1 0x04

#ifndef USE_MODEM
#define M_DIMMER 0x01
#define M_POWER 0x02
#define M_ACK 0x03
#define M_CRQST 0x04
#endif

#ifndef USE_UDP
#define PORT_RX 7701
#define PORT_TX 7702
#define ACK 98
#define INIT 25
#define POWER 12
#define DIMMER 76
#endif

// gvars
BYTE dimmers[NUM_NODES]; // 1 byte per node - current dimmer at each node
HWORD powers[NUM_NODES]; // 2 bytes per node - current power at each node
BYTE frames[NUM_NODES]; // Current timeout framecount for each node
HWORD active; // 1 bit per node - 1 if active
BYTE id; // 16 values - this must be ID 1 for master
HWORD voltage; // Own voltage reading
HWORD current; // Own current Reading
HWORD dimmer; // Own dimmer setting
HWORD power; // Own power value
#ifdef USE_MODEM
BYTE addrL[NUM_NODES];  // address for each node
BYTE addrH[NUM_NODES];
BYTE rxbuf[256];
BYTE len;
BYTE txbuf[256];
#endif

// Wifi
#ifdef USE_UDP
BYTE wfcp;
UDP_Socket socketRx;
UDP_Socket socketTx;
WORD pc_ip = 0;  // Use IP protocol to get ip address
#endif

// func protos
void inline init();  // setup registers
HWORD adc(BYTE an);  // read specified ADC

#ifdef USE_WIFI
void inline initWifi();  // init Wi-Fi connection
void inline processUDP();
#endif

#ifdef USE_MODEM
void inline initUART();  // setup UART to talk to modem
void inline initModem();  // initialize Modem
void inline processModem();  // called once per frame
void inline sendUART(BYTE* d, BYTE len);  // send data to Modem
BYTE inline rcvUART(BYTE* d, BYTE len);  // return 0 if no / bad data
void inline sendModem(BYTE id, BYTE opcode, BYTE dimmer);  // send data over PLC
BYTE inline rcvModem(BYTE data[4]);  // id, opcode, 2B data - ret 0 if no data
void inline setCHK(BYTE* data, BYTE len);  // add checksum to packet
#endif

int main(void) {
    init();
    BYTE framecount = 0x00;
    while(1) {
        #ifdef USE_WIFI
        processUDP();
        #endif
        #ifdef USE_MODEM
        processModem();
        #endif
        current = adc(AD_AN0);
        voltage = adc(AD_AN1);
        power = current * voltage;
        VREFCON2 = dimmer;
    }
    return 0;
}

void inline init() {
    OSCCON = 0xFC;  // Clock config
// PORTA
// 0: AN0 - PM I in
// 1: AN1 - PM V in
// 2: DACOUT - PWM out
// 5: SS1 - Wifi SPI
// 6: RST for Wifi
TRISA = 0x13;
ANSELA = 0x27;

// PORTB
// 0-3 is ID
// 4: Modem BR0
// 5: Modem BR1
// 6: Modem RESB
TRISB = 0x0F;
ANSELB = 0x30;

// PORTC
// 3: SCK1 Wifi
// 4: SDI1 Wifi
// 5: SD01 Wifi
// 6: TX1 Modem
// 7: RX1 Modem
TRISC = 0x90;
ANSELC = 0x68;

// PORTD
// 0: CRC Modem
// 1: T_REQ Modem
// 2: TX_DATA Modem
// 3: RX_DATA Modem
TRISD = 0x05;
ANSELD = 0x0A;

// PORTE
// 0: INTW Wifi
TRISE = 0x01;
ANSELE = 0x02;

// ID - PORTB0-3
id = PORTB & 0x0F;

// ADC I - AN0, ADC V - AN1
ADCON1 = 0x80;
ADCON2 = 0xB8;
ADCON0 = AD_ON;

// DACOUT
VREFCON0 = 0x00;
VREFCON1 = 0xE0;

// UART - Modem
// RXDATA RE3
// TX_DATA RE2
// RXD TX1
// TXD RX1
// T_REQ RE1
// Wifi init
#ifdef USE_UDP
    initWifi();
    UDPInit();
    socketRx = UDPOpen(PORT_RX,NULL,PORT_RX);
    socketTx = 0;
#endif

#ifdef USE_MODEM
    initUART();
    initModem();
#endif

HWORD adc(BYTE an) {
    HWORD ret;
    ADCON0 = an | AD_GO | AD_ON;
    while(ADCON0 & AD_GO) {}
    ret = ADRESH | (ADRESL << 8);
    return ret;
}

#ifdef USE_WIFI
    void inline initWifi() {
    }
#endif

#ifdef USE_MODEM
    void inline initUART() {
    }
#endif

void inline initModem() {
    // reset modem
    LATB6 = 0;
    LATB6 = 1;

    // do reset request
    txbuf[0] = 0x3F;
    txbuf[1] = 3;
    txbuf[2] = 0x21;
    sendUART(txbuf, len);

    // do config set - set AMIS49587 datasheet
    txbuf[0] = 0x3F;
    txbuf[1] = 0x26;
    txbuf[2] = 0x71;
    txbuf[3] = 0x00;
    txbuf[4] = 0x00;
    txbuf[5] = 0x00;
    txbuf[6] = 0x00;
    txbuf[7] = 0x00;
    txbuf[8] = 0x00;
    txbuf[9] = 0x00;
txbuf[10] = 0x00;  
txbuf[11] = 0x00;  
txbuf[12] = 0x00;  
txbuf[13] = 0x00;  
txbuf[14] = 0x00;  
txbuf[15] = 0x00;  
txbuf[16] = 0x00;  
txbuf[17] = 0x00;  
txbuf[18] = 0x00;  
txbuf[19] = 0x00;  
txbuf[20] = 0x00;  
txbuf[21] = 0x00;  
txbuf[22] = 0x00;  
txbuf[23] = 0x00;  
txbuf[24] = 0x00;  
txbuf[25] = 0x00;  
txbuf[26] = 0x00;  
txbuf[27] = 0x00;  
txbuf[28] = 0x00;  
txbuf[29] = 0x03;  // fs H = 10kHz  
txbuf[30] = 0x6A;  // fs L  
txbuf[31] = 0x06;  // fm H = 20kHz  
txbuf[32] = 0xD4;  // fm L  
txbuf[33] = 0x00;  
txbuf[34] = 0x00;  
txbuf[35] = 0x0F;  
txbuf[36] = 0x00;  
txbuf[37] = 0xC0;  
txbuf[38] = 0x8C;  
txbuf[39] = 0x00;  
txbuf[40] = 0x00;  

// Send config until modem confirms okay  
BYTE okay = 0;  
while (!okay) {  
    sendUART(txbuf, 41);  
    rcvUART(rxbuf, len);  
    if (rxbuf[2] == 0x72) // check if config is okay  
        okay = 1;  
}

// Send data to modem over UART  
void inline sendUART(BYTE* d, BYTE len) {

    BYTE not_ok = 1;  

    // Request oky to send from modem  
    // when T_REQ pulled low, modem responds with a status packet  
    // modem is ready to receive if first bit of second byte is 1  
    RCSTA1 = RCSTA1 | 0x10; // CREN = 1  
    while (not_ok) {  
        LATE = 1;  
        LATE1 = 0; // T_REQ pulled low  
        while (RC1IF == 0) {} // wait for read of a byte  
        BYTE d = RCREG1;  
        // 'd' should equal 0xF
while(RC1IF == 0) {}
    d = RCREG1; // bit 0 == 1 if okay
    if(d & 0x01 > 0)
        not_ok = 0;
    // read other 2 bytes
while(RC1IF == 0) {}
    d = RCREG1;
while(RC1IF == 0) {}
    d = RCREG1;

RCSTA1 = RCSTA1 & 0xEF; // CREN = 0
// Now okay to send a frame
// first setup checksum
setCHK(d,len);
TXSTA1 = TXSTA1 | 0x20;
for(BYTE i = 0; i <= len; i++) {
    while(TXSTA1 & 0x02 == 0) {} // wait for tx buf to empty - TRMT == 0
    TXREG1 = d[i];
}
TXSTA1 = TXSTA1 & 0xDF;
// Receive data from modem over UART
BYTE inline rcvUART(BYTE* rcv, BYTE* lenout) {
    RCSTA1 = RCSTA1 | 0x10; // CREN = 1
    if(RC1IF == 0)
        return 0;
    // check if there is data ready
    BYTE d = RCREG1;
    // 'd' should equal 0x3F - start byte
while(RC1IF == 0) {}
    BYTE len = RCREG1; // get length of message
    // Receive data into a buffer
    for(BYTE i = 0; i < len; i++) {
        while(RC1IF == 0) {}
        rcv[i] = RCREG1;
    }
    // clear the flag
    RCSTA1 = RCSTA1 & 0xEF; // CREN = 0
    *lenout = len;
}
// send data to a slave over PLC
BYTE inline sendModem(BYTE id, BYTE opcode, BYTE dimmer) {
    txbuf[0] = 0x3F; // Start byte
txbuf[1] = 12; // length
txbuf[2] = 0x51; // data request
txbuf[3] = 0x77;
txbuf[4] = 0x00; // master address is assumed to be 0x000
    txbuf[5] = 0x00;
txbuf[6] = addrH[id];
txbuf[7] = addrL[id];
// Byte 8 is a buffer
txbuf[9] = id;
txbuf[10] = opcode;
sendUART(txbuf, 12);

BYTE d[4];
rcvModem(&d); // receive DATA_CONFIRM packet
if(rxbuf[2] == 0x52)
    return 1; // tx is good
else
    return 0;
}

// Receive data over PLC from a slave
BYTE inline rcvModem(BYTE* data) {
    // id, opcode, 2B data - assumed length of 4
    if(rcvUART(rxbuf) == 0) // check if data available
        return 0;
    for(BYTE i = 0; i < 4; i++)
        data[i] = rxbuf[i+11]; // place data into buffer
    if(rxbuf[2] == 0x50) // check if receive okay
        return 1; // receive okay
    else
        return 0;
}

// add checksum to a packet
void inline setCHK(BYTE* data, BYTE len) {
    BYTE chk = 0;
    for(BYTE i = 0; i < len; i++) {
        chk += data[i];
    }
data[len] = chk;
}

// called once per frame
void inline processModem() {
    static BYTE framecount = 0; // used to determine timeouts

    BYTE okay = 0;
    // send dimmer to all slaves
    for(BYTE i = 0; i < NUM_NODES; i++) {
        if(((active >> i) & 0x01) {
            while(!okay) {
                okay = sendModem(i,M_DIMMER,dimmers[i]);
            }
        }
    }
    // receive packets until all received
    BYTE data[4];
    while(rcvModem(data)) {
        if(data[1] == M_POWER) {
            powers[data[0]] = data[2];
            powers[data[0]] = powers[data[0]] & ((HWORD)data[3] << 8);
            frames[data[0]] = framecount;
            active = active | (0x01 << data[0]);
        }
    }
}
} else if(data[1] == M_CRQST) {
    // ACK the request to begin transmission
    sendModem(data[0],M_ACK,0);
}

// recheck if all nodes are active via timeout
for(BYTE i = 0; i < NUM_NODES; i++) {
    if(framecount - frames[i] > 2) {
        active = active & (~0x01 << i));
        // make node dropout notifications here
    }
}
framecount++;
}
#endif
#endif USE_UDP

void inline processUDP() {
    static BYTE sendAck = 0;
    static BYTE sendPower = 0;
    if(UDPIsGetReady(socketRx)) {
        BYTE d;
        UDPGet(&d);
        // State machine
        if(d == INIT) {
            sendPower = 0;
            sendAck = 1;
            // Get pc ip address using IP protocol
            // pc_ip = 0;
        }
        else if(d == DIMMER) {
            if(sendAck) {
                sendPower = 1;
                sendAck = 0;
            }
            // Get dimmer values
            BYTE di = 1;
            while(UDPGet(&d)) {
                dimmers[di++] = d;
            }
            sendPower = 2;
        }
        if(sendAck) {
            if(socketTx != 0) {
                UDPClose(socketTx);
            }
            socketTX = UDPOpen(PORT_TX,pc_ip,PORT_TX);
            while(UDPIsPutReady < 33) {} // UDPPut(ACK);
UDPPut(active & 0xFF);
UDPPut(active & 0xFF00 >> 8);
for(int i = 0; i < 31; i++)
    UDPPut(0);
UDPFlush();
}
else if(sendPower == 2) {
    sendPower = 1;
    while(UDPIsPutReady < 33) {};
    UDPPut(POWER);
    for(int i = 0; i < 16; i++) {
        UDPPut(power[i] & 0xFF);
        UDPPut((power[i] & 0xFF00) >> 8);
    }
    UDPFlush();
}

#endif
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.lang.String;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.util.*;

public class GUI {
    public static void main(String[] args) {
        final int NUMBER_OF_NODES = 16;
        NodeData[] node1;
        boolean[] active = {false, false, false, false, false, false, false,
            false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false};
        double[] powers;

        // Create the Modem interface
        Callback c = new Callback();
        ModemIF mif = new ModemIF(c);
        powers = mif.power();

        // Create the array of nodes
        node1 = new NodeData[NUMBER_OF_NODES];
        for (int i = 0; i < node1.length; i++) {
            node1[i] = new NodeData();
            node1[i].name = "Node" + (i + 1);
            node1[i].ID = i;
            node1[i].power = powers[i];
            node1[i].dimmer = node1[i].dimmer;
        }

        // Create the window
        Window myWindow = new Window(800, 600, node1, active, mif);
        myWindow.setTitle("Dimmer Control");
        myWindow.setLocationRelativeTo(null);
        myWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Import an image and use as custom icon
        BufferedImage image = null;
        if (myWindow.getClass().getResource("icon.png") != null) {
            try {
                image = ImageIO.read(myWindow.getClass().getResource("icon.png"));
            } catch (IOException e) {
e.printStackTrace();
}
myWindow.setIconImage(image);
}
myWindow.setResizable(false);
myWindow.setVisible(true);

while(true) {
    powers = mif.power();
    for (int i=0;i<node1.length;i++) {
        node1[i].power = powers[i];
    }
    mif.update();
}

//Main
static class Callback implements NodeStatusCallback {
    @Override
    public boolean statusChanged(boolean[] active) {
        for (int i=0;i<active.length;i++) {
            // TODO stuff
            //
        }
        return true;
    }
}

@SuppressWarnings("serial")
class Window extends JFrame implements ActionListener,
                                WindowListener,
                                WindowStateListener,
                                ChangeListener,
                                ListSelectionListener {

    JLabel positionName;
    static MyComponent gauge;
    private static Vector<String> v;
    static JList<String> jl;
    static JSlider dimmer;
    int currentNode;

    ModemIF mif;
    NodeData[] node1;
    boolean[] active;

    int current;
    int j;
    int theCurrentNode;

    public Window(int width, int height, NodeData[] node1, boolean[] active,
                  ModemIF mif) {
        this.mif = mif;
        this.node1 = node1;
        this.active = active;
    }
final int DIMMER_MIN = 0;
final int DIMMER_MAX = 100;
final int UNIT_HEIGHT = height - 98;
JMenuBar menuBar;
JMenu menu;
JMenuItem menuItem;
JPanel panel;

setSize(width, height); // Set the window size

// MENU
menuBar = new JMenuBar(); //Create the menu bar

//Build the first menu option
menu = new JMenu("Options");
menu.setMnemonic(KeyEvent.VK_O);
menuBar.add(menu);

//a group of JMenuItems
menuItem = new JMenuItem("Rename Node");
menuItem.addActionListener(this);
menu.add(menuItem);

menuItem = new JMenuItem("Reconnect", KeyEvent.VK_R);
menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_R,
ActionEvent.CTRL_MASK));
menuItem.addActionListener(this);
menu.add(menuItem);

menuItem = new JMenuItem("Exit", KeyEvent.VK_X);
menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F4,
ActionEvent.ALT_MASK));
menuItem.addActionListener(this);
menu.add(menuItem);

//Build the second menu option
menu = new JMenu("Help");
menu.setMnemonic(KeyEvent.VK_H);
menuBar.add(menu);

//a group of JMenuItems
menuItem = new JMenuItem("About");
menuItem.addActionListener(this);
menu.add(menuItem);

setJMenuBar(menuBar);
addWindowListener(this);
addWindowStateListener(this);

// Get the frame's container to place stuff in
Container contentPane = getContentPane();

JPanel titles = new JPanel();
titles.setLayout(new GridLayout(1, 3, 0, 0));
titles.setSize(800, 30);
contentPane.add(titles, "North");

//add contents
JPanel can;
can = new JPanel()
JLabel title1 = new JLabel("System Nodes");
title1.setFont(new Font("SansSerif", Font.BOLD, 16));
title1.setHorizontalAlignment(SwingConstants.CENTER);
can.add(title1);
titles.add(can);
can = new JPanel();
JLabel title2 = new JLabel("Power Usage");
title2.setFont(new Font("SansSerif", Font.BOLD, 16));
title2.setHorizontalAlignment(SwingConstants.CENTER);
can.add(title2);
titles.add(can);
can = new JPanel();
JLabel title3 = new JLabel("Dimmer Level");
title3.setFont(new Font("SansSerif", Font.BOLD, 16));
title3.setHorizontalAlignment(SwingConstants.CENTER);
can.add(title3);
titles.add(can);

// All of our user interface controls will go in this panel
// It will be arranged left-to-right, and will contain 3 panels inside it
JPanel controls = new JPanel();
controls.setLayout(new GridLayout(1, 3, 0, 0));
controls.setSize(800, 570);
contentPane.add(controls, "South");

// Panel 1: Node Elements
panel = new JPanel();
v = new Vector<String>();
jl = new JList<String>(v);
jl.setFont(new Font("Serif", Font.PLAIN, 18));

// put in list to display on screen
for (int i=0;i<active.length;i++) {
    if (active[i]) {
        v.add("" + node1[i].name);
    }
}
jl.setListData(v);
jl.setSelectedIndex(0);
jl.ensureIndexIsVisible(0);
jl.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
jl.setVisibleRowCount(-1);
JScrollPane listScroller = new JScrollPane(jl);
listScroller.setPreferredSize(new Dimension(200, UNIT_HEIGHT));

// add listener for the list of nodes
ListSelectionModel listSelectionModel = jl.getSelectionModel();
listSelectionModel.addListSelectionListener(this);
panel.add(listScroller);
controls.add(panel);

// Panel 2: Power Usage
panel = new JPanel();
current = Window.jl.getSelectedIndex();
theCurrentNode = findNodeID(current);
gauge = new MyComponent(node1, theCurrentNode);
gauge.setPreferredSize(new Dimension(261, UNIT_HEIGHT/2));
panel.add(gauge);

positionName = new JLabel();
positionName.setText(node1[theCurrentNode].power + "W");
positionName.setFont(new Font("Serif", Font.PLAIN, 48));
positionName.setHorizontalAlignment(SwingConstants.CENTER);
positionName.setPreferredSize(new Dimension(261, UNIT_HEIGHT/2));
panel.add(positionName);

controls.add(panel);

// Panel 3: Dimmer Control
panel = new JPanel();
dimmer = new JSlider(JSlider.VERTICAL, DIMMER_MIN, DIMMER_MAX,
(int)node1[theCurrentNode].dimmer);
dimmer.addChangeListener(this);

//Turn on labels at major tick marks.
dimmer.setMajorTickSpacing(10);
dimmer.setMinorTickSpacing(1);
dimmer.setSnapToTicks(true);
dimmer.setPaintTicks(true);
dimmer.setPaintLabels(true);
dimmer.setPreferredSize(new Dimension(200, UNIT_HEIGHT));
panel.add(dimmer);
controls.add(panel);

/**
* React to window events.
*/
public void windowClosing(WindowEvent e) {
    mif.shutdown();
    System.exit(0);
}

// called if window is resized
public void windowStateChanged(WindowEvent e) {}
public void windowIconified(WindowEvent e) {}
public void windowDeiconified(WindowEvent e) {}
public void windowOpened(WindowEvent e) {}
public void windowClosed(WindowEvent e) {}
public void windowActivated(WindowEvent e) {}
public void windowDeactivated(WindowEvent e) {}

/**
* A listener for the Slider
*/
public void stateChanged(ChangeEvent e) {
    JSlider source = (JSlider)e.getSource();
    if (!source.getValueIsAdjusting()) {
        double level = (double)source.getValue(); // dimmer level (or slider position)
currentNode = jl.getSelectedIndex();
}
```java
mif.dimmer(level, currentNode);
theCurrentNode = findNodeID(currentNode);
node1[theCurrentNode].dimmer = level;
}

/**
 * A listener for the Menu
 */
@override
public void actionPerformed(ActionEvent e) {
    String input;

    // check which option has been clicked
    JMenuItem clicked = (JMenuItem)e.getSource();
    String name = clicked.getText();
    if (name.equals("Rename Node")){
        // ask the user for input
        input = "";
        ImageIcon icon = null;
        if (getClass().getResource("renameNode.png") != null) {
            icon = new ImageIcon(getClass().getResource("renameNode.png"));
        }
        input = JOptionPane.showInputDialog(null,
                "Enter the new node name:", "Add Node", icon, null, null);
    }
    else {
        input = JOptionPane.showInputDialog(null,
                "Enter the new node name:);
    }
    if (input != null && !input.isEmpty()) {
        // Rename node by removing and replacing the element in the JList
        current = Window.jl.getSelectedIndex();
        v.remove(current);
        v.add(current, input);
        theCurrentNode = findNodeID(current);
        node1[theCurrentNode].setName(input);
    }
    else if (name.equals("Reconnect")) {
        mif.shutdown();
        String ip = null;
        while (ip == null || ip == "") {
            ip = JOptionPane.showInputDialog(null, "Enter master node IP (look on your router):");
            active = mif.init(ip); // TODO check for a "null" return which indicates an error
        }
        // put in list to display on screen
        for (int i=0; i<active.length; i++) {
            if (active[i]) {
                v.add("" + node1[i].name);
            }
        }
        jl.setListData(v);
    }
```
else if (name=="Exit") {
    System.exit(0);
}
else if (name=="About") {
    AboutWindow about = new AboutWindow(Window.this);
    about.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
    about.setSize(500,450);
    about.setLocationRelativeTo(null);
    about.setVisible(true);
}

//A listener for the List of Nodes
public void valueChanged(ListSelectionEvent e) {
    if (!e.getValueIsAdjusting()) {
        // call ModemIF methods
        current = Window.jl.getSelectedIndex();
        theCurrentNode = findNodeID(current);
        positionName.setText(node1[theCurrentNode].power + "W");
        // update the Power Indicator (since we expect to receive
        power values between 0W
        // and 125W we divide by 125 so that the decimalPower value is
        kept between 0 and 1
        Slice[] newIndicator = { new Slice(1.25, Color.black,
            node1[theCurrentNode].power/125) };
        Window.gauge.updateIndicator(newIndicator);
        dimmer.setValue((int)node1[theCurrentNode].dimmer);
    }
}

/**
 * @param current
 * @return theCurrentNode
 */
public int findNodeID(int current) {
    j = 0; //WHAT IF THE NODE IS ZERO???
    theCurrentNode = 0;
    for (int i=0;i<active.length;i++) {
        if (j<=current) {
            if (active[i]) {
                j++;
                theCurrentNode = i;
            }
        } else {
            i = active.length; //forced exit of the for-loop
        }
    }
    return theCurrentNode;
}

} //Window

/**
 * Slice
 */
class Slice {
double value;
Color color;
double decimalPower;
public Slice(double value, Color color) {
    this.value = value;
    this.color = color;
}
public Slice(double value, Color color, double decimalPower) {
    this.value = value;
    this.color = color;
    this.decimalPower = decimalPower;
}
} //Slice

/** MyComponent */
@SuppressWarnings("serial")

class MyComponent extends JComponent {
    Color theGrey = new Color(238, 238, 238);
    Slice[] slices = {
        new Slice(30, theGrey),
        new Slice(15, new Color(255, 0, 0)),
        new Slice(10, new Color(255, 255, 0)),
        new Slice(45, new Color(0, 255, 0))
    };
    Slice[] cover = {
        new Slice(100, theGrey)
    };
    Slice[] nub = {
        new Slice(100, Color.red)
    };
    Slice[] indicator;

    MyComponent(NodeData[] node1, int theCurrentNode) {
        //determine which node is selected
        // this decimalPower value must be between 0 and 1
        Slice[] indicator = {
            new Slice(1.25, Color.black, node1[theCurrentNode].power/125)
        };
        updateIndicator(indicator);
    }
    public void paint(Graphics g) {
        drawPie((Graphics2D) g, getBounds(), slices);
        //draw cover
        Rectangle bounds = getBounds();
        bounds.setBounds(bounds.x + bounds.width/2 - (bounds.width/2)/2 - 2,
                         bounds.y + bounds.height/2 - (bounds.height/2)/2,
                         bounds.width/2, bounds.height/2);
        drawPie((Graphics2D) g, bounds, cover);
        //draw indicator
        drawIndicator((Graphics2D) g, getBounds(), indicator);
        //draw nub
        bounds.setBounds(bounds.x + bounds.width/2 - (bounds.width/8)/2 - 2,
                         bounds.y + bounds.height/2 - (bounds.height/8)/2,
                         bounds.width/8, bounds.height/8);
        drawPie((Graphics2D) g, bounds, nub);
    }

    void drawPie(Graphics2D g, Rectangle area, Slice[] slices) {
        double total = 0.0D;
        for (int i = 0; i < slices.length; i++) {
            total += slices[i].value;
        }
        double curValue = 60.0D;
        int startAngle = 0;
for (int i = 0; i < slices.length; i++) {
    startAngle = (int) (curValue * 360 / total);
    int arcAngle = (int) (slices[i].value * 360 / total);
    g.setColor(slices[i].color);
    g.fillArc(area.x, area.y, area.height, area.height,
              startAngle, arcAngle);
    curValue += slices[i].value;
}

void drawIndicator(Graphics2D g, Rectangle area, Slice[] slices) {
    double total = 100.0D;
    double curValue = 60.0D - slices[0].decimalPower*70.0D;
    int startAngle = 0;
    for (int i = 0; i < slices.length; i++) {
        startAngle = (int) (curValue * 360 / total);
        int arcAngle = (int) (slices[i].value * 360 / total);
        g.setColor(slices[i].color);
        g.fillArc(area.x, area.y, area.height, area.height,
                  startAngle, arcAngle);
        curValue += slices[i].value;
    }
}

public void updateIndicator(Slice[] newIndicator) {
    indicator = newIndicator;
    this.repaint();
}

} // MyComponent
PC GUI Code – ModemIF.java

/* ModemIF.java
 * Processes network connection to the master node
 */
import java.net.*;

public class ModemIF implements Runnable {

    // Constants
    private double SCALE_POWER = 0.0102 * 0.004;
    private double SCALE_DIMMER_SLOPE = 21.0/100.0;
    private double SCALE_DIMMER_OFFSET = 10;

    // Data
    private double[] power;
    private double[] dimmer;
    private boolean[] active;
    NodeStatusCallback callback;
    String ip;

    // Thread
    Thread thread_comms;
    boolean initOK = false;
    boolean threadError = false;
    boolean needShutdown = false;

    // UDP
    DatagramSocket socket;
    InetAddress masterIP;

    ModemIF(NodeStatusCallback callback) {
        this.callback = callback;
        power = new double[16];
        dimmer = new double[16];
        active = new boolean[16];
    }

    // GUI calls this to shutdown thread and sockets
    public void shutdown() {
        needShutdown = true;
    }

    private synchronized boolean isInitOK() {
        return initOK;
    }

    private synchronized void setInit(boolean b) {
        initOK = b;
    }

    // GUI calls this to init connection
    public boolean[] init(String _ip) {
        ip = _ip;

        // Start comms
        thread_comms = new Thread(this);
        thread_comms.start();
    }
}
// wait until initialization is done
boolean i = initOK;
while(i == false) {
    i = isInitOK(); // must use a function due to Java threading issues
}

//System.out.println(initOK);

// check if error occurred
if(threadError)
    return null;
else
    return active;

// return current power for all nodes to GUI
public double[] power() {
    double[] ret = new double[power.length];
    for(int i = 0; i < power.length; i++) {
        ret[i] = power[i];
    }
    return ret;
}

// accept current dimmer from GUI
public void dimmer(double level, int ID) {
    dimmer[ID] = level;
}

public void update() {
}

// Threaded call - does all net processing
public void run() {
    initOK = false;
    needShutdown = false;

    byte rcvbuf[] = new byte[Common.MCUBUF];
    byte sendbuf[] = new byte[Common.PCBUF];
    DatagramPacket sendpacket;
    DatagramPacket rcvpacket;

    try {
        // Allocate resources
        socket = new DatagramSocket(Common.MCUPORT);
        socket.setSoTimeout(Common.TIMEOUT);
        socket.setBroadcast(true);

        sendbuf[0] = Common.INIT;
        sendpacket = new DatagramPacket(sendbuf, sendbuf.length, InetAddress.getByName(ip), Common.PCPORT);
        rcvpacket = new DatagramPacket(rcvbuf, rcvbuf.length);

        // Start initialization
// Send INIT until ACK rcv'd
boolean okay = false;
while(!okay) {
    socket.send(sendpacket);
    //System.out.println("INIT snt");
    try {
        socket.receive(rcvpacket);
        if(rcvpacket.getData()[0] == Common.ACK) {
            okay = true;
            //System.out.println("ACK rcv'd");
        } else {
            //System.out.println("BAD - " + rcvpacket.getData()[0] + " rcv'd");
        }
    } catch(SocketTimeoutException err) {
        okay = false;
    }
}

// Extract active nodes from ACK message
byte active0 = rcvpacket.getData()[1];
byte active1 = rcvpacket.getData()[2];
int bit = 0x01;
for(int i = 0; i < 8; i++) {
    active[i] = ((active0 & bit) > 0);
    active[i+8] = ((active1 & bit) > 0);
    bit = bit << 1;
} //for(int i = 0; i < 16; i++) {
    // System.out.print(active[i] + " ");
    //
    //System.out.println();

    // extract correct IP address
    masterIP = rcvpacket.getAddress();
    //System.out.println("Master Node IP: " + masterIP);
    sendpacket.setAddress(masterIP);
    setInit(true);
    //System.out.println("INIT IS DONE");

    socket.setSoTimeout(Common.TIMEOUT);

    // Loop here until shutdown
    while(!needShutdown) {
        try {
            SendDimmer(sendpacket);
            socket.send(sendpacket);
            //System.out.println("DIMMER snt");

            socket.receive(rcvpacket);
            //System.out.println("End of Loop 1");
            if(rcvpacket.getData()[0] == Common.POWER) {
                ReceivePower(rcvpacket);
                //System.out.println("POWER rcv'd");
            }
        } catch(SocketTimeoutException err) {
            //System.out.println("SocketTimeoutException");
        }
    }
}
//System.out.println("End of Loop 2");

} catch (Exception e) {

}

// sleep to not overload the master node
Thread.sleep(Common.TIMEOUT);

} catch(SocketException err) {
    threadError = true;
} catch(Exception err) {

} // Clean up resources
if(socket != null)
    socket.close();

// Extract power from packet
private void ReceivePower(DatagramPacket rcvpacket) {
    for(int i = 0; i < 16; i++) {
        int tmp0 = rcvpacket.getData()[i*2+1];
        int tmp1 = rcvpacket.getData()[i*2+2];
        int tmp2 = tmp1 << 8;
        tmp0 = tmp0 & 0x00FF;
        tmp2 = tmp2 & 0xFF00;
        int tmp3 = tmp0 | tmp2;
        power[i] = tmp3*SCALE_POWER;
    }
    //for(int i = 0; i < 16; i++) {
    //    System.out.print(power[i] + " ");
    //}
    //System.out.println();
}

// Setup and send dimmer data into a packet
private void SendDimmer(DatagramPacket sendpacket) {
    byte sendbuf[] = new byte[Common.PCBUF];
    for(int i = 0; i < 16; i++) {
        sendbuf[i + 1] = (byte)(SCALE_DIMMER_SLOPE*dimmer[i]+SCALE_DIMMER_OFFSET);
    }
    sendbuf[0] = Common.DIMMER;
    sendpacket.setData(sendbuf);
}
PC GUI Code – NodeData.java

/* NodeData.java */
public class NodeData {
    public double power;
    public double dimmer;
    public int ID;
    public String name;

    public void setPower(int power) {
        this.power = power;
    }
    public double getPower() {
        return power;
    }

    public void setDimmer(int dimmer) {
        this.dimmer = dimmer;
    }
    public double getDimmer() {
        return dimmer;
    }

    public void setName(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
}

PC GUI Code – NodeStatusCallback.java

/* NodeStatusCallback.java */
public interface NodeStatusCallback {
    public abstract boolean statusChanged(boolean[] active);
}
PC GUI Code – Common.java

/* Common.java */
public class Common {
    public final static int PCPORT = 7701;
    public final static int MCUPORT = 7702;
    public final static int TIMEOUT = 500;
    public final static int MCUBUF = 33;
    public final static int PCBUF = 17;
    public final static int INIT = 25;
    public final static int ACK = 98;
    public final static int DIMMER = 76;
    public final static int POWER = 12;
}

PC GUI Code – AboutWindow.java

/* AboutWindow.java */
import java.awt.*;
import javax.swing.*;
@SuppressWarnings("serial")
public class AboutWindow extends JDialog {
    JLabel label;
    JPanel pane;

    public AboutWindow(JFrame frame) {
        super(frame, "About", true);
        setLayout(new FlowLayout());

        pane = new JPanel();
        label = new JLabel("<html>Wireless Dimmer Control Using Power Line Communication</html>");
        label.setFont(new Font("SansSerif", Font.BOLD, 16));
        label.setHorizontalTextPosition(SwingConstants.CENTER);
        pane.add(label);
        add(pane);

        pane = new JPanel();
        label = new JLabel("<html><br>This program was developed by</br>" +
            "<p>Bernard Grégoire</p>" +
            "<br>with the help of</br>" +
            "<p>Dustin Morscheck</p>" +
            "<br>as part of the requirements for the degree of</br>" +
            "<br>Bachelor of Science</br>" +
            "<br>in</br>" +
            "<br>Electrical and Computer Engineering</br>" +
            "<br>in the</br>" +
            "<br>Faculty of Engineering</br>" +
            "<br>of the</br>" +
            "<br>");
        add(pane);
    }
References

http://www.webopedia.com/TERM/W/Wi_Fi.html [March 10, 2013]

Technology," Electronic Design [Online]. Available:  
http://electronicdesign.com/energy/power-line-communications-emerge-core-networking- 
technology [March 10, 2013]

Technology Inc., 2012.


[5] Microchip, "MRF24WB0MA/MRF24WB0MB Data Sheet 2.4 GHz, IEEE Std. 802.11b RF  


[7] Bel Fuse Inc., "Type 5MF / 5MFP Fast Blow Fuse Series". Jersey City, NJ, USA: Bel Fuse  
Inc., 2012.

General Semiconductor, 2011.

[9] ROHM Semiconductor, "1A Variable/Fixed Output LDO Regulators," Rev. 001. Kyoto,  

AZ, USA: Microchip Technology Inc., 2010.


[22] ON Semiconductor, "LM358 - Single Supply Dual Operational Amplifiers". Phoenix, AZ,
USA: Semiconductor Components Industries, LLC, 2011.


