Title

Power Capture (PowCap) Board for Non Intrusive Load Monitoring and Power Line Communication Exploration and Development

Author

Balakrishnan, Vikram

Publication Date

2013

University of California
Los Angeles

Power Capture (PowCap) Board for Non Intrusive Load Monitoring and Power Line Communication Exploration and Development

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Electrical Engineering

by

Vikram Balakrishnan

2013

Abstract of the Thesis

Power Capture (PowCap) Board for Non Intrusive Load Monitoring and Power Line Communication Exploration and Development

by

Vikram Balakrishnan

Master of Science in Electrical Engineering

University of California, Los Angeles, 2013

Professor Greg Pottie, Chair

Mains electrical lines in buildings and houses contain a wealth of information about devices that can potentially be exploited through research. However, accessing information through the circuit breaker can be quite difficult due to dangerous high voltages and lack of infrastructure for data logging. The Power Capture Board (PowCap) system was created to solve this problem of end-to-end access. It stores high quality analog signals from a circuit breaker to an open centralized database. It is composed of a customized high power tolerant analog front-end block connected to a digitizer that is an Ettus Research software defined radio board (USRP). The USRP has the ability to collect high-resolution analog voltages with a 24-bit ADC and generates 1 Terabyte of high-resolution data a day. These rich datasets are downloadable through the web and can now be generated on a daily basis to provide signal processing opportunities that did not exist before.

The thesis of Vikram Balakrishnan is approved.

William Kaiser

Wentai Liu

Greg Pottie, Committee Chair

University of California, Los Angeles

2013

This thesis is dedicated to the numerous individuals at UCLA and at home who worked hard to give me a second chance; this paper would not exist if not for their efforts and belief in me.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This work could not have been completed without my comrades at the NESL lab. "No graduate student left behind!"

CHAPTER 1

Introduction

With urban development, energy expenditure in cities increases dramatically. According to a report from Department of Energy [Hut99], office equipment will be the fastest-growing commercial end use sector between 1998 and 2020. The recent data from the Commercial Buildings Energy Consumption Survey (CBECS), a national sample survey project of the U.S. Energy Information Administration, reported that 19% of the total energy of U.S. office buildings is attributed to plug load energy use, such as office equipment, the computers, and other energy use [plu11]. So their power management is important. In a demand response (DR) framework, a building gets signals (demands) from the utility to reduce the power of appliances (response) [AE08, AE07].

However, plug loads are very diverse; just blindly shutting them off can be very problematic as they may not be safely turned off. Therefore, classifying the type of plug load and recognizing the state of the load is useful for the building energy management system. For example, the energy management system can shut down preempted loads (such as lamps or idle computers) and save non-preempted loads (such as busy computers).

My research was inspired by an excellent paper out of The University of Washington[GRP10] showing that the electrical noise traveling along the AC signal in an average home contains clues about the devices running on the circuit. When devices are turned on or off, they leave a transient marker on the line. Additionally many modern devices utilize switching-mode power supplies (SMPS) for their more compact design and higher efficiency, but this switching leaves traces on the mains AC signal in the form of high frequency noise. By processing these signals, it is possible to identify which devices are running and how much power they are using. Looking at noise on the AC line is not a new idea, but the algorithms that exist

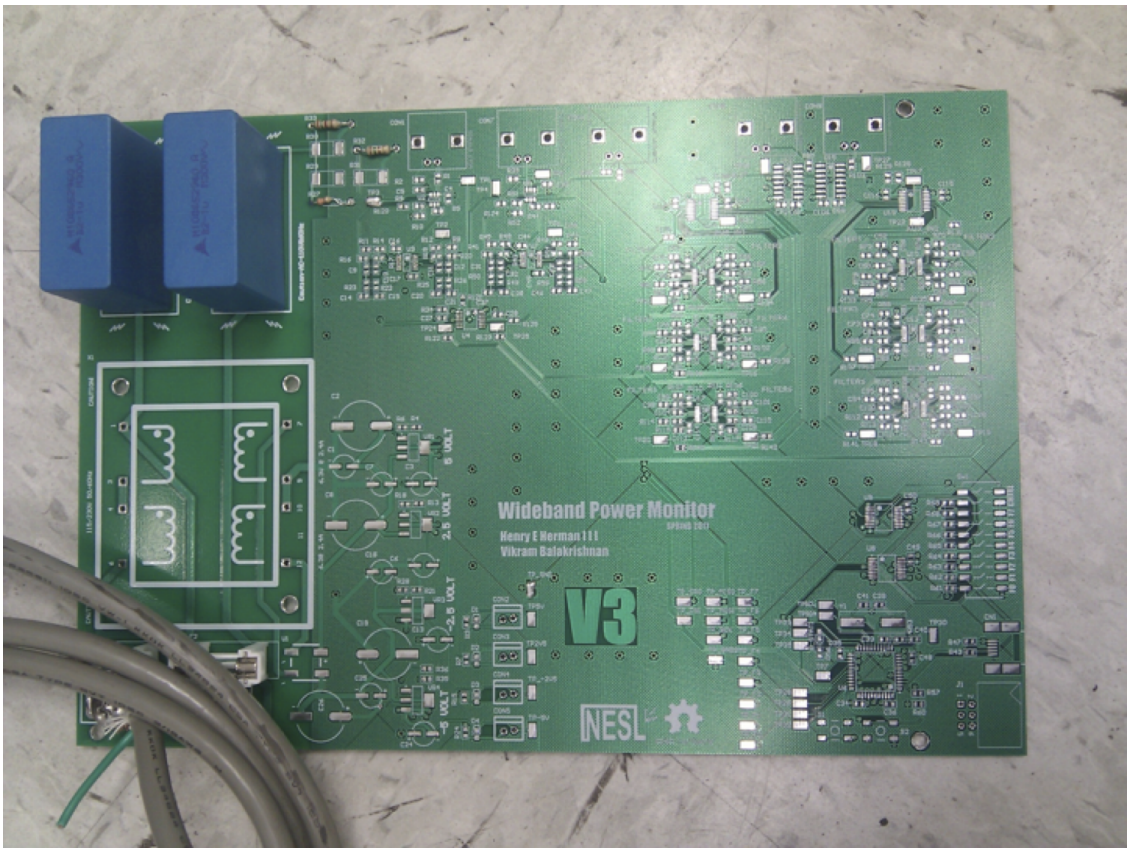

Figure 1.1: Power Capture Board, revision 1.

for device detection are often closely guarded secrets of the groups that develop them. The Power Capture Board (PowCap) was thought up as a universal hardware platform that could be used to spur more open research into this area (see Figures 1.1 and 1.2), in addition to providing a versatile general use wide-band analog front end.

The rest of this paper is organized as follows: Chapter 2 describes the design of our first revision hardware unit, the Power Capture Board; Chapter 3 goes over design and analysis of a power line transceiver; Chapter 4 discusses our final hardware revision and analyzes the data from the most recent deployments and Chapter 5 presents an experiment and analysis accomplished with our hardware system.

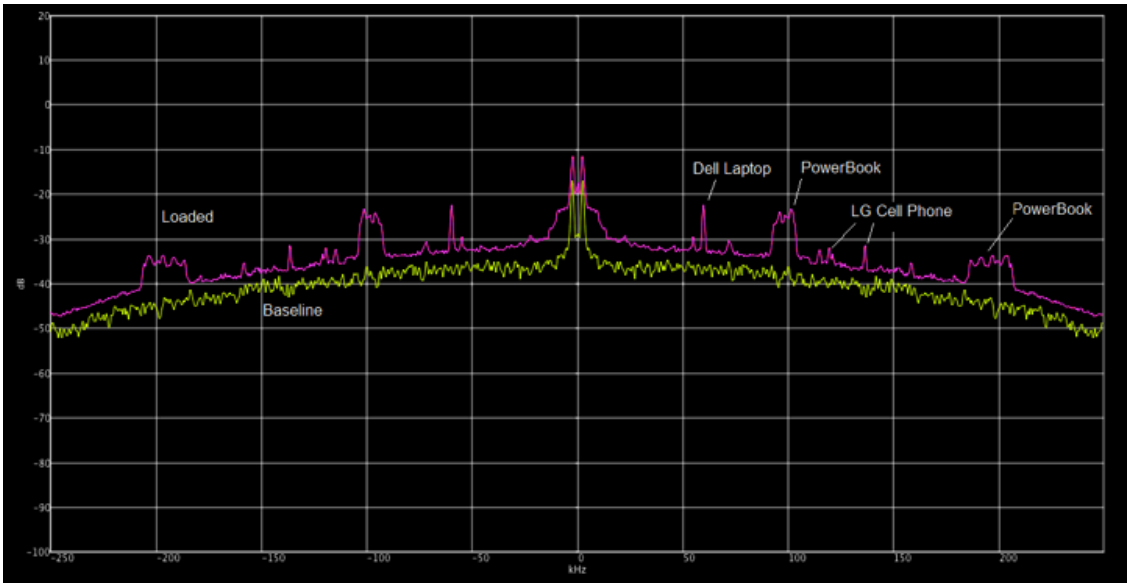

Figure 1.2: An example of the noise that can be seen on the line created by, among other things, power supplies of consumer electronics.

CHAPTER 2

The Power Capture Board (PowCap)

The Power Capture Board (PowCap) is designed to take in two signals: AC voltage and AC current. The voltage reading is picked up directly off the mains via capacitive coupling that is then sent through a resistor divider that attenuates the signal to safe voltages. The capacitive coupling serves an additional safety purpose: the board and users are protected by Y-rated safety caps designed to breakdown "open." The current is delivered from a wideband current transformer via BNC connector. Once these signals are captured, the user has several options. For power measurements, the main 60Hz AC signal is of primary concern, but for any noise analysis, the 60Hz line will dominate readings. To filter out this AC wave, a Twin-T notch filter that can be accessed through a digital interface to a programmable microcontroller. From there, the signal can be filtered again with a full range from a low pass of 1KHz to a high pass at 1MHz and six band pass ranges in between. These filters are all accessible through the microcontroller, and they are all optional; for each filter stages, there exists a bypass option. Finally, a programmable gain stage at the output readies our signal before being fed to an external USRP unit. Each of these stages is described in Figure 2.1. Finally, the board contains its own linear power supply built in to supply noise free DC rails to all of the ICs.

While designed with AC mains analysis in mind, the board has been shown to be remarkably versatile. The features discussed have been used to read and condition a wide variety of analog signals and prepared them for digitization through a USRP device. GnuRadio is a powerful open platform toolkit designed for software-defined radio construction, this makes it particularly easy to apply to any analog signal analysis.

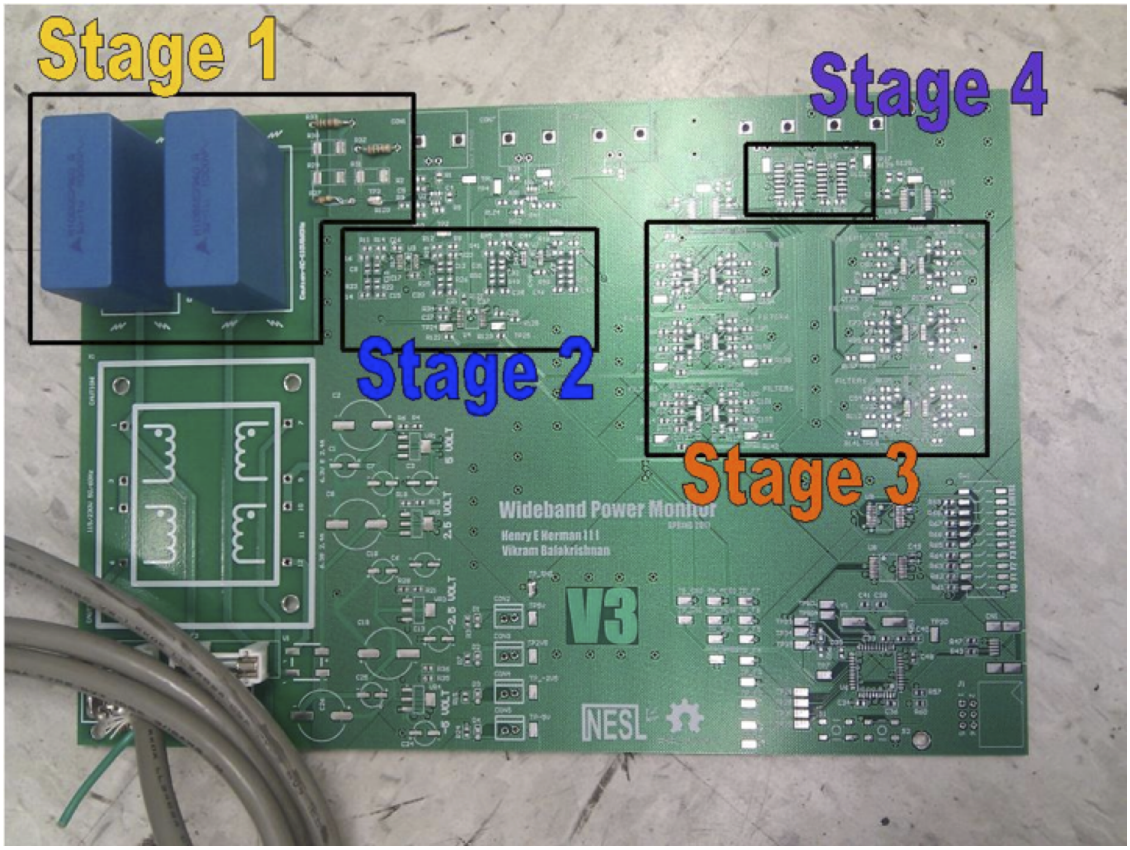

Figure 2.1: The main signal path of the PowCap board contains four stages: 1) Coupling 2) Notch Filtering 3) Band Filtering 4) Gain.

2.1 Input Connections (Stage 1)

Stage 1 (see Figure 2.2) is our voltage and current input: how we bring these signals from the real world into our Analog circuitry. The voltage and current front ends are explained in more detail below. After this input stage, the following conditioning stages are mirrored for current and voltage paths.

2.1.1 Current Front End

Our current sensor is a high quality wideband current transformer from Pearson Electronics, Inc[Inc]. A high end model was selected for its feature set. The inductor coil is able to open completely so that it can be attached around fixed wires, as would be expected with most main lines that come in from the street, without needing to cut them. This transformer

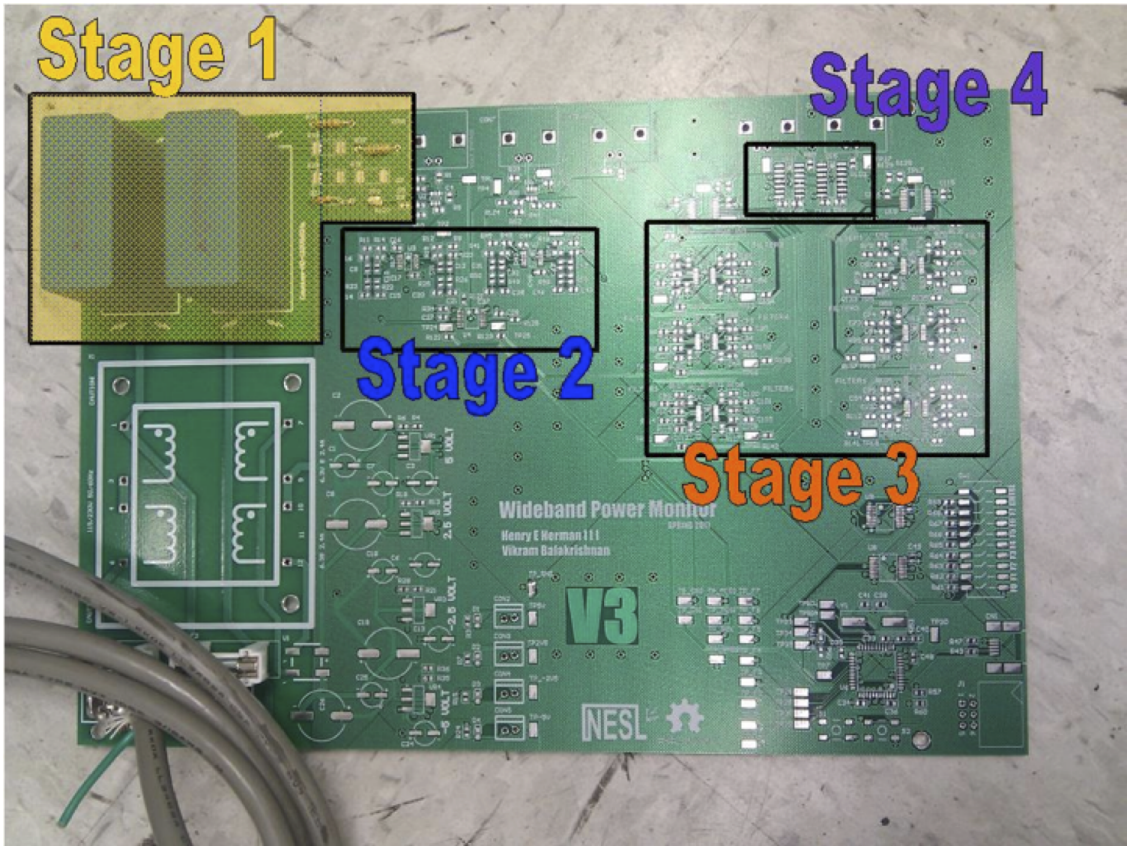

Figure 2.2: The first stage of the signal path on the PowCap board: Capacitive coupling and a voltage divider for the voltage path. (Note: BNC connections bring in the current signal from an external current transformer.)

measures the current of the enclosed wire, and then produces a voltage output. It has a 50ohm output impedance. The 411C[pea] has a band pass of 25Hz to 20MHz and the sensitivity is 0.1V/A. This voltage output is available through a BNC connection that plugs directly into the PowCap board. From there, the signal can connect to a Twin-T Notch or be sent directly to the filter bank. While most switch mode power supplies[Pre98] switch in the frequency range of 50kHz to 1MHz we are interested in seeing what other types of EMI might be present. Low frequency events such as motors will also likely be able to be identified. Future applications could easily use a less advanced current transformer.

2.1.2 Voltage Front End

Figure 2.2 shows our voltage input consisting of a simple voltage divider in series with safety rated capacitors. Y2 capacitors are rated for up to 300VAC, but are designed to withstand much larger voltage spikes. Y2 caps should be used whenever their failure could result in electric shock. There are multiple stages of self-healing dielectrics and, if the dielectrics can't heal, the capacitor is designed to fail "open." All of these characteristics make Y rated capacitors perfect for protecting users and the board from the high voltage mains. Behind these capacitors is a simple voltage divider that feeds into a buffer circuit. It should be noted that this voltage front end acts like a high pass filter, with the cutoff frequency set by the values of the voltage divider resistors (in our design, about 20Hz).

The LTspice model is available as a supplement.

2.2 Preliminary Filter (Stage 2)

Stage two (see Figure 2.3) provides an option of filtering out of the 60Hz mains signal via a Twin-T Notch Filter.

After the signal is lowered to a reasonable voltage (nominally 4.2V), it is isolated with a buffer circuit and then passed into the first filter stage. In order to more easily observe higher frequency effects, it is essential to filter out the 60Hz mains signal. This improves our ability to analyze the residual high frequency noise. This low amplitude "noise" was originally riding on the 60Hz wave, but there is a risk of losing vital fidelity due to the high amplitude 60Hz signal. We make this noise the dominant signal by removing, or filtering, this 60Hz signal, giving us a much better look at the high frequency switching signals, commonly referred to as EMI, riding on top of the AC signal. However, if we want to calculate fundamental usage statistics, like power, we need to be able to look at the 60Hz wave; to provide flexibility, the filter is fed through a MUX along with a bypass line. Through the use of an on-board microcontroller, or manual on-board switch, a multiplexer can send either signal path through to the next stage. The implemented 60Hz notch filter topology consists of two cascaded active

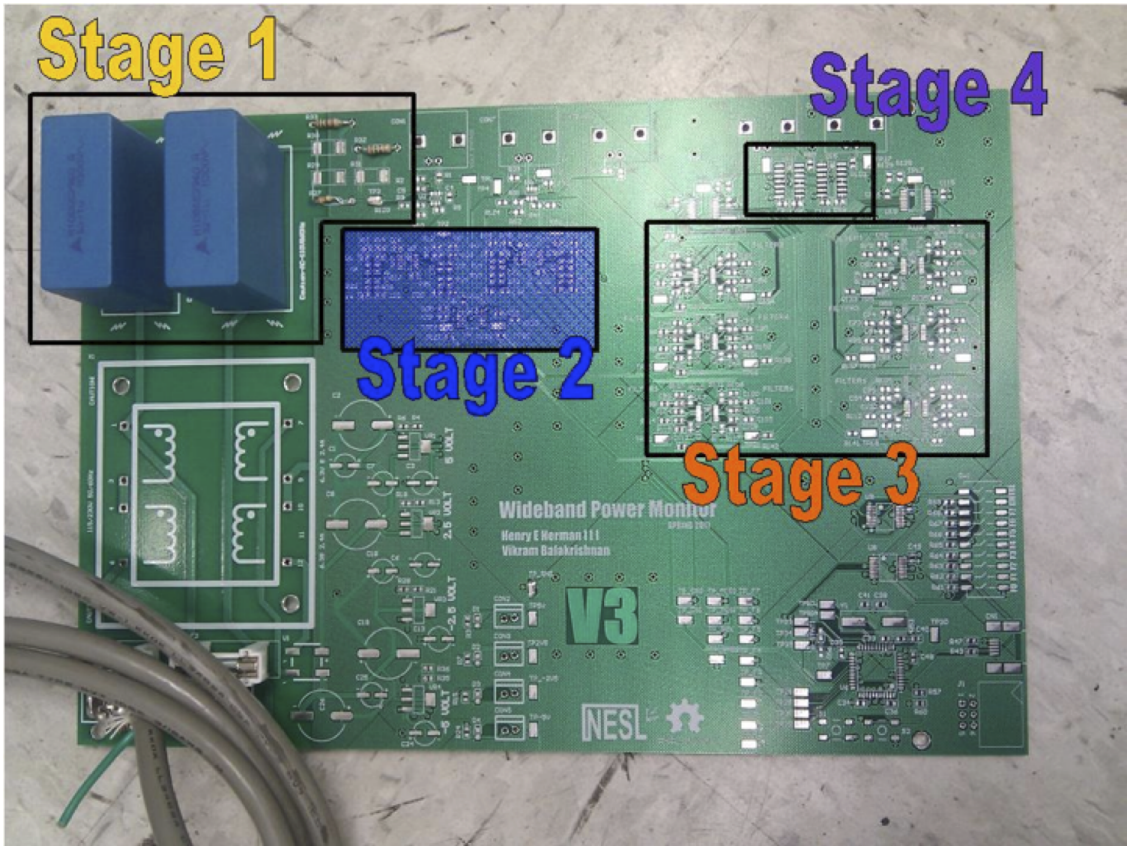

Figure 2.3: The second stage of the signal path on the PowCap board: a Twin-T notch filter on both current and voltage signal paths.

Twin-T notch filters[Inc69]. The entire stage was simulated using LTspice[Tec] (see Figure 2.4), a free circuit design software package from Linear Technology.

2.3 Variable Filter Stage (Stage 3)

After being isolated through another buffer stage, the signal is sent to a variable filter array (see Figure 2.5). This stage offers us the option to apply custom analog filtration to identify and isolate the most data rich, or EMI rich, frequency bands. For example, many switching power supplies operate in the high tens of kilohertz. For this range it would be sensible to run our signals through the 1KHz-100KHz band-pass filter.

We include the following analog filters on our board to do this without losing any information: 1kHz low-pass, 1kHz to 100kHz band-pass, 100kHz to 250kHz band-pass, 250kHz

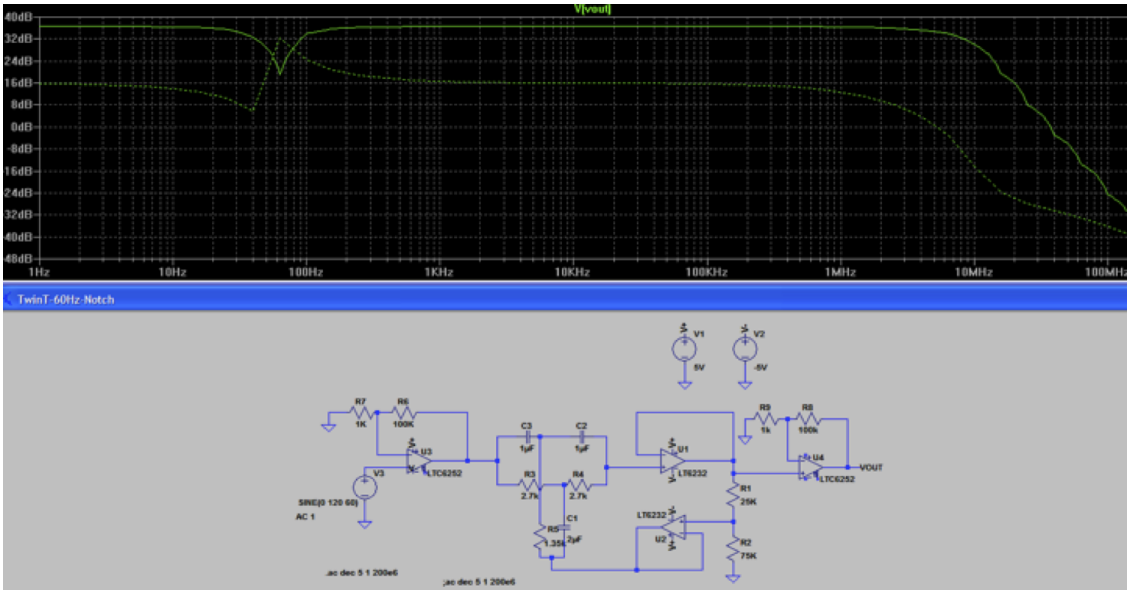

Figure 2.4: Schematic and frequency analysis of the 60Hz Twin-T Notch Filter.

to 500kHz band-pass, 500kHz to 2MHz band-pass, and 2MHz high-pass filter. Our filters are designed with a dual op-amp Sallen-Key topology[Jon01]; a well-documented topology allowing for easy filter edge modification. Supplementary LTspice models are available for further exploration of the filter response of these circuits.

2.4 Programmable Gain Amplifier (Stage 4)

The final step before the signal leaves our board is a programmable gain stage (see Figure 2.6). This allows us to condition our signal so that it is appropriately scaled for its destination. For example, the Analog-to-Digital Converter (ADC) on a USRP require an input signal of no more than 2V peak-to-peak swing. At the same time, if the signal is significantly lower than 2Vp-p, we can boost the voltage so that we do not lose any resolution; that is we can maximize our dynamic range. For this task, we employ National Semiconductor's LMP8100 Programmable Gain Amplifier[lmp].

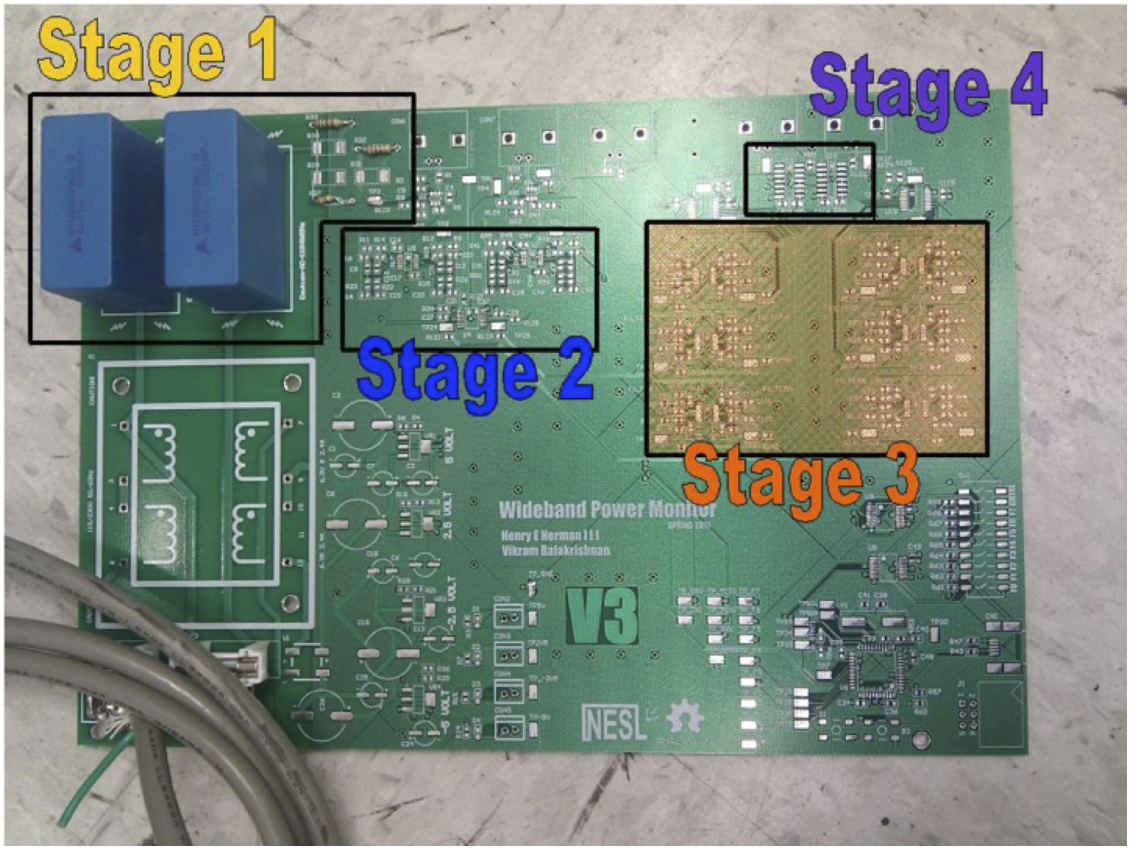

Figure 2.5: The third stage of the signal path on the PowCap board consists of an array of filters for each signal path: a 1kHz low-pass, 1kHz to 100kHz band-pass, 100kHz to 250kHz band-pass, 250kHz to 500kHz band-pass, 500kHz to 2MHz band-pass, and a 2MHz high-pass filter.

2.5 Supplementary Blocks

All ICs need power and on a noise-sensitive board like PowCap, a particularly "clean" power supply was needed. An off-the-shelf switching power supply is potentially risky as, even with good routing practice, the supply's switching noise could bleed throughout the board and affect the primary signal path. Buying an external linear power supply seemed wasteful, as a custom power supply could easily be designed/implemented. A custom power supply (PSU) also gives full flexibility to accommodate any voltage rails required, which expands our choices for ICs. Power is provided by a parallel path from the wall to a Tamura 120/10V transformer[tam] (see Figure 2.7). The AC signal is half-wave rectified and then regulated

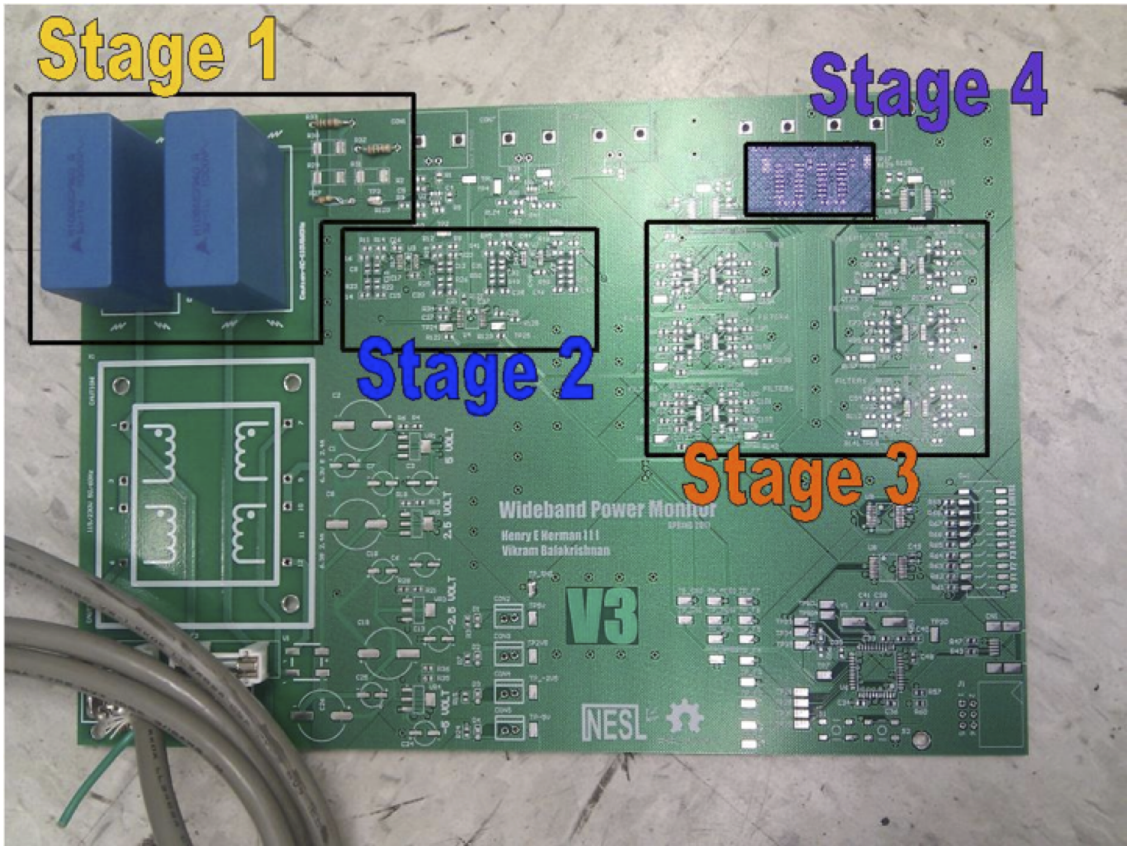

Figure 2.6: The fourth stage of the signal path on the PowCap board: programmable gain amplifiers for each signal path.

into four rails: +/- 2.5V and +/-5V. The lower voltage range is required by the quad op-amps used in all of our filter topologies, while the MUX's and microcontroller require the 5V rail.

For the multiplexer choices, several combinations of 2x1 and a 8x1 MUXs were incorporated to accomplish computer controlled routing between and around the filters (see Figure 2.7). A 2x1 MUX sets whether the 60Hz notch or the bypass path is outputted. A larger 8x1 MUX connects all of the Low/High/Band -pass filters, still leaving a couple of connections free for additional inputs that can be routed manually. Again, the circuitry is mirrored, once for the voltage path and once for the current path; that is why the majority of signal circuitry appears to be duplicated. For troubleshooting and as a backup, a dipswitch is attached to all mux connections to allow for manual switching between pathways.

The MUX's along with the output gain stages can all be controlled by a programmable mi-

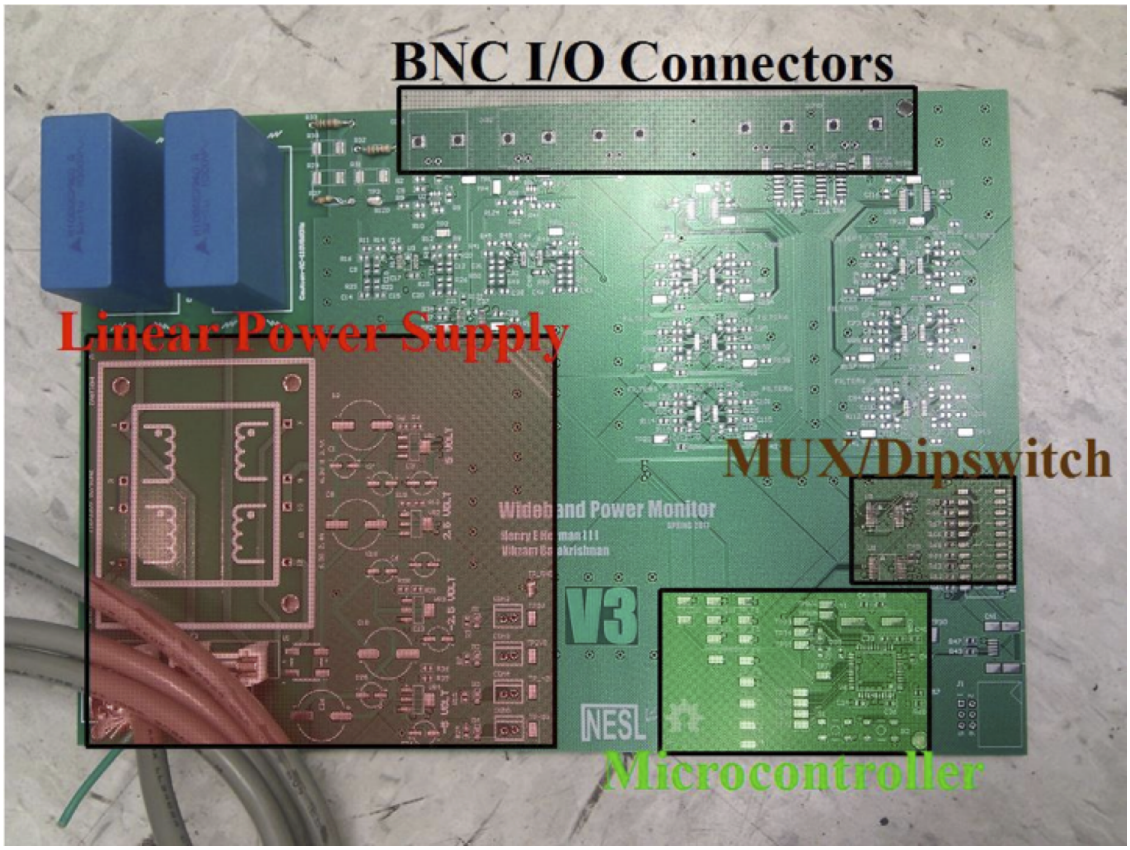

Figure 2.7: This figure highlights the main supplementary blocks on the PowCap board: the power supply (PSU), input and out(I/O) connections, mechanical switches, and microcontroller (MCU).

crocontroller (see Figure 2.7). The microcontroller utilized on PowCap is an ATMEGA32U4; it is a simple 8-bit controller with USB interface[atm]. This is the same chip used in common DIY development boards like the Teensy[PR]. The software uses a lightweight USB framework for AVRs (LUFA[Cam]). It is loaded with a DFU USB "bootloader" so that it can be programmed without using any specialized programming hardware. The firmware can be viewed in the appendix.

For easy testing and interface with other devices, I/O is accomplished with an array of BNC connectors (see Figure 2.7) that can: 1) insert input signals, connected or disconnected with 0ohm resistors on the board. 2) Output our conditioned signals from our programmable op-amps. These additional I/O paths are useful to test custom inputs through a signal generator, or to look directly at the output signal on a scope.

2.6 Prototyping

Like most projects, PowCap went through several levels of design and revisions. The following sections trace our path from theoretical conception to a highly functional analog interface that can sense both voltage and current; and amplify relevant bandwidths.

2.6.1 Voltage Divider

The first step was to see if 115VAC line voltage could be coupled onto a board and safely divided down to manageable amplitude. Due to local electronic stocks, the first design used X2 rated caps. X2 caps can take high voltage but don't fail open; they are recommended primarily for low-pass filtering. With the idea of a simple capacitive coupling, we began piecing together a simple circuit. The results were immediate, though it would take some fine-tuning to perfect the component values. The X caps were 2.2uF, though there were also experiments done with several different capacitor sizes (10nF and .1uF). The goal was to attenuate the voltage signal down to a level either the USRP could take in, 2Vp-p, and later a level the buffer could handle, about 2.5Vp-p. The resistor values had to be chosen carefully so that the resulting RC filter circuit does not have too high a cutoff frequency. That is, it is OK to lose some low frequency data, but it is important not to lose any high frequency switching noise.

A collection of various test topologies can be seen in Figure 2.8 , assembled on early stage prototype boards.

2.6.2 60Hz Filtering

After we could reliably extract our attenuated line voltage signal, the next step was to filter out the 60Hz. We picked up a simple single-stage topology from a filter design book, and put it onto a board. The op-amp needed power, so we designed a simple linear power supply that served as the foundation for the multi-rail PSU implemented in the PowCap board. The resulting prototype circuit can be seen in Figure 2.9. This early 60Hz filter model is

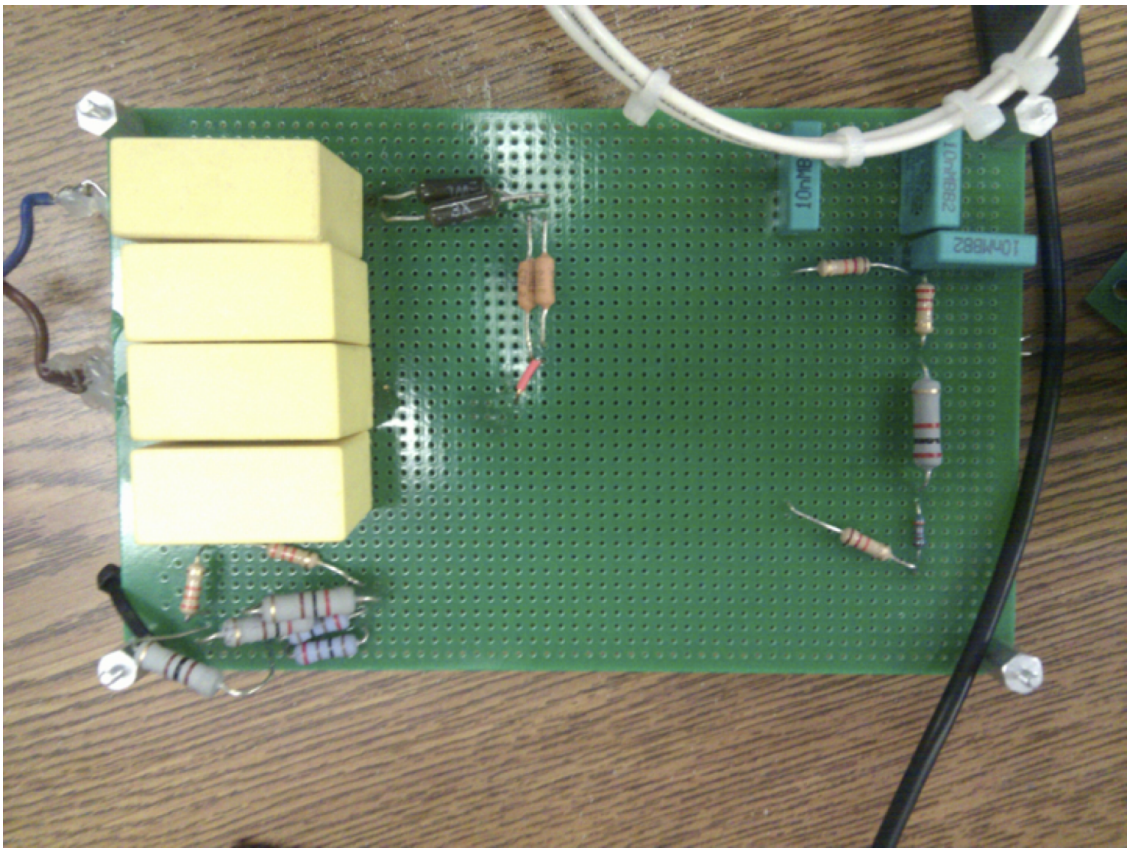

Figure 2.8: Prototype resistor divider circuit. The caps on the left are supporting two different topologies. The caps on the right are .1uF Y2 caps, here they are connected with some experimental values. The LTspice models from these experiments are submitted as supplemental documentation.

in the LTspice schematic in Figure 2.10. Simulations of the schematic are shown in Figures 2.11 and 2.12.

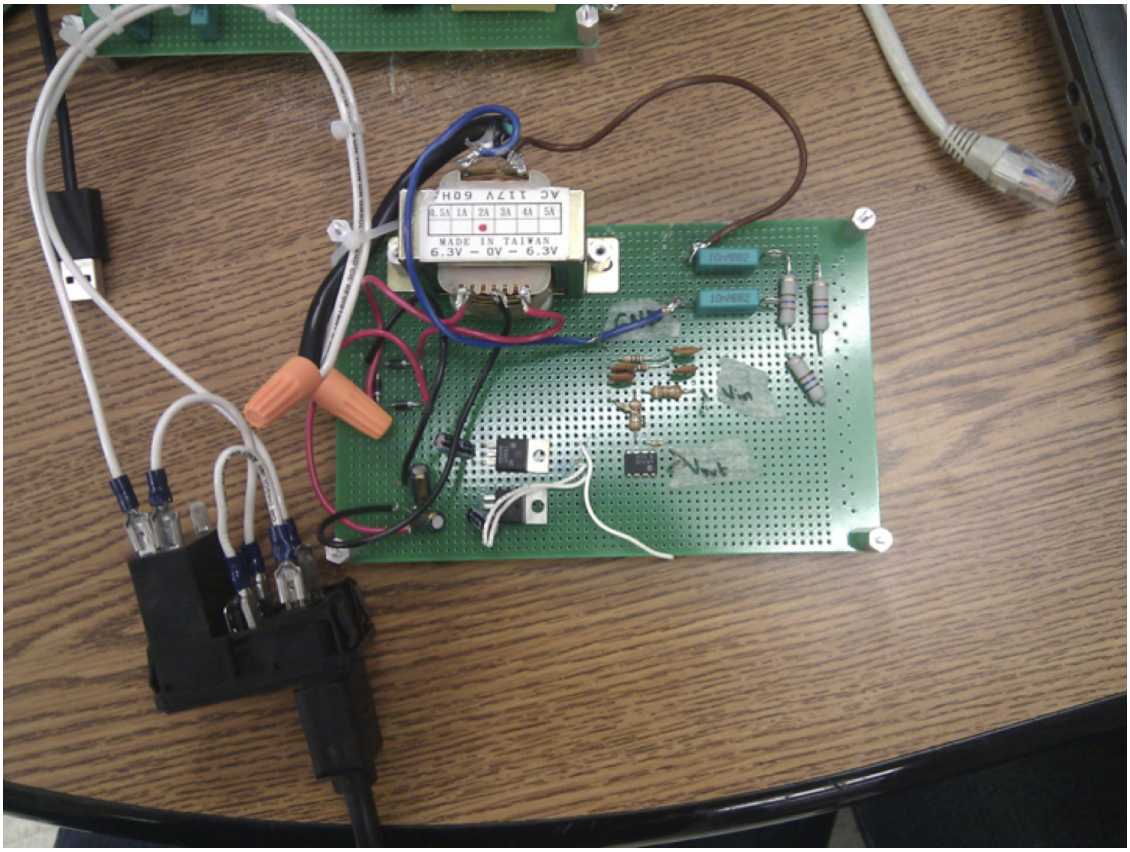

Figure 2.9: A prototype resistor divider and 60Hz notch filter. We are using a simple switch, with a built in fuse-holder, as a go-between for the line input to the board. This device can be seen as black housing with metallic protrusions on the bottom left. On the board, the transformer feeds stepped down voltage, at a 100:12 ratio, to a half-wave rectifier. The resulting capacitor-smoothed DC voltage is then sent through two voltage regulators to get rails of +/-2.5V to feed the op-amp. Not surprisingly, this is very similar to the design adopted in the PowCap board.

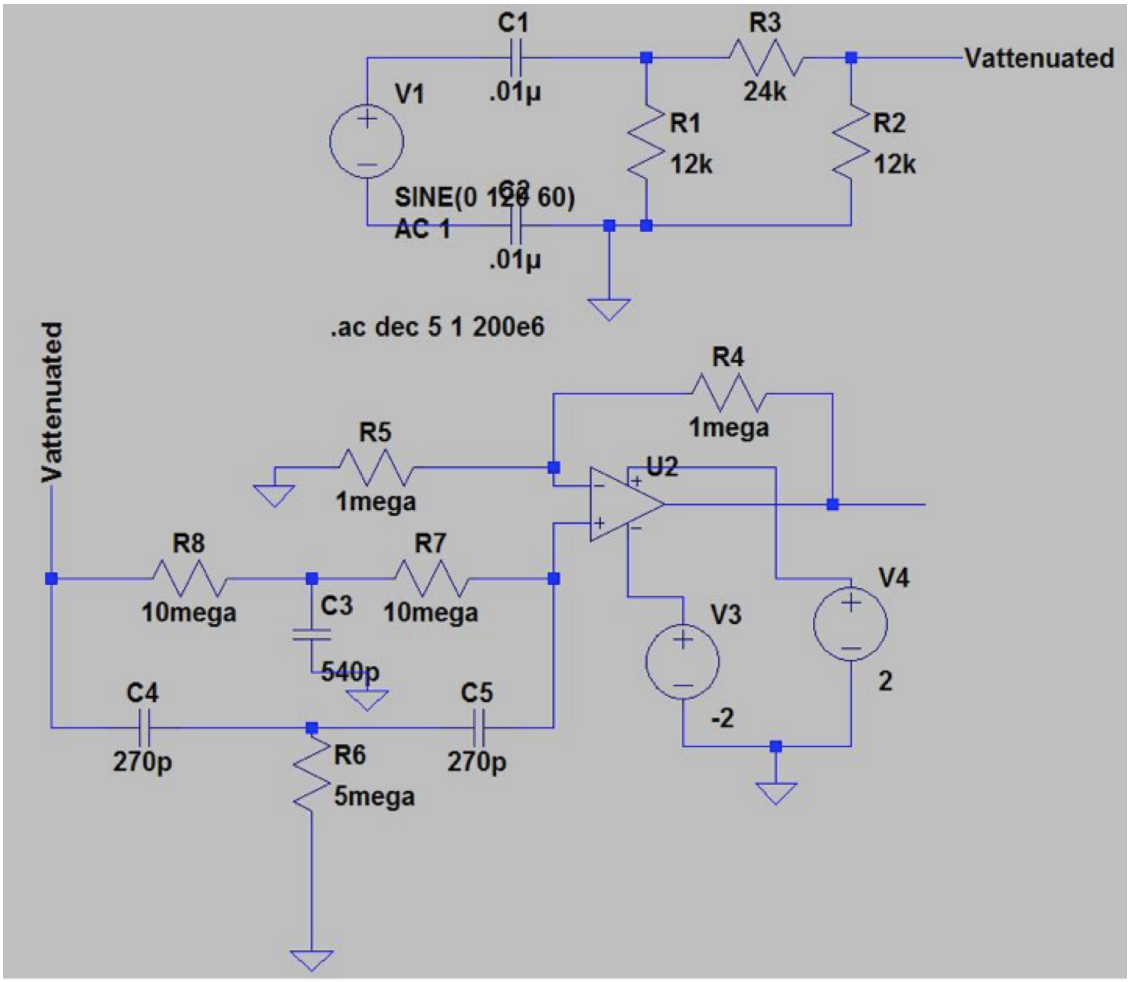

Figure 2.10: The prototype 60Hz notch filter design in schematic form. Included at the top of the diagram is the input capacitive coupling and resistor divider circuit. This was included due to a lack of a buffer stage between the input and the notch filter; as a result the notch filter is loading the resistor divider and vice versa.

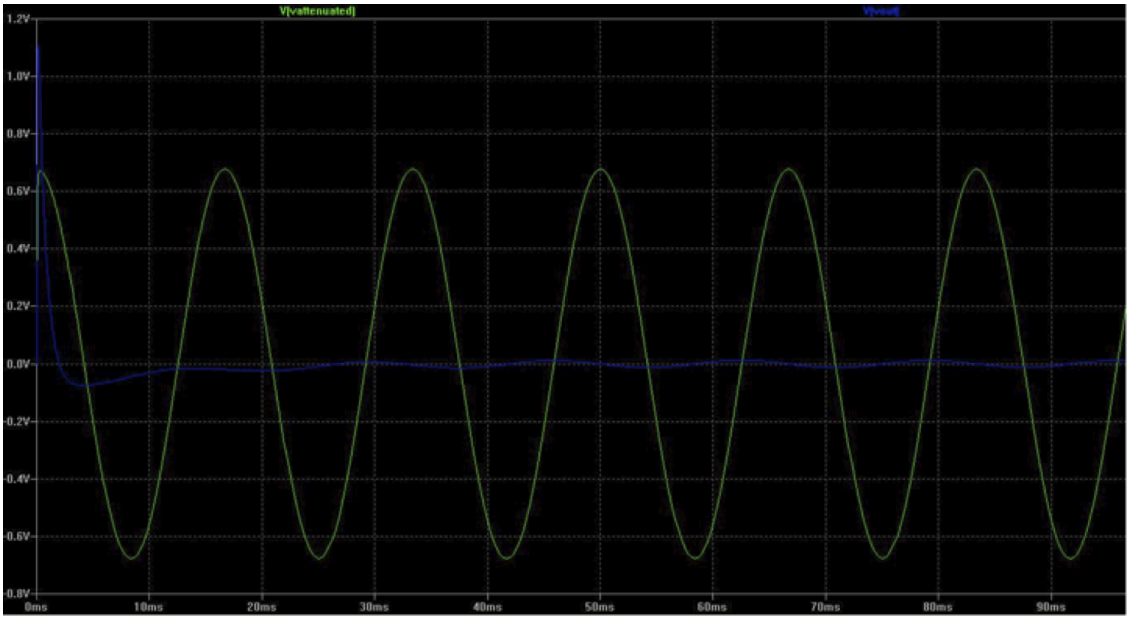

Figure 2.11: The prototype 60Hz notch filter signal response. The input 60Hz sign wave (green) is filtered out as shown by the output signal (blue).

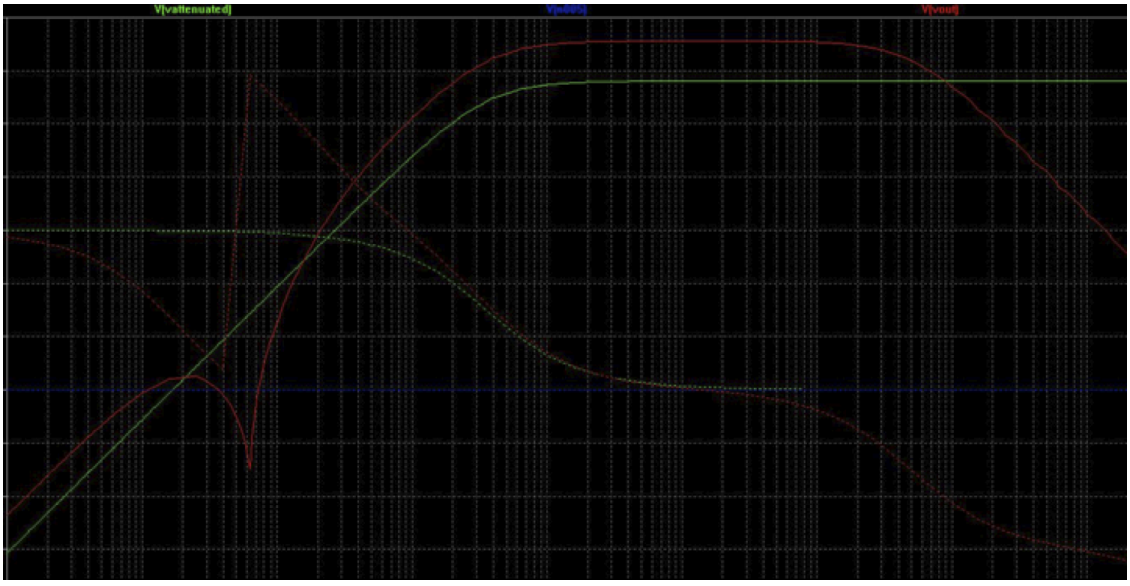

Figure 2.12: Frequency response of the prototype 60Hz notch filter.

CHAPTER 3

Power Line Communication

3.1 Introduction

While sensing information from the mains, it became clear that the effects of signal propagation along the circuits in the home have adverse and unpredictable effects on the signals themselves. To better understand these effects, it is helpful to have the capability to transmit, in addition to receive, signals across the lines. This gives us information to develop good circuit models, while also providing a good foundation to incorporate power line communication (PLC) capability into our designs. Already we know our original PowCap board will need two more channels, one voltage and one current, in order to be able to cover both phases of a typical home. There is some coupling between the phases, but there are also "dead zones" in one phase that are not visible when sensing on the other[GRP10]. For larger buildings, it is to be expected that dead zones will develop due to a combination of power line parasitics and the low amplitude of emi signals. In these cases, having additional boards that can be deployed to different areas and communicate between themselves for collaboration on load identification is a natural step for scaling up existing systems. The significance of PLC capability and additional applications for our PLC design are discussed further in chapter 6.

This chapter explores a working communication hardware platform from which a USRP could be used to modulate and then transmit or demodulate/receive signals over a conventional power line circuit in the house or workplace. The power lines or "mains" of a building makes for a very noisy environment, so it seemed that the same techniques developed for transmitting through increasingly crowded airwaves would adapt well to the EMI filled spec-

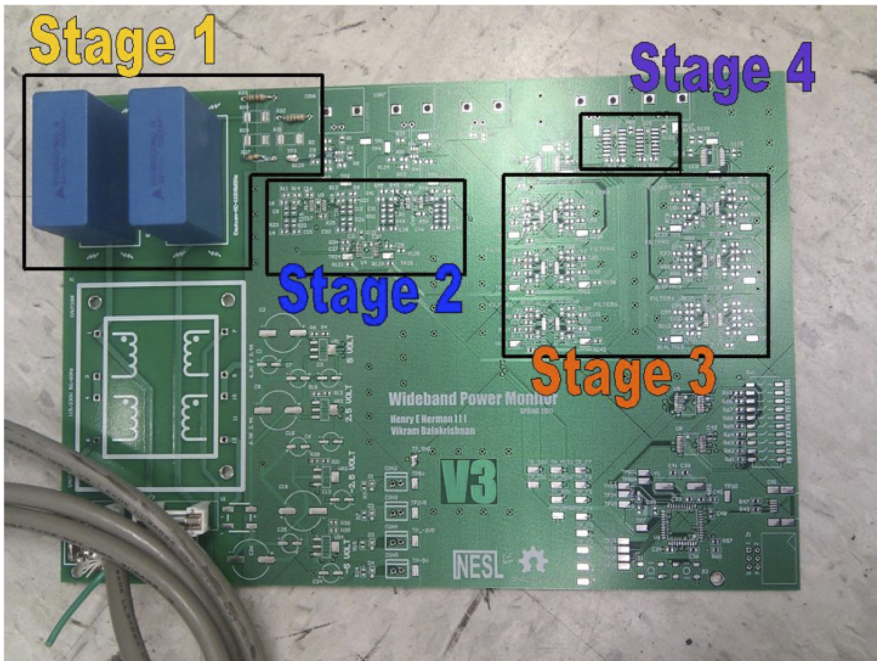

Figure 3.1: The Power Capture Board.

trum of power lines. However, as with any real-world application, the first step was to build functional hardware. Once I was comfortable transmitting simple signals, and eventually higher bandwidths of shifted baseband data, I transitioned to modulated signals, starting with AM and FM and created a reliable foundation for QAM and OFDM.

This chapter will begin with a simple explanation of the hardware along with difficulties encountered during design and deployment. While this application is heavy on communication theory, the lessons learned during hardware design had a strong effect on choosing transmission features. For example, I noticed a strong resonation with my coupling transformer and my high pass filter boosting my transmission power at 100kHz. So, taking advantage of this, I did much of my experimentation around this frequency. The extra power maximized my SNR and pushed my signal further through, often highly lossy, mains circuit lines.

3.2 Hardware Design

Figure 3.1 is a picture of the original Power Capture board (PowCap). As demonstrated earlier, PowCap is designed to take in two signals, AC voltage and AC current. The voltage

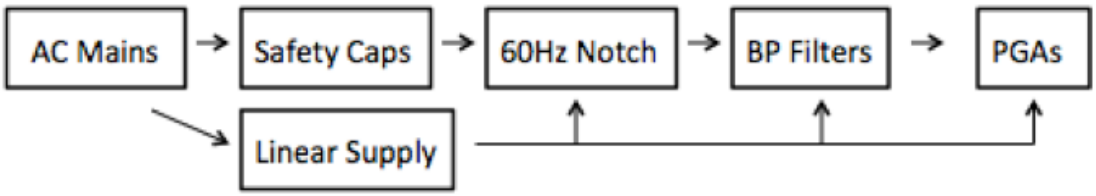

Figure 3.2: PowCap Board operational flowchart.

reading is picked up directly off the mains through a resistor divider that attenuates our signal to safe voltages; the board is protected by Y-rated safety caps. The current is delivered from a wideband current transformer via BNC connector. Once these signals are captured, the user has a lot of options. For power measurements, it is important to see the main AC signal, but for any noise analysis, the 60Hz line will dominate readings. To filter out the AC wave, we have a Twin-T notch filter that can be accessed through our interface for the programmable microcontroller. From there, the signal can be filtered again with everything from a low pass of 1KHz to a high pass at 1MHz and a half dozen band pass ranges in between. Again these filters are all accessible through the microcontroller, and they are all optional; for each filter stage, there exists a bypass option. Finally, there is a programmable gain stage at the output before the signal is fed to a USRP unit. This signal flow is modeled in Figure 3.2. The board also has its own linear power supply built in to supply noise free DC rails to all of the ICs.

While designed with AC mains analysis in mind, the board is actually quite versatile. The features discussed could be used to read and condition any analog signal and prepare it for digitization through a USRP device. GnuRadio is a powerful open platform toolkit designed for software-defined radio construction, but it is easy to apply to any analog analysis.

For this project, the PowCap board was my model for a power line receiver. At a minimum, all that was needed was to add was a transmitter. The PLC board (see Figure 3.3) is essentially a downsized PowCap board, with bi-directional functionality.

In addition to a high-power current amplifier to transmit a signal out, the method of coupling to the line was also changed. Instead of using Y capacitors, I instead used a coupling transformer; specifically a Schaffner Pulse Transformer[sch]. The benefit was a traditionally

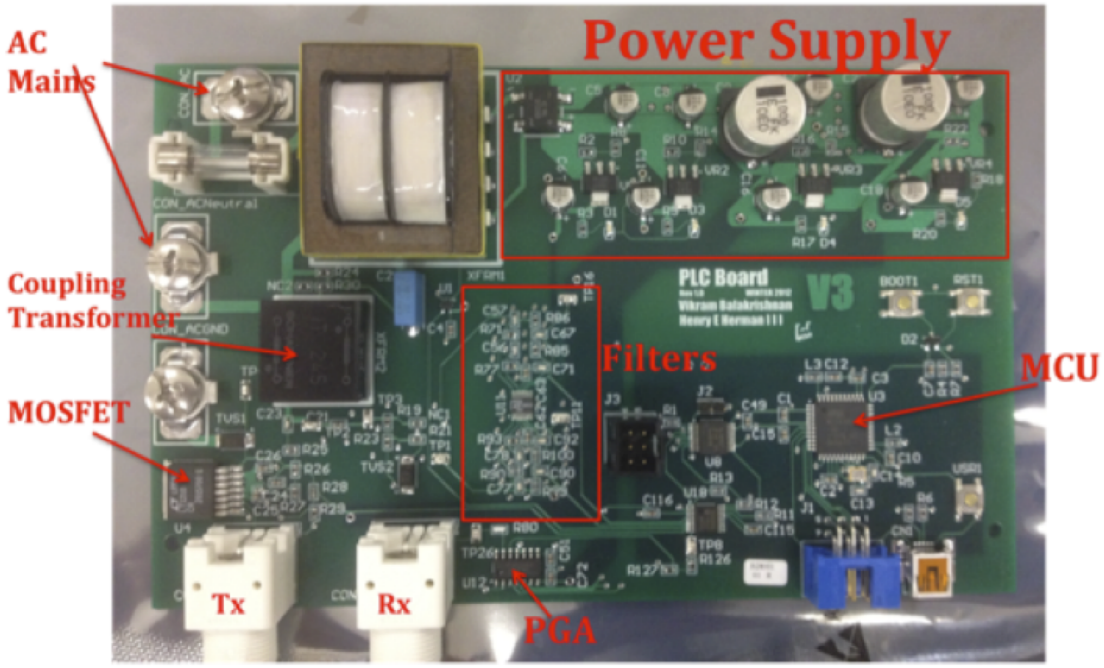

Figure 3.3: The Powerline Communication Board.

safer system, as transformers provide industry-standard, and consumer electronics required, galvanic isolation for the secondary. The risk of incorporating inductive coupling was potentially inducing some resonant effects, which would also affect performance of other active circuits like the filters. Sure enough, despite careful isolation, the frequency response of my board was not immune to the transformer's effects. In the attached video, two amplitude peaks can clearly be seen between 1kHz and 1MHz and that matches up with the phase diagram when analyzing part of the receiving circuit. A resonance effect is clearly visible at 100kHz. This effect was seen in my early simulations, as I did not expect my high pass circuit on the primary side to interact so severely with my band pass circuits on the secondary. It is only the combination of these effects that creates the frequency response seen in Figure 3.10.

This issue was addressed by removing one filter and turning it into a dual buffer stage. Now there is only a high pass effect that cuts off at 35kHz as can be seen in Figure 3.5. There is good reason to return to a coupling capacitor for the next revision, as it would allow me a wider access to the spectrum on the powerline.

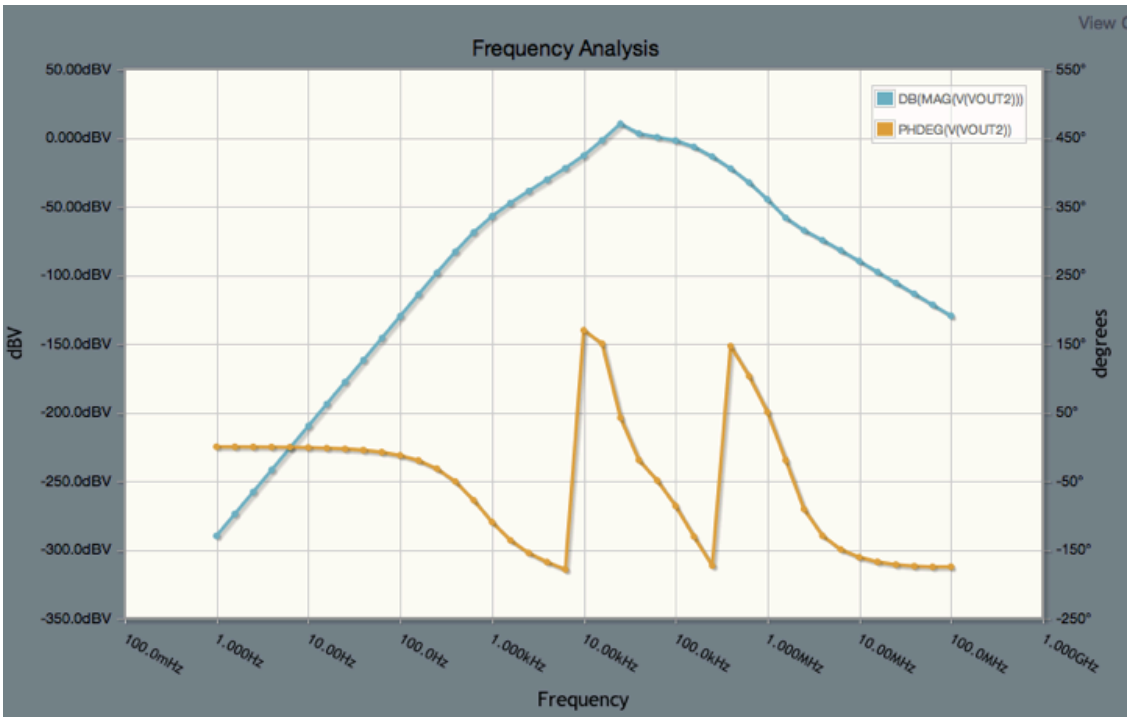

Figure 3.4: Frequency response of receiving circuit and bandpass filter with inductive effects of transformer.

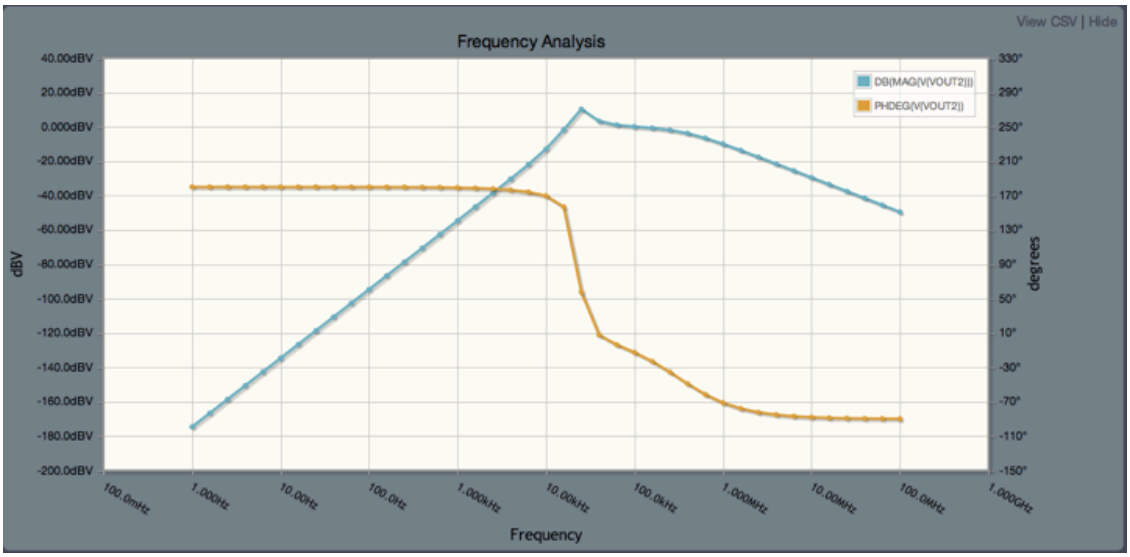

Figure 3.5: Frequency response of receiving circuit and bandpass filter with inductive effects of transformer.

The filters are designed with a Sallen-Key topology for easy analysis and even easier modification. Please look at the LTspice models if you are curious. Videos of its operation can seen in the attached documentation.

The final step before the signal leaves the board is a trip through a programmable op-amp. This allows the user to condition our signal so that it is appropriate for the destination of the output. For example, if we are feeding this signal into the Analog-to-Digital Converter (ADC) on a USRP, we would want to ensure that the output signal is not more than 2V peak-to-peak. At the same time, if the signal is significantly lower than 2Vp-p, we can boost the voltage so that we do not lose any resolution. For this task, we employ National Semiconductor's LMP8100 Programmable Gain Amplifier.

The MUX's along with the output gain stages can all be controlled by a programmable microcontroller. The microcontroller is a ATMEGA32U4; it is a simple 8-bit controller with USB interface. This is the same chip used in common DIY development boards like the Teensy. The software uses the lightweight USB framework for AVRs (LUFA). It is loaded with a DFU USB "bootloader" so it can programmed without using any programming hardware. The firmware is discussed in Appendix A.

The transmitter utilizes a Linear Technology 1210 Current Amplifier to drive our signal across the line. It is heavily over specified and is capable of delivering an amp of current. I did not want to risk the signal dissipating, or not propagating across functional distances, due to lack of power. It is now clear that much lighter FETs can be used, but the flexibility of a powerful transmitter was useful for analyzing power versus distance capabilities.

3.3 Baseband

Figures 3.6 and 3.7 show scope screenshots a recorded audio sample. The baseband signal is in the auditory range, 0-20kHz (Figure 3.6) which is then shifted to 100kHz-120kHz (Figure 3.6).

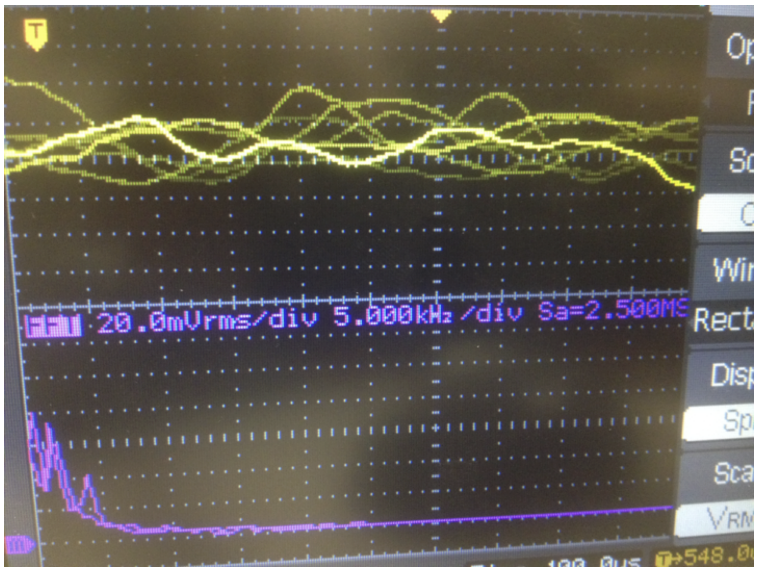

Figure 3.6: "Baseband" audio signal, DC to 20kHz.

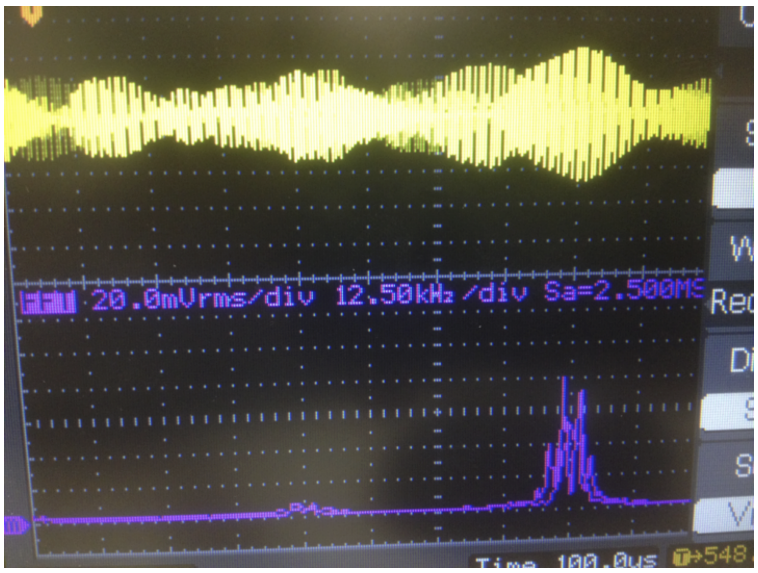

Figure 3.7: Baseband signal shifted to higher frequencies, 100kHz to 120kHz.

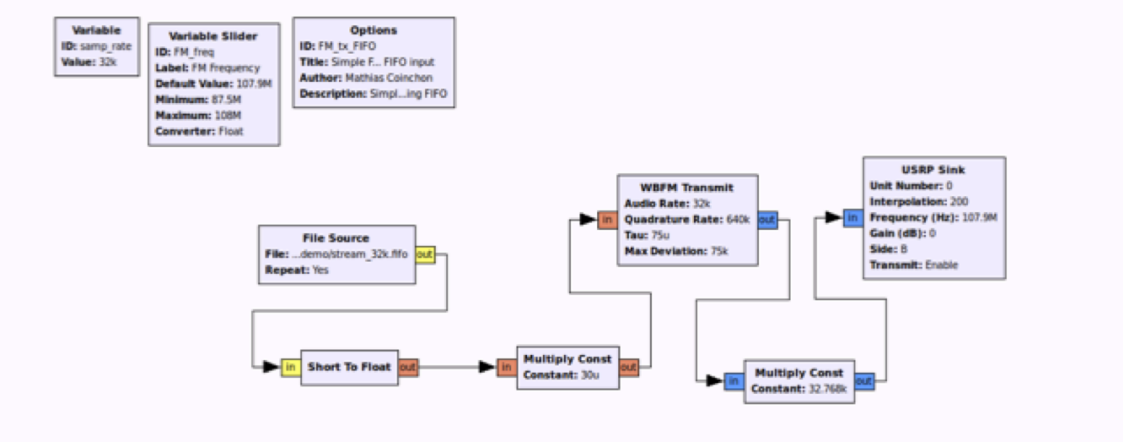

Figure 3.8: Gnuradio companion signal processing diagram for a simple FM transceiver.

3.4 Frequency Modulation

My first transmitter implementation is shown in Figure 3.8. I transmitted a 500kHz bandwidth signal at various points between 100kHz and 1MHz., and incorporated the narrow band FM block. My frequency deviation was 5kHz.

A FM transceiver was built, sourced from a data file containing my baseband audio data, by the block diagram. Again I was only transmitting a 500kHz bandwidth signal with 5kHz deviation. This self-imposed limitation was mainly due to hardware constraints. We will discuss environmental bandwidth limitations later in this chapter.

3.5 Evaluation

I have included videos of my experimentation. The first two show simple proof of concept: my board is able to both transmit and receive signals across household/lab electronic wiring. To provide a less noisy environment, which was necessary to analyze the behavior of my board, an isolation transformer was used as the "power line environment."

The next step was transmitting more complex data. This was accomplished by using an audio recording of my laptop playing a popular song, several minutes in length. This file was recorded and stored through the tools available in GNURadio. Since I had a high pass filter on the receiver path of my board, I could not transmit this signal at its baseband

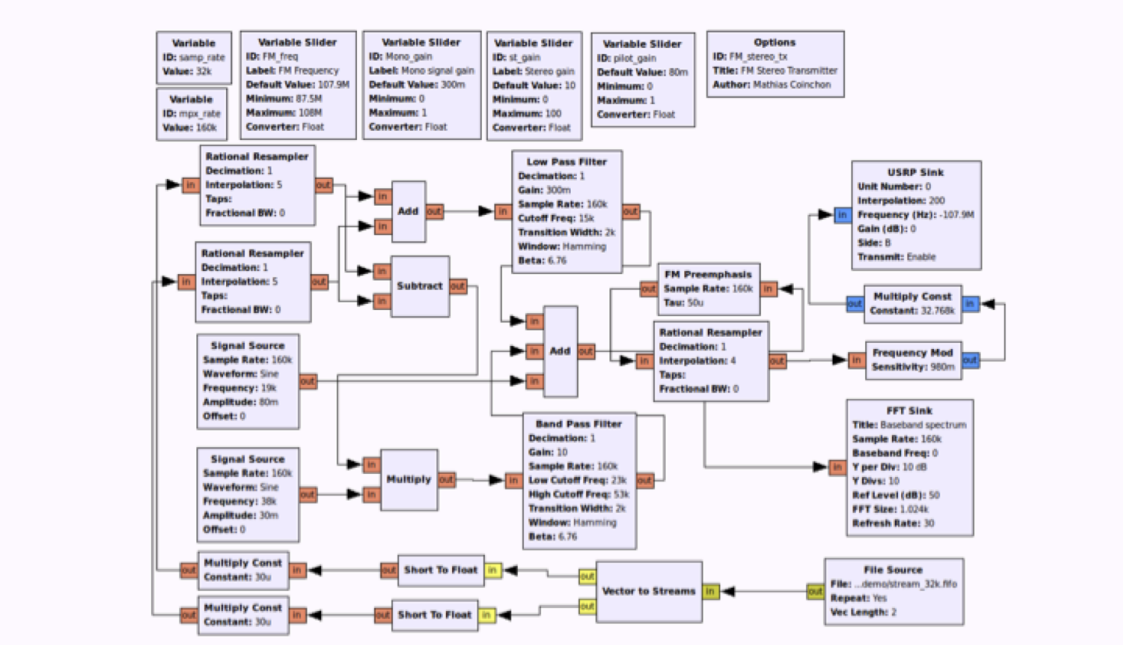

Figure 3.9: Gnuradio companion signal processing diagram for a simple FM transceiver[rad].

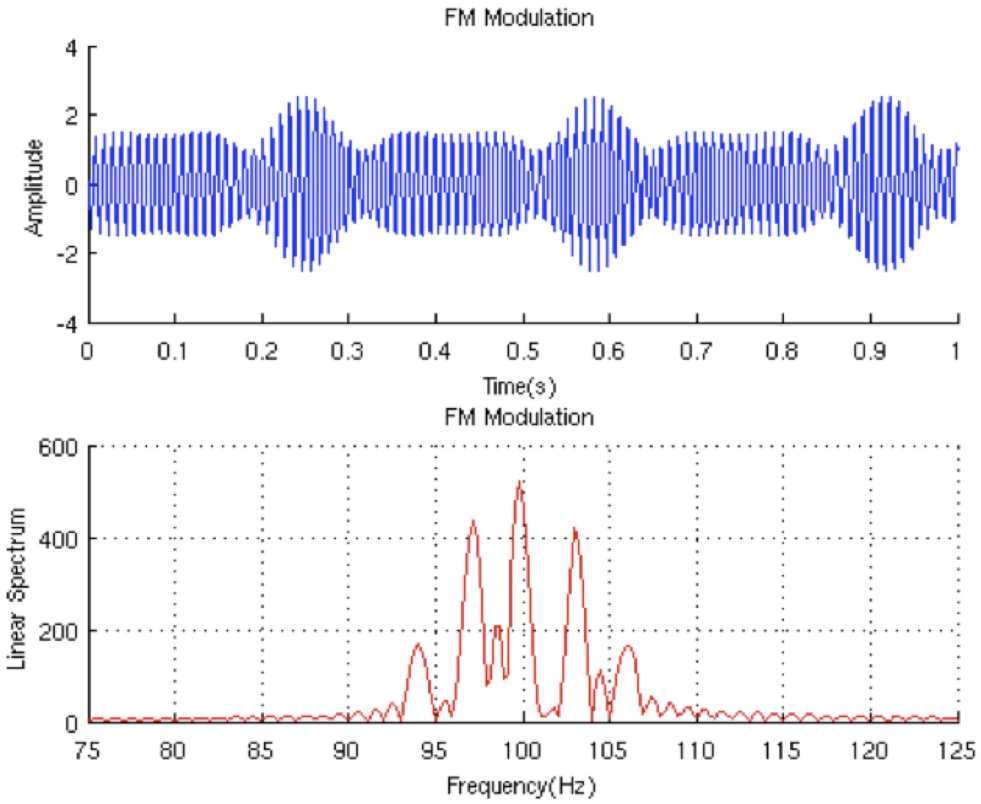

Figure 3.10: A frequency modulation implementation visualized on MATLAB. The data was exported from python.

(0-20kHz) and expect my board to receive it. So I simply shifted it to a higher frequency. I successfully transmitted the song at 100kHz - 500kHz, and 500kHz - 1Mhz range. The required bandwidth was of course 20kHz, or the audible frequencies. The third video is a quick demonstration of this output signal on an oscilloscope. It is very similar to the pictures shown in Figures 3.6 and 3.7.

The next step was to modulate the frequency in FM. The details of the modulation are described in the previous section. Given the severe limitations of power lines at higher frequencies, especially above 1MHz, I limited the transmitting bandwidths to between 100kHz and 1MHz. My transmitting daughterboard is also a low frequency daughterboard, which has a limit of about 3MHz for transmission. There is a reluctance to go very far above these frequencies as most household wiring is only designed to accommodate signals up to 400Hz[PBA10, ti]. That said, many retail PLC systems operate in the kilohertz to megahertz range, and detailed modeling of building wiring as transmission lines can be performed for exact specifications. In the NESL lab setting, however, as frequencies go higher, my experiments showed the losses over distance increase exponentially. Hundreds of kilohertz was the sweet spot for my experiments. I was able to get transmission throughout my lab and apartment (about 50 feet in wall length, but possibly much longer in wiring). I have read from other experiments that such propagation occurs at detectable levels in signals around 150kHz throughout large buildings. This is promising for any application of powerline communications.

After transmission and reception of an audio sample, there is a significant increase in white noise, but much of it can easily be filtered out. I have included the unfiltered, raw, recording so that the effects of the transmission are unaltered. This audio sample was converted to a data file in GNU Radio and was then frequency modulated and transmitted into an isolation transformer. The same board then received the signal (through a different path) and then sent the data into the USRP. I filtered out the frequencies besides 100-500kHz and demodulated the remaining signal in that range. These values were then stored to another data file. The attached "transmitted" .mp3 recording was recorded from that demodulated signal.

CHAPTER 4

The PowCapMobile Deployment Suite

From the breadth of prior study, it became clear that there was a lack of real life raw data available online for analysis. A deployed system, in a realistic environment with real loads and a natural usage pattern, would provide invaluable data for analysis and testing. Such data could be used on everything from cutting-edge NILM algorithms, on both high and low frequency data, to occupancy sensing/detection algorithms, to power usage models; the analysis possibilities are endless. To obtain this data, a new board would be required, one that is optimized for field deployment. This board is the PowCapMobile which, when combined with a Linux box and a USRP, is capable of taking four input signals, conditioning them, and a high fidelity digital recording. These systems, together, are referred to as the PowCapMobile Deployment Suite: a shockproof, robust, analog data-acquisition system. In this chapter we will analyze the components of this system, take an in-depth look a the data we acquired from a private homestead, and discuss the database in place at NESL that makes this invaluable data, for the first time, accessible to the research community.

4.1 The PowCapMobile Board

From revision 1 there were several areas that were targeted for improvement in revision 2. This board would need to be deployable and handle two phases of power, meaning it would have to handle two voltage and two current paths. The board would also need to be shrunk in order to be deployed at remote locations.

Another functional correction was the use of an IC level shifter between the MCU and the PGAs. The MCU ran off of 5V while the input signals on the PGA required the input

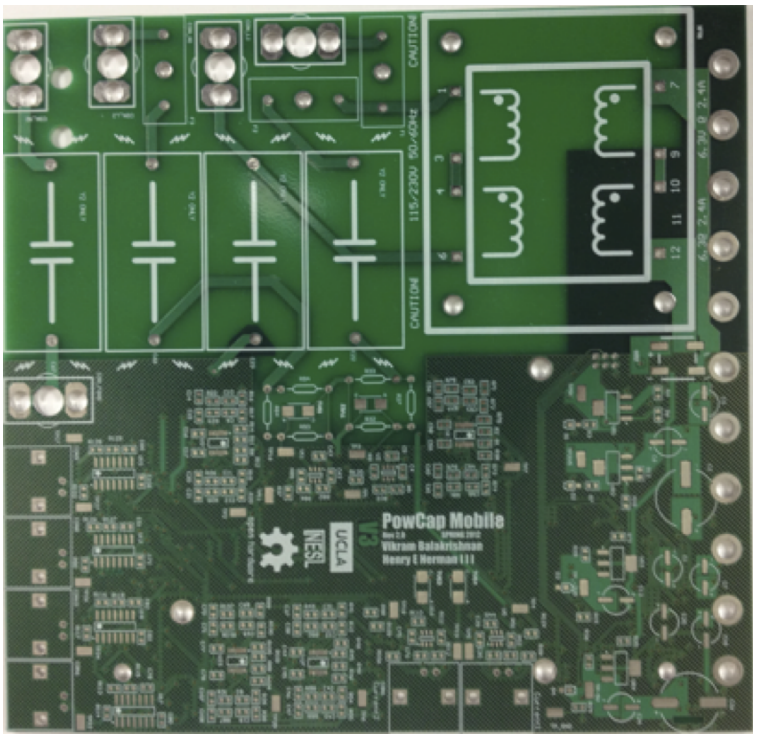

Figure 4.1: The PowCapMobile Board.

signals to be no larger than 2.5V. This error was not noticed immediately in revision 1 but eventually started creating faulty behavior in the amplifiers. Functionality was restored by manually shifting the signal, but in revision 2 the issue was fixed with a Maxim level-shifting IC.

The power rails generated from the linear power supply were just barely acceptable in revision 1. There was an unexpected voltage drop after the transformer and rectifier stage that left the supplied voltages to the regulators on the edge of their operational range. To improve the accuracy and reliability of our voltage supply, the transformer was upgraded slightly to a 25VAC output as opposed to a 20VAC output in revision 1. This gave us, via center-tap, DC rails of +/- 12.5V instead of +/-10V, which resulted in high quality outputs and more stable behavior from the voltage regulators.

The signal path of revision 2, called PowCapMobile for it's smaller layout, is shown in Figure 4.1. Much of it is similar to revision 1 so only major differences will be discussed in any depth.

4.1.1 Primary Signal Path

PowCapMobile is designed to take in four signals: two AC voltages and two AC currents. The main approach was to remove unnecessary circuitry coupled with a clever layout structure. Many of our diagnostic connections were also removed in the second revision in the name of space. These diagnostics included testpoints and manual switches, to control multiplexer functionality, which we no longer needed now that the firmware was fully operational. The main functional blocks are described below in more detail (see Figure 4.3):

Stage 1:

Much like revision 1, the voltage signals are capacitively coupled onto the board via Y2 safety caps and dropped via resistor dividers. The current signals are recorded with an external current transformer and connected as voltage signals to BNC inputs.

Stage 2:

From there the signals are routed through a MUX controlled 60-Hz notch filter. The number of notch filters was doubled from two to four to accommodate the new voltage and current lines.

Stage 3:

After the notch, the signals are routed through one of four MUX controlled band pass filter stage. The main difference in this signal path compared to revision one is that we removed six filter options in stage 3 with one bandpass option. So instead of a total of twelve filters for two signal paths in this stage, we now utilize four filters for four signal paths. Again, by using a dual op-amp Sallen-Key topology, this filter is easily modifiable to any desired frequency range. For our purposes, 100kHz to 250kHz was experimentally determined the ideal band to pickup household SMPS EMI.

Stage 4:

And finally a PGA stage before leaving the board via BNC connector. There are four LMP8100 amplifiers as a final stage before each BNC output. This serves to regulate the output voltage in order to maximize the dynamic range of the USRP ADCs.

To achieve the smaller footprint we desired, without sacrificing signal integrity, the ma-

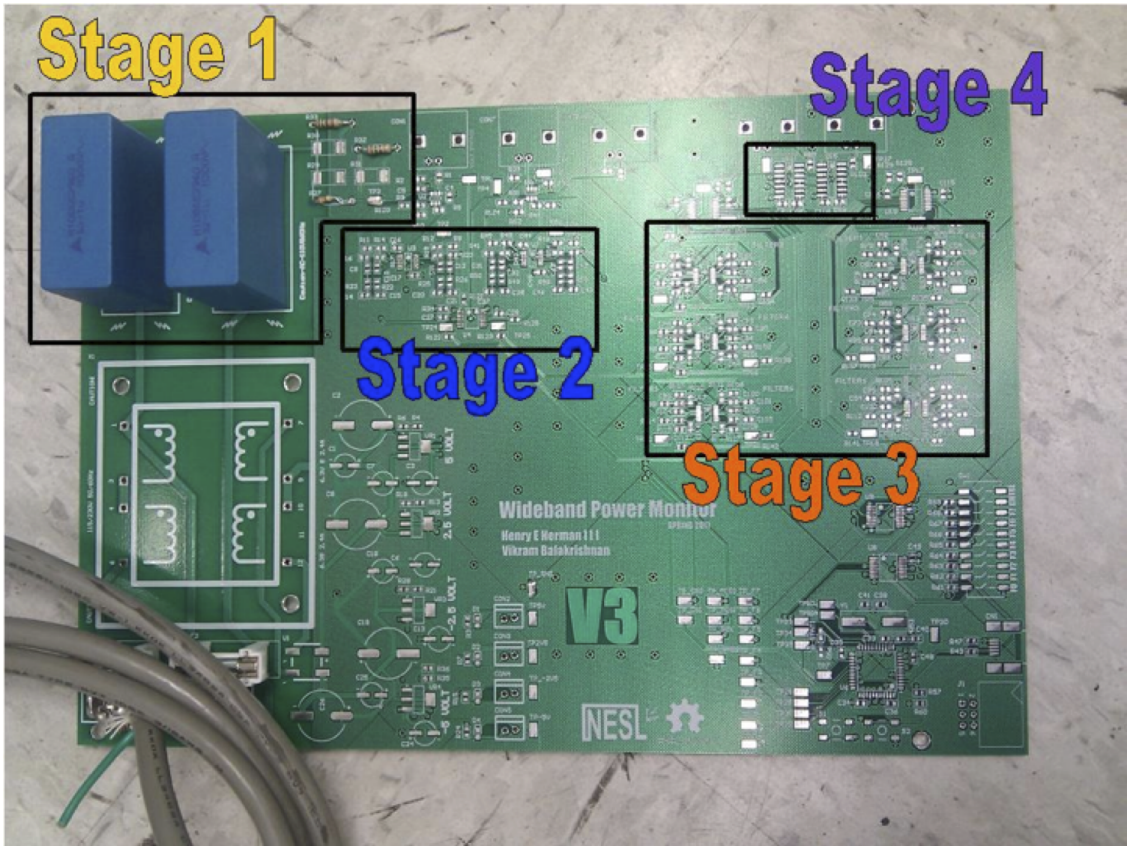

Figure 4.2: The main signal path of the PowCap board contains four stages: 1) Coupling 2) Notch Filtering 3) Band Filtering 4) Gain. Note how when compared with the new PowCapMobile revision in Figure 4.3, it is easy to follow the same signal path flow.

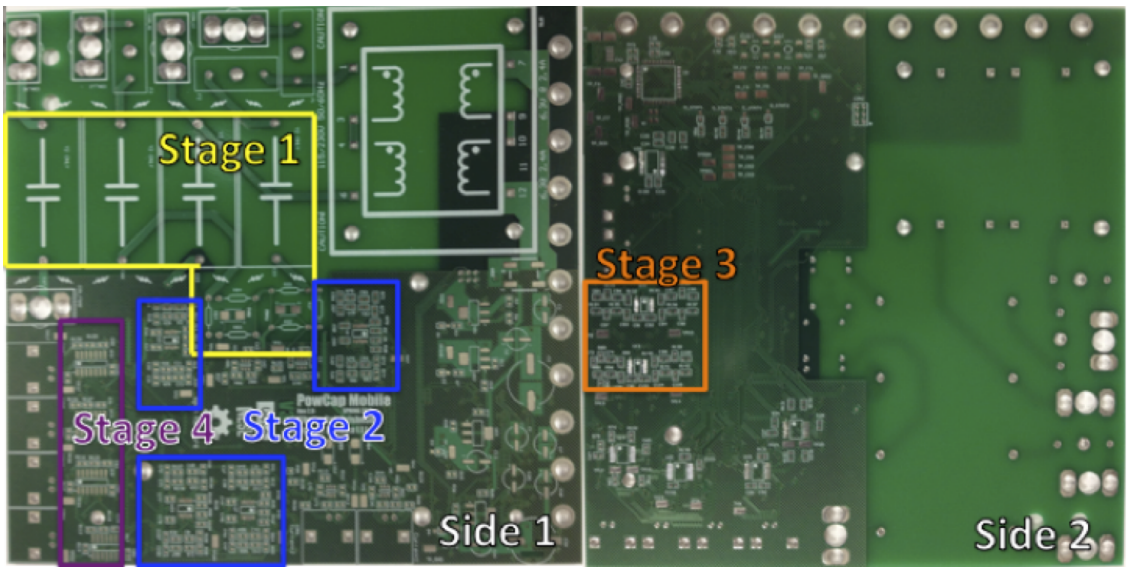

Figure 4.3: The signal path stages of the PowCapMobile Board.

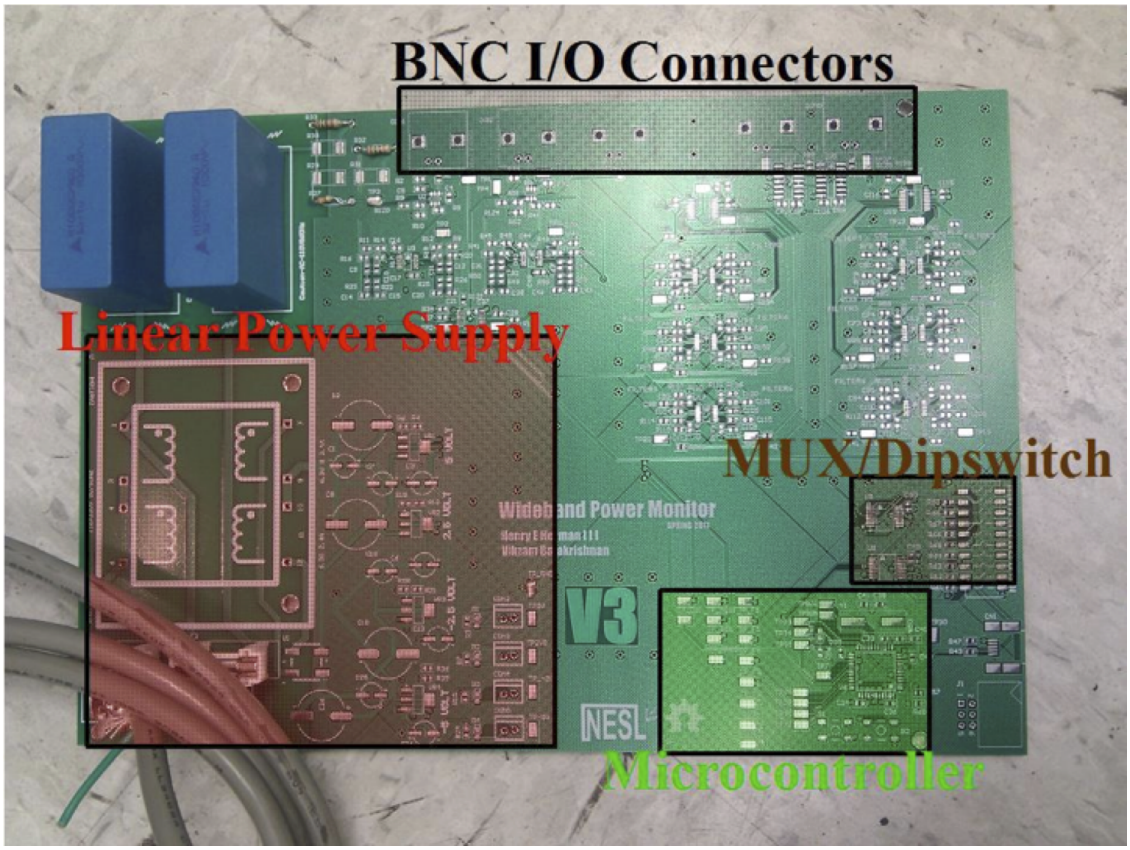

Figure 4.4: The supplementary blocks of the PowCap Board. The same system blocks can be seen in our PowCapMobile design in Figure 4.5.

jority of the digital components were moved to the back side of the board. This included the MCU, MUXs, level shifter chip, USB interface, and mcu programmer interface (see Figure 4.5). In particular, the level shifter utilized a Maxim MX3014.

Since the PowCapMobile was going to be deployed in environments where the system could potentially receive environmental interference, additional safety and protection features were implemented. Input fuses to limit current drawn by the board provided high level protection for components and material that might have shorted the board to ground. Additional ESD circuit protection was placed at key nodes throughout the board where high sensitivity ICs might be at risk. These protection circuits also, theoretically, shielded exposed nodes from occasional high voltage spikes that occur in in the home due to loading irregularities or environmental factors like lightning discharge.

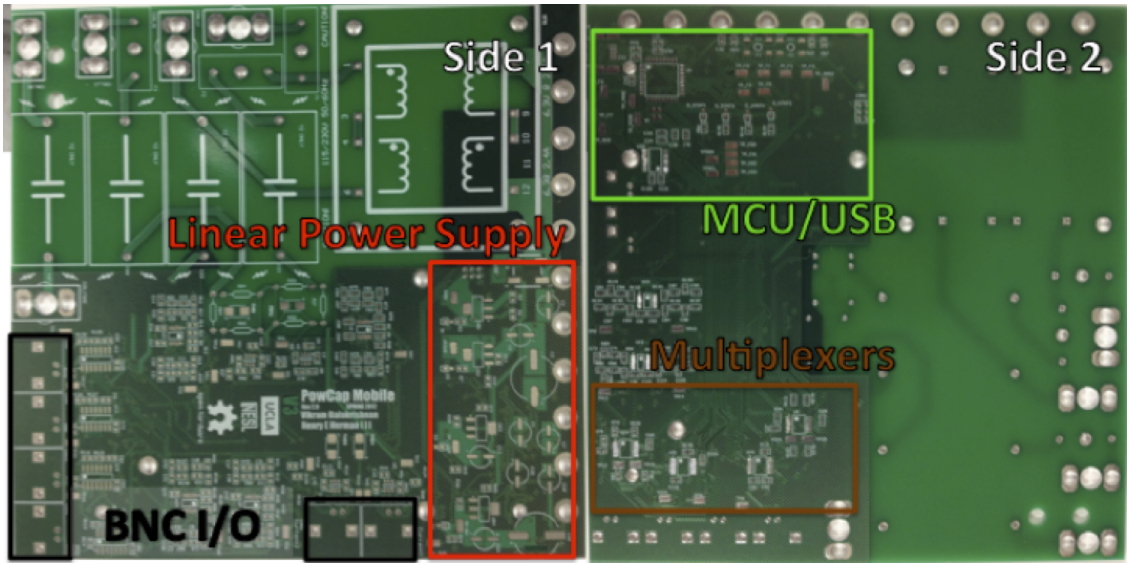

Figure 4.5: The supplementary blocks of the PowCapMobile Board.

4.2 The Universal Software Radio Peripheral (USRP)

From our custom board the conditioned signals are fed into a Universal Software Radio Peripheral (USRP) Model 1 (see Figure 4.6). This version features an Altera Cyclone FPGA, 64 MS/s dual ADCs, and USB 2.0 PC interface allowing up to 8MS/s data rate. The USRP1 can take two daughterboards, for which we use two LFRX board. LFRX daughterboards (see Figure 4.7) utilize high-speed op-amps for each input (each board has two) and couples them to the ADC. The ADC output is processed by the FPGA into two real-mode signals or a single I-Q pair. We use the latter format for all of our sampling through the PowCap deployments. These boards are optimized to handle "low frequency" input signals ranging from DC to 30MHz, more than suitable a range for SMPS EMI which usually hovers in the hundreds of kilohertz.

While the USRP system, along with the GNURADIO software package, is capable of doing a great deal of pre and post processing, we limit its functionality to data conversion. Once the raw data is recorded into a data file, it is accessible for analysis via any analysis tool. Our analysis is done primarily using the Python programming language and MATLAB.

Unpacking the data: Our I-Q data stream is stored as a complex32 binary stream. I have posted example code in Appendix B and C demonstrating how to easily convert this data

Figure 4.6: The native case of the USRP1 utilized in the PowCapMobile Deployment Suite[usr].

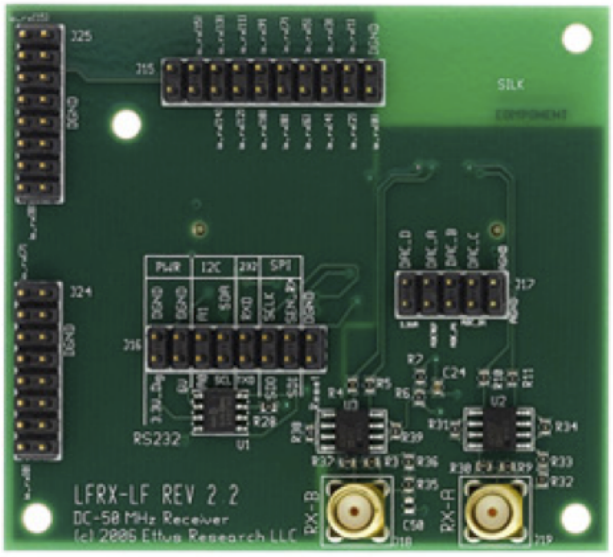

Figure 4.7: One of the two Low Frequency Receiver (LFRX) daughterboards[lfr] used in the USRP1.

format to a data structure that is readily recognizable within MATLAB and Python.

4.3 The PowCapMobile Deployment Suite

For a complete deployment, the key components of data acquisition via PowCapMobile board were integrated into the PowCapMobile Deployment Suite. The three main components are: Linux PC running gnuradio, PowCapMobile Board, and a USRP1 with two LFRX daughterboards (see Figures 4.8 and 4.9).

4.3.1 Case

For security and stability, all three devices are securely mounted into a shock resistant container. The PowCapMobile board, in particular, was mounted with insulating screws and was isolated from the metal drawer in which it was attached. This was important to prevent the high voltage mains being shorted to the metal ground of the box, potentially endangering users and damaging other electrical components.

4.3.2 External Connections

The Deployment Suite interfaces with each voltage phase through standard wall plugs. Standard NEMA 5 insulated plugs can be extracted several feet from the case to be plugged in nearby.

Current is connected through the use of two 60A 0-3V output current transformers. Each current transformer is hooked up to a screw mount adapters on top of external BNC interfaces. This particular connection provides a flexible input interface for a wide variety of analog signals should the box be needed to record data from a different source.

4.3.3 System interfaces

The PC, USRP, and PowCapMobile all interface through USB 2.0 connections. The board hardware is directed via a LUFA virtual serial port. The BNC outputs from the PowCap-

Figure 4.8: Front panel view of the PowCapMobile deployment suite. Note: The tray with the PC is on the right and the tray with the USRP1 and PowCapMobile Board is on the left.

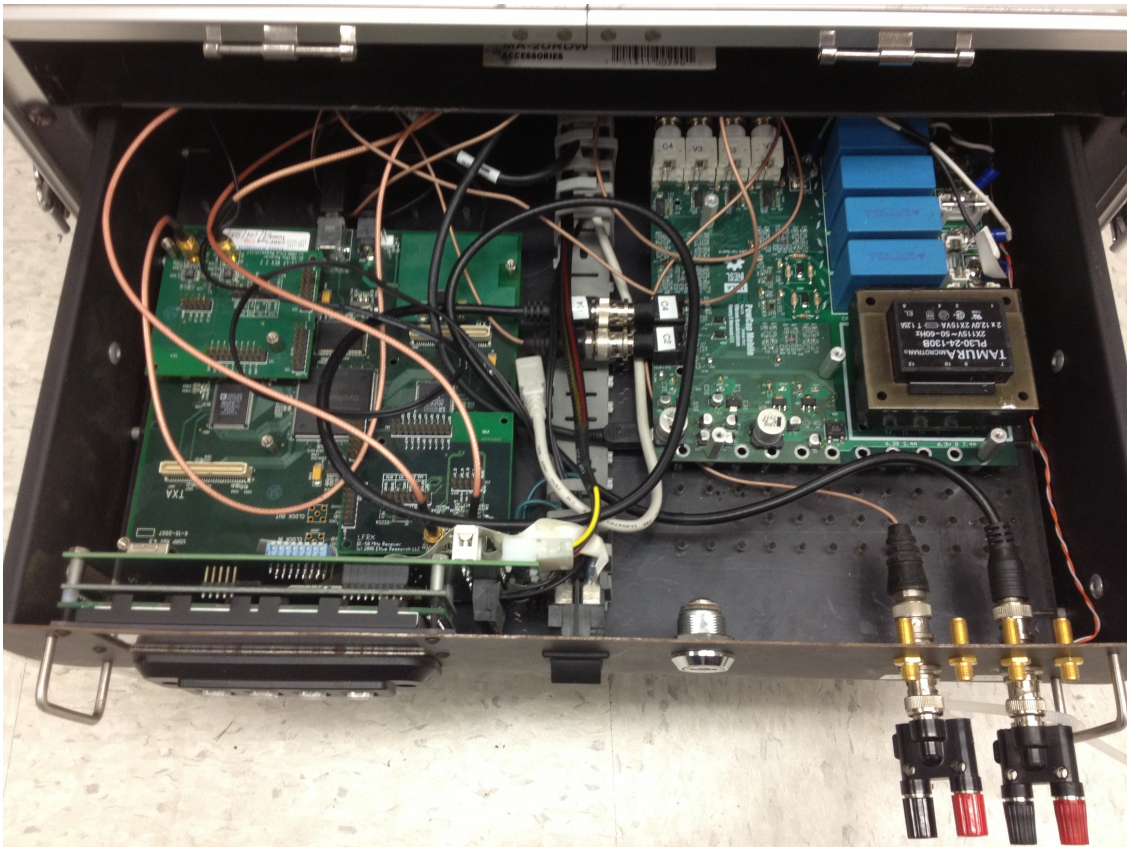

Figure 4.9: View of USRP1 (left) and PowCapMobile (right) mounted inside of our PowCapMobile Deployment suite.

Mobile are directly attached to SMA inputs on the USRP daughterboards. The USRP is attached to the Linux PC with USB and is controlled directed by system drivers included in gnuradio.

4.3.4 Auto-connectivity

The Linux OS, on power up, automatically connects into the NESL lab via a persistent VPN call. This allows for true remote data collection and development, as anyone who has VPN access into the lab can then access the Linux box and perform live experiments or setup long-term data collection.

4.3.5 Data Collection Protocols

At full sample rates, even short time samples can result in huge file sizes. To deal with this deluge of data, I came with some basic protocols for data recording. Files can be segmented into smaller pieces, the default being 1 hour segments. One hour of data at a 1MHz sample rate results in a file size of 25GB per phase (one phase includes voltage and current lines).

By segmenting our data files, we minimize the chance of file corruption. Additionally, it isolates our entire data set from unexpected errors in the data collection script. While at lower sampling rates it might be possible to stream data live across a local internet connection, at these full sampling rates such "wireless" transfers were unfeasible. Instead two 5.1TB external hard disks were attached to the PC for data storage. As they filled up, they were physically brought to the lab and uploaded to dedicated NAS server. Through the use of a custom script, hardisks could be added or removed during the course of data collection and new files would be automatically stored in any new recognized drives. This theoretically allows for data collection over long spans of time and is limited only by laboratory server space.

4.4 Experimental Setup (Deployment)

For our experimental setup (see Figures 4.10 and 4.11), we were generously given access to a large sized family home in the Greater Los Angeles area. This house had access to outlets for each phase located next to the main breaker panel. One current transformer was attached to each current phase entering the house while a voltage connection was made with a socket associated with each phase.

To provide a rich recording for deep spectral analysis, a full sample rate of 1MHz was used. Over two phases, this resulted in about 10 days of raw data; each phase is stored in one-hour segments. Using custom scripts, the data is easily traversable with the simple user input of start and end times. The files have all been uploaded to a dedicated PowCap NAS server (see Figures 4.12, 4.13, and 4.14). From here the data is accessible to any group who wishes to analyze the experimental data, and efforts are under way to make the data accessible to those outside the UCLA community as well. This would be either by uploading the raw data to a 3rd party "big data" server and/or by making the NAS server accessible via some external protocol.

Using a simple Python script, the data can be plotted as shown in Figure 4.15. The labels "V1" and "V3" correspond to labeled I/O connectors on the deployment suite. This particular window shows just a 200 millisecond snapshot of the voltage and current lines on the first day of data acquisition. This window length is useful for check features of the voltage and current signals visually, and was mainly used debug faulty or noisy connections. Analysis of such rich datasets provides memory challenges, especially when looking at the data in full fidelity.

"Ground truth" was provided by existing power monitoring sensors (see Figures 4.16 and 4.17). Voltage, current, and power levels are provided for each phase and a subpanel containing an assortment of loads in the house. The first phase of experimentation was simply validation of the recorded data; that is, ensuring measured values matched up with these ground-truth calculations. The next phase involves taking detailed information about individual loads in the home to compare their transition events with various features extracted

Figure 4.10: PowCapMobile deployed in a private residence.

Figure 4.11: Current transformers attached to the mains. Current readings are relayed as voltage values over BNC to the PowCapMobile board.

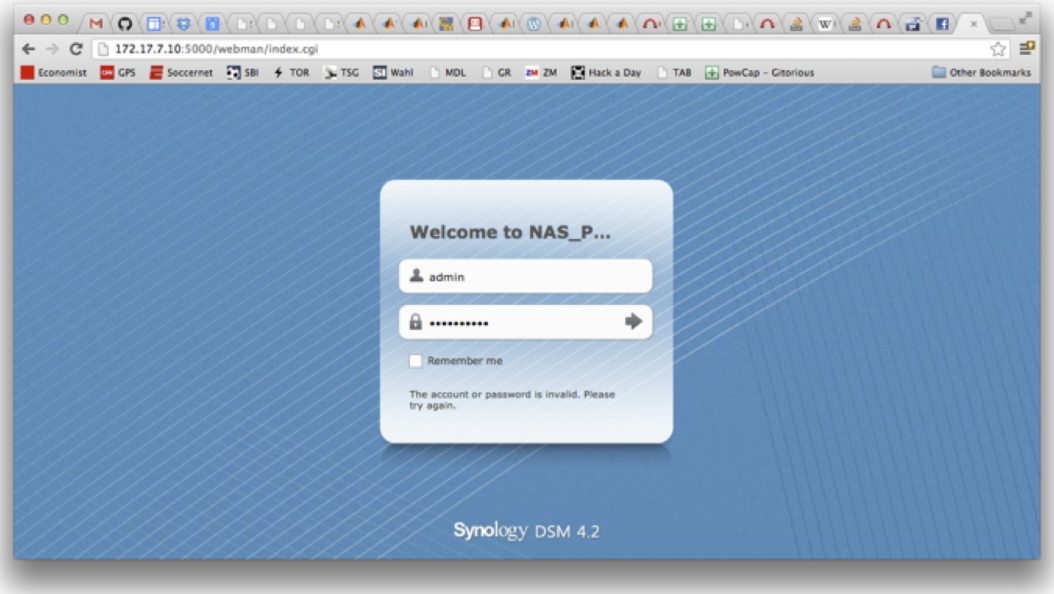

Figure 4.12: Custom NAS server to house data from residential home.

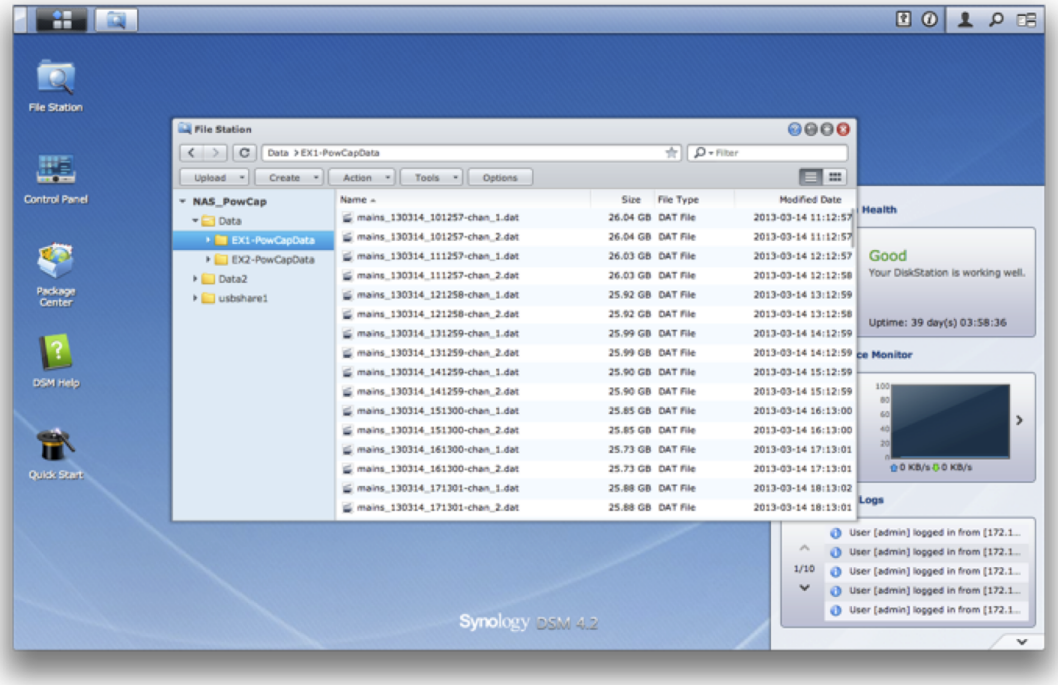

Figure 4.13: Data storage on dedicated NESL NAS server.

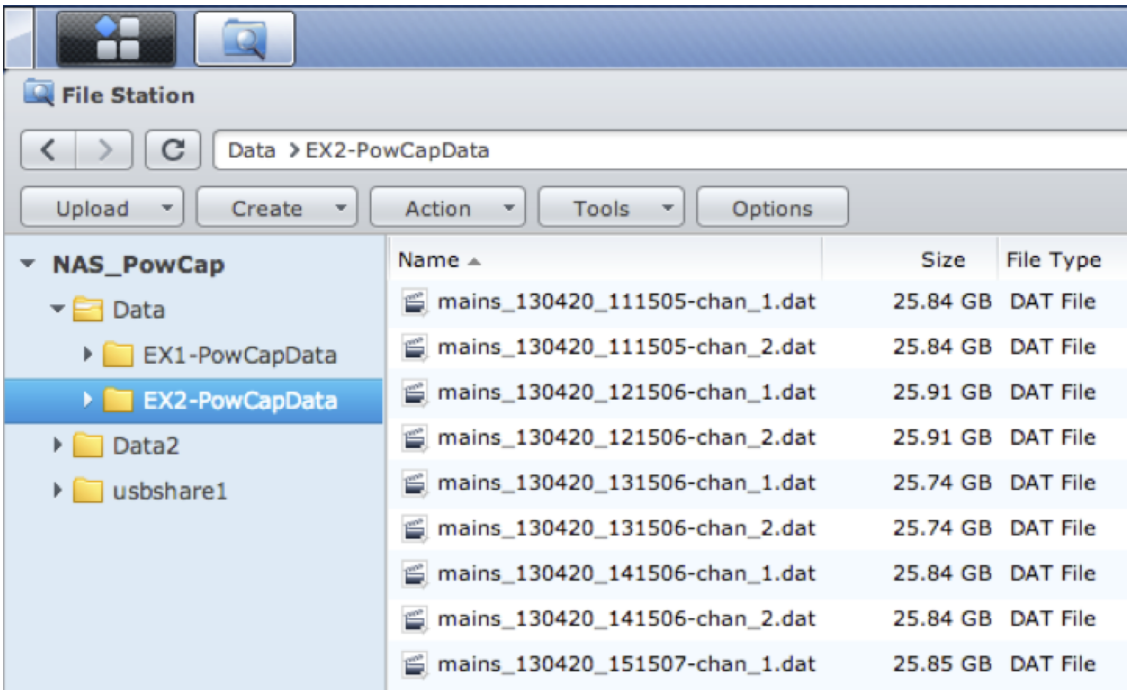

Figure 4.14: File storage format on dedicated NAS server. Note, individual phase data has been captured and stored in time-stamped and date-stamped one-hour segments.

from the raw sample sets.

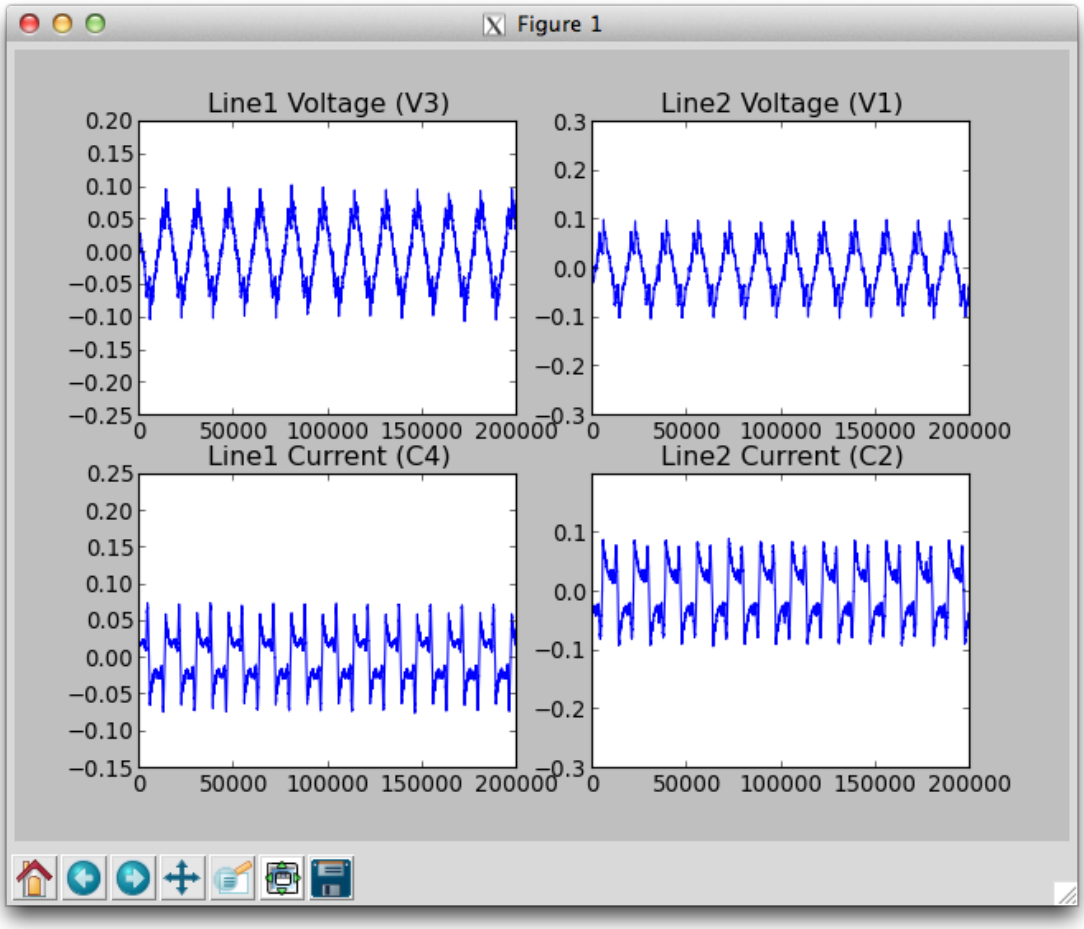

Figure 4.15: Here we see 200 millisecond windows of data as displayed by our custom python analysis software.

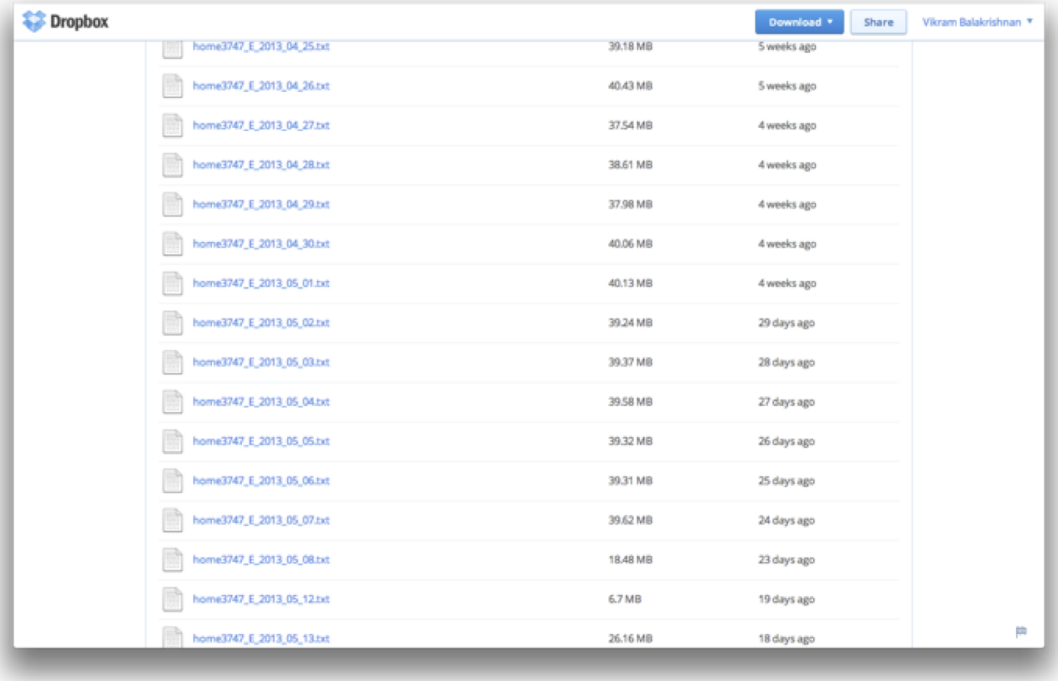

Figure 4.16: "Ground truth:" current, voltage, and power data provided by sensors already installed in the home. Data is stored in daily files within the Dropbox folder.

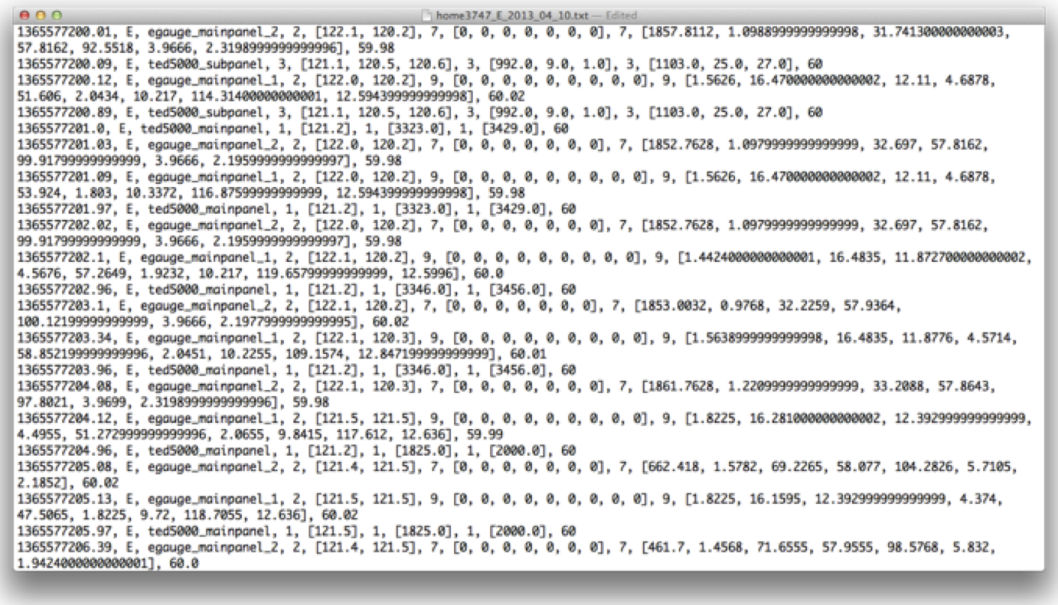

Figure 4.17: "Ground truth:" current, voltage, and power data provided by sensors already installed in the home. Here we can see how the data is stored in text files. This data is extracted and establishes a basis by which we compare our experimental measurements.

CHAPTER 5

PowCap: Experiment and Validation

After the hardware development resulted in a working PowerCapture board, collaboration with groups specializing in signal analysis was an obvious application. Through these collaborations, initial verification and validation of our data procurement methodology was accomplished. The results of analyzing this data led to a load detection framework and a formal proposition for a new, low overhead, power-sensing system. The experiments, analysis, system proposition, and conclusions are presented in this chapter. The work presented in this chapter was done in close collaboration with two students from Professor Lei He's Design Automation Lab (EDA) at UCLA: Ph.D candidate Wei Wu and Ph.D candidate Wenyao Xu.

5.1 Experiment Setup

In our experiments, we implemented a prototypical system methodology. Approaching the NILM problem from a fresh perspective meant capturing as much data as possible within the entire noise spectrum. As a result, the prototypical hardware actually used was designed to be on the higher end as shown in Figure 5.1. However, the quality of load identification possible with solely low frequency analysis shifted the focus to the potential of a low cost system; we will explore this idea in detail later in this chapter.

During data collection, we use *seven* different appliances, including a) LED lamp, b) incandescent lamp, c) Macbook Pro, d) Fan[1] (Speed 2), e) Fan (Speed 1), f) Heater and g) Microwave oven, as a case study to verify the proposed system. Figure 5.2 is a picture of all

[1]The fan has two speed options, and Speed 2 is stronger than Speed 1

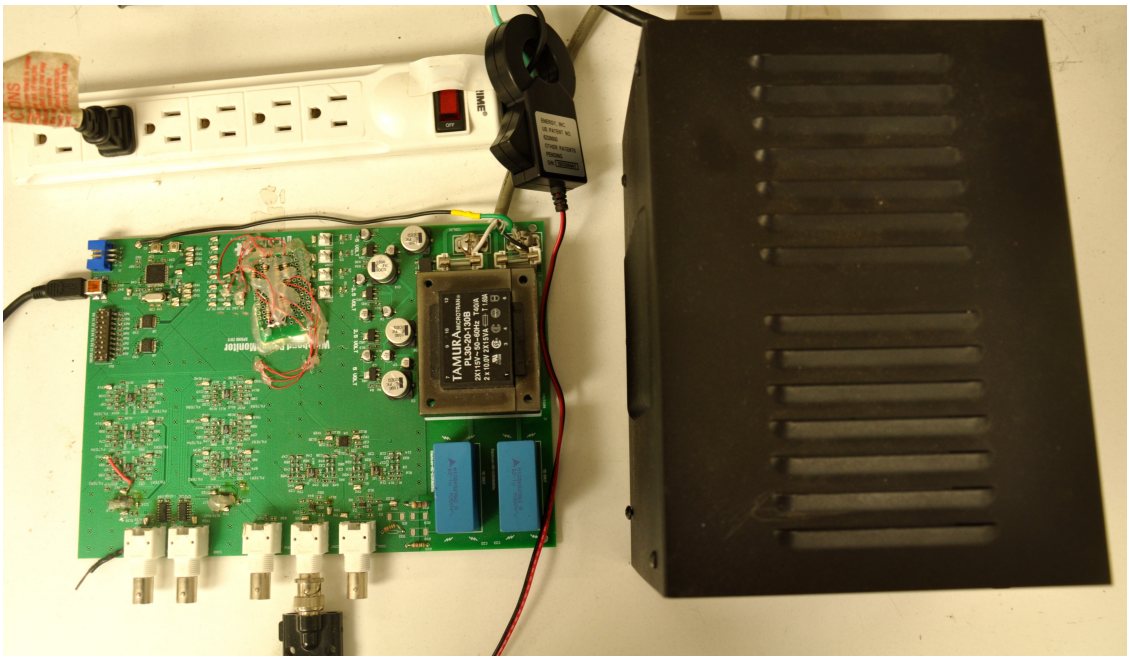

Figure 5.1: High end, prototypical hardware: an isolation transformer for safety, an insulation-stripped power strip for current sensing of one or more devices, and the PowCap Board for signal conditioning.

the appliances evaluated in our experiments. The nominal power is each appliance is listed in Table 5.1.

To evaluate the algorithm on our prototype, we first perform the feature training by turning on the appliances consecutively and generated the feature library for future appliance matching. Next, the current and voltage data are collected for each appliance at 500Hz and the proposed nearest-neighbor algorithm is applied to identify the appliance.

5.2 Algorithm

5.2.1 Algorithm Overview

The framework of the proposed algorithm for appliance feature extraction and recognition is illustrated in Figure 5.3. It consists of two stages, the training stage and the recognition stage.

Figure 5.2: Appliances used in the experiments

Table 5.1: The Description of Appliances in the Experiments

Appliance ID	Name	Power (W)
a	LED Lamp	15W
b	Incandescent Lamp	25W
c	Macbook	200W
d	Fan (Speed 2)	50W
e	Fan (Speed 1)	30W
f	Small Heater	180W
g	Microwave Oven	1000W

Training Stage

In the training stage, the algorithm learns the features from the training sequence. Here we define the training sequence as a sequence of sampled raw data with ground truth (the appliance label). Figure 5.4 shows an example of training data with 7 different appliances.

Features are extracted from each window cell to form a local feature vector. All the local feature vectors from the same appliance training data are then pooled together and quantified through an unsupervised clustering algorithm to construct the golden feature vector. The center of each generated cluster is treated as a unique feature primitive for appliances. All of the feature sets together form the feature library for recognition.

Recognition Stage

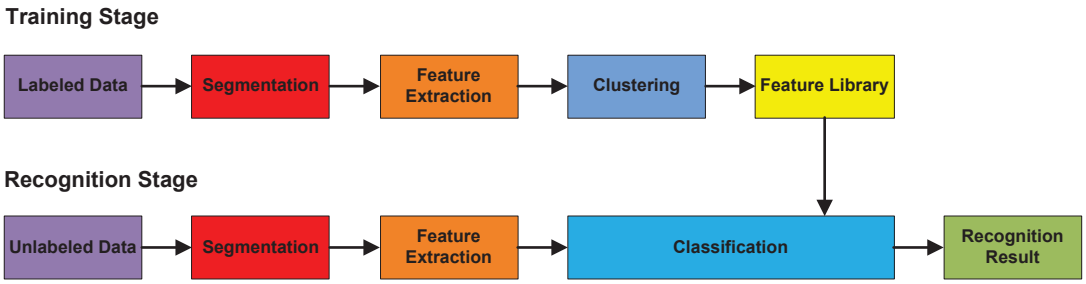

Figure 5.3: Algorithm flow

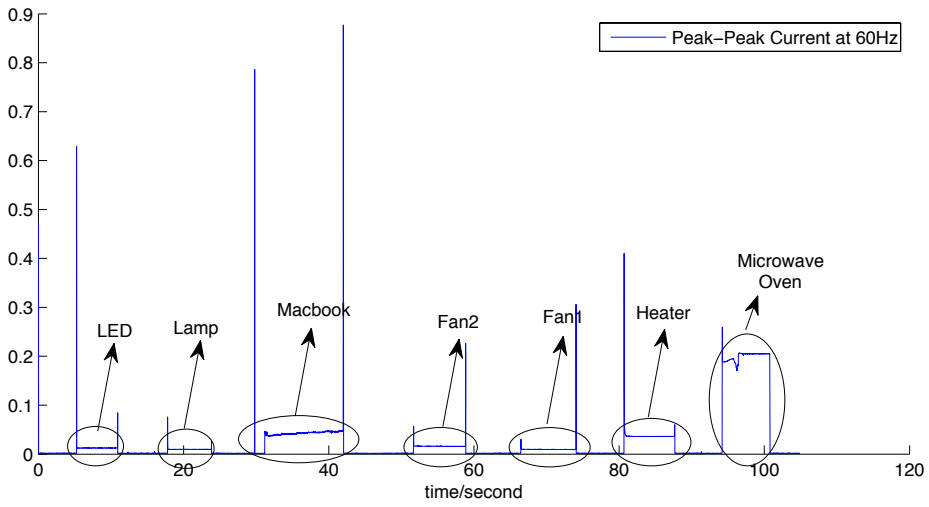

Figure 5.4: Training data with 7 different appliances

In the recognition stage, we segment the test data and extract the feature to represent the input. Then we classify the unknown test data to the appliance class that has the most similar features. In the remainder of this section, we present the details of the key components of this framework.

5.2.2 Appliance Features

Various features, such as voltage, current, power, power factor, spectrum and noise pattern, have been adopted to distinguish different appliances [PRK07, GRP10, KBN10, GOB11, JSB12]. As far as how the feature is extracted, we can divide the features into two groups: features that require high sample rates (high frequency features) and low sample rate features (low frequency features).

In the spectrum of the current/voltage signals, the noise pattern is a typical high sample rate feature. These high frequency features are usually known as electromagnetic interference (EMI) of appliances and can be utilized to identify the appliance with a proper data acquisition and processing system. However, the expense and effort required to extract these EMI features is a common drawback of these systems. Normally, a high sample rate (up to 1MHz) is usually required to capture the EMI signature, which results in high cost and also high computational complexity of the appliance identification system. Moreover, most of the appliances in a building are parallel connected to the powerline. In other words, their EMI signatures are mixed together. Therefore, even if we can measure and identify the appliances from the EMI signature, there is no ground to know which appliance is plugged in which power strip.

Besides EMI, other features, such as the Peak-Peak current, real power, reactive power and power factor, are also valuable information for alternating current (AC) power systems. In our work, we define those features as shown below:

- ***Peak-peak current*** (I_{P-P}): the difference between the maximum and minimum current in one cycle of 60Hz, $I_{P-P} = \max(i) - \min(i)$;

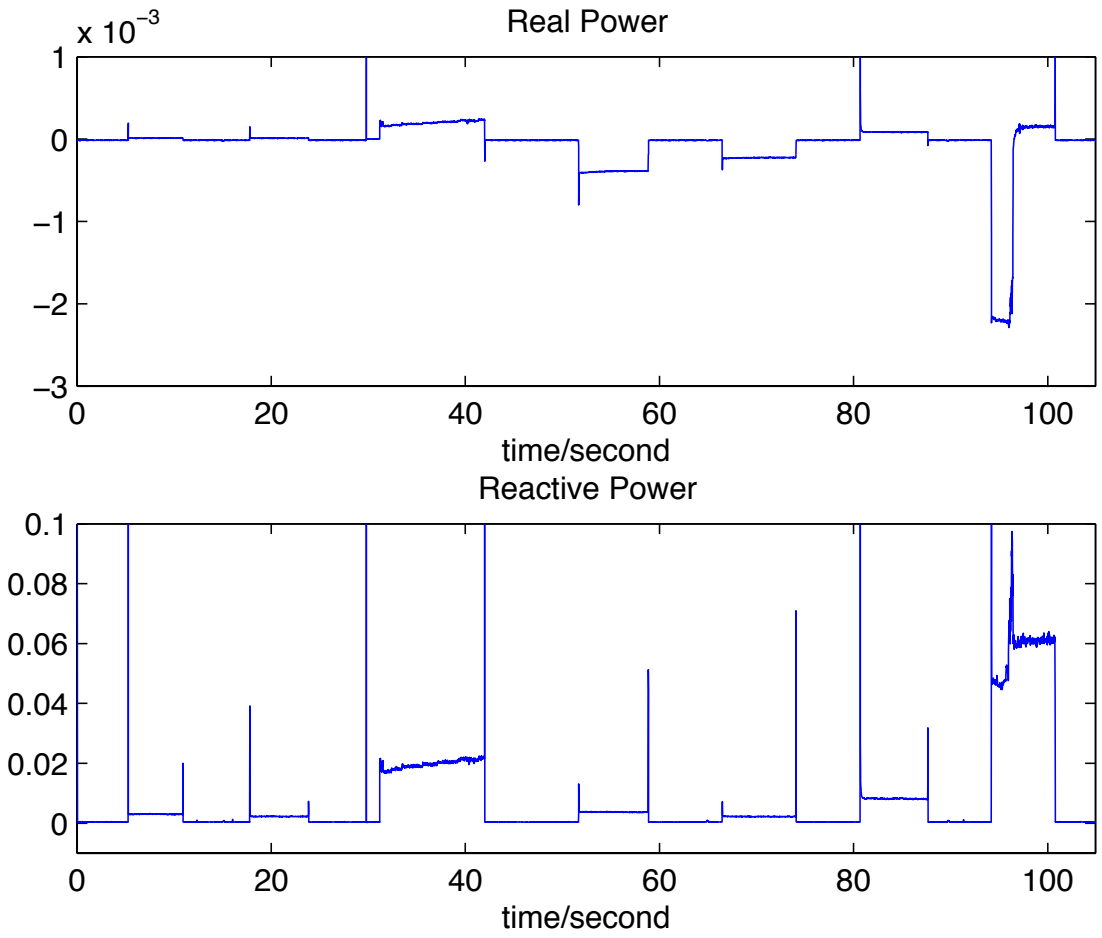

Figure 5.5: Real power and reactive power

- ***Real (Active) Power*** (P): The power calculated by integrating the product of the current and the voltage. $P_{real} = \frac{1}{\tau}\int_0^\tau i(t)v(t)dt$;
- ***Apparent Power*** $(|S|)$: The apparent power is calculated by the rms voltage and rms current. In alternating current systems, $|S| = \frac{1}{2}I_{P-P}V_{P-P}$
- ***Reactive Power*** (Q): As the real/reactive and the apparent power satisfies: $|S|^2 = P^2 + Q^2$, then once we get the P and $|S|$, it is easy to calculate the Q.
- ***Power Factor***: The power factor is the ratio of active power to the total power.

Figure 5.5 shows the real power, reactive power and the peak-peak current of a few appliances. It is not difficult to visually observe the difference in these features from appliance to appliance.

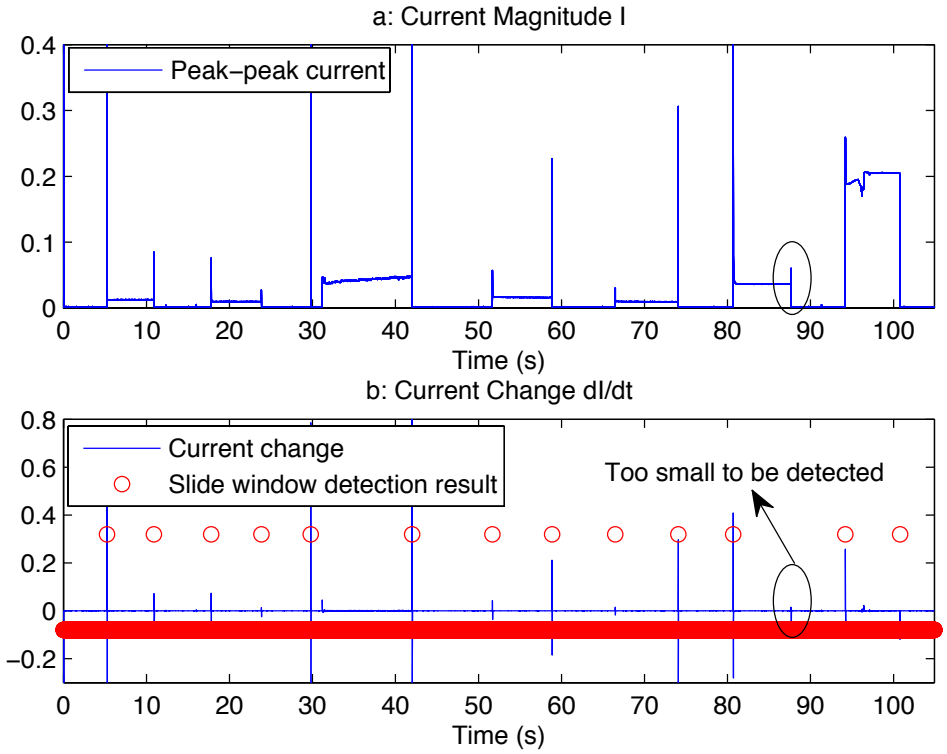

Figure 5.6: Peak-peak current and the slide window identification on current transition

In our algorithm we chose the real power, the reactive power and the peak-peak current as our 3 key features. We did not use the power factor and the apparent power for 2 reasons: 1) they are correlated with the real power and the reactive power; 2) the identification accuracy using the aformentioned 3 features was sufficient to identify our chosen appliances.

5.2.3 Segmentation

In order to identify an appliance from the raw data, we have to first know when the appliance is running and when the information is only background noise. As a result, the data segmentation is of great importance to the appliance identification. We tried two different approaches of data segmentation: first applying slide window detection to detect the change of current, and second using a firm threshold to detect magnitude of the current. Below are the detailed discussions of these two methods. For our algorithm, we chose the current threshold method.

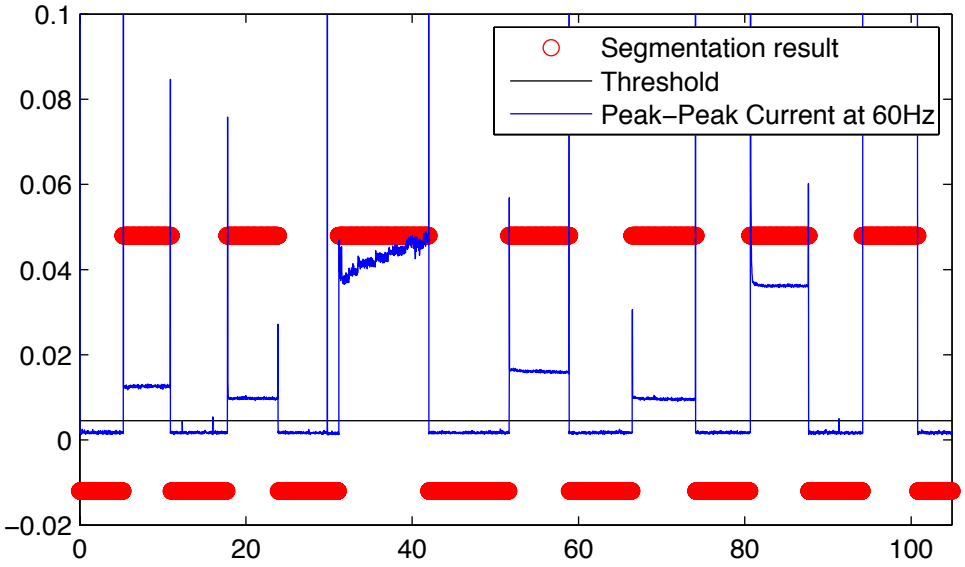

Figure 5.7: Extract the centroid from the distribution of three features

Transition Detection using Slide Window

Each turn on and off behavior of the appliance is combined with changes in current or power. The idea of segmentation is to detect the transitions in the current sequence. An example of current transition is illustrated in Figure 5.6. Figure 5.6a is the original current data, while Figure 5.6b depicts the change of current (blue curve) and the identification result (red circles).

In this example, most of the current transitions are identified. However, it is quite unstable and it may miss some small transitions as illustrated in Figure 5.6b.

Current Threshold

Another straightforward approach to identify the segment of data is to use a hard current threshold, which is the adopted method in our algorithm. As only one appliance can be plugged into one outlet at a time, the current drain from each outlet can be either 0 or the current of the appliance. As there is no background load in our application, we can apply the current threshold approach. Figure 5.7 illustrates the detection results of the threshold approach.

Generally, the slide window approach evaluates the change of current which, as a transient

behavior, is easily contaminated by transient noise on the power line. Particularly, for appliances with a large protective capacitor, it is difficult to identify the current transition because the transition might be smoothed by the capacitor. If we try to cancel the noise by calculating the average current in a short time window, the averaging itself serves to smooth the features of the current transition. However, the current threshold approach evaluates the average of the current in each time window ($1/60s$), which only filters the noise and nominally maintains an accurate value for the current, leading to better performance in this application.

5.2.4 Feature Extraction and Feature Library

Three features are first extracted from the raw data in each time window. These three features, real power, reactive power, and peak-peak current, are not usually constant but appear as a distribution due to the measurement noise and the other interferences. The extracted features of a LED (the 1st appliance in Figure 5.4) over a few time windows are illustrated in Figure 5.8.

To build the feature library, we do not store the extracted feature in all time windows. As illustrated in 5.8, we only calculate the centroid of these features for each appliance and store them in a 3-dimensional space. After a training stage with each of the appliances, we generate a lookup table containing the feature centroid, $Feature = (P, Q, I_{P-P})$, for future appliance identification.

5.2.5 Low-complexity Appliance Matching

Once we have the feature centroid of the appliance from the training stage, we then apply a Nearest Neighbor [AMN98] algorithm to match the appliance in question to the closest appliance feature in the lookup table. In other words, the algorithm selects an appliance by minimizing the distance between the measured feature centroid, $F_{Measured}$ and the feature centroid in the library.

The size of the feature lookup table in Nearest Neighbor is linear related to the number

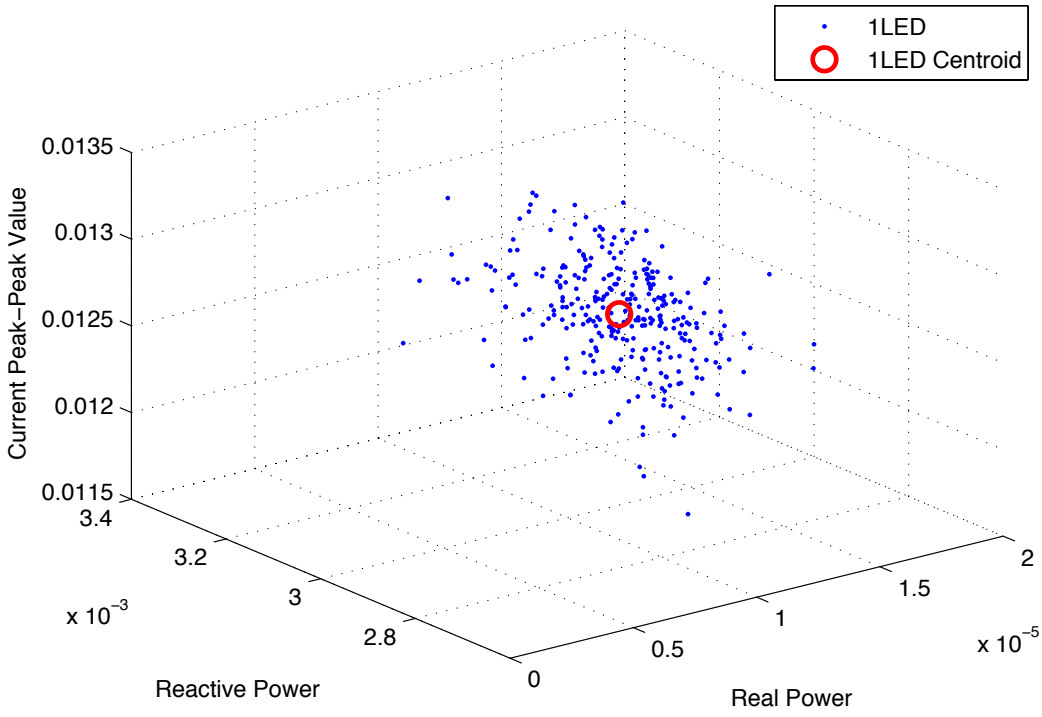

Figure 5.8: Extract the centroid from the distribution of three features

of appliance types and features. In this paper, we extracted *three* different features, and the space complexity will be only:

$$S(n) = 3n \tag{5.1}$$

where n is the number of appliance types. In Nearest Neighbor, the matching time complexity is also proportional to the appliance type number [AMN98]. In this way, we can see that the Nearest Neighbor based matching algorithm has linear complexity in both space and time. Because we target a light weight system at the outlet level, the nearest neighbor approach is more suitable for our application.

Nearest Neighbor is not the only solution for matching applications. For example, Nearest Subspace (NS) [Tsu99] is also a well-known approach for feature matching. In this paper, we used Nearest Neighbor instead of Nearest Subspace because of the concerns on time and space complexity.

5.3 Feature Space Distribution

To investigate the unique characteristics of these appliances, we put them in the feature space and plotted the distribution. After a few improvements on the algorithm, most of the appliances (with the exception of the Macbook) can be identified by the proposed system at a precision higher than 97%.

5.3.1 Transition Noise:

As mentioned in [GOB11], the identification of appliances with small power consumption is a common problem for the appliance identification system. During the first trial, the proposed algorithm was not functioning well when distinguishing the LED lamp (LED) from the incandescent lamp (Lamp) and the Fan operating at speed 1. To figure out the reason, we checked their feature distribution in both the training session and the identification session and found that the centroid was not correctly representing the distribution of features. The centroid deviated due to the transition noise experienced during the ON/OFF state of the appliance. As illustrated in Figure 5.9, the features of the LED lamp are wildly distributed in the feature space. The dots far away from the centroid correspond to the spiking transient observed during the ON/OFF transition in the LED (seen in Figure 5.4).

In Figure 5.9, those dots, contaminated by transition noise, noticeably shift the position of the centroid. To alleviate the effect of transition noise, we screen the signal around the transition period during both the training and identification stages. The centroid calculated after removing the transient noise is shown in Figure 5.8. Finally, after the training process, we get the feature centroid of all *seven* appliances in Figure 5.10.

5.3.2 Identification

The Nearest Neighbor identification of the appliance is achieved by finding the closest centroid (in Figure 5.10) to the measured feature centroid. In figure 5.10, it is still difficult to distinguish the LED from the lamp as their centroids are very close and the distribution of

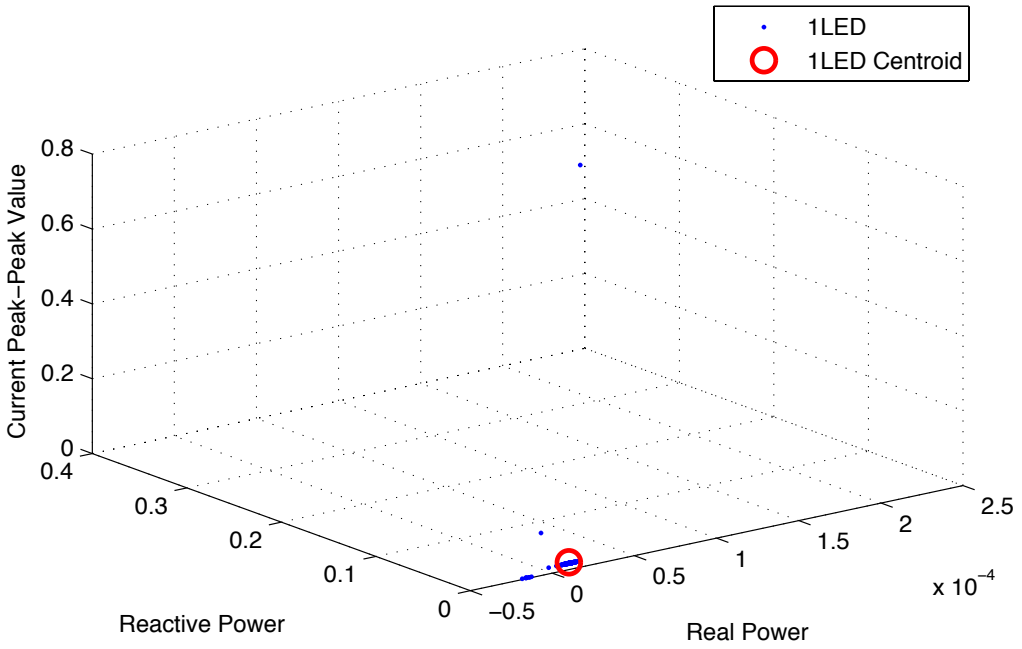

Figure 5.9: LED features with transition noises

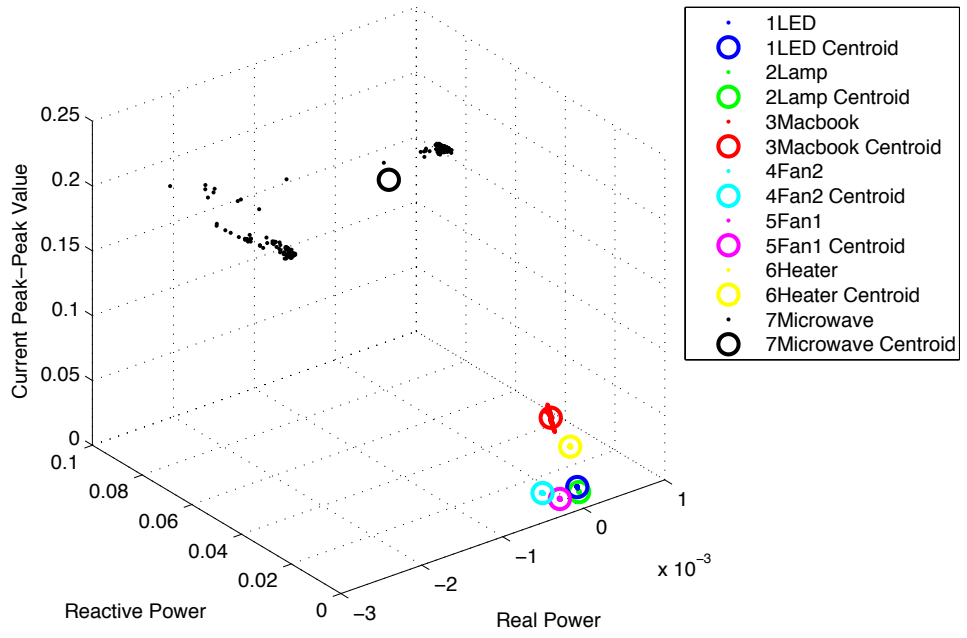

Figure 5.10: Features distributions and centroids for all 7 appliances

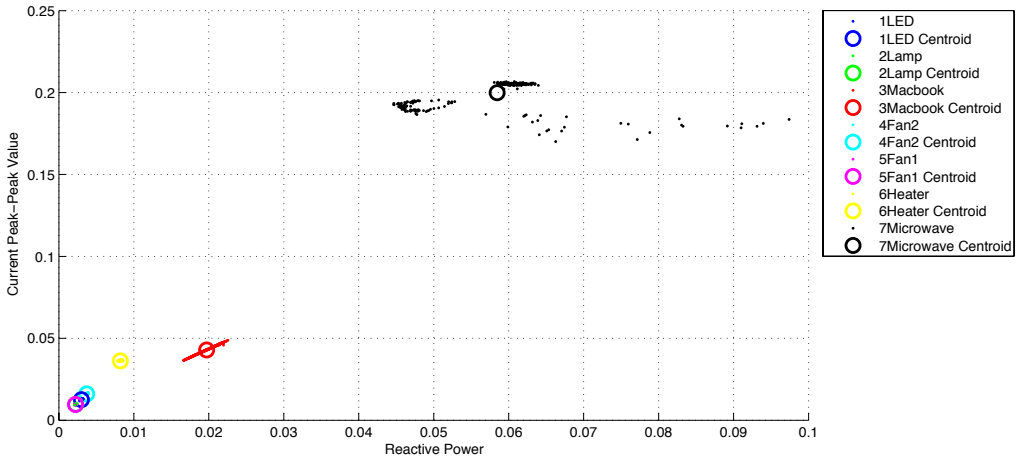

Figure 5.11: $P - I_{P-P}$ view of Features distributions and centroids

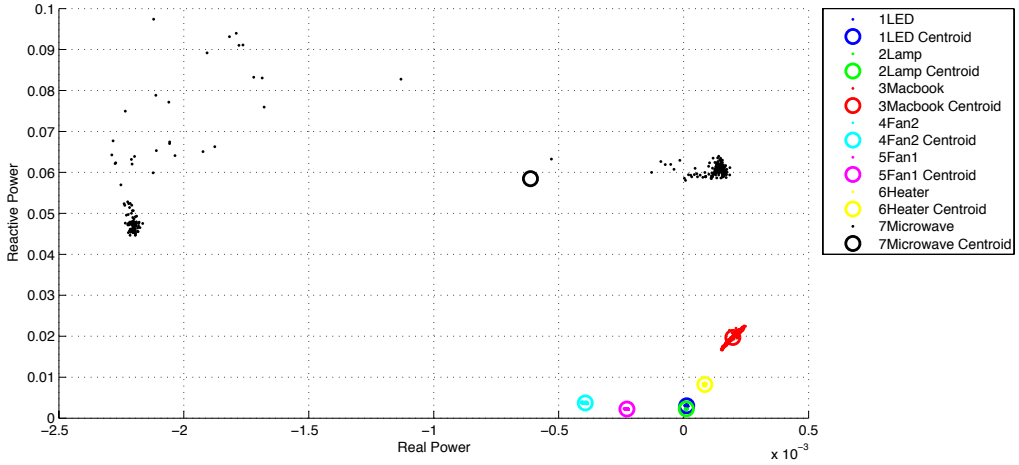

Figure 5.12: $P - Q$ view of Features distributions and centroids

features is overlapped in the feature space. However, if we switch to the 2-D view of $P-I_{P-P}$ in Figure 5.11, it is much easier to distinguish between them.

Also, from the view of $P - I_{P-P}$ (Figure 5.11), the Lamp and the Fan1 are overlapped, but if we switch to the view of $P - Q$ (Figure 5.12), they are clearly separated.

In other words, the feature selecting is critical to the accuracy of identification. Appliances that looks the same or similar in one dimension could be distinct in another dimension. In the proposed algorithm framework, it is possible to improve the accuracy for future identification by adding more meaningful feature-to-feature space.

Table 5.2: Confusion Table of Recognition on 7 Difference Appliances

	a	**b**	**c**	**d**	**e**	**f**	**g**	**Total**	**Recall**
a	116	0	0	0	0	0	0	116	100%
b	0	107	7	0	0	0	2	116	92%
c	0	0	116	0	0	0	0	116	100%
d	0	0	0	116	0	0	0	116	100%
e	0	0	0	0	116	0	0	116	100%
f	0	0	44	0	0	70	2	116	60%
g	0	0	0	0	0	0	116	116	100%
Total	116	107	167	116	116	70	120		
Precision	100%	100%	69%	100%	100%	100%	97%		

5.3.3 Appliance Features

Features unique to certain appliances are also interesting. In Figure 5.12, we can observe that the features of the LED, lamp, Fan1, Fan2 and Heater are centralized as they are stably working in one mode. However, the features of the Macbook look like a strip because the charging current of the Macbook increases during the charging process. In yet another example we see that the features of the microwave oven are distributed in 2 parts. The dots on the left-hand side can be explained by the mechanical rotating tray, and the right-hand part is generated by the heating function of the microwave oven.

5.3.4 Negative Real Power

An abnormal result we observed was that the real power sometimes becomes negative. To analyze this, we plot the real power verses time in Figure 5.13.

In Figure 5.13, the real power is negative even when there is no active appliance. This negative real power is not caused by the appliance, but the measurement system. Further investigation indicates that the negative real power is introduced by the phase difference between the current and the voltage signals. This is due to the mismatched delay in these

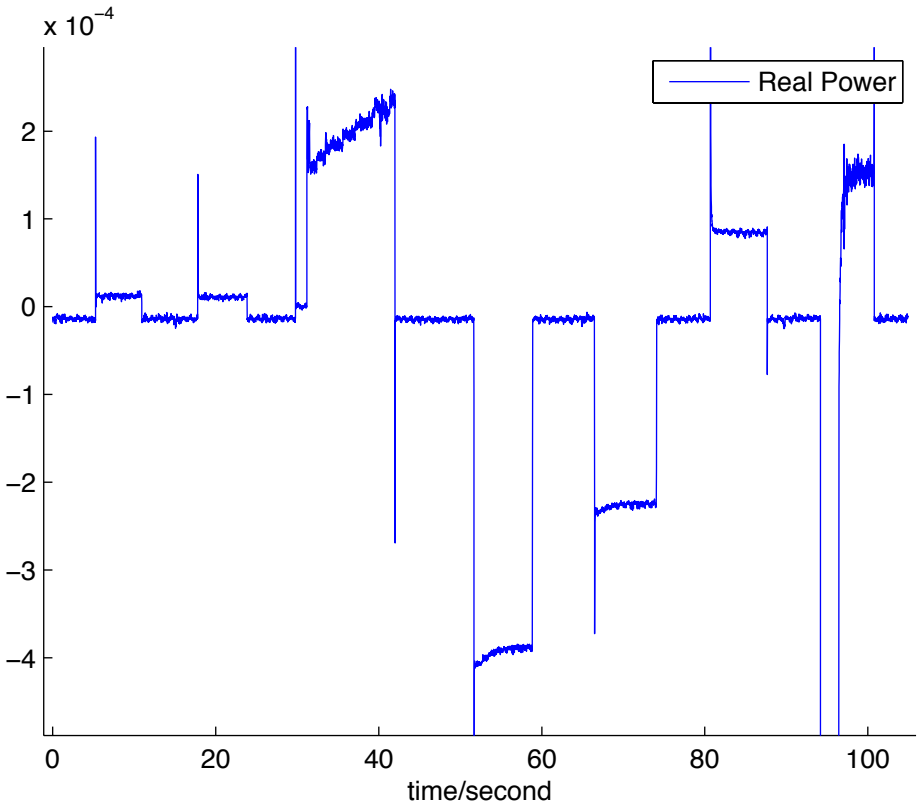

Figure 5.13: Real Power P

two channels. This means the "real power" we calculated is the real power of our entire system, but not the correct one for each load. The results could potentially be compensated for accuracy once the analog hardware-induced effects are properly analyzed and modeled.

5.4 Recognition Results

For this part, we discuss the recognition performance of our load identification system. For performance evaluation, we adopt a 10-fold cross validation strategy. Specifically, we divide the whole dataset into 10 sets. At one time, 5 sets are used to build the training data and the remaining 5 sets for testing. The whole procedure iterates 10 times.

Table 5.2 shows the confusion table with respect to appliance recognition. In Table 5.2, we notice that a) LED lamp, d) Fan (Speed 2) and e) Fan (Speed 1) are always correctly recognized and never confused with other appliances. This reinforces our visual observations that the electrical features from these devices are significantly different. Contrastingly, c) Macbook Pro and f) Heater are confused with each other the most. Specifically, the corresponding precision and recall are 69% and 60% respectively. This observation can be explained by the sophisticated charging system of the Macbook. It sometimes generates abnormal pulses during the charging and the features of these pulses usually deviate from the centroid and create a failed identification. This indicates that the selected features in Section 3 are not reliable enough to distinguish them. Furthermore, it is interesting that appliances with similar power are not necessarily "close" to each other in a feature space [such as b)incandescent lamp and e)fan (Speed 1)], and those with distinct powers are sometimes alike to each other [such as c)Macbook and g)Heater]. We consider these issues central to our efforts to improve the recognition performance in future work.

5.5 Proposed System

In this section, we introduce the hardware architecture and main components of proposed load identification system in details. While our experiments were done with hardware that

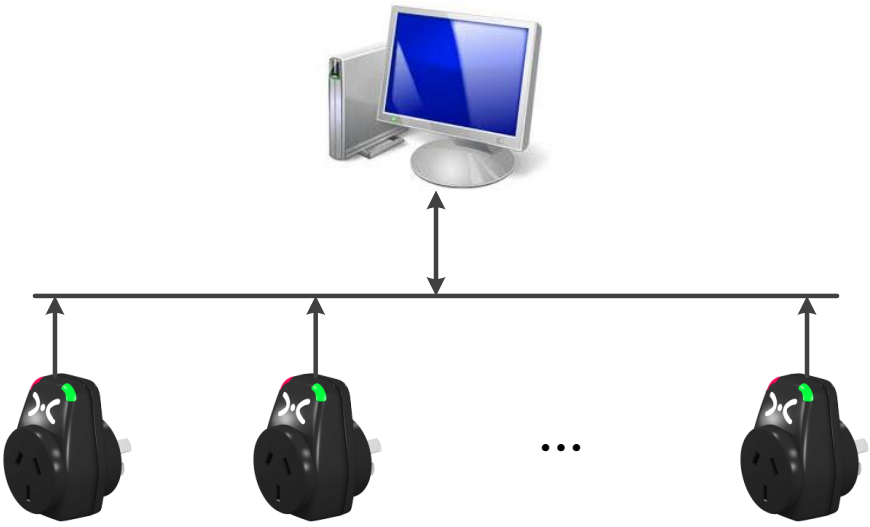

Figure 5.14: System Topology.

was capable of wide-band signal capture and conditioning, we bypassed much of this functionality and down-sampled our ADC data to 500Hz. Replacing this "data dumbing down" procedure with simple, cheaper hardware, the hardware system should be able to be integrated in a real outlet.

5.5.1 System Architecture

The topology of the proposed system under demand response framework is illustrated in Figure 5.14. It consists of a central "server" and a few load identification systems at the outlet level. Each node identification system is designed to satisfy *three* functionalities: 1)identify the appliance from the sensed raw data, such as the load current and the voltage on the power line; 2)communicate with the "server", for example, send the identification result to the "server" and receive the commands from it; 3)turn ON/OFF the appliance.

To avoid raw data transmission from the load identification system to the "server", all the identification work is conducted at the outlet level. This concept also enhances the scalability of the system as each identification system at outlet level is lightweight to the "server".

Figure 5.15 is the hardware architecture of the outlet level system. There are *five* com-

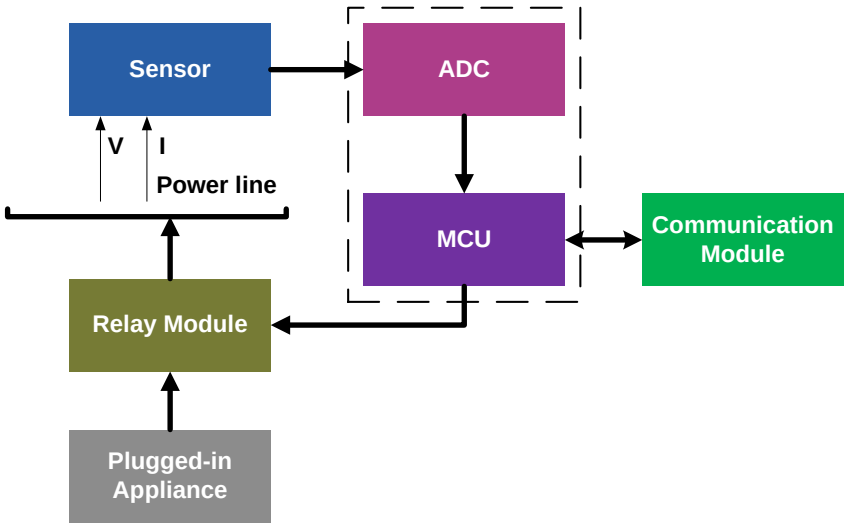

Figure 5.15: Load Identification Hardware Architecture

ponents including sensors, a relay module, ADCs, a Microcontroller Unit (MCU) and a communication module. As the relay and communication modules are easily accessible from the electronic markets, the following discussion will focus on the sensor, the ADC and the MCU components.

5.6 Analytical Review

This chapter discusses a low cost and efficient load identification system with good experimental accuracy. The algorithm flow includes two phases. In the first phase, the algorithm learns the features of each appliance by a sequence of raw data with appliance label (the ground truth). While in the second phase, it adopts the nearest neighbor matching to find the closest matched appliance in a 3-dimension feature space. This is done by considering *three* features: the real power, reactive power and the peak-peak current.

Since the algorithm only uses these three features at a data rate of 60Hz, the sample rate of raw data can be as low as 500Hz. Under this sample rate, the proposed system is far more efficient than existing EMI approaches, which usually require sophisticated hardware that can filter and sample at frequencies around 1MHz. Moreover, the computational complexity is also very low as the Nearest Neighbor algorithm is easy to implement and compute. Our

experiments on 7 appliances suggest that our proposed system could get accurate identification results while employing very simple hardware and minimal computational power.

As a preliminary prototype, there are also a few limitations and potential future improvements under the current hardware and algorithm framework. First, the proposed system will need to have good accuracy when identifying appliances with small power consumption. However, for appliances with very similar power consumption and power factor, such as two notebooks with different brands, it is very difficult to identify them using only low frequency features. Given the modern requirements of large buildings, it is possible that simply classifying a load as a "notebook" would be sufficient to determine whether power to the load could be switched off. Second, we need to maintain the feature library for identification processes. It becomes inconvenient to identify the unknown appliance if it has to go through the learning process to be registered in the library. One possible solution is to change the identification module to a classifier and include a confidence factor. Then the system will try to identify the appliance under test to one category. If the classifier is not confident enough, that is, the confidence factor is too low, the appliance under test will be considered as "unknown" to avoid an unplanned deactivation of the outlet by the building's energy management system. Another approach could involve an exhaustive load characterization with the goal of building a large "ground truth" library. Coupled with a confidence factor and some calibration, this could prevent the need for excessive "load learning" once the system is deployed to an operational environment.

CHAPTER 6

Detection Algorithms: A Broader Analysis

Our experimental efforts in the previous chapter yielded low-frequency features that enable highly accurate load detection. The features implemented are basic, however, and K-Nearest Neighbor (kNN) is a very simplistic machine learning algorithm. The focus of this chapter will be to dive deeper into potential features and apply a wider array of machine learning algorithms to the loads tested. For this analysis we import normalized features extracted from our raw data into a tabspace text file. Such files can be easily imported into an open source data mining software such as Orange [ora]. We will use this software to construct and test a variety of features and algorithm types.

6.1 2D Feature Set evaluated via 10-fold Cross Validation

Here we compare each algorithm implemented on a smaller number of features to see how each perform in such a case. There was no scientific method used to select Real and Reactive Power as the two features over other permutations, they were chosen simply to compare the capability of the chosen machine learning algorithms. The normalized data is plotted in Figure 6.1.

6.1.1 Naive Bayes

Naive Bayes classifier is a probabilistic classifier that assumes high level of independence between features [Nai]. The Naive Bayes classifier comes from Bayes theorem: $P(B|A) = \frac{P(B|A)P(A)}{P(B|A)P(A)+P(B|\bar{A})P(\bar{A})}$ which relates the probabilities of A and B with the conditional probabilities of A and B. The classifier is set up to use relative frequency to determine prior class

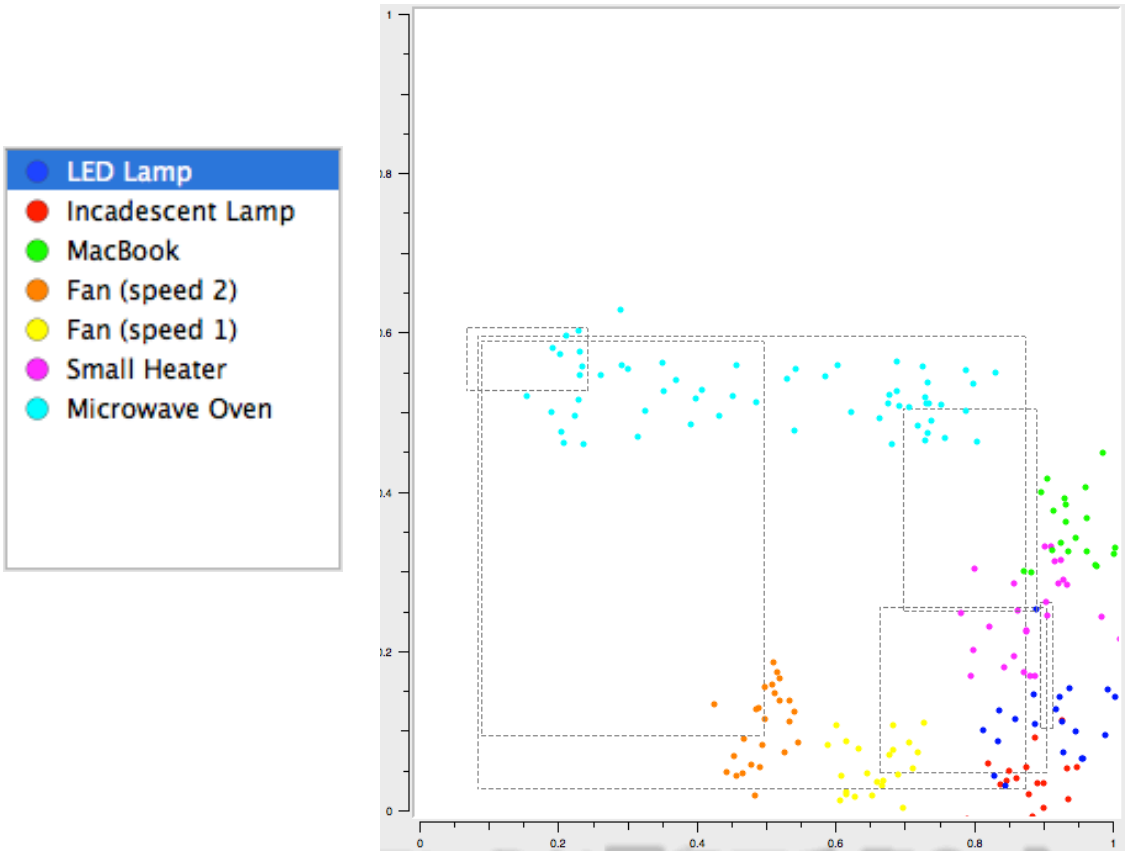

Figure 6.1: Scatter plot of normalized Real and Reactive Power.

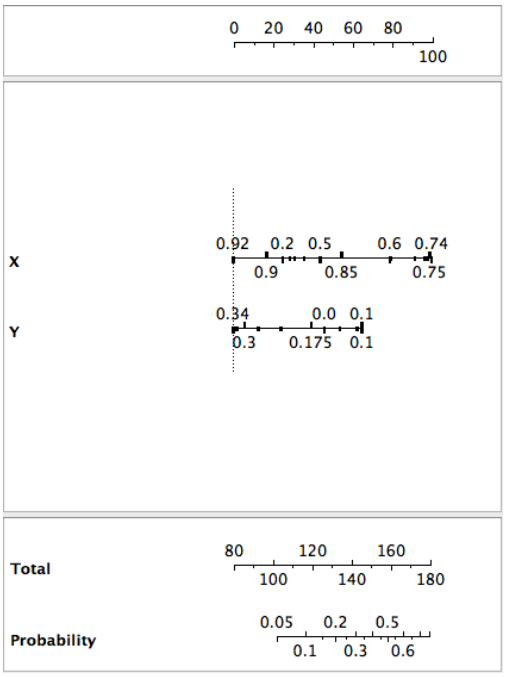

Figure 6.2: Nomogram of Naive Bayes classifier for the Macbook.

probabilities instead of the Laplace estimate. The LOESS window is set to 0.5 with 100 LOESS sample points. The nomogram of our classifier for the Macbook load can be seen in Figure 6.2. The confusion table for the detection accuracy of all seven loads is shown in Table 6.1. The identification accuracy is quite good for all loads, over 90%, except for the LED lamp and the small heater.

6.1.2 Support Vector Machines (SVM)

Support Vector Machines classifies data by constructing a hyperplane in the attribute space with the goal of maximizing the margin between instances of different classes [SVM]. We implement this classifier by specifying a "linear kernel" classifier:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1}^{n}$$

and setting the numerical tolerance to 0.001. We will discuss SVM parameter optimization later on in this chapter. An excerpt of the SVM prediction performance is shown in Figure 6.3. The complete confusion matrix is shown in Table 6.2. Similar to Naive Bayes we notice a difficulty when classifying the LED lamp and the Incandescent lamp as they are often

Table 6.1: Confusion Table of Recognition utilizing Naive Bayes

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	65%	15%	0%	0%	0%	20%	0%	20
Bulb	10%	90%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	90%	0%	0%	10%	0%	20
Fan2	0%	0%	0%	96%	4%	0%	0%	25
Fan1	0%	0%	0%	8%	92%	0%	0%	25
Heater	0%	0%	36%	0%	0%	64%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	15	21	27	26	24	22	60	195

Table 6.1: This is performed with 10-fold Cross Validation using a 2D feature set extracted from experimental data. Note: difficulty differentiating between LED and Incandescent bulbs as well as Macbook vs. Heater.

mistaken for each other. The other major identification difficulty involves the small heater being mistaken for the Macbook. We saw this same issue with our kNN using three features. We will address ways to overcome this problem later in this chapter.

6.1.3 Nearest Neighbor (kNN)

Nearest neighbor, as described in the previous chapter, identifies loads by finding the shortest distance to the nearest feature centroid [KNN]. The algorithm is repeated here in order to fairly compare this kNN iteration with the other tested algorithms using our normalized 2-dimensional feature set in a 10-fold cross validation methodology. The confusion matrix can be seen in Table 6.3.

Similar to our results in the previous chapter, we see "confusion" when identifying between the Macbook and small heater. There is a also a persistent confusion when identifying between the LED and incandescent bulb. This problem seems apparent for all classification methods.

	X	Y	Class label	SVM
1	0.304	0.166	Fan (speed 2)	Fan (speed 2)
2	0.300	0.111	Fan (speed 2)	Fan (speed 2)
3	0.398	0.145	Fan (speed 2)	Fan (speed 2)
4	0.436	0.131	Fan (speed 2)	Fan (speed 2)
5	0.386	0.121	Fan (speed 2)	Fan (speed 2)
6	0.316	0.141	Fan (speed 2)	Fan (speed 2)
7	0.278	0.102	Fan (speed 2)	Fan (speed 2)
8	0.344	0.159	Fan (speed 2)	Fan (speed 2)
9	0.317	0.148	Fan (speed 2)	Fan (speed 2)
10	0.350	0.112	Fan (speed 2)	Fan (speed 2)
11	0.514	0.112	Fan (speed 1)	Fan (speed 2)
12	0.534	0.138	Fan (speed 1)	Incandescent Lamp
13	0.544	0.060	Fan (speed 1)	Incandescent Lamp
14	0.532	0.090	Fan (speed 1)	Incandescent Lamp
15	0.547	0.054	Fan (speed 1)	Incandescent Lamp
16	0.545	0.078	Fan (speed 1)	Incandescent Lamp
17	0.467	0.075	Fan (speed 1)	Fan (speed 2)
18	0.523	0.080	Fan (speed 1)	Incandescent Lamp
19	0.572	0.086	Fan (speed 1)	Incandescent Lamp
20	0.535	0.106	Fan (speed 1)	Incandescent Lamp
21	0.702	-0.006	Incandesce...	LED Lamp
22	0.728	0.034	Incandesce...	LED Lamp
23	0.694	0.039	Incandesce...	LED Lamp
24	0.674	0.080	Incandesce...	LED Lamp
25	0.781	0.017	Incandesce...	LED Lamp
26	0.695	0.003	Incandesce...	LED Lamp

Figure 6.3: Prediction table excerpt of SVM classifier.

Table 6.2: Confusion Table of Recognition utilizing Support Vector Machines

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	50%	45%	0%	0%	0%	5%	0%	20
Bulb	90%	10%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	95%	0%	0%	5%	0%	20
Fan2	0%	0%	0%	96%	4%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	12%	0%	28%	0%	0%	60%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	31	11	26	24	26	17	60	195

Table 6.2: This is performed with 10-fold Cross Validation using a 2D feature set extracted from experimental data. Note: extreme difficulty differentiating between LED and Incandescent bulbs as well as Heater vs. Macbook and LED.

Table 6.3: Confusion Table of Recognition utilizing Nearest Neighbor

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	60%	30%	0%	0%	0%	10%	0%	20
Bulb	30%	70%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	90%	0%	0%	10%	0%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	8%	0%	8%	0%	0%	84%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	20	20	20	25	25	25	60	195

Table 6.3: This is performed with 10-fold Cross Validation using a 2D feature set extracted from experimental data. Note: difficulty in differentiating between LED and Incandescent bulbs.

6.2 3D Feature Set evaluated via 10-fold Cross Validation

Continuing with our experimental data set, we will now test all three features using 10-fold cross validation. This will serve as a useful comparison in the next section when we will use a separate training data set to train our algorithm models. The scatter plot of the 3-dimensional data set can be seen in Figure 6.4. The workflow for our analysis using Orange visual programming can be seen in Figure 6.5.

Using a 10-fold cross validation analysis we generate identification models for Naive Bayes, SVM, and Classification Tree [Cla] (see Figures 6.6, 6.7, and 6.8). These models, along with simple kNN analysis, give us the improved confusion matrices seen in Tables 6.4, 6.5, 6.6, and 6.7. The successful predictions of each algorithm are compared in Figure 6.9. While the Classification Tree seems to be the best overall performer, Nearest Neighbor and SVM perform as well or better when detecting the Macbook. When analyzing the performance of the SVM algorithm, the extra feature set seemed to improve the accuracy significantly for LED and Incandescent detection. The LED false positives labelled as Incandescent in the

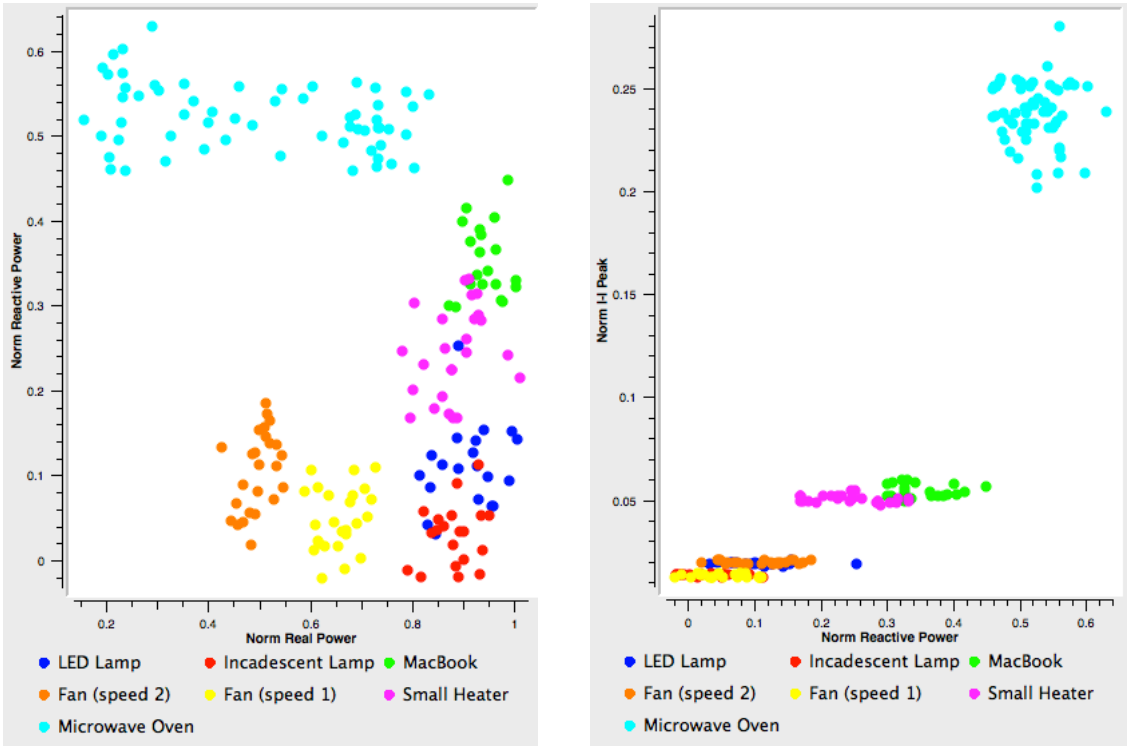

Figure 6.4: Scatter plot of normalized Peak to Peak Current, Real Power, and Reactive Power.

2D set was replaced with a majority correctly identified as incandescent. This recognition could still be improved and we will address this goal in the next section.

6.3 3D Feature Set with Training Data

A 3-dimensional feature set was shown to be effective at load identification with reasonable accuracy. This section will use a custom selected training data set with which we train our classifiers before then exposing them to the experimental data set used in the previous sections. Our experimental data set (see Figure 6.11) was chosen for its particularly "noisy" characteristics. Our training data (see Figure 6.10) is chosen carefully to be a cleaner representation of our load characteristics with these features. The goal with using this training set is to get better detection by providing clearer expectations for expected attribute feature spaces. The efficacy of this approach is immediately apparent with the performance of the simple kNN algorithm (see Table 6.8) where the identification accuracy is perfect.

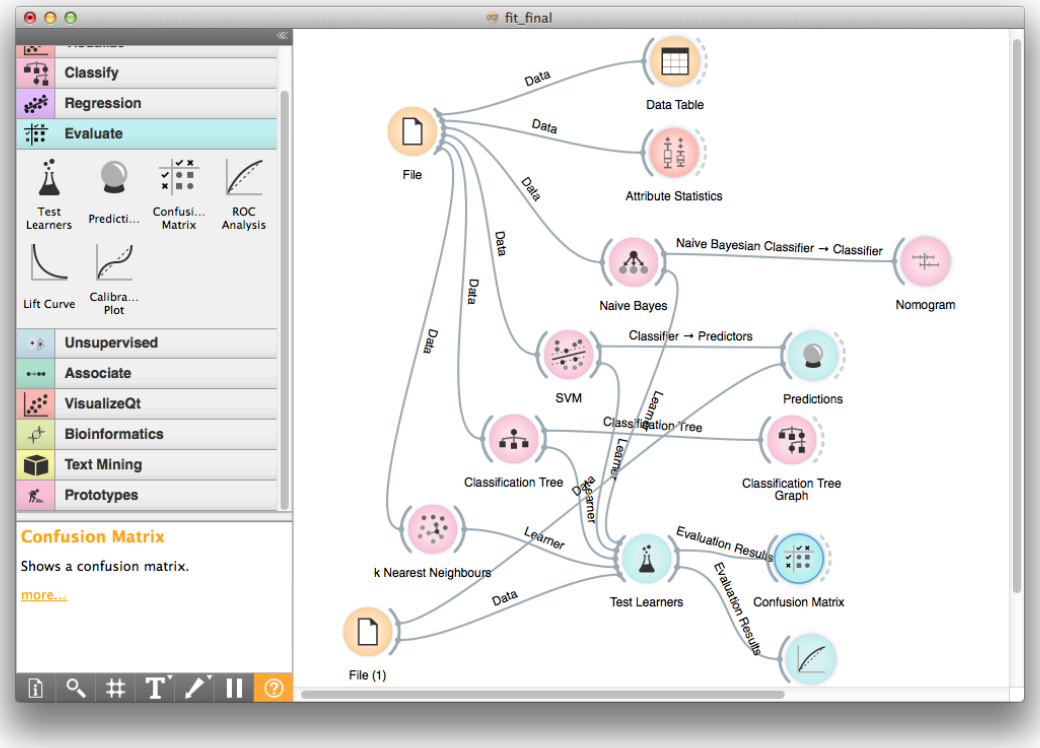

Figure 6.5: Orange workflow for analysis of 3D feature set with 10-fold cross validation.

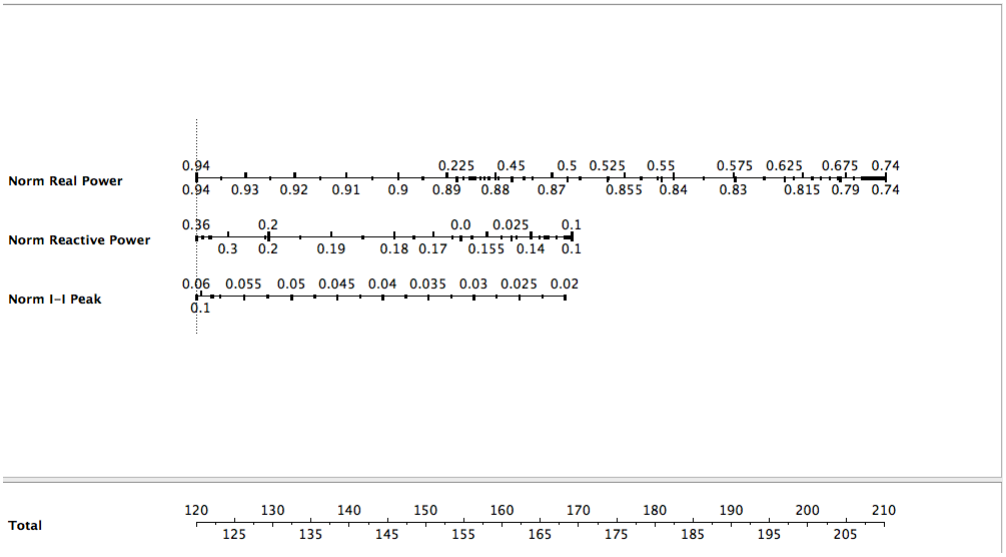

Figure 6.6: Nomogram of Naive Bayes classifier for the LED Lamp.

	Norm Real Power	orm Reactive Pow	Norm I–I Peak	Class label	SVM
99	0.857	0.285	0.050	Small Heater	MacBook
100	0.875	0.225	0.051	Small Heater	Small Heater
101	0.780	0.247	0.051	Small Heater	Small Heater
102	0.904	0.261	0.051	Small Heater	MacBook
103	0.887	0.168	0.052	Small Heater	Small Heater
104	0.921	0.285	0.050	Small Heater	MacBook
105	0.799	0.202	0.052	Small Heater	Small Heater
106	0.795	0.169	0.050	Small Heater	Small Heater
107	0.926	0.315	0.051	Small Heater	MacBook
108	0.875	0.225	0.052	Small Heater	Small Heater
109	1.011	0.215	0.052	Small Heater	Small Heater
110	0.821	0.231	0.052	Small Heater	Small Heater
111	0.880	0.169	0.052	Small Heater	Small Heater
112	0.905	0.245	0.055	Small Heater	Small Heater
113	0.902	0.331	0.051	Small Heater	MacBook
114	0.872	0.173	0.050	Small Heater	Small Heater
115	0.916	0.313	0.049	Small Heater	MacBook
116	0.913	0.326	0.050	MacBook	MacBook
117	0.962	0.366	0.052	MacBook	MacBook
118	0.926	0.336	0.051	MacBook	MacBook

Figure 6.7: Prediction table excerpt of SVM classifier.

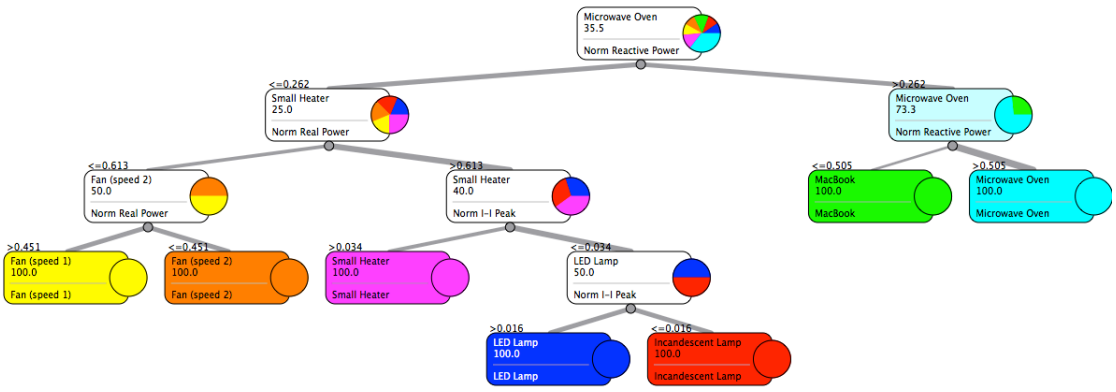

Figure 6.8: Decision tree for 3-dimension data set.

Table 6.4: Confusion Table of Recognition utilizing Nearest Neighbor

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	80%	15%	0%	0%	0%	5%	0%	20
Bulb	15%	85%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	95%	0%	0%	5%	0%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	0%	0%	8%	0%	0%	92%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	19	20	21	25	25	25	60	195

Table 6.4: This is performed with 10-fold Cross Validation using a 3D feature set extracted from experimental data.

Table 6.5: Confusion Table of Recognition utilizing Naive Bayes

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	80%	15%	0%	0%	0%	5%	0%	20
Bulb	5%	95%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	90%	0%	0%	10%	0%	20
Fan2	0%	0%	0%	96%	4%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	0%	0%	36%	0%	0%	64%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	17	22	27	24	26	19	60	195

Table 6.5: This is performed with 10-fold Cross Validation using a 3D feature set extracted from experimental data. Note: significant difficulty identifying Heater vs. Macbook.

Table 6.6: Confusion Table of Recognition utilizing Support Vector Machines

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	45%	50%	0%	0%	0%	5%	0%	20
Bulb	35%	65%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	90%	0%	0%	10%	0%	20
Fan2	0%	0%	0%	96%	4%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	0%	0%	20%	0%	0%	80%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	16	23	23	24	26	23	60	195

Table 6.6: This is performed with 10-fold Cross Validation using a 3D feature set extracted from experimental data. Note: severe difficulty separating LED and Incandescent Bulb loads.

Table 6.7: Confusion Table of Recognition utilizing Classification Tree

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	100%	0%	0%	0%	0%	0%	0%	20
Bulb	0%	100%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	85%	0%	0%	10%	5%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	0%	0%	8%	0%	0%	92%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	20	20	19	25	25	25	60	195

Table 6.7: This is performed with 10-fold Cross Validation using a 3D feature set extracted from experimental data.

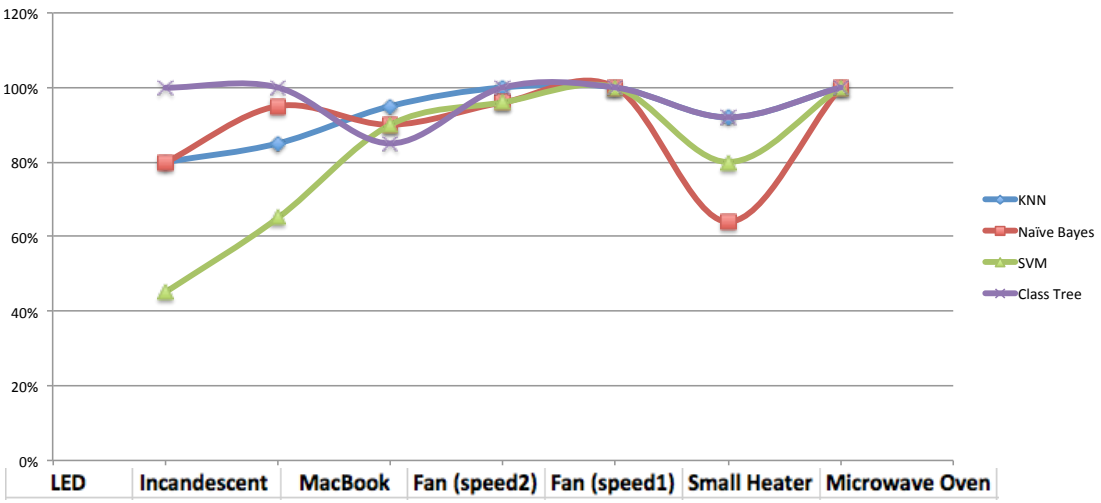

Figure 6.9: Scatter plot of machine learning algorithm successful predictions when utilizing 3-dimensional feature set.

This makes sense as the tighter load features in the hand-picked training data yields more precise classifier centroids, providing a more accurate model by which to judge and identify the centroids extracted from the experimental data.

Naive Bayes (see Table 6.9) had some improvement when differentiating between LED and incandescent lamps, but seemed to perform slightly poorer than 10-fold cross validation when differentiating between fan speeds. SVM (see Table 6.10) trained on training data resulted in a severe preference for identifications of the lamp loads as LED, resulting in a high false positive as LED for the incandescent events. Some of this LED lamp dominance even leaked into our detection for small heater. Meanwhile, all Macbook events were labelled as small heater, indicating a complete loss of accuracy in Macbook detection. In our 10-fold cross validation scheme the small heater was occasionally mistaken for the Macbook but only 10% of the time. Compared to the minor improvements in the classification of other loads, this iteration of our SVM algorithm was a bit of a disaster. Later in this section we will discuss how we can optimize the SVM algorithm to take better advantage of our custom training data set.

The Classification Tree (see Table 6.11) was able to resolve the inaccuracies in Macbook detection, correctly disaggregating it from the heater and microwave oven; resulting in a

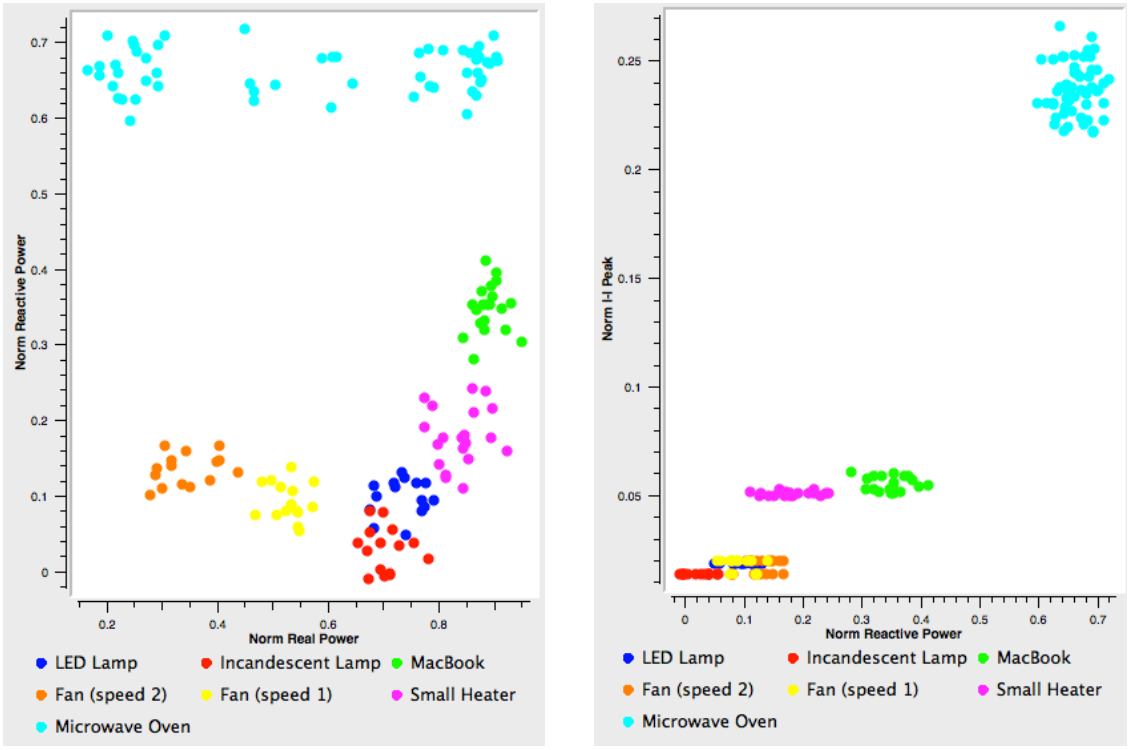

Figure 6.10: Scatter plot of training data: normalized Peak to Peak Current, Real Power, and Reactive Power.

100% identification accuracy. A comparison of successful load predictions can be seen as a scatter plot in Figure 6.15.

6.3.1 SVM Optimization

Given the complete meltdown of the SVM model when using a training set, there was an effort to see what went wrong and if the SVM algorithm could be better optimized. The training data seemed to improve or perfect load identification with all other algorithms, so it seemed fair to assume there was a problem with the way SVM was implemented. Using a built-in parameter selection tool in LIBSVM, the derived optimal parameters were entered into our SVM tool in Orange. The SVM model was then retrained yielding an "optimized" model. This optimization results are displayed in a new confusion matrix (Table 6.12) after having been tested the same experimental data. Surprisingly this results in a near perfect identification of our loads, with some remaining inaccuracy when separating the Macbook

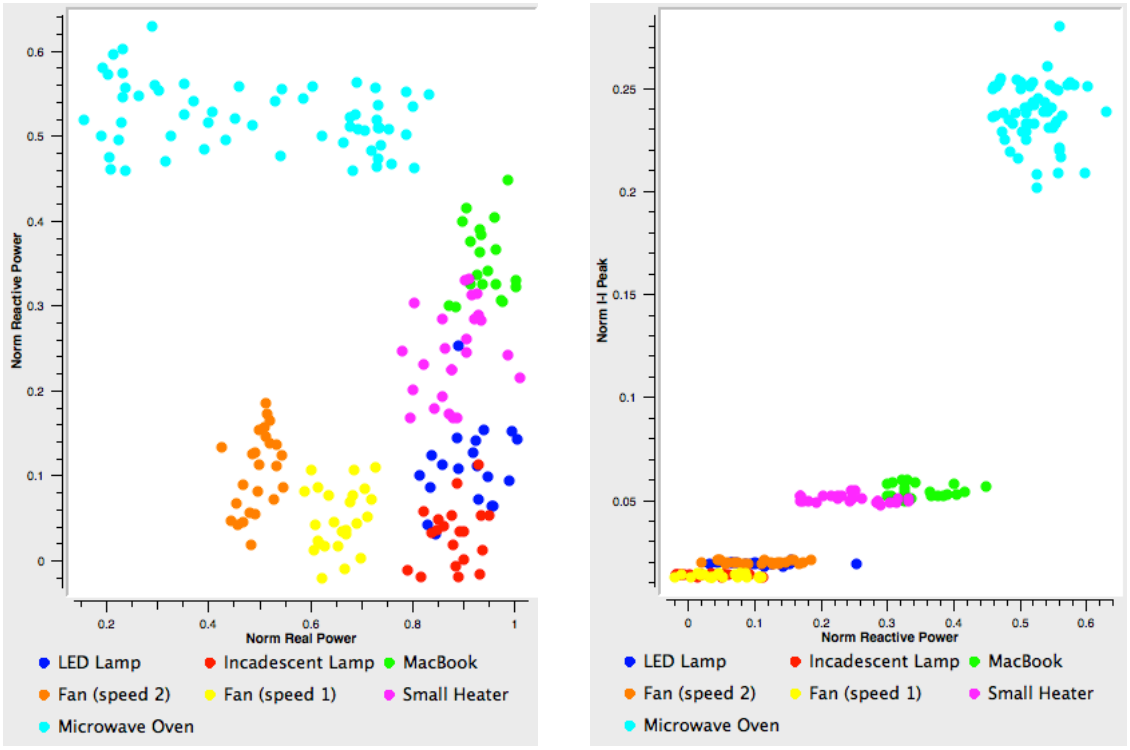

Figure 6.11: Scatter plot of experimental data: normalized Peak to Peak Current, Real Power, and Reactive Power.

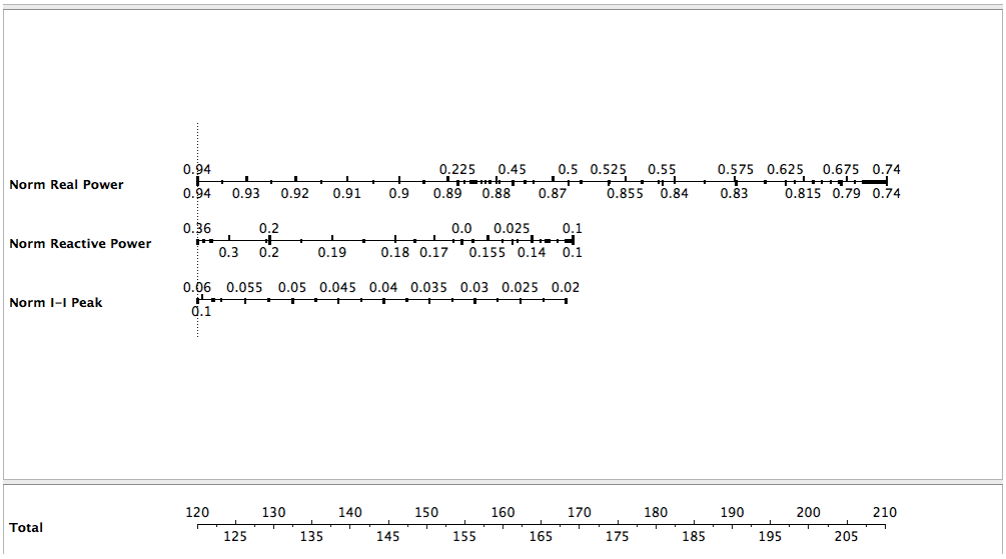

Figure 6.12: Nomogram of Naive Bayes classifier for the LED Lamp.

	Norm Real Power	orm Reactive Pow	Norm I–I Peak	Class label	SVM
1	0.534	0.111	0.019	Fan (speed 2)	Incandescent Lamp
2	0.468	0.090	0.019	Fan (speed 2)	Fan (speed 2)
3	0.490	0.128	0.019	Fan (speed 2)	Fan (speed 2)
4	0.499	0.114	0.021	Fan (speed 2)	Fan (speed 2)
5	0.467	0.046	0.021	Fan (speed 2)	Fan (speed 2)
6	0.509	0.158	0.021	Fan (speed 2)	Fan (speed 2)
7	0.527	0.073	0.020	Fan (speed 2)	Incandescent Lamp
8	0.542	0.124	0.020	Fan (speed 2)	Incandescent Lamp
9	0.486	0.126	0.019	Fan (speed 2)	Fan (speed 2)
10	0.484	0.019	0.020	Fan (speed 2)	Fan (speed 2)
11	0.495	0.082	0.019	Fan (speed 2)	Fan (speed 2)
12	0.513	0.147	0.020	Fan (speed 2)	Fan (speed 2)
13	0.479	0.057	0.020	Fan (speed 2)	Fan (speed 2)
14	0.443	0.047	0.021	Fan (speed 2)	Fan (speed 2)
15	0.519	0.138	0.020	Fan (speed 2)	Fan (speed 2)
16	0.511	0.185	0.021	Fan (speed 2)	Fan (speed 2)
17	0.515	0.173	0.020	Fan (speed 2)	Fan (speed 2)
18	0.547	0.086	0.019	Fan (speed 2)	Incandescent Lamp
19	0.426	0.134	0.020	Fan (speed 2)	Fan (speed 2)
20	0.457	0.043	0.021	Fan (speed 2)	Fan (speed 2)

Figure 6.13: Prediction table excerpt of SVM classifier.

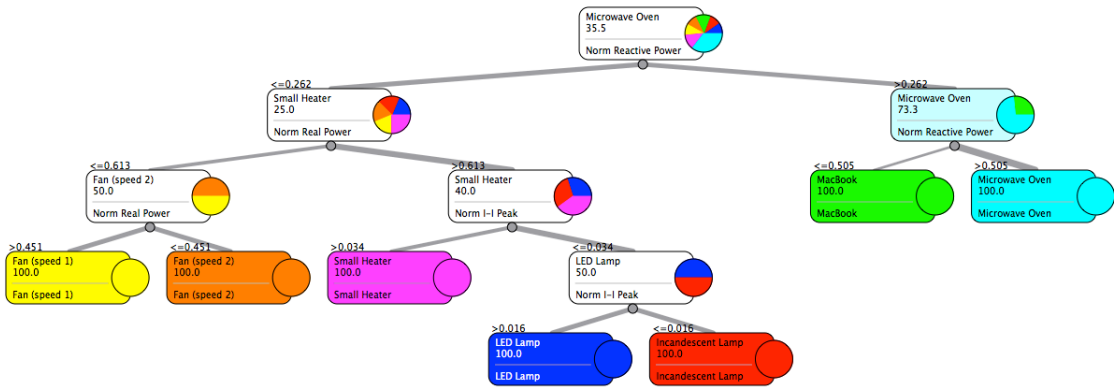

Figure 6.14: Decision tree for 3-dimension data set.

Table 6.8: Confusion Table of Recognition utilizing Nearest Neighbor

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	100%	0%	0%	0%	0%	5%	0%	20
Bulb	0%	100%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	100%	0%	0%	0%	0%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	0%	0%	0%	0%	0%	100%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	20	20	20	25	25	25	60	195

Table 6.8: This is performed with algorithms using a 3D feature set extracted from a training data set and tested on an experimental data set.

Table 6.9: Confusion Table of Recognition utilizing Naive Bayes

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	85%	15%	0%	0%	0%	0%	0%	20
Bulb	10%	90%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	90%	0%	0%	10%	0%	20
Fan2	0%	0%	0%	96%	4%	0%	0%	25
Fan1	0%	0%	0%	4%	96%	0%	0%	25
Heater	0%	0%	36%	0%	0%	64%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	19	21	27	25	26	18	60	195

Table 6.9: This is performed with algorithms using a 3D feature set extracted from a training data set and tested on an experimental data set. Note: there is still significant inaccuracy when identifying the Heater vs. Macbook.

Table 6.10: Confusion Table of Recognition utilizing Support Vector Machines

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	95%	0%	0%	0%	0%	5%	0%	20
Bulb	90%	10%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	0%	0%	0%	100%	0%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	12%	0%	0%	0%	0%	88%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	40	2	0	25	25	43	60	195

Table 6.10: This is performed with algorithms using a 3D feature set extracted from a training data set and tested on an experimental data set. Note: there is extreme inaccuracy when trying to identify the Incandescent Bulb vs. the LED load.

Table 6.11: Confusion Table of Recognition utilizing Classification Tree

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	100%	0%	0%	0%	0%	0%	0%	20
Bulb	0%	100%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	100%	0%	0%	0%	0%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	0%	0%	8%	0%	0%	92%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	20	20	19	25	25	25	60	195

Table 6.11: This is performed with algorithms using a 3D feature set extracted from a training data set and tested on an experimental data set.

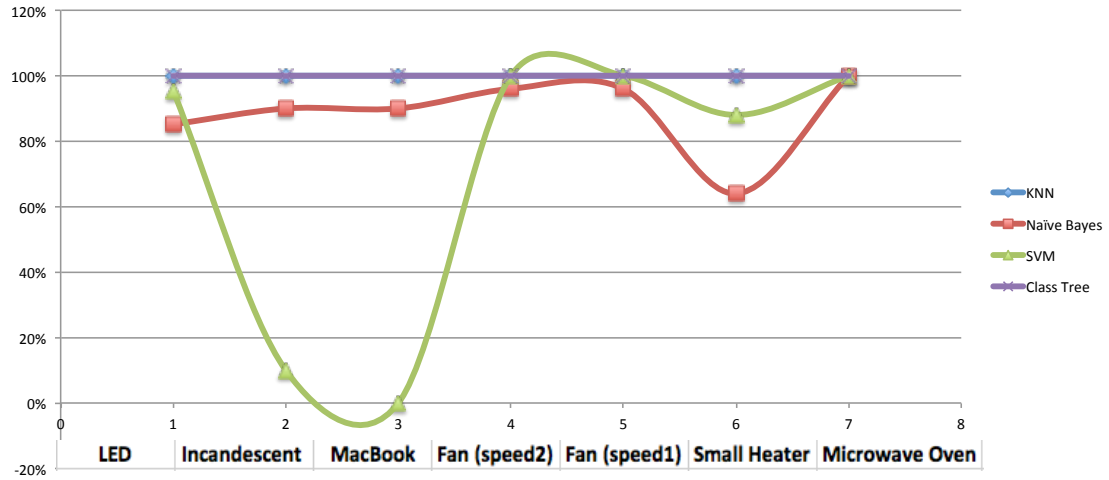

Figure 6.15: Scatter plot of machine learning algorithm successful predictions when utilizing 3-dimensional feature set extracted from training data.

from the heater. A scatter plot comparing this new SVM model to the other algorithms trained on the training data set can be seen in Figure 6.17. A comparison of our SVM algorithms extracted from 3-dimensional data sets can be seen in Figure 6.16.

6.4 4D Feature Set evaluated via 10-fold Cross Validation

To push SVM and Naive Bayes to perfect identification we need to incorporate additional features. While they no longer contribute to the narrative of a low-frequency load detection system, the use of high frequency features in conjunction with the low-frequency features we've already extracted should be a powerful tool for load analysis. In this section we will use "Center Frequency" as a fourth feature, extracted from the high frequency switching behavior some loads power supplies inject back on to the mains. This new feature dataset is plotted in the scatter plot in Figure 6.18.

The confusion matrices for the four machine learning algorithms can be seen in Tables 6.13, 6.14, 6.15, and 6.16. While SVM joined Nearest Neighbor and Classification Tree with perfect identifications, Naive Bayes still mis-categorized some Fan (speed 1) events as Fan (speed 2). A comparison of algorithm prediction accuracies is presented as a scatter plot in Figure 6.19. We solve the remaining misclassifications from the Naive Bayes model in the

Table 6.12: Confusion Table of Recognition utilizing "Optimized" Support Vector Machines

	LED	Bulb	Macbook	Fan2	Fan1	Heater	Microwave	Total
LED	100%	0%	0%	0%	0%	0%	0%	20
Bulb	0%	100%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	90%	0%	0%	10%	0%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	0%	0%	0%	0%	0%	100%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	20	20	18	25	25	27	60	195

Table 6.12: This is performed with algorithms using a 3D feature set extracted from a training data set and tested on an experimental data set. An optimized parameter selection results in a superior model and near perfect classification results.

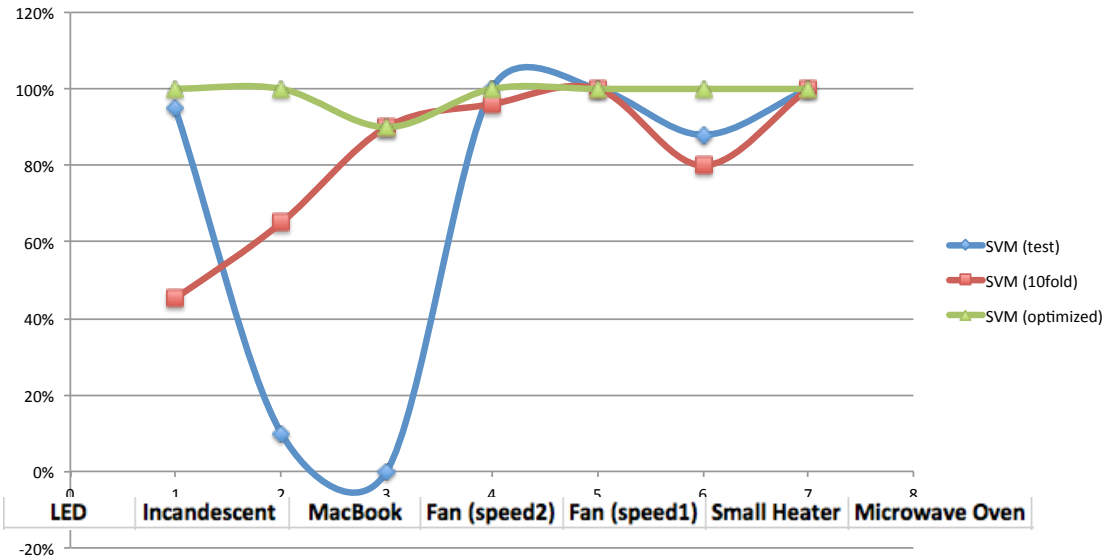

Figure 6.16: Scatter plot of Support Vector Machines learning algorithm successful predictions when utilizing 3-dimensional feature set extracted from training data. The three SVM models are compared here on their performance identifying the same 3-dimensional feature set.

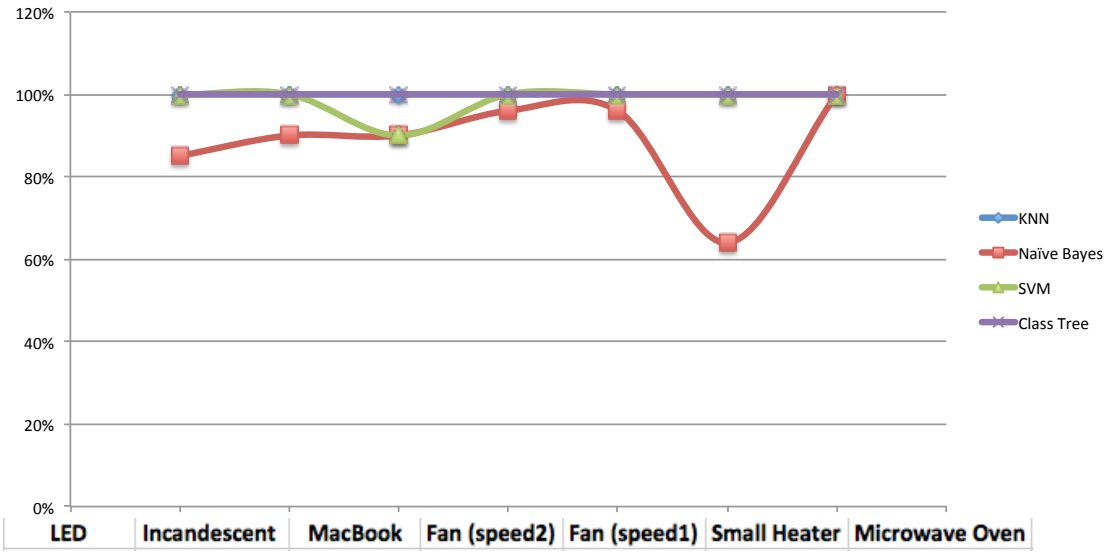

Figure 6.17: Scatter plot of machine learning algorithm successful predictions when utilizing 3-dimensional feature set extracted from training data. The SVM algorithm is plotted using the most accurate (optimized) model.

next section.

6.4.1 Feature Extraction: Center Frequency

To observe high frequency effects, raw voltage signals sampled at 1mega-sample/second is subjected to a Fourier Transform (FFT) which reveals the power spectral density (PSD) at all frequencies from 0 to 500kHz. By windowing and traversing this frequency spectrum, we can detect consistent noise injection at certain frequencies over the significant random noise that exists on the line. Using an experimentally calibrated threshold (see Figure 6.20) we detect and associate the location of these energy contributions to each of our loads. The average frequency at which these detections take place (within the windowed area) is called the "center frequency" and it is this value that we included as a feature in this section. In some cases, loads inject noise at multiple frequencies; sometimes due to harmonics but other times due to multiple operational modes of the power supply (PSU) or load. For the sake of simplifying this feature, we only used a subjectively determined "dominant" center frequency per load. We can see in Figure 6.18 with the Microwave Oven that even this one

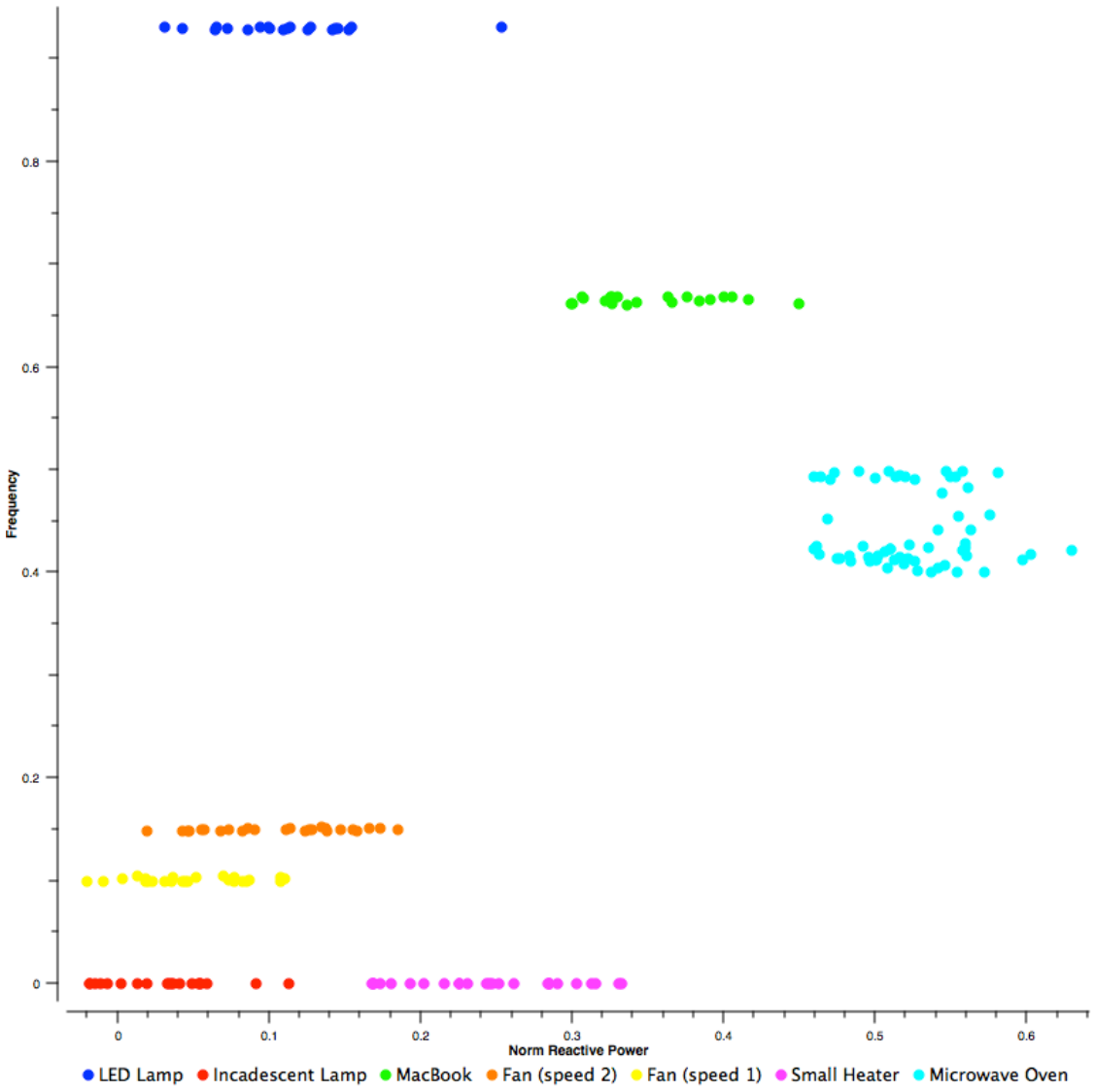

Figure 6.18: Scatter plot of normalized Center Frequency vs. normalized Reactive Power.

Table 6.13: Confusion Table of Recognition utilizing Nearest Neighbor

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	100%	0%	0%	0%	0%	5%	0%	20
Bulb	0%	100%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	100%	0%	0%	0%	0%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	0%	0%	0%	0%	0%	100%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	20	20	20	25	25	25	60	195

Table 6.13: This is performed with 10-fold Cross Validation using a 4D feature set extracted from experimental data.

Table 6.14: Confusion Table of Recognition utilizing Naive Bayes

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	85%	15%	0%	0%	0%	0%	0%	20
Bulb	10%	90%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	90%	0%	0%	10%	0%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	4%	96%	0%	0%	25
Heater	0%	0%	0%	0%	0%	100%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	20	20	20	26	24	25	60	195

Table 6.14: This is performed with 10-fold Cross Validation using a 4D feature set extracted from experimental data.

Table 6.15: Confusion Table of Recognition utilizing Support Vector Machines

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	100%	0%	0%	0%	0%	0%	0%	20
Bulb	0%	100%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	100%	0%	0%	0%	0%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	0%	0%	0%	0%	0%	100%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	20	20	20	25	25	25	60	195

Table 6.15: This is performed with 10-fold Cross Validation using a 4D feature set extracted from experimental data.

Table 6.16: Confusion Table of Recognition utilizing Classification Tree

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	100%	0%	0%	0%	0%	0%	0%	20
Bulb	0%	100%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	100%	0%	0%	0%	0%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	0%	100%	0%	0%	25
Heater	0%	0%	0%	0%	0%	100%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	20	20	19	25	25	25	60	195

Table 6.16: This is performed with 10-fold Cross Validation using a 4D feature set extracted from experimental data.

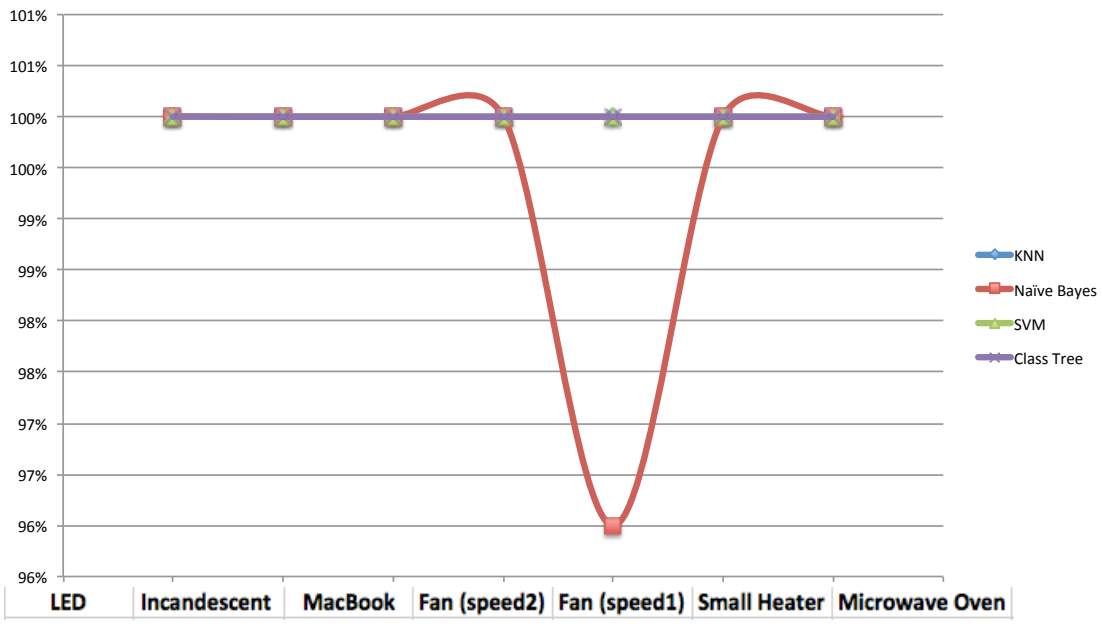

Figure 6.19: Scatter plot of machine learning algorithm successful predictions when utilizing 4-dimensional feature set.

center frequency event still resulted in what appears to be a 2-stage operational behavior. This likely is due to a different power supply being used to power the panel and controls while a higher capacity PSU supplies oven operation.

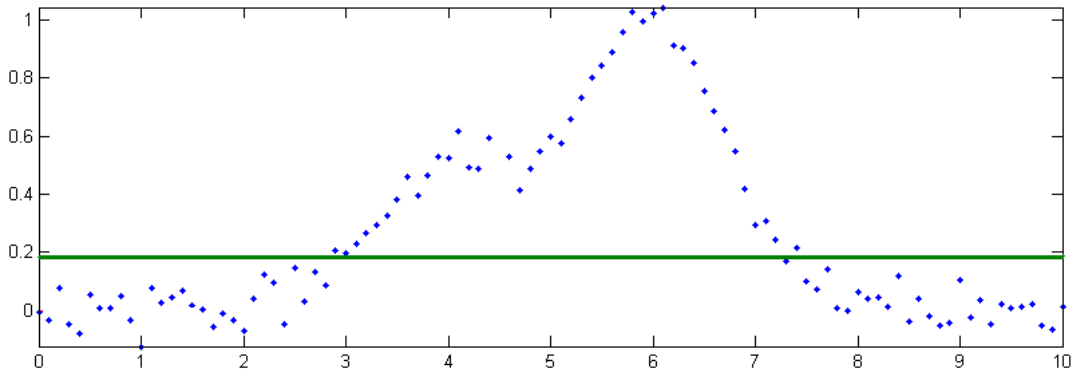

Figure 6.20: Thresholding of power spectral density (PSD) for feature extraction.

6.5 6D Feature Set evaluated via 10-fold Cross Validation

6.5.1 Features: Transitional Voltage Spikes

Up until now the established feature set only contains "regular" events that occur when a load is running. Peak-to-Peak Current, Real/Reactive Power, and Center Frequency are all behaviors that are visible throughout the operation of a load and give useful insight into the way the load draws power. For the next feature set, it seemed prudent to move away from these relatively "steady" features and instead look at transient effects that these loads generate on the line. When switching a new load on and off, there are noticeable voltage and current spikes on the line. By analyzing and characterizing these spikes hopefully we can give our learning algorithms enough data to completely isolate each load type from others and perfectly identify all of our loads. Given the various behaviors of our diverse load set, the voltage spike feature is separated into two categories: Voltage Spike at device turn-on (rising), and Voltage Spike at device turn-off (falling). These two features have been normalized and plotted as a scatterplot in Figure 6.21.

This new feature set, when coupled with the previously discussed four features, results in a perfect load identification with all our learning algorithms. Naive Bayes, our last remaining algorithm yet to achieve perfect classification, with these two transient features finally performs an unblemished characterization. The confusion matrix showing Naive Bayes classifier's correct prediction can be seen in Table 6.17.

6.6 Optimizing Feature Sets

Now that we have shown that all of our algorithms can achieve perfect classification, the next step is to go back and see which features can be filtered out while preserving our identification accuracy. In other words, what is the fewest number of features that can result in a perfect classification? At the same time, looking back to our analysis at the beginning of this chapter, can we achieve perfect classification with as few as two feature sets?

One motivation that will guide our search is the use of features extracted from voltage-

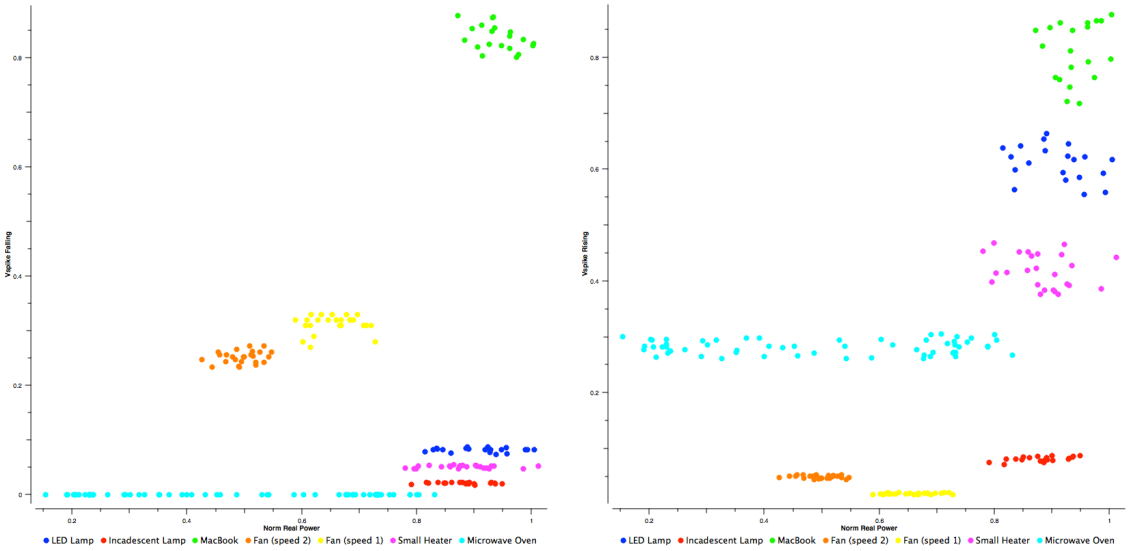

Figure 6.21: Scatter plot of normalized Falling Voltage Spike (left) and Rising Voltage Spike (right) vs. normalized Real Power.

Table 6.17: Confusion Table of Recognition utilizing Naive Bayes

	LED	**Bulb**	**Macbook**	**Fan2**	**Fan1**	**Heater**	**Microwave**	**Total**
LED	85%	15%	0%	0%	0%	0%	0%	20
Bulb	10%	90%	0%	0%	0%	0%	0%	20
Macbook	0%	0%	100%	0%	0%	0%	0%	20
Fan2	0%	0%	0%	100%	0%	0%	0%	25
Fan1	0%	0%	0%	4%	96%	0%	0%	25
Heater	0%	0%	0%	0%	0%	100%	0%	25
Microwave	0%	0%	0%	0%	0%	0%	100%	60
Total	20	20	20	26	24	25	60	195

Table 6.17: This is performed with 10-fold Cross Validation using a 6D feature set extracted from experimental data.

only measurements. While combining current and voltage features allows for powerful detection and identification, collecting current data is significantly more difficult than voltage data. For one, voltage signal noise can be seen "upstream" and "downstream." So a voltage connection at one outlet can perceivably see all the noise injected from nearby plugs regardless of where the connection is in the circuit. A current probe at the same point would only be able to detect current usage at outlets that are further downstream from the outlet being measured. All prior series outlets and any parallel lines would be invisible. As a result, to capture the current usage of an entire building, current probes would need to placed on the mains entering the building; a utility area that is not always easily accessible.

We can summarize our goals as follows: 1) Can we achieve perfect classification with as little as a 2-dimensional feature space? 2) Can we achieve perfect classification with features extracted only from voltage measurements?

To take a analytically quantitative approach to find our ideal features, we utilize a kNN algorithm to compute the quality of each feature pair. To rank each feature pair, we calculate a "score" for each feature set. This score is calculated with the following expression:

$$\bar{P} = E(P_f(y|x)) = \frac{1}{N}\sum_{i=1}^{N} P_f(y_i|x_i)$$

Evaluating feature permutations the top scoring feature pairs are displayed in Figure 6.22. Of these feature pairs, three can be obtained from only voltage measurements: 1) Frequency vs. Rising Voltage Spike, 2) Frequency vs. Falling Voltage Spike, and 3) Rising Voltage Spike vs. Falling Voltage Spike . The three highest scoring feature sets (see Figures 6.23, 6.24 and 6.25) are evaluated over our four machine learning algorithms in Figures 6.26, 6.27, 6.28, and 6.29. From these plots it is clear that we can achieve perfect classification with two 2-dimensional feature sets: 1) Frequency vs. Peak to Peak Current and 2) Frequency vs. Rising Voltage Spike. Of the three features that comprise these two feature pairs, only Frequency and Rising Voltage Spike can be obtained from voltage only measurements. Therefore we can confidently say that the optimal feature pair is *Frequency vs. Rising Voltage Spike*.

100.00 : Frequency, Norm I-I Peak
100.00 : Frequency, Vspike Rising
100.00 : Vspike Falling, Frequency
100.00 : Norm I-I Peak, Vspike Rising
100.00 : Vspike Falling, Frequency
100.00 : Vspike Rising, Frequency
100.00 : Vspike Rising, Norm I-I Peak
100.00 : Frequency, Norm I-I Peak

Figure 6.22: Top scoring feature pairs.

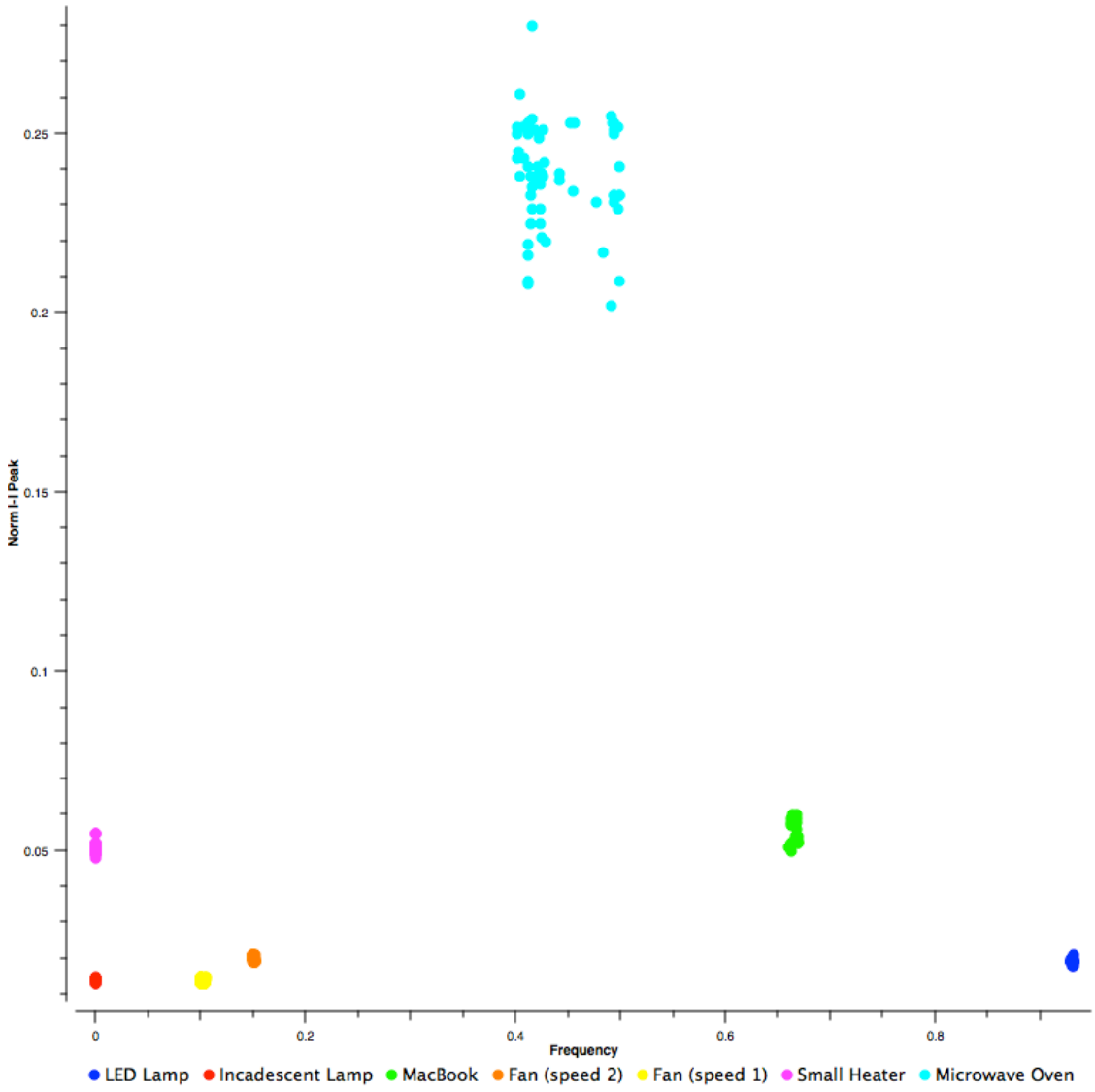

Figure 6.23: Scatter plot comparing Frequency vs. Peak to Peak Current

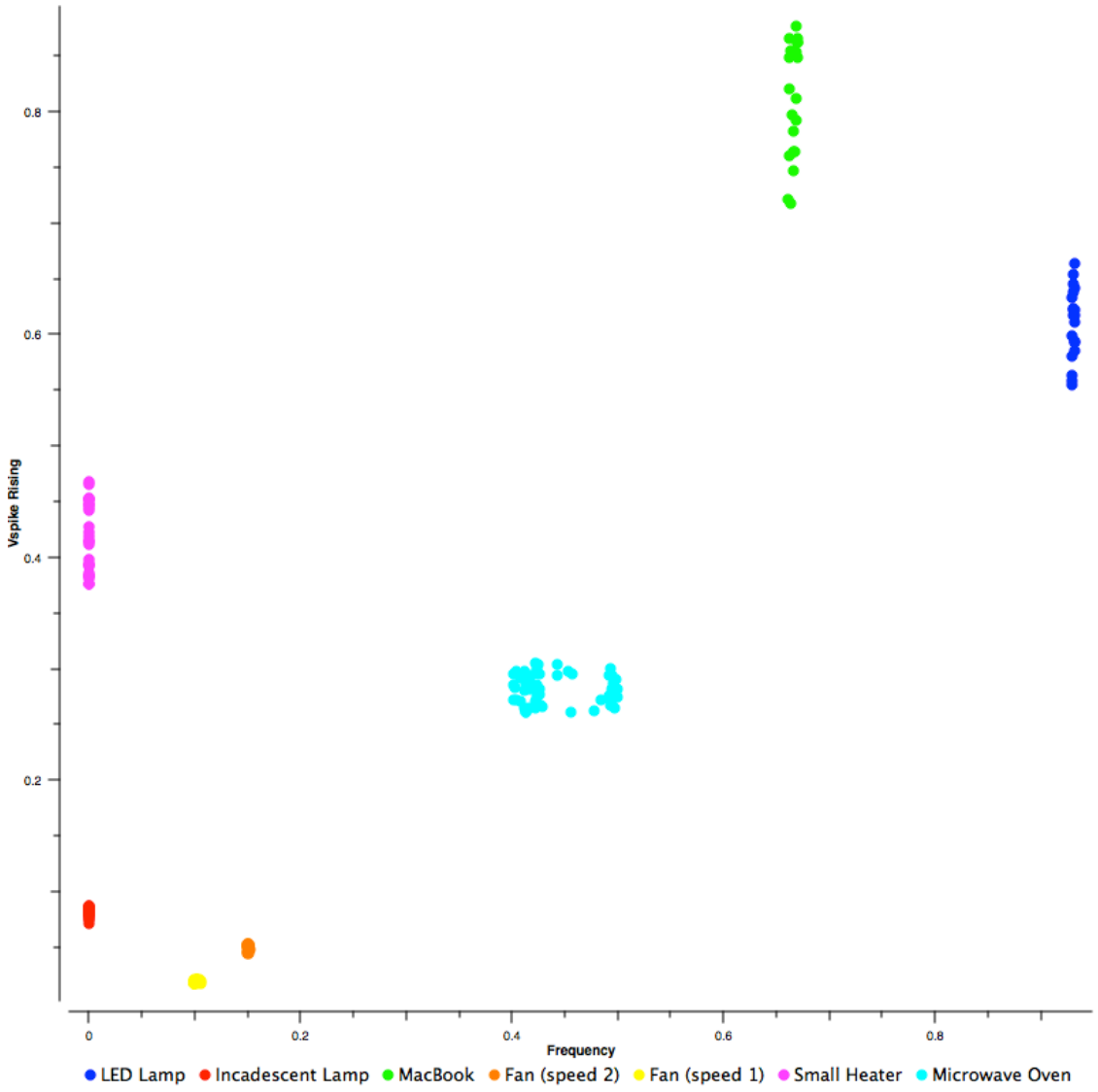

Figure 6.24: Scatter plot comparing Frequency vs. Rising Voltage Spike

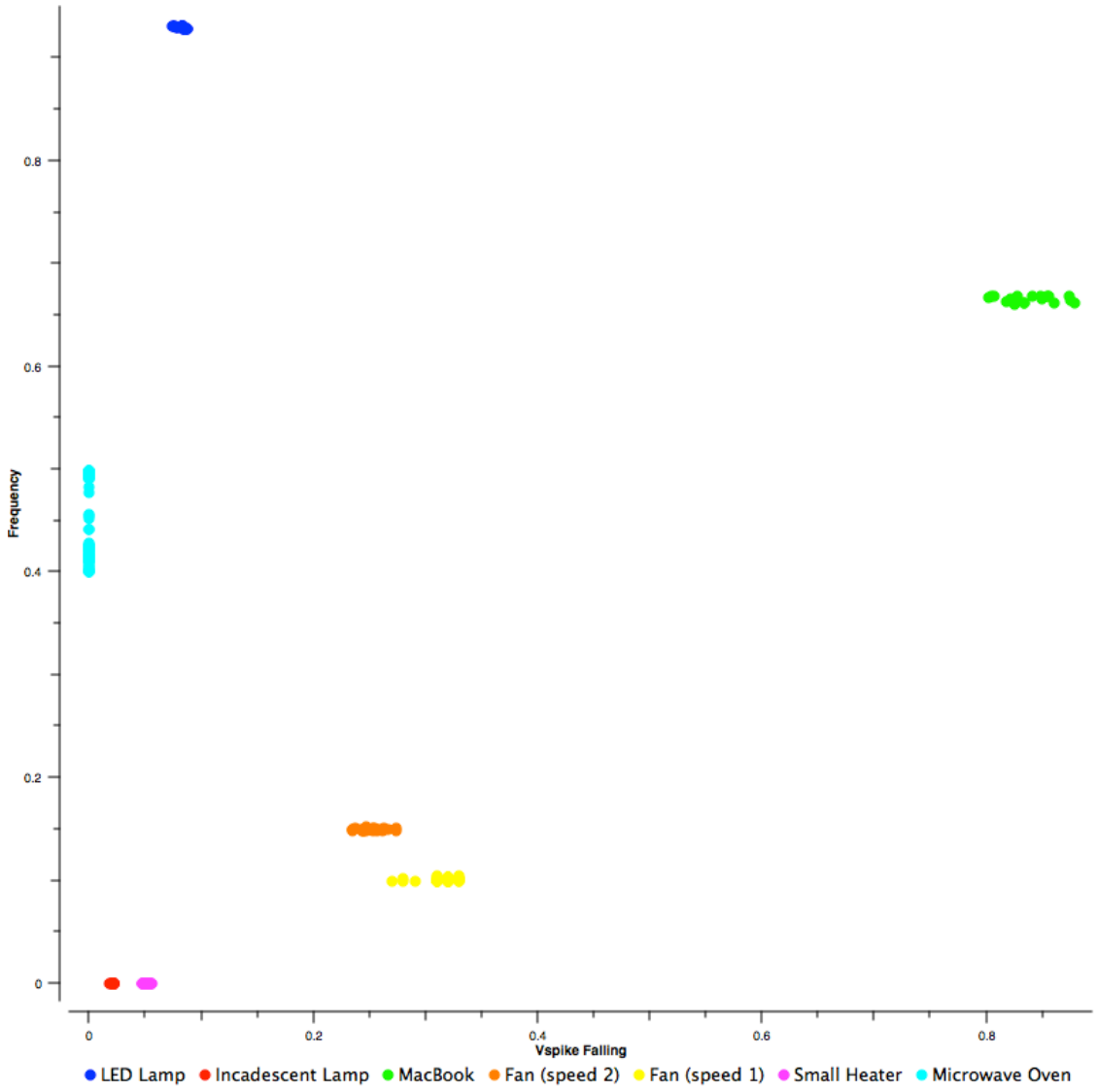

Figure 6.25: Scatter plot comparing Falling Voltage Spike vs. Center Frequency

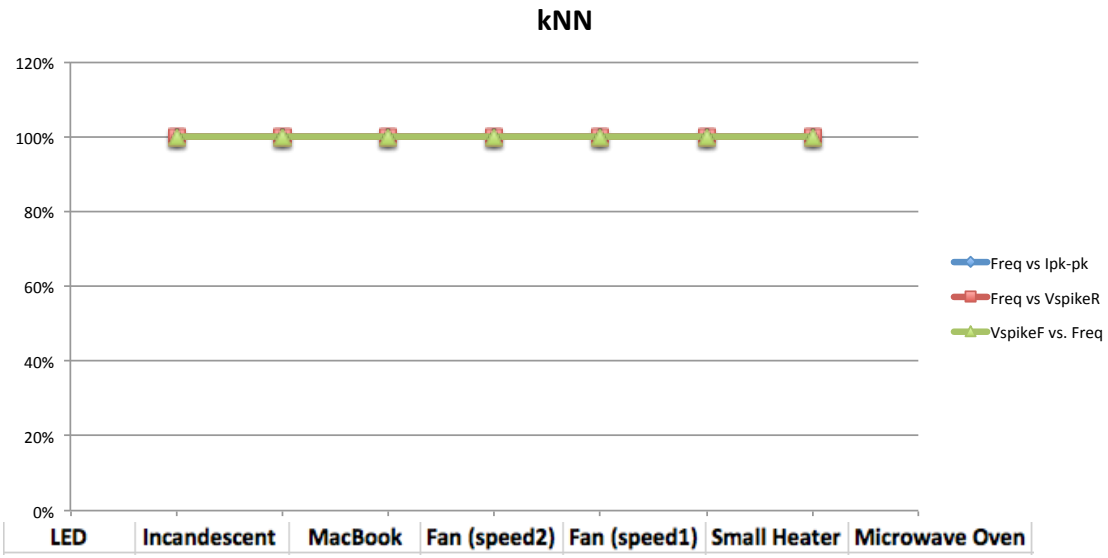

Figure 6.26: Nearest Neighbor classification accuracy when run on optimal feature pairs.

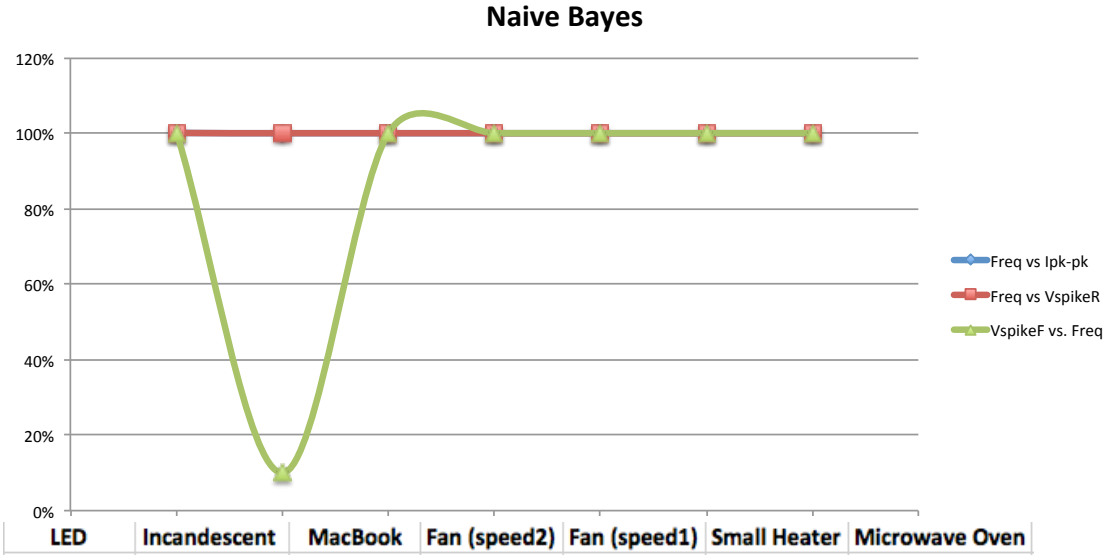

Figure 6.27: Naive Bayes classification accuracy when run on optimal feature pairs.

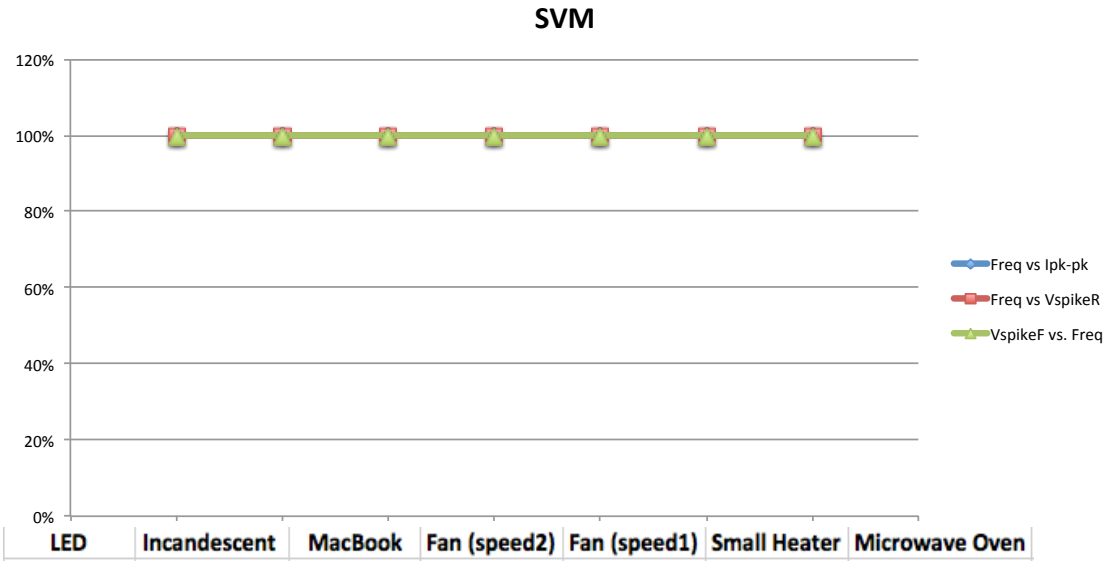

Figure 6.28: SVM classification accuracy when run on optimal feature pairs.

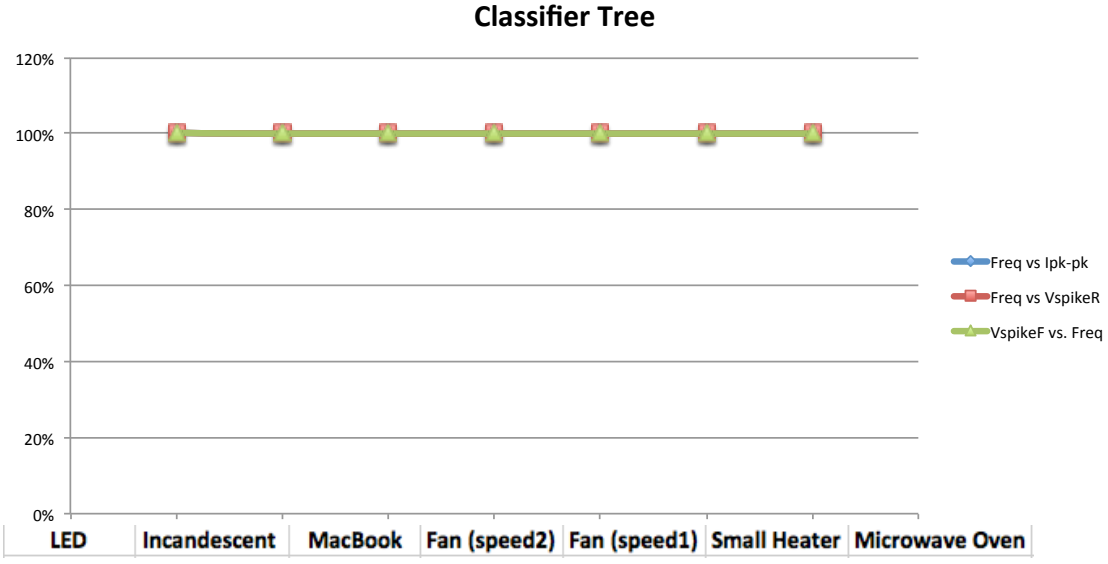

Figure 6.29: Classification Tree classification accuracy when run on optimal feature pairs.

CHAPTER 7

Conclusions

For many NILM detection schemes there is a strong need for accurate sensing of the power lines. To perform sensing in such a volatile and even hazardous environment, specialized hardware is required. The PowCap board was presented as our answer for this need, demonstrating functionality for receiving data at one voltage (capacitively coupled) and at one current line (current transformer to BNC). On board filtering and gain stages were provided via serial controlled firmware with a python front end. After extracting key features from these signals we showed we can accurately predict the loads attached, with perfect accuracy for most of the experimental loads. The loads that had poorer prediction ratios had clear weaknesses with respect to the chosen features. There is good reason to think that further analysis to discover additional features, and incorporating them into our algorithm, will improve on these results.

It was demonstrated that the PowCap Board acts as a natural receiver in a communication system where the power lines act as the transmission medium. It was a small jump to implement, with simple circuitry, a transmitter in parallel with our mains coupled "receiver" thereby allowing the board to function as a power line transceiver. We demonstrated, via experiments with baseband and FM modulation, that the theory was sound and outlined the opportunity for modern modulation schemes used in crowded airways to be implemented in the noisy atmosphere of the power lines.

In this thesis we showed how the design of these prior systems lead to the development of the PowCapMobile Board. This board, when combined with a Linux box and a USRP, is capable of taking four input signals, conditioning them, and converting them to digital data at a very high fidelity. These systems, together, are referred to as the PowCapMobile

Deployment Suite: a shockproof, robust, analog data-acquisition system. Files are stored on local drives in a complex binary format and are segmented into one hour divisions. The tools are highly customizable, allowing for custom file lengths, sample rates, digital gain, and target directory. Existing experimental data has been uploaded to a dedicated NAS server, which allows for easy access for anyone interested in analyzing household mains data. The files are read with a custom python software toolkit (see Appendix D) that windows the data and allows for various plotting or file-storage options.

We have presented several systems that provide flexible sensing and transmitting capabilities over the power line. In the following sections we discuss our findings and further explore their significance and potential looking forward.

7.1 PowCap: Looking Forward

Sensing voltage and current with the PowCap board resulted in high accuracy load detection using highly optimized low frequency features. This gave good promise to the potential for a low-cost low sample-rate load detection system; potentially one cheap enough to install at each outlet with relay functionality. This ties in nicely to some hopes for a "smart grid" that would allow for remote detection and control of loads in homes. Ostensibly, this control would be given to power providers: the electric company. When each outlet detects the type of load plugged in and notifies the smart meter, via wireless connection or power line communication, the system can decide if higher power but low priority loads should be shut off.

This simple application serves to balance the loading of the grid throughout the day. Electricity, unlike gas for a vehicle, cannot be stored easily; or rather, it cannot be stored efficiently. As a result, when a power plant generates electricity, it needs to be utilized immediately or it goes to waste. This has led power plants to heavily research load usage patterns, in an attempt to better predict load needs so that plants can wind up and wind down energy generation as demand fluctuates throughout the day. This creates a few problems, the most costly of which is unnecessary overhead. Power grids are often designed for

their peak requirements, not their average requirements. This makes sense as in the middle of a hot day, air conditioning usage is likely to drive energy consumption to a peak. If the grid cannot support this peak load, then there would be blackouts. However, by designing a grid for this peak situation, it means the grid is often underutilized. There is extra capacity that, for most of the time, is going to waste. This capacity is not cheap, it is realized in infrastructure as thicker wires, larger transformers, more generators (running idle in non-peak scenarios), and more power plants. If loads could be balanced throughout the day, month, and year, then utilities could optimize the use of the infrastructure to minimize operational and baseline costs. This means less wasted energy, which means cheaper energy costs.

Back to our smart meters, it is easy to see how having knowledge of individual load characteristics and control of the power supply to such loads, a power company might utilize self-defined personal preferences from a customer's profile to make decisions about what loads are essential and what loads are non-essential at any given time. A washer turning on in the middle of a hot day might be a low-priority for a household, and be a good candidate to be disconnected as everyone's air conditioners are driving up the demand. As the evening comes and the weather cools, demand drops so the washer switches back on and continues its cycle. In this way, the energy demands are "balanced" so that they are as constant as possible throughout the day, month, and year.

7.1.1 Machine Learning Algorithms

In the process of developing a system for load monitoring and control, we took the time to analyze how popular machine learning algorithms perform when executed on our data. It is clear that certain algorithms, like kNN and Classifier Tree, are simply better suited for accurately classifying data of this variety and scope. For more advanced algorithms like SVM and Naive Bayes, we demonstrated how a flexible feature set and good training data goes a long way towards ensuring optimal results. Finally, we worked our way back from a 6-dimensional feature set to a simple 2-dimensional feature set derived from voltage characteristics only: *Frequency vs. Rising Voltage Spike*. Demonstrating perfect characterization

with only two features that can be obtained from a more accessible point (voltage connection at any outlet) we clearly demonstrated the value of an in-depth algorithm analysis. At one point in our analysis a 4-dimensional feature set still yielded imperfect results with a Naive Bayes classifier; but only through the addition of extra features and retroactive analysis of feature permutations was a superior classification accuracy reached with fewer data points than any other successful approach.

The main question looking forward is how to modify our approach when dealing with more complex experimental setups. A half-dozen loads in a lab setting are much more easily identified with simple algorithms and a couple of features than three dozen loads in an industrial building. A diverse set of features is going to be a key consideration, along with a clever algorithm selection. A classifier tree seems an effective foundation for sifting through a large number of classifications and making the most of each feature set. Deploying multiple feature sets per branch or utilizing other learning algorithms within a branch is a potential method for solving difficult branch-level classifications. Like the work in this thesis has shown going from two features, to six features, and back to two features again; often learning methodologies must be overpowered to achieve accuracy, only then can they be paired back down to an optimal efficiency. It is likely that for more complex situations, a large number of features and algorithms will need to be employed in parallel. Once the scope of the environmental system is understood, the complexity of the classification system can be brought down in a way that minimally impacts accuracy. The hope should be that, with time and experience, commonalities between certain environments can be exploited. As different types of loads are identified in different types of environments, certain approaches will hopefully become consistently more successful than others. With enough data a potential meta-classifier could be designed which identifies the type of environment and then deploys the algorithms and optimizations best suited for that space.

7.2 PowCap with PLC: Looking Forward

When sensing in the environment of the mains, it is useful to understand how signals propagate. To this end, our research into a transceiver that operates over the power line gives both insight and functionality into a closely related area to NILM. Off the shelf systems, like The Energy Detective (TED)[TED], utilize power line communication to transfer information between sensors and the main processing unit. In our own systems, the PowCap Board is more than capable of processing the signals and acting like a receiver. Simply adding a transmitter function opens up the possibility to have multiple devices throughout a large building, communicating with each other, sharing information about new transitions, and compensating for "dead spaces" in large circuits caused by transmission losses. Using the advances in communication theory coupled with known models of mains circuits, optimal encoding techniques can be applied to overcome noise and operational bandwidth limitations.

The benefit of power line transmission does not end there. An interesting issue with NILM as a field is the question of privacy. Most of the data that can be acquired from any outlet can also be seen by potentially malicious viewers. External or remote electrical outlets could be used by a snooper to view the noise characteristics of a home and potentially identify load usage. Looking at load usage patterns, or even simply monitoring the activity on the mains, this viewer could then determine household patterns of activity. Low activity might suggest the owners are not at home, and might motivate a robbery. To obscure such malicious uses of NILM systems, it might be an effective strategy to "mimic" the signature of loads that are not actually on. The same NILM system that recognizes certain features as associated with a certain load, could convert those features back into an output and transmit them out onto the power lines. The goal here is not traditional power line communication, but rather to simply make it appear that certain loads are running when they are not. The internal software could easily resolve what loads are being transmitted versus those that are being sensed, and prevent any disruption to the installed load detection algorithms. However, to an external viewer, certain loads would appear to turn "on" and "off" with no clear pattern. This might preclude this viewer from committing a more malicious act, such as a robbery,

by making the household appear to have unpredictable schedules.

A similar situation took place in the NESL lab where a lab member's house was being monitored, intentionally and voluntarily, by a TED unit. The owner of the building was insisting on charging for utilities based on the TED readings, instead of simply forwarding the energy company bills. To prove that systems like the TED are unreliable, the PowCap system was used to analyze the TED system and discerned the details of its PLC protocol. Once it was clear that the sensors spoke to the MCU through the power line, a simple capacitively coupled MOSFET was tuned to the transmission frequency, around 150kHz, and broadcast a high amplitude sine wave to jam these signals. The result was that the TED could not resolve any changes in state, and continued to report the previous state. The reference documents[jam] cite a link to a video demonstrating this effect, where a load is turned on, jammed, and then turned off. Due to the jamming, the TED continues to report the load, and overall energy consumption, as being the same as when the light was on even though it was off. While there isn't always a jamming signal being broadcast at transmits frequencies, it was a convincing argument that a system primarily utilizing power line communications at reasonably high frequencies was in danger of losing signal integrity especially from other high power loads using switching power supplies operating at similar frequencies. The injected EMI could easily act as a jamming signal to suppress or disrupt the PLC functionality. Such a vulnerable system was not a good candidate for a billing methodology, whereas the power meter, which is hardwired onto the mains voltage and current, provides secure and direct measurements.

7.2.1 Proposed IEEE Specification: Load ID Broadcasting

An obvious approach to load identification that arose from this research was the idea of self-identifying loads. The idea is that, under a hypothetical industry-wide spec, power supplies would include self-identifying start-up circuitry. Since it is clear we can observe heavily filtered high frequency EMI injected onto the line, detecting primary current sinking and sourcing functionality is trivial. This start-up functionality could encode detailed information

about the device type, serial identification, and revision. Further, it could also include information about the device's current mode of operation, or list all modes of operation and allow any sensing system to distinguish between them.

The beauty of this model is that there is little to no overhead cost. The additional circuitry required to encode such a transmission on the PSU is negligible if it is incorporated at the integrated circuit (IC) stage. Newer PSUs are already incorporating FPGAs for high levels of programmability and customizability; from these Power Management IC (PMIC) solutions, even dense startup coding could be implemented with minimal effort from design teams. Sensing would also require little to no additional development. Modern smart meters already incorporate FPGAs into their power meters, and take detailed data on voltage and current usage. Programming them to interpret simple encoded messages onto predetermined frequencies would likely only require a software update of the meters. These systems are currently over-designed in order to accommodate future developments such as the system proposed here, and such functionality likely exists already. This particular capability is well documented in this paper due to our extensive analysis of sensing signals on the mains; both intentionally transmitted signals and coupled noise signatures.

With the understanding the technical capability exists, the next step would be define the specifics of how such a protocol would be defined. Frequency "bands" would be defined for different classes of loads, lower power loads being given lower frequencies while higher power loads would be tasked to transmit their code at higher frequencies. This would scale well with the transmission characteristics of powerlines in the home and workplace, allowing the "strongest" loads to transmit at the least efficient frequencies, and the "weakest" (low power) loads to transmit at optimized powerline frequencies. Within these power-dependent frequency bands, additional delineation would be defined by load type. An example of such segmentation, using common consumer electronic power levels [powb, powa], is as follows: 0 to 10W, 10 to 25W, 25 to 100W, 100 to 1kW, and 1kW plus. Coding methodologies that exist for overcrowded wireless communication would apply nicely to these narrow bands where many loads might be starting up in a coordinated fashion. Being able to decode multiple simultaneous startups would be a minimum requirement for such a system, but this

is an obtainable goal for an industry-wide adopted protocol.

7.3 PowCapMobile Deployment Suite: Looking Forward

As shown earlier, these other system designs culminated in the PowCapMobile Deployment Suite: combining USRP data converters and a PC for a fully functional deployable unit. While this system is optimized for high voltage readings on two of the channels, it could conceivably take data from any source. This means the deployment suite can act as an analog front end to any source signal, with the BNC-connected "current inputs" ideal for lower voltages and the "voltage inputs" capable of handling higher voltages. These signals can be conditioned to good amplitudes for data conversion, and then fed into the USRP. The fact that the hardware is all automated through simple python scripts simplifies this functionality into a very accessible platform. Individuals with no hardware experience can obtain high quality digitization of any real world signals they wish to study and analyze. For students and individuals with strong signals and programming backgrounds, this is an invaluable tool.

Continuing in the spirit of "enabling" other users, the data acquired from the deployment in a family home is available on a dedicated NESL NAS server. This data can be visualized, segmented, under sampled, and/or downloaded to a private machine. When combined with provided ground truth information, this data repository becomes the first of its kind to offer extremely high fidelity data on a real household load network. For the first time, students, academics, and professionals can analyze and experiment on data parsed for their specific needs without having to custom build, install, and operate a high voltage hardware sensing system. For teams needing a hardware system for their own data collection and analysis, our system is fully open source and available for recreation. The PowCapMobile board, firmware, Gnuradio code, and system design are all published and free to use; as all academic research should be. The purpose of this is to really open up the field to those who otherwise would not be able to lend their talents in signal analysis, machine learning, and general signal processing. Perhaps the next leap forward in load detection will come from an individual

who was simply looking at data provided on a NESL NAS server.

APPENDIX A

Firmware excerpts for PowCap MCU

A.1 commands.c

Includes SPI setup for PGA control and MUX signals for filter selection.

```
#include "commands.h"
#include <stdlib.h>
#include "SPI.h"



void cb_gainC1(char *param) {       //current1 gain
    char *s;
    int16_t val;
    s = param;
    val = strtol(param,&s,16);
    for (int i=0;i<val;i++) {
        LEDs_ToggleLEDs(LEDS_LED1);
    }
    SELECT1();
    SPI_MaterTransmit(val);
    UNSELECT1();

}

void cb_gainV1(char *param) {       //voltage1 gain
    char *s;
    int16_t val;
    s = param;
    val = strtol(param,&s,16);
    for (int i=0;i<val;i++) {
        LEDs_ToggleLEDs(LEDS_LED1);
    }
```

```
    SELECT0();
    SPI_MaterTransmit(val);
    UNSELECT0();

}

void cb_gainC2(char *param) {       //current2 gain
    char *s;
    int16_t val;
    s = param;
    val = strtol(param,&s,16);
    for (int i=0;i<val;i++) {
        LEDs_ToggleLEDs(LEDS_LED1);
    }
    SELECT3();
    SPI_MaterTransmit(val);
    UNSELECT3();

}

void cb_gainV2(char *param) {       //voltage2 gain
    char *s;
    int16_t val;
    s = param;
    val = strtol(param,&s,16);
    for (int i=0;i<val;i++) {
        LEDs_ToggleLEDs(LEDS_LED1);
    }
    SELECT2();
    SPI_MaterTransmit(val);
    UNSELECT2();

}

void cb_filterV1(char *param) {     //voltage BP filter MUX decision "n"
    char *s;
    int16_t val;
    s = param;
    val = strtol(param,&s,16);

    switch(val) {
    case 0:             //BP ON
        PORTF |=(1<<4);
        LEDs_TurnOnLEDs(LEDS_LED1);
```

```
        break;
    case 1:             //BP OFF
        PORTF &=~(1<<4);
        LEDs_TurnOffLEDs(LEDS_LED1);
        break;
    }
}

void cb_filterC1(char *param) {     //current1 BP filter MUX decision "n"
    char *s;
    int16_t val;
    s = param;
    val = strtol(param,&s,16);

    switch(val) {
    case 0:                 //BP OFF
        PORTF |=(1<<5);
        LEDs_TurnOnLEDs(LEDS_LED1);
        break;
    case 1:                 //BP ON
        PORTF &=~(1<<5);
        LEDs_TurnOffLEDs(LEDS_LED1);
        break;
    }
}

void cb_filterV2(char *param) {     //voltage BP filter MUX decision "n"
    char *s;
    int16_t val;
    s = param;
    val = strtol(param,&s,16);

    switch(val) {
    case 0:                 //BP ON
        PORTF |=(1<<6);
        LEDs_TurnOnLEDs(LEDS_LED1);
        break;
    case 1:                 //BP OFF
        PORTF &=~(1<<6);
        LEDs_TurnOffLEDs(LEDS_LED1);
        break;
    }
}
```

```
void cb_filterC2(char *param) {     //current BP filter MUX decision "n"
    char *s;
    int16_t val;
    s = param;
    val = strtol(param,&s,16);

    switch(val) {
    case 0:                 //BP OFF
        PORTF |=(1<<7);
        LEDs_TurnOnLEDs(LEDS_LED1);
        break;
    case 1:                 //BP ON
        PORTF &=~(1<<7);
        LEDs_TurnOffLEDs(LEDS_LED1);
        break;
    }
}




void cb_notchV1(char *param) {      //voltage1 60hz notch filter MUX decision "n"
    char *s;
    int16_t val;
    s = param;
    val = strtol(param,&s,16);

    switch(val) {
    case 0:
        PORTF |=(1<<0);
        LEDs_TurnOnLEDs(LEDS_LED1);
        break;
    case 1:
        PORTF &=~(1<<0);
        LEDs_TurnOffLEDs(LEDS_LED1);
        break;
    }
}


void cb_notchC1(char *param) {      //current1 60hz notch filter MUX decision
    char *s;
    int16_t val;
```

```
    s = param;
    val = strtol(param,&s,16);
    LEDs_ToggleLEDs(LEDS_LED1);
    switch(val) {
    case 0:
        PORTF |=(1<<1);
        LEDs_TurnOnLEDs(LEDS_LED1);
        break;
    case 1:
            LEDs_TurnOffLEDs(LEDS_LED1);
        PORTF &=~(1<<1);
        break;
    }
}


void cb_notchV2(char *param) {      //voltage2 60hz notch filter MUX decision "n"
    char *s;
    int16_t val;
    s = param;
    val = strtol(param,&s,16);

    switch(val) {
    case 0:
        PORTE |=(1<<6);
        LEDs_TurnOnLEDs(LEDS_LED1);
        break;
    case 1:
        PORTE &=~(1<<6);
        LEDs_TurnOffLEDs(LEDS_LED1);
        break;
    }
}


void cb_notchC2(char *param) {      //current2 60hz notch filter MUX decision
    char *s;
    int16_t val;
    s = param;
    val = strtol(param,&s,16);
    LEDs_ToggleLEDs(LEDS_LED1);
    switch(val) {
    case 0:
        PORTD |=(1<<7);
```

```
        LEDs_TurnOnLEDs(LEDS_LED1);
        break;
    case 1:
            LEDs_TurnOffLEDs(LEDS_LED1);
        PORTD &=~(1<<7);
        break;
    }
}
```

A.2 filters.c

Sets startup conditions for gain and MUX selections.

```
#include <avr/io.h>
#include "filters.h"


void filters_MasterInit(void) {

    //Turn off JTAG by switching JTD pin high (twice within 4 cycles)
    //-this gives us access to i/o pins PF4, PF5, PF6, and PF7 on the atmega16u4
    MCUCR |= (1<<7);
    MCUCR |= (1<<7);


    //sets voltage and current BP MUX to passthrough/OFF (PSU line)
    PORTF &= ~(1<<PF4);
    PORTF &= ~(1<<PF5);
    //PORTF |= (1<<PF4);
    //PORTF |= (1<<PF5);

    //sets voltage and current BP MUX to passthrough/OFF (alt line)
    PORTF &= ~(1<<PF6);
    PORTF &= ~(1<<PF7);

    //sets 60Hz Notch OFF for Voltage and Current lines (PSU input)
    //PORTF |= ((1<<PF0)|(1<<PF1));
    PORTF &= ~(1<<PF0);
    PORTF &= ~(1<<PF1);
```

```
    //sets 60Hz Notch OFF for V and I lines (alternate input)
    PORTE &= ~(1<<PE6);
    PORTD &= ~(1<<PD7);

/*
    //sets voltage MUX to 60Hz Notch passthrough  (001)
    PORTF &= ~(1<<PF4);
    PORTF &= ~(1<<PF5);
    PORTF |= (1<<PF6);
    //sets current MUX to 60Hz notch passthrough (000)
    PORTF &= ~(1<<PF7);
    PORTE &= ~(1<<PE6);
    PORTD &= ~(1<<PD7);
*/

}
```

A.3 FSM.c

```
#include <avr/io.h>
#include "FSM.h"
#include "commands.h"
#include <ctype.h>
void init_callbacks() {
    for(int i=0;i<NUMBER_CMDS;i++){
        cb_command[i].id = 0;
        cb_command[i].fxn = NULL;
    }
}

void setup_callbacks() {
    init_callbacks();
    cb_command[0].id = GAINC1;
    cb_command[0].fxn = cb_gainC1;
    cb_command[1].id = GAINV1;
    cb_command[1].fxn = cb_gainV1;
    cb_command[2].id = FILTERV1;
    cb_command[2].fxn = cb_filterV1;
    cb_command[3].id = FILTERC1;
    cb_command[3].fxn = cb_filterC1;
```

```
    cb_command[4].id = NOTCHV1;
    cb_command[4].fxn = cb_notchV1;
    cb_command[5].id = NOTCHC1;
    cb_command[5].fxn = cb_notchC1;
    cb_command[6].id = GAINC2;
    cb_command[6].fxn = cb_gainC2;
    cb_command[7].id = GAINV2;
    cb_command[7].fxn = cb_gainV2;
    cb_command[8].id = FILTERV2;
    cb_command[8].fxn = cb_filterV2;
    cb_command[9].id = FILTERC2;
    cb_command[9].fxn = cb_filterC2;
    cb_command[10].id = NOTCHV2;
    cb_command[10].fxn = cb_notchV2;
    cb_command[11].id = NOTCHC2;
    cb_command[11].fxn = cb_notchC2;

}


int16_t search_callback(COMMAND *cmd) {

    for (int i=0;i<NUMBER_CMDS;i++) {

        if (cmd->command == cb_command[i].id) {
            //LEDs_ToggleLEDs(LEDS_LED1);
            cb_command[i].fxn(cmd->param);
            return 0;
        }
    }
    return -1;
}

void init_FSM() {
    FSM_State = WAITING_CMD;
    init_command(&cmd);
}

void init_command(COMMAND *cmd) {
    cmd->command = 0;
    cmd->param[0] = '\0';
}

void set_command(COMMAND *cmd, int16_t c) {
```

```
    if(VALIDCMD(c)) {
        cmd->command = c;
        FSM_State = PARAM1;
    } else {

        init_FSM();
    }
}

void set_param0(COMMAND *cmd, int16_t p) {
    if(isxdigit(p)) {
        cmd->param[0] = (char)p;
        cmd->param[1] = '\0';
        FSM_State = PARAM2;
    } else {
        init_FSM();
    }
}

void set_param1(COMMAND *cmd, int16_t p) {
    if(isxdigit((int)p)) {
        cmd->param[1] = (char)p;
        cmd->param[2] = '\0';
        FSM_State = RUN;
    } else {
        init_FSM();
    }
}

void process_byte(int16_t rxbyte) {
    switch(FSM_State) {
        case WAITING_CMD:
            set_command(&cmd, rxbyte);
            break;
        case PARAM1:
            set_param0(&cmd, rxbyte);
            break;
        case PARAM2:
            set_param1(&cmd, rxbyte);
            break;
        case RUN:
            if(isspace(rxbyte)) {
                search_callback(&cmd);
                init_FSM();
```

```
                break;
            }
        default:
            init_FSM();
            break;
    }
}
```

A.4 SPI.c

Defines SPI control registers and sets MOSI and SCK pins.

```
#include <avr/io.h>
#include "SPI.h"

void SPI_MasterInit(void) {
     //Set Direction For MOSI and SCK PIN
    DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK)|(1<<DD_SS)|(1<<DD_SS2)|(1<<DD_SS3)|(1<<DD_SS4);
    //Set SPI Control Registers
    PORT_SPI = PORT_SPI | PSS;
    PORT_SPI = PORT_SPI | PSS2;
    PORT_SPI = PORT_SPI | PSS3;
    PORT_SPI = PORT_SPI | PSS4;
    SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0)|(1<<CPOL);
}

void SPI_MaterTransmit(char cData) {
    SPDR = cData;
    while(!(SPSR & (1<<SPIF)));
}
```

APPENDIX B

Python Source for PowCapMobile Data Acquisition

Modified Gnuradio code to take in 4 channels of data stored into two files. Files are segmented into custom lengths of time (default is 1 hour), this is to prevent generating very large data files that are more prone to corruption. Finally, the files are stored in locally specified locations, and recording stops once target directories no longer contain enough space for the next file (with some healthy buffer).

```
from gnuradio import gr
from gnuradio import uhd
from gnuradio import eng_notation
from gnuradio.eng_option import eng_option
from optparse import OptionParser
import datetime
import tempfile

import sys
import time
import numpy
import logging
import os
import configobj



fconfig = file("uhd_to_file.ini")
config = configobj.ConfigObj(infile = fconfig)

log = logging.getLogger("To File")

log.setLevel(logging.INFO)
log.addHandler(logging.StreamHandler(sys.stdout))
```

```
class GetData_top_block(gr.top_block):
    def __init__(self):
        gr.top_block.__init__(self)

        parser = OptionParser(option_class=eng_option)
        parser.add_option("-a", "--args", type="string", default="",
                          help="UHD device address args , [default=%default]")
        parser.add_option("", "--spec", type="string", default=None,
                          help="Subdevice of UHD device where appropriate")
        parser.add_option("-A", "--antenna", type="string", default=None,
                          help="select Rx Antenna where appropriate")
        parser.add_option("-s", "--samp-rate", type="eng_float", default=1e6,
                          help="set sample rate (bandwidth) [default=%default]")
        parser.add_option("-f", "--freq", type="eng_float", default=None,
                          help="set frequency to FREQ", metavar="FREQ")
        parser.add_option("-g", "--gain", type="eng_float", default=None,
                          help="set gain in dB (default is midpoint)")
        parser.add_option("", "--avg-alpha", type="eng_float", default=1e-1,
                          help="Set fftsink averaging factor, default=[%default]")
        parser.add_option ("", "--averaging", action="store_true", default=False,
                          help="Enable fftsink averaging, default=[%default]")
        parser.add_option("", "--ref-scale", type="eng_float", default=1.0,
                          help="Set dBFS=0dB input value, default=[%default]")
        parser.add_option("--fft-size", type="int", default=1024,
                          help="Set number of FFT bins [default=%default]")
        parser.add_option("--fft-rate", type="int", default=30,
                          help="Set FFT update rate, [default=%default]")
        parser.add_option("-t","--timeout", type="int", default=3600,
                          help="Set length of data files (seconds), [default=%default]")
        parser.add_option("-p","--filePath", type="string", default="/media/PowCapRawData/↩
            testData",
                          help="Set path where you want data saved, [default=%default]")
        (options, args) = parser.parse_args()
        if len(args) != 0:
            parser.print_help()
            sys.exit(1)
        self.options = options
        self.show_debug_info = True

        self.u = uhd.usrp_source(device_addr=options.args,
                stream_args=uhd.stream_args('fc32', channels=range(2) ) )

        self.filePath = options.filePath
        if self.filePath == "distributed":
```

```
            self.filePathID = 1
            print "Drive List (for distributed storage):"
            for drive in config["drives"]:
                print drive
        else:
            self.filePathID = 0

        self.timeout = options.timeout

    #Set both daughterboards to RX
    #Each daughterboard produces one complex binary IQ data stream.
        self.u.set_subdev_spec("A:AB B:AB")

        self.u.set_samp_rate(options.samp_rate)
        input_rate = self.u.get_samp_rate()

        # set initial values
        if options.gain is None:
            # if no gain was specified, use the mid-point in dB
            g = self.u.get_gain_range()
            options.gain = float(g.start()+g.stop())/2

        if options.freq is None:
            # if no freq was specified, use the mid-point
            r = self.u.get_freq_range()
            options.freq = float(r.start()+r.stop())/2

        #self.set_gain(options.gain)

        # Set the antenna
        if(options.antenna):
            self.u.set_antenna(options.antenna, 0)

        self.mainsdatas = []

        self.startStream()

    def getfname(self):
        date = datetime.datetime.now()

        #drives = ['Pandora', 'Box']
        drives = config["drives"]
        #print drives
        driveFound = False
```

```
        if self.filePathID:
            log.info("Running distributed file storage.")
            for drive in drives:
                self.filePath = "/media/" + drive + "/PowCapData"
                log.info("(distributed) Checking drive " + drive + ".")
                if not ( self.getspace() ):
                    log.info("(distributed) Enough space, saving to: " + drive + ".")
                    fname1 = self.filePath + "/mains_" + \
                        date.strftime("%y%m%d_%H%M%S")+ "-chan_%d" % 1 + ".dat"
                    fname2 = self.filePath + "/mains_" + \
                        date.strftime("%y%m%d_%H%M%S")+ "-chan_%d" % 2 + ".dat"
                    driveFound = True
                    break
#                elif x == len(drives):
            if not driveFound:
                log.warning("(distributed) Not enough space in ALL LISTED drives!")
                sys.exit(1)
        elif not ( self.getspace() ):
            fname1 = self.filePath + "/mains_" + \
                date.strftime("%y%m%d_%H%M%S")+ "-chan_%d" % 1 + ".dat"
            fname2 = self.filePath + "/mains_" + \
                date.strftime("%y%m%d_%H%M%S")+ "-chan_%d" % 2 + ".dat"

        log.debug("Write filename %s,%s" % (fname1,fname2))
        return (fname1,fname2)

    fname = property(getfname)

    def getspace(self):
        #returns the number of free Bytes on target drive
        s = os.statvfs(self.filePath)
        space = ( (s.f_bsize * s.f_bavail) / (1024*1024) )
        #switch out bavail for bfree to find free blocks for priviledged and unpriviledged↩
             users respectively
        log.info("Check space remaining in target drive: %dMB" % space)
        log.debug("preferred file system block size: %s" % str(s.f_bsize))
        log.debug("number of blocks available to non-super user: %s" % str(s.f_bavail))

        #required space (25MB/s required)
        spaceReq = 25 * self.timeout

        #Data at 1e6 sample rate consumes about 17MB/sec of space, this prevents ↩
            overloading the HD (with buffer of 25MB/sec)
```

```
        if space < ( spaceReq ):
            log.warning("Not enough space in target drive!")
            return 1
        else:
            log.info("Target drive has enough space: (%sMB required)" % spaceReq)
            return 0

    def startStream(self):
        fnames = self.fname
        for n in range(len(fnames)):
            mainsdata = gr.file_sink(gr.sizeof_gr_complex, fnames[n])
            self.mainsdatas.append(mainsdata)
            self.connect( (self.u, n), self.mainsdatas[n])

    def restart(self):
        #stops taking data and restarts the process
        for n in range(len(self.mainsdatas)):
            self.disconnect( (self.u, n), self.mainsdatas[n])

        self.mainsdatas = []


    def run(self, max_noutput_items=100000):
        log.info("Starting capture")
        self.start(max_noutput_items)
        try:
            log.info("Going to sleep for %d secs" % self.timeout)
            time.sleep(self.timeout)
        except KeyboardInterrupt:
            self.stop()
            sys.exit(1)
        log.info("Capture complete, stopping capture")
        self.stop()
        log.info("Wait for capture to return")
        self.wait()

        self.restart()

        self.startStream()

        self.run()


if __name__ == '__main__':
```

```
    try:
        GetData_top_block().run()
    except KeyboardInterrupt:
        log.info("Exiting")
```

APPENDIX C

MATLAB Source for PowCapMobile Data Conversion

The IQ converters in the USRP store their digitized outputs as complex binary values, where one channel is defined by the value of the real component and the other channel is defined by the imaginary component.

This simple script converts 32 bit complex IQ data into real values stored in a column vector.

```
%
% Copyright 2001 Free Software Foundation, Inc.
%
% This file is part of GNU Radio
%
% GNU Radio is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 3, or (at your option)
% any later version.
%
% GNU Radio is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with GNU Radio; see the file COPYING.  If not, write to
% the Free Software Foundation, Inc., 51 Franklin Street,
% Boston, MA 02110-1301, USA.
%

function v = read_complex_binary (filename, count)

  %% usage: read_complex_binary (filename, [count])
```

```
%%
%%  open filename and return the contents as a column vector,
%%  treating them as 32 bit complex numbers
%%

m = nargchk (1,2,nargin);
if (m)
  usage (m);
end

if (nargin < 2)
  count = Inf;
end

f = fopen (filename, 'rb');
if (f < 0)
  v = 0;
else
  t = fread (f, [2, count], 'float');
  fclose (f);
  v = t(1,:) + t(2,:)*i;
  [r, c] = size (v);
  v = reshape (v, c, r);
end
```

APPENDIX D

Python Source for PowCapMobile Data Conversion and Analysis

D.1 Analysis Toolset

Here is our Python toolset for analyzing our collected data.

```
from gnuradio import gr
import numpy as np
import matplotlib.pyplot as plt
import scipy
from StringIO import StringIO
import time
import logging
import os
import sys
from datetime import datetime , timedelta

log = logging.getLogger("To File")
log.setLevel(logging.INFO)
log.addHandler(logging.StreamHandler(sys.stdout))

class FindFile():
    def __init__(self, folderPath, date, time):



        self.path = folderPath
        self.date = date
        self.time = time
        self.datetime = datetime.strptime( self.date + self.time , '%y%m%d%H%M%S' )

        log.info("Searching for file containing data at...")
```

```
        log.info(self.datetime)
        log.info("...in path: "+self.path)

        for r,d,f in os.walk(self.path):

            file_list = f

            #we only want to look at given path and not any deeper dirs
            break

        if len(file_list) == 0:
            log.warning("No files found in given path!")
        else:
            log.info("%d files found in path" % len(file_list))
            #check for files with word "mains" in them
            file_list = [k for k in file_list if 'mains' in k]

            #check for files with required start date
            #file_list = [k for k in file_list if self.date in k]

            found_list = []

            file_list.sort()

            #latest files for quick look
            self.latestFiles = [ file_list[-4] , file_list[-3], file_list[-2] , file_list↩
                [-1] ]

            for files in file_list:
                datetime_temp = datetime.strptime( files[6:12] + files[13:19] , '%y%m%d%H%↩
                    M%S' )
                if self.datetime >= datetime_temp:
                    found_list.append(files)

            #print found_list
            file_list = found_list
            file_list.sort()

            if len(file_list) == 0:
                log.warning("No file found with requested date")
            else:
                self.foundPath = [ file_list[-2] , file_list[-1] ]
                log.info("Found data files: %s" % ' , ' .join(map(str,self.foundPath)))
                self.checkFile()
```

```
    #We check to see if our file is large enough to likely contain our data
    def checkFile(self):
        for chan in self.foundPath:
            s = os.stat( self.path + chan )
            size_bytes = s.st_size

            #define datetime of found file
            file_datetime = datetime.strptime( chan[6:12] + chan[13:19] , '%y%m%d%H%M%S' )
            #compute how far found file is from requested data
            diff_datetime = self.datetime - file_datetime

            #print "%d seconds" % diff_time

            #since data is recorded at 17MB/s: 1 second = 1024*1024*17 bytes of data
            if diff_datetime > timedelta ( seconds = (size_bytes/(1024*1024*17)) ):
                log.warning("FileCheck: %s may not contain requested data point" % chan↩
                    [20:26])
            else:
                log.info("FileCheck: %s is likely within range for requested data" % chan↩
                    [20:26])

class ParseData():
    def __init__(self, filePath, startSec, endSec, winSec, sample_rate = 1e6):

        plt.ion()
        self.fig = plt.figure()
        self.sub = self.fig.add_subplot(221)
        self.sub2 = self.fig.add_subplot(223)
        self.sub3 = self.fig.add_subplot(222)
        self.sub4 = self.fig.add_subplot(224)

        self.samp_rate = int(sample_rate/2)     #i and q channels are 1/2 complex rate
        self.num_channels = 2
        self.startpoint = startSec * self.samp_rate   #set startpoint in file
        self.endpoint = (endSec-startSec) * self.samp_rate       #set endpoint in file
        self.fileName = filePath                #set file path
        self.win_size = int( winSec * self.samp_rate )#set window size in file
        #note, self.samp_rate = 1 second of samples for each channel

        fileobj = open(self.fileName[0])
        fileobj2 = open(self.fileName[1])
        jcount = 0
        count = 0
```

```
        chunk_size = self.win_size*16 #reads 200K steps into file = 200ms

        fileobj.seek( int(self.startpoint * 16) )
        fileobj2.seek( int(self.startpoint * 16) )

        while count < self.endpoint/self.win_size:
            chunk = fileobj.read( chunk_size )
            chunk2 = fileobj2.read( chunk_size )
            Data = np.reshape( np.frombuffer(chunk, dtype=scipy.complex64), (-1, 1))#self.↩
                num_channels))
            Data2 = np.reshape( np.frombuffer(chunk2, dtype=scipy.complex64), (-1, 1))#↩
                self.num_channels))
            #print Data
            log.info("Plot %d of %d" % (count+1, int(self.endpoint/self.win_size)) )
            log.info("(traversing %d seconds in %f second windows)" % ( int(endSec-↩
                startSec), winSec ) )
            self.scope_data(Data, Data2)
            #need sleep time for plotting
            time.sleep(.05)

            count = count+1

        raw_input("press any key to exit")


    def chunk(self, fObject):
        while True:
            segment = fObject.read( int(self.win_size*16) )
            yield segment


    def scope_data(self, Data, Data2):
        dat1 = Data.real
        dat2 = Data.imag
        dat3 = Data2.real
        dat4 = Data2.imag
        if not hasattr(self, "run_once"):
            self.run_once = True
            x = np.arange( 0, len(dat1) )
            self.line = self.sub.plot(x, dat1)[0]
            self.line2 = self.sub2.plot(x, dat2)[0]
            self.line3 = self.sub3.plot(x, dat3)[0]
            self.line4 = self.sub4.plot(x, dat4)[0]
            self.sub2.set_title("Line1 Current (C4)")
```

```
            self.sub.set_title("Line1 Voltage (V3)")
            self.sub4.set_title("Line2 Current (C2)")
            self.sub3.set_title("Line2 Voltage (V1)")


        log.info( "c_binary length: %d" % len(Data) )
        log.info( "real bin length: %d" % len(dat1) )
        log.info( "imag bin length: %d" % len(dat2) )

        x = np.arange(0, len(Data))
        self.line.set_data(x, dat1 )
        self.line2.set_data(x, dat2 )
        self.line3.set_data(x, dat3 )
        self.line4.set_data(x, dat4 )

        self.fig.canvas.draw()
```

D.2 Test Bench

From this test bench, start time and end times and commands utilize the toolset to traverse files, provide some optional processing, and either plot them or store them in new files.

```
from PowCapTools import ParseData
from PowCapTools import FindFile

def main():
        fileName = '/home/henry/NeslStore/vikram/powcapData/Jason-Drive/↩
            mains_130313_030126-chan_1.dat'
        start_at_second = 1
        end_at_second = 20
        window_second = 200e-3
        sampRate = 1e6
        analysis = "plot"


        fileCat = '/NASPOW/Data1/EX2-PowCapData/'

        year = '13'
        month = '04'
        day = '20'
```

```
        datestr = year+month+day
        hour = '11'
        minute = '15'
        second = '05'
        timestr = hour+minute+second
        f_search =  FindFile(fileCat, datestr, timestr)

        fileCatPath = [ fileCat + f_search.foundPath[0], fileCat + f_search.foundPath[1] ]

        p = ParseData(fileCatPath, 0, 20, window_second, sampRate)



main()
```

References

[AE07] M.H. Albadi and EF El-Saadany. "Demand response in electricity markets: An overview." In *Power Engineering Society General Meeting, 2007. IEEE*, pp. 1–5. Ieee, 2007.

[AE08] MH Albadi and EF El-Saadany. "A summary of demand response in electricity markets." *Electric Power Systems Research*, **78**(11):1989–1996, 2008.

[AMN98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. "An optimal algorithm for approximate nearest neighbor searching fixed dimensions." *J. ACM*, **45**(6):891–923, November 1998.

[atm] "8-bit Microcontroller with 32K Bytes of Flash and USB Controller Datasheet." `http://www.atmel.com/Images/doc7766.pdf`.

[Cam] Dean Camera. "Lightweight USB Framework for AVRs (LUFA)." `http://www.fourwalledcubicle.com/LUFA.php`. Accessed: 2011-06-02.

[Cla] "Classification: Basic Concepts, Decision Trees, and Model Evaluation." `http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf`.

[GOB11] H. Gonçalves, A. Ocneanu, and M. Bergés. "Unsupervised disaggregation of appliances using aggregated consumption data." 2011.

[GRP10] Sidhant Gupta, Matthew S. Reynolds, and Shwetak N. Patel. "ElectriSense: single-point sensing using EMI for electrical event detection and classification in the home." In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Ubicomp '10, pp. 139–148, New York, NY, USA, 2010. ACM.

[Hut99] M.J. Hutzler. *Annual Energy Outlook 2000: With Projections Through 2020*. DIANE Publishing, 1999.

[Inc] Pearson Electronics Inc. "Wide-Band Current Transformers." `http://www.pearsonelectronics.com/news/178`. Accessed: 2011-05-10.

[Inc69] National Semiconductor Inc. "High Q Notch Filter." *National Semiconductor Linear Brief*, **5**:1–2, 1969.

[jam] "TED Jamming with PowCap." `http://www.youtube.com/watch?v=gulcci1qDS8`.

[Jon01] Bruce Jones. "More Filter Design on a Budget." *Texas Instruments Application Report*, **SLOA96**:2–11, 2001.

[JSB12] D. Jung, A. Savvides, and A. Bamis. "Tracking appliance usage information in residential settings using off-the-shelf low-frequency meters." In *Proceedings of the 49th Annual Design Automation Conference*, pp. 163–168. ACM, 2012.

[KBN10] J.Z. Kolter, S. Batra, and A.Y. Ng. "Energy disaggregation via discriminative sparse coding." In *Proc. Neural Information Processing Systems*, 2010.

[KNN] "The Nearest Neighbor Algorithm." `http://web.engr.oregonstate.edu/~tgd/classes/534/slides/part3.pdf`.

[lfr] "LFRX Daughterboard." `https://www.ettus.com/product/details/LFRX`.

[lmp] "LMP8100 Programmable Gain Amplifier Data Sheet." `http://www.ti.com/lit/ds/symlink/lmp8100.pdf`.

[Nai] "GENERATIVE AND DISCRIMINATIVE CLASSIFIERS: NAIVE BAYES AND LOGISTIC REGRESSION." `http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf`.

[ora] "Open source data visualization and analysis for novice and experts. Data mining through visual programming or Python scripting. Components for machine learning. Add-ons for bioinformatics and text mining. Packed with features for data analytics." `http://orange.biolab.si/`.

[PBA10] A. Pinomaa, H. Baumgartner, J. Ahola, and A. Kosonen. "Utilization of Software-Defined Radio in power line communication between motor and frequency converter." In *Power Line Communications and Its Applications (ISPLC), 2010 IEEE International Symposium on*, pp. 172–177, 2010.

[pea] "Pearson Current Monitor Model 411C Data Sheet." `http://www.pearsonelectronics.com/pdf/411C.pdf`.

[plu11] "Commercial Buildings Energy Consumption Survey.", Nov. 28 2011.

[powa] "High Power Selector Guide." `http://www.powerint.com/sites/default/files/product-docs/highpwr_selectorguide.pdf`.

[powb] "Low Power Selector Guide." `http://www.powerint.com/sites/default/files/product-docs/selectorguide.pdf`.

[PR] Paul and Robin. "Teensy USB Development Board." `http://www.pjrc.com/teensy/`. Accessed: 2011-06-21.

[Pre98] Abraham I. Pressman. *Switching Power Supply Design (2nd ed.)*. McGraw-Hill, New York, 1998.

[PRK07] S. Patel, T. Robertson, J. Kientz, M. Reynolds, and G. Abowd. "At the Flick of a Switch: Detecting and Classifying Unique Electrical Events on the Residential Power Line." In *Proceedings of Ubicomp 2007*, pp. 1 – 6, Aug. 2007.

[rad] "Radio Electronics: Resources and Analysis for Electronics Engineers." `www.radio-electronics.com`.

[sch] "LMP8100 Programmable Gain Amplifier Data Sheet." http://www.schaffner.com/components/en/_pdf/DatasheetIT_single_seriese35.pdf.

[SVM] "A User's Guide to Support Vector Machines." http://pyml.sourceforge.net/doc/howto.pdf.

[tam] "Three Flange Dual Primary 30VA PC Board Power Transformer Datasheet." http://www.mouser.com/ds/2/397/PL30-XX-130B-34069.pdf.

[Tec] Linear Technology. "Design Simulation and Device Models." http://www.linear.com/designtools/software/. Accessed: 2011-04-21.

[TED] "The Energy Detective (TED) Home Energy Meter)." http://www.theenergydetective.com/.

[ti] "Power Line Communication (PLC) Solutions." http://www.ti.com/lsds/ti/apps/smartgrid/plcmodem/overview.page?DCMP=plc&HQS=plc.

[Tsu99] K. Tsuda. "Subspace classifier in the Hilbert space." *Pattern recognition letters*, **20**(5):513–519, 1999.

[usr] "USRP1." https://www.ettus.com/product/details/USRPPKG.