SeeGreen: A Tool For Real-time Distributed Monitoring of Home Electricity Consumption

by

Joshua Daniel Kaufman

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the Requirements for the degrees of Bachelor of Science in Electrical Engineering and Master of Engineering in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology.

Massachusetts Institute of Technology
May 2001

Author_____
Department of Electrical Engineering and Computer Science
May 11, 2001

Certified
by_____
Michael Hawley
Thesis Supervisor

Accepted
by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

SeeGreen: A Tool For Real-time Distributed Monitoring of Home Electricity Consumption

by

Joshua Daniel Kaufman

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the Requirements for the degrees of Bachelor of Science in Electrical Engineering and Master of Engineering in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology.

Massachusetts Institute of Technology
May 2001

ABSTRACT

This thesis presents SeeGreen, a system designed to enable easy collection of distributed information about power usage (quantity and quality) in residential facilities. In Europe many appliances are emerging that incorporate power measuring devices for power regulation. However, products for retrofitting old appliances or recording measurements from other power consuming devices are not available.

Due to aging of appliances, varying consumer habits and appliance standby modes, estimating the power consumption behavior of a device is remarkably difficult. Products in "off" (or standby) modes consume an estimated ten percent of the electrical energy used in a house. With built in AC switches, SeeGreen has the capacity to minimize or even eliminate this unnecessary waste.

Perhaps the largest downfall of modern monitoring systems is a lack of real-time (immediate) feedback about an appliance's consumption. Feedback of this nature highly encourages change in consumer behavior. SeeGreen facilitates reduced electricity consumption by providing information to consumers and automatic control devices.

Acknowledgements

Several people were crucial in supporting my ideas, which stabilized my shifting interests long enough to actually complete a project. To Eugene, Rhonda, Michael, Laura, Emily, Andrea, Danstucky, and Chris Newell I owe a big THANK YOU.

Thanks to Mike Hawley for pushing me at the beginning of the project to do extensive research. This became the foundation of my introductory sections.

Most importantly, the one person without whom this thesis would not have been completed, is Tazeen Mahtab. Tazeen is responsible for developing the PC-side Java code that allows PC interaction with SeeGreen nodes. A big thanks to Tazeen for all of her help.

Contents

1 INTRODUCTION

I felt a need to fill a major void in my knowledge about my apartment and I realized that this void was commonplace. Where does the energy that comes into my home go? The monthly utility bill was motivating, but not informative.

I wanted to know how much power my *aged* refrigerator used in my *home*, because refrigerators change as they grow old and are subjected to different environments. But I didn't just want an average over the month. I wanted to know *when* it used the electricity and how that corresponded to my habits and activities. I didn't just want a profile; I wanted immediate feedback. I wanted something, telling me what was going on everywhere in my home at any given moment. Not mysterious monthly utility bills.

Thus, I set about devising a system that would *measure*, not calculate or speculate, the electrical power departing an outlet. I wanted the information available everywhere and it seemed logical to use the communication medium that I was already connected to, the power line. There was no need to mess around with HF radio links or special Ethernet lines. The information about every outlet would be available in the most logical place... at every outlet. The intent of this thesis is to demonstrate that distributed, networked power monitoring could be a useful asset in energy conservation.

Thesis Organization

The environment is in a sad state of affairs. If this is not already clear, it will be by the end of Chapter 2. Chapter 3 discusses some of the current solutions to the problem including methods of gathering information about electricity consumption in our homes. Then, in Chapter 4 and 5, the SeeGreen system is presented and evaluated. Progress and results of SeeGreen are described and displayed and finally,

in Chapter 6, the future of research needed in this area is spelled out for future endeavourers.

2 THE PROBLEM

Anyone who reads the news or surfs the web is already aware of the environmental problems that are currently facing humanity and their incurring compromise of our future. Naturally, much speculation has been generated about the causes and catalysts of the situation and they are numerous and diverse. In this thesis, I focus on one very specific issue that threatens the global environment: the pollution associated with the production and consumption of household electricity. This section will briefly elucidate the magnitude and nature of this specific problem.

2.1 Current electricity generation threatens everyone

The UN-affiliated Intergovernmental Panel on Climate Change (IPCC) unanimously agreed in January 2001 that new evidence shows more clearly than ever that temperature increases are caused mostly by pollution, not natural fluctuations. As a result of this, snow cover has decreased, the duration of river and lake ice is shorter, and heat-trapping CO_2 has more than tripled since 1750. The panel concluded that global temperatures were expected to rise between 1.4 and 5.8 degrees Celsius over the next century [ENS 2001] resulting in coastal flooding, droughts, and worldwide decreased agricultural production.

Contrary to common myth, the third world energy producers are not the key culprits. One quarter of the world's air pollution is generated in the United States, a country with less than 5% of the world's population. There are many activities that contribute to air pollution; however, **making electricity causes more air pollution than any other industry** [ENS 2001]. The United States consumes nearly three and a half trillion kilowatt-hours of electricity annually. The next largest consumer is China with around 1 trillion kilowatt-hours and more than four times the population [CIA 2000].

In 2000, coal-fired plants generated 52% of the United States electrical power [Sweet 2001]. The burning of coal is responsible for major players in the greenhouse gas saga i.e. CO_2, SO_2, and others. In fact, electric utilities in general account for over 70% of SO_2 emissions. In addition to greenhouse warming, SO_2 is also responsible for devastating acid rains in many parts of the world including northern U.S. and Canada.

Figure 2.1: Electricity plants produce over 70% of air pollution

2.2 *Household electricity use is significant*

Homes are major consumers of electricity. The Rocky Mountain Institute's (RMI) Senior Research Director, Bill Browning, says that buildings use more than a third of the total electricity produced in the United States. Much of this is residential.

In addition to lightening loads on the electrical grid, decreasing home electricity use can significantly alleviate financial burdens on many families. RMI estimates that the average family spends $1500/year on utility bills. More than half of this can be saved with simple changes in habit. Lighting alone consumes approximately one fourth of the electricity used in the United States. This is equivalent to 120 coal burning power plants. During the summer months, air conditioners alone account for 28% of the peak load [CA Energy 2000]. Our homes are significant contributors to our worldwide environmental concerns.

2.3 *People care*

The worldwide environmental movement is larger and growing faster than ever. As increasing numbers of people are directly impacted by environmental degradation,

the desire to reduce environmental degradation is heightened. The popularity of new companies such as Green Mountain Energy, which allow energy consumers to choose renewable energy for a slightly higher fee, stand as testimony to the increasing desires of the public to be ecologically conscious.

This growing population of concerned citizens wants to change their behavior to minimize their impact. Unfortunately, consumers of electricity (i.e. every home owner or tenant) have very little connection to knowledge about where their electricity is going. Presently, the only available information comes at the end of each month when the bill arrives.

There are many people in this country that crave to act in environmentally sound ways, but lack the necessary knowledge. They want immediate feedback on their actions.

2.4 Information is unavailable

Unfortunately, information and real-time feedback about the distribution of energy usage in a house is hard to find. The products that are available [Chapter 3] are designed with other goals in mind and are consequently very limited in function for this purpose.

Home electricity meters... do not encourage, facilitate, or even indicate behavior changes that could lead to reduced consumption.

Home electricity meters were obviously not designed to combat homeowner's ignorance about electricity consumption. They are difficult to read and sometimes even difficult to find. They do not provide any specific information about the distribution in time or space of electricity used in a house. They do not encourage, facilitate, or even indicate behavior changes that could lead to reduced consumption. Nor do they indicate appliances and apparati that use excessive electricity. In short,

they do nothing more than provide electric companies with a number with which to bill a homeowner.

One might argue that appliances come with the important energy usage information attached. However, this does not reflect the real nature of an appliance's electrical usage. The power consumption of an appliance is never constant; instead it varies tremendously with its surrounding conditions, usage practices, and age. Energy specifications accurate upon purchase become invalid after only a short period. Furthermore, many people move into situations where they use appliances that they did not purchase.

In addition to the shifting consumption habits of appliances, many appliances have unknown and unexpected consumption tendencies. Commonly referred to as *Energy Vampires*, most modern appliances use electricity even when they appear to be turned off [Figure 2.2]. For example, cable boxes consume almost as much electricity when turned off as when they are on. In fact, researchers at UC Berkeley and the Lawrence Berkeley National Laboratory found that, on average, 10 percent of total electrical consumption in a house comes from machines that are "turned off" [Jossi 2001]. This drain costs consumers about $80 a year. The nationwide price tag is roughly $3.5 billion (5-10 GW).

Overall, the net effect of these characteristics negates the usefulness of power specifications that come with new appliances. The way to obtain *useful* information about an appliance's energy consumption is to have real-time measuring and feedback, because only this can reflect *every* variation in trait and use.

2.5 *Information distribution is a better solution than control*

Many methods for controlling houses automatically have been proposed and a few have been implemented. These methods include thermostat adjustment schemes,

motion sensing technologies, and load balancing controllers. Unfortunately, these devices are not the complete solution. In much the same way that doing someone's homework does not help him or her learn the material, controlling someone's house to be more efficient does not educate him or her about energy conservation. The benefits of education cannot be understated. The lessons learned by having access to real-time energy usage feedback are lessons that can be extended to all facets of living a low-impact life. The goal is not only to decrease the financial burden on electricity consumers; the goal is also to create smarter, more energy-conscious consumers.

2.6 Summary

There is a significant environmental threat facing the world. A large portion of this threat comes from the production of household electricity. People who would like to address this problem need to understand the basics about their electrical consumption. I assert that the best solution is to provide them with tools to gain access to this information, so that they can address their own specific needs.

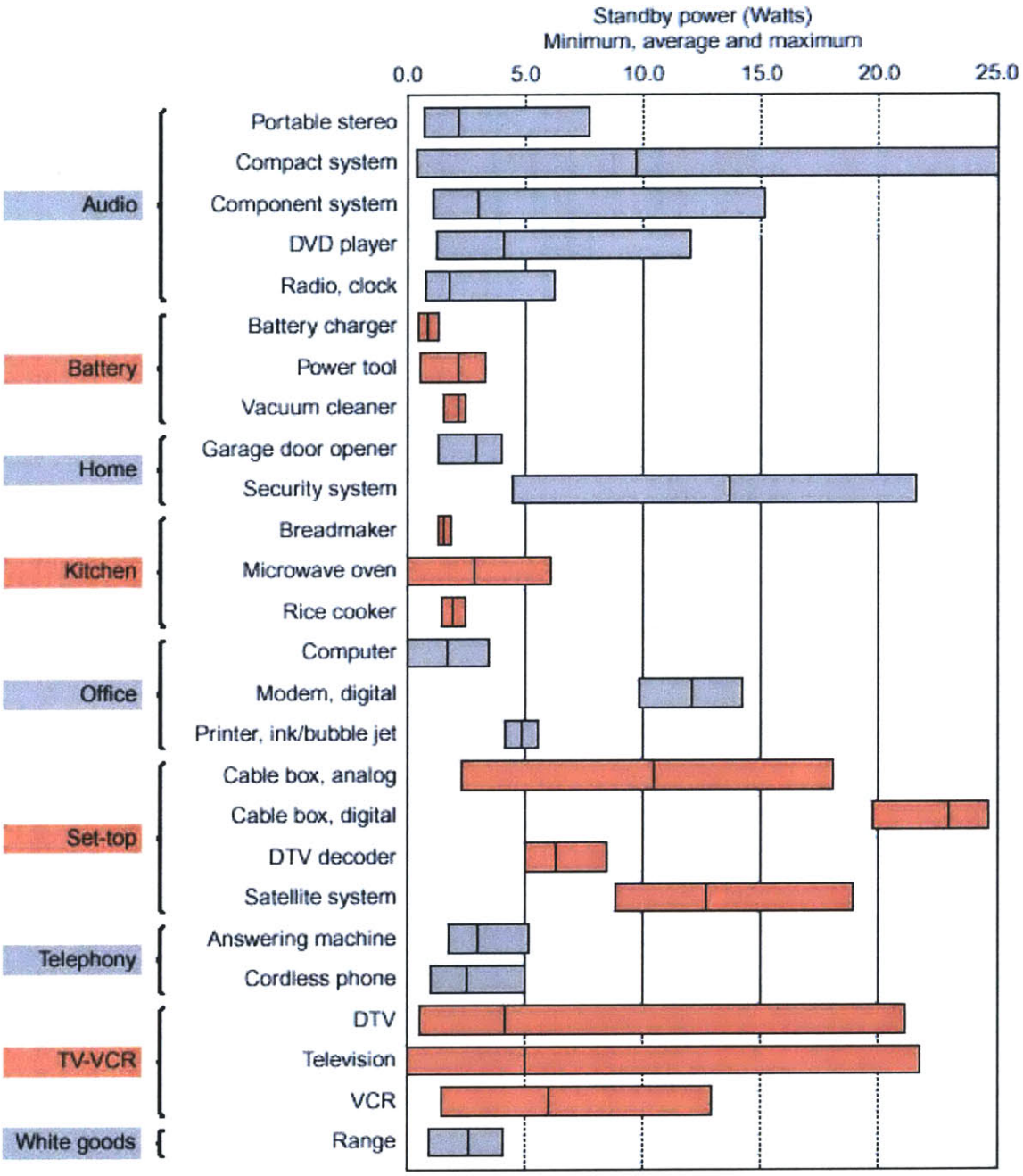


Figure 2.2: Measurement of standby power for appliances in a typical US home.

From Meier 1999

3 HISTORY OF SOLUTIONS

Like all others, this project sources ideas and implementations from many other projects that have come before it. Pieces of it have been successfully implemented by other scientists and engineers. I have divided the various related experiments and products into five categories: power line communication, home automation, collective metering, distributed metering, and standby power reduction.

3.1 Power Line Communication

Power line communication (PLC) is an important building block of the SeeGreen system. It allows data to be communicated around the house without necessitating additional communication infrastructure. This greatly reduces the demands placed on the user and eases installation. Some experiments with radio frequency (RF) links within a house have been successful; however, they do not easily transmit through some structures, require higher power consumption, and are often technically challenging to implement.

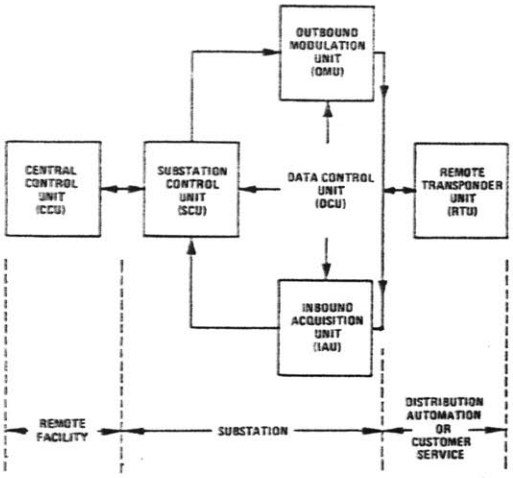


Figure 3.1: TWACS system – first steps in Power Line Communication

The first serious power line communication systems were discussed at the Control of Power Systems Conference & Exposition in 1980 in Piscataway, NJ [Orban 1980]. In 1982 IEEE released details of a PLC design [Figure 3.1] called Two-Way Automatic Communication for Distribution Systems (TWACS), which boasted a brave 32 baud [Mak 1982]. By 1984 improvements in this system had doubled the bit rate [Mak 1984]. At this point there was a boom of PLC systems and advancements in

reliability, noise susceptibility, and bit rate led to the development of the practical home automation system.

One of the first large scale power line communication systems was the X-10 product line which is still in use in many homes. Home Automation, Inc. and Smart Housekeeper, for example, sell a variety of X-10 related products. X-10 has a few significant drawbacks, most notably that it doesn't effectively deal with the highly noisy environment intrinsic to power lines. Although X-10 is still popular among amateur home electronic enthusiasts, engineers no longer commonly implement it.

There are many alternatives to X-10 available on the market either as complete packages or as chips requiring implementation. One such chip is the SGS-Thomson chip, ST7537, used by SeeGreen and advocated by High Tech Horizon (HTH) from Switzerland. HTH provides support, products, and services to facilitate the implementation of the ST7537. At 2400 baud, its performance is among the most reliable (but not fastest) of the power line communication schemes. Modern PLC systems achieve (in localized areas) speeds of more than 100 kbps and are predicted to eventually reach 1 Mbps [Strassberg 1996].

3.2 Home Automation

Home Automation products control various aspects of houses remotely or even automatically. The many products that exist for implementing home automation include light dimmers, web-thermostats, and remote electrical outlet switches. These are commonly implemented with power line communication schemes [3.1] for information distribution. The most common goals are control over lights and appliances and to enable security applications. Home automation products for distributed electricity monitoring are not yet available on the market [3.4].

There are a couple noteworthy vendors of home automation products *similar* to SeeGreen. JG Home Automation offers home automation products including motion detectors to turn your lights off when no one is in the room. The company claims several energy saving features, but doesn't provide tools for monitoring. Metricom sells PLC chip devices including outlets with the receive-only PLC-2 chip for controlling relays and dimmers. These products allow remote outlet shutdown (an analogous feature to the SeeGreen shutdown mode).

3.3 Collective Metering

Collective metering is essentially all that is available now in the way of household electricity monitoring. These are products that measure the complete (collective) electricity usage of a house at the juncture between the house and the external power line. Sometimes these data can be used to analyze specific loads by looking at signature current patterns.

"It is difficult to conserve electricity when you can't measure it easily. By getting immediate feedback on your electrical consumption, you can take measures to lower your electric expenditures."
-E-mon website

Clearly, it's important to be able to measure, record, and even analyze the complete electricity usage in a house. Power companies need to be able to justify their monthly bills. Therefore, it should come as no surprise that a plethora of monitoring systems exist that record measurements at the interface between the house and the power grid. Energy Monitoring Technologies, Inc. (E-Mon) makes such meters of all varieties and even offers analysis software for examining the electricity consumption in a house as a function of time.

In 1996, the Electric Power Research Institute (EPRI) took these measurements one-step further by developing the Non-Intrusive Appliance Load- Monitoring System (NIALMS). This device was able to perform rudimentary analysis of specific appliance loads from the collective home electricity usage [EPRI 1996]. This information was used to identify faulty appliances or abnormal energy usage patterns. In 1997, seven utility companies implemented EPRI's NIALMS in trial setups [EPRI 1997].

More recently, Metricom developed a product for networking home electricity meters with PLC technology and a radio link. They tested their equipment with a project at Fort Irwin (http://www.pnl.gov/forscom/fortirwin.html). Metricom has also researched identification of loads from induction and distortion signatures to provide some details about the distributed energy consumption within the house.

A consulting company, Silicon Energy, takes a slightly different approach to providing useful electricity monitoring services. They manually collect data from meters and energy bills and then perform extensive analysis of energy usage. The company is more service oriented than product oriented. The target is large business looking to reduce their "third largest expenditure."

Using a variety of measuring tools, some groups have been able to compile energy data for various appliances. The Rocky Mountain Institute has a great collection of information about household appliances from its years of research.

3.4 Distributed Metering

The ultimate goal of this project is to create a system of distributed, real-time metering. This type of device measures power at every juncture in the house where electricity is consumed for a detailed view of the processes that use electricity. The first mention of creating such a device may be from John M Hunt (Oregon Graduate

Center) in his paper titled "Electrical Energy Monitoring and Control System for the Home" published in 1986 in the IEEE Transactions on Consumer Electronics. His paper is a report on a primitive device, which fits into every outlet to be monitored and reports the power consumption on the power lines [Figure 3.2]. He created a second device for reading and displaying that information on an LCD display. I have been unable to find products on the market that operate similarly to JM Hunt's device or even provide similar information.

An environmentally conscious company called Seattle City Lights offers a service called "Advanced Metering" where they will collect information about the distribution of electrical consumption in a business by sending in a representative with a probe. The collected data returns as a time average, which does not provide real time feedback.

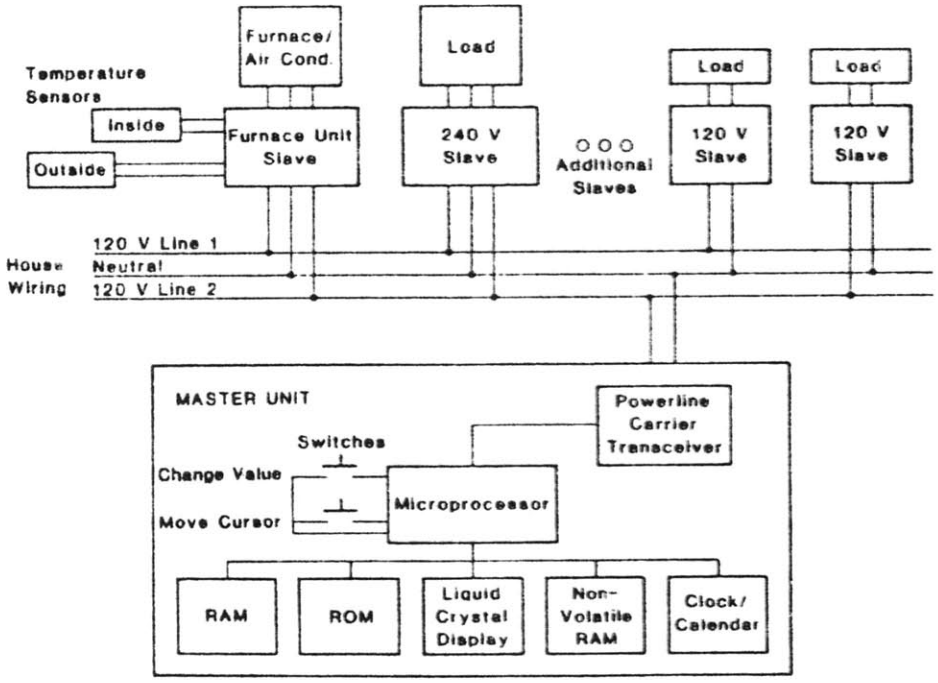


Figure 3.2: JM Hunt's system outline.
From Hunt 1986

A spin-off from the Italian appliance company, Merloni, called WR@P is developing the products most closely related to the devices discussed in this thesis. WR@P began developing power quantity and quality metering systems that were embedded directly into appliances. The power information is communicated over the house power lines using a similar protocol to the one employed in this system. Understanding the difficulty of embedding their product into all new appliances and the need for information from older appliances, WR@P has begun developing a product intended to retrofit appliances with this capability. The new product (still under development as of May 2001) is manifested in a form very similar to SeeGreen's box that fits into an outlet and has the capacity for an appliance to plug into it directly.

3.5 Standby Power Reduction

One important feature of SeeGreen is the ability to disable outlets supplying power to appliances in standby mode [4.1.5]. The problem of standby power loss, being of significant proportions [2.4], has already been addressed in many other ways; however, these solutions have been disappointingly ineffectual.

San Jose-based Power Integrations (www.powerint.com) offers a chip that can reduce the standby drain of appliances by 75 to 90 percent. The company has sold more than 500 million of these devices called TOPSwitch or TinySwitch to manufacturers like General Electric, Sony, and Panasonic. Unfortunately, however, they are being significantly underutilized because most companies are choosing to ignore the issue. They are in such a hurry to get products to the market that they use cheap power supplies with high standby losses [Jossi 2001].

4 SEEGREEN SOLUTION

The SeeGreen solution puts a node at each electrical outlet that collects data on electrical use at that point and puts that data back onto the power line network. Each node has a serial link that a PC can tap into to receive all the data from all the nodes on the power line. Additionally, a relay at each outlet allows for remote switching of the outlet from the PC.

Figure 4.1: Conceptual view of SeeGreen in deployment.

SeeGreen employs between 1 and 16 nodes per network, each connected by the same power line circuit. Implementation and activation of a node involves nothing more complex than plugging it in [Figure 4.2]. Calibration and startup happen automatically upon power up.

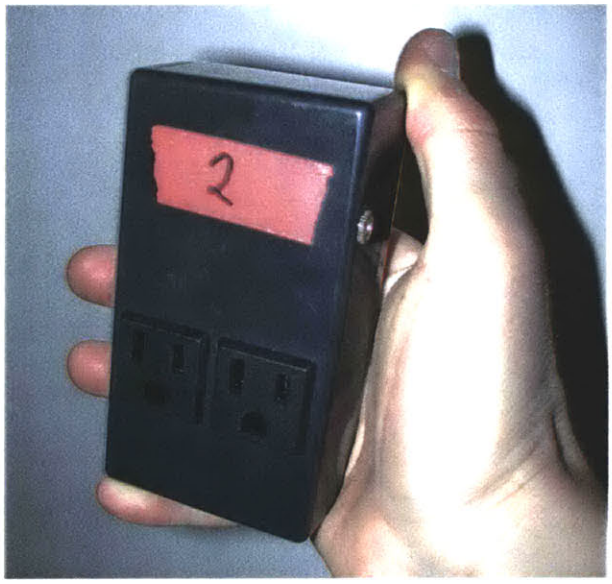


Figure 4.2: The SeeGreen Outlet Box

4.1 *Hardware*

The box for each node is a modified plug case from Enclosures of America, Inc (EAI) made from black ABS plastic. The backside of each box has a built-in male AC plug with a grounding pin. The front side of the boxes has been carefully machined to provide a snug fit for the outlet inserts [Figure 4.3].

The fundamental operations of the device are to measure the current and the voltage on the power line many times during a handful of cycles. A microprocessor (PIC16F877) controls all of the processes (analog sensors, power switches, modem, etc) and performs all of the calculations. Earlier models using a smaller processor (PIC16F84) ran out of memory due to the demands of floating point calculations.

4.1.1 Evolution of thoughts about hardware

The immediate need, as discussed in previous sections, was for a compact, sealed and safe device to fit between the outlet and the appliance's power cord. The device was to be non-intrusive and as small as possible to minimize clumsiness and required space. Finally, the packaging should not consist of more than one component (i.e. no additional plugs, connectors, etc.).

In addition to this "outlet box," the original design included a second device used strictly as a link between the PC and the information on the power line. This second device would have only a coupling to the power line and a

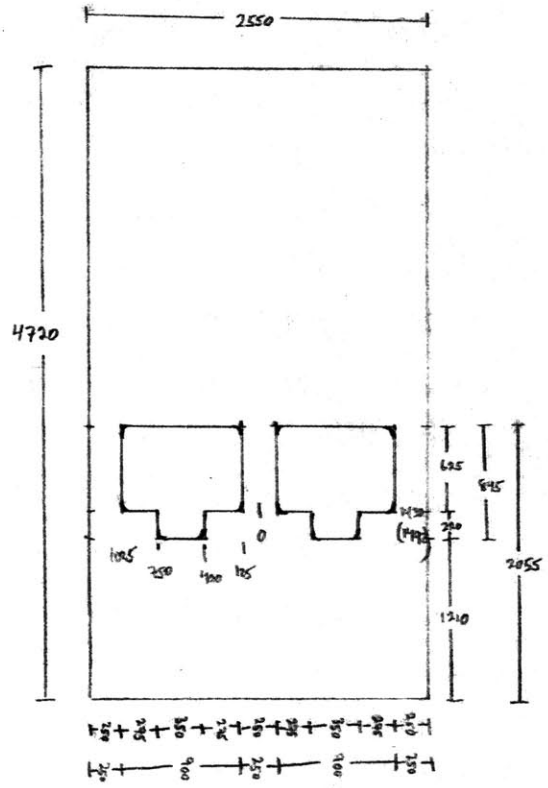


Figure 4.3: Blueprints for outlet box machining.

translator to send the appropriate serial information to the computer.

As the project progressed it became apparent that two separate devices would be more expensive, clumsy, and inappropriate. The two separate devices were merged into one box, which with only a couple of extra components, performed all of the functions of two boxes, including power monitoring, power line communication, and PC linking. The new device greatly simplified design and application; however, it still required a four-layer printed circuit board.

4.1.2 Power Supply

SeeGreen's power supply was designed to be robust and energy-efficient. The power comes directly from the 120AC line that the box is plugged into for measurement and communication. Limiting the power consumption of the SeeGreen outlet box was a top priority, lest the system defeat its purpose by consuming more power than it helps save. Furthermore, lower power demand allows a physically smaller power supply. Experiments with one SeeGreen node plugged into another indicate that the power consumption of one node is less than one Watt.

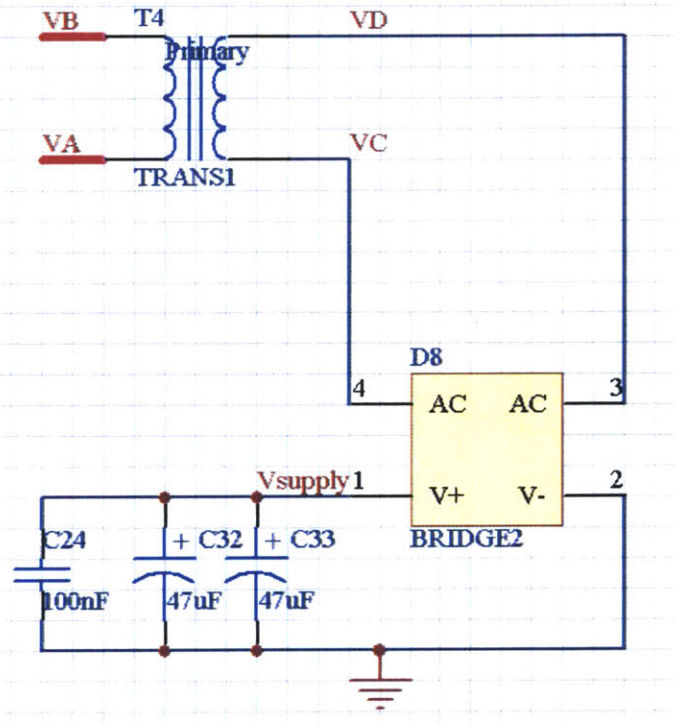


Figure 4.4: Power Transformer and Bridge Circuit for

The voltage reduction and maintenance electronics consist of several stages. A 10:1 2VA step-down transformer provides isolation from the mains and a reasonable working voltage for digital electronics (Figure 4.4). The power line modem amplifier needs a 12V supply and the PIC, signal amplifiers and filters need a 5V supply. Thus, SeeGreen incorporates two step-down switching supplies to efficiently (better than 90%) step down the full-bridge-rectified alternating voltage to the needed supply rails (Figure 4.5). Furthermore, the signal amplifiers need a very accurate intermediate voltage around which to base their signals. The 5V line is tight (less than one percent ripple); however, for even better accuracy a 3.3V linear regulator was installed to feed the amplifier baseline voltage.

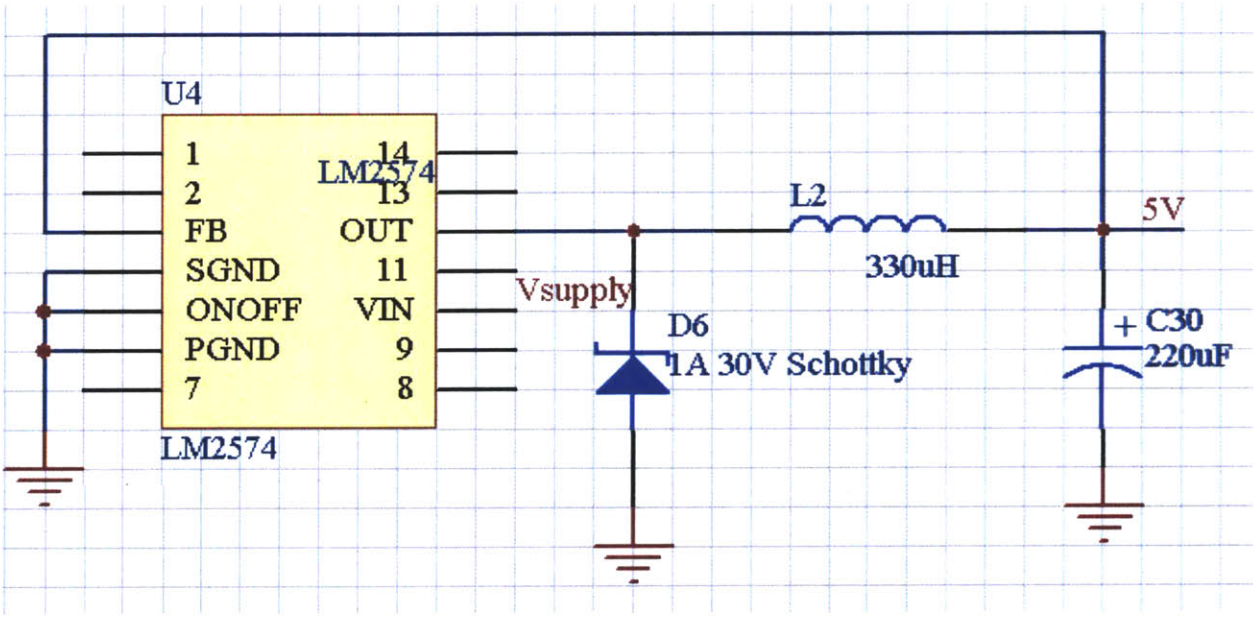


Figure 4.5: Switching 5V step-down regulator (also similar to 12V step-down)

4.1.3 Power Measurement

Measuring the power on an AC line is more difficult than on a DC line because the timing of current and voltage measurements is critical. The current and voltage on an AC line are constantly oscillating and a current measurement at one time

multiplied by a voltage measurement at a later time gives no information about the power. The measurements must be taken as closely to one another as possible. However, the story is even more complex than that. Inductive and capacitative loads can cause uneven distribution of power throughout a cycle. A purely resistive load can be calculated from a single measurement, but once any dynamics are introduced the power must be measured many times over at least one cycle. SeeGreen samples the voltage and current several hundred times over several cycles. The samples are averaged to find a mean power exiting the device. Additionally, the squares of the voltage and current are summed over the samples to facilitate finding a voltage and current root-mean-square (V_{rms} and I_{rms}) for later use calculating the power factor.

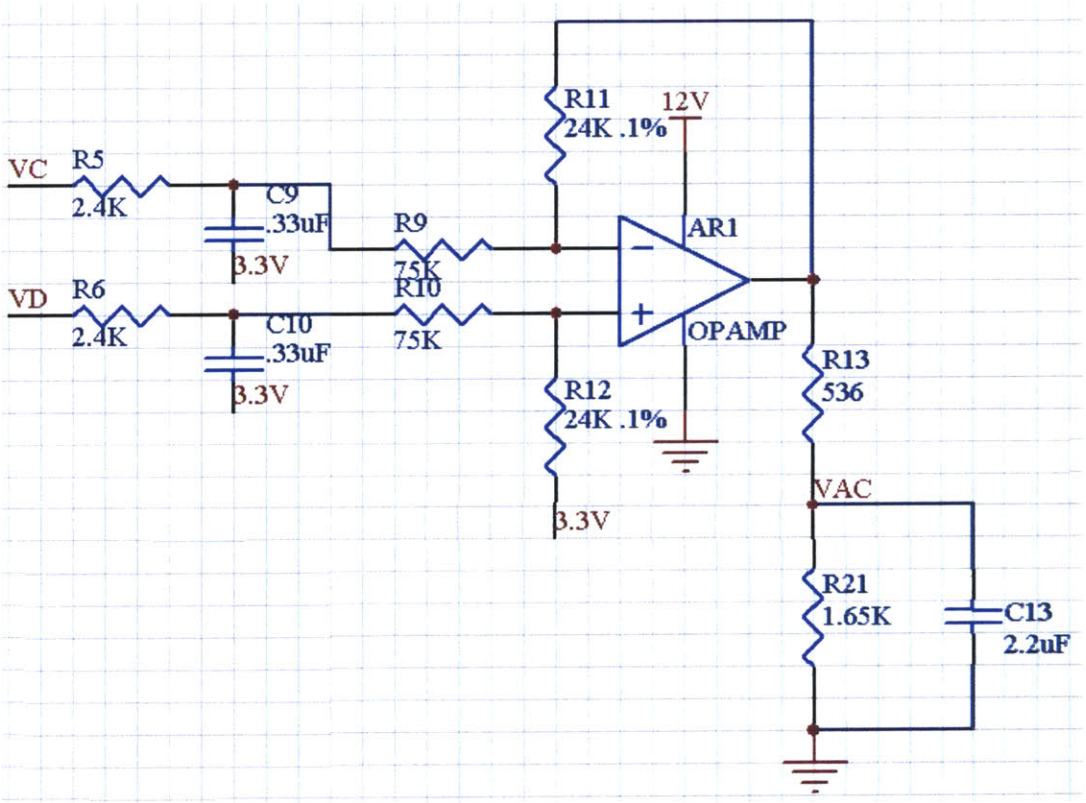


Figure 4.6: Differential amplifier and filters for power line voltage measurements.

Analog to Digital Converters (ADCs) on the PIC is a multistage process (Figure 4.6). The voltage signal begins on the low side of the power transformer (the safe side!). This American Zettler (44135) transformer has a $12V_{rms}$ output for a $120V_{rms}$ input. From there, the signal is low-pass filtered, scaled to the appropriate amplitude for the PIC and then filtered again. The filters began as tight filters with breakpoints around 100Hz. This setup proved to overly suppress the important 60Hz, and I was forced to back the breakpoint out slightly beyond 200Hz.

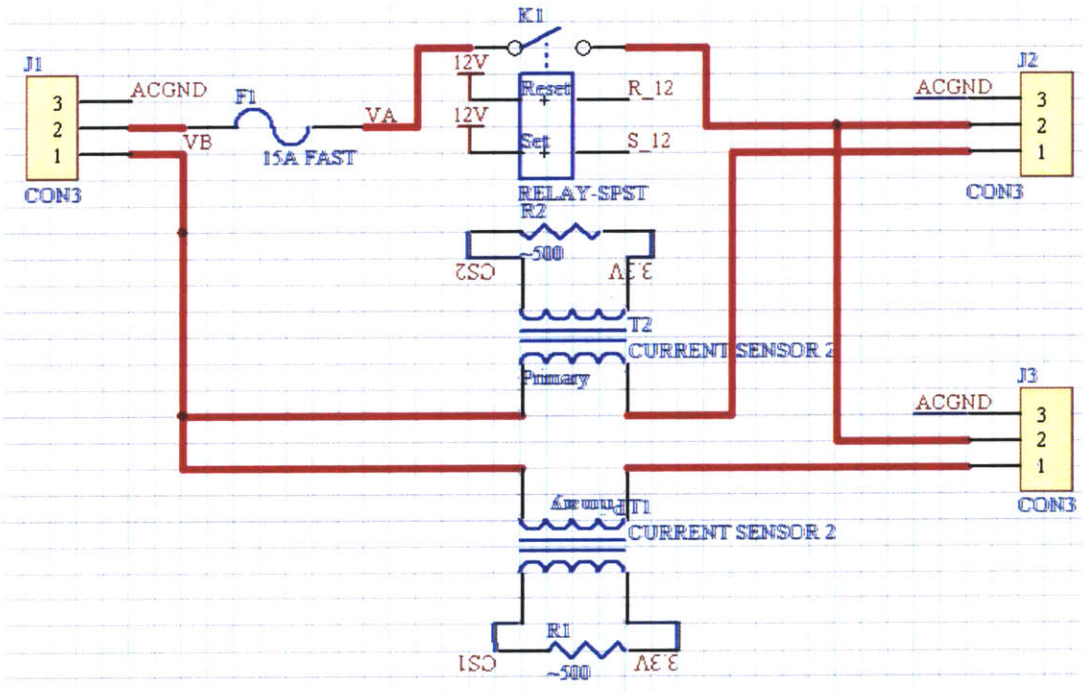


Figure 4.7: Current sensing transformers isolate sensitive measuring circuits from the power line and the latching relay provides maintenance free switching.

4.1.4 Power Line Modem and coupling

SeeGreen's power line communications are centered around an IC produced by SGS-Thomson, the ST7537. This chip accepts data from a serial line (RS-232) and converts it to power line compatible data modulated at 132.45kHz with a complete bit rate of up to 2400 baud. In addition the chip provides the inverse function, putting data onto an RS-232 line from the 132.45kHz signal on the power line. It also provides a carrier detect, essential for effective networking schemes.

The ST7537 drives a complimentary Darlington transistor pair for amplification and current sourcing (Figure 4.8). The power dissipation of these devices is sufficient to warrant a TO-92 package rather than the preferred surface mount package. These transistors drive a small transformer (T-933) that is coupled to the power line with a 200V 100nF capacitor.

4.1.5 Latching Relay and Remote Shutdown

SeeGreen's outlet box includes a switch to enable remote shutdown. For a target capacity of 12A at 120VAC, the only sensible option was a mechanical relay. Other switches were considered and other relays (i.e. solid state) were tested; however, for the price and size a mechanical relay was most suited to the application.

The first complete version of SeeGreen (version 2.0) incorporated a typical 15A non-latching surface mount relay. However, in a blatant violation of my low-energy consumption standards, the relay required more than half a Watt just to maintain state! This relay was replaced in SeeGreen 2.1 by a latching relay with permanent magnets to maintain its state without power input [Figure 4.7]. Unfortunately, the highest current rating for a surface mount latching relay available on the market is 10A and thus, the target of 12A was reduced to 10A and fuses were adjusted accordingly.

The boxes are rated to 1200W and have been tested to 1000W. Although they are fused at 10A and should not fail dangerously, I do not recommend exceeding the power limits.

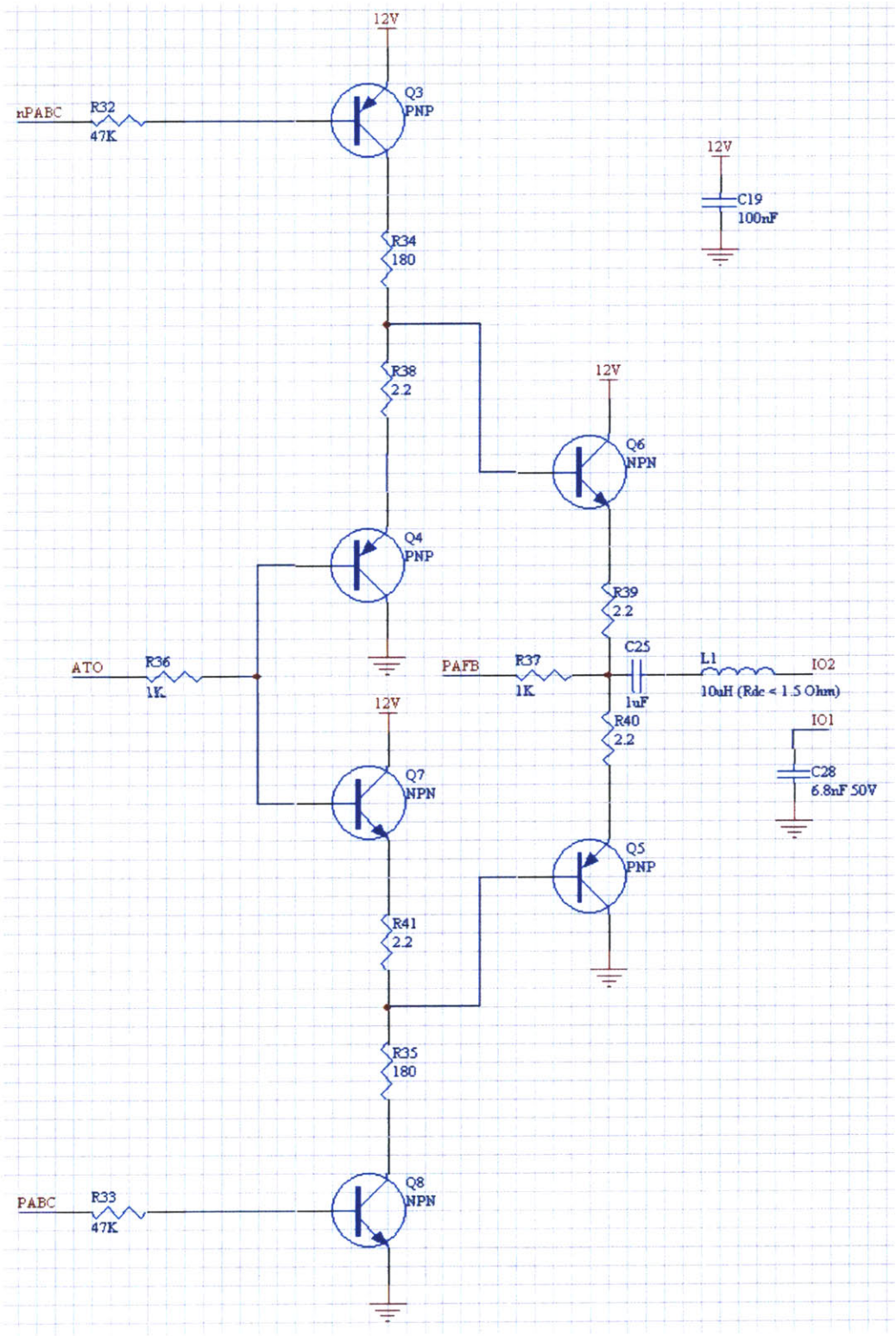


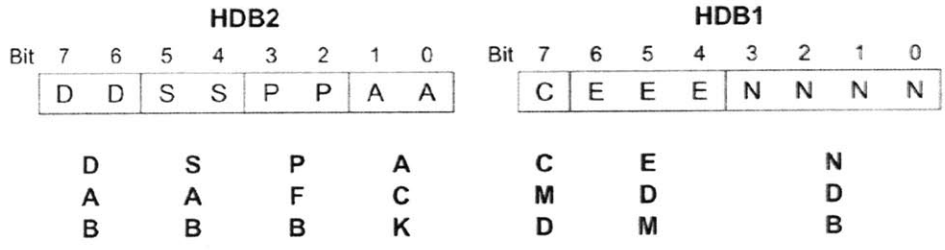
Figure 4.8: Modem amplifier circuit for driving power line coupling transformer.

4.2 *Software*

The software was designed to be simple and easy-to-use. The actual amount of information passed around by SeeGreen nodes is small in comparison with many networking schemes. Due to the inherent noisiness of the power lines, the data transfer schemes needed to be robust in the face of errors. Once the data has been communicated and stored, the PC software is responsible for real-time graphing while the MATLAB scripts were designed for post-processing analysis.

4.2.1 Protocol and Error Checking

For communication over the power lines a simple and reliable communication protocol was needed. The modem chip came with a recommended protocol, which was simple and easily implemented in the space requirements of the PIC microprocessor. In my application, the protocol, Scaleable Node Address Protocol (SNAP), consists of a sync byte, two header bytes, the data bytes, and two error-checking bytes. The sync byte is a specially chosen pattern (Ob01010100) that signifies the beginning of a new packet to the receiver. The header bytes



DAB = Number of **D**estination **A**ddress **B**ytes
SAB = Number of **S**ource **A**ddress **B**ytes
PFB = Number of **P**rotocol specific **F**lag **B**ytes
ACK = **ACK**/NAK bits
CMD = Co**M**man**D** mode bit
EDM = Error **D**etection **M**ethod
NDB = Number of **D**ata **B**ytes

Figure 4.9: SNAP header byte protocol

contain all of the information about the number of data bytes, the error-checking method, requested responses, and other settable parameters (Figure 4.9).
Good error-checking algorithms are essential for successful power line communication. Power lines, even in residential environments, are very noisy due to the many various loads attached to a single circuit. Additionally, copper wires strung out around the frame of a house should be expected to pick up a considerable level of radiated noise from any nearby transmitters. To combat these issues, there are two methods of preserving signal integrity. The hardware line of defense involves generating a strong signal on the power line to maintain a presence well above the noise level. The trade off to this solution is increased power consumption (classic power vs. signal integrity tradeoff). In this application (power monitoring for conservation), I naturally felt inclined to maintain a high priority of minimizing the power consumption of the transmitter. The second approach to dealing with the noise issue is in software with solid error-checking algorithms. This implementation incorporates two bytes for error checking, which if evenly distributed, correspond to a 1 in 2^16 or 99.9985% chance of getting it right. That is to say, that if the message is received incorrectly due to noise, there is a very small chance, .0015%, of thinking that it is the right message to receive. These error-checking bytes could have been increased to 4 bytes for even better assuredness of proper message reception; however, the tradeoff is two fold. First, larger error-checking schemes require more processing power and secondly, more bytes transmitted means a higher chance of making a mistake. In this application, more than 2 error-checking bytes would have been counterproductive.

Many error-checking algorithms generate error-checking codes by taking the last couple of bytes of a sum of the data in the message. With large message packets, this can be a very simple and moderately effective method of calculating error-checking bytes. However, this method has serious drawbacks for small packets such as the packets sent between SeeGreen nodes. For example if only 9 data bytes are being sent, then the maximum sum is 2304 meaning that the error code has a 1 in 2304

probably of giving a false positive. In response to this dilemma, a method called CRC-16 has been developed for generating error-checking codes that utilize the entire 16 bits of space with only a small message. The technique involves integer division by a large constant and the application of it can be seen in both the PIC and the Java code (Appendix B).

4.2.2 PIC Code

On the most basic level, the PIC code consists of a simple loop between making power measurements and then sending that information to the modem chip for transmitting. The details off these two operations involve considerably more complicated operations.

SeeGreen measures power in two different ways, both of which are important when considering power usage. The first measurement is real or actual power. This is typically what people think of when discussing the power consumption of appliances. However, for some loads, in particular heavily inductive loads such as motors, a second type of power, reactive power, is also very important. A large inductor (assume no resistance for now) connected to a 60 Hertz power line will move a lot of current (V/(L*2*■*f) peak); however, it won't (in this ideal case) consume any real power. The reactive power is high; the real power is low. This is an important consideration because transmission lines are clearly not perfect conductors and sloshing electricity back and forth through this inductor causes significant loss in the transmission lines. The second measurement recorded by SeeGreen, Powerfactor, is a ratio of the power used by the appliance to the power sloshed around from the factory. A purely resistive load will have a power factor of one.

To measure the power from a non-constant voltage source, the voltage and current must be sampled many times over several cycles. One sample in SeeGreen's sampling code consists of four measurements in rapid succession, a voltage

measurement, two current measurements at different sensitivities, followed by a final voltage measurement. Since they are constantly changing, it is important to measure the current and the voltage at the same time. Since this is not feasible with the PIC architecture, I make a close approximation by sampling the voltage before and after the current is sampled and then averaging the values. For every sample the voltage is multiplied by the current and added to a sum later to be divided by the number of samples to find the mean. In addition, the current and voltage are independently squared and added to sums because the power factor can be found by the following equation:

$$PF = \langle P \rangle / (\langle I^2 \rangle * \langle V^2 \rangle) \quad (4.1)$$

where <x> denotes the mean of all samples x.

Transmission of this data involves the standard complications of multiple nodes utilizing a common busy (and noisy) communication medium. The software supports up to 16 nodes on the same house power line, which requires nearly 20% of the bandwidth. Because of the high usage of the medium, random transmissions are likely to collide resulting in lost or garbled data. To avoid this problem, SeeGreen incorporates a system similar to an informal round robin. When a node has collected its power data (collection happens approximately every 2 seconds) it goes onto the line to wait its turn to transmit. If another node is transmitting (say node n) then as soon as it is finished node n+1 is given the opportunity to transmit. If node n+1 happens to not be online, then any listening nodes wait a specified amount of time, and then give node n+2 the opportunity to transmit. When a listening node's turn arrives, it transmits and returns to collect more power data.

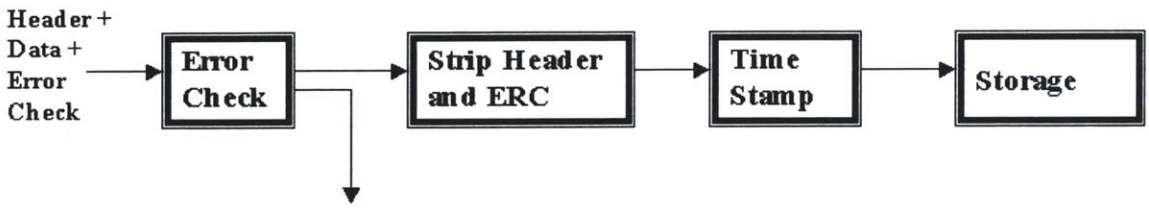


Figure 4.10: Dataflow for incoming packets from the power line

4.2.3 Java Scripts – PC Side Software

The Java code for this project allows a personal computer to interface with the data streaming in from the wall outlet. The incoming data (RS-232 format) can be stored and graphed in real time with a variety of options (Figure 4.11). Up to four graphs can be displayed at a time and each has independently adjustable time scales and fully automatic ranging. The graphs update automatically when new data arrives from the outlet.

The control of the software is divided among several threads. SerialThread monitors the serial line and watches for anatomically correct packets. When a new packet arrives from the outlet, the Java code first checks its CRC-16 value to ensure that no errors have occurred during transmission (Figure 4.10).

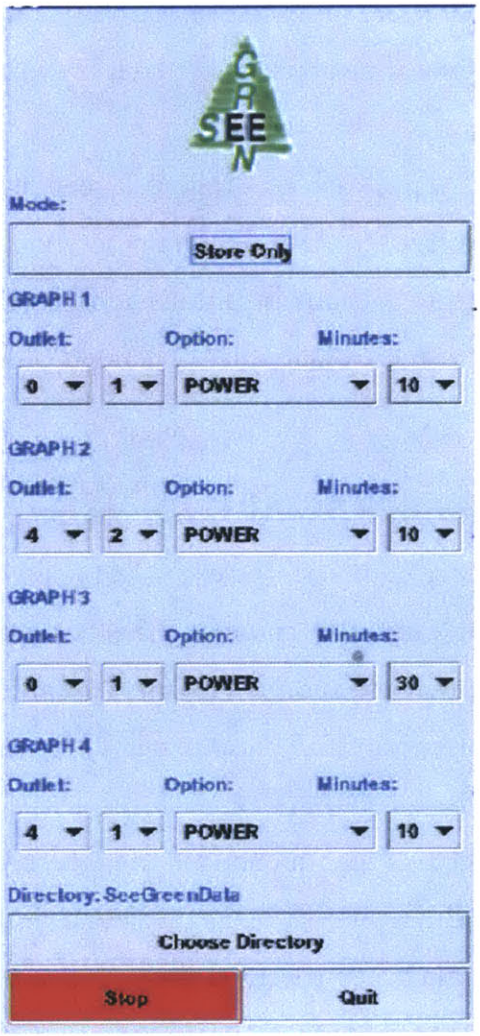


Figure 4.11: SeeGreen Control Panel

CRCError is an abstract class that contains static methods for computing the checksum.

After the packet has been validated, all headers and error-checking codes are striped from it and the data bytes are converted (i.e. scaled) to produce actual power and powerfactor data. A time stamp is assigned to each data block. TimeConverter is a procedural abstraction containing static methods for converting between time represented as a Calendar object and time as the number of seconds from Jan 1 00:00:00 2001. This data is stored in both binary and text files with a separate directory for each day and a separate file in that directory for each outlet. This format proved most useful for later browsing and post-processing with MATLAB scripts.

ControlListener, of the listener class, watches the buttons for events generated on the control panel and then updates the panels and graphs accordingly. Graph is a data type that contains information for the graph of a particular outlet. Graph gets its data from the binary files generated by SerialThread.

4.2.4 MATLAB

MATLAB scripts were used to post-process much of the data to look for trends and interesting anomalies. These scripts read data from the binary files generated by the Java code and then perform graphing and statistical functions on that data with a few user inputs. The source has been included in Appendix B.

One of the functions of the MATLAB scripts is to calculate the total energy involved with a process. This involves a simple integrated extrapolation of the available power data over a specified time. This energy estimation can be useful for evaluating the efficiency of a process (5.3).

5 Testing and Results

Already SeeGreen has proven to be a useful, informative, and entertaining device. Several extended home experiments have been performed with remarkable results. This section will describe some of the experiments and provide examples of some of the results.

5.1 Description of the Field Tests

All of the field tests were conducted in homes in the Cambridge, MA area. With the notable exception of the HVAC system, the differences between the loads placed on appliances in different living environments is insignificant since all homes are maintained at nearly the same conditions (i.e. comfortable for humans). Therefore, the results of these tests can be easily extrapolated to very different geographic locations on the assumption that the living spaces are similar.

The two main locations of testing were an apartment near Central Square and a dormitory on the MIT campus. Technically four people live in the central square apartment; however the number of occupants at any given time is typically closer to six. The appliances and usage patterns are typical of a Boston apartment setting. The dormitory room is a mid-sized double with a small refrigerator, stereo, and some lighting.

The experiments were conducted mostly at times when people occupied the buildings. As is typically in American homes, the occupants depart the house for most of the day and operate within the house during the mornings and evenings. These are the times when the most useful data could be collected.

5.2 *Individual processes*

One of the classic comparisons between new and old technologies is the fluorescent versus incandescent debate. It has been well documented that compact fluorescent lights (CFLs) use less energy per unit light produced than incandescent bulbs. It is less clear how their specific numbers compare and how their consumption profiles look over time.

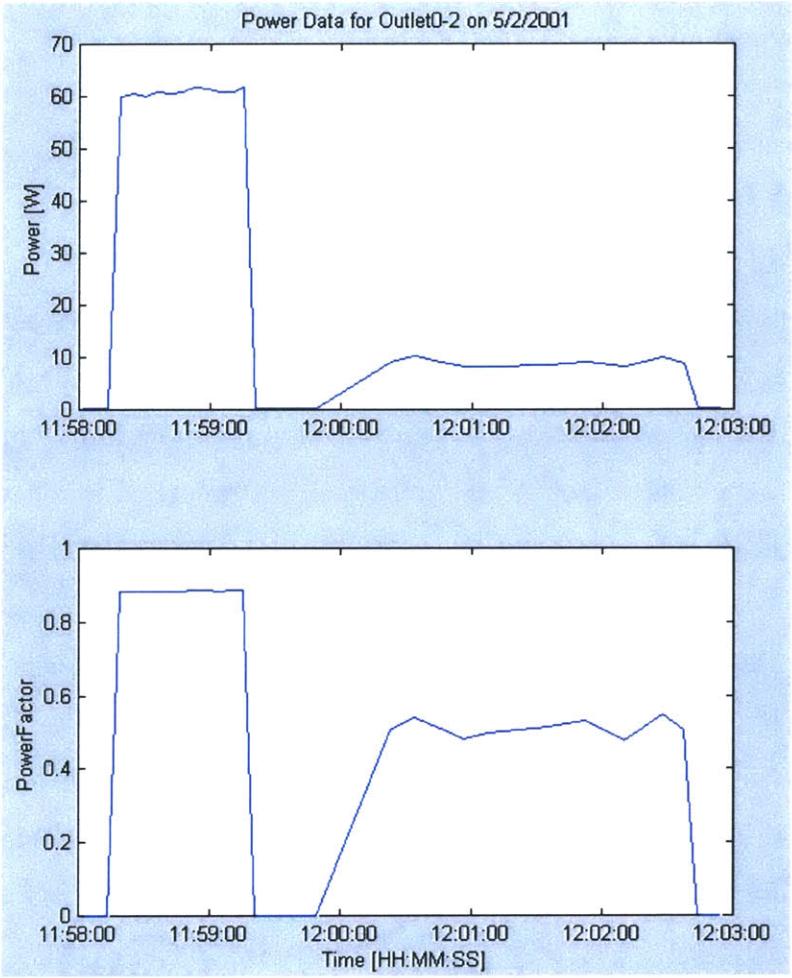


Figure 5.1: Incandescent light and fluorescent light comparison

Using the SeeGreen tools, I was able to easily monitor these two bulbs for a few minutes each to make this comparison [Figure 5.1]. The bulbs used were a 60W incandescent bulb producing 840 lumens and a CFL producing 900 lumens. The incandescent bulb, as expected, behaved like a simple resistor, jumping to a full 60W power immediately upon turn on and returning to zero on turnoff. The power factor was basically unity; there may have been some noise due to the long lamp wires that brought down the power factor

slightly lower than unity. The fluorescent bulb behaved much differently. Upon turn-on there was a short warm up period, which consisted of higher than average power draw. Additionally, I noticed a smaller surge in power on shutdown. It is common knowledge that power cycling incandescent bulbs shortens their life expectancy; however, one should not frequently power cycle fluorescent for an entirely different reason: increased electricity consumption.

The powerfactor of the CFL was much lower than the incandescent, a distinct disadvantage for the utility company. In fact, the CFL generated enough noise that it was occasionally difficult for the SeeGreen monitoring tools to communicate without error.

The startup patterns of a personal computer are well reflected in its electricity consumption [Figure 5.2]. The computer being analyzed is a Pentium III with a 400Mhz processor. As can almost be read directly from the power graph: the computer was turned on at 13:16:35, system check was completed just before 13:17:30 at which time Windows began to boot. At around 13:18:00, the processor prompted for a username and password. The lull around this time is the

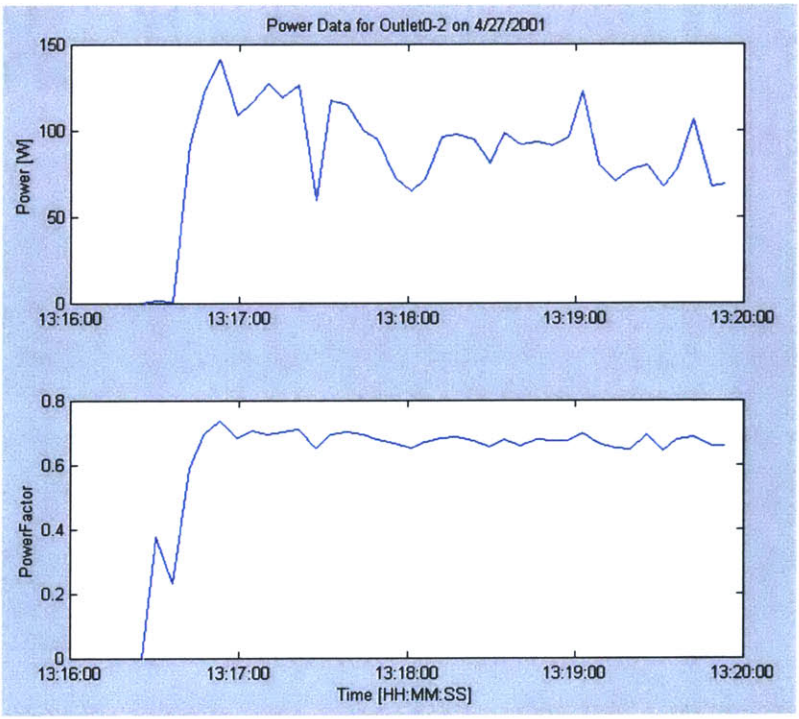


Figure 5.2: Pentium computer startup sequence

delay while I typed on the keyboard. After the password was entered, Windows finished booting. I believe the two spikes in power during the final booting stages are related to the loading of large pictures.

The most exciting of the individual processes monitored during the course of this project involved a soldering iron from the lab (Figure 5.3). The iron operated at full power (60W) until it was actually hotter than the user-specified temperature. Then it used zero power for a while until it had cooled back to slightly below the specified temperature. The feedback loop involved with regulating the soldering iron temperature is obvious from the graph. At around 13:01, I turned the knob on the iron to increase its temperature. The feedback loop behavior is again exhibited with a slightly higher equilibrium power for the slightly higher maintenance temperature.

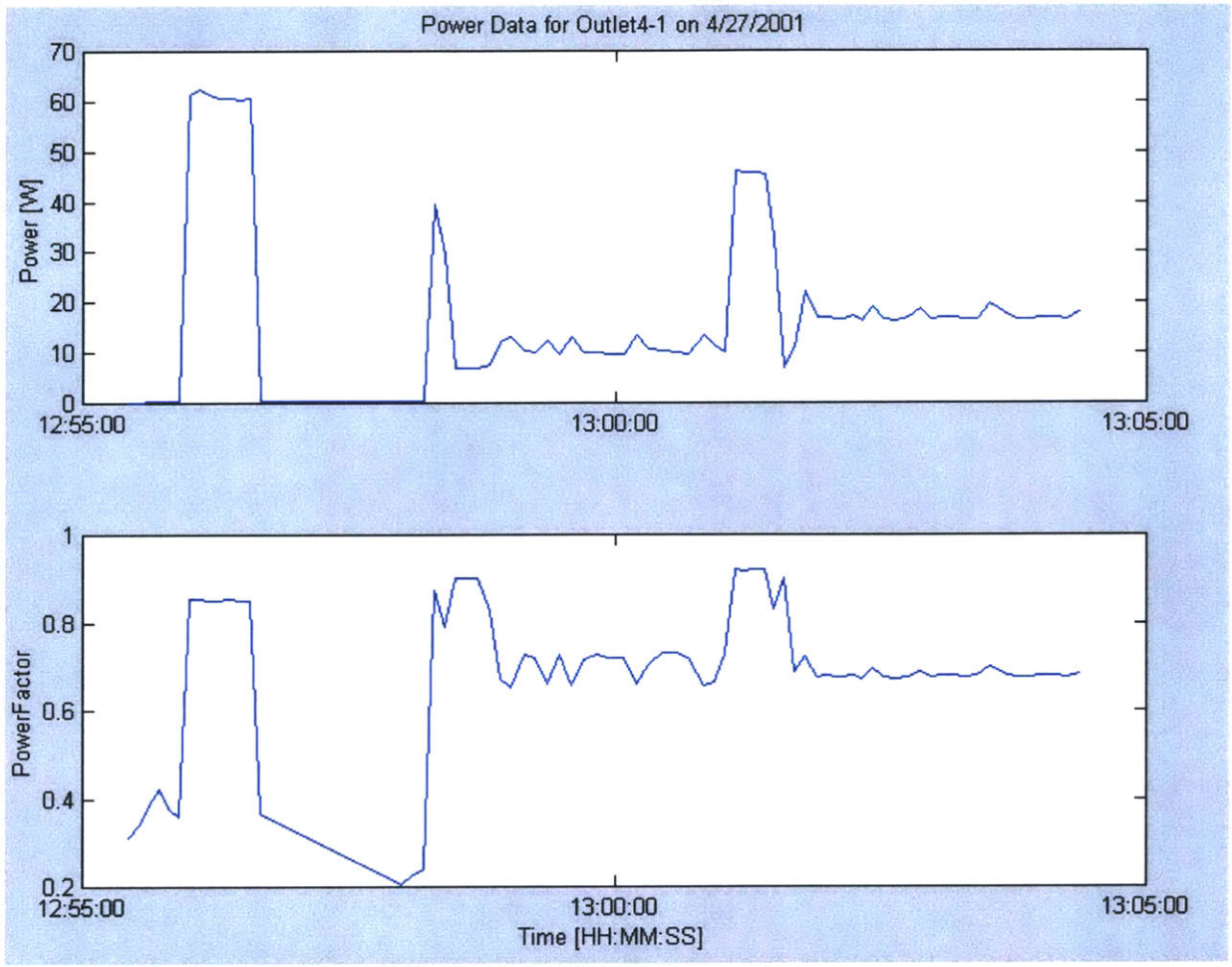


Figure 5.3: Soldering iron startup and equilibrium maintenance

5.3 *Efficiency Calculations*

In addition to comparative studies of analogous processes, tools such as SeeGreen make it possible to make absolute statements about efficiency. Consider the example of making a cup of hot water. Many appliances can perform this task, but how efficiently can this be done?

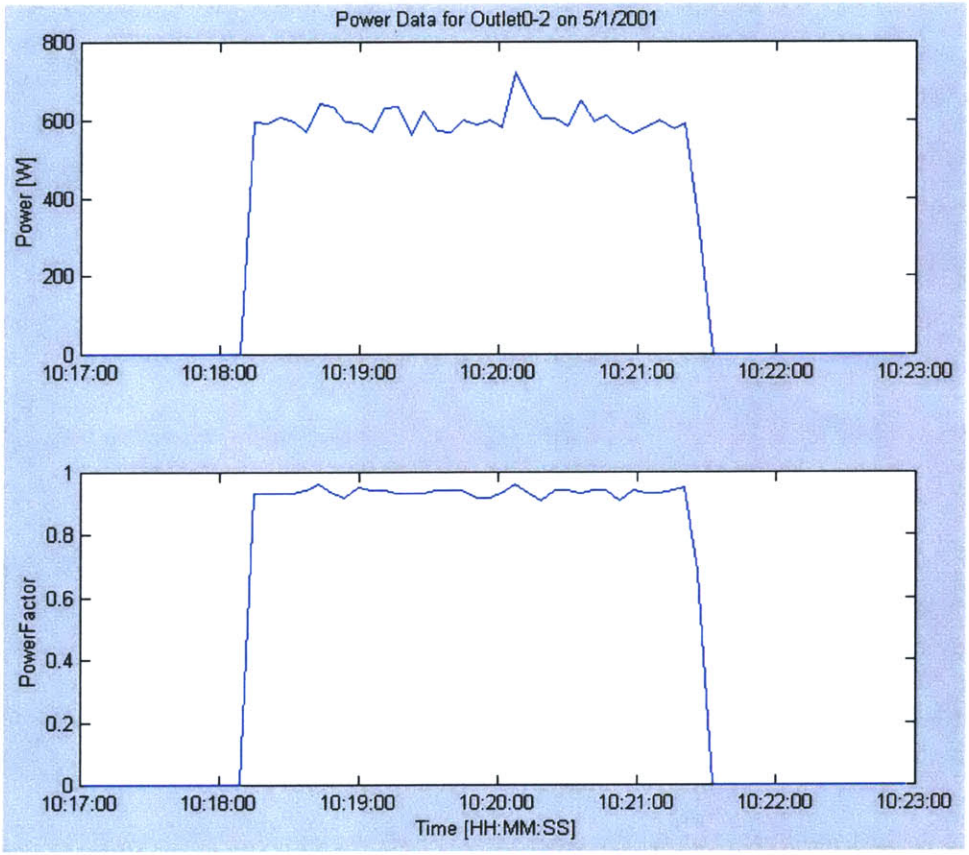


Figure 5.4: This coffee percolator was about 87% efficient at heating water.

In the single experiment performed in this area, 350 grams (slightly more than a cup) of water was raised 72°C (10°C to 82°C) by a Black & Decker Brew N Go, a personal coffee percolator. SeeGreen nodes monitored the power consumption (Figure 5.4) and with the help of a MATLAB script for post-processing the power was integrated

to energy. This process required 120,000 Joules or about 33 Watt-Hours of energy. Raising the temperature of this amount of water by 72°C requires an energy input of 105,000 Joules. The overall efficiency of the percolator: 87%.

Even though it is not always possible to calculate the minimum energy needed for a process, it can still be interesting to measure the energy used. For example, how much electrical energy is consumed by the toaster at my apartment to make a single slice of toast? How much did that slice cost?

The toaster ran at around 600W for the entire toasting time (Figure 5.5); it was not a very clever toaster. It is easy to tell by the powerfactor, that the only thing in the toaster is a resistive element. The integrating MATLAB script, applied to this data set concludes that 81000 Joules (23 WHr) were required to make my toast. At the going rate for electricity (around 12 cents/kWh), this piece of toast cost me almost 3 mils (i.e. $0.003). I could save more than a mil by putting in two pieces of toast at a time! Electricity is much too inexpensive.

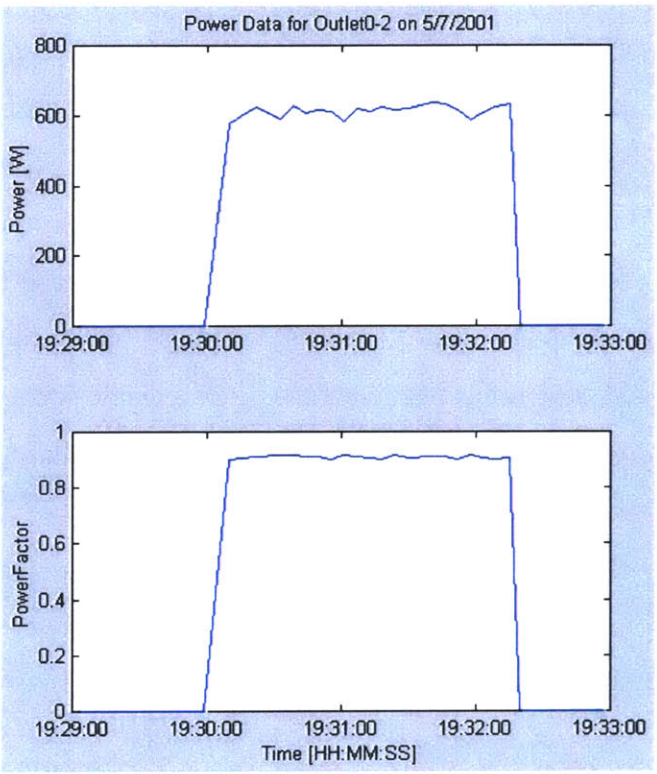


Figure 5.5: An old toaster requires 23WHr to

5.4 *Extended Processes*

Monitoring an appliance for an extended period of time can teach us many things. Often a bizarre or unexpected appliance behavior only appears at odd hours of the day

or during obscure scenarios. To achieve a sense of the long-term electrical consumption behavior of a full size refrigerator, I left a SeeGreen monitor on it for a couple of days. The results were interesting and almost useful.

The refrigerator under surveillance is a Frigidaire 125A with an overhead freezer. The appliance has been in service for at least 4 years. Several important observations were made about the nature of this appliance. First, it has regular compressor cycles that consume around 130W when running (Figure 5.6). The mean draw from the outlet is around 60W, which gives a duty cycle of a little less than 50%. The beginning of each compressor startup forced a current surge to initiate the compressor motor spinning. This surge typically drew up to 500W. The most unexpected event was a daily defrosting cycle typically around 5pm during which time the electrical draw went to around 500W for more than 20 minutes. This event was always followed by a compressor run of more than twice normal length during which time the refrigerator recovered its previously cool temperature. The powerfactor during all of the compressor cycles was nearly perfect, once the motor was started. I believe the refrigerator may compensate some for its inductive motor load. However, during the defrost cycle the power factor was slightly lower indicating a noisier heating system.

From these extended usage charts, it becomes easy to calculate expenditures for an appliance (Figure 5.7). For this example, I have employed a MATLAB script approximately a discrete time integral. At a rate of $0.12/kWh, this refrigerator costs its owners around $0.20/day.

5.5 Networking

One final aspect of the system that required testing was the ability of multiple nodes to network without quarrelling. The network algorithms were engineered to support up to sixteen nodes on one power line system. Because of time and labor constraints,

only four nodes were constructed. These four nodes existed on the same network, each communicating in turn without any problems. Because of this ability, it was possible to have several outlets reporting back to the PC at one time (Figure 5.8).

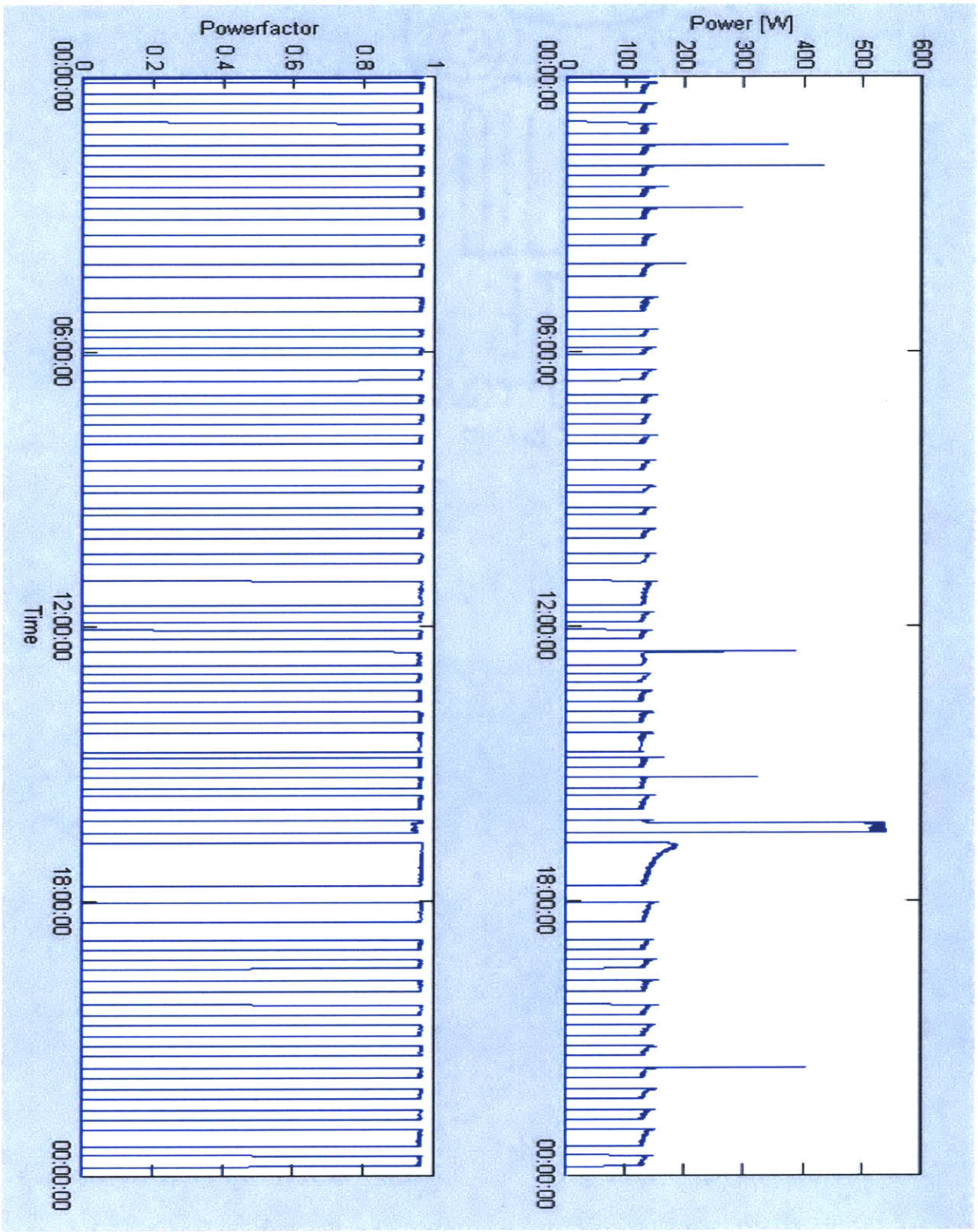


Figure 5.6: 24 Hour Refrigerator Surveillance (Power and Powerfactor versus Time)

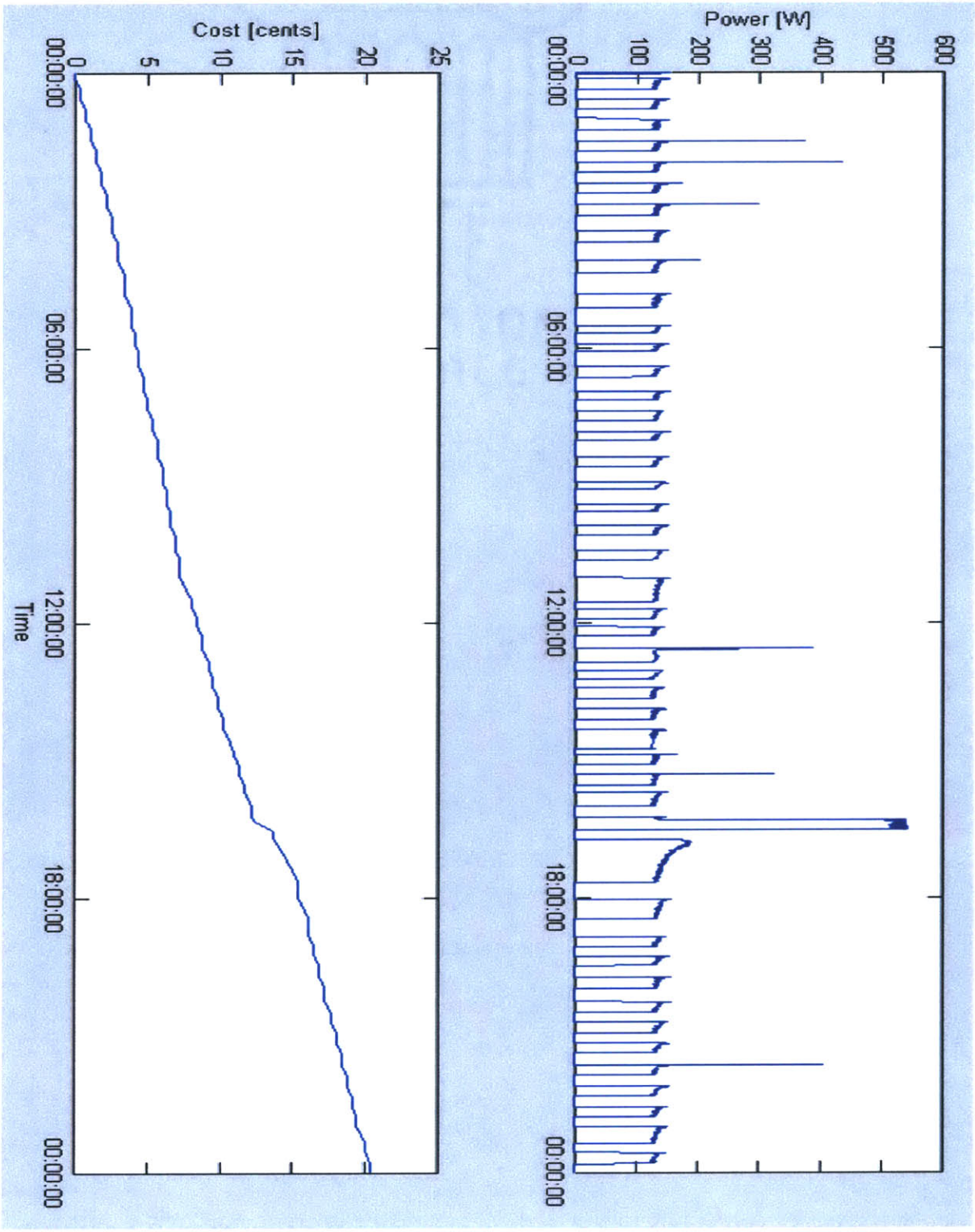


Figure 5.7: 24 Hour Refrigerator Surveillance (Power and Cost versus Time)

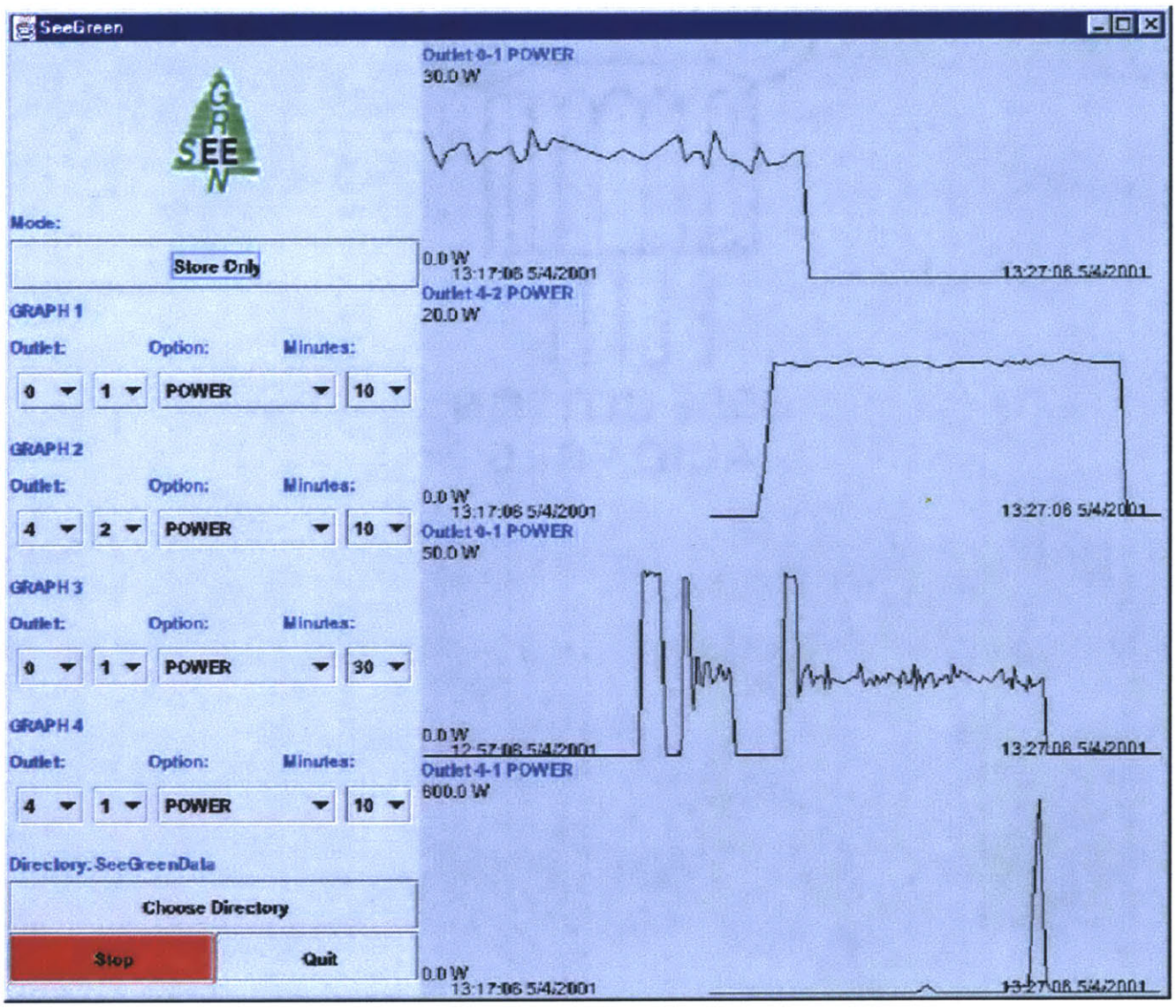


Figure 5.8: SeeGreen PC software storing and graphing multiple nodes on the same network.

5.6 Comments about Performance

Overall, I was very pleased with the performance of the SeeGreen prototypes. Naturally, there were many unanticipated problems that arose during the course of the project, but I was fortunate enough to have found reasonable solutions for most of these.

It was very satisfying to take SeeGreen from the lab to the home environment and have it function on the first attempt. In fact, because of the lower noise environment, SeeGreen functioned better (i.e. more reliably sent and received packets) at home than in the lab.

It seemed sometimes that nodes would get into a mode where they ignored all other nodes and broadcast over the top of other broadcasts. As expected, this made reception of either packet impossible. This mode could be terminated by power cycling the troubled node.

The PC software was easy to use and reliable. One drawback was a tendency for the software to get backlogged with packets while the graphing function was enabled. Graphing in java seemed to consume a significant portion of the computer processing resources. The problem could be avoided by leaving the program in "store only" mode for the majority of the time and only switching to graph mode for short periods of time.

5.7 Speculation about Integration

The information obtained by these field tests was only moderately useful for conservation. I feel slightly more connected to my appliances now, having a good sense for what is happening within them as they operate. I've learned a few simple things. For example, never leave a stereo on with the volume turned all the way

down. The power consumption of a stereo is almost independent of its volume. But the really useful stuff has yet to be developed. This is the hardware foundation upon which wonderful things can be developed.

The real benefit to the conservation movement will come after two important steps are taken. First, these devices must be sufficiently miniaturized to fit into every connection point in a home including outlets, lights, and thermostats. They should be incorporated into the walls and not a protruding accessory. Secondly, the development of active, real-time displays will advance the product from a secret, hidden device to a device actively integrated into the average American life. These displays could take many forms. Imagine, for example, a sculpture with a power cord. The sculpture in some beautiful and artistic way displays a qualitative summary of the electrical power usage in the whole house (information that it gets through its power cord). As the electricity used in the house declines, the sculpture becomes even more beautiful.

Alternatively, one could develop large visual displays, perhaps similar to SeeGreen's computer screen display that relays to its users quantitative information in the form of graphs, charts, or numbers. This display would be hung in a common area where it would be easily seen and people could learn from it.

Creative ideas such as these are the key to creating a foundation of general knowledge about energy use. Presently, the average American has little or no grasp on the meaning of an amp, watt, or kilowatt-hour. We fully understand the workings of money and can quickly calculate the interest earned on a tax-free 401K adjusted for inflation. In many ways it is all about priorities. And until we make energy a priority and learn about its workings, our impact on the earth will continue to be huge.

6 Future Work

Through the course of any large project, one should expect to have "better" ideas. During this project, I had many ideas that I was unable to act on due to time and resource constraints. Ten times as many people could have attacked this project for ten times as long and still there would have been plenty to do. Here are some of my suggestions for the most important of the ideas that never had a chance to be realized:

The majority of the space and weight of the SeeGreen outlet box is consumed by the power supply transformer, bridge, and capacitors. If these could be replaced, the decreased cost and size would be tremendous. Therefore I propose an AC step-down power supply. It could be constructed to be more efficient than the current bridge and much smaller and lighter than the transformer. The engineering challenges involved in such a switching power supply are not to be taken lightly. The two DC-DC switching power supplies already in the box were challenging; an AC-DC switching supply could be a thesis by itself ...but the rewards would be unquestionably significant.

Another major simplification to the design would be the addition of a purely analog power sensor. Suppose for example a variable gain amplifier was employed. The current and voltage could be continuously multiplied and filtered in the analog world such that the microprocessor would only have to measure one voltage and this voltage would be proportional to the power on that line. One of the more significant advantages of this would be the immensely decreased requirements on the processor. Almost no calculations would need to be performed. The processors largest responsibilities would be to serve as an ADC, calculate error-checking codes, and generate serial data. A much smaller and less expensive microprocessor could replace the PIC16F877. Plus, power measurements, filtered by a capacitor, need only

be measured once (not averaged over hundreds of samples) and could therefore be done in rapid succession for a more continuous data stream.

A third advancement would be replacing the current sensing transformers. The original model of this device did not employ current-sensing transformers at all. Instead a tiny resistor (.02 Ohm) was put in series with the outlet load and the voltage across it was measured. Due to isolation and power consumption issues, I opted to replace this resistor with the current transducer; however, the resistor was, naturally, much cheaper than the transducer and if done properly this method could save significant money and space.

In the future, I would like to see similar devices keeping tabs on household electricity consumption in a standardized way. I envision these devices as tiny boxes that easily fit into existing in-wall outlet boxes and light fixtures for minimal extra cost. They would constantly report their measurements around the house for active displays or control electronics. Someday, someone with a Sociology degree will be required to study the potential for integrating information about our homes into our home lives. Conservation must someday become a routine part of life and this awareness can only begin when we, and the devices we live with, have access to all of the necessary information to make this possible.

References

CA Energy (2000)
http://www.energy.ca.gov/contingency/2000_electricity/reducing_use.html

CIA (2000) The World Fact Handbook
http://www.odci.gov/cia/publications/factbook/index.html

ENS (2001) Evidence of Rapid Global Warming Accepted by 99 Nations Environmental News Service Jan 22.

EPRI (1996) Non-Intrusive Appliance Load-Monitoring System: A revolutionary yet cost-effective way to perform end-use load surveys Report: BR-106086 Jan 1996.

EPRI (1997) Noninstrusive Appliance Load Monitoring System (NIALMS): Beta-Test Results Report: TR-108419 Sep 1997.

Hunt (1986) Electrical Energy Monitoring and Control System for the Home IEEE Transactions of Consumer Electronics, Vol. CE-32 August p. 578-583

Jossi, Frank (2001) Standby Sucks *Wired* Feb. p. 80.

Mak, Sioe T. Reed, Don L. (1982) TWACS, A New Viable Two-Way Automatic Communication System For Distribution Networks *IEEE Transactions on Power Apparatus & Systems.* n 8 Aug p 2941-2949

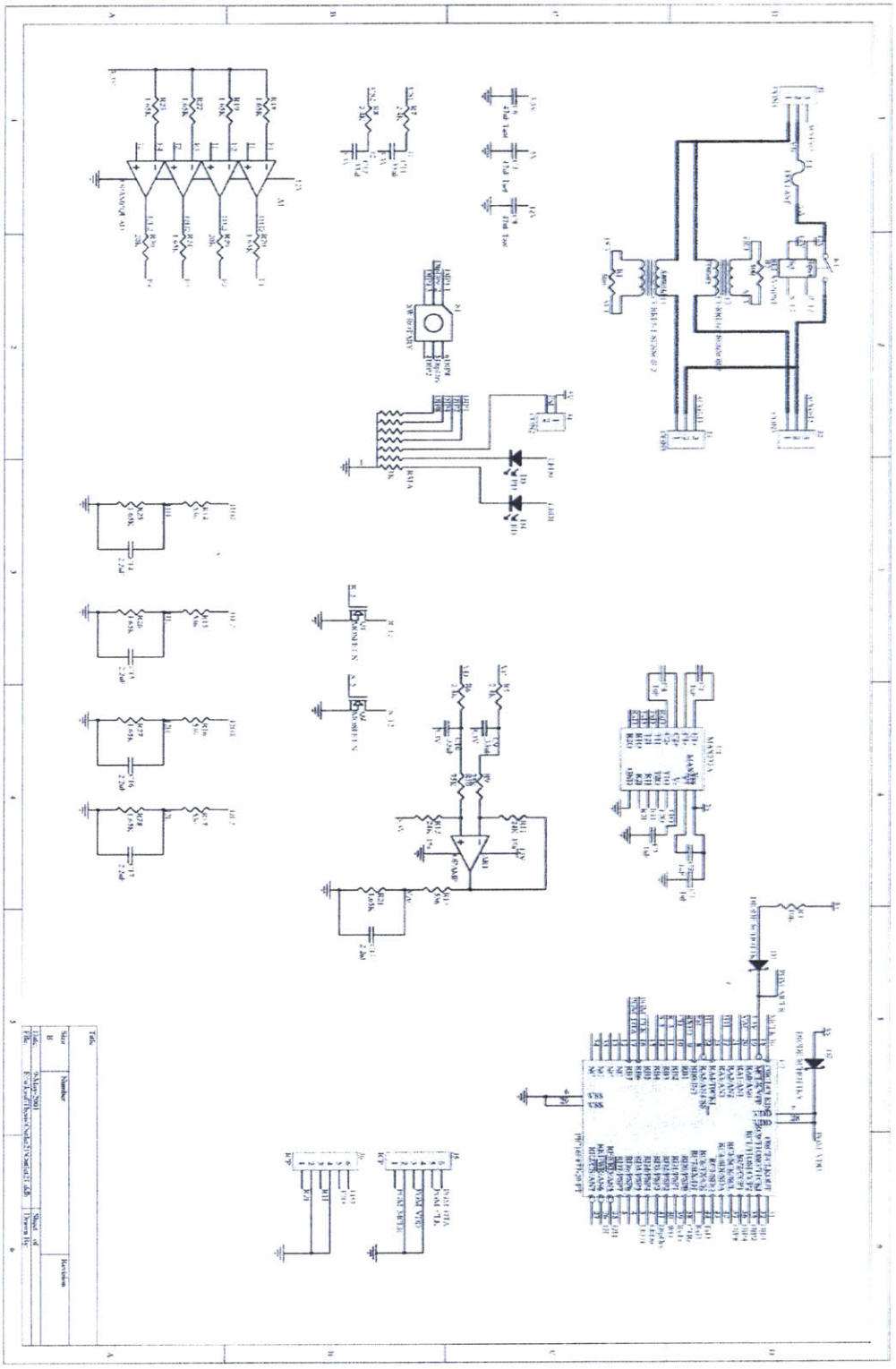
Mak, Sioe T. (1984) New Method of Generating TWACS Type Outbound Signals For Communication on Power Distribution Networks *IEEE Transactions on Power Apparatus & Systems.* n 8 Aug p 2134-2140

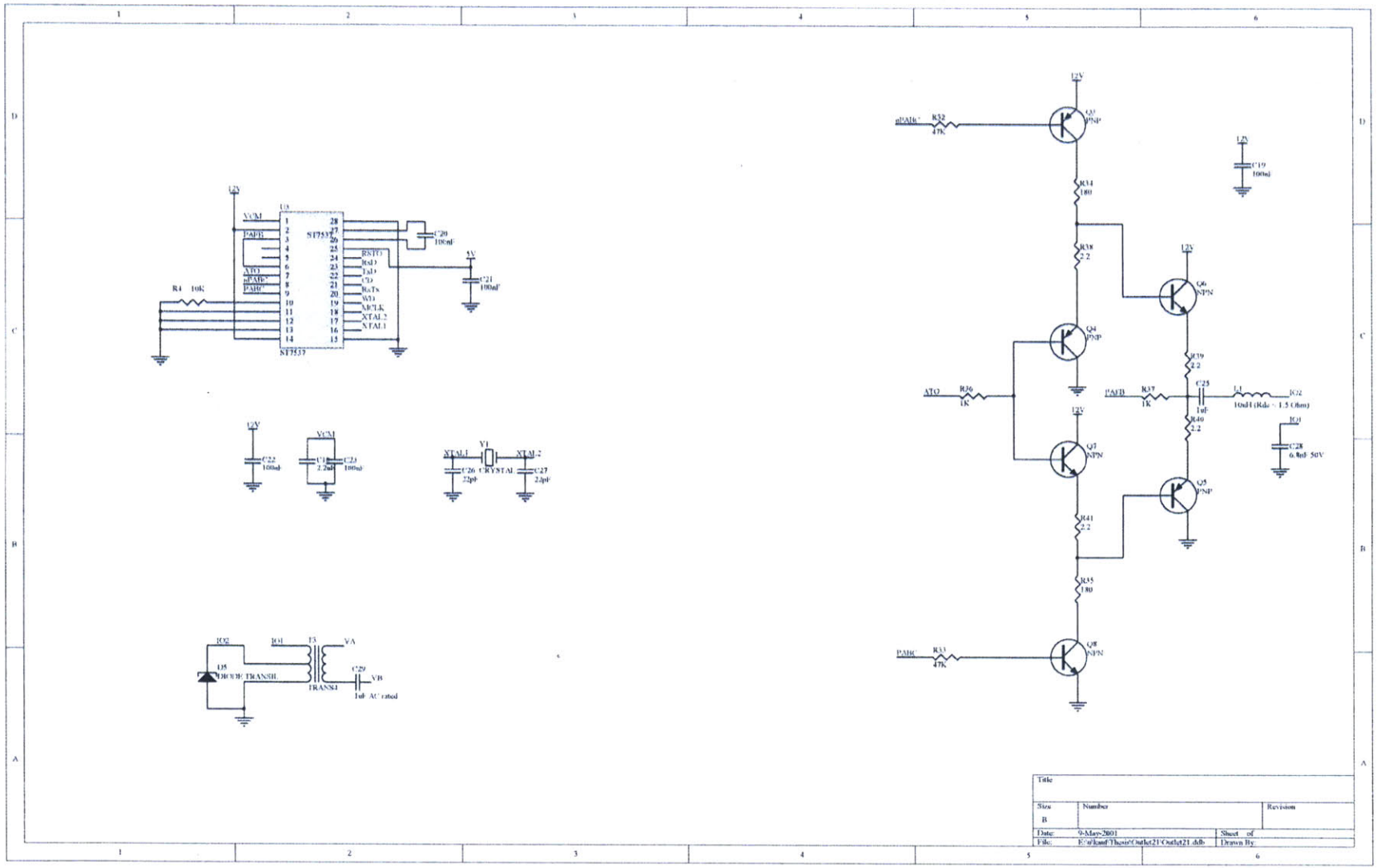
Orban, Julius (1980) TWACS EM DASH The 60 Hz Power Line Communication System. *Conference Record - Control of Power Systems Conference & Exposition.* n. 80CH1540-4REG5, Piscataway, NJ, p 81-89

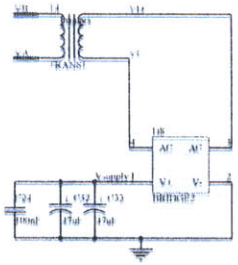
Strassberg, Dan (1996) Powerline Communication: Wireless Technology EDN June 6, p. 71-78

Sweet, W (2001) An Unnatural Rush to Natural Gas? IEEE Spectrum Jan. p. 83.

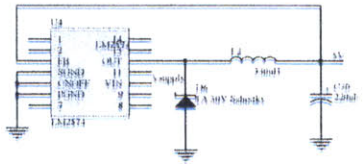
Appendix A: Schematics



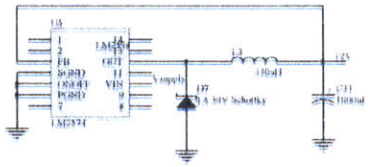

12V
VCM
PAFB
ATO
nPATE
PABC
R4 10K
U3
ST7537
C20 100nF
C21 100nF
5V
C22 100nF
C23 100nF
Y1
CRYSTAL
C26 22pF
C27 22pF
D5
DIODE TRANSIL
T2
T1
T3
VA
VB
C29
1uF AC rated
R32 47K
R34 180
R38 2.2
Q3 PNP
Q4 PNP
Q6 NPN
Q7 NPN
Q5 PNP
Q8 NPN
R33 47K
R35 180
R41 2.2
R40 2.2
R39 2.2
R36 1K
R37 1K
C25 1uF
L1 10uH (Rdc = 1.5 Ohm)
C19 100nF
C28 6.8nF 50V
Title
Size B
Number
Revision
Date: 9-May-2001
Sheet of
File: E:\Kanf\Thesis\Outlet21\Outlet21.ddb
Drawn By:

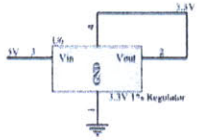


5V Switching



12V Switching





Title		
Size: B	Number	Revision
Date: 9-May-2003		Sheet of
File: E:\erikmf\Thesis\Outlet21\Outlet21.ddb		Drawn By:

Appendix B: Code

PIC Code

```
/*outlet21.c: This PIC program was designed for See Green Outlet 2.1.
It will send power and power factor information over the RS232
line  */

#include <16F877.H>
#include <math.h>

#define ClockFreq 11059200
#use Delay(clock=ClockFreq)

#fuses HS,NOPROTECT,PUT,NOWDT,NOLVP

#use fast_io(A)
#use fast_io(B)
#use fast_io(C)
#use fast_io(D)
#use fast_io(E)

//TRIS
#define A_TRIS 0xFF             //for analog inputs
#define B_TRIS 0xE7             //for lcd output drivers
#define C_TRIS 0x9F
#define D_TRIS 0x00
#define E_TRIS 0xFF

//Unit Correction Factors
#define VoltageScale 80.85
#define VoltageShift 2.50
#define CurrentScaleH 3.01
#define CurrentScaleL 0.456
//#define Current2ScaleH 3.01
//#define Current2ScaleL 0.456

//Other necessary constants
#define LowThreshUpper      950         //Defines when to switch over to
high current channel; up to 1024
#define LowThreshLower      50
#define TimePerTick 4/ClockFreq         //8e-6s per tick for 4 MHz
clock; 3.2e-6 for 10Mhz
#define PowMult 37                      //multiply powers by this before
converting to 16 bit
#define PowFacMult 255                  //multiply powerfactor by this
before converting to 8 bit
#define MaxSendWaitTime_ms 200          //how long to wait on a busy
power line
#define PacketSize 12                   //including sync byte
#define RefVolt 3.3                     //should be the voltage of the
zener*10
#define ValidReadWait 50                //50 uS delay before analog
reads
```

```
#define RefSamples 3                    //total sample time =
~PowerSamples*(ValidReadWait+20)*6 per line
#define PowerSamples 400
#define Intersample_ms 200
#define MaxOverFlows 5                  //each overflow is 23.6mS; 5
should be 118mS

//Analog Appropriations
#define CHAN_Ref 0
#define CHAN_VAC 1
#define CHAN_Current1High 3
#define CHAN_Current1Low 4
#define CHAN_Current2High 5
#define CHAN_Current2Low 6

//Port B appropriations
#define PIN_RSTO   PIN_B1
#define PIN_CD     PIN_B2
#define PIN_Reset  PIN_B3
#define PIN_Set    PIN_B4

//Port C Appropriations
#define PIN_DIP1   PIN_C0
#define PIN_DIP2   PIN_C1
#define PIN_DIP4   PIN_C2
#define PIN_DIP8   PIN_C3

//Port D appropriations
#define PIN_Ctrl   PIN_D0
#define PIN_RxTx   PIN_D1
#define PIN_WD     PIN_D2
#define PIN_DipDrv PIN_D3
#define PIN_LED0   PIN_D4
#define PIN_LED1   PIN_D5

#use rs232(baud=2400, xmit=PIN_C6, rcv=PIN_C7)

//Universal measured parameters
float Power1, Power2, PowerFactor1, PowerFactor2;
float Current1HighBias, Current1LowBias, Current2HighBias,
Current2LowBias;
int Data[PacketSize];
int Name, Turn, GreenState, RedState;
int OverFlows;

#include <CRC16.h>
#include <myrobin.h>

//watchdog timer stuff for ST7537

#INT_RTCC
clock_isr()
  {
  output_high(PIN_WD);
  delay_us(10);
  output_low(PIN_WD);
```

```
  }

#INT_TIMER2
read_time_out()
  {
    if (OverFlows >= MaxOverFlows) {
      /*GreenState = 1 - GreenState;
      if (GreenState)
        output_high(PIN_LED1);
        else output_low(PIN_LED1); */
      OverFlows = 0;
      Turn = ((Turn + 1) & 0x0F);
      }
    else OverFlows++;
  }

void Initialize(void);
void GetMeasurements(void);
float WhatsOnChannel(int Channel, long Samples);
long QuickCheck(int Channel);
int SendPacket(void);
unsigned long Float2Long(float F);
unsigned int Float2Int(float F);
int GetName(void);
void SetErrorBytes(void);
void SetReset(int On);            //sending a 1 turns AC on

void main()
{
  Initialize();
  while(1) {
    GetMeasurements();
    SendPacket();
    delay_ms(Intersample_ms);
  }
}

void Initialize(void)
{
  disable_interrupts(GLOBAL);

  set_tris_b(B_TRIS);
  set_tris_c(C_TRIS);
  set_tris_d(D_TRIS);

  setup_adc_ports (ALL_ANALOG);
  setup_adc (ADC_CLOCK_INTERNAL);
  port_b_pullups(false);

  set_rtcc(0);
  setup_counters(RTCC_INTERNAL, RTCC_DIV_128);
  enable_interrupts(RTCC_ZERO);
  enable_interrupts(INT_TIMER2);

  enable_interrupts(GLOBAL);
```

```
  output_high(PIN_RxTx);                //enables receiving stuff from
the lines

  Name = GetName();
  SetReset(1);                          //turn on AC power

  Current1HighBias = WhatsOnChannel(CHAN_Current1High, 100);
  Current1LowBias = WhatsOnChannel(CHAN_Current1Low, 100);
  Current2HighBias = WhatsOnChannel(CHAN_Current2High, 100);
  Current2LowBias = WhatsOnChannel(CHAN_Current2Low, 100);
  //VoltageShift = WhatsOnChannel(CHAN_VAC, 10000);

  output_high(PIN_LED0);                //signify successful
initialization
  output_high(PIN_LED1);
  delay_ms(300);
  output_low(PIN_LED0);
  output_low(PIN_LED1);
}



void GetMeasurements(void)
{
  long VA, VB, IH, IL;
  float V, I, VScum, IScum, VIcum;
  float RefScale;
  long j;

  //output_high(PIN_LED0);

  RefScale = (float)RefVolt/WhatsOnChannel(CHAN_Ref, RefSamples);

  VScum = 0;
  IScum = 0;
  VIcum = 0;

  for(j=0;j<PowerSamples;j++)
  {
    VA = QuickCheck(CHAN_VAC);
    IH = QuickCheck(CHAN_Current1High);
    IL = QuickCheck(CHAN_Current1Low);
    VB = QuickCheck(CHAN_VAC);

    V = ((float)(VA+VB))/2*RefScale;                          //average
the voltages on either side of the current measurement
    V = (VoltageShift - V)*VoltageScale;                      //this
also inverses the voltage for channel 1

    if ((IL > LowThreshUpper) || (IL < LowThreshLower))
      I = ((float)IH-Current1HighBias)*RefScale*CurrentScaleH;
      else I = ((float)IL-Current1LowBias)*RefScale*CurrentScaleL;

    VScum += V*V;
    IScum += I*I;
    VIcum += V*I;
```

```
  } //end sampling for-loop
  //computations for line 1 power and powerfactor
  Power1 = (VIcum/(PowerSamples-1));
  PowerFactor1 = VIcum/(sqrt(VScum*IScum));

  //output_low(PIN_LED0);
  //output_high(PIN_LED1);

  RefScale = (float)RefVolt/WhatsOnChannel(CHAN_Ref, RefSamples);

  VScum = 0;
  IScum = 0;
  VIcum = 0;

  for(j=0;j<PowerSamples;j++) {
    VA = QuickCheck(CHAN_VAC);
    IH = QuickCheck(CHAN_Current2High);
    IL = QuickCheck(CHAN_Current2Low);
    VB = QuickCheck(CHAN_VAC);

    V = ((float)(VA+VB))/2*RefScale;
    V = (VoltageShift - V)*VoltageScale;

    if ((IL > LowThreshUpper) || (IL < LowThreshLower))
      I = ((float)IH-Current2HighBias)*RefScale*CurrentScaleH;
      else I = ((float)IL-Current2LowBias)*RefScale*CurrentScaleL;

      VScum += V*V;
      IScum += I*I;
      VIcum += V*I;
  } //end sampling for-loop
  //computations for line 2 power and powerfactor
  Power2 = (VIcum/(PowerSamples-1));
  PowerFactor2 = VIcum/(sqrt(VScum*IScum));

  //output_low(PIN_LED1);
}

float WhatsOnChannel(int Channel, long Samples)
{
  long Val;
  float CumVal=0;
  long j;

  set_adc_channel(Channel);
  delay_us(ValidReadWait);

  for(j=0;j<Samples;j++)
  {
    Val = read_adc();
    CumVal += (float)Val;
  }

  return CumVal/Samples;
}
```

```
long QuickCheck(int Channel)
{
  set_adc_channel(Channel);
  delay_us(ValidReadWait);
  return read_adc();
 }

int SendPacket(void)
{
  int PacketBytes = 12;
  int i;
  long RollOvers,t;
  float time;
  unsigned long ErrorBytes;

  //output_high(PIN_LED1);
  GreenState = 1;
  //Generate all header bytes
  Data[0] = 0x54;                                  //SYNC = 01010100
  Data[1] = 0x00;                                  //HDB2 = no address
bits; no protocol flags; no ACK
  Data[2] = 0x47;                                  //HDB1 = 01000111 ; no
CMD; CRC16; 7 data bytes
  //Data[2] = 0x09;           //HACK!               //HDB1 = 00001000 ;
no CMD; none; 16 data bytes

  //Generate all 7 data bytes
  Data[3] = Name;                                                  //Name
  Data[4] = (unsigned int)(Float2Long(Power1*PowMult) >> 8);
//high byte of power1
  Data[5] = (unsigned int)(Float2Long(Power1*PowMult) & 0x00FF);
//low byte of power1
  Data[6] = (unsigned int)(Float2Long(Power2*PowMult) >> 8);
//high byte of power2
  Data[7] = (unsigned int)(Float2Long(Power2*PowMult) & 0x00FF);
//low byte of power2
  Data[8] = Float2Int(PowerFactor1*PowFacMult);    //power factor 1
  Data[9] = Float2Int(PowerFactor2*PowFacMult);    //power factor 2

  //Generate 2 bytes for CRC16
  ErrorBytes = GenerateCRC();
  Data[10] = (unsigned int)(ErrorBytes >> 8);              //this is
correct
  Data[11] = (unsigned int)(ErrorBytes & 0x00FF);

  if (MyTurn())
  {
    output_low(PIN_RxTx);
    delay_ms(50);
    for(i = 0; i<PacketSize; i++)
      putc(Data[i]);
    delay_ms(60);
    output_high(PIN_RxTx);

    //output_low(PIN_LED0);
    //output_low(PIN_LED1);
```

```
    return 1;
  }
  else
  {
    //output_low(PIN_LED0);
    //output_low(PIN_LED1);
    return 0;
  }
}

unsigned long Float2Long(float F)
{
//this should convert F to its nearest integer less than 65536
if (F >= 65535) return 65535;
if (F <= -65535) return 65535;
if (F < 0) return (unsigned long)(-1*F);
return (unsigned long)F;
}

unsigned int Float2Int(float F)
{
//this should convert F to its nearest integer less than 256
if (F >= 255) return 255;
if (F <= 0) return 0;
return (unsigned int)F;
}

int GetName()
{
  int Name;
  output_high(PIN_DipDrv);
  delay_us(100);
  Name = 1*input(PIN_DIP1);
  Name += 2*input(PIN_DIP2);
  Name += 4*input(PIN_DIP4);
  Name += 8*input(PIN_DIP8);
  output_low(PIN_DipDrv);
return Name;
}

void SetReset(int On)
{
  if (On) {
    output_high(PIN_Set);
    delay_ms(5);
    output_low(PIN_Set);
    }
  else {
    output_high(PIN_Reset);
    delay_ms(5);
    output_low(PIN_Reset);
    }
}

/*crc16.h: This program was designed to generate the SeeGreen error
checking codes
```

```
used by the onboard PIC*/

//CRC-16 Parameters
#define INIT 0xFFFF
#define WIDTH 16
#define POLY 0x1021
#define BITMASK 0x8000

unsigned long CrunchByte(unsigned long reg, int ch)
{
  int i;
  unsigned long uch;
  unsigned long topbit = BITMASK;

  uch = (unsigned long) ch;

  reg ^= (uch << (WIDTH-8));
  for (i=0;i<8;i++) {
    if (reg & topbit)
      reg = (reg << 1)^POLY;
    else
      reg <<= 1;
    }
return reg;
}

unsigned long GenerateCRC(void)
{
  int i;
  unsigned long CRC = INIT;

  //crunch each byte individually:
  for (i=0;i<PacketSize-2;i++)
    {
    CRC = CrunchByte(CRC, Data[i]);
    }
  return CRC;
}
```

MATLAB Scripts

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%CostFinder.m%%%%%%%%%%%%%%%%%%%%%%%%%
%This script was written by Joshua Kaufman on April 30, 2001.
%It takes a section of the SeeGreen data record as specified in the
CONSTANTS section and
%creates a running total of the cost of the electrical energy used.
The electricity cost
%rate used here is $0.12/kWh, a typical rate in a moderately expensive
region.

clear TimeData PowerData PowerFactorData TimeNumber PD PFD cents

%%%%%%%%%%%%CONSTANTS%%%%%%%%%%%%%%
%These define which section of data will be reported
rate = 12;          %cents per kWh
year = 2001;
month = 5;
day = 5;
outlet = [4 2];
InputFileName = ['C:\SeeGreen\SeeGreenData\Day' int2str(month) '_',
int2str(day) '_',int2str(year) '\Outlet',...
   int2str(outlet(1)) '-' int2str(outlet(2)) '.bin']
MORNING=datenum(year, month, day, 0, 0, 0);
NIGHT = datenum(year, month, day+1, 0, 0, 0);

STARTFILE = MORNING;
%STARTFILE = datenum(year, month, day, 13, 10, 0);
ENDFILE = NIGHT;
%ENDFILE = datenum(year, month, day, 13, 10, 0);

%%%%%%%%%%%%%%%%GET DATA FROM FILE%%%%%%%%%%%%%%%%%
[FID, Message] = fopen(InputFileName, 'r', 'ieee-be');
if (FID == -1)
   error(Message);
end

i=1;

fseek(FID,0,1);
DP = floor(ftell(FID)/12);
frewind(FID);

while (i <= DP)
   TD(i) = fread(FID, 1, 'int32');
   PD(i) = fread(FID, 1, 'float32');
   PFD(i) = fread(FID, 1, 'float32');
   i = i + 1;
end

%%%%%%%%%%%%%%DATA PROCESSING%%%%%%%%%%%%%%%%%%%
TimeNumber = (TD ./ (24*60*60)) + datenum(2001,1,1);

%select the areas specified by STARTFILE and ENDFILE
i = 1;
```

```
while((i <= DP) & (TimeNumber(i) < STARTFILE))
   i = i + 1;
end

j = 1;

while ((i <= DP) & (TimeNumber(i) < ENDFILE))
   TimeData(j) = TimeNumber(i);
   PowerData(j) = PD(i);
   PowerFactorData(j) = PFD(i);
   j = j + 1;
   i = i + 1;
end

DataPoints = j-1;

%integrate to find the energy and scale to find the cummulative cost
total
cents(1) = 0;

for i = 1:DataPoints-1
   cents(i+1) = cents(i) +
((PowerData(i)+PowerData(i+1))/2)*(TimeData(i+1)-TimeData(i));
end
cents = cents*24/1000*rate;

%%%%%%%%%%PLOTTING%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(42);
subplot(2,1,1)
plot(TimeData, PowerData)
ylabel('Power [W]');
datetick('x','HH:MM:SS');

subplot(2,1,2)
plot(TimeData, cents)
xlabel('Time')
ylabel('Cents [W]');
datetick('x','HH:MM:SS');
title(['Cost Data for Outlet' int2str(outlet(1)) '-' int2str(outlet(2))
' on ' int2str(month) '/' int2str(day) '/' int2str(year)]);
```

5563-58