Intelli-Home Electronics Interpreter

Project Report for Senior Project Design

Team name

Faraz Milani, EE
Kelvin Liang, EE
Young Min Kim, EE
Matthew Cai, CPE
Mentored by: Arthur Zhang, G.P. Li

Senior Project Design
Professor Mark Bachman
The Henry Samueli School of Engineering
The University of California, Irvine

March 21, 2014

Executive Summary

In North America alone, $7 billion is spent annually on phantom power—power being consumed while a device is plugged into the outlet but not in use. Any device that can be plugged into an electrical outlet is called a plug load. Our project is a plug load recognition and management system to be used within the average-sized American home for the purposes of reducing phantom power. The idea is to create a system able to automatically recognize electronic devices that are plugged in throughout the home, and an Android mobile application that allows the homeowners to shut off outlets that are not in use. Once the devices are recognized, the user is able to easily set personalized schedules using the mobile application dictating when specific outlets will be on and off, in effort to prevent as much phantom power waste as possible.

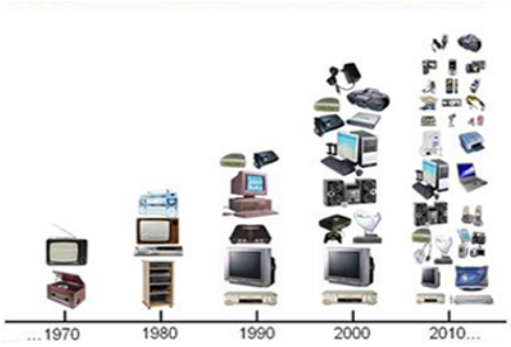
We designed this system by breaking it into three main components. First, there is the outlet module that the user plugs his or her device into. Next, the device recognition component identifies the device. Finally, the device name is sent to the Android mobile application via WiFi where the user is able to view the plug load and switch the outlet on or off. The main hardware of our project includes a current probe, digital signal processing board, and Raspberry Pi. Our "base station" and user interface is the Android mobile application that functions on both tablets and smart phones.

The deliverables of our project are (1) a functioning prototype of the system, including the outlet module, device recognition system, and mobile application; (2) a software program for user interface of the mobile application; (3) a software program for automatic load recognition; (4) a video displaying the recognition and on/off capability of the system; (5) a technical report with our plans, test results, and algorithm development; (6) a demonstration of our product.

The final result of our project is a system that works seamlessly, allowing the user to recognize over five devices entered into the database. In order to use our system, the user opens the application, shakes the device to scan for devices, and controls the recognized devices by using the on and off button. Devices stored in the database include TV, audio speakers, and Macbook Pro, and other common household devices. IHEI has an averaged 3.15-second delay for switching the outlet module on and off, and 17.45-second delay from the moment the device is plugged in to when the device is displayed on the mobile application. By eliminating the need for the user to program the system manually, our plug-and-play system is extremely user friendly and suitable for any average home in the U.S.

Introduction

In 1980, the average US home contained approximately 3 consumer electronics devices. Today that number has increased to 25, a number that will continue growing due to constant innovation and high demand for more advanced consumer technology. Consumer electronics and computer equipment currently take up around 15% of electricity consumption globally [1]. The International Energy Agency (IEA) expects the energy use by these types of devices to triple by 2030 if no action is taken to increase energy efficiency.



Plug load devices are accountable approximately 15% of the electrical consumption in our households. It is foreseen that the percentage of energy consumption in residential areas due to plug load devices can reach up to 30% by 2030 [2]. Most plug loads draw power even when are idle and not in use, which is extremely energy inefficient. The power that is wasted as an appliance or device is not in use is known as "phantom power". North America alone wastes $7 billion a year on phantom power [3], which is an obvious problem. As a country that is slowly moving towards the idea of Zero Net Energy (ZNE) homes, houses capable of producing their own energy and consuming very little energy, efforts must be made to lower the energy use of plug load devices.

In our busy daily lives, unplugging devices that are not being used seems very inconvenient, or maybe even impossible. For people who are gone all day at work, parents who are constantly running errands out of the house, and children who can be quite forgetful, making sure to turn off and unplug appliances like chargers, televisions, Blu-ray players, video game consoles etc. is a difficult task. With all the resources available today for wireless connectivity, home automation, and mobile applications, there is a great need for an automated system to take care of turning off and unplugging devices that are not in immediate use. Our project aims at creating such a system; one that is able to intelligently recognize any basic household plug load and control/schedule its use via a simple mobile application. This could greatly cut down the phantom energy waste in homes, leading to electricity bill savings and less stress on the environment for electricity production.

Background

Home energy management is a popular field today with all the developments in wireless technology and big data, and there is great potential to save a lot of energy, money, and the environment. Many companies are working on systems that interact with utility companies in order to schedule larger appliances like washing machines and dishwashers around the peak hours of electricity use in order to save homeowners money. However, with the growth of electronic plug load devices per home over the last thirty years and the amount of energy wasted due to these devices, our group is focusing specifically on monitoring plug loads.

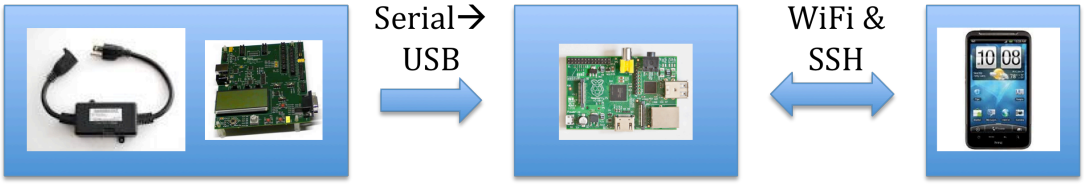
Within the scope of managing home plug loads, there has already been some work done by some larger companies and several startup companies. Home security companies such as alarm.com and ADT have created features in their security systems that allow the homeowner to remotely turn on specific outlets. However, with most home security companies a monthly subscription fee for the service is required, which is not ideal for people looking for a cheap and easy way to manage their plug load power use. Also, many security companies require installment of a touch screen panel or light fixtures within the home, making upgrades or changes in the system difficult and expensive.

For those who are interested in simply controlling the activity of their electrical outlets around the house, a few startup companies have created simpler and cheaper solutions. Ninja Blocks and Valta, for example, have designed energy management systems that allow users to turn specific plug loads around the house on and off and personalized create scheduling systems that turn devices on and off based on the time of the day. The system architecture of these systems requires a base station that communicates with their outlet adapters, adding an extra hardware component and complicating the system. Other products such as Belkin's WeMo Insight Switch have outlet adapters that communicate directly with the mobile application that comes with the product, instead of having a base station that controls the wireless communication. The products mentioned above use wireless protocols such as RF 433.92 MHz, ISM frequency band, and WiFi.

The common downfall to all the current solutions to managing the phantom power of plug loads is that the users must manually program their system. This means once the user assigns a certain outlet module to a device, if they want to plug in a new device they have to manually reassign that module to the new device via some provided software. Those who are not tech-savvy do not have the patience or knowledge to do this, making the current systems inconvenient. IHEI solves this flaw by implementing automatic device recognition—when the user shakes their Android device, our system will automatically recognize the current device and reprogram the system.

Project Design

Our project produced a home plug load management system with automatic device recognition focused on preventing phantom power consumption, controlled by an Android mobile application. The three subsystems of our product are shown below in a block diagram. The wireless communication between the Device Recognition system and the Mobile Application system was done using WiFi and Secure Shell (SSH) protocol. We chose to use WiFi protocol because the one of our design requirements was to create a user-friendly product. We considered using Zigbee and Z-Wave protocol which are popular in home automation, but this would require us to add an extra piece of hardware to act as the base station and communicate with the Outlet Module system. Because our aim was to make the system as simple as possible, we choice to go with WiFi where the Mobile Application System could communicate directly to the outlet level and decrease our response times for on/off capability and device recognition. SSH is a cryptographic network protocol designed for security and integrity of data, which we chose to keep the information and control of devices in the reach of the user and nobody else. Faraz and Kelvin specialized in the automation and Outlet Module design, Young wrote the device recognition algorithm, and Matt developed the Android mobile application.



Outlet Module | Device Recognition | Mobile Application

<u>Subsystems of our project</u>

1. **Outlet Module**
    a. Fluke Current Probe
    b. TI 2530 DSP Board
    c. Power Switch Tail II
2. **Device Recognition**
    a. Raspberry Pi v.2
    b. Device Recognition Algorithm
    c. WiFi Dongle
3. **Mobile Application**
    a. Shake-Enabled
    b. On/Off
    c. Scheduling

Outlet Module

One end of the outlet module is plugged into an electrical outlet, and the user plugs in his or her device into the other end. Our module uses these components:

- Current Probe
- TI2530 Digital Signal Processor on SmartRF05 board
- Power Switch Tail II



Our Fluke current probe is what allows the outlet module to receive real time current reading from the plug load device. This current probe is clamped around either end of the Power Switch Tail, which allows the module to take accurate current measurements of any plug load and feed the readings continuously into the DSP board for sampling. We chose the Fluke current probe because of its accuracy when taking AC current measurements.

In order to sample the waveforms of current that are drawn when the Power Switch Tail is on, we used the TI2530 DSP board due to its high sampling rate of 32kHz. As the DSP samples the waveform, the sampled points are transmitted to the Raspberry Pi at a 38,400 baud rate via a serial to USB connection to be processed. The high sampling rate is important because each device has a unique current waveform, often having sharp peaks that can only be captured with a high sampling rate. If precise samples are not taken, distinguishing between different devices with similar current waveforms will be difficult and the device recognition abilities will be hindered. The TI2530 is also a low power chip, which is one of our key design requirements.

The Power Switch Tail II is the component that is connected both to the electrical outlet, as well as the users plug load. We chose this component because it contains a relay inside, which is triggered by 3V. If 3V is sent to power switch, it turns on and current is allowed to flow. When 0V is sent, the switch opens and does not allow any current to flow. By leaving the switch at 0V at all times except for when the plug load is in use, phantom power is eliminated.

Device Recognition

The Device Recognition subsystem receives sampled data from the Outlet Module, runs the data through a device recognition algorithm, and determines the current plug load.

- Device Recognition Algorithm
- Raspberry Pi v.2 with 8GB SD card, Raspbian OS
- WiFi Dongle

The Device Recognition Algorithm is an extremely important component of our system, as it eliminates the need for the user to manually reprogram the system for every new device. How it works is that we put a number of device "templates" into a database stored on the Raspberry Pi. These device templates are created by sampling each device multiple times and taking a point by point average of the samples, storing them into text files. Whenever the Outlet Module delivers new points to the Raspberry Pi, the points are stored into a new text file. The algorithm then compares the new text file with the text files in the database, and runs a series of checks. There are three layers of checks in our algorithm:

1. Cross Correlation
2. Peak Time Matching
3. Point Matching

The algorithm runs a cross-correlation check on the two sets of points, compares the peak values, and look at the difference between each point in both text files. Each of the checks has a certain threshold value at which it either passes or fails the check. The three checks above each have a certain weighting (Cross-correlation = 0.1, Peak Time Matching = 0.6, Point Matching = 0.3). When the algorithm reads in the data of an unknown device, if the sum of the checks 1-3 is greater than or equal to 0.9, then the algorithm chooses that device as the recognized device.

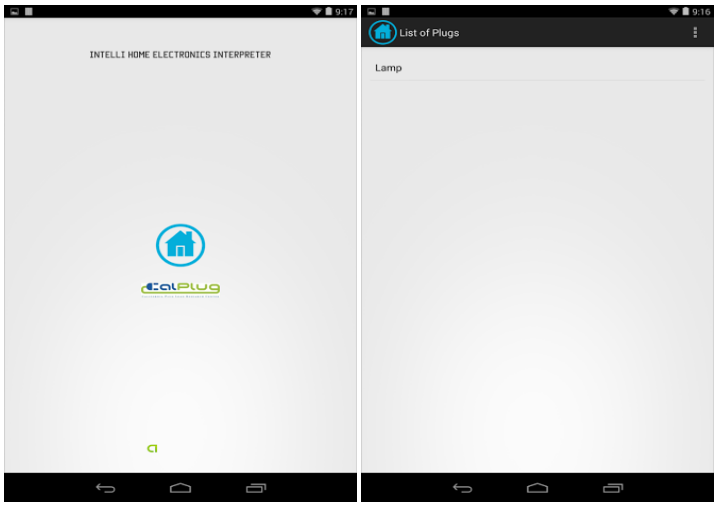
We chose to use a Raspberry Pi in our project because it is a low cost microprocessor that has strong processing power (ARM11 processor, 700Mhz) without sacrificing too much power (130-350mA). The processing power is important to us because the device recognition algorithm is computation intensive and is the cause for most of the response time delay in our system. By using the Raspberry Pi, we were able to execute large amounts of computation quickly, as well as consume very little power. The Raspberry Pi also has a large amount of storage space, allowing us to store a large amount of device templates to compare incoming data with. The WiFi dongle allows our system to communicate with the mobile application easily via Secure Shell protocol.

Mobile Application

The Mobile Application is the base station and user interface of our project. It lists the device that is plugged in and allows the user to control his or her devices.

- Shake-Enabled
- On/Off

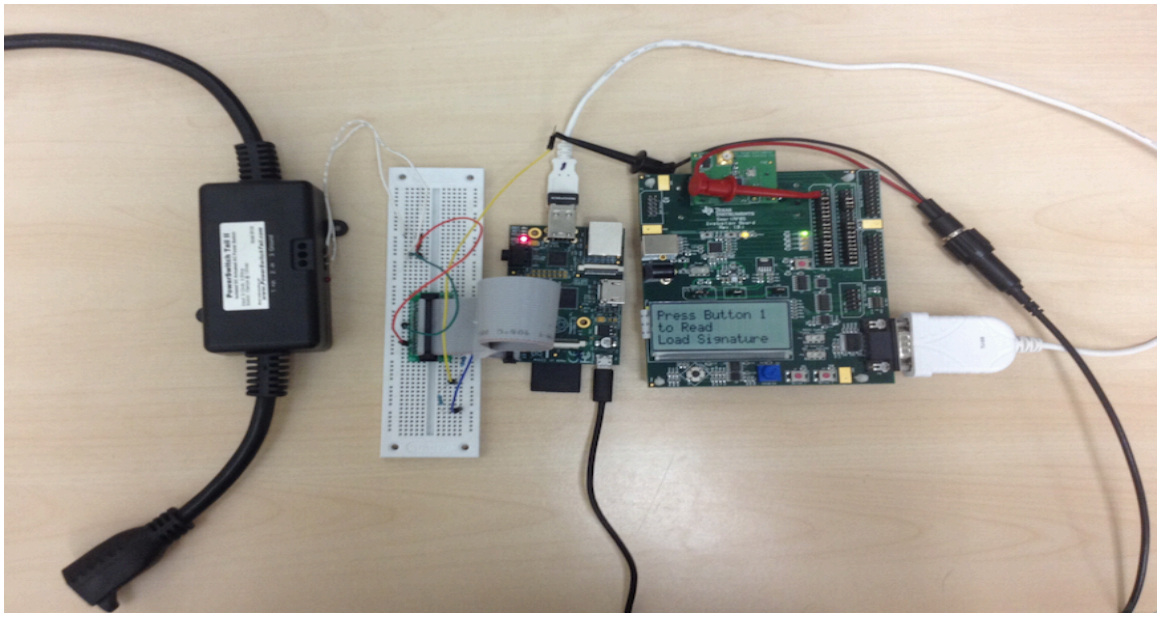
- Scheduling



One of our design requirements is to make a system that is easy to use. We achieved this by creating a shake-enabled mobile application. All the user needs to do is open the application and shake their Android device to scan for the device that is plugged in. A box that says "Scanning for device" pops up immediately, notifying the user that the system is working on identifying the device. The name of the device will then pop up on the List of Plugs home screen, at which point the user can turn it on or off. At this point users can use scheduling to automate their phantom power saving. Scheduling is a feature of the mobile application that allows the user to program times at which their outlet should turn on and off. We added this feature to make the system even more convenient to use, as most people know when they will be using certain devices and may not want to manually switch off devices everyday.

Each time the user's Android device is shaken while the IHEI application is running, a command is sent to the Outlet Module to begin sampling, and another command sent to the Device Recognition system to wait for the points from the Outlet Module and run the algorithm once they are received. After this process, a text file in the Raspberry Pi is updated with the name of the device. The mobile application is constantly checking and updating the List of Plugs screen based on the name in this text file, therefore the current device is updated onto the application shortly after. The Android application was programmed in Java using the Android ADT in the Eclipse IDE.

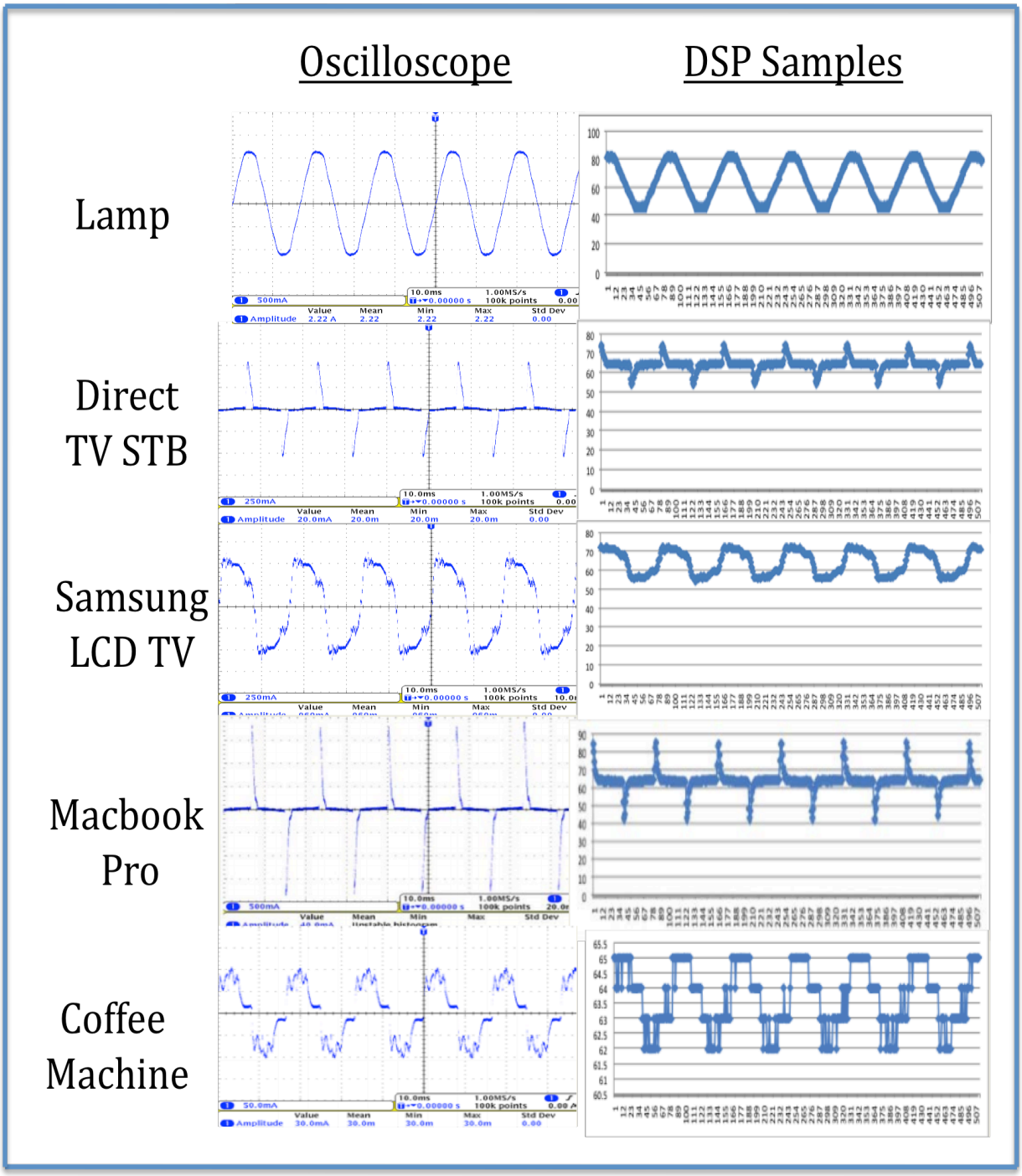
Final Prototype



Characterization

We tested our prototype in a few different ways to make sure it was functioning correctly in respect to our design requirements. Our first test was to examine the quality of our waveform sampling based on load signature. Load signature is the unique current vs. time waveform that distinguishes one plug load from another. To test the sampling precision of our Outlet Module, we analyzed the load signature plots of five different devices on an oscilloscope, and compared them with the plots of the sampled points from the TI2530 DSP for the same devices. The sampled points were plotted using the XY scatter plot function in Microsoft Excel. The resulting plots are shown below.

Load Signature



As can be seen by comparing the two sets of plots above, the overall shape of the actual load signatures and the DSP sampled load signatures are extremely similar for most

devices, other than a slight offset. However, a difference can be seen between the two plots in the case of the Coffee Machine. In the DSP sampled plot for Coffee Machine, the peaks look flat compared to the oscilloscope plot. We attribute this error to the several sharp peaks that exist in the actual Coffee Machine load signature shown on the left. One solution to this clipping problem would be increase the sampling frequency and decrease the resolution in the DSP board code in order to log more points of the original waveform and catch the peaks.

Another measurement we took was in effort to determine the accuracy of our device recognition algorithm. In order to do this, we ran the algorithm on each of the five devices ten times and observed the patterns of behavior that occurred. From this, we were able to check what layers of the algorithm each device was prone to pass, and which it would usually fail. Based off of these statistics, we were able to give a more precise weight on different layers of the algorithm in order to maximize its accuracy. The three different layers that we tested and displayed below are cross-correlation, peak matching, and point matching. Based off of the statistics below, we came to the conclusion that we should give cross-correlation the least weight (0.1), peak matching the most weight (0.6), and point matching a middle valued weight (0.3). In our final prototype, our system recognized the correct device 100% of the time, supporting our weighting choice. The statistics of the algorithm are shown on the next page.

Next, we tested different response times of our system. First, we measured the time it took for the outlet switch to turn on/off after the user had pressed the button. This was done by taking the average of 30 tests at different times of the day, and averaging the results. We chose to do the tests at 9AM, 3PM, and 10PM to get a realistic response time measurement at times where the WiFi network was very congested, as well as times where there were not many people using WiFi. The results of this test are shown on the next page. We also measured the time it took from when the user shook the device to when the name of the device showed up on their mobile application. Again, this test was done 30 times at different times of the day to get a non-biased value. The results are shown on the next page.

Algorithm Statistics

	Test#	1	2	3	4	5	6	7	8	9	10	Y %
Lamp	C	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	100
	PeM	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	100
	PoM	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	100
Direct TV STB	C	N	Y	Y	N	N	Y	Y	N	N	N	40
	PeM	Y	N	Y	Y	Y	Y	N	Y	Y	Y	80
	PoM	N	Y	Y	Y	Y	N	Y	Y	Y	Y	80
Samsung LCD TV	C	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	100
	PeM	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	100
	PoM	Y	N	Y	N	Y	Y	Y	N	Y	Y	70
Macbook Pro	C	N	N	Y	N	N	N	N	Y	N	N	20
	PeM	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	100
	PoM	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	90
Coffee Machine	C	N	Y	N	N	N	N	Y	Y	N	N	30
	PeM	N	Y	Y	N	Y	Y	Y	N	Y	N	60
	PoM	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	100

C	Correlation
PeM	Peak Matching
PoM	Point Matching

App. Statistics

Function	Time (seconds)
On/Off	3.15
Shake/Refresh	17.45

Conclusion

Our project ended up coming together quite smoothly, as we were able to create a system of multiple components that worked together seamlessly. Our system is able to recognize any of the devices that we put into the database, and updates the mobile application in a timely manner allowing the user to control his or her device. The original goal of our project was to create a system that allows users to cut down their phantom power consumption of plug load devices in a user-friendly way. Upon completion, the outlet module of our system is able to cut all power usage when switched off, and our shake-enabled application makes checking for and controlling devices as easy as shaking the device and looking at the List of Plugs for the latest device that has been plugged in.

While this system is a great working prototype, there are a few things that we would improve upon in the next version of the prototype. First of all, we think our current design could be reduced in terms of the amount of components used. We would replace the Raspberry Pi and DSP board with a single WiFi enabled microprocessor with the capability of sampling at extremely fast speeds. This would simplify our system by requiring less interfacing between the multiple pieces of hardware. Next, we would work on optimizing the device recognition algorithm in order cut down recognition time. The processing occurring in the algorithm accounts for most of the response time (approximately 10 seconds), so there is definitely potential to speed up this process. Finally, we would increase the number of devices in the database, in order to give users a larger variety of plug loads that they could use with IHEI.

The idea behind cutting down phantom power in plug loads is more than just a way to save users hundreds of dollars every year, but a way to save the environment from harmful toxins and provide environmental awareness in general. Generating the amount of electricity that is wasted in the U.S. each year in the form of phantom power creates 20 million tons of C02 that is emitted into our atmosphere. Global warming and other environmental disasters occur due to this type of activity, and by using IHEI users will be able to expand their knowledge on the current status of our environment and work to preserve the earth. Also, the amount of money that could be saved from phantom power waste annually, $3 billion, is enough to power 2.5 million homes alone. If IHEI was implemented in all homes in the U.S the effect on our economy and especially our environment would be great. In hopes to help solve our tendency to use more power than necessary, our project will be present at the 2015 Solar Decathlon in Irvine.

Team

Faraz Milani—Team Leader, Outlet Module and Automation Expert

Faraz was the team leader of the project, making sure that the system was working every step of the way. He kept in touch with the advisor, Arthur Zhang, weekly in order to update him on recent successes, setbacks, and to ask any questions the group may have. He also worked on the Outlet Module, specifically on receiving serial data from the DSP board to the Raspberry Pi via the USB port. Finally, he wrote the automation script allowed the system to run as a whole.

Kelvin Liang—DSP and Outlet Module Expert

Kelvin programmed the DSP board, allowing us to sample the current waveforms of any device plugged into the Outlet Module. He also integrated the power switch into the outlet module, allowing the using to switch off the outlet module remotely.

Young Min Kim—Algorithm Expert

Young worked on designing and coding the device recognition algorithm. He developed different layers of tests to ensure the device recognition had 100% correct device recognition every time.

Matthew Cai—Mobile Application Expert

Matthew developed the Android mobile application that serves at the base station and user interface of our project. He implemented a shake-enabled application that scans for plugged in devices every time the user shakes their mobile device.

Arthur Zhang, G.P. Li—Advisors

Arthur is the Technology Manager of the Calplug Research Center located in Calit2 on UCI campus. He provided us with a space to work in the Calplug building, some funding for our project, and other tools needed for hardware testing. Along with this, Arthur aided us with our presentation content and met with our group biweekly.