Energy Guard: Home Energy Management

Spencer Sullivan, Tyler Ensey, Gabriel Holland, and Omar Mohammed

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — This paper presents the design of a low cost home energy management system named Energy Guard. The product outlined is a three outlet power strip that monitors power and energy output for any device that is connected to it. The design of the strip is outlined including how it calculates power through the voltage and current circuit. The power supply for the strip is detailed and the parts needed. The data management is outlined showing how the server is used as a main hub to gather data being created. This data is viewed online through a web portal.**

***Index Terms* — Energy Measurement, Power Measurement, Voltage Measurement, Software Algorithms.**

I. Introduction

The increased expansion of wireless connectivity in residences and commercial buildings has created an opportunity for developers to design products which tap into this widely available resource. Although home automation systems exist, most consumers are hesitant to install them because of the costly and intrusive nature of their installation. We feel that instead of forcing consumers to endure the hassle of a complex installation, a plug and play system would simplify this process. Energy Guard, a power strip which can be accessed from a smart phone or home computer allows users to remotely monitor and control their appliances' energy consumption. With the tap of a finger, appliances can be turned off and on from any location with a connected device. It is our hope that when consumers are given this increased control over their homes and businesses that energy consumption will decrease.

II. Power Supply

The power strip will receive its power from a household wall outlet. This supply is rated for 120 Vrms at 60 Hz. The part we have chosen to use in the design for the supply is and EMI Filter which uses an IEC class C14 connector shown in figure 1.

Fig. 1. EMI Filter with an IEC C14 Male Connection with fuse and switch built in.

This part has a built in 250V 15A fuse. This fuse is needed for protection of over currents damaging the components of the system. The device also has a switch which is useful for shutting off the system during testing. Figure 2 shows the circuit diagram for the EMI filter. As its name suggests, this part offers filters to mitigate noise and offer lower impedance for the system.[1]

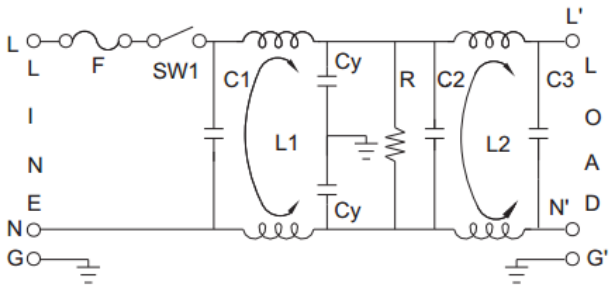

Fig. 2. Circuit Diagram of EMI filter showing the switch, fuse and filters.

Not only will the power strip need a 120V 15A AC supply, but it will also need two DC supply voltages. The two voltages are 3.3V and 5V. The 3.3V line is needed to power the MCU and the 5V line is needed to switch the relays. We achieve these voltages by using an AC/DC converter. The converter will input 120V AC and output 5V. That 5V output will be fed into a linear regulator that will output 3.3V giving us all of the proper voltages needed to operate the various components of the system. Figure 3 shows the circuit diagram for this power supply.

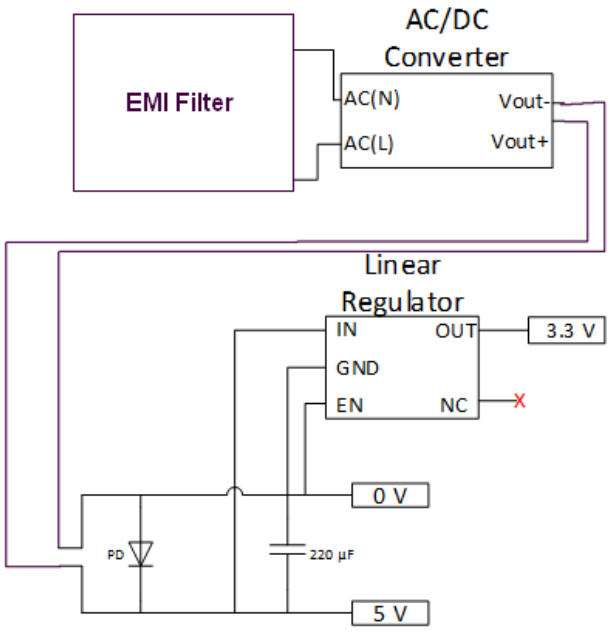

Fig. 3. Circuit Diagram of Power supply providing 3.3V and 5V needed to power all of the components in our system.

III. Power Circuits

The power strip will have 3 outlets all capable of reading the current output of the device that is connected. There are 4 circuits all together that perform the function of reading power. One voltage reading circuit and 3 identical current reading circuits. Figure 4 shows the voltage circuit we used. [2]

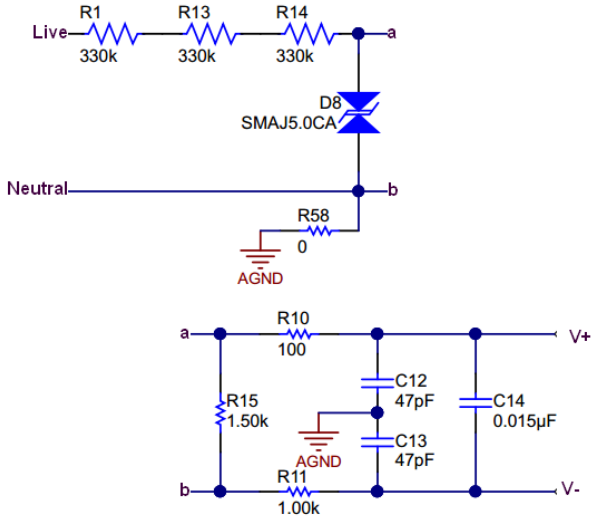

Fig. 4. Voltage Reading Circuit. Simple voltage ladder with divider to lower the input voltage for the MCU. Also some filters are used to reduce noise. Wires a and b are connected.

This circuit has three 330K resistors in series along with a voltage divider to lower the input voltage below 1.4 V so that the pin of the MCU can safely read the voltage. The voltage will actually be in the range in mV. The MCU will use some power algorithms to calculate the actual voltage by using the offset and gain of the system. This is mentioned in greater detail on the power algorithm section. There is only one voltage circuit in our design since all of the voltages will be the same for each outlet. Each outlet will have a different current which will determine the power output of the outlet. Figure 5 shows the current circuit used in our design.

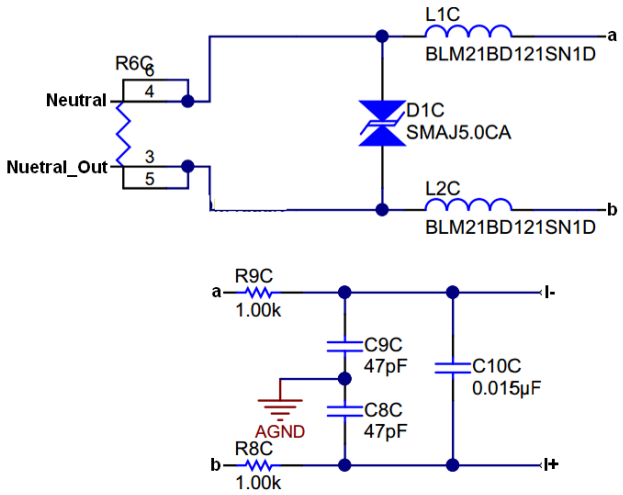

Fig. 5. Current monitoring circuit. This utilizes a shunt resistor for the purpose of using a voltage output to determine the current. The circuit is connect at wires a and b.

This current measurement design uses a shunt resistor in series with the neutral of the power supply and the neutral of the outlet. The resistor has a known resistance in our case 2m Ohms. A voltage is induced across the resistor when a current is present. Using ohms law we can deduce the Amperage that the device is using. The other components shown help to deduce noise and other unwanted signals using filters. The output of both the voltage and current measurement are differential with a positive and negative output. Also the output is so small that the MCU may have trouble reading the voltage. The use of an instrumentation amplifier is needed. This circuit will convert the differential output to a single sided output. Also the output will have a gain associated with it so the voltage output will increase for the MCU to get a proper reading. Figure 6 shows the instrumentation amplifier we used for the current.

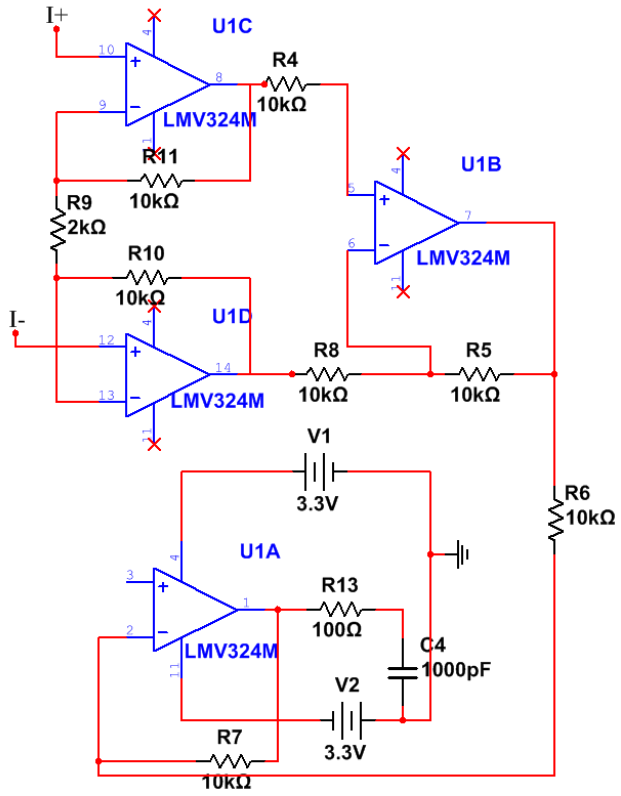

Fig. 6. Instrumentation amplifier used to convert a differential output to a single sided. Capacitors and other elements not shown for noise. The output is measured across C4.

IV. Relays And Relay Driver

Each of the three outlets will have a relay to control if the circuit should be off or on. This is controlled by the user which is mentioned in a greater detail later in this document. The relays have a 5V coil voltage which will switch the relay to a 120V line or to ground to be turned off. The relays will be controlled by a relay driver. The driver will be controlled by inputs from the MCU. Figure 7 shows a simplified block diagram of how the relays and relay driver are connected. The relay coil will have a constant 5V on one terminal. The relay driver will complete the circuit by sending the appropriate signal from the MCU to complete the 5V circuit which will switch the relay on or off.

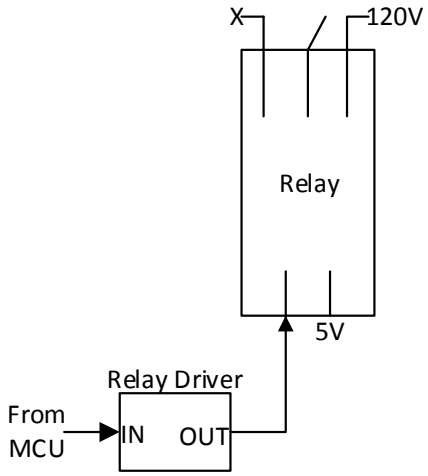

Fig. 7. Simplified Relay block diagram. The relay driver has much more than one input and output.

V. CC3200MOD MCU

When choosing a microcontroller for our design, a few factors had to be examined, cost, performance, and usability. The Texas Instruments' CC3200 is a standalone microcontroller with Internet-on-a-chip capabilities. The first of its kind, it is able to connect and communicate with servers to manipulate databases through an external access point or, instead, generating its own. With built-in Wi-Fi connectivity, security features, and helpful APIs, past experience is not required to ensure quick, relatively hassle-free development.

Internet-on-a-chip devices are especially practical for customers on a budget, allowing them to purchase a single device which provides the full range of connectivity options available without having to purchase or rely on the full functionality of home wireless networks. Having the ability for MCUs to communicate without being dependent on an access point also provides enhanced stability to the system. If a wireless network unexpectedly becomes inoperable with this microcontroller only a power source is needed to ensure full functionality.

With the extensive amount TI example code provided through their CC3200 SDK, our familiarity with coding other CC3200 microcontrollers, online example videos, and their documentation outlining the requirements for starting up and making the microcontroller fully functional, we decided a TI microcontroller was the best choice for our design. Eventually we decided on the CC3200 MOD shown in figure 8. We decided on this package for its ease of Wi-Fi hardware and easy to read reference designs for PCB implementation. The MOD is well documented so it was easy to research and implement.

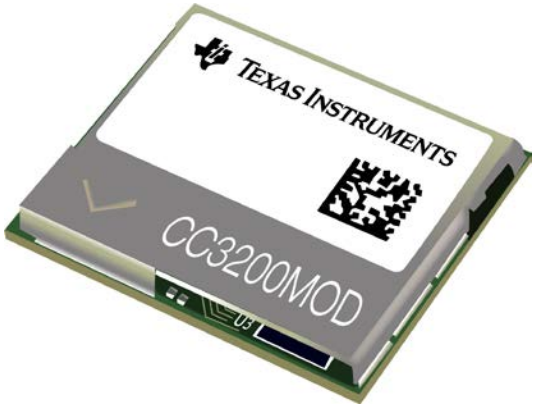

Fig. 8. CC3200 MOD. Chosen for ease of hardware implementation including the antenna needed for wifi.

VI. Embedded Code Flow

The process of programming an embedded device must be detail-focused to ensure nothing is overlooked which could hamper its functionality. When reviewing our design we divided its tasks into two stages boot-up and normal operation which will each be described in detail below.
The boot-up and configuration process requires the device to be powered through the electrical socket which will start the boot-up process. Once boot-up is initiated the microcontroller, in station mode, will connect to the server, and continuously check the database for an account number which is associated with its device ID. The user will sync the device to their account by entering the device's ID number into a field from their app and then clicking the app's 'sync' button. The device ID is printed on the outer shell of the device. During the syncing process, if the user would prefer the device to connect to a home Wi-Fi network and disable station mode, they may do so by entering the SSID and password of the network in the fields below the device ID field before clicking the 'sync' button. If connection to this network cannot be established the MCU will send a failed connection flag which will prompt a notification to the user on their app where they will be able to repeat the syncing and configuration process.
Once the device has connected to the server and it has been synched to a user account it will then enter into a normal operating mode which will loop infinitely until it is powered off or cannot establish connection with the server. The main purpose of normal operation mode is to measure and relay power measurements from the connected appliances to the server every 1.024 seconds, and to check the 'power on' status of appliances every 4.096 seconds. The 'power on' status determines whether power should be supplied to the specified appliance through its socket. If the 'power on' status is zero, the appliance will power off and if one, power will be supplied.
If connectivity is broken an interrupt will trigger and assign a value of either -1 to the intStatus flag variable which regulates the loop condition encompassing normal operation mode. Once the loop is terminated, the device will try to reconnect to its specified access point, and if unsuccessful, will return to station mode, and continue back into normal operation.

VII. Power Measurement Algorithm

For our design the cc3200 had to collect voltage and current samples from each socket, calculate V and I RMS values, true power, reactive power, apparent power, and the power factor. The algorithms used in the power measurement section of our device had to accommodate some of the setbacks and design of the CC3200. The analog to digital converter has a 12-bit resolution, but TI recommends to have at least a 14-bit resolution to accurately measure power. To make this possible the cc3200 sums 64 consecutive samples gathered from the ADC to get a single, more accurate measurement. With a sampling rate of 62.5 kHz for each pin, these summations, taken after sampling, will not severely distort or delay measurement samples. With a sampling time of 1.024 secs at 62.5 kHz, and having every 64 individual samples summed for greater accuracy, there will be exactly 1000 samples per pin to compare. The ADC can only measure voltages between 0V and 1.467V, to accurately translate these discretized values back into a closer to analog form, the discrete values of the ADC must first be divided by 4096 and then multiplied by 1.467. After samples are collected and processed through the ADC first V and I RMS are calculated through the equations listed below:

$$V_{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} {V_{IN+n}}^2} \quad (1)$$

$$V_{in} = \frac{661\left(V_{out} - V_{offset}\right)}{K_V} \quad (2)$$

$$I_{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} {I_{IN+n}}^2} \quad (3)$$

$$I_{in} = \frac{661\left(V_{out} - V_{offset}\right)}{K_I * R_{Shunt}} \quad (4)$$

Equations (1) to (4) use the Voltage offset from the instrumentation amplifier. The Voltage coming out needs to be subtracted from the offset. The K value is the scaling factor or the gain of the op amp circuit. Then from calculating the RMS values, the algorithm calculates true power by summing the factor of voltage and current samples with the same array index value and dividing them by total number of samples taken. Apparent power is simple calculated by multiplying the I and V RMS values together. To calculate reactive power take the square root of the difference of apparent power squared and true power squared. And finally after calculating these values the phase can be calculated by taking the arccosine of true power over apparent power. Equation (5) through (8) are needed to calculate true power shown as P in (5), apparent power shown as S in (6), reactive power shown as Q in (7), and finally power factor shown as PF in (8).

$$P = \frac{\sum_{n=1}^{N} V_{IN*n} * I_{IN*n}}{N} \quad (5)$$

$$S = V_{RMS} * I_{RMS} \quad (6)$$

$$Q = \sqrt{S^2 - P^2} \quad (7)$$

$$PF = \cos\theta = \frac{P}{S} \quad (8)$$

VII. Software Overview

Our project is going to have a large amount of software to upload to the microcontroller in order for the system to function how we would like. Also the database will house all of the data the system creates. The items we need to test are the Database and Server, and the Android App GUI.The database will be housed on the server and will collect all of the data the system creates. The data is sent to server via the microcontroller's wireless connection. We need to test and make sure any power data is being sent and housed correctly in the correct tables. The best way to do this is to only send one signal at a time. For example, we can create a voltage and a current on one of the microcontroller's inputs. That data should be routed to the correct location. Check the location in the database and make sure the data is populated. The android app GUI is going to take extensive and tedious testing to make sure every aspect is working correctly. We should test it on as many device as we can to ensure there is no problem with cross device implementation. A good way to test this system is see that it can handle every feature that it is programmed to accomplish. Then try to create errors that could crash the app and make sure error handling works correctly. This is a long and tedious process but will mitigate any issues that may come up a later time.

IX. Website –energyguard.tk

As part of the overall system, the website serves to connect the user to the system. The website gives the user the ability to access their account from their mobile or from a computer based on the preference and convenience of the user.

The website provides a user interface, so that the users can manipulate the settings of the system, and view the data returned from the other subsystems within the system. The user interface is shown in figure 9.

Fig. 9. The log in page for the web interface. Users will see this to log on to the web portal.

The other subsystems will complete the necessary calculations and tasks to realize the data that will be displayed on the website. The website only takes care of calculations over a period of time. The website displays energy used over the past hour, past day, and current month. The website sends requests to the server to retrieve the data from the database, and displays this data to the user. The website also sends the user configured settings to

the server. The server will then save these settings to the database. The functionality of the Power Strip will depend on these user settings

X. Data Flow

This section will describe and explain the flow of the data in regards to the Above Entrance Mounted Device (AEMD). The main objective of this device will be to sense a change in the occupancy count of the room. The objective will not be to keep track of the occupancy count since this will be done by the database. However, the objective will be either to increment or decrement the value that is stored in the database.The database, and the web application will only need to know about the data that results from the AEMD. Therefore, the data flow in regards to the AEMD will begin at the sensors that are attached to the microcontroller on the AEMD. Once the sensor senses a change or the sensor is triggered, the signal will be sent to the microcontroller on the AEMD.
The data flow will then progress from the sensors to the microcontroller where the microcontroller will perform calculations on the signal, in order to understand whether this was an increase or a decrease in the occupancy count of the room. After the microcontroller knows whether to increment or decrement the occupancy count of the room, this data will be sent from the microcontroller to the web server. The web server will then pull the occupancy count that is already stored in the database, and either increment or decrement the value. Once the value is updated, the server will then store the value in the database again. This value will be used throughout the web application, and the Power Strip, as the functionalities will differ based on the value of the occupancy count of the room.

XI. Software Development

Development of user software can be divided into two sections, front-end and back-end development, and with this distinction in classification, languages are also classified by which best accomplish these tasks. The front end was done in HTML, CSS, and JavaScript. The majority of the HTML and CSS was built using a website builder called Wix. Wix provides an HTML editor to its users and allowed us to create the front end quickly, as well as create a user interface that is attractive to users. The HTML and CSS source code created was then taken, and placed on out apache tomcat server, in order to establish the connection between the front end and the back end. Initially, we had developed the front end using our own HTML, and CSS source code, but decided to use Wix for a few different reasons. The main reason was that users like an attractive user interface. Wix provided a means to achieve this attractive user interface. Wix was also chosen because of the quick development time, and the simplicity of taking the HTML source code and placing it on the team apache tomcat server. Because the code was in HTML and CSS, this also allowed the website to be easily accessed anywhere, including other devices like cellphones and tablets. The user experience was an important aspect of the system as a whole, and especially the website because most of the interaction between the system and the user is done through the website. The server side development was done using JavaServer Pages (JSP), where SQL queries were executed, and results were sent back to the JavaScript on the front end. The Chart JS framework was also used in the implementation of the charts that display the energy used.

XII. User Experience (UX)

The first aspect of the user experience is the amount of data that is displayed on each page, and how much effort the user has to put in to progress through the different flows in the web application. The page flow of the application will be vital to the user experience to ensure that the user is not wasting his or her time. As part of this, there is only one main page other than the login page, so the user does not need to worry about anything else. After logging in, the page displayed to the user will display all the data the user needs, and creates a format that is attractive to the user. This page displays the three sockets in a list format, and the sockets have an associated icon because icons improve the user experience by decreasing the necessity to read text. Each socket in this list will also have an “On” and “Off” button that alternate colors in order to represent which is on and which is off. The socket will also have the current power reading if the socket is turned on, otherwise, it will display a message saying the socket if off. Each socket will also have an associated chart that shows the average energy used over a specific period of time. The user can then choose to view the energy used over the past hour, past day, or throughout the current month.

XIII. Database

The database will be depended on by each of the other subsystems in the system. The database is important because it provides a means of consolidating all the data related to the subsystems, rather than having each of the subsystems having its own data separately. The database will be divided into tables and columns in order to keep each value organized with a unique key for each row entered into the database. The database contains a table that is used to keep track of the user’s data when creating an account. The database also contains a table that keeps track of the sockets associated with the power strip that is associated with a specific user account. This table will also

contain the power readings that were received from the power strip based on the user and the socket. This table will also keep track of when the socket was last turned on. The third table in the database design is used to keep track of all the other times the socket on a specific power strip was turned on and off. So once the user turns a specific socket off from the website, this third table is populated with when the socket was turned on and when it was turned off. These values are then used to calculate the amount of energy used by the socket over a specific period of time.

XIV. Server

The microcontrollers will need to transfer data to be displayed to the user in the website, and vice versa; the website will need to transfer data to change the functionality of the microcontrollers because the functionality of the microcontrollers may vary based on the settings specified by the user through the website. The server serves as the central hub of the project. It receives data from the microcontrollers and stores the data into the database. The server also retrieves data from the database and sends it to the microcontroller, when necessary. This data is saved into the database after being sent to the server over Wi-Fi. The importance of having this central hub is that it consolidates all of the data, and consolidates the network links between the main components of the project. The website sends and retrieves data to and from the microcontrollers, and the microcontrollers send and retrieve data to and from the website through the use of the server and the database. The server is configured to redirect users to the login file in the correct directory, and handles the requests and responses. For example, the JavaScript program can invoke a post method which sends a request to the server. The server handles this request by executing the correct file that is referred to, as well as passing any parameters specified. After the server executes the correct file, it returns a response back to the JavaScript code where the response can be parsed and used to display data to the user. In Energy Guard, the server receives a request from the JavaScript program, executes the corresponding JavaServer Pages (JSP) file, which executes a SQL query to the database. The results of the query are then parsed using the Java ResultSet object, and sent back to the JavaScript by the server. The JavaScript then parses the response, and does the necessary calculations before displaying the data to the user.

VII. Conclusion

We believe that energy guard can offer simple home energy management at a lower cost than most existing systems. Cost is not the only factor to consider. The ability to turn off certain devices remotely will appeal to users who may forget to turn off a certain appliance. Also the fact that historic data can be viewed also to see how much energy has been used over a period time will be a great help to those trying to save on energy usage.

Acknowledgement

The authors wish to acknowledge Boeing and Leidos for providing funding for our project. Also we wish to thank the faculty of the University of Central Florida for supporting us with answering any questions we may have. Specifically we want to thank those who have agreed to be on our senior design committee.

Biography

Omar is a Computer Engineering Major that is currently working for Florida Blue. His plans are to continue working for Florida Blue as a Software Engineer.

Spencer Sullivan is graduating from the University of Central Florida in April of 2016. He plans to graduate with a Bachelors in the field of electrical engineering. He has currently been an intern with Siemens Energy since 2012. He will continue with Siemens Energy after he graduates.

Gabriel Holland is a Senior computer engineering student at the University of Central Florida. With an interest in cyber-security and embedded systems, he plans to continue his education by earning a aster's degree in computer engineering with a focus in embedded systems after graduation

Tyler Ensey is a senior of the electrical engineering program at the University of Central Florida. He plans on continuing working for NASA after his graduation.

REFERENCES

[1] Delta Electronics, “Power Entry Module EMI Filters”, from http://www.deltaww.com/filecenter/Products/download/04/0406/power_entry_filter/AR.pdf

[2] Texas Instruments, 2014, “Three-Outlet Smart Power Strip”, from http://www.ti.com/lit/ug/tidu453/tidu453.pdf