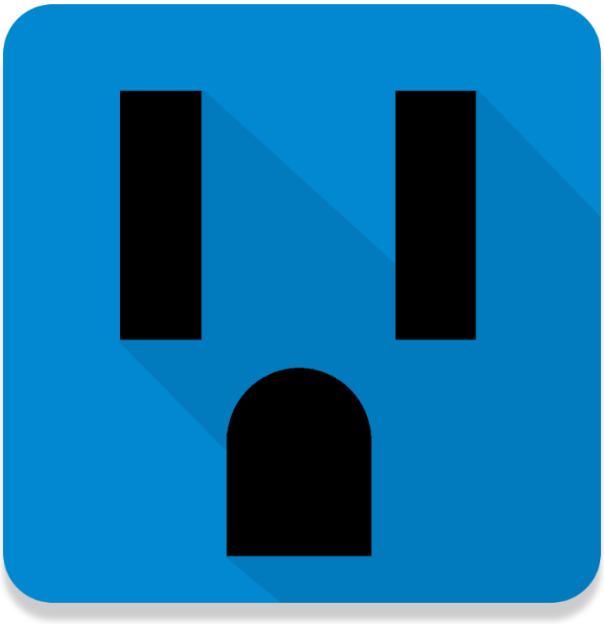
Smart Power Strip



Dannie Alfaro
Advisor: Bridget Benson, Computer Engineering
Cal Poly State University San Luis Obispo
Electrical Engineering Program
June 2015

Abstract

The Smart Power Strip revolutionizes all common power strips. Users have the ability to remotely toggle the individual outlets on the Smart Power Strip from a smart phone app and can also monitor how much energy the devices connected to the Smart Power Strip consume.

Acknowledgements

I would like to thank my advisor, Bridget Benson for always giving guidance on how I should implement specific parts of my project and for being readily available to answer any questions I had.

I would also like to thank Joshua Logier, for his knowledge in Android development and for helping me develop the app for the Smart Power Strip.

Lastly, I would like to thank my parents for supporting me in everything I do and for paying for my education.

Table of Contents

Table of Figures

Table of Tables

Introduction

Home automation has recently become very popular due to its much higher affordability and simplicity. Before the widespread use of smart phones, people viewed automating homes as an unnecessary luxury, but now smart phones make it very easy to interact with new home automation systems. Unfortunately some of these home automation systems can still be very expensive and not all electronic devices can be automated.

This project attempts to create an alternative form of home automation that is inexpensive and supports a wider range of electronic devices. The smart power strip created for this project can support any electronic device that connects to the standard North American outlet.

This smart power strip is wirelessly controlled by a smart phone app that allows users to remotely toggle individual outlets as long as the smart phone and the smart power strip are on the same network. Most people already own smart phones, which removes the need for extra hardware to control the smart power strip. As well as being able to toggle the outlets, users will be able to see on their smart phones how much energy devices connected to the smart power strip are using at any given time. Ideally this will make people more aware of how much energy they are spending at any given time and help people consume less energy, while saving money on electricity.

Requirements

A project like the Smart Power Strip does not require very expensive hardware to implement so cost was not a major liming factor to picking a design. The most important factor for picking a design was keeping the system as standalone as possible while still keeping a high amount of functionality. Taking these things into consideration while choosing a design, these are the requirements I felt were most important for keeping a high amount of functionality.

- Houses more than 2 outlets
- Communicates over a Wi-Fi connection
- Consumes less than 50kWh per year
- Use parts that are inexpensive and easily replaceable
- Toggles individual outlets through smart phone app
- Works without any setup

Secondary Goals:

- Implement timing feature for toggling Smart Power Strip
- Allow users to toggle and monitor Smart Power Strip from remote networks

Background

The idea for the Smart Power Strip stemmed from my interest in home automation. I did a lot of research on existing smart devices used for home use before I came to the idea of the Smart Power Strip.

I wanted to create a smart device that was extremely practical, a device that would get used every day, and something that people would be interested in buying. I wanted my smart device to be easily understood and that people would understand its functionality without any explanation. So I decided that there is no smart home device that is more practical than a smart outlet. I took this idea and stretched it a bit further because no one ever uses only one outlet. From here the idea of the Smart Power Strip was born, this way people would have multiple smart outlets confined to a single package.

Most smart devices utilize users existing Internet connections, either wired through an Ethernet cable or wirelessly through Wi-Fi. The Smart Power Strip was designed to operate off of Wi-Fi to minimize the need for extra cables, which would cost extra money and require some installation by the user. The tradeoff to not using Ethernet is that the networking becomes a little more complex. The networking portion of the Smart Power Strip must allow for both the server and client to send and receive data.

All smart devices use some sort of CPU that handles the tasks that people or mechanical systems normally do. For my design I wanted to use a CPU that has over 16 GPIOs, interfaces with peripherals through USB, and can be programmed in a wide variety of languages. I decided to use a Raspberry Pi B+ because it is basically a desktop computer that is the size of a credit card. The raspberry pi is extremely customizable and incredibly powerful.

Networking

Networking is the ability to be able to link multiple devices together to be able to exchange information and operate interactively. Networking can be done through wires, mainly Ethernet cables, or wirelessly through many different kinds of network protocols.

In my Smart Power Strip, I am using networking to send information from a smart phone to the raspberry pi that tells the pi which outlets to toggle, to request information from the raspberry pi about how much energy devices connected to the outlets are using, and to send data from the raspberry pi to a smart phone about energy consumed by devices connected to the outlets. The only information needed to make this networking work is the raspberry pi's IP address which is static. The raspberry pi always waits for a device to communicate with it so the device's IP connecting to the IP does not matter as long as they are on the same network.

Transmission Control Protocol

The most common networking protocol is TCP/IP which are two distinct protocols but are often used together. TCP (Transmission Control Protocol) breaks a message into smaller packets and then transmits them over the Internet. The packets are later received at another destination specified by the IP (Internet Protocol) and the packets are put back together until they form the original message. The great thing about TCP is it has built in error checking. The sending end numbers every packet of data and sends how many packets it is sending to the receiving end, this way the receiving end knows what order the packets of data should go in and how many packets it is expecting. The transfer of data will continue until the receiving end receives all the packets of data in the correct order, only then will communication end between the transmitting and receiving ends.

Universal Datagram Protocol

For the Smart Power Strip I chose to use UDP (Universal Datagram Protocol) over the very popular TCP/IP. There are two reasons for why UDP is better for my application than TCP. First of all UDP is much easier to implement in software than TCP. The second reason is that UDP is much faster than TCP, which is very desirable for updating data on the smart phone in nearly real time. The reason for UDP being faster than TCP is that UDP does not have any built in error checking like TCP. Also unlike TCP, UDP is not connection based. UDP can send a stream of packets to another device and that would be the end of the communication. There is no guarantee that the packets sent will reach their destination or hold valid data but it is not a big deal for the Smart Power Strip. Worst case scenario a button on the smart phone app would have to be pressed more than once to resend the data.

Figure 1 below displays a data flow diagram for UDP that shows the basics of how UDP functions.

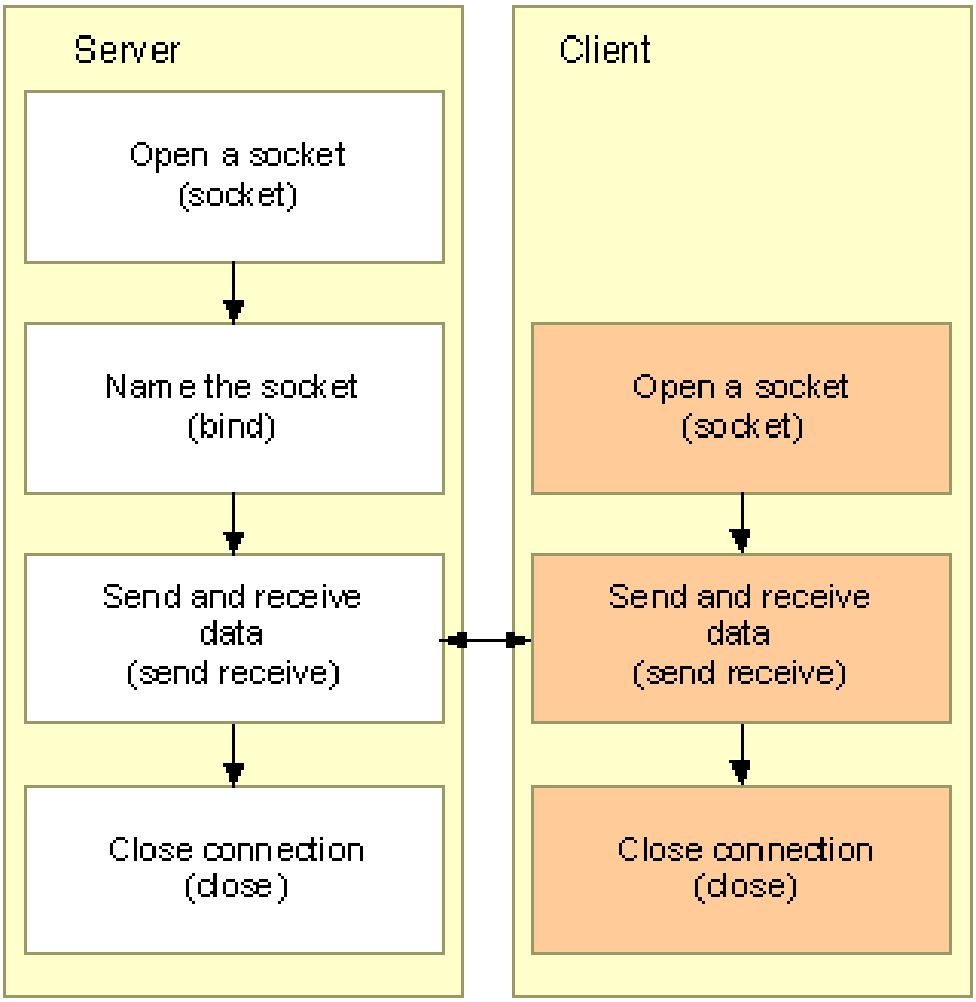


Figure 1: UDP Data Flow Diagram

Processing

As stated before a Raspberry Pi B+ is the main processing unit for the Smart Power Strip. The pi is the brains of the Smart Power Strip and this is where almost all the calculations and processes executed in software happen. The Raspberry Pi B+ contains a 700 MHz low power ARM1176JZFS applications processor. This processor has much more processing power than what is needed for the Smart Power Strip, but the pi is cheap and very easy to work with.

Design

The hardware and software used in this project are equally important to the functionality of the Smart Power Strip. The hardware handles reading currents from devices connected to the outlets, records data from outlets, and physically toggles the outlets. The software handles the networking, data parsing, and graphical user interface. The individual hardware parts and software used will be further described below.

Hardware

This section provides a list of the hardware components used in the smart power strip and then provides images/circuit diagrams of each of the main components in the system while discussing why these particular components were selected. The section concludes with a block diagram showing the integration of all the components.

Hardware Components

4	Leviton T5320-W 15 Amp, 125 Volt, Duplex Receptacle
4	SCT 013-030 3.5mm Output Split-core Current Transformer
1	Raspberry Pi Model B+
1	emonTx V3 – Electricity Monitoring Transmitter Unit
1	JBtek 8 Channel DC 5V Relay
1	KEDSUM STC Download Cable USB 2.0 to TTL 6PIN Serial Converter
1	Ralink RT5370 WiFi Adapter
1	8GB Samsung MicroSD Card
1	2.5A USB Power Supply
1	9VAC, 1.11A AC-AC Power Supply

Hardware Considerations

This section explains why the above hardware was chosen for the Smart Power Strip.

Leviton T5320-W 15 Amp, 125 Volt, Duplex Receptacle

These duplex receptacles are the standard outlets used in North America. They are cheap, durable, NEMA compliant, and meet the 2008 NEC requirement. All receptacles sold now are up to modern codes so the brand really doesn't matter.

SCT 013-030 3.5mm Output Split-core Current Transformer

This current transformer is suitable for measuring and monitoring AC currents up to 30 Amps. It is ideal for the Smart Power Strip because devices connected it to is shouldn't be pulling anywhere near 30 Amps. Also this current transformer is very compact and has a wide enough opening to fit the 18AWG wire going through it. The 3.5mm output is ideal for being used in combination with the emonTx V3 energy sensor node, but the caps of the current transformers outputs can also be easily removed to use internal wires on custom built circuits or hardware.

Figure 2 displays what the output of the current transformer looks like and how the voltage is regulated by two back to back zener diodes.

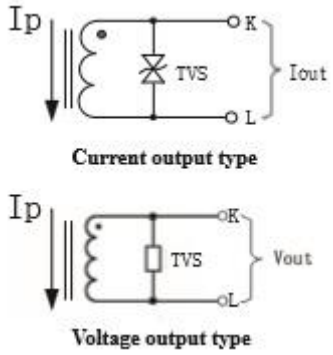


Figure 2: Current Transformer Output Schematic

Raspberry Pi Model B+

The Raspberry Pi Model B+, seen below in *Figure 3* was chosen to be the main processing unit for the Smart Power Strip because it is very small, cheap, incredibly powerful, and highly programmable.

The pi has a surface area roughly the size of a credit card and is almost completely flat except for the GPIO, USB ports, and Ethernet port. Having the same capabilities as a desktop computer running Linux, the pi is a steal costing only $35. This was probably the cheapest option for the amount of processing power a board can have that is the same size. Since the pi is running a Linux kernel it can run all sorts of files including PHP and Python files which are both used for the Smart Power Strip.

Another key feature that was taken into account when choosing the pi is that is has 4 USB 2.0 ports. This is important because the Smart Power Strip needs at least 2 USB ports, one for the serial connection between the pi and the emonTx V3 and the second port for the wireless adapter. Having the extra ports is highly desirable because then a USB keyboard and mouse can also be connected to easily program and alter settings on the pi.

The pi parses the data it receives through a serial USB connection and prepares it for sending through UDP in a Python script. The pi is also configured to be an apache web server whenever it turns on, so it is always ready to receive data from any device as long as they are both on the same network.

The pi's GPIO, seen below in *Figure 4*, drives an 8-channel relay that toggles the individual outlets on the Smart Power Strip. The GPIO on the pi is controlled by a PHP file on the web server that waits to receive the specific address of a GPIO and then toggles its state.

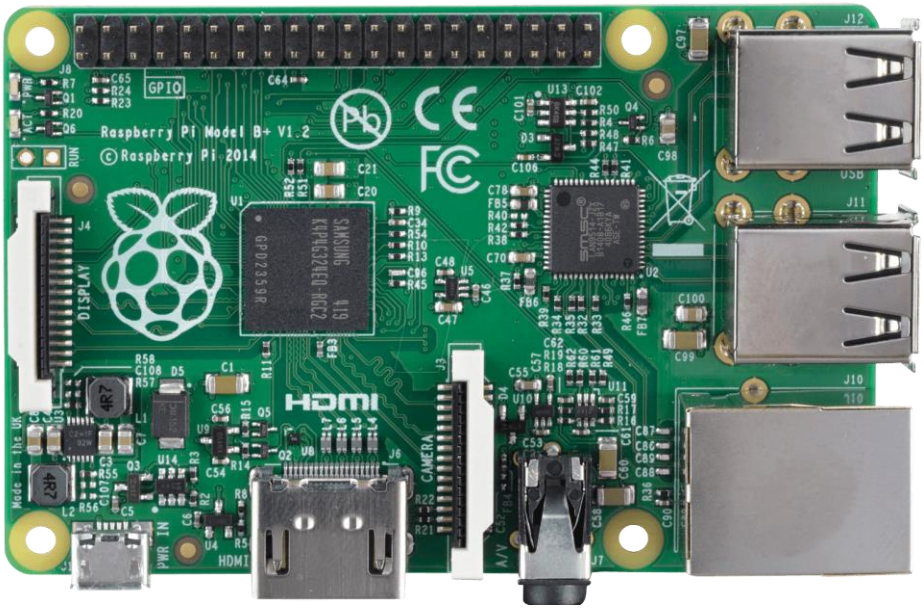


Figure 3: Raspberry Pi B+ Board Layout

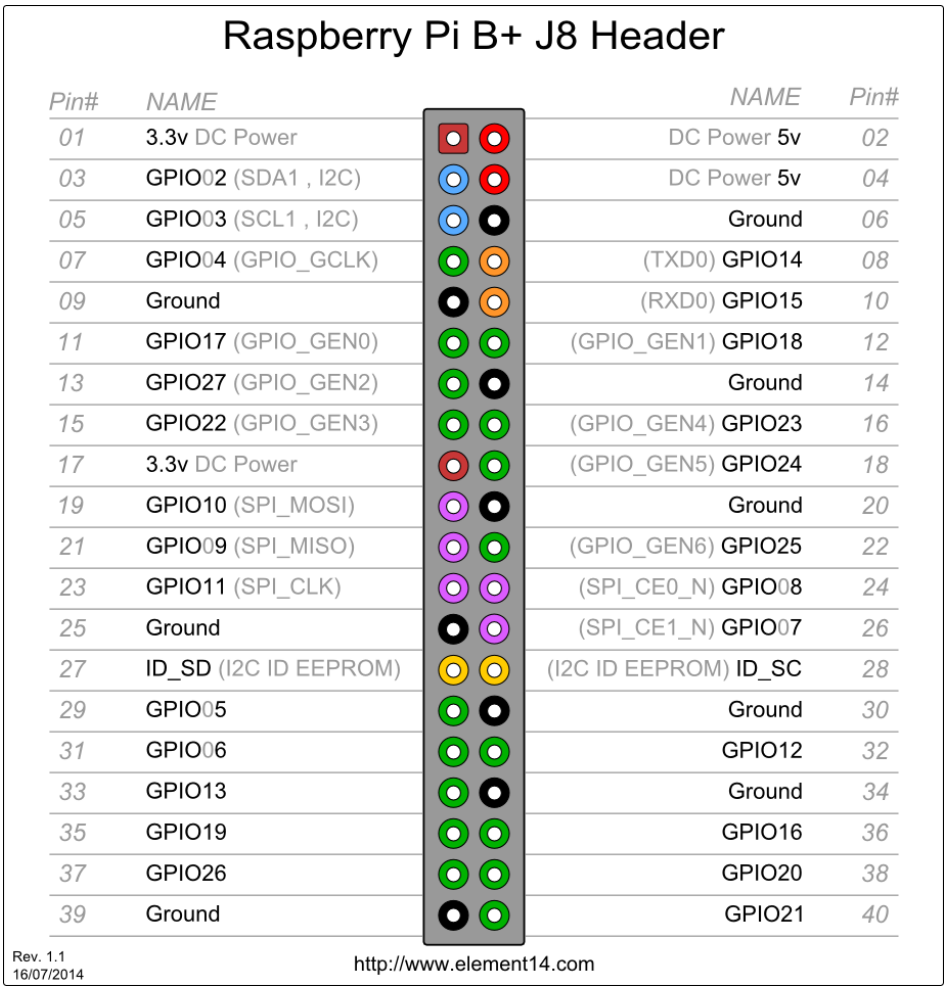


Figure 4: Raspberry Pi B+ GPIO Pinout

emonTx V3 – Electricity Monitoring Transmitter Unit

The emonTx V3 was probably the least justifiable component to be included in the Smart Power Strip. It is much larger than the pi and also costs much more than the pi, but it is very easy to interface with the 3.5 mm outputs of the current transformers. A schematic for the inputs of the emonTx can be seen below in *Figure 6.*

The board runs of an ATmega328 that a lot of Arduinos also run off of so it was very easy to program this board using the Arduino IDE. The layout of the emonTx is figured below in *Figure 5.* The emonTx V3 also comes with its very own C library with many function calls designed to interface with the current probes. The emonTx board has four 3.5 mm inputs, so there is no need to take apart the current probes to interface them with the emonTx. The board has native wireless capabilities, but they are not required for this project because the emonTx and pi sit next to each other in the Smart Power Strip enclosure. Therefore it is easier to just transfer data from the emonTx through the UART to one of the pi's USBs. The only thing that matters then is the address of the USB port and the baud rate.

The emonTx V3 is not necessary for this project because the Raspberry Pi can parse the data from the current probes, but that requires extra circuitry to amplify the current values read by the current probes. The 3.5mm inputs on the emonTx and the emonTx library designed for the ATmega328 are why this part is included in the Smart Power Strip.

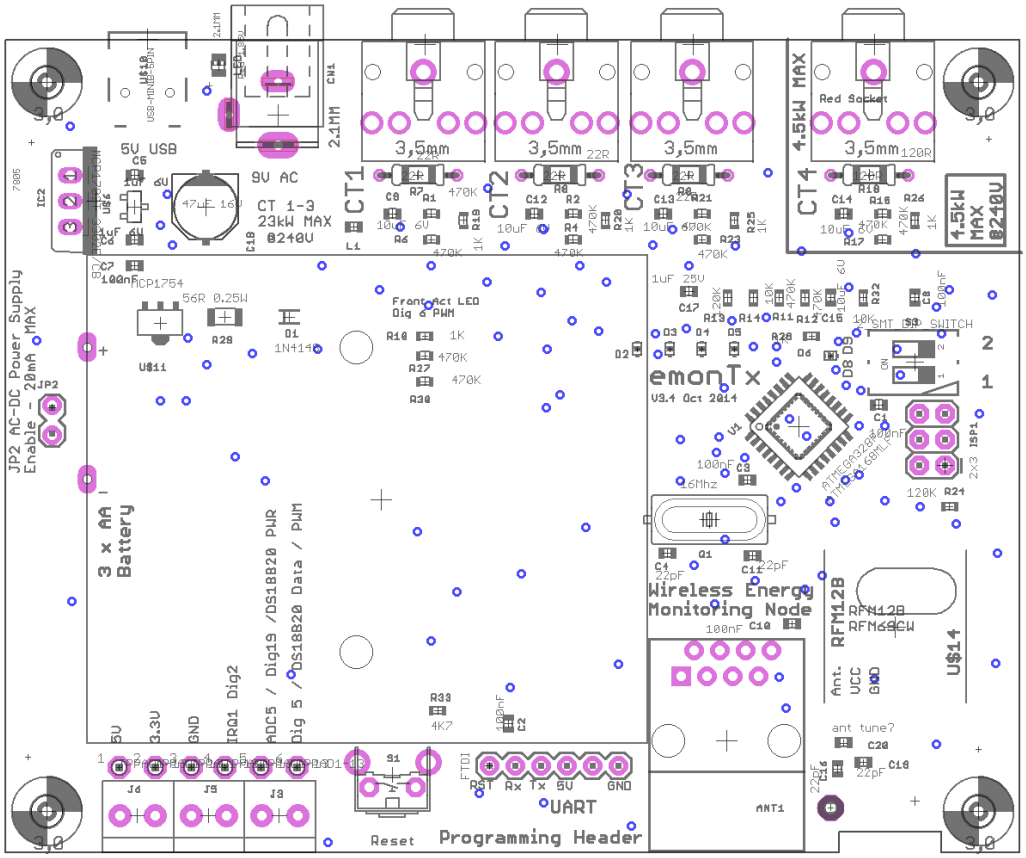


Figure 5: emonTx Board Layout

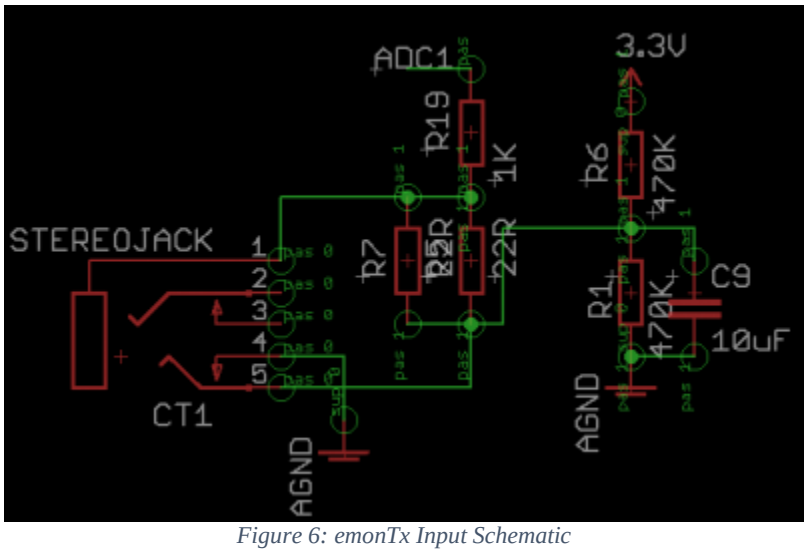


Figure 6: emonTx Input Schematic

JBtek 8 Channel DC 5V Relay

Some sort of relay is required for the Smart Power Strip to be able to toggle individual outlets. The relay also acts as a buffer between the high currents running through the outlets and the pi controlling the separate channels of the relay. The JBtek 8-Channel Relay is an active low relay that runs off of a 5V source, which the pi can easily supply from its GPIO. A schematic for the 8-Channel relay can be seen below in *Figure 7*. Unfortunately there are 2 un-used channels which take up a tiny bit of extra space in the Smart Power Strip's enclosure, but they only other option was a 4-channel relay.

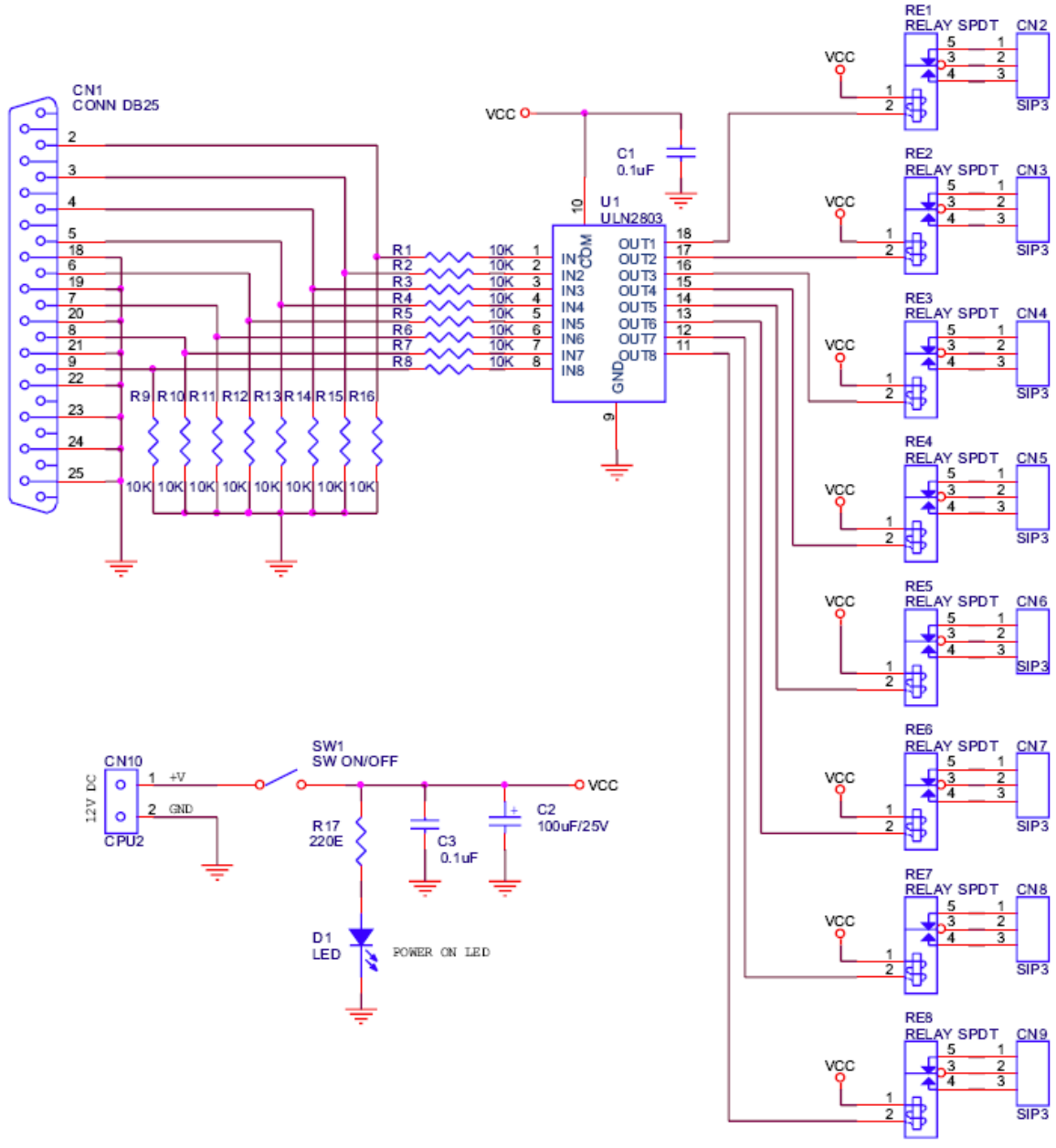


Figure 7: 8-Channel Relay Schematic

KEDSUM STC Download Cable USB 2.0 to TTL 6PIN Serial Converter

This TTL to 6 pin serial converter is necessary for programming new sketches onto the emonTx. It is also required for the serial data transfer between the emonTx's UART to one of the pi's USB ports. A nice perk of this particular converter is that I have the option to choose whether the converter outputs 5V or 3.3V on its VCC pin. Below in *Figure 8* is a schematic for the USB to TTL converter.

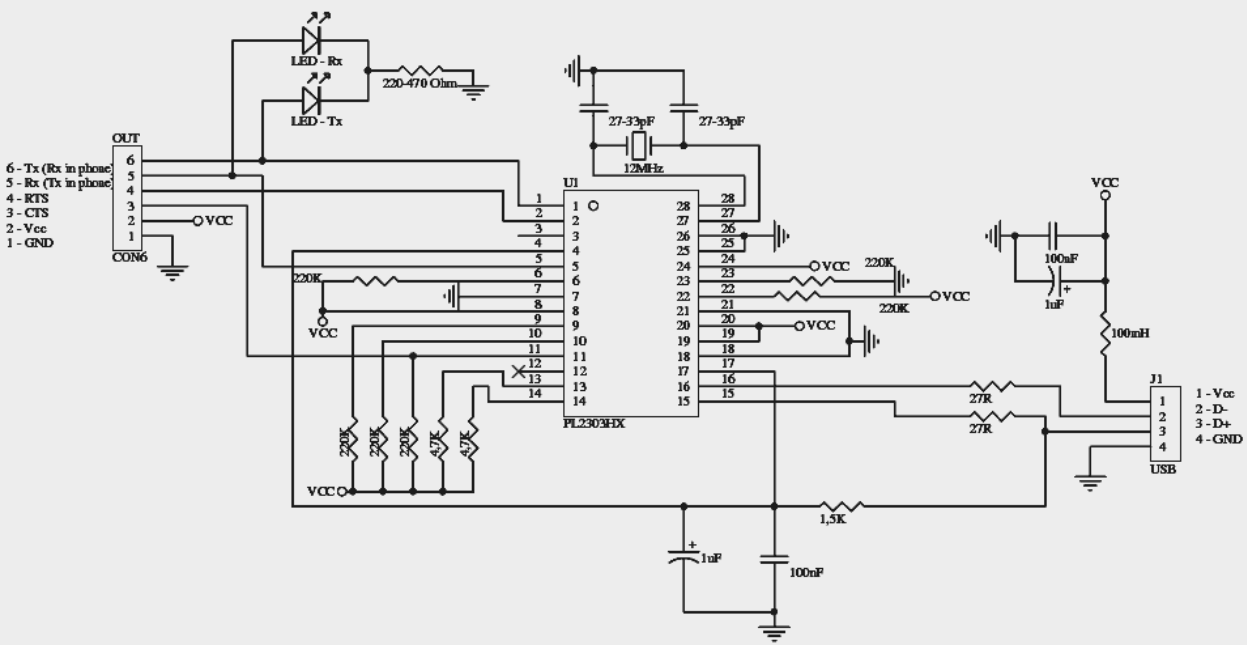


Figure 8: USB to TTL Converter Schematic

Ralink RT5370 WiFi Adapter

A WiFi adapter is necessary for the pi to be able to send and receive data. This particular chipset automatically installs drivers on the pi so it works right when it is plugged in. Users also have the possibility to turn off the low power state on this chipset, which is ideal for the Smart Power Strip.

8GB Samsung MicroSD Card

The SD card is used to boot the pi's OS as well as to store all the code that was written for the pi. 8GB is more than enough storage for storing the data for the Smart Power Strip since the old data gets deleted once new data becomes available.

Power Supplies

The pi and emonTx each require their own power supplies. The emonTx requires an AC power supply to be able to read large AC currents so the 9VAC, 1.11A AC-AC power supply was necessary for the desired functionality of the emonTx.

Overall System

Figure 9 represents a very high level block diagram of how all the hardware is integrated into the Smart Power Strip. For a more detailed diagram of the individual interconnections of the Smart Power Strip's hardware refer to *Appendix.*

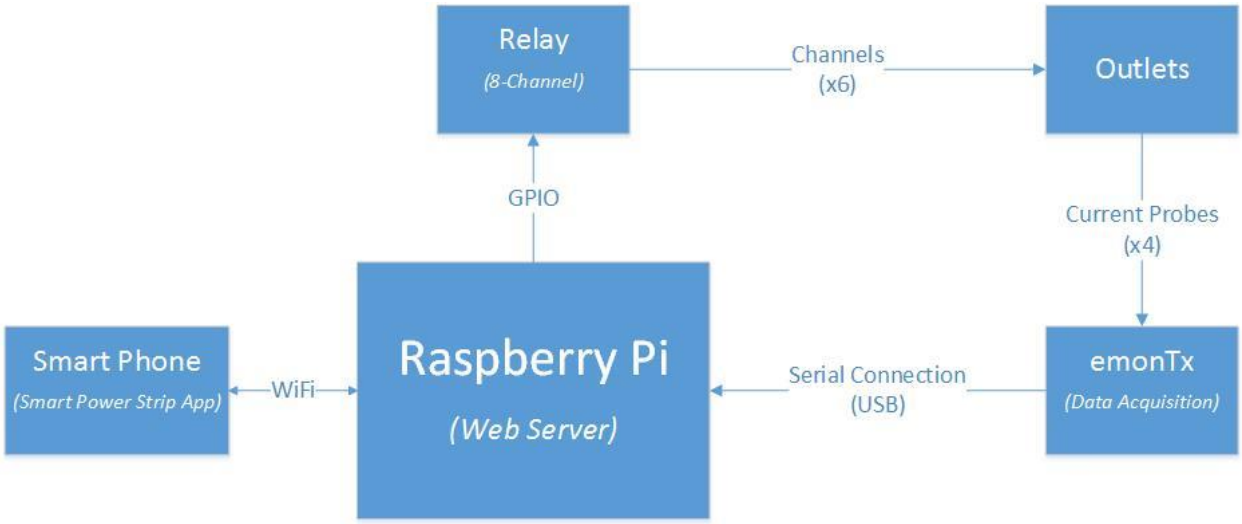


Figure 9: Smart Power Strip Hardware Block Diagram

Software

There were various types of software that were used for the different components in the Smart Power Strip. The code for the Raspberry Pi was written in PHP and Python in an ordinary text editor, the code for the EmonTx V3 was written in C++ in the Arduino IDE, and the code for the Smart Power Strip app was written in Java in Android Studio. This section will describe the software used for the individual components in the Smart Power Strip.

EmonTx V3

The software required for the emonTx is the most simple and straight forward out of all the software required for the Smart Power Strip. As stated before the emonTx runs off of an ATmega328 chip that is found on most Arduinos so the Arduino IDE was used to load sketches onto the emonTx.

The emonTx has a lot of documentation that can be found online so it was quite easy to find example sketches written to read data from the current probes and transfer it serially. The emonTx has its own set of libraries on GitHub written in C++, and the GitHub repository for the libraries is frequently updated. Since the libraries were written in C++, I chose to write my program for the emonTx in C++ over C so I could fully utilize all the functions in the libraries.

The logic flow of the data written for the emonTx is very simple. To begin there is a setup sequence where an object for each of the current transformers is initialized and calibrated with a particular phase shift value and current calibration value. Then the emonTx enters an infinite loop where every .5 seconds the emonTx serially writes whatever values it has calculated for the apparent power. Data is sent for each of the 4 current transformers and it is parsed on the Raspberry Pi. This code also makes sure that garbage data due to small currents being read on the probes is written as zero to the Raspberry Pi.

The current transformers only read current but there is a function designed to convert the current readings into power called calcVI(). The emonTx is constantly taking current readings but as soon as the calcVI() function is called, it takes an average of the currents recorded. This is the value that the pi later receives. The emonTx is also capable of reporting real power, voltage, current, or power factor.

Figure 10 is a visualization of the text above. It represents the flow of data in the emonTx.

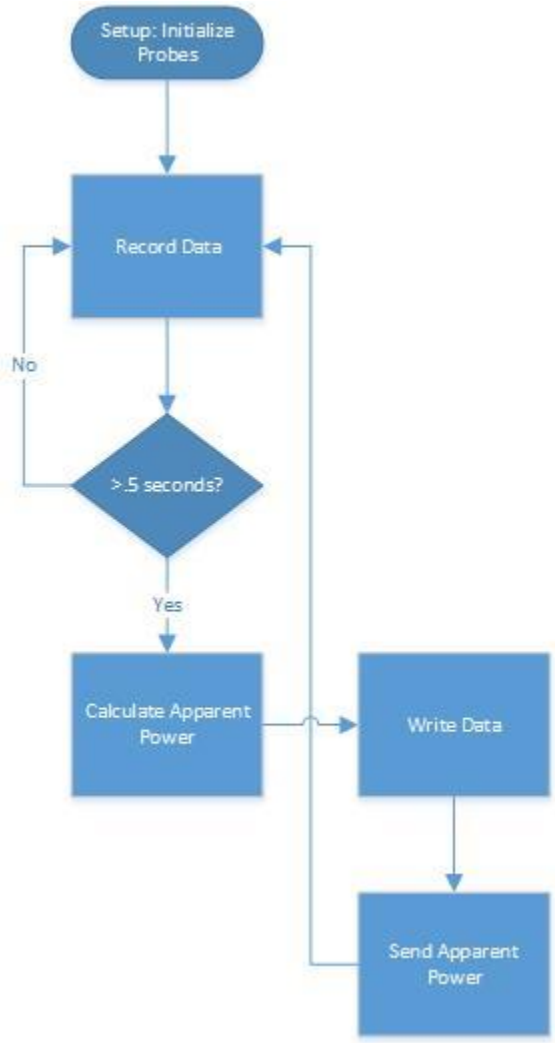


Figure 10: emonTx Software Flow

Raspberry Pi B+

The Raspberry Pi required a lot of different types of programming to be done in order to do what the Smart Power Strip required. There are four major tasks that the pi does so this section will be broken down into four subsections: webserver software, data parsing software, timers software, and networking software.

Web Server

Using the Raspberry Pi as a web server is what allows all the networking between the Smart Power Strip app and the pi to take place. The primary function of the web server is to be able to communicate with the GPIO on the Raspberry Pi wirelessly, but before that could happen a special library had to be installed on the pi to enable simple communication with the GPIO.

Users can always interface with the GPIO on the pi, but a special library called Wiring Pi condenses multiple lines of code into a single command line. This makes the process of communicating with the GPIO a little slower than a C program, but the delay is not that important for the Smart Power Strip. The syntax for a command using this library is the term "write" followed by the pin I was to address and what state I want it to be in. An example can be seen later on in the code appendix.

Once the Wiring Pi library was installed the only thing left to do to set up the web server is install the Apache Http Server. This can be done by entering a single line in the command prompt terminal. Then to control the outlets through the web server, a PHP file was created which specified the addresses of the GPIO I wanted to access. From the Smart Power Strip app side of things, the only thing I need to know is the location of the PHP file on the webserver and which GPIO pins I want to toggle.

Figure 11 below summarizes the individual roles of the web server, php file, and app.

During the testing phases of the Smart Power Strip, there was a JavaScript file that allowed users to control the outlets from a web app. This was later removed and changed to be included in the functionality of the Smart Power Strip app using native android modules.

Figure 11: Web Server Block Diagram

Parsing Data

The Raspberry Pi interprets and saves the data it receives on one of its USB ports from the emonTx. The script that runs and parses all the data on the pi is written in Python. This script requires the address of the USB port on the pi, the baud rate, and how often I want the pi to take a reading from the USB port.

The pi checks to see if there is data every .5 seconds and if there is data it splits the data it receives into four separate float arrays. Then it checks to see if the value put into the array is zero or non-zero. If it is zero, the pi doesn't count it as part of a running average. If the value is non zero, it adds one to a count keeping track of non-zero values. After twenty readings have been taken from the USB port of the pi, it takes the sum of all the values in the float array and divides it by the number of non-zero values recorded. It does this four times, once for every float array that represents a current probe. Then the pi reads from a text file whatever the results from the last run were and adds the new values to it. The old value in the text file is then replaced by the sum of the old value and the new value. At the end of this process the float arrays are cleared and the counts are reinitialized.

This data parsing Python script is launched in the boot sequence of the pi so the pi is ready to receive data as long as it is turned on. *Figure 12* is designed to help visualize how the pi parses the data in receives.

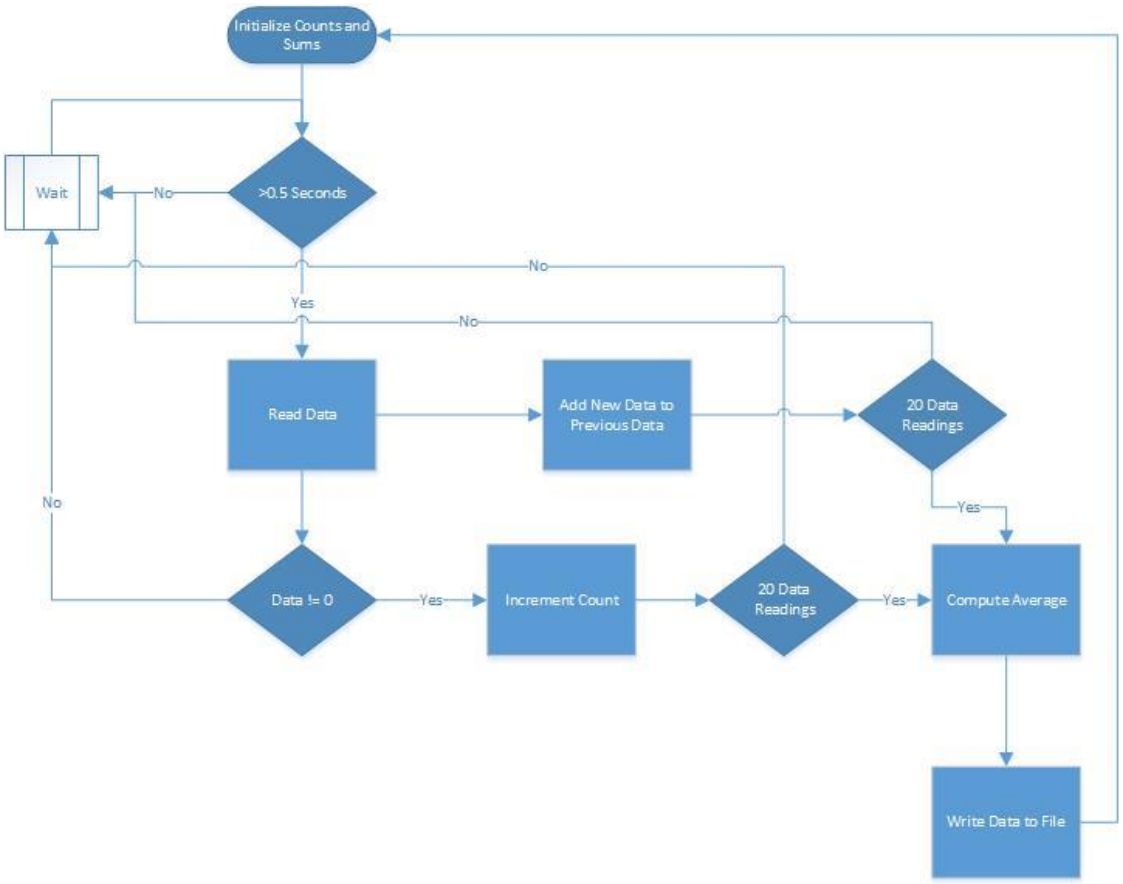


Figure 12: Python Data Parsing Data Flow Diagram

Timers

The Raspberry Pi also handles the timers that are used in the Smart Power Strip. The Smart Power Strip app allows users to specify a specific time to toggle any outlet and the Raspberry Pi makes sure this happens at the specified time.

Acquiring the data needed to know when to toggle the outlets is very easy since the pi already receives data regarding which outlets to toggle. Once that data is received the pi creates an individual thread for each timer and that thread goes to sleep until the timer goes off. The thread is deleted if the timer is reset or cancelled by the user setting up a new timer or cancelling the timer on the Smart Power Strip app.

Below is a software flow diagram of how the timers are handled on Raspberry Pi (*Figure 13*).

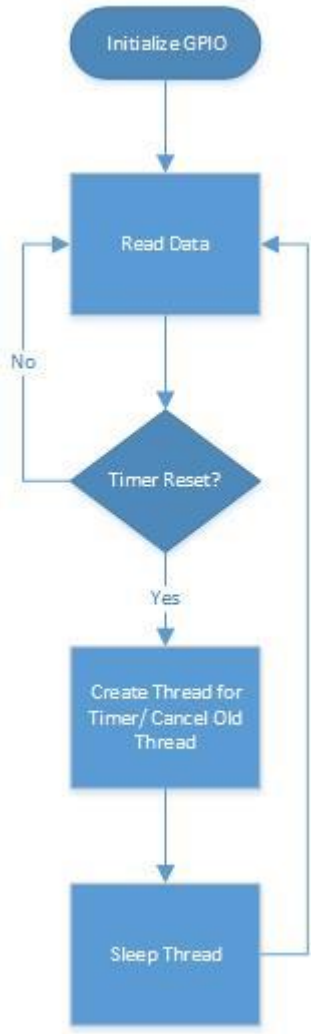


Figure 13: Timers Software Flow Diagram

Networking

The last bit of software that the Raspberry Pi needs to run is the networking code. As mentioned in the networking section earlier, the pi communicates with a smart phone wirelessly through UDP. The networking portion of the Smart Power Strip is also a Python script that constantly runs as long as the pi is on.

Acquiring the IP address of the Raspberry Pi is the very first thing this script does. The pi's IP address is necessary for binding the address to a port where the Smart Power Strip app will be communicating with the pi. Once the IP address is bound to a port, the script checks to see if there is any device requesting data on the port. If there is a device requesting data on the port, the pi opens the file where the data parsing Python script has been saving data, reads the data currently in the file, and sends that data on another port to the device requesting data.

Once all of that takes place the file is closed and the pi stops sending data until there is a new request for new data. Then the previous steps happen all over again. A summary of how all of this happens can be seen below in *Figure 14.*

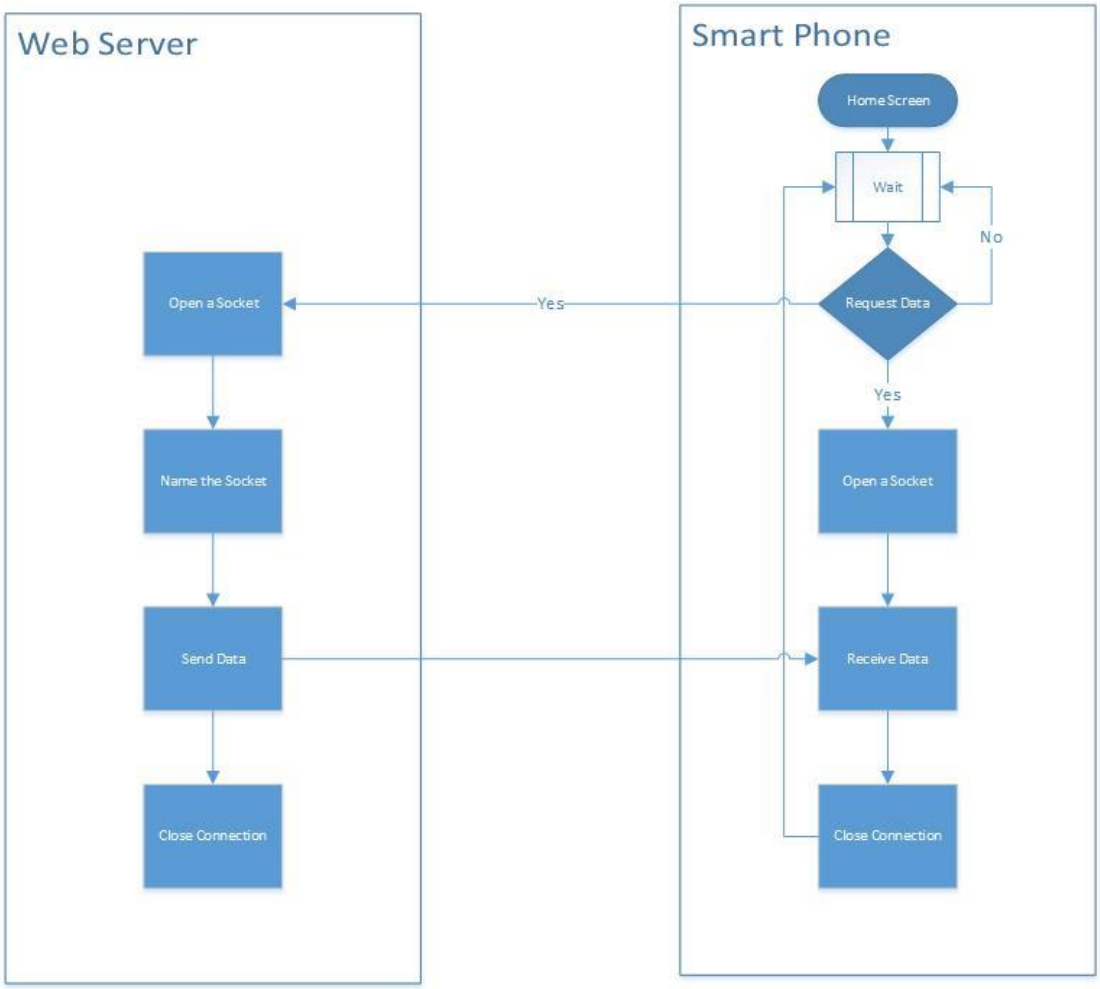


Figure 14: Networking Data Flow Diagram

Smart Power Strip App

The Smart Power Strip app's core functionality is actually extremely basic. On the receiving end of things, the app receives a string of ASCII characters and interprets what the characters mean. It splits the string into four pieces and each piece corresponds to the amount of energy consumed by a pair of outlets. The app updates energy consumption bar graphs every time it receives a new string of data. With this same data, the app also displays a pie chart that shows a distribution of how much energy some outlets are consuming compared to other ones.

The main part of the app that sends data to the Smart Power Strip is the remote control portion of the app. The remote control sends the address of an IO pin on the pi to the web server, and then the pi sets the state of the pin to either high or low. The refresh button of the app allows users to update the energy consumption data by sending a request to the web server and once the web server receives this request it sends a new string of data.

Problems Encountered

This section describes all the problems that I ran into while designing or constructing different aspects of the Smart Power Strip. I encountered problems with both the hardware and the software.

Hardware

Most of the problems in the hardware occurred from issues related to the emonTx, but were not exclusive to only the emonTx. The table below will describe all of the problems I encountered with the emonTx as well as miscellaneous other hardware problems (*Table 1*).

Table 1: Smart Power Strip Hardware Problems and Solutions

Problems	Solutions
Sending data over the emonTx's UART to the Raspberry Pi's GPIO produces garbage data even when the BAUD rates match	Instead of sending data to the GPIO of the pi, I used the USB 2.0 to TTL 6PIN Serial Converter to send data from the emonTx's UART to a USB port on the Pi. This method also requires a Python script to read data off of the USB port.
emonTx produces non-zero readings when outlets are off	Modified sketch programmed onto the emonTx so that whenever the current is less than a certain threshold, the emonTx reports it is reading zero for current.
emonTx current readings differ from actual current	Calibrated the current probes using real multi-meter then adjusted values in the sketch for the emonTx.
Powering emonTx from Raspberry Pi's 5V output results in the emonTx reporting values that were 10-100 times smaller than the true current value.	Powered the emonTx using a 9VAC, 1.11A AC-AC Power Supply instead, then the emonTx reported the correct current values. This is probably due to the 5V DC from the pi not being a sufficient reference to read the AC current of most electronics.
The 9VAC, 1.11A AC-AC Power Supply is heavy and takes up a lot of space.	Discarded old packaging and made new package for Smart Power Strip that was larger and supported more weight.
Six outlets on the Smart Power Strip but emonTx only supports up to four current probes	Connected one current sensor to two outlets and used the last current sensor to record how much current the Raspberry Pi and emonTx use.

Software

Unlike the hardware problems which were confined to primarily one piece of hardware, the software had problems all over the place. The reason for this is that the software handles many different tasks. *Table 2* describes some of the software bugs I encountered and their solutions.

Table 2: Smart Power Strip Software Problems and Solutions

Problems	Solutions
Parsing data and handling networking on the pi requires scripting. I don't know any scripting languages.	I learned Python well enough to write the code I needed.
Couldn't send multiple lines of data at once from the emonTx to the pi serially. Makes parsing data challenging.	Sent all the data necessary in a single line and ordered it so the position of the data corresponded to the position of the outlet.
Raspberry Pi sends exponentially decreasing values for energy consumed when it should be nearly constant.	Fixed energy calculation in the parseData.py script by resetting counters after 20 readings.
Requesting data from the pi on the Smart Power Strip App sometimes returned unrefreshed data.	Increased the amount of time you have to wait before requesting new data from the pi.
After a long time of inactivity on the pi, it is impossible to request data on the pi without manually refreshing the network connection on the pi.	Turned off the low power state on the Wi-Fi adapter so it is ready to send or receive data at any time.
The scripts on the pi that handle the data parsing and networking need to be run manually using a keyboard or SSH.	Made the scripts run on the boot up of the pi and made them run in the background using crontab
Every time the pi reconnects to a network it gets assigned a new IP	Made the router/hotspot assign a static IP to the pi
Charts used on Smart Power Strip App don't scale correctly.	Emailed author of chart library then downloaded new release of library.

Testing

In order to make the debugging process of the Smart Power Strip project very simple, all the individual hardware components were tested individually before interfacing them with another piece of hardware.

Raspberry Pi GPIO

Before actually connecting the Raspberry Pi to the relay, I used a test rig that had some resistors and LEDs connected to the pi's GPIO. This way I could test if the wiring pi library was functioning as intended. Using a simple script on the pi, I was able to toggle the LEDs on my test rig. Below is a schematic of what the test rig looked like (*Figure 15).*

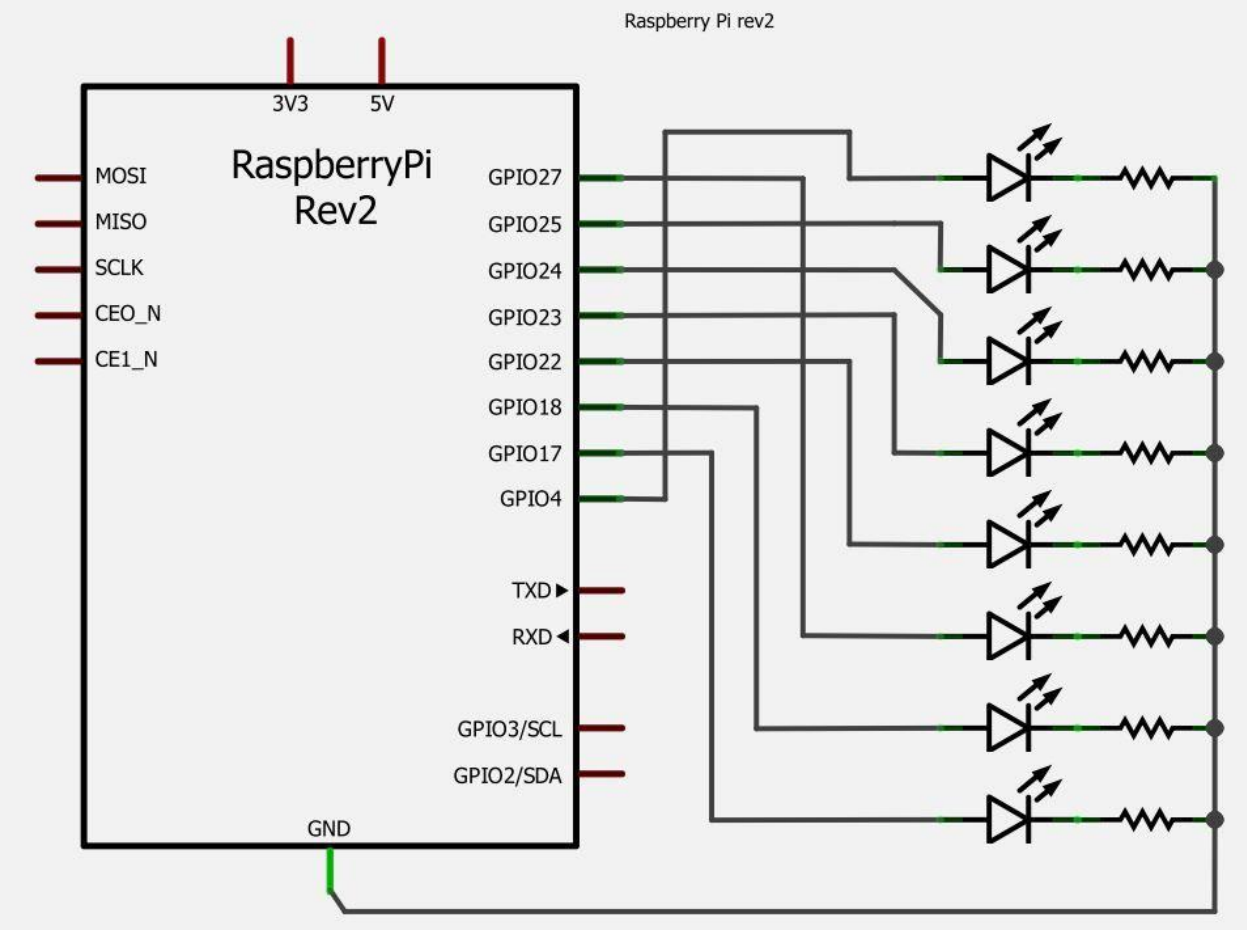


Figure 15: Raspberry Pi GPIO Test Rig

Raspberry Pi GPIO+ Relay

Once the GPIO on the pi was working as intended, I proceeded to check whether the relay would work while it was connected between the pi's GPIO and the old test rig with the LEDs. During this process I found out that the relay is active when the signal it receives is low. Therefore whenever I wanted to turn on an LED, I actually had to make sure that the signal sent out on the pi's GPIO was pulled all the way down to zero. It was very important that I figured this out early on because this affects the way that the outlets are toggled from the Smart Power Strip App.

Raspberry Pi GPIO + emonTx + Current Probes

This stage of hardware testing was the trickiest and also took the longest to integrate. The first step I took was to use the sketch that came loaded by default onto the emonTx and connect the UART of the emonTx to the UART of the Raspberry Pi. Unfortunately, no matter what I tried the communication between the two UARTs never worked. Most of the time the Raspberry Pi received no data but sometimes it would receive a line of random characters that was also wrong.

Since sending data over the UART did not work, I proceeded to try the USB 2.0 to TTL 6PIN Serial Converter. This allowed for seamless data transfer from the emonTx to the Raspberry Pi. I was able to use the same code that I had written for the UART of the pi with the converter. The only thing that had to be changed was the address of where I was expecting to get data from and in this case it was the address of the USB port on the pi. This code is featured at the end of this document in the *Appendix*.

Once I knew I was successfully transferring data between the emonTx and the Raspberry Pi, I attempted to integrate the current probes. The emonTx has libraries and physical ports that allowed me to easily integrate the probes but they still need to be used in a particular way. Below is a table that shows the different loads and load configurations that were used to test the current probes (*Table 3).*

Table 3: Current Probe Tests and Results

Device	Configuration	Readings	Results
Lamp 20W, 1/6A	- Current carrying wire opposing direction of arrow on probe - Single winding -3.3V DC Supply	-Negative Power -Small Current <1uA -Small Voltage ~0V	Incorrect
Lamp 20W, 1/6A	- Current carrying wire same direction of arrow on probe - Single winding -3.3V DC Supply	-Positive Power -Small Current <1uA -Small Voltage ~0V	Incorrect
Lamp 20W, 1/6A	- Current carrying wire same direction of arrow on probe - 7x single winding -3.3V DC Supply	-Positive Power - ~1A current -Small Voltage ~0V	Incorrect
Microwave 1.2kW, 10A	- Current carrying wire same direction of arrow on probe - 7x single winding -3.3V DC Supply	-Positive Power - ~70A current --Small Voltage ~0V	Incorrect
Microwave 1.2kW, 10A	- Current carrying wire same direction of arrow on probe - Single winding -3.3V DC Supply	-Positive Power - ~10A current - Small Voltage ~0V	Incorrect
Microwave 1.2kW, 10A	- Current carrying wire same direction of arrow on probe - Single winding -9VAC Power Supply	- ~1200 W power - ~10A current - ~120Vrms	Correct
Lamp 20W, 1/6A	- Current carrying wire same direction of arrow on probe - Single winding -9VAC Power Supply	- ~20 W power - ~1/6A current - ~120Vrms	Correct

Once I determined the current probes were configured correctly, I wanted to calibrate the probes so that the readings they took for current were closer to the actual currents being carried through the wires. The results of the calibration can be seen below in *Table 4*.

Table 4: Current Probe Calibration Results

Probe	Before	Calibration Factor	After	Actual
#1	~17 W	90.9	20 W	19.8 W
#2	~17 W	90.9	20 W	19.8 W
#3	~17 W	90.9	20 W	19.8 W
#4	~19 W	16.6	20 W	19.8 W

After the current probes were calibrated, there was a small problem where the current probes would report non-zero readings when electronics were turned off. This was resolved in software by setting a very low threshold for the current so the emonTx would report that the current is zero even if the current read on the probes was very small.

Raspberry Pi + emonTx UART

The emonTx is designed to send data wirelessly over Wi-Fi, but since the emonTx and the Raspberry Pi are a matter of inches apart in the Smart Power Strip I wanted to try sending the over the emonTx's UART.

In order to program the emonTx, a TTL to USB converter is required and conveniently this is the same method used for sending data over the emonTx's UART. To begin testing the UART, I sent data from the emonTx to a computer running Putty. The data came through just as I expected it to so I proceeded to try to send data from the UART of the emonTx to the UART of the Raspberry Pi.

I immediately ran into a problem because the UART of the emonTx has 5 pins and the UART of the pi only uses 2 pins, Tx and Rx. When I sent data over the UART to the pi using wires, the pi printed out blank lines or random ASCII characters when I ran the parseData.py script. I tried to fix this problem in software many different ways, but I never determined the cause of not being able to correctly receive the data on the Rx pin of the Raspberry Pi.

Since I already had the TTL to USB converter and I had used it before to transfer data over the UART of the emonTx to my computer, I decided to plug in the USB side of the converter to the Raspberry Pi to try to receive data. Using the same parseData.py script with a new port to read data from, the pi received the correct information from the emonTx's UART.

Below is *Table 5* showing the results of failed and successful attempts at sending data over UART.

Table 5: UART Communication Results

Configuration	Sent Data	Received Data
emonTx UART → TTL to USB converter → Computer running Putty	'0.00 0.00 0.00 0.00' 'data!'	'0.00 0.00 0.00 0.00' 'data!'
emonTx UART → 2 Wires → Tx and Rx pins on Raspberry Pi	'0.00 0.00 0.00 0.00'	
emonTx UART → TTL to USB converter → Tx and Rx pins on Raspberry Pi	'0.00 0.00 0.00 0.00' 'data!'	'0.00 0.00 0.00 0.00' 'data!'

Results and Conclusion

Each of the hardware components chosen for the Smart Power Strip worked as intended, minus the UART on the emonTx. This problem was fixed by transferring data over a serial connection using a TTL to USB converter. The software needed for the pi and emonTx work as intended, but the interface of the Smart Power Strip App needs to be modified to allow users to set timers. As of now, the Smart Power Strip:

- Houses 6 outlets
- Communicates with the Smart Power Strip App over a Wi-Fi connection
- Consumes less than ~15kWh per year
- Works without any setup
- Sets timers on the outlets upon request

The Smart Power Strip App currently doesn't have an interface for setting timers for the outlets, but it has most of the necessary functionality for communicating with the Smart Power Strip. Currently, the Smart Power Strip App:

- Toggles individual outlets
- Receives data from the Smart Power Strip
- Reports the amount of Energy consumed by the outlets
- Shows a distribution of total energy used

The Smart Power Strip meets almost all the requirements discussed in the *Requirements* section. The one primary requirement it doesn't meet is the price requirement. Unfortunately the emonTx approximately doubles the overall cost of the Smart Power Strip. The secondary objective to implement timers was achieved. Unfortunately, the Smart Power Strip and Smart Power Strip app are currently required to be on the same network to communicate with each other.

Future Work

To decrease the overall cost of parts for the Smart Power Strip, I would like to create a new version of the Smart Power Strip that does not use the emonTx. This would reduce the cost of parts by about 100 USD. I would also like to build custom current probes and circuitry to interface with the Raspberry Pi to reduce the amount of space required to house all the hardware required by the Smart Power Strip.

The main feature that I would like to add is an interface for the Smart Power Strip App to be able to set timers for the Smart Power Strip. The Raspberry Pi is already running a script that looks at the data the Smart Power Strip app sends and sets timers accordingly. The only problem is there is currently no way of modifying the data the Smart Power Strip app sends without reprogramming the app onto a smart device. Lastly I would like to make communications between the Smart Power Strip and app more secure so that I could potentially one day allow for communications over different networks.

References

[1] O. Ozel, “Transmission with Energy Harvesting Nodes in Fading Wireless Channels: Optimal Policies”, *IEEE J. Sel. Areas Commun.*, vol. 29, no. 8, pp. 1732-1743, Sep. 2011.

[2] Diffen.com, 'TCP vs UDP - Difference and Comparison | Diffen', 2015. [Online]. Available: http://www.diffen.com/difference/TCP_vs_UDP. [Accessed: 02- May- 2015].

[3] Wiki.python.org, 'UdpCommunication - Python Wiki', 2015. [Online]. Available: https://wiki.python.org/moin/UdpCommunication. [Accessed: 02- May- 2015].

[4] Cases for your Raspberry Pi, 'Tutorial - How to give your Raspberry Pi a Static IP Address', 2015. [Online]. Available: http://www.modmypi.com/blog/tutorial-how-to-give-your-raspberry-pi-a-static-ip-address. [Accessed: 02- May- 2015].

[5] Pyserial.sourceforge.net, 'Short introduction — pySerial 2.7 documentation', 2015. [Online]. Available: http://pyserial.sourceforge.net/shortintro.html. [Accessed: 02- May- 2015].

[6] Wiki.openenergymonitor.org, 'EmonTx V3 - OpenEnergyMonitor Wiki', 2015. [Online]. Available: http://wiki.openenergymonitor.org/index.php?title=EmonTx_V3. [Accessed: 02- May- 2015].

[7] Pi4j.com, 'The Pi4J Project - Pin Numbering - Raspberry Pi Model B+', 2015. [Online]. Available: http://pi4j.com/pins/model-b-plus.html. [Accessed: 02- May- 2015].

[8] Docs.python.org, '15.3. time — Time access and conversions — Python 2.7.10 documentation', 2015. [Online]. Available: https://docs.python.org/2/library/time.html. [Accessed: 08- Jun- 2015].

[9] Docs.python.org, '8.8. sched — Event scheduler — Python 2.7.10 documentation', 2015. [Online]. Available: https://docs.python.org/2/library/sched.html. [Accessed: 08- Jun- 2015].

[10] G. Henderson, 'WiringPi/WiringPi2-Python', *GitHub*, 2015. [Online]. Available: https://github.com/WiringPi/WiringPi2-Python. [Accessed: 08- Jun- 2015].

[11] S. Kildall, 'Raspberry Pi: Launch Python script on startup', *Instructables.com*, 2015. [Online]. Available: http://www.instructables.com/id/Raspberry-Pi-Launch-Python-script-on-startup/. [Accessed: 08- Jun- 2015].

[12] Openenergymonitor.org, 'Calibration | OpenEnergyMonitor', 2015. [Online]. Available: http://openenergymonitor.org/emon/buildingblocks/calibration. [Accessed: 08- Jun- 2015].

[13] Openenergymonitor.org, 'Report: Yhdc SCT-013-000 Current Transformer | OpenEnergyMonitor', 2015. [Online]. Available: http://openenergymonitor.org/emon/buildingblocks/report-yhdc-sct-013-000-current-transformer. [Accessed: 08- Jun- 2015].

[14] Wiringpi.com, 'WiringPi', 2015. [Online]. Available: http://wiringpi.com/. [Accessed: 08- Jun- 2015].

[15] Raspberrypi.org, 'Raspberry Pi • View topic - Edimax Wifi adapter seems to go to sleep if not used (solved', 2015. [Online]. Available: https://www.raspberrypi.org/forums/viewtopic.php?t=61665. [Accessed: 08- Jun- 2015].

[16] Raspberrypi.org, 'Raspberry Pi Documentation', 2015. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/models/README.md#modelbplus. [Accessed: 09- Jun- 2015].

Appendix

This appendix covers in depth hardware circuit diagrams. It also covers the calculations done in software and various types of code that runs on the Raspberry Pi, the emonTx, and the app.

Circuit Diagram

The circuit diagram in *Figure 16*, shows the interconnections of the Raspberry Pi, emonTx, 8-channel relay, outlets, 6 pin TTL to USB converter, and current probes. A high level block diagram can be seen earlier in this document at the end of the *Hardware Considerations* section (*Figure 9*).

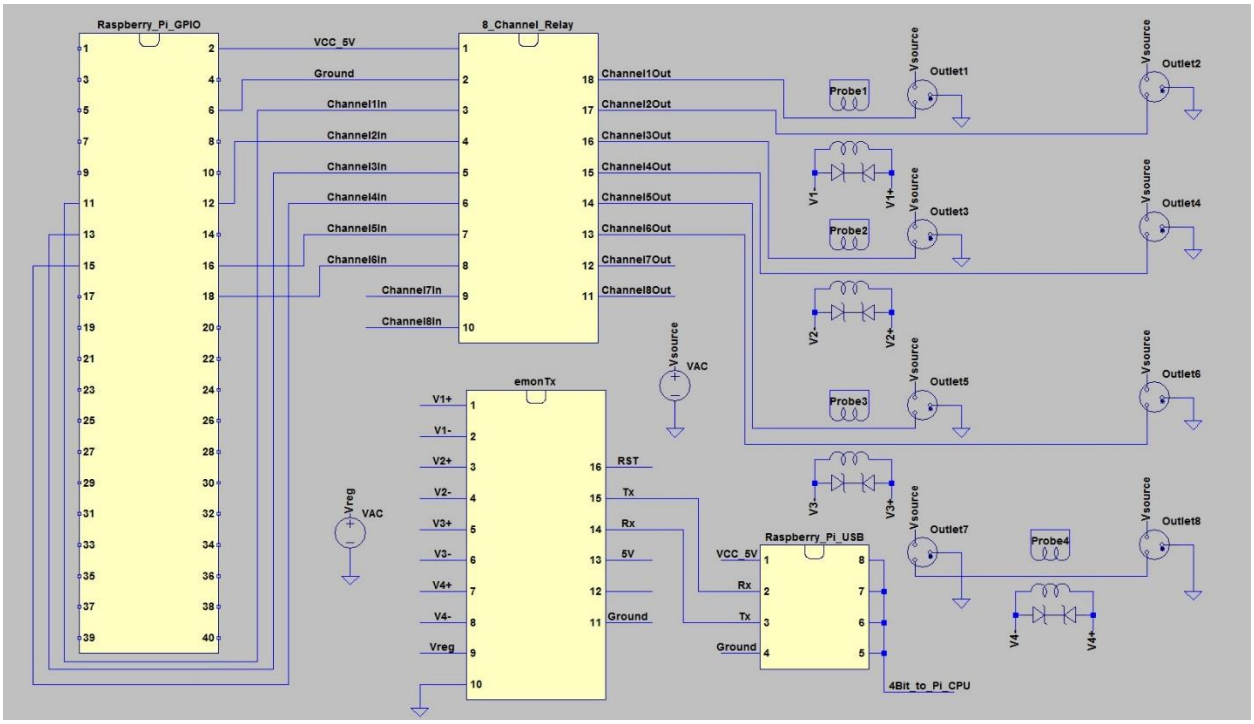


Figure 16: Overall System Connections

Calculations

This section covers the calculations done in software for calculating power consumption, calculating energy consumption, and setting timers.

Power Calculation

A power calculation is necessary for obtaining a value for the amount of energy consumed. It is also the easiest calculation to do since the current probes only report current. The calculation for power was simply the current recorded by the current probe multiplied by the reference voltage of 120V.

$$Power = Voltage_{120V} x\ Current\ Sensor\ Reading$$

Energy Calculation

The energy calculation is much more complex than the power calculation. The reason for this is that the energy calculation is time dependent. The Raspberry Pi gets 20 power calculations every 10 seconds, so every energy calculation is based off of a period of 10 seconds. The amount of non-zero power calculations are averaged and then divided by the number of seconds in an hour to get the amount of energy consumed in a 10 second interval.

$$Energy_{10sec} = Power_{Avg/10sec}\ x\ \frac{Period_{10sec}}{Seconds/Hour_{3600sec}}$$

Timer Calculation

This timer calculation calculates how long to sleep each thread for an outlet before toggling the state of that outlet. All the times on the Raspberry Pi lie between 0:00 to 23:59. If a timer is set for a time earlier than the current time, then 24 hours in seconds are added onto the timer and the difference between that and the current time equal the sleep time. If the timer is set for a time later than the current time, then the sleep time is equal to the timer minus the current time.

$$Sleep\ Time = Timer + 24\ Hours_{86400sec} - Current\ Time$$
OR
$$Sleep\ Time = Timer - Current\ Time$$

emonTx Code

dataAcquisition.cpp

```
// ------------------------------------------------------------------------//
// Name:    Smart Power Strip Data Acquisition
// By:      Dannie Alfaro
//          dataAcquistion.cpp
// ------------------------------------------------------------------------//
//
// Project Name:    Smart Power Strip
// Date Modified:   4/07/15
// Software:        Arduino IDE
// Hardware:        emonTx v.3 (ATmega328)
// Files Needed:    EmonLib.h, dataAcquistion.cpp
// Connections:     6-pin TTL to USB Converter
// Purpose:         Senior Project
// ------------------------------------------------------------------------//

// Tell emonLib this is the emonTx V3
#define emonTxV3

// Include Emon Library
#include "EmonLib.h"

// Create four instances
EnergyMonitor ct1, ct2, ct3, ct4;

unsigned long lastpost = 0;
const byte LEDpin=       6;                                  // emonTx V3 LED

void setup()
{
  pinMode(LEDpin, OUTPUT);
  digitalWrite(LEDpin,HIGH);

  Serial.begin(9600);

  // Calibration, phase_shift
  ct1.voltage(0, 130, 1.7);
  ct2.voltage(0, 130, 1.7);
  ct3.voltage(0, 130, 1.7);
  ct4.voltage(0, 130, 1.7);

  // CT Current calibration
  ct1.current(1, 90.9);
  ct2.current(2, 90.9);
  ct3.current(3, 90.9);

  // CT 4 is high accuracy @ low power -  4.5kW Max
  ct4.current(4, 16.6);

  lastpost = 0;

  delay(2000);
  digitalWrite(LEDpin,LOW);
```

```
}

void loop()
{
  if ((millis()-lastpost)>=500)
  {
    lastpost = millis();

    // .calcVI: Calculate all. No.of half wavelengths (crossings), time-out
    ct1.calcVI(20,2000);
    ct2.calcVI(20,2000);
    ct3.calcVI(20,2000);
    ct4.calcVI(20,2000);

    // Print to serial

    if(ct1.powerFactor < .2 && ct1.powerFactor > -.2)
    {
      Serial.print("0.00");
      Serial.print(' ');
    }
    else
    {
      //ct1.serialprint();
      Serial.print(ct1.apparentPower);
      Serial.print(' ');
    }

    ct2.Irms=ct2.Irms-.15;
    if((ct2.powerFactor < .1 && ct2.powerFactor > -.1) || ct2.Irms < .01)
    {
      Serial.print("0.00");
      Serial.print(' ');
    }
    else
    {
      ct2.Irms=ct2.Irms+.15;
      //ct2.serialprint();
      Serial.print(ct2.apparentPower);
      Serial.print(' ');
    }

    ct3.Irms=ct3.Irms-.15;
    if((ct3.powerFactor < .2 && ct3.powerFactor > -.2) || ct3.Irms < .01)
    {
      Serial.print("0.00");
      Serial.print(' ');
    }
    else
    {
      ct3.Irms=ct3.Irms+.15;
      //ct3.serialprint();
      Serial.print(ct3.apparentPower);
      Serial.print(' ');
    }

    if(ct4.powerFactor < .2 && ct4.powerFactor > -.2)
```

```
    {
      Serial.print("0.00");
      Serial.print(' ');
    }
    else
    {
      //ct4.serialprint();
      Serial.print(ct4.apparentPower);
      Serial.print(' ');
    }

    Serial.println();

    // Note: the following measurements are also available:
    // - ct1.apparentPower
    // - ct1.Vrms
    // - ct1.Irms
  }
}
```

Raspbery Pi Code

parseData.py

```
# ------------------------------------------------------------------------
# Name:      Smart Power Strip Data Parsing
# By:        Dannie Alfaro
#            parseData.py
# ------------------------------------------------------------------------
#
# Project Name:      Smart Power Strip
# Date Modified:     4/07/15
# Software:          Python 2.7
# Hardware:          Raspberry Pi B+
# Files Needed:      serial.py, parseData.py, templog.txt, testlog.txt
# Connections:       6-pin TTL to USB Converter
# Purpose:           Senior Project
# ------------------------------------------------------------------------

import serial

port = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=.5)

#Arrays Hold Temporary Data
outlet1_data = [0]*(20) #outlet1_data=[0,...,0]
outlet2_data = [0]*(20)
outlet3_data = [0]*(20)
outlet4_data = [0]*(20)

#Counts Hold Number of Non-Zero Data Entries
count1 = 0
count2 = 0
count3 = 0
count4 = 0
i = 0

#Function Definitions
def filewrite(rcv):
    logfile = open("templog.txt", "w")
    logfile.write(rcv)
    logfile.close
def testwrite(rcv):
    testfile = open("testlog.txt", "a")
    testfile.write(rcv)
    testfile.close

while True:
    rcv = port.readline()
    if not rcv:
        bool = 1
    else:
        outlet1_data[i], outlet2_data[i], outlet3_data[i], outlet4_data[i] =
            rcv.split(" ", 3)

    print(repr(rcv))
    testwrite(rcv)
```

```
outlet1_data[i] = float(outlet1_data[i])
outlet2_data[i] = float(outlet2_data[i])
outlet3_data[i] = float(outlet3_data[i])
outlet4_data[i] = float(outlet4_data[i])

if outlet1_data[i] != 0.0:
    count1 += 1
if outlet2_data[i] != 0.0:
    count2 += 1
if outlet3_data[i] != 0.0:
    count3 += 1
if outlet4_data[i] != 0.0:
    count4 += 1

i += 1

#Do averages here and write to file
if i == 19:

    if count1 == 0:
        count1 = 1
    if count2 == 0:
        count2 = 1
    if count3 == 0:
        count3 = 1
    if count4 == 0:
        count4 = 1

    sum1 = (1.0/360.0) * sum(outlet1_data) / count1
    sum2 = (1.0/360.0) * sum(outlet2_data) / count2
    sum3 = (1.0/360.0) * sum(outlet3_data) / count3
    sum4 = (1.0/360.0) * sum(outlet4_data) / count4

    logfile = open("templog.txt", "r")
    temp = logfile.readline()
    logfile.close()

    outlet1Temp, outlet2Temp, outlet3Temp, outlet4Temp = temp.split("
        ",3)
    outlet1Temp = float(outlet1Temp)
    outlet2Temp = float(outlet2Temp)
    outlet3Temp = float(outlet3Temp)
    outlet4Temp = float(outlet4Temp)

    total1 = sum1 + outlet1Temp
    total2 = sum2 + outlet2Temp
    total3 = sum3 + outlet3Temp
    total4 = sum4 + outlet4Temp

    print(str(total1) + " " + str(total2) + " " + str(total3) + " " +
        str(total4))
    writeLine = str(total1) + " " + str(total2) + " " + str(total3) + " "
        + str(total4)

    filewrite(writeLine)
```

```
#reinitialize all variables
outlet1_data = [0]*(20) #outlet1_data=[0,...,0]
outlet2_data = [0]*(20)
outlet3_data = [0]*(20)
outlet4_data = [0]*(20)
i = 0
count1 = 0
count2 = 0
count3 = 0
count4 = 0

print("Complete")
```

txrx_data.py

```
# ------------------------------------------------------------------------
# Name:      Smart Power Strip Send and Receive Data
# By:        Dannie Alfaro
#            txrx_data.py
# ------------------------------------------------------------------------
#
# Project Name:      Smart Power Strip
# Date Modified:     5/29/15
# Software:          Python 2.7
# Hardware:          Raspberry Pi B+
# Files Needed:      socket.py, time.py, wiringpi.py, txrx_data.py
#                    templog.txt, timefile.txt
# Connections:       Wi-Fi
# Purpose:           Senior Project
# ------------------------------------------------------------------------

import socket
import time
import wiringpi2 as wiringpi

UDP_PORT_SEND = 5005
UDP_PORT_RECV = 5006

#Get IP
MY_IP = socket.gethostbyname(socket.getfqdn())

print(MY_IP)

recv_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
recv_socket.bind((MY_IP, UDP_PORT_RECV))

last_time = 0

#Function Definitions
def readData():
    file = open("templog.txt", "r")
    line = file.readline()
    file.close()
    return str(line)

def writeData(times):
    timefile = open("timefile.txt", "w")
    timefile.write(times)
    timefile.close

#Main
while True:
    print("b/c kyle said so")
    data, addr = recv_socket.recvfrom(1024)
    temp = str(addr)
    data = str(data)
    print(temp)
    print(data)
    writeData(data)
```

```
    print("Time Saved")
    if data:
        data = "nope!"
        a, UDP_IP, b = temp.split("'", 2)
        send_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        try:
            send_socket.connect((MY_IP, 80))
            print(readData())
            send_socket.sendto(readData(), (UDP_IP, UDP_PORT_SEND))
            print("sent")
            send_socket.close()

        except Exception:
            running = False
            recv_socket.close()
            print("we goofed")
            raise
recv_socket.close()
```

timers.py

```
# ------------------------------------------------------------------------
# Name:      Smart Power Strip Timers
# By:        Dannie Alfaro
#            timers.py
# ------------------------------------------------------------------------
#
# Project Name:      Smart Power Strip
# Date Modified:     6/1/15
# Software:          Python 2.7
# Hardware:          Raspberry Pi B+
# Files Needed:      Timer.py, time.py, wiringpi.py, timers.py
#                    timefile.txt
#                    templog.txt, timefile.txt
# Connections:       N/A
# Purpose:           Senior Project
# ------------------------------------------------------------------------

import wiringpi2 as wiringpi
from time import sleep
from threading import Timer
import time

#Function Definitions#
#Call initializeGPIO followed by readData
def initializeGpio():
    wiringpi.wiringPiSetup()
    wiringpi.pinMode(0,1)
    wiringpi.pinMode(1,1)
    wiringpi.pinMode(2,1)
    wiringpi.pinMode(3,1)
    wiringpi.pinMode(4,1)
    wiringpi.pinMode(5,1)

def readData():
    file = open("timefile.txt", "r")
    line = file.readline()
    file.close()
    return str(line)

def tOn1(): wiringpi.digitalWrite(0,0)
def tOn2(): wiringpi.digitalWrite(1,0)
def tOn3(): wiringpi.digitalWrite(2,0)
def tOn4(): wiringpi.digitalWrite(3,0)
def tOn5(): wiringpi.digitalWrite(4,0)
def tOn6(): wiringpi.digitalWrite(5,0)
def tOff1():wiringpi.digitalWrite(0,1)
def tOff2():wiringpi.digitalWrite(1,1)
def tOff3():wiringpi.digitalWrite(2,1)
def tOff4():wiringpi.digitalWrite(3,1)
def tOff5():wiringpi.digitalWrite(4,1)
def tOff6():wiringpi.digitalWrite(5,1)

def getTime():
    return time.asctime(time.localtime(time.time()))
```

```
def sleepTime(trigger_time):
    hour, min, sec = getTime()[11:19].split(":", 2)
    temp = (int(hour) * 3600 + int(min) * 60 + int(sec))
    timer = 3600 * trigger_time
    if timer >= temp:
        sleep_time = timer - temp
    else:
        timer = timer + 86400
        sleep_time = timer - temp
    #print sleep_time
    return sleep_time

def printAlarm(outlet, state, sleep_time):
    if state == 1:
        print "Outlet", outlet, "OFF at",
time.asctime(time.localtime(time.time()+sleep_time))
    else:
        print "Outlet", outlet, "ON at",
time.asctime(time.localtime(time.time()+sleep_time))

#MAIN
#Initialization
initializeGpio()
out1temp = -1
out2temp = -1
out3temp = -1
out4temp = -1
out5temp = -1
out6temp = -1
flag1 = 0
flag2 = 0
flag3 = 0
flag4 = 0
flag5 = 0
flag6 = 0

#Need a Loop here
while True:

    data = readData()

    out1,out2,out3,out4,out5,out6 = data.split(" ", 5)

    if out1 != out1temp:
        out1temp = out1

        if flag1 == 1:
            timer1.cancel()
            print "Timer 1 Cancelled"

        if int(out1[0]) == 2:
            print "No Timer Set for Outlet 1"
        else:
            if(int(out1[0]) == 0):
                timer1 = Timer(sleepTime(int(out1[1:3])), tOn1, ())
                #timer1 = Timer(10, tOn1, ())
```

```
            timer1.start()
        else:
            timer1 = Timer(sleepTime(int(out1[1:3])), tOff1, ())
            #timer1 = Timer(10, tOff1, ())
            timer1.start()

        flag1 = 1
        printAlarm(1, int(out1[0]), sleepTime(int(out1[1:3])))

if out2 != out2temp:
    out2temp = out2

    if flag2 == 1:
        timer2.cancel()
        print "Timer 2 Cancelled"

    if int(out2[0]) == 2:
        print "No Timer Set for Outlet 2"
    else:
        if(int(out2[0]) == 0):
            timer2 = Timer(sleepTime(int(out2[1:3])), tOn2, ())
            timer2.start()
        else:
            timer2 = Timer(sleepTime(int(out2[1:3])), tOff2, ())
            timer2.start()

        flag2 = 1
        printAlarm(2, int(out2[0]), sleepTime(int(out2[1:3])))

if out3 != out3temp:
    out3temp = out3

    if flag3 == 1:
        timer3.cancel()
        print "Timer 3 Cancelled"

    if int(out3[0]) == 2:
        print "No Timer Set for Outlet 3"
    else:
        if(int(out3[0]) == 0):
            timer3 = Timer(sleepTime(int(out3[1:3])), tOn3, ())
            timer3.start()
        else:
            timer3 = Timer(sleepTime(int(out3[1:3])), tOff3, ())
            timer3.start()

        flag3 = 1
        printAlarm(3, int(out3[0]), sleepTime(int(out3[1:3])))

if out4 != out4temp:
    out4temp = out4

    if flag4 == 1:
        timer4.cancel()
        print "Timer 4 Cancelled"

    if int(out4[0]) == 2:
```

```
            print "No Timer Set for Outlet 4"
        else:
            if(int(out4[0]) == 0):
                timer4 = Timer(sleepTime(int(out4[1:3])), tOn4, ())
                timer4.start()
            else:
                timer4 = Timer(sleepTime(int(out4[1:3])), tOff4, ())
                timer4.start()

            flag4 = 1
            printAlarm(4, int(out4[0]), sleepTime(int(out4[1:3])))

    if out5 != out5temp:
        out5temp = out5

        if flag5 == 1:
            timer5.cancel()
            print "Timer 5 Cancelled"

        if int(out5[0]) == 2:
            print "No Timer Set for Outlet 5"
        else:
            if(int(out5[0]) == 0):
                timer5 = Timer(sleepTime(int(out5[1:3])), tOn5, ())
                timer5.start()
            else:
                timer5 = Timer(sleepTime(int(out5[1:3])), tOff5, ())
                timer5.start()

            flag5 = 1
            printAlarm(5, int(out5[0]), sleepTime(int(out5[1:3])))

    if out6 != out6temp:
        out6temp = out6

        if flag6 == 1:
            timer6.cancel()
            print "Timer 6 Cancelled"

        if int(out6[0]) == 2:
            print "No Timer Set for Outlet 6"
        else:
            if(int(out6[0]) == 0):
                timer6 = Timer(sleepTime(int(out6[1:3])), tOn6, ())
                timer6.start()
            else:
                timer6 = Timer(sleepTime(int(out6[1:3])), tOff6, ())
                timer6.start()

            flag6 = 1
            printAlarm(6, int(out6[0]), sleepTime(int(out6[1:3])))

    print(out1, out2, out3, out4, out5, out6)

    print(getTime())

    time.sleep(5)
```

gpio.php

```
// ------------------------------------------------------------------------
// Name:     Smart Power Strip Input/Output
// By:       Dannie Alfaro
//           gpio.php
// ------------------------------------------------------------------------
//
// Project Name:     Smart Power Strip
// Date Modified:    2/15/15
// Software:
// Hardware:         Raspberry Pi B+
// Files Needed:     gpio.php
// Connections:      N/A
// Purpose:          Senior Project
// ------------------------------------------------------------------------

<!-- This page is requested by the JavaScript, it updates the pin's status
and then print it -->
<?php
//Getting and using values
if (isset ($_GET["pin"]) && isset($_GET["status"]) ) {
    $pin = strip_tags($_GET["pin"]);
    $status = strip_tags($_GET["status"]);
    //Testing if values are numbers
    if ( (is_numeric($pin)) && (is_numeric($status)) && ($pin <= 7) && ($pin
>= 0) && ($status == "0") || ($status == "1") ) {
        //set the gpio's mode to output
        system("gpio mode ".$pin." out");
        //set the gpio to high/low
        if ($status == "0" ) { $status = "1"; }
        else if ($status == "1" ) { $status = "0"; }
        system("gpio write ".$pin." ".$status );
        //reading pin's status
        exec ("gpio read ".$pin, $status, $return );
        //printing it
        echo ( $status[0] );
    }
    else { echo ("fail"); }
} //print fail if cannot use values
else { echo ("fail"); }
?>
```

Smart Power Strip App Code

MainActivity.java

```
// ---------------------------------------------------------------------------
// Name:    Smart Power Strip Input/Output
// By:      Josh Logier
//          MainActivity.java
// ---------------------------------------------------------------------------
//
// Project Name:    Smart Power Strip App
// Date Modified:   6/8/15
// Software:        Android Studio
// Hardware:        Android Smart Device
// Files Needed:    CustomListAdapter.java, MyMarkerView.java,
//                  OnStartFragment.java, MainActivity.java,
//                  PieChartFragment.java, SimpleFragment.java,
//                  RemoteControlFragment.java, BarChartFragment.java
// Connections:     Wi-Fi
// Purpose:         Senior Project
// ---------------------------------------------------------------------------

package com.example.smartpower.smartpowerstripapp;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.res.Configuration;
import android.os.Bundle;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.app.ActionBarDrawerToggle;
import android.util.Log;
import android.util.TypedValue;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
```

```
import java.net.InetSocketAddress;
import java.net.SocketException;

public class MainActivity extends ActionBarActivity {

    protected SharedPreferences prefs;

    private ListView mDrawerList;
    private DrawerLayout mDrawerLayout;
    private ActionBarDrawerToggle mDrawerToggle;
    private String mActivityTitle;
    private boolean toggle = false;

    private Integer[] iconArray = { R.drawable.ic_home, R.drawable.ic_remote,
R.drawable.ic_log, R.drawable.ic_breakdown, R.drawable.ic_settings, null,
null};
    private String[] optionsArray = { "Home", "Remote Control", "Power
Consumption", "Distribution", "About" };
    private String[] mDrawerTitles = {"Smart Power Strip", "Remote Control",
"Power Consumption", "Distribution"};
    private int selectedPosition = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        prefs =
getSharedPreferences("com.example.smartpower.smartpowerstripapp",
Context.MODE_PRIVATE);

        //Initializes networking so that we can get new data from the
Raspberry Pi
        dataReceiver();
        requestData();

        LayoutInflater inflater = this.getLayoutInflater();
        View rowView = inflater.inflate(R.layout.mylist, null, true);
        TextView text = (TextView) rowView.findViewById(R.id.text1);

        mDrawerList = (ListView) findViewById(R.id.navList);
        //scales the navigation drawer differently for a tablet and phone
similar to Google's apps
        if (isTablet(this)) {
            mDrawerList.getLayoutParams().width = (int)
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, 320,
getResources().getDisplayMetrics());
        } else {
            mDrawerList.getLayoutParams().width = (int)
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, 300,
getResources().getDisplayMetrics());
        }

        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
        mActivityTitle = getTitle().toString();

        //Launches proper fragment when navigation drawer option is selected
```

```
        addDrawerItems();
        setupDrawer();

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setHomeButtonEnabled(true);

        //Initial screen when app loads
        onStartFragment();
    }

    @Override
    protected void onPostCreate(Bundle savedInstanceState) {
        super.onPostCreate(savedInstanceState);
        // Sync the toggle state after onRestoreInstanceState has occurred.
        mDrawerToggle.syncState();
    }

    private void addDrawerItems() {
        CustomListAdapter adapter = new CustomListAdapter(this, optionsArray,
iconArray);

        mDrawerList.setAdapter(adapter);

        mDrawerList.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                selectedPosition = position;
                updateFragment();
                mDrawerLayout.closeDrawer(mDrawerList);
            }
        });

        selectedPosition = 0;
        updateFragment();
    }

    private void onStartFragment() {
        FragmentManager fm = getFragmentManager();
        OnStartFragment frag = new OnStartFragment();

        FragmentTransaction ft = fm.beginTransaction();
        ft.replace(R.id.content_frame, frag);
        ft.addToBackStack(null);
        ft.commit();

        mActivityTitle = mDrawerTitles[0];
        invalidateOptionsMenu();
    }

    private void updateFragment() {
        /* Getting reference to the FragmentManager */
        FragmentManager fm = getFragmentManager();
        Toast toast = Toast.makeText(this, "", Toast.LENGTH_SHORT);

        if (selectedPosition == 0) {
```

```
            onStartFragment();
            //sets the action bar title back to the app name when on "home
screen"
            mActivityTitle = mDrawerTitles[0];
        }
        else if (selectedPosition == 1) {
            RemoteControlFragment rFragment = new RemoteControlFragment();

            // Replace fragment
            FragmentTransaction ft = fm.beginTransaction();
            ft.replace(R.id.content_frame, rFragment);
            ft.addToBackStack(null);
            ft.commit();

            mActivityTitle = mDrawerTitles[1];
            toast.setText("Remote control selected");
            toast.show();
        }
        else if (selectedPosition == 2) {
            BarChartFragment chartFrag = new BarChartFragment();

            FragmentTransaction ft = fm.beginTransaction();
            ft.replace(R.id.content_frame, chartFrag);
            ft.addToBackStack(null);
            ft.commit();

            mActivityTitle = mDrawerTitles[2];
            toast.setText("Power Usage");
            toast.show();
        }
        else if (selectedPosition == 3) {
            PieChartFragment chartFrag = new PieChartFragment();

            FragmentTransaction ft = fm.beginTransaction();
            ft.replace(R.id.content_frame, chartFrag);
            ft.addToBackStack(null);
            ft.commit();

            mActivityTitle = mDrawerTitles[3];
            toast.setText("Usage Breakdown");
            toast.show();
        }
        else if (selectedPosition == 4) {
            CharSequence[] ch = { "Smart Power Strip - Dannie Alfaro\n\nApp
Version: " + getVersion() +
                    "\nBy: Josh Logier\n\nThe goal of this app is to
interface seamlessly with " +
                    "the Smart Power Strip in order to remotely control
connected devices and track power usage." };
            showDialog("About", ch);
        }
    }

    private void setupDrawer() {
        mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout,
R.string.navigation_drawer_open, R.string.navigation_drawer_close) {
```

```
            /** Called when a drawer has settled in a completely open state.
*/
            public void onDrawerOpened(View drawerView) {
                super.onDrawerOpened(drawerView);
                getSupportActionBar().setTitle("Options");
                invalidateOptionsMenu(); // creates call to
onPrepareOptionsMenu()
            }

            /** Called when a drawer has settled in a completely closed
state. */
            public void onDrawerClosed(View view) {
                super.onDrawerClosed(view);
                getSupportActionBar().setTitle(mActivityTitle);
                invalidateOptionsMenu(); // creates call to
onPrepareOptionsMenu()
            }
        };

        mDrawerToggle.setDrawerIndicatorEnabled(true);
        mDrawerLayout.setDrawerListener(mDrawerToggle);
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        mDrawerToggle.onConfigurationChanged(newConfig);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.main, menu);

        //Sets progress bar as not visible initially
        MenuItem loading = menu.findItem(R.id.progress_bar);
        loading.setVisible(false);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_refresh) {
            //Request new data from Raspberry Pi
            Thread t = requestData();
            t.start();
            try {
                t.join();
            } catch (InterruptedException e) {
```

```
                e.printStackTrace();
            }

            return true;
        }

        // Activate the navigation drawer toggle
        if (mDrawerToggle.onOptionsItemSelected(item)) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onBackPressed() {
        // change the title
        FragmentManager fm = this.getFragmentManager();
        fm.popBackStackImmediate();
        Fragment cur = fm.findFragmentById(R.id.content_frame);


getSupportActionBar().setNavigationMode(ActionBar.NAVIGATION_MODE_STANDARD);

        if (mDrawerLayout.isDrawerOpen(mDrawerList)) {
            mDrawerLayout.closeDrawer(mDrawerList);
        }
        else if (getFragmentManager().getBackStackEntryCount() > 1) {
            if (cur instanceof OnStartFragment) {
                selectedPosition = 0;
                getSupportActionBar().setTitle(mDrawerTitles[0]);
            } else if (cur instanceof RemoteControlFragment) {
                selectedPosition = 1;
                getSupportActionBar().setTitle(mDrawerTitles[1]);
            } else if (cur instanceof BarChartFragment) {
                selectedPosition = 2;
                getSupportActionBar().setTitle(mDrawerTitles[2]);
            } else if (cur instanceof PieChartFragment) {
                selectedPosition = 3;
                getSupportActionBar().setTitle(mDrawerTitles[3]);
            }
        }
        else
            super.onBackPressed();
    }

    public static boolean isTablet(Context context) {
        return (context.getResources().getConfiguration().screenLayout
                & Configuration.SCREENLAYOUT_SIZE_MASK)
                >= Configuration.SCREENLAYOUT_SIZE_LARGE;
    }

    public void showDialog(String title, CharSequence[] items) {
        Log.d("DEBUG", "function showdialog");
        final CharSequence[] fitems = items;
```

```
        AlertDialog.Builder lmenu = new AlertDialog.Builder(this,
AlertDialog.THEME_DEVICE_DEFAULT_LIGHT);

        final AlertDialog ad = lmenu.create();
        lmenu.setTitle(title);
        lmenu.setMessage(fitems[0]);
        lmenu.setPositiveButton("Close", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                ad.dismiss();
            }
        });
        lmenu.show();
    }

    //Gets version number to display in about so I don't need to remember to
update this
    private String getVersion() {
        PackageInfo pInfo = null;
        try {
            pInfo = getPackageManager().getPackageInfo(getPackageName(), 0);
        } catch (PackageManager.NameNotFoundException e) {
            e.printStackTrace();
        }
        return pInfo.versionName;
    }

    //Sends one packet to the Raspberry Pi that requests updated data
    public Thread requestData() {
        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {
                if (checkConnection()) {
                    try {
                        Log.d("DEBUG", "STARTING THREAD REQUEST DATA");
                        DatagramSocket clientsocket = new
DatagramSocket(null);
                        byte[] receivedata = new String("data!").getBytes();
                        DatagramPacket send_packet = new
DatagramPacket(receivedata, receivedata.length,
                                //Sets the local address we are sending to
                                InetAddress.getByAddress(new byte[]{(byte)
192, (byte) 168, (byte) 43, (byte) 40}), 5006);
                        clientsocket.send(send_packet);
                        clientsocket.close();
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
                else {
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            Toast.makeText(getApplicationContext(), "Not
connected to Smart Power Strip", Toast.LENGTH_SHORT).show();
                        }
                    });
```

```
                }
            }
        });
        return t;
    }

    private void dataReceiver() {
        final SharedPreferences.Editor editor = prefs.edit();

        new Thread(new Runnable() {
            public void run() {
                try {
                    DatagramSocket clientsocket = new DatagramSocket(null);
                    clientsocket.setReuseAddress(true);
                    clientsocket.setBroadcast(true);
                    clientsocket.bind(new InetSocketAddress(5005));
                    final byte[] receivedata = new byte[1500];

                    while(true) {
                        DatagramPacket recv_packet = new
DatagramPacket(receivedata, receivedata.length);
                        Log.d("UDP", "S: Receiving...");
                        clientsocket.receive(recv_packet);
                        final String receivedstring = new
String(recv_packet.getData());

                        //Need to update UI code on main thread
                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                Toast.makeText(getApplicationContext(), "" +
receivedstring, Toast.LENGTH_SHORT).show();
                                String [] splits = receivedstring.split(" ");
                                float value1 = Float.parseFloat(splits[0]);
                                float value2 = Float.parseFloat(splits[1]);
                                float value3 = Float.parseFloat(splits[2]);
                                float value4 = Float.parseFloat(splits[3]);

                                float total = value1 + value2 + value3 +
value4;
                                editor.putFloat("total_power", total);

                                editor.putFloat("outlet1_power", value1);
                                editor.putFloat("outlet2_power", value2);
                                editor.putFloat("outlet3_power", value3);
                                editor.putFloat("outlet4_power", value4);
                                editor.commit();
                            }
                        });

                        InetAddress ipaddress = recv_packet.getAddress();
                        int port = recv_packet.getPort();
                        Log.d("UDP", "IPAddress : " + ipaddress.toString());
                        Log.d("UDP", "Port : " + Integer.toString(port));
                    }

                } catch (SocketException e) {
```

```
                    Log.e("UDP", "Socket Error", e);
                } catch (IOException e) {
                    Log.e("UDP", "IO Error", e);
                }
            }
        }).start();
    }

    public boolean isOnline() {
        Runtime runtime = Runtime.getRuntime();
        try {

            Process ipProcess = runtime.exec("/system/bin/ping -c 5
192.168.43.40");
            int     exitValue = ipProcess.waitFor();
            return (exitValue == 0);

        } catch (IOException e)          { e.printStackTrace(); }
        catch (InterruptedException e) { e.printStackTrace(); }

        return false;
    }

    private boolean checkConnection() {
        try {
            byte[] b = new byte[]{(byte) 192, (byte) 168, (byte) 43, (byte)
40};
            InetAddress addr = InetAddress.getByAddress(b);
            if (addr.isReachable(325))
                return true;
            else
                return false;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return false;
    }
}
```

RemoteControlFragment.java

```
// ------------------------------------------------------------------------
// Name:    Smart Power Strip Input/Output
// By:      Josh Logier & Dannie Alfaro
//          RemoteControlFragment.java
// ------------------------------------------------------------------------
//
// Project Name:    Smart Power Strip App
// Date Modified:   6/8/15
// Software:        Android Studio
// Hardware:        Android Smart Device
// Files Needed:    CustomListAdapter.java, MyMarkerView.java,
//                  OnStartFragment.java, MainActivity.java,
//                  PieChartFragment.java, SimpleFragment.java,
//                  RemoteControlFragment.java, BarChartFragment.java
// Connections:     Wi-Fi
// Purpose:         Senior Project
// ------------------------------------------------------------------------

package com.example.smartpower.smartpowerstripapp;

import android.app.AlertDialog;
import android.app.Fragment;
import android.content.Context;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.preference.ListPreference;
import android.preference.Preference;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.Switch;
import android.widget.Toast;

import java.io.IOException;
import java.net.Inet4Address;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.prefs.Preferences;

public class RemoteControlFragment extends Fragment {
    Context activity;
    private boolean networkFlag = false;
    protected SharedPreferences prefs;
```

```
    private String[] defaultNames = {"Outlet 1", "Outlet 2", "Outlet 3",
"Outlet 4", "Outlet 5", "Outlet 6"};

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        /* Creating view corresponding to the fragment */
        View v = inflater.inflate(R.layout.fragment_remote, container,
false);
        activity = getActivity();
        prefs =
activity.getSharedPreferences("com.example.smartpower.smartpowerstripapp",
Context.MODE_PRIVATE);

        //Initial ping to Pi on launch
        Thread t = checkConnection();
        t.start();

        final Switch switch1 = (Switch) v.findViewById(R.id.switch1);
        switch1.setChecked(prefs.getBoolean("switch1", false));
        final Switch switch2 = (Switch) v.findViewById(R.id.switch2);
        switch2.setChecked(prefs.getBoolean("switch2", false));
        final Switch switch3 = (Switch) v.findViewById(R.id.switch3);
        switch3.setChecked(prefs.getBoolean("switch3", false));
        final Switch switch4 = (Switch) v.findViewById(R.id.switch4);
        switch4.setChecked(prefs.getBoolean("switch4", false));
        final Switch switch5 = (Switch) v.findViewById(R.id.switch5);
        switch5.setChecked(prefs.getBoolean("switch5", false));
        final Switch switch6 = (Switch) v.findViewById(R.id.switch6);
        switch6.setChecked(prefs.getBoolean("switch6", false));

        final Button button1 = (Button) v.findViewById(R.id.button1);
        final Button button2 = (Button) v.findViewById(R.id.button2);
        final Button button3 = (Button) v.findViewById(R.id.button3);
        final Button button4 = (Button) v.findViewById(R.id.button4);
        final Button button5 = (Button) v.findViewById(R.id.button5);
        final Button button6 = (Button) v.findViewById(R.id.button6);

        button1.setText(prefs.getString("outlet1", "Outlet 1"));
        button2.setText(prefs.getString("outlet2", "Outlet 2"));
        button3.setText(prefs.getString("outlet3", "Outlet 3"));
        button4.setText(prefs.getString("outlet4", "Outlet 4"));
        button5.setText(prefs.getString("outlet5", "Outlet 5"));
        button6.setText(prefs.getString("outlet6", "Outlet 6"));

        switchToggle(switch1, 0);
        switchToggle(switch2, 1);
        switchToggle(switch3, 2);
        switchToggle(switch4, 3);
        switchToggle(switch5, 4);
        switchToggle(switch6, 5);

        buttonToggle(button1, 1);
        buttonToggle(button2, 2);
        buttonToggle(button3, 3);
        buttonToggle(button4, 4);
        buttonToggle(button5, 5);
```

```
        buttonToggle(button6, 6);

        return v;
    }

    public void buttonToggle(final Button button, final int num) {
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                changeName(button, num);
            }
        });
    }

    public void switchToggle(Switch sw, final int num){
        final SharedPreferences.Editor editor = prefs.edit();

        sw.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                Toast toast = Toast.makeText(activity, "",
Toast.LENGTH_SHORT);
                Thread t = checkConnection();

                t.start();
                try {
                    t.join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

                if (networkFlag) {
                    // Perform action on click
                    if (isChecked) {
                        toast.setText("Outlet " + (num + 1) + " ON");
                        toast.show();
                        relayToggle(num, 1);

                        if (num == 0)
                            editor.putBoolean("switch1", true);
                        else if (num == 1)
                            editor.putBoolean("switch2", true);
                        else if (num == 2)
                            editor.putBoolean("switch3", true);
                        else if (num == 3)
                            editor.putBoolean("switch4", true);
                        else if (num == 4)
                            editor.putBoolean("switch5", true);
                        else if (num == 5)
                            editor.putBoolean("switch6", true);
                        editor.commit();

                    } else {
                        toast.setText("Outlet " + (num + 1) + " OFF");
                        toast.show();
                        relayToggle(num, 0);
```

```
                        if (num == 0)
                            editor.putBoolean("switch1", false);
                        else if (num == 1)
                            editor.putBoolean("switch2", false);
                        else if (num == 2)
                            editor.putBoolean("switch3", false);
                        else if (num == 3)
                            editor.putBoolean("switch4", false);
                        else if (num == 4)
                            editor.putBoolean("switch5", false);
                        else if (num == 5)
                            editor.putBoolean("switch6", false);
                        editor.commit();
                    }
                } else {
                    if (isChecked)
                        buttonView.setChecked(false);
                    else
                        buttonView.setChecked(true);

                    CharSequence[] ch = {"Please connect to the same WiFi
network as the smart power strip, then try again."};
                    showDialog("Connection Error", ch);
                }
            }
        });
    }

    private void relayToggle(int outlet, int state) {
        new WebView(activity).loadUrl("http://192.168.43.40/gpio.php?pin=" +
outlet + "&status=" + state);
    }

    private void showDialog(String title, CharSequence[] items) {
        Log.d("DEBUG", "function showdialog");
        final CharSequence[] fitems = items;

        AlertDialog.Builder lmenu = new AlertDialog.Builder(activity,
AlertDialog.THEME_DEVICE_DEFAULT_LIGHT);

        final AlertDialog ad = lmenu.create();
        lmenu.setTitle(title);
        lmenu.setMessage(fitems[0]);
        lmenu.setPositiveButton("Close", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                ad.dismiss();
            }
        });
        lmenu.show();
    }

    private Thread checkConnection() {
        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {
```

```
                try {
                    byte[] b = new byte[]{(byte) 192, (byte) 168, (byte) 43,
(byte) 40};
                    InetAddress addr = InetAddress.getByAddress(b);
                    if (addr.isReachable(325))
                        networkFlag = true;
                    else
                        networkFlag = false;
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });

        return t;
    }

    //Method can be used to pop up a dialog box to change the person you want
to call without the need to register again
    void changeName(final Button button, final int num) {
        LayoutInflater factory = LayoutInflater.from(getActivity());
        final View textEntryView =
factory.inflate(R.layout.change_name_dialog, null);
        final AlertDialog.Builder alert = new
AlertDialog.Builder(getActivity(), AlertDialog.THEME_DEVICE_DEFAULT_LIGHT);
        alert.setView(textEntryView);
        final EditText newName = (EditText)
textEntryView.findViewById(R.id.newName);

        final SharedPreferences.Editor editor = prefs.edit();

        alert.setTitle("Change outlet name?")
                .setPositiveButton("OK",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int
whichButton) {
                                String temp = newName.getText().toString();
                                if (!temp.equals("")) {
                                    button.setText(temp);
                                    editor.putString("outlet" + num, temp);
                                    editor.apply();
                                }
                            }
                        })
                .setNegativeButton("Cancel",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int
whichButton) {
                                dialog.cancel();
                            }
                        });
        alert.show();
    }
}
```

PieChartFragment.java

```
// ------------------------------------------------------------------------
// Name:    Smart Power Strip Input/Output
// By:      Josh Logier
//          PieChartFragment.java
// ------------------------------------------------------------------------
//
// Project Name:    Smart Power Strip App
// Date Modified:   6/8/15
// Software:        Android Studio
// Hardware:        Android Smart Device
// Files Needed:    CustomListAdapter.java, MyMarkerView.java,
//                  OnStartFragment.java, MainActivity.java,
//                  PieChartFragment.java, SimpleFragment.java,
//                  RemoteControlFragment.java, BarChartFragment.java
// Connections:     Wi-Fi
// Purpose:         Senior Project
// ------------------------------------------------------------------------

package com.example.smartpower.smartpowerstripapp;

import android.content.Context;
import android.content.res.Configuration;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.github.mikephil.charting.charts.PieChart;
import com.github.mikephil.charting.components.Legend;
import com.github.mikephil.charting.components.Legend.LegendPosition;


public class PieChartFragment extends SimpleFragment {

    public static PieChartFragment newInstance() {
        return new PieChartFragment();
    }

    private PieChart mChart;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_piechart, container,
false);

        mChart = (PieChart) v.findViewById(R.id.pieChart1);
        mChart.setDescription("");
        mChart.setCenterText("Power\nUsage\n(%)");
        mChart.setCenterTextSize(22f);
        mChart.setUsePercentValues(true);

        // radius of the center hole in percent of maximum radius
        mChart.setHoleRadius(45f);
```

```
        mChart.setTransparentCircleRadius(50f);

        mChart.setData(generatePieData());

        Legend l = mChart.getLegend();
        l.setForm(Legend.LegendForm.CIRCLE);
        int orientation =
getActivity().getResources().getConfiguration().orientation;
        if (orientation == Configuration.ORIENTATION_PORTRAIT) {
            if (isTablet(getActivity())) {
                l.setTextSize(24f);
                l.setPosition(Legend.LegendPosition.BELOW_CHART_CENTER);
            }
            else {
                l.setTextSize(19.5f);
                l.setPosition(Legend.LegendPosition.BELOW_CHART_LEFT);
            }
        }
        else {
            if (isTablet(getActivity()))
                l.setTextSize(26f);
            else
                l.setTextSize(24f);
            l.setPosition(LegendPosition.RIGHT_OF_CHART_CENTER);
        }

        return v;
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        Legend l = mChart.getLegend();

        // Checks the orientation of the screen
        if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
            l.setPosition(LegendPosition.RIGHT_OF_CHART_CENTER);
            mChart.invalidate();
        } else if (newConfig.orientation ==
Configuration.ORIENTATION_PORTRAIT){
            if (isTablet(getActivity()))
                l.setPosition(Legend.LegendPosition.BELOW_CHART_CENTER);
            else
                l.setPosition(LegendPosition.BELOW_CHART_LEFT);

            mChart.refreshDrawableState();
        }
    }

    public static boolean isTablet(Context context) {
        return (context.getResources().getConfiguration().screenLayout
                & Configuration.SCREENLAYOUT_SIZE_MASK)
                >= Configuration.SCREENLAYOUT_SIZE_LARGE;
    }
}
```

BarChartFragment.java

```
// ----------------------------------------------------------------------
// Name:        Smart Power Strip Input/Output
// Modified By: Josh Logier
//              BarChartFragment.java
// ----------------------------------------------------------------------
//
// Project Name:    Smart Power Strip App
// Date Modified:   6/8/15
// Software:        Android Studio
// Hardware:        Android Smart Device
// Files Needed:    CustomListAdapter.java, MyMarkerView.java,
//                  OnStartFragment.java, MainActivity.java,
//                  PieChartFragment.java, SimpleFragment.java,
//                  RemoteControlFragment.java, BarChartFragment.java
// Connections:     Wi-Fi
// Purpose:         Senior Project
// ----------------------------------------------------------------------

/*
 * Copyright 2014 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *       http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.smartpower.smartpowerstripapp;

import android.content.Context;
import android.content.res.Configuration;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.FrameLayout;

import com.github.mikephil.charting.charts.BarChart;
import com.github.mikephil.charting.components.Legend;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.components.YAxis;
import com.github.mikephil.charting.listener.OnChartGestureListener;
```

```
public class BarChartFragment extends SimpleFragment implements
OnChartGestureListener {

    public static BarChartFragment newInstance() {
        return new BarChartFragment();
    }

    private BarChart mChart;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_barchart, container,
false);

        // create a new chart object
        mChart = new BarChart(getActivity());
        mChart.setDescription("");
        mChart.setOnChartGestureListener(this);

        MyMarkerView mv = new MyMarkerView(getActivity(),
R.layout.custom_marker_view);

        mChart.setMarkerView(mv);
        mChart.setHighlightIndicatorEnabled(true);
        mChart.setDrawGridBackground(false);
        mChart.setDrawBarShadow(false);
        mChart.setData(generateBarData(4));
        mChart.getAxisRight().setEnabled(false);

        XAxis xAxis = mChart.getXAxis();
        xAxis.setEnabled(true);

        // programatically add the chart
        FrameLayout parent = (FrameLayout) v.findViewById(R.id.parentLayout);
        parent.addView(mChart);

        Legend l = mChart.getLegend();
        l.setForm(Legend.LegendForm.CIRCLE);
        int orientation =
getActivity().getResources().getConfiguration().orientation;
        if (orientation == Configuration.ORIENTATION_PORTRAIT) {
            if (isTablet(getActivity())) {
                l.setTextSize(24f);
                l.setPosition(Legend.LegendPosition.BELOW_CHART_CENTER);
            }
            else {
                l.setTextSize(18f);
            l.setPosition(Legend.LegendPosition.BELOW_CHART_LEFT);
            }

        }
        else {
            if (isTablet(getActivity()))
                l.setTextSize(24f);
            else
                l.setTextSize(18f);
```

```
            l.setPosition(Legend.LegendPosition.BELOW_CHART_CENTER);
        }

        return v;
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        Legend l = mChart.getLegend();

        // Checks the orientation of the screen
        if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
            l.setPosition(Legend.LegendPosition.BELOW_CHART_CENTER);
            mChart.invalidate();
        } else if (newConfig.orientation ==
Configuration.ORIENTATION_PORTRAIT){
            l.setPosition(Legend.LegendPosition.BELOW_CHART_LEFT);
            mChart.invalidate();
        }
    }

    public static boolean isTablet(Context context) {
        return (context.getResources().getConfiguration().screenLayout
                & Configuration.SCREENLAYOUT_SIZE_MASK)
                >= Configuration.SCREENLAYOUT_SIZE_LARGE;
    }

    @Override
    public void onChartLongPressed(MotionEvent me) {
        Log.i("LongPress", "Chart longpressed.");
    }

    @Override
    public void onChartDoubleTapped(MotionEvent me) {
        Log.i("DoubleTap", "Chart double-tapped.");
    }

    @Override
    public void onChartSingleTapped(MotionEvent me) {
        Log.i("SingleTap", "Chart single-tapped.");
    }

    @Override
    public void onChartFling(MotionEvent me1, MotionEvent me2, float
velocityX, float velocityY) {
        Log.i("Fling", "Chart flinged. VeloX: " + velocityX + ", VeloY: " +
velocityY);
    }

    @Override
    public void onChartScale(MotionEvent me, float scaleX, float scaleY) {
        Log.i("Scale / Zoom", "ScaleX: " + scaleX + ", ScaleY: " + scaleY);
    }

    @Override
    public void onChartTranslate(MotionEvent me, float dX, float dY) {
```

```
            Log.i("Translate / Move", "dX: " + dX + ", dY: " + dY);
        }
    }
```

OnStartFragment.java

```
// ---------------------------------------------------------------------------
// Name:     Smart Power Strip Input/Output
// By:       Josh Logier
//           OnStartFragment.java
// ---------------------------------------------------------------------------
//
// Project Name:     Smart Power Strip App
// Date Modified:    6/8/15
// Software:         Android Studio
// Hardware:         Android Smart Device
// Files Needed:     CustomListAdapter.java, MyMarkerView.java,
//                   OnStartFragment.java, MainActivity.java,
//                   PieChartFragment.java, SimpleFragment.java,
//                   RemoteControlFragment.java, BarChartFragment.java
// Connections:      Wi-Fi
// Purpose:          Senior Project
// ---------------------------------------------------------------------------

package com.example.smartpower.smartpowerstripapp;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

public class OnStartFragment extends Fragment {
    protected SharedPreferences prefs;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Defines the xml file for the fragment
        View v = inflater.inflate(R.layout.fragment_start, container, false);
        TextView totalPower = (TextView) v.findViewById(R.id.totalPowerText);
        TextView totalCost = (TextView) v.findViewById(R.id.totalCostText);

        prefs =
getActivity().getSharedPreferences("com.example.smartpower.smartpowerstripapp
", Context.MODE_PRIVATE);
        float power_total = prefs.getFloat("total_power", (float)0);
        if (power_total >= 1000)
            totalPower.setText("Total Energy Used:       "
+Integer.toString((int)(power_total / 1000)) + " kWh");
        else
            totalPower.setText("Total Energy Used:       " +
Integer.toString((int)power_total) + " Wh");

        totalCost.setText("Cost*:                              $" +
String.format("%.3f", power_total * (float)0.000121));
```

```
            return v;
        }
    }
```

CustomListAdapter.java

```
// ------------------------------------------------------------------------
// Name:    Smart Power Strip Input/Output
// By:      Josh Logier
//          CustomListAdapter.java
// ------------------------------------------------------------------------
//
// Project Name:    Smart Power Strip App
// Date Modified:   6/8/15
// Software:        Android Studio
// Hardware:        Android Smart Device
// Files Needed:    CustomListAdapter.java, MyMarkerView.java,
//                  OnStartFragment.java, MainActivity.java,
//                  PieChartFragment.java, SimpleFragment.java,
//                  RemoteControlFragment.java, BarChartFragment.java
// Connections:     Wi-Fi
// Purpose:         Senior Project
// ------------------------------------------------------------------------

package com.example.smartpower.smartpowerstripapp;

import android.app.Activity;
import android.content.Context;
import android.content.res.Configuration;
import android.util.TypedValue;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;

public class CustomListAdapter extends ArrayAdapter<String> {

    private final Activity context;
    private final String[] itemname;
    private final Integer[] imgid;

    public CustomListAdapter(Activity context, String[] itemname, Integer[]
imgid) {
        super(context, R.layout.mylist, itemname);

        this.context = context;
        this.itemname = itemname;
        this.imgid = imgid;
    }

    public View getView(int position, View view, ViewGroup parent) {
        LayoutInflater inflater = context.getLayoutInflater();
        View rowView = inflater.inflate(R.layout.mylist, null, true);

        TextView txtTitle = (TextView) rowView.findViewById(R.id.text1);
        ImageView imageView = (ImageView) rowView.findViewById(R.id.icIcon);
```

```
        if (isTablet(context)) {
            txtTitle.setTextSize(TypedValue.COMPLEX_UNIT_SP, 16);
        }
        else {
            txtTitle.setTextSize(TypedValue.COMPLEX_UNIT_SP, 14);
        }

        LinearLayout.LayoutParams params1 = new LinearLayout.LayoutParams(new
LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,
LinearLayout.LayoutParams.WRAP_CONTENT));
        if (imgid[position] != null) {
            params1.setMargins(25, 0, 0, 0);
            txtTitle.setLayoutParams(params1);

            imageView.setImageResource(imgid[position]);
        }
        else {
            params1.setMargins(0, 0, 0, 0);
            txtTitle.setLayoutParams(params1);

            params1.setMargins(dpToPix(10), 0, 0, 0);
            imageView.setLayoutParams(params1);
        }
        txtTitle.setText(itemname[position]);
        return rowView;
    };

    public static boolean isTablet(Context context) {
        return (context.getResources().getConfiguration().screenLayout
                & Configuration.SCREENLAYOUT_SIZE_MASK)
                >= Configuration.SCREENLAYOUT_SIZE_LARGE;
    }

    //This method allows layout params to scale identically across any device
by converting dp to pixels for that specific device
    private int dpToPix(float dp) {
        // The gesture threshold expressed in dp
        final float GESTURE_THRESHOLD_DP = dp;

        // Get the screen's density scale
        final float scale =
context.getResources().getDisplayMetrics().density;
        // Convert the dps to pixels, based on density scale
        int mGestureThreshold = (int) (GESTURE_THRESHOLD_DP * scale + 0.5f);
        return mGestureThreshold;
    }
}
```

SimpleFragment.java

```
// ------------------------------------------------------------------------
// Name:    Smart Power Strip Input/Output
// By:      Josh Logier
//          SimpleFragment.java
// ------------------------------------------------------------------------
//
// Project Name:    Smart Power Strip App
// Date Modified:   6/8/15
// Software:        Android Studio
// Hardware:        Android Smart Device
// Files Needed:    CustomListAdapter.java, MyMarkerView.java,
//                  OnStartFragment.java, MainActivity.java,
//                  PieChartFragment.java, SimpleFragment.java,
//                  RemoteControlFragment.java, BarChartFragment.java
// Connections:     Wi-Fi
// Purpose:         Senior Project
// ------------------------------------------------------------------------

package com.example.smartpower.smartpowerstripapp;

import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.Typeface;
import android.os.Bundle;
import android.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.github.mikephil.charting.charts.ScatterChart;
import com.github.mikephil.charting.charts.ScatterChart.ScatterShape;
import com.github.mikephil.charting.data.BarData;
import com.github.mikephil.charting.data.BarDataSet;
import com.github.mikephil.charting.data.BarEntry;
import com.github.mikephil.charting.data.ChartData;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.data.PieData;
import com.github.mikephil.charting.data.PieDataSet;
import com.github.mikephil.charting.data.ScatterData;
import com.github.mikephil.charting.data.ScatterDataSet;
import com.github.mikephil.charting.utils.ColorTemplate;
import com.github.mikephil.charting.utils.FileUtils;

import java.util.ArrayList;

public abstract class SimpleFragment extends Fragment {

    private Typeface tf;
    protected SharedPreferences prefs;

    //Our color palette (green, yellow, orange, red, purple, indigo)
```

```
    public static final int[] MATERIAL_COLORS = new int[]{Color.rgb(56, 142,
60), Color.rgb(251, 192, 45), Color.rgb(245, 124, 0), Color.rgb(211, 47, 47),
Color.rgb(123, 31, 162), Color.rgb(48, 63, 159)};

    public SimpleFragment() {

    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        tf = Typeface.createFromAsset(getActivity().getAssets(), "OpenSans-
Regular.ttf");
        return super.onCreateView(inflater, container, savedInstanceState);
    }

    /*protected BarData generateBarData(int dataSets, float range, int count)
{

        ArrayList<BarDataSet> sets = new ArrayList<BarDataSet>();
        ArrayList<String> xVals = new ArrayList<String>();

        prefs =
getActivity().getSharedPreferences("com.example.smartpower.smartpowerstripapp
", Context.MODE_PRIVATE);

        xVals.add(prefs.getString("outlet1", "Outlet 1"));
        xVals.add(prefs.getString("outlet2", "Outlet 2"));
        xVals.add(prefs.getString("outlet3", "Outlet 3"));
        xVals.add(prefs.getString("outlet4", "Outlet 4"));
        xVals.add(prefs.getString("outlet5", "Outlet 5"));
        xVals.add(prefs.getString("outlet6", "Outlet 6"));

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

            ArrayList<BarEntry> entries = new ArrayList<BarEntry>();

            //entries =
FileUtils.loadEntriesFromAssets(getActivity().getAssets(),
"stacked_bars.txt");

            for(int j = 0; j < count; j++) {
                entries.add(new BarEntry((float) (Math.random() * range) +
range / 4, j));
            }

            BarDataSet ds = new BarDataSet(entries, "Power Usage Per Outlet
(Watts)");
            ds.setColors(MATERIAL_COLORS);
            sets.add(ds);
        }

        BarData d = new BarData(xVals, sets);
        d.setValueTypeface(tf);
        d.setValueTextSize(18f);
        return d;
    }*/
```

```
    protected BarData generateBarData(int count) {

        ArrayList<BarDataSet> sets = new ArrayList<BarDataSet>();
        ArrayList<String> xVals = new ArrayList<String>();

        prefs =
getActivity().getSharedPreferences("com.example.smartpower.smartpowerstripapp
", Context.MODE_PRIVATE);

        xVals.add(prefs.getString("outlet1", "Outlet 1/2"));
        xVals.add(prefs.getString("outlet2", "Outlet 3/4"));
        xVals.add(prefs.getString("outlet3", "Outlet 5/6"));
        xVals.add(prefs.getString("outlet4", "Outlet 7/8"));

        ArrayList<BarEntry> entries = new ArrayList<BarEntry>();

        //entries =
FileUtils.loadEntriesFromAssets(getActivity().getAssets(),
"stacked_bars.txt");
        entries.clear();

        for(int j = 0; j < count; j++) {
            //entries.add(new BarEntry((float) (Math.random() * range) +
range / 4, j));
            entries.add(new BarEntry(prefs.getFloat("outlet" + (j + 1) +
"_power", 0), j));
        }

        BarDataSet ds = new BarDataSet(entries, "Energy Usage Per Outlet
(Watthours)");
        ds.setColors(MATERIAL_COLORS);
        sets.add(ds);

        BarData d = new BarData(xVals, sets);
        d.setValueTypeface(tf);
        d.setValueTextSize(18f);
        return d;
    }

    protected ScatterData generateScatterData(int dataSets, float range, int
count) {

        ArrayList<ScatterDataSet> sets = new ArrayList<ScatterDataSet>();

        ScatterShape[] shapes = ScatterChart.getAllPossibleShapes();

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

            ArrayList<Entry> entries = new ArrayList<Entry>();

            for(int j = 0; j < count; j++) {
                entries.add(new Entry((float) (Math.random() * range) + range
/ 4, j));
            }

            ScatterDataSet ds = new ScatterDataSet(entries, getLabel(i));
```

```
            ds.setScatterShapeSize(12f);
            ds.setScatterShape(shapes[i % shapes.length]);
            ds.setColors(MATERIAL_COLORS);
            ds.setScatterShapeSize(9f);
            sets.add(ds);
        }

        ScatterData d = new ScatterData(ChartData.generateXVals(0, count),
sets);
        d.setValueTypeface(tf);
        return d;
    }

    /**
     * generates less data (1 DataSet, 4 values)
     * @return
     */
    protected PieData generatePieData() {

        int count = 4;

        ArrayList<Entry> entries1 = new ArrayList<Entry>();
        ArrayList<String> xVals = new ArrayList<String>();

        prefs =
getActivity().getSharedPreferences("com.example.smartpower.smartpowerstripapp
", Context.MODE_PRIVATE);

        xVals.add(prefs.getString("outlet1", "Outlet 1"));
        xVals.add(prefs.getString("outlet2", "Outlet 2"));
        xVals.add(prefs.getString("outlet3", "Outlet 3"));
        xVals.add(prefs.getString("outlet4", "Outlet 4"));

        for(int i = 0; i < count; i++) {
            //xVals.add("entry" + (i+1));
            entries1.add(new BarEntry(prefs.getFloat("outlet" + (i + 1) +
"_power", 0), i));
            //entries1.add(new Entry((float) (Math.random() * 60) + 40, i));
        }

        PieDataSet ds1 = new PieDataSet(entries1, "Outlet Usage Percentage");
        ds1.setColors(MATERIAL_COLORS);
        ds1.setSliceSpace(2f);
        ds1.setValueTextColor(Color.WHITE);
        ds1.setValueTextSize(12f);

        PieData d = new PieData(xVals, ds1);
        d.setValueTypeface(tf);
        d.setValueTextSize(18f);

        return d;
    }

    protected LineData generateLineData() {

//        DataSet ds1 = new DataSet(n, "O(n)");
//        DataSet ds2 = new DataSet(nlogn, "O(nlogn)");
```

```
//        DataSet ds3 = new DataSet(nsquare, "O(n\u00B2)");
//        DataSet ds4 = new DataSet(nthree, "O(n\u00B3)");

        ArrayList<LineDataSet> sets = new ArrayList<LineDataSet>();

        LineDataSet ds1 = new
LineDataSet(FileUtils.loadEntriesFromAssets(getActivity().getAssets(),
"sine.txt"), "Sine function");
        LineDataSet ds2 = new
LineDataSet(FileUtils.loadEntriesFromAssets(getActivity().getAssets(),
"cosine.txt"), "Cosine function");

        ds1.setLineWidth(2f);
        ds2.setLineWidth(2f);

        ds1.setDrawCircles(false);
        ds2.setDrawCircles(false);

        ds1.setColor(ColorTemplate.VORDIPLOM_COLORS[0]);
        ds2.setColor(ColorTemplate.VORDIPLOM_COLORS[1]);

        // load DataSets from textfiles in assets folders
        sets.add(ds1);
        sets.add(ds2);

//        sets.add(FileUtils.dataSetFromAssets(getActivity().getAssets(),
"n.txt"));
//        sets.add(FileUtils.dataSetFromAssets(getActivity().getAssets(),
"nlogn.txt"));
//        sets.add(FileUtils.dataSetFromAssets(getActivity().getAssets(),
"square.txt"));
//        sets.add(FileUtils.dataSetFromAssets(getActivity().getAssets(),
"three.txt"));

        int max = Math.max(sets.get(0).getEntryCount(),
sets.get(1).getEntryCount());

        LineData d = new LineData(ChartData.generateXVals(0, max),  sets);
        d.setValueTypeface(tf);
        return d;
    }

    protected LineData getComplexity() {

        ArrayList<LineDataSet> sets = new ArrayList<LineDataSet>();

        LineDataSet ds1 = new
LineDataSet(FileUtils.loadEntriesFromAssets(getActivity().getAssets(),
"n.txt"), "O(n)");
        LineDataSet ds2 = new
LineDataSet(FileUtils.loadEntriesFromAssets(getActivity().getAssets(),
"nlogn.txt"), "O(nlogn)");
        LineDataSet ds3 = new
LineDataSet(FileUtils.loadEntriesFromAssets(getActivity().getAssets(),
"square.txt"), "O(n\u00B2)");
```

```
        LineDataSet ds4 = new
LineDataSet(FileUtils.loadEntriesFromAssets(getActivity().getAssets(),
"three.txt"), "O(n\u00B3)");

        ds1.setColor(ColorTemplate.VORDIPLOM_COLORS[0]);
        ds2.setColor(ColorTemplate.VORDIPLOM_COLORS[1]);
        ds3.setColor(ColorTemplate.VORDIPLOM_COLORS[2]);
        ds4.setColor(ColorTemplate.VORDIPLOM_COLORS[3]);

        ds1.setCircleColor(ColorTemplate.VORDIPLOM_COLORS[0]);
        ds2.setCircleColor(ColorTemplate.VORDIPLOM_COLORS[1]);
        ds3.setCircleColor(ColorTemplate.VORDIPLOM_COLORS[2]);
        ds4.setCircleColor(ColorTemplate.VORDIPLOM_COLORS[3]);

        ds1.setLineWidth(2.5f);
        ds1.setCircleSize(3f);
        ds2.setLineWidth(2.5f);
        ds2.setCircleSize(3f);
        ds3.setLineWidth(2.5f);
        ds3.setCircleSize(3f);
        ds4.setLineWidth(2.5f);
        ds4.setCircleSize(3f);


        // load DataSets from textfiles in assets folders
        sets.add(ds1);
        sets.add(ds2);
        sets.add(ds3);
        sets.add(ds4);

        LineData d = new LineData(ChartData.generateXVals(0,
ds1.getEntryCount()), sets);
        d.setValueTypeface(tf);
        return d;
    }

    private String[] mLabels = new String[] { "Company A", "Company B",
"Company C", "Company D", "Company E", "Company F" };
//    private String[] mXVals = new String[] { "Jan", "Feb", "Mar", "Apr",
"May", "Jun", "Jul", "Aug", "Sep", "Okt", "Nov", "Dec" };

    private String getLabel(int i) {
        return mLabels[i];
    }
}
```

MyMarkerView.java

```
// ----------------------------------------------------------------------
// Name:        Smart Power Strip Input/Output
// Modified By: Josh Logier
//              MyMarkerView.java
// ----------------------------------------------------------------------
//
// Project Name:    Smart Power Strip App
// Date Modified:   6/8/15
// Software:        Android Studio
// Hardware:        Android Smart Device
// Files Needed:    CustomListAdapter.java, MyMarkerView.java,
//                  OnStartFragment.java, MainActivity.java,
//                  PieChartFragment.java, SimpleFragment.java,
//                  RemoteControlFragment.java, BarChartFragment.java
// Connections:     Wi-Fi
// Purpose:         Senior Project
// ----------------------------------------------------------------------

/*
 * Copyright 2013 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.smartpower.smartpowerstripapp;

import android.content.Context;
import android.widget.TextView;

import com.github.mikephil.charting.components.MarkerView;
import com.github.mikephil.charting.data.CandleEntry;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.utils.Utils;

/**
 * Custom implementation of the MarkerView.
 *
 * @author Philipp Jahoda
 */
public class MyMarkerView extends MarkerView {

    private TextView tvContent;

    public MyMarkerView(Context context, int layoutResource) {
```

```
        super(context, layoutResource);

        tvContent = (TextView) findViewById(R.id.tvContent);
    }

    // callbacks everytime the MarkerView is redrawn, can be used to update
the
    // content (user-interface)
    @Override
    public void refreshContent(Entry e, int dataSetIndex) {

        if (e instanceof CandleEntry) {

            CandleEntry ce = (CandleEntry) e;

            tvContent.setText("" + Utils.formatNumber(ce.getHigh(), 0,
true));
        } else {

            tvContent.setText("" + Utils.formatNumber(e.getVal(), 0, true));
        }
    }

    @Override
    public int getXOffset() {
        // this will center the marker-view horizontally
        return -(getWidth() / 2);
    }

    @Override
    public int getYOffset() {
        // this will cause the marker-view to be above the selected value
        return -getHeight();
    }
}
```