

Power Analyzer

Ryan Rabello and Nathan Zimmerly

May 26, 2018

Advisor(s):

Dr. Rob Frohne

# Abstract

The goal of this project was to provide insight into power consumption in order to significantly decrease its usage. Our system is able to show its users the areas that are using the most electricity. This can be used to identify and prioritize upgrades to high consumption areas. Also, making individuals aware of this data will not only save the environment but also their budgets.Our project aims to be a tool that will help individuals find inefficiencies in their consumption habits.

Table of Contents

[1 Abstract 2](#_Toc516491168)

[1 Acknowledgments 7](#_Toc516491169)

[2 Introduction and Objectives 8](#_Toc516491170)

[2.1 Introduction 8](#_Toc516491171)

[2.2 Project Objectives and Requirements 8](#_Toc516491172)

[2.2.1 Project Objectives 8](#_Toc516491173)

[2.2.2 Project Requirements 8](#_Toc516491174)

[2.2.3 Future Work 8](#_Toc516491175)

[3 Instructions 8](#_Toc516491176)

[3.1 Hardware 8](#_Toc516491177)

[3.1.1 Installation 8](#_Toc516491178)

[3.1.2 Wemos Software 10](#_Toc516491179)

[3.2 Software 10](#_Toc516491180)

[3.2.1 Requirements 11](#_Toc516491181)

[3.2.2 Installation 11](#_Toc516491182)

[4 Theory and Design Process 12](#_Toc516491183)

[4.1 Hardware 12](#_Toc516491184)

[4.1.1 Introduction 12](#_Toc516491185)

[4.1.2 Voltage and Current Measurement 13](#_Toc516491186)

[4.1.3 Assumptions 17](#_Toc516491187)

[4.2 Software 18](#_Toc516491188)

[4.2.2 Translating received data (Data Calculations) 19](#_Toc516491189)

[4.2.3 Archiving Strategy 22](#_Toc516491190)

[4.2.4 Running in production 22](#_Toc516491191)

[5 Design 23](#_Toc516491192)

[5.1 Hardware 23](#_Toc516491193)

[5.1.1 General 23](#_Toc516491194)

[5.1.2 Embedded Board 23](#_Toc516491195)

[5.1.3 External Antenna 23](#_Toc516491196)

[5.1.4 Measurement PCB (Printed Circuit Board) 24](#_Toc516491197)

[5.2 Software 25](#_Toc516491198)

[5.2.1 API Design 25](#_Toc516491199)

[5.2.2 Backend 26](#_Toc516491200)

[5.2.3 Frontend 27](#_Toc516491201)

[5.2.4 Hosting 28](#_Toc516491202)

[6 Hardware Cost 28](#_Toc516491203)

[7 Safety 29](#_Toc516491204)

[7.1 Hardware 29](#_Toc516491205)

[7.2 Software 29](#_Toc516491206)

[8 Results and Conclusion 29](#_Toc516491207)

[8.1 Conclusion 29](#_Toc516491208)

[8.2 Software Results 30](#_Toc516491209)

[8.3 Hardware Results 30](#_Toc516491210)

[9 References 31](#_Toc516491211)

[9.1 Component Resources 31](#_Toc516491212)

[9.2 Software Documentation 32](#_Toc516491213)

[10 Appendices 33](#_Toc516491214)

[10.1 GitHub 33](#_Toc516491215)

[10.2 Calculations Notebook 33](#_Toc516491216)

[10.3 Schematic 33](#_Toc516491217)

[10.4 API Endpoints 34](#_Toc516491218)

[Figure 1 - Current Transformer (right) and Jack Socket (left) 9](#_Toc516491219)

[Figure 2 - PCB Labels (left) and Connector Terminal Block (right) 9](#_Toc516491220)

[Figure 3 - Wemos D1 Mini Pro 10](#_Toc516491221)

[Figure 4 - Wemos Wi-Fi Info 10](#_Toc516491222)

[Figure 5 - Hardware Block Diagram 12](#_Toc516491223)

[Figure 6 - Breaker Box Diagram 13](#_Toc516491224)

[Figure 7 - Voltage and Current In Phase 14](#_Toc516491225)

[Figure 8 – Current Lagging Voltage 14](#_Toc516491226)

[Figure 9 - Relationships Between Types of Power 15](#_Toc516491227)

[Figure 10 - Voltage Divider Circuit 15](#_Toc516491228)

[Figure 11 - Current Transformer to ADC Circuit 17](#_Toc516491229)

[Figure 12 - Block Diagram of Software Modules 18](#_Toc516491230)

[Figure 13 - Normalized Voltage Measurements 20](#_Toc516491231)

[Figure 14 - FFT Results 20](#_Toc516491232)

[Figure 15 - Normalized Current Voltages 21](#_Toc516491233)

[Figure 16 - Real Current Running Through the CT 21](#_Toc516491234)

[Figure 17 - FFT of Translated Current Measurements 22](#_Toc516491235)

[Figure 18 - Backend Block Diagram 26](#_Toc516491236)

[Figure 19 - Database Model Relationships 26](#_Toc516491237)

[Figure 20 - Frontend Page Navigation Map 27](#_Toc516491238)

[Table 1 - Cost Summary 28](#_Toc516491239)

[Table 2 - API Endpoints 34](#_Toc516491240)

# Acknowledgments

We would like to acknowledge the help and support of our Advisor Dr. Rob Frohne, the Capstone Director, Dr. Doug Logan, Dr. Montgomery Buell, and our friends and family. Also, God for giving us the strength to finish this project despite the crippling sense of senioritis.

# Introduction and Objectives

## Introduction

The United States alone consumed 1.4 trillion kilowatt-hours of electricity in 2016 according to the U.S. Energy Information Administration (Fickling). The majority of this electricity was not generated using clean sources. Knowing that this is a huge concern, we started asking people about their power consumption habits. We found a trend of people being significantly more watchful of the electricity they were consuming when they were responsible for paying their own electricity. We believe that this is not only due to people wanting to save money, but also them having a greater awareness of the electricity they are using. However, the power bill that people receive only gives the amount of electricity used, not how it was used. Our projects principal objective was to raise awareness of people’s power consumption through intuitive data visualization.

Although there are products currently for sale that do similar things to our device, they are all expensive enough to discourage people from purchasing them. Our goal was to make our device available to be made for less than any of the devices currently on the market.

## Project Objectives and Requirements

### Project Objectives

1. Lower power consumption through intuitive data visualization.

### Project Requirements

1. Build an embedded device for data collection.
2. Store data from the embedded device on a server.
3. Display consumption data online.
4. Email Notifications for various power usage events.
5. Have a total hardware cost under $150

### Future Work

1. Authentication
2. Electric bill prediction
3. In-depth data analysis
4. Export data feature
5. Hardware casing

# Instructions

## Hardware

### Installation

1. Clip the current transformers around each of the hot (black) wires coming out of the breakers that data is going to be collected from.
2. If there is room available, add another breaker for the power analyzer. Otherwise, create a junction with one of the breaker circuits.
3. Plug the current transformers into the power analyzer using the jack sockets. The current transformer and jack sockets are shown below in Figure 1.



Figure 1 - Current Transformer (right) and Jack Socket (left)

1. Plug the hot, neutral, and ground wires into the labeled connections on the connector terminal block (shown in Figure 2).   
   Note: the labels are on the printed circuit board (PCB).



Figure 2 - PCB Labels (left) and Connector Terminal Block (right)

1. To connect to the server, click the reset button on the Wemos D1 Mini Pro (shown in Figure 3).

Note: On the Wemos D1 Mini Pro there is a small resistor with a 0 on it at the top (pointed at with the red arrow in Figure 3). This needs to be moved to the yellow circle in Figure 3. Then, the external antenna must be plugged into the connection at the top right of the Wemos, with the antenna outside the breaker box.



Figure 3 - Wemos D1 Mini Pro

### Wemos Software

1. For the Wemos to connect to the local Wi-Fi network, one must download either Atom or Visual Studio Code with the PlatformIO extension and open the project. Under the “src” folder, there is a file called “main.cpp.” Open this file, and near the beginning, there will be a couple lines of code that look similar to that shown in Figure 4 below. One must replace “Your-Wi-Fi-Network-Name-Here” with ones Wi-Fi Network SSID (name) and “Your-Wi-Fi-Password-Here” with the network's password (leaving the quotation marks).



Figure 4 - Wemos Wi-Fi Info

## Software

Note: The software portion of this project is split up into two distinct sub-projects; a frontend (website) and a backend (server). The definitions of frontend and backend are discussed more in section [5.2.1.1](#Introduction). Because these projects are completely separate from each other their descriptions (requirements, safety features etc.) will mostly be separated into two distinct sections; frontend and backend. However, through careful design, the frontend and backend come together to form the web application that is the software section of this project.

All of the software written and used in the software portion of this project is free and open source. The frameworks/libraries used are well known and well maintained. This and a number of other project design choices contribute to the sustainability of this project. To learn more about the sustainability of the project go to section [6.2.1.1](#Sustainability).

### Requirements

#### Frontend

The frontend project is written in JavaScript(JS) and uses the following libraries to function well.

1. npm (NodeJS) – The Node Package Manager (NPM) finds and downloads software based as defined by a package.json file.
2. ReactJS – A library built and used by Facebook for rendering UIs on web pages.
3. Axios – A library for creating HTTP requests in JS.
4. ChartJS – A library for creating beautiful charts and other graphs to display data.

#### Backend

The backend is written in Python version 3 and uses Django, a framework for building full stack web applications, as a request handler and object relational mapper (ORM).

### Installation

First off, all of our code is open source on GitHub. You can see the link to the code in the appendix.

#### Frontend

To download and start the frontend project follow the following steps.

1. Install NPM by navigating to <https://www.npmjs.com/get-npm>.
2. Clone the frontend project by running the following command

git clone https://github.com/power-analyzer/frontend.git

1. Enter the frontend directory (cd ./frontend)
2. Run the following command to install the necessary libraries required for the project.

npm install

1. Run the following command to start the server running on your machine.

npm start

#### Backend

To download and start the backend project follow the following steps.

1. Install python3 from <https://www.python.org/downloads/>.
2. Clone the frontend project by running the following command

git clone https://github.com/power-analyzer/backend.git

1. Enter the backend directory (cd ./backend)
2. Run the following command to install the necessary libraries required for the project.

pip install -r requirements.txt

1. Run the following commands to setup the database and begin running the server. The makemigrations and migrate commands are used to initialize the database with the most current models defined by the ORM.

python manage.py makemigrations

python manage.py migrate

python manage.py runserver

Navigate to <http://localhost:4200> and to see the web application in action. Note: The live version of the web application is available at <http://35.165.91.189/admin>.

# Theory and Design Process

## Hardware

### Introduction

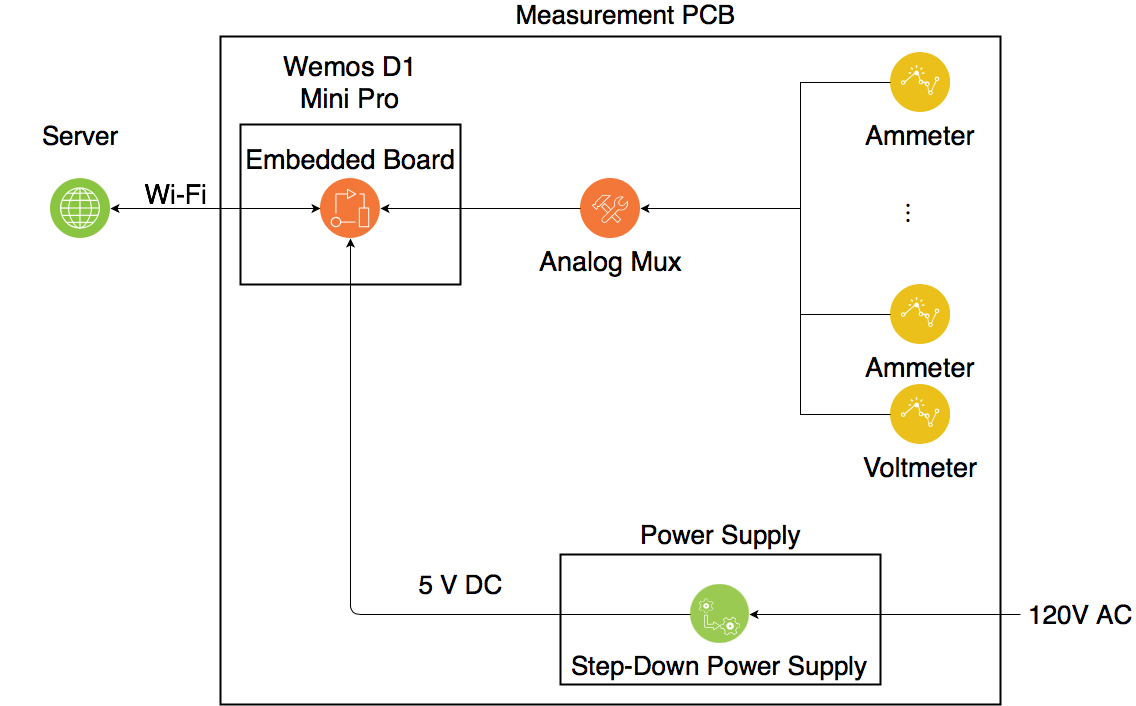


Figure 5 - Hardware Block Diagram

The hardware of the Power Analyzer has one purpose. Accurately collect the voltage across and current going through each of the circuits that need monitoring in the breaker box it is installed in. The voltage should be a relatively constant 120 V AC for the U.S., so the magnitude does not need to be measured, but it requires no more hardware to do so. However, some of the breakers will be out of phase with others. This is because for the three-phase power is coming into the breaker box, Phase A will be at -120V AC with Phase B, and phase C will be at 120V AC with phase B. The only difference between negative and positive AC voltage is that the phases are . Power companies attempt to evenly split the loads across phases A and B and C and B to minimize power loses due to heat. This causes the phase difference across some of the voltages. Most breaker boxes are set up so that the breakers on the left are connected to phases A and B, whereas the breakers on the right are connected to phases C and B. Figure 6 shows the basic wiring of a breaker box.

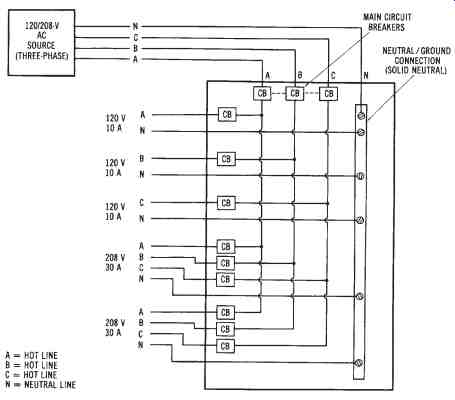


Figure 6 - Breaker Box Diagram

The current will be different through each breaker, so it must be measured at each circuit that needs to be monitored. To measure the current and voltage, two parts of each must be measured, the magnitude and the phase. The phase difference between the voltage and current determine the power factor which will be discussed in more detail in Section 4.1.2.4. Then, once the data is collected and stored, it must be transferred to the server over Wi-Fi.

The following Sections are broken down into how the hardware was designed.

### Voltage and Current Measurement

#### Theory

Power is calculated: , where is power, is current, is voltage, and is the phase difference between I and V. In AC (Alternating Current) electricity, the voltage and current are sine waves fluctuating between positive and negative values. With a purely resistive load, the current is in phase with the voltage as shown in Figure 7. However, the voltage and current may shift to a more positive or negative-phase independent of each other. Capacitance causes the current to lead the voltage and inductance causes the current to lag the voltage as shown in Figure 8. This causes the power to sometimes be negative (when the voltage and current have opposite signs), which causes power to flow backwards through the circuit. This causes more power being wasted as heat in the system.

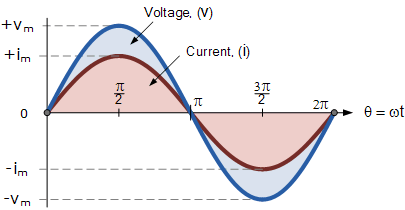


Figure 7 - Voltage and Current In Phase

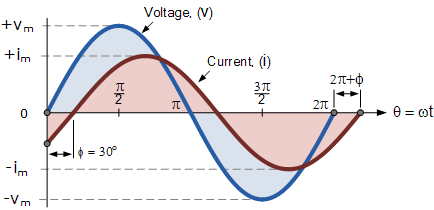


Figure 8 – Current Lagging Voltage

The way the phase difference between voltage and current is normally expressed is through the p.f. (power factor). It is calculated as: , where is the angle between apparent power S and real power W (shown in Figure 9) or the phase difference between the voltage and current waves. The goal is to have a p.f. = 1 or . With lower power factors (usually caused by inductive loads such as motors), power companies will charge extra on the power being used because they have more losses due to heat.

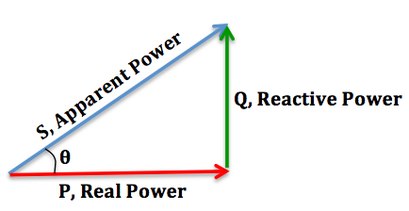


Figure 9 - Relationships Between Types of Power

#### Voltage Magnitude Measurement

Voltage measurement is relatively simple, because as long as the voltage is between zero and three volts, the microcontroller (Wemos D1 Mini Pro) can measure it. So, the voltage running through the breaker box simply needs to be shrunk in magnitude from 120V to 1.5V, and then offset by 1.5 volts (to make the sine wave range from 0V to 3V and eliminate the negative voltage). The magnitude can be shrunk easily using a voltage divider circuit as shown in Figure 10.

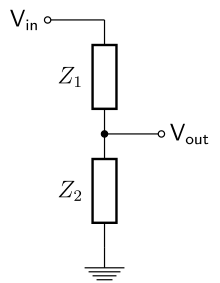


Figure 10 - Voltage Divider Circuit

In Figure 10, Z1 and Z2 are resistors. , and by adjusting the resistors to the correct ratio, Vout can be made significantly smaller compared to the original voltage, and then just to calculate the original voltage, simply multiply Vout by the inverse of that ratio , to get back to the original voltage.

The DC offset of 1.5V is a little more difficult. Inductors allow DC (direct current) to flow through, but not AC. Capacitors, on the other hand, allow AC to flow through them, but not DC. So, if the AC waveform is run through a capacitor, with the DC offset running through an inductor, before they are wired together, the AC and DC components are forced to combine, giving the original AC wave with a DC offset. However, through simulation (and testing), it was found that using a large resistor was equivalent to using an inductor. This was perfect because inductors are relatively expensive, and they take up a much larger space than resistors. This can be seen under the “Power from Breaker Box” section of the schematic in Section 10.2.

To measure the actual magnitude of the voltage, the sine wave must be measured at its peak. Fortunately, the power in the U.S.A has a constant 60Hz frequency. This allows me to measure the wave for exactly one period (1/60Hz) and then after converting that back from my analog-to-digital converter values to the values that are actually being measured, a Fourier transform can be taken to find the magnitude and phase of the voltage. The Fourier transform also picks out the 60Hz information which filters out any noise. Because of the capacitor in the circuit, there will be a phase delay between the actual voltage and what is measured at the ADC. However, using circuit analysis, the voltage measured at the ADC (without the offset) can be divided by 0.821063e-2+j0.116435e-3 and the value will be converted back to the original waveform magnitude and phase.

Once the voltage measurements are captured, they are sent to the server for calculations. The calculation process is discussed in [section 4.2.2](#_Translating_received_data).

#### Current Magnitude Measurement

The current cannot be measured directly by the microcontroller. Instead, a current transformer along with a burden resistor (such as R7 on the schematic in section 10.2) must be used to create a voltage that is equivalent to the current running through the wire. Once the voltage measured (using the method described in section 4.1.2.2), the original current can be recalculated. The circuit from the current transformer to the ADC is shown below:

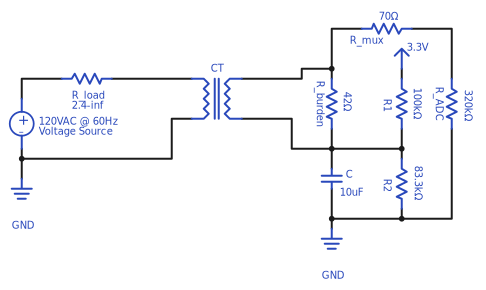


Figure 11 - Current Transformer to ADC Circuit

Using circuit analysis, the following formula was found giving the current being measured through the CT, where “Vadc” is the voltage measured at the ADC (not the actual ADC value) without the DC offset, and “N” is the windings ration on the current transformer (CT).



Note: all calculations are done on the server to allow more sampling time for the embedded board.

#### Power Factor Measurement

Once the Fourier transforms of the voltage and current have been calculated on the server, the 60Hz points are chosen and multiplied. This leaves a complex number for power where the real power is the real part of the number and imaginary part of the number is the reactive power. The cosine of the ratio of the apparent power and the real power results in the power factor.

### Assumptions

A few assumptions were made during the hardware design process. They are written below:

1. The breaker boxes this device is installed in have room to place the device (inside the casing).
2. The breaker boxes this device is designed to have at most 15 circuits needing to be measured, or room for multiple devices.
3. The household this device is used in has a Wi-Fi connection.
4. The breakers on the left side of the breaker box are out of phase with the breakers on the right (explained in Section 4.1.1).

## Software

#### Introduction

Web applications are applications that run on the web. They are different from normal applications in that they don’t run locally on a single computer. There are different technologies and design methodologies behind web apps. Each technology offers advantages and disadvantages. The technologies chosen allow for the separation of a frontend, which is code that will run on the client (in a web browser), and the backend, which is code that runs on one server. The two blocks are connected using an Application Programming Interface (API). In particular, we opted to use a JSON API.



Figure 12 - Block Diagram of Software Modules

#### API – Application Programming Interface

The is the way that the frontend and backend communicate. The two modules communicate by using HTTP (Hyper Text Transfer Protocol) requests. These requests are very common when browsing the internet. Http requests are just text as such when sending data it is important to define a syntax or form with which that data is sent. Two common data syntaxes are XML(eXtensible markup language) and JSON(JavaScript Object Notation). JSON was selected to be the API for this project because of its native support in JavaScript.

There are also different types of HTTP requests. The most common two are GET and POST requests. GET requests are used for retrieving information and POST requests are used for sending information.

The API is designed such that there are explicit requests that manipulate or retrieve specific data.

#### Frontend

In react, different views, or JS modules that contain data and render elements on the document object model (DOM), can be combined to create a fully functional user interface. An example of a typical request process in React would be as follows. 1) The user clicks a button. 2)An event, that initiates a request for data from the server, is fired. 3) The server responds 4) JS makes the data available to React. 5) React renders data into the DOM.

Using React, JS, and HTTP requests, a UI that displays power consumption data can now be created. See the design section for more details.

#### Backend

The backend portion of this project has two main functions. Handling HTTP requests and manipulating data. In this case, Django is used for both the handling of requests and also manipulating data.

To handle requests the server matches the URL against a list of defined paths. If the current path doesn’t match any in the list a 404 (page not found) error is thrown. However, if the URL does match a route then the route handler associated with that route is invoked or called. An example of a route handler pertaining to this project would be the /buildings route handler. This route handler returns a list of all the buildings that are stored in the database. To get the required information to fulfill the request, the handler must query the database.

For this project, data in a database is organized using Django’s Object Relational Mapper (ORM). Database models are created to match the requirements of the project. This project implements four main models; a building, device, circuit, and measurement. Using these models in a hierarchical approach consumption data can be organized to facilitate the retrieval, storage or manipulation of data.

### Translating received data (Data Calculations)

The power monitoring device lacks the computational power required to translate some of the measured data at a rate that is sufficient for measurement. This means that the server must run some calculations on the data before it can be stored. The most notable is that for each circuit the device is sending an array of voltages it has measured. It is the server’s responsibility to decode the complex voltage or current from these arrays. A fast Fourier transform (FFT) is used for this process.

The process for finding the power is as follows.

1. First, the server receives two arrays, voltage, and current. These arrays are filled with the raw data from the Analog to Digital Convert (ADC). The data, which is a sine wave, needs to be scaled to match the real voltages and currents measured.
2. The array of electrical current values will need to be translated an additional time to covert the measurements from voltage values to current values.
3. Once the voltage and current values are correct throughout time. These sine waves need to be converted to their phasor form. To accomplish this an FFT is performed.
4. Now that the currents and voltages are in phasor form they can be multiplied together to form the complex power
5. From the complex power, the magnitude of the power can be obtained. This is the number that the power companies charge for, so it is stored. The power factor can also be calculated from the complex power it is stored as well. Lastly, the current and voltage phasors are stored so that there is no loss of information.

#### Voltage

The following formulas convert the ADC values into appropriate voltages.

At this point the array looks something like the following.

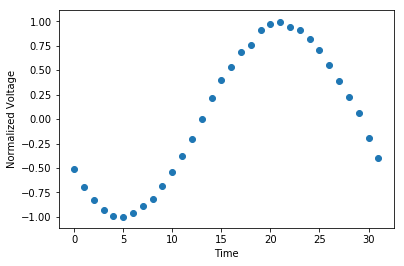


Figure 13 - Normalized Voltage Measurements

The information desired from this graph is the phasor form of the graph. To get this we perform an FFT on the data. The obtained graph is shown below.

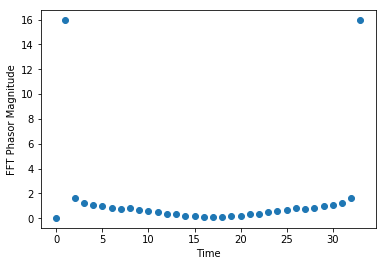


Figure 14 - FFT Results

The results of this FFT are then translated using the following formulas. The phasor representing 60Hz is the second and last element of the array.

#### Current

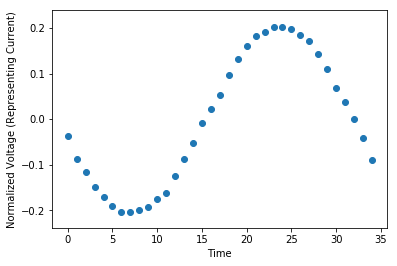


Figure 15 - Normalized Current Voltages

The same process is used for converting ADC digital values to their real voltages. However, before the FFT is calculated, the voltages must be converted to current values. To do this the following equation was derived by Zimmerly using a Matlab and Maple script.

The graph representing the magnitude of the scaled complex current is shown below.

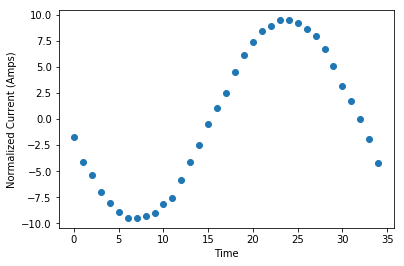


Figure 16 - Real Current Running Through the CT

Now the FFT of this array can be taken the result of which is shown below.

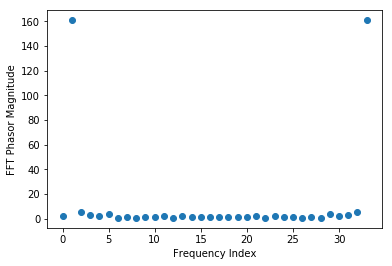


Figure 17 - FFT of Translated Current Measurements

The second value or last value is retrieved scaled and then multiplied with the voltage phasor to obtain the complex power.

To see exactly how this calculation is performed in python see the [calculations section](#_Calculations_Notebook) in the appendix.

#### Error Reduction

Because this process uses about 68 voltage and current measurements to calculate the power, an error in one measurement or point would be averaged out during the FFT. This is desirable in building a robust error free system and contributes to the project’s accuracy.

### Archiving Strategy

Because data will be constantly uploaded to the server an archiving scheme is required to increase performance and decrease the size of the database. An archiving strategy that doesn’t reduce the accuracy of the power consumption data was required. The strategy works as follows.

There are two models for measurements; unarchived and archived. When a new measurement is submitted by the monitoring device the server will compare the current time with the oldest unarchived measurement. If that measurement is older than 30 minutes the server will aggregate those measurements into one measurement that represents the power consumption for the entire 30-minute interval.

### Running in production

There is a specific way in which the server should be run in production. This way is to use WSGI which is a Web Server Gateway Interface built for python. This interface allows Django to send and receive HTTP requests on any platform that runs WSGI and python.

There are a couple of other things to consider when running the server in production. Http requests sent by the server need to have a couple of headers present so that web browsers will be able to send and receive data to them. The most notable of which is the Access-Control-Allow-Credentials header which needs to be set to \*. Django has been configured to do this.

Another important facet of running in production pertains to the frontend. Because React is used for this project there is a build process for the frontend app that optimizes the code. When an application is running in production there is no need for it to have complicated error checking or a quick compile time. React gathers all required (ie. Removes debugging) JS and puts it into one file. React then compresses that file so that load times will be minimized.

# Design

## Hardware

### General

Because of the repetition from measuring up to 15 breaker circuits, the voltage and current measurement parts are all wired into a multiplexer. From there, the multiplexer output is connected directly to the analog-to-digital converter (ADC) on the embedded device.

### Embedded Board

The embedded board that was chosen for this project is the Wemos D1 Mini Pro. The main reason this board was chosen is because it has Wi-Fi that is set up to easily connect an external antenna to the board. This is important because if the device is placed inside the breaker box, it will block the Wi-Fi signal because it is a conductor. Because of this, the antenna has to be outside the breaker box in order to connect to the buildings Wi-Fi network. Other reasons include that the device is cheap ($5.00), it has enough pins for the project, it is small, and it has an ADC for measuring the magnitude of the voltage and currents.

#### Embedded Board Software

The embedded board has the following basic algorithm:

Initialization:

1. Test each input for a connection.
2. Add connected circuits to testing list.

Loop:

1. Sample 35 points in one period of the voltage.
2. Wait exactly 100ms (6 periods at 60Hz) from the start of the voltage measurement, and then repeat for each breaker circuit.
3. Send data to the server.

Note: Calculations for converting the ADC value into magnitude and PF are calculated on the server to increase the embedded board's efficiency.

### External Antenna

The external antenna for the embedded board simply needed to be able to be placed outside the breaker box because the steel box attenuates the Wi-Fi signal significantly. The Diymall 2.4G WiFi Antenna with SMA Cable 3DBI Gain Antennas U.FL to for Arduino CC3000 ESP8266 was chosen because of its price.

### Measurement PCB (Printed Circuit Board)

The measurement PCB measures the current (through each breaker circuit) and voltage, then sends the data in a readable way to the embedded board.

#### Power

The PCB includes a connector terminal block for connecting 14-26 AWG (American Wire Gauge) wire from a breaker to the board. This terminal block connects the neutral and ground connections together so that nobody will be electrocuted if the hot wire is accidentally switched with either neutral or ground. Then, the power is run through a Bel Fuse Inc. SMP 1.251.5A fuse. After that, it is run through an AC-DC Power Module 5V 700mA V1 power supply which powers the embedded board.

#### Current Transformers (CTs)

The current transformer that was chosen was the LGDehome 100A/50mA SCT-013-000 Non-invasive AC Current Sensor Split Core AC Current Sensor Transformer. This current transfer was selected because it clips around the wire it is “measuring,” making it so that the wire does not need to be removed from the breaker for installation. Also, the 3.5mm jack allows it to be easily connected with the PCB. This specific current transformer was the cheapest current transformer that would clip around the wires that would work. Cheaper current transformers can be soldered to the board where the 3.5mm Jack connectors would go, but they did not have the clip for ease of installation.

#### Analog Multiplexer

The analog multiplexer used in this project was the Everlight Electronics Co Ltd H11L1S(TA). This was chosen because it was the cheapest analog multiplexer that would work for voltages between 0 and 3.3 volts. Also, it only has a 70Ω input resistance.

#### Resistors, Capacitors, and Accuracy

To receive accurate results within 3% for the current and 2% for the voltage, 1% resistors and 25% capacitors must be used. Changing the resistors to 5% results in 7% current and 10% voltage accuracy. Finally. The most important resistor to be accurate for the current is R7 for I0 and the equivalent resistor for each of the other ammeter circuits in revision 2 of the PCB. Also, note that 2% of the ammeter error comes from the current transformer (Wall), while the rest is from the PCB. For the voltmeter circuitry: R1, R2, R3, and R4 are significantly more important to have accurate for the best results compared to the other resistors. The calculations for these percentages are shown on the Github page. For more accurate results, measure the resistors and capacitors oneself, and then plug them into the Thevenin voltage and current circuit equations and use those values and equations on the server.

## Software

### API Design

The development cycle for full stack web apps works from the API out. The API is a crucial step to designing any full stack project. It is responsible for communicating between the two portions of the app. It must be carefully designed because if one changes the API by manipulating one part of the project, the other part of the project will have to change. It is tedious to have to continually update two separate projects. As such, it is advantageous to define the API first, by explicitly writing out the paths of data transfer. This is how this project was designed.

It was evident that information needed to be passed to the backend from the device, information needed to be requested from the backend, and information needed to be delivered to the frontend to satisfy those requests. There were clearly three main categories, data addition, model exploration, and data retrieval.

#### Data addition

To add data to the server Zimmerly’s device sends a POST request to the /datapoints/<mac>/batch URL on the server. The server receives this request translates and stores the data then acknowledges the request with an OK (Status 200), indicating that everything worked out well.

#### Model Exploration

For scalability and performance reasons, it is not a good idea to download all stored data when a user might only want to see information about one circuit. As such the API is designed to expose buildings, then devices then circuits. For example, to get all the data from a single circuit, one must ask for the list of devices, then ask for the list of circuits attached to a particular device. Once the ID of the circuit is known, the data for that circuit can be obtained directly.

While this process does sound like it has extraneous steps in it. It allows the data to be traversed in an efficient tree-like manner.

#### Data Retrieval

As mentioned before, if the ID of the circuit is known, the measurements for that circuit can be downloaded at the following server URL.

/circuits/<int:id>/<str:end\_date>/<str:start\_date>

This specific route handler delivers all the archived measurements for the circuit id from end\_date to start\_date. For a detailed list of the all the API endpoints, take a look at the [API Endpoints Section](#_API_Endpoints).

### Backend



Figure 18 - Backend Block Diagram

In general, the backend is designed to route URLs to specific route handlers. These route handlers are designed to do any number of things including saving or retrieving data from the database, using the mapping of the Object Relational Mapper(ORM). Route handlers also have access to the email module which can send emails.

#### Models

Data is organized into three main groups; devices, circuits, and measurements. These database models are defined using Django’s ORM which offers the functionality to link models to each other. The linking method used in for this project was a many-to-one link or relationship. This means that many models have a field that links to one other model. This many-to-one relationship is used to link measurements to circuits and circuits to devices.

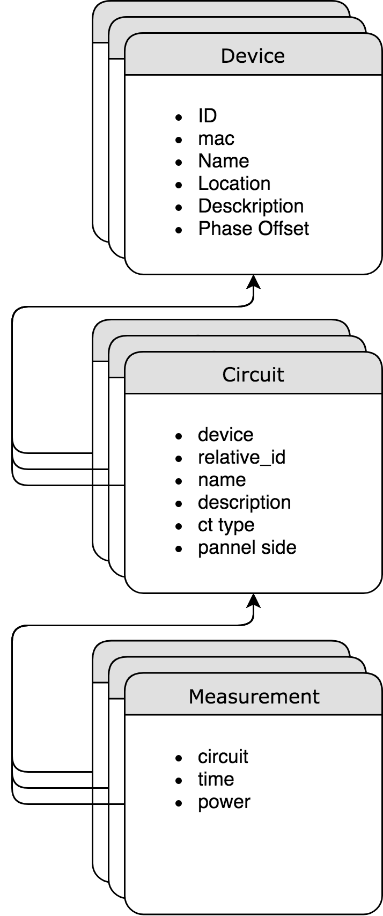


Figure 19 - Database Model Relationships

The resulting hierarchical structure organizes the data in an intuitive way that facilitates scalability and performance. Organizing the models in this way allows for all the models to be efficiently retrieved.

### Frontend



The frontend web app utilizes ReactJS aka React. React allows UI elements to be rendered into HTML and CSS. React does this in the following way. Information is made available through some module, in this case, the request service. React elements, in this block diagram called Page Render Components, take this information and manipulate it, as desired by the programmer, and send it to react for rendering. React takes the information and renders it as HTML and CSS.

#### Page organization

The pages on the frontend are organized according to the below block diagram. In this block diagram pages that were implemented for this project are darker gray and pages that fall under the category of future work are greyed out.

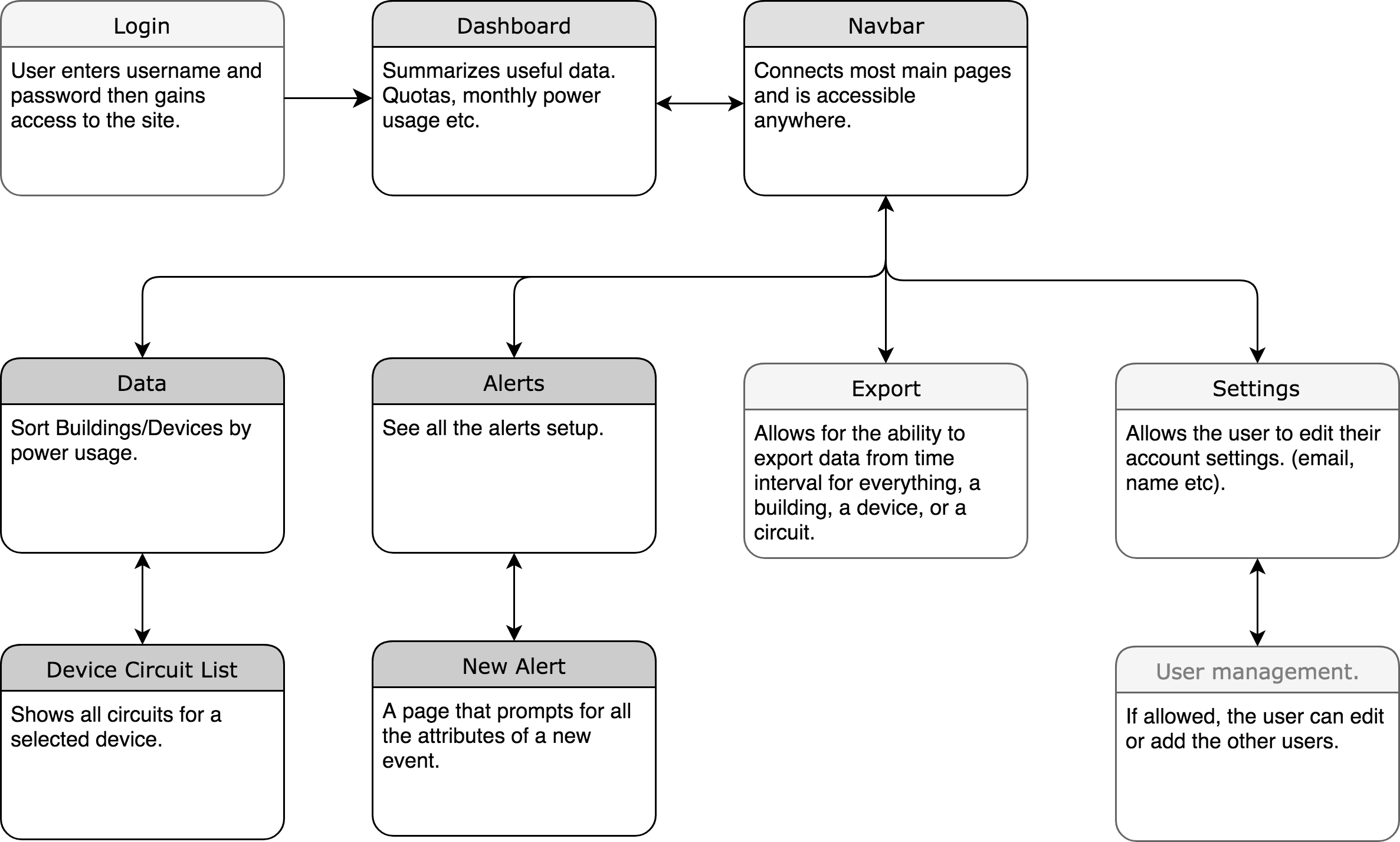


Figure 20 - Frontend Page Navigation Map

### Hosting

The project is hosted on Amazon Web Services (AWS) in a virtual machine. The server is running Ubuntu, however, any close Linux derivative should be easy to configure in the same way. The server handles requests for the backend and delivers the frontend to clients.

The frontend is available at the root URL of the server. This means that when the server IP or domain name is typed into the web browser the complied react app is delivered.

The react app communicates with the live Django server which is available on port 8000 of the server. Nginx and Apache (tools for routing network traffic) can be used to proxy pass information from the /server URL to the live Django server as well.

#### Emails

Django provides an easy to use wrapping of a common python SMPT (Simple Mail Transfer Protocol) package. For my project I used a specific service to create a free SMTP server, however, any SMTP server can be configured by editing the /power-analyzer/settings.py file.

#### Software Sustainability

The software is built on well-maintained and open source frameworks and libraries. The two largest parts of the project React and Django will be around for a long time for the following reasons. React is built and used in production by the engineers at Facebook. Django an actively contributed and popular framework.

The other libraries used in this project aren’t backed by large companies or aren’t the size of Django. However, they do have a good-sized community of developers contributing to and maintaining the projects.

# Hardware Cost

|  |  |
| --- | --- |
| Component | Price |
| Wemos D1 Mini Pro | $5.00 |
| External Antenna | $4.00 |
| Power Supply | $0.72 |
| PCB | $3.60 per PCB (if five are bought) |
| Analog Multiplexer | $0.80 |
| Connector Terminal Block | $0.52 |
| Fuse | $.79 |
| 3.5mm Jack Socket | $0.18 per socket |
| Current Transformer | $6.85 |
| Resistors and Capacitors | $1.00 (estimate) |
| Case | $5.00 (estimate) |
| Shipping and Taxes | $15.00 (estimate) |
| Total: | $45.98-$141.88 |

Table - Cost Summary

# Safety

## Hardware

Because of the high voltage and currents running through a breaker box, a few safeguards have been implemented into the hardware. First, the connector terminal block which connects 120V AC power from the breaker box using 14 AWG wire has been wired so that if the hot wire is connected in the wrong spot, it will connect straight to ground. This keeps the user from being electrocuted from mistakenly wiring the board wrong. Second, a 1.5 A fuse is integrated into the hardware so that if the board gets a short it will burn the fuse instead of lighting on fire or melting.

Also, to be legally allowed to be placed in the breaker box, all components working with 120V AC root mean square (RMS) have to support up to 600V AC RMS. This has been completed by using a fuse that supports 600V and using three 0805 resistors in series which individually support 212V which totals to 636V supported across the three resistors.

Note: It is advised that a separate breaker is installed in the breaker box to power the Power Analyzer. If this is not possible, a wire junction is recommended to be installed in the breaker box.

## Software

The software portion of this project will be physically isolated from the hardware. The only form of contact is through Wi-Fi which does not pose a safety threat to the hardware. Furthermore, the software section of this project is only a monitoring service. That means that the software is only listing to the device and sending acknowledgments back. It is not controlling the hardware in any way. If the service is compromised, the worse it could do is not acknowledge the devices requests.

The form of safety that the software section does consider is the safety of information or privacy. Power consumption data might not seem like data that needs to be protected, however, in an age where information is power, it is best to keep things secure. For web apps, typically, safety features are only implemented on the backend portion of the project. This is because frontend code is manipulatable and viewable by the client which means that virtually no security can be implemented in front-end code. For this project, the use case is a do it yourself person who runs the project their local network.

# Results and Conclusion

## Conclusion

The core project goals were met. The hardware successfully measures the current and voltage of up to 15 breaker boxes per PCB and sends the data to the server. The server successfully calculates the data it receives into useful information which it compresses and stores. The data is shown in several different ways on the front-end of the software. There is still much work that can be invested in improving this project, but the product designed is usable for its given purpose.

## Software Results

At this point, the software has been designed in a modular way. This means that the project will easily be able to adapt to any future changes required by design changes. The critical path of the software is complete. The backend can receive data from the device, run calculations on that data, store the data, and the frontend can request and display the data. Additionally, the software portion of this project implements email notifications.

## Hardware Results

The hardware is successfully reading the correct waveforms and transmitting the data. From calculations, using 1% accurate resistors and 25% accurate capacitors, the error was within 3% for the ammeter. However, only 1% of this was due to the PCB, the other 2% was due to the current transformer. The voltmeter measured within 2% accuracy for the voltage. Although these results can be improved, they are satisfactory. Testing was done with a combination 1%, 5%, and 10% accurate resistors and capacitors and error ranged to about 8% for both the current and voltage. However, this was on the first revision of the PCB where components that were not supposed to be added to the PCB were added manually which increases error in the signal. Still, the results are about what the error margins were calculated for the 5% accurate resistors. This gives confidence in the ability of the hardware to produce results within the calculated error using 1% resistors (3% current and 2% voltage). Future work for the hardware includes creating a more accurate device that can withstand larger variations in resistors, making an insulated casing for the PCB, and connecting the microcontroller through the servers front-end.

# References

Fickling, Meera. “Per Capita Residential Electricity Sales in the U.S. Have Fallen since 2010.” U.S. Energy Information Administration, 26 July 2017, [www.eia.gov/todayinenergy/detail.php?id=32212](http://www.eia.gov/todayinenergy/detail.php?id=32212).

Wall, Robert. “YHDC SCT-013-000 Current Transformer.” Learn | OpenEnergyMonitor, [http://www.learn.openenergymonitor.org/electricity-monitoring/ct-sensors/yhdc-sct-013-000-ct-sensor-report](http://learn.openenergymonitor.org/electricity-monitoring/ct-sensors/yhdc-sct-013-000-ct-sensor-report.).

Figure 6: <http://thebearden.co/wiring/200-electrical-distribution-wiring-diagram.html>

Figures 7 & 8: <https://www.electronics-tutorials.ws/accircuits/phase-difference.html>

Figure 9: <https://en.wikipedia.org/wiki/Power_factor#/media/File:Power_triangle_diagram.jpg>

## Component Resources

Wemos D1 Mini Pro: <https://wiki.wemos.cc/products:d1:d1_mini_pro>

External Antenna: <https://www.amazon.com/Diymall-Antenna-Antennas-Arduino-ESP8266/dp/B00ZBJNO9O/ref=sr_1_1?s=electronics&ie=UTF8&qid=1526526728&sr=1-1&keywords=Diymall+2.4G+WiFi+Antenna+with+SMA+Cable+3DBI+Gain+Antennas+U.FL+to+for+Arduino+CC3000+ESP8266&dpID=41WwCILclSL&preST=_SY300_QL70_&dpSrc=srch>

Power Supply: <https://www.itead.cc/wiki/AC-DC_Power_Module_5V_700mA>

Opto-Isolator Datasheet: <https://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/6N137%20Series.pdf>

Multiplexer Datasheet: <http://www.everlight.com/file/ProductFile/H11L1.pdf>

Connector Terminal Block Datasheet: <https://media.digikey.com/pdf/Data%20Sheets/Phoenix%20Contact%20PDFs/1792876.pdf>

Fuse: https://www.mouser.com/datasheet/2/643/ds-cp-smp-series-1313136.pdf

Jack Socket: <https://www.amazon.com/gp/product/B015FIUEOI/ref=oh_aui_detailpage_o02_s00?ie=UTF8&psc=1>

Current transformer: <https://www.amazon.com/LGDehome-SCT-013-000-Non-invasive-Current-Transformer/dp/B075541WVT/ref=sr_1_3?ie=UTF8&qid=1526508049&sr=8-3&keywords=current+transformer&dpID=51aSpsHYdQL&preST=_SX342_QL70_&dpSrc=srch>

## Software Documentation

Python Documentation: <https://docs.python.org/3/index.html>

Django Documentation: <https://docs.djangoproject.com/en/2.0/>

JavaScript Documentation: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>

React Documentation: <https://reactjs.org/docs/react-api.html>

Chart JS Documentation: <http://www.chartjs.org/docs/latest/>

Axios Documentation: <https://github.com/axios/axios>

Bootstrap Documentation: <https://getbootstrap.com/docs/4.0/getting-started/introduction/>

Reactstrap Documentation: <https://reactstrap.github.io/>

# Appendices

## GitHub

All the software is located on Github and can be found by using the following links.

Hardware: <https://github.com/power-analyzer/hardware/>

Frontend: <https://github.com/power-analyzer/frontend>

Backend: <https://github.com/power-analyzer/backend>

## Calculations Notebook

A PDF of the python notebook is located at <https://github.com/power-analyzer/backend/raw/master/Voltage%20and%20Current%20Calculations.pdf>

## Schematic



## API Endpoints

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| URL | Method | Description | Request body (IF POST) | Response body |
| /datapoints/register/<str:mac> | GET | This will create a new Device object with the given mac address if it doesn’t already exist. | N/A | {"mac": <device.mac>} |
| /datapoints/<str:mac>/batch/ | POST | This will upload one measurement to a number of circuits. The server will save “power”, “voltage”, “current” and “phase” if they are provided. | {  "v":[datapoints],  "1":[datapoints],  ...  "15":[datapoints]  } | {"status":"success"} |
| /datapoints/(buildings|devices|circuits)/<int:id>/<str:start\_date> | GET | Get the (possibly aggregated) datapoints for the entity from start date to now. |  |  |
| /datapoints/(buildings|devices|circuits)/<int:id>/<str:start\_date>/<str:end\_date> | GET | Get (possibly aggregated) datapoints in between start\_date and end\_date |  |  |
| /datapoints/(buildings|devices|circuits)/ | GET | Get a minified list of all the entities. |  |  |
| /datapoints/(buildings|devices|circuits)/<int:id> | GET | Get everything about a particular entity. |  |  |

Table - API Endpoints