Santa Clara University
Scholar Commons

Computer Engineering Senior Theses

Engineering Senior Theses

6-11-2019

Synergy: An Energy Monitoring and Visualization System

Sarah Johnson

Pearce Ropion

Follow this and additional works at: https://scholarcommons.scu.edu/cseng_senior

Part of the Computer Engineering Commons

Recommended Citation

Johnson, Sarah and Ropion, Pearce, "Synergy: An Energy Monitoring and Visualization System" (2019). *Computer Engineering Senior Theses*. 150.
https://scholarcommons.scu.edu/cseng_senior/150

This Thesis is brought to you for free and open access by the Engineering Senior Theses at Scholar Commons. It has been accepted for inclusion in Computer Engineering Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Date: June 11, 2019

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Sarah Johnson
Pearce Ropion

ENTITLED

Synergy: An Energy Monitoring and Visualization System

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREES OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING
BACHELOR OF SCIENCE IN WEB DESIGN AND ENGINEERING

B. Dezfouli
Thesis Advisor

[illegible]
Department Chair

Synergy: An Energy Monitoring and Visualization System

by

Sarah Johnson
Pearce Ropion

Submitted in partial fulfillment of the requirements
for the degrees of
Bachelor of Science in Computer Science and Engineering
Bachelor of Science in Web Design and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 11, 2019

Synergy: An Energy Monitoring and Visualization System

Sarah Johnson
Pearce Ropion

Department of Computer Engineering
Santa Clara University
June 11, 2019

ABSTRACT

The key to becoming a more sustainable society is first learning to take responsibility for the role we play in energy consumption. Real-time energy usage gives energy consumers a sense of responsibility over what they can do to accomplish a much larger goal for the planet, and practically speaking, what they can do to lower the cost to their wallets. *Synergy* is an energy monitoring and visualization system that enables users to gather information about the energy consumption in a building – small or large – and display that data for the user in real-time. The gathered energy usage data is processed on the edge before being stored in the cloud. The two main benefits of edge processing are issuing electricity hazard warnings immediately and preserving user privacy. In addition to being a scalable solution that intended for use in individual households, commercial offices and city power grids, *Synergy* is open-source so that it can be implemented more widely. This paper contains a system overview as well as initial finding based on the data collected by *Synergy* before assessing the impact the system can have on society.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The world runs on electricity. Whether it be lighting homes, pumping water, powering factories, or even, in some cases, opening a door. However, energy usage is difficult to gauge. Energy bills often get paid without knowledge of how much energy was used and where. For example, many large complexes leave their lights and devices on throughout the night. On-demand electricity is expensive; both to our wallets and our planet. On average, American households spend $112 per month on their electricity bill[5]. Often times, electrical bills can spike due to devices that have been left powered on or malfunctioned, and can go undetected for years. Evidence has shown that monitoring domestic electricity usage plays a large role in reducing consumption [33]. Energy monitoring solutions allow people to be conscientious about how much energy they are using. These solutions are essential in keeping energy costs down for the consumer by revealing noninvasive lifestyle changes to lower energy consumption, such as making a habit of turning off lights or devices when not in use [33]. However, there are a multitude of challenges in implementing universal energy monitoring.

The challenges of energy monitoring stems from a problem with energy management. While monitoring specifies the process of keeping track of the energy usage, management refers to the coordinated efforts to habituate a reduction in energy consumption [40]. At any scale, energy management requires a coordinated effort between all actors. Actors could include the residents of a single household, or the employees within an office building. There must be consensus among the actors to realize lowered energy usage; be that an understanding that employees must turn off their monitors and devices at the end of the workday, or that all lights must be turned off in the home before leaving. Without proper energy management, energy monitoring would have much less meaning. At the same time, without energy monitoring, it would be much more difficult to assess how reductions in energy consumption could be realized.

1.2 Background

Monitoring energy usage is not a new concept. Electricity usage is consistently monitored by power companies in order to charge consumers for the energy that they use. However, there has been very little effort to deliver these monitoring capabilities to the individual. Instead, consumers are charged a lump sum without any distinction as to where that energy went or knowledge of how to reduce its usage. Placing energy management solutions in the hands of consumers enables them to actively institute measures for saving energy. The difficulty of this solution comes when one considers how these solutions could be enabled within households. Over 70% of the residential homes within the US were built prior to 2000 [45], meaning that they simply do not have the infrastructure to adequately equip energy monitoring and management devices without remodeling.

1.2.1 Electrical Panel Solutions

In order to accommodate some of the challenges that come with energy monitoring, many existing solutions have made their products compatible with other technologies in an effort to attract a larger audience. Sense [35] is a monitoring device that connects to a home's electrical panel. The device monitors electrical throughput in order to determine how much electricity is being used by the household at any given time. Sense also provides a cloud-based visualization service accessible via a web-based and smart phone application. Sense allows for precise detection of home activity including when electricity is being used and which device is using it. It can also notify users when it detects unnecessary energy usage from a device. However, precise detection that Sense provides introduces privacy and security concerns with regards to home safety. Data of every electrical action is monitored and immediately uploaded to Sense's cloud servers. If bad actors were to intercept this data stream it could give a general idea of when a user is home or not, leaving homes vulnerable [21; 23]. Constantly uploading data also means that the home's bandwidth is clogged with the hundreds of megabytes that Sense uploads to the cloud every day. Furthermore, the device itself is rated to a maximum of 200 amps meaning it can only be placed in the electrical panel of a residential home. Placing the device in a larger complex, such as an office building would cause it to overload, making the scalability of such a system impossible.

Neurio [13] is another system which is installed in the home's electrical panel. Similar to Sense, it provides cloud-based data visualization that requires a constant connection to the internet. Neurio can also be integrated with solar panels mounted on the home's roof and supports submetering of major appliances like electrical vehicles. However, these additional features fall victim to the same drawbacks as Sense; primarily that none of the home's private usage data is stored locally, and the system has not been designed to be used outside of a standard residential home.

1.2.2 Electrical Outlet Solutions

There are also some alternative energy monitoring systems such as TP-Link's Smart Plugs [41]. These devices can be plugged into an existing power outlet on the wall or on a power strip. They can then monitor energy usage of a single appliance which can be plugged in the front side of the smart plug. Similar to both Sense and Neurio, TP-Link Smart Plugs stream their monitoring usage data to the cloud. Users can then access this visualization data through their smart phone and web applications. Furthermore, the plugs are also smart home enabled, meaning they can connect to existing smart home systems such as Amazon's Alexa and Google Home to be turned on and off remotely. Unfortunately, each smart plug can only monitor a single device at a time and costs \$20 each, which can become expensive given how many devices reside in the modern home. This presents issues when it comes to the scalability of the product.

Other existing solutions have attempted to solve problems with measuring energy usage in low power devices [27] and power fluctuations in networking devices [24]. Monitoring in a low power environment involves precise monitoring devices that are unsuitable for consumer scalability. These types of devices would be best suited for monitoring the energy usage of the monitoring devices themselves or other similar sensors. Similarly, networking devices such as load balancers, switches, routers and modems use a lot of energy on a daily basis and these devices are always on and always processing. Although simple monitoring on these devices would be helpful, the real use case comes when attempting to detect security vulnerabilities or cyber-attacks. A denial of service (DoS) attack spams a networking device with hundreds of thousands of packets in order to attempt to block traffic to the services provided. Generally, when a DoS starts, the device's processor picks up, increasing its power usage. Energy Monitoring on networking devices could help detect DoS attacks when they start, instead of finding them hours later when it is too late.

1.3 Our Solution

In order to combat the security and scalability issues of existing systems, we have developed *Synergy*, a reliable, scalable, open-source, and privacy-protecting energy measurement tool. The data is generated between various subsystems for processing and storage, and only after initial processing it is periodically uploaded to the cloud. The system has an integrated electricity usage visualization dashboard that displays real-time power usage statistics. Furthermore, the system is able to detect sudden spikes or irregularities in usage, informing users to the possibility of a blown fuse or other energy leakage by enabling quick alert generation.

Keeping the system on a local area network (LAN) enables the system to not require a constant internet connection. This allows the system to be scaled for larger complexes by allowing multiple devices to communicate wirelessly and has the added benefit of working in remote areas that have decreased Internet access. For example, using a LAN is much more sensible if a person leaves their stove on and needs an immediate warning. *Synergy* does not need to upload data to feed it back to the person. Instead, the real-time data is transmitted directly to the user. While there are plenty of ways that already exist to monitor energy usage, most of these methods fall short in security and scalability due to their reliance on cloud computing. By processing on the edge, our system is able to avoid these issues and empower users to take responsibility for their energy consumption more securely and on a much larger scale than is currently possible.

While there are plenty of ways that already exist to monitor energy usage, most of these methods fall short in security and scalability due to their reliance on cloud computing. By processing on the edge, our system will be able to avoid these issues and empower users to take responsibility for their energy consumption more securely and on a much larger scale than is currently possible.

Chapter 2

Project Requirements

In order to ensure the effectiveness of our system, we identified requirements that can be categorized as critical functional, critical non-functional, recommended, or suggested for implementation.

2.1 Critical Functional Requirements

These requirements define what our system will do. They are the essential elements of our system, and without them, our system would not operate.

- Gather electrical energy usage data from connected devices
- Visualize data in a real-time manner
- Visualize history of data gathered

The main objective in creating our system is to collect and analyze a building's energy usage data. In order to do this, we will have to first and foremost gather energy usage data to analyze, and then transform that data in a meaningful way; in this case, that entails visualizing the data.

2.2 Critical Non-Functional Requirements

In order to obtain the necessary functionality, our system will need to have certain properties. The following requirements are crucial in determining what our system will be able to do.

- Securely store usage data
- Measure energy usage at an accurate rate
- Be scalable to larger complexes

An energy monitoring system that is not accurate is useless. Our system will have 97% accuracy in measuring energy consumption, as that is the highest grade AC monitoring device we can obtain. Likewise, our primary goal is to make an energy monitoring device that can be scaled for larger buildings, such as offices and dorms, so scalability is crucial to our system.

2.3 Recommended Requirements

Recommended requirements define a set of requirements that we would like implement and are likely to be implemented but are not critical for the implementation of the system.

- Alert users of potential energy usage inconsistencies
- Send user-requested reminder notifications
- Modularize components for ease of expansion

Sending alerts based off of anomalous energy consumption and allowing users to request notifications about the energy consumption of certain devices would allow our system to become interactive rather than passive. Likewise, the more modular our system is, the easier it will be to install and work with.

2.4 Suggested Requirements

These requirements define objectives that we would like to accomplish but would only implement once all the other requirements have been completed. They have little effect on the outcome of the system and would only serve to further the implementation of the system.

- Connect to existing smart home devices
- Predict future usage trends and generate suggestions
- Maximize sampling rate

Our device itself should be small and affordable so that it can easily be used throughout an entire complex in order to give the best results. A higher sampling rate would increase precision of our analysis, and would be useful in making predictions of future energy consumption trends.

All of our requirements were able to be implemented with the exception of future usage trend predictions. This is because, the more we looked into it, there were more and more aspects to it that would be required to get it working how we envisioned it that it became a project of its own, outside the scope of our senior design project.

Chapter 3

Use Cases

In order to identify the different use cases of a system, we separated cases into viewing energy usage and editing the settings of the system. These use cases helped us better understand exactly how to design and implement the system. In Figure 3.1, the left side represents the available settings and the right side represents different ways of viewing different sets of energy usage. A detailed description of each individual use case follows.

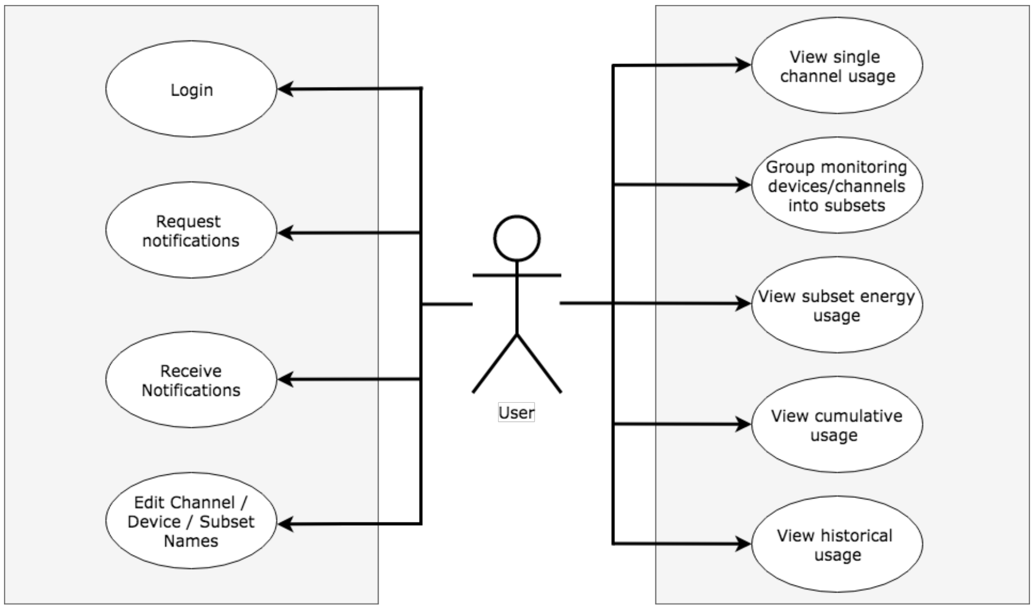


Figure 3.1: Use Case Diagram

3.1 Settings Based Use Cases

3.1.1 Login

To ensure that each actor is only able to access their data, or the data that is appropriate for them, it is important to have the actors login. This way, actors can access their own energy usage data. It also allows the ability to implement a role based user system so that a potential administrator could see the cumulative data but not per device information which could be a privacy violation. This user type system is discussed further in section 11.2

- **Goal**: Login to allow actors to access their energy usage data
- **Pre-condition**: Actor has device and account
- **Post-condition**: Actor has ability to see their energy usage data
- **Exceptions**:
  - Actor leaves blank fields
    - *Response:* Prompt user for information
  - Actor enters incorrect information
    - *Response:* Prompt user to create account or check that information entered is correct

3.1.2 Rename Channel / Device / Subset

Actors need to be able to easily identify how much energy each device is using. Because of this, they need to be able to change the name of a channel, device or a subset of the two to something meaningful to them, such as ”television” or ”living room”.

- **Goal**: Change the user facing name of the specified channel, device or subset
- **Precondition**: Channel, device or subset exists within system and is collecting usage data
- **Postcondition**: Channel, device or subset has been renamed
- **Exceptions**:
  - Item to be renamed does not exist within monitoring system
    - *Response:* Notify actors that item doesn’t exist
  - An error occurred whilst renaming the item
    - *Response:* Notify actors of error and prompt actors to refresh the page and retry naming the item

3.1.3 Activate New Device

To ensure that the system remains modular, actors are able to easily activate new monitoring devices. When plugging in a new device, actors are able to connect to the device using bluetooth 4.1 via their smartphone in order to provide it with LAN credentials. Adding new devices to the environment increases the breadth of monitoring and give further insight into an actor’s energy consumption.

- **Goal**: Add a new monitoring device to the system
- **Precondition**: New monitoring device has been connected to power
- **Postcondition**: System collects usage data from new monitoring device
- **Exceptions**:
  - New monitoring device has not been connected to the LAN
    - *Response:* Notify actors that no devices have been found and prompt actors to make sure devices are plugged in and to connect to it using bluetooth.
  - New monitoring device was found but an error occurred whilst connecting
    - *Response:* Notify actors of error and prompt actors to restart the monitoring device

3.1.4 Request Notifications

Synergy allows actors to request notifications when channels are left powered on at a certain time. This allows actors to remain informed about their energy consumption and give them an opportunity to identify unnecessary energy usage.

- **Goal**: Receive notifications when energy usage meets a specified condition
- **Precondition**: Usage data exists, condition is available for data set
- **Postcondition**: Actor receives notifications when condition is met
- **Exceptions**:
  - No usage data exists
    - *Response:* Notify actor that no data is being collected
  - Condition can never be met
    - *Response:* Notify actor that condition could never be met and prompt use to change condition
  - Actor has not provided contact information
    - *Response:* Notify user that no contact information has been set and prompt user to provide contact information

3.1.5 Receive Notifications

Synergy allows actors to receive notifications about their energy usage. Energy usage is used to generate a support vector machine (SVM) in order to classify energy usage data to identify outliers. Outliers can be used to find faulty or malfunctioning devices as well as short-circuits.

- **Goal**: Receive notifications based on real-time energy usage
- **Precondition**: System is collecting energy usage data
- **Postcondition**: System notfies users of outliers in neergy usage consumption
- **Exceptions**:
  - Not enough data is available to generate a support vector machine
    - *Response:* Notify actor that at least a week of energy usage collection data is required to predict outliers.
  - Outlier is incorrect
    - *Response:* Allow actor to inform system that outlier is incorrect so that it can use actor supplied information to inform future outlier decisions.

3.2 Energy Usage Based Use Cases

3.2.1 Group Monitoring Channels/Devices into Subsets

Actors have the option to group multiple channels, devices and even other subsets in order to visualize the energy usage of a subset of the total. For example, rather than knowing the energy usage of a toaster alone, an actor may be interested in the energy consumption of their kitchen as a whole. Because of this, users are able to arrange channels, devices and subsets in ways that are meaningful to them.

- **Goal**: Group together a set of channels, devices or other subsets
- **Precondition**: Monitoring channels/devices/subsets exist on the system
- **Postcondition**: Monitoring channels/devices/subsets are grouped together in a subset
- **Exceptions**:
  - Actors attempts to group channels/devices/subsets that they do not have access to
    - *Response:* Notify actor about no permissions to access channels/devices/subsets

3.2.2 View Single Channel Energy Usage

Actors have the option to view the energy usage of a single channel in real time. Energy usage can be viewed in a multitude of graphical and statistical formats including pie, bar and line charts as well as how many Amps are being used and the cost of that energy in the local currency at rate per killowatt hour set by the actor. This view is the building block of the system and from here, actors can expand the energy usage net to multiple channels, devices and subsets which is further discussed in section 3.2.3.

- **Goal**: View the energy usage data for a single channel
- **Precondition**: Channel exists and energy usage data exists
- **Postcondition**: Actor can visualize the usage data on screen
- **Exceptions**:
  - Channel does not exist
    - *Response:* Notify actor that the specified channel does not exist
  - Energy usage data does not exist
    - *Response:* Notify actor that no usage data exists for the channel
  - An error occurred fetching the data from the database
    - *Response:* Retry and if second fetch fails, notify actor that fetch failed and prompt actor to retry search or restart the system

3.2.3 View Subset Energy Usage

Actors can expand their energy usage visualization abilities by creating subsets. A subset can consist of a group of any number of channels, monitoring devices and other subsets. Additionally, actors have the option to further expand their view when creating a chart wherein they have the option to create a chart-based subset which can also be comprise of multiple channels, monitoring devices and pre-existing subsets. Similar to viewing the energy usage of a single channel, actors can view this information in a multitude of ways, however, the options laid out in section 3.2.2 can be expanded to separate how individual subset members are displayed on the chart as other options including viewing the average usage of each member and displaying a severity scale based on that member's average over time. Subsets are useful for grouping items together for example, grouping all the devices in the kitchen into a group called ”Kitchen”.

- **Goal**: View the energy usage of a group of monitoring devices and/or channels
- **Precondition**: All included monitoring devices and/or channels exist and are collecting usage data
- **Postcondition**: Actor can visualize the combined energy usage on screen
- **Exceptions**:
  - Monitoring devices and/or channels do not exist

        * *Response:* Notify actor that the specified monitoring devices and/or channels do not exist
    - Energy usage data does not exist
        * *Response:* Notify actor that no usage data exists for the specified monitoring devices and/or channels
    - An error occurred fetching the data from the database
        * *Response:* Retry and if second fetch fails, notify actor that fetch failed and prompt actor to retry search or restart the system

3.2.4 View Cumulative Energy Usage

Actors have the ability to view how much energy the system is cumulatively using across all of their monitoring devices. This enables actors to view their total energy usage. When creating energy usage metrics and charts, actors also have many of the same options available to them as outlined in section 3.2.2 and section3.2.3. Cumulative energy usage can be viewed in both realtime as well as historically which will be further discussed in section 3.2.5.

- **Goal**: Visualize all the data collected for a specified user including all of their monitoring devices and channels
- **Precondition**: User exists and energy usage data exists for the specified user
- **Postcondition**: Actor can visualize the combined energy usage on screen
- **Exceptions**:
    - Energy data does not exist
        * *Response:* Notify actor that no usage data exists
    - An error occurred fetching the data from the database
        * *Response:* Retry and if second fetch fails, notify actor that fetch failed and prompt actor to retry search or restart the system

3.2.5 View Historical Energy Usage

In additional to viewing realtime energy usage, actors have the ability to view their historical energy usage which displays as a snapshot of the time period specified. As explained in section 3.2.4, actors can view their cumulative energy usage of all time or the have the ability select a date range. The based on the duration of the date range specified, the system will automatically take a snapshot of the specified dates so as to reduce the amount of data that needs to be transferred and visualized. For example, if an actor would like to view the energy usage between January and March, the system will generate the average energy usage of each day and display that. However, if the user would like to visualize the energy usage between January 1st and January 3rd, the system will instead generate the average energy usage of each hour to display. This dynamic energy usage calculation gives the actor fine grain control over how they view their energy usage without adding additional overhead to the network and the graphing software.

- **Goal**: Visualize energy usage of a channel, monitoring device of subset between two specified dates.
- **Precondition**: Energy usage exists for the specified date period
- **Postcondition**: Actor can visualize the historical energy usage on screen
- **Exceptions**:
  - Energy usage does not exist during specified time period
    - *Response:* Notify actor that no usage data exists for the specified time period and prompt user to try a different time period
  - An error occurred fetching the data from the database
    - *Response:* Retry and if second fetch fails, notify actor that fetch failed and prompt actor to retry search or restart the system

Chapter 4

Conceptual Model

Conceptual models illustrate what the initial design of the project will look like. The following figures show the concepts for the front-end visualization dashboard. Each figure portrays a different view of the application with its own functionality.

4.1 Visualization Dashboard Home

Figure 4.1 shows the dashboard that will be displayed to the actor when logged in. By default it will show cumulative usage. The buttons on the left hand side will enable actors to change to, for example, the historical energy usage tab. The settings icon will open the settings pane as seen in Figure 4.3. The user icon will open the user pane. On the dashboard itself, there are a number of charting options including pie charts, sunburst diagrams and line charts that can show one's energy usage in different formats. These charts will be configurable by the actor.

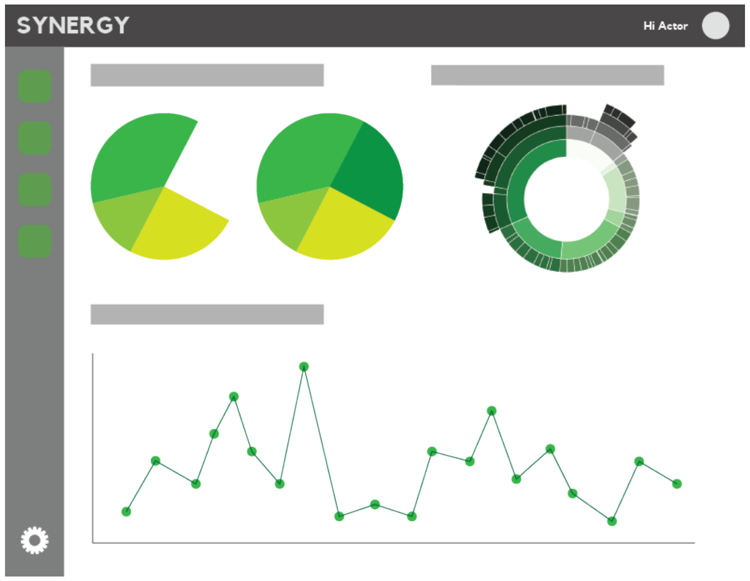


Figure 4.1: Visualization Dashboard

4.2 Menu Control

When an actor hovers over the sidebar menu buttons on the left hand side, the menu will expand allowing actors a more detailed description of what each button does as seen in Figure 4.2.

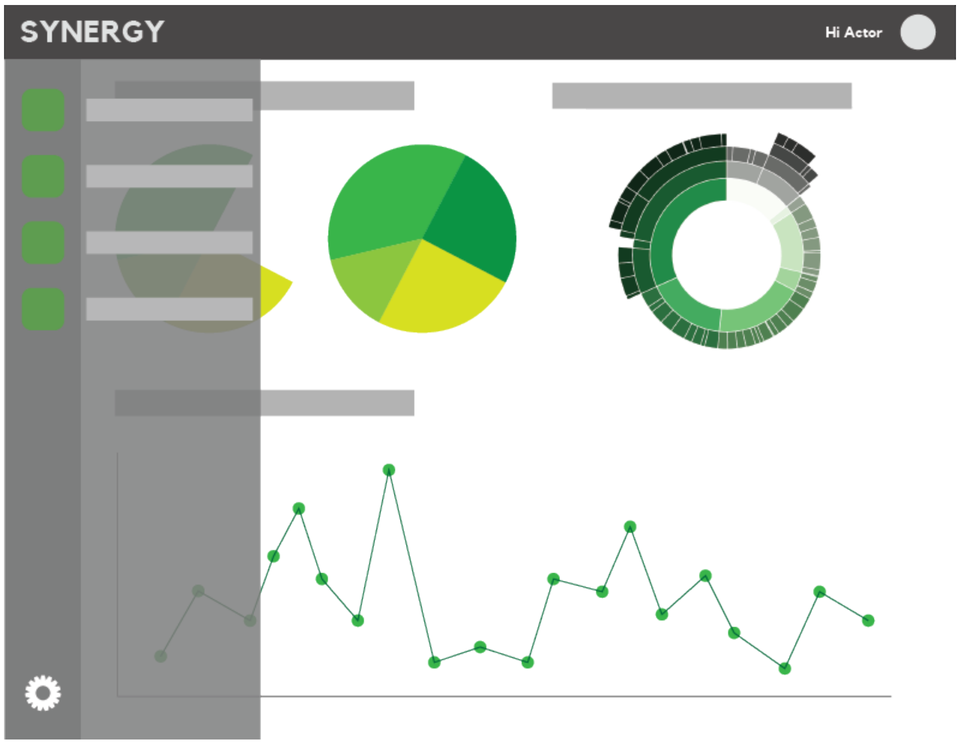


Figure 4.2: The sidebar can be expanded to see more details

4.3 Settings Pane

The settings pane allows actors to edit the system's settings. For users, this means the ability to activate monitoring devices, rename devices, rename channels and group devices and channels together. For administrators, this means the ability to group sets of users together and rename those groups. The final version of the settings pane will have many more options than those shown in Figure 4.3.

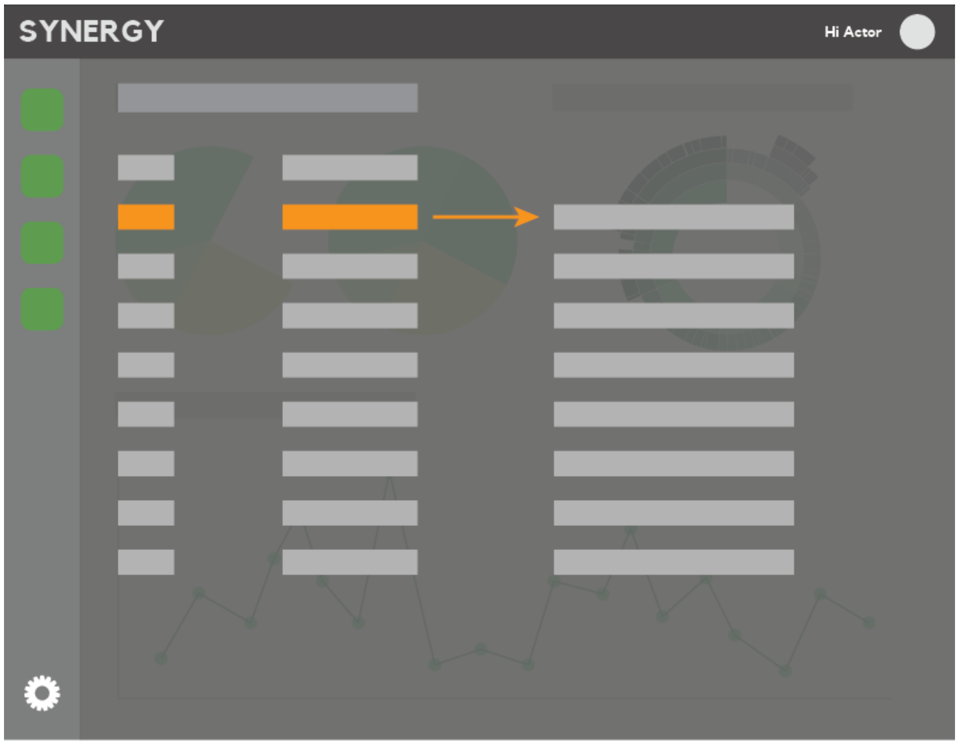


Figure 4.3: Settings can be accessed through the settings icon for renaming and grouping

Chapter 5

User Interface

The visualization platform provides a simple user interface that allows the user to interact with the system. It consists of three primary sections; settings, real-time charts and historical charts. The settings pane allows users to edit connected monitoring devices, create subsets and set reminders. The later two pane's gives users different options to view their energy usage.

When a user first connects, they will be presented with a log-in page as seen in Figure 5.1. Users can create an account using their email and have the option to enter the size of their family, the cost of their energy usage per kWh which can be determined from the user's energy bill. These numbers are then used to calculate energy usage statistics and costs per family member. Users also have the option to enter their phone number if they would like to set-up reminders. All of this information can be added/changed at a later from the user's profile.

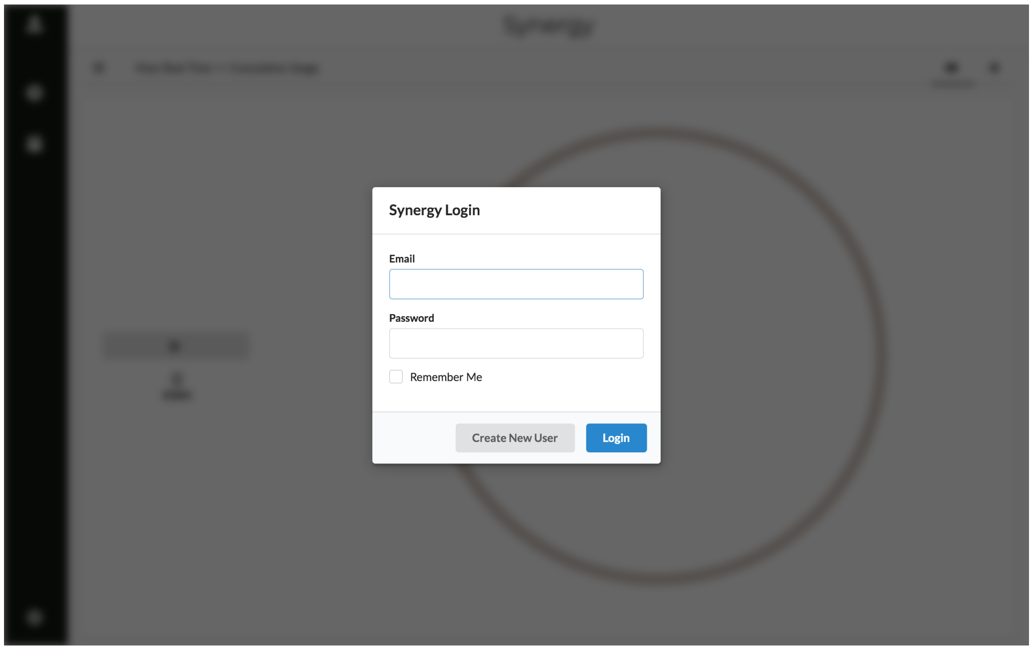


Figure 5.1: Login Page to Access Visualization Dashboard

5.1 Settings Pane

The settings icon gives users the ability to view any existing channels, monitoring devices, subsets and reminders. Channels and monitoring devices make up the core of the system. A monitoring device can consist of 1 or more channels. The settings pane allows users to search for any named channel or monitoring device and will then redirect them to the appropriate panel. Figure 5.2 shows what the list of channels looks like. From here, users can view whether a channel is currently powered on. Additionally, they have the option to view that channel in a chart, edit any notifications that are attached to the channels or edit the channel's name.

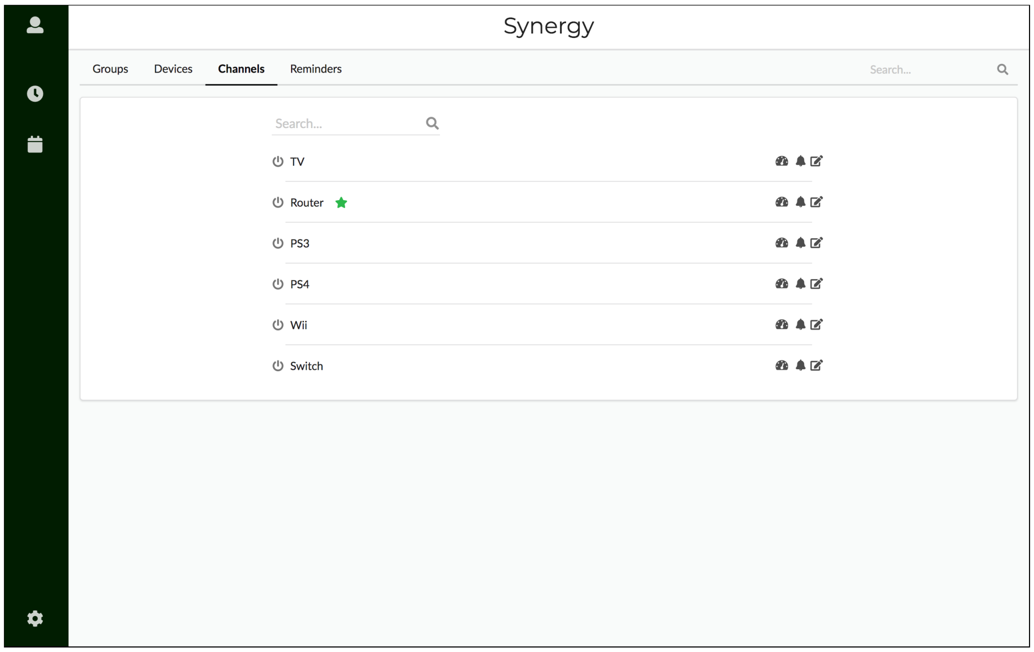


Figure 5.2: View and edit all channels available in system

Similar to the channels panel, Figure 5.3 shows the monitoring device panel which displays the devices that have been added to the system. Upon selecting any device, the right side will be populated with all of the channels that exist on that device. From here, users can also access any of the options available in the channels list. Unlike the channels list. Individual monitoring devices can be removed from the system using the delete button.

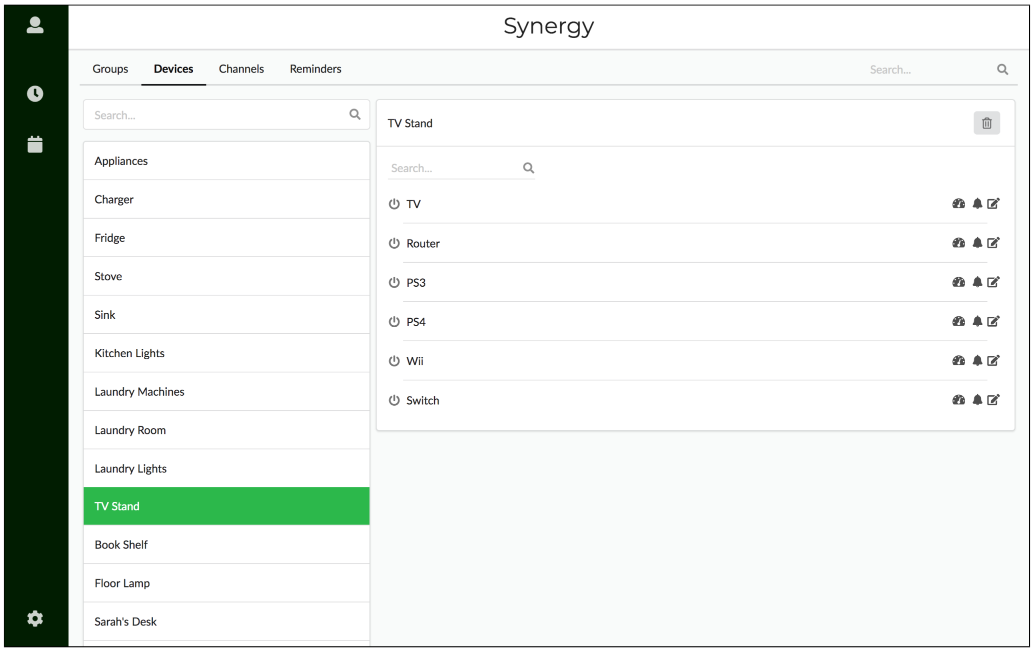


Figure 5.3: View and edit all monitoring devices available in system

Both monitoring devices and channels can be grouped into subsets from the groups panel. Figure 5.4 illustrates the group creation process. Users have the option to choose a group name as well as 2 or more group members. Members can consist of other groups, devices or channels. Once created, groups can be viewed in the groups panel. Similar to the monitoring devices panel, created groups are listed on the left hand side. Selecting a group will show its members on the right hand side. Group members are separated by type and can be searched for using the integrated search box. The groups panel can be seen in Figure 5.5.

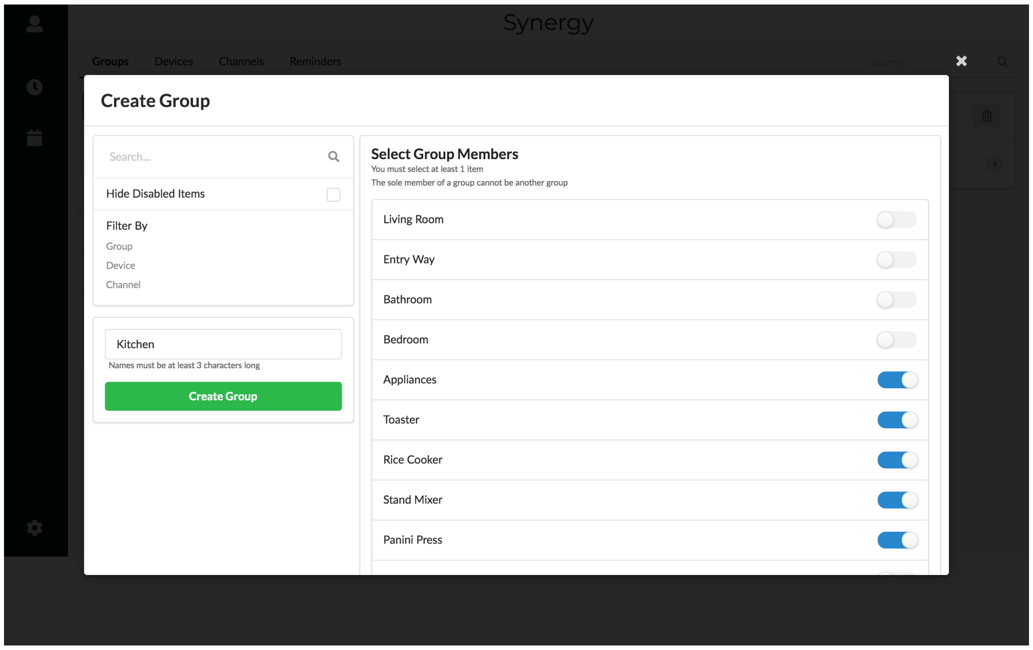


Figure 5.4: Create a new group with a combination of groups, devices and channels

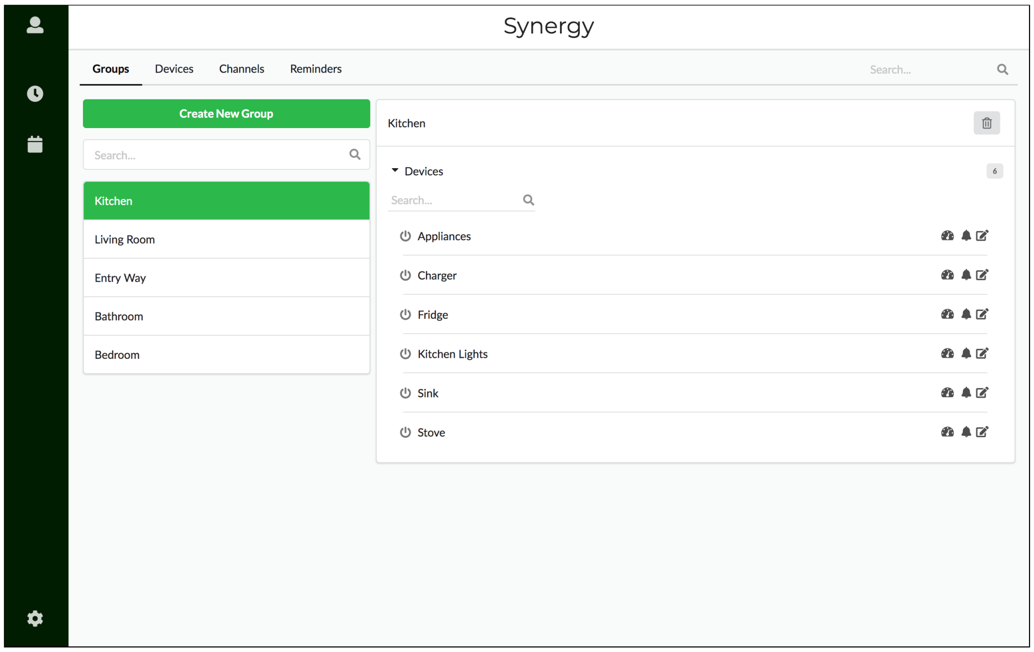
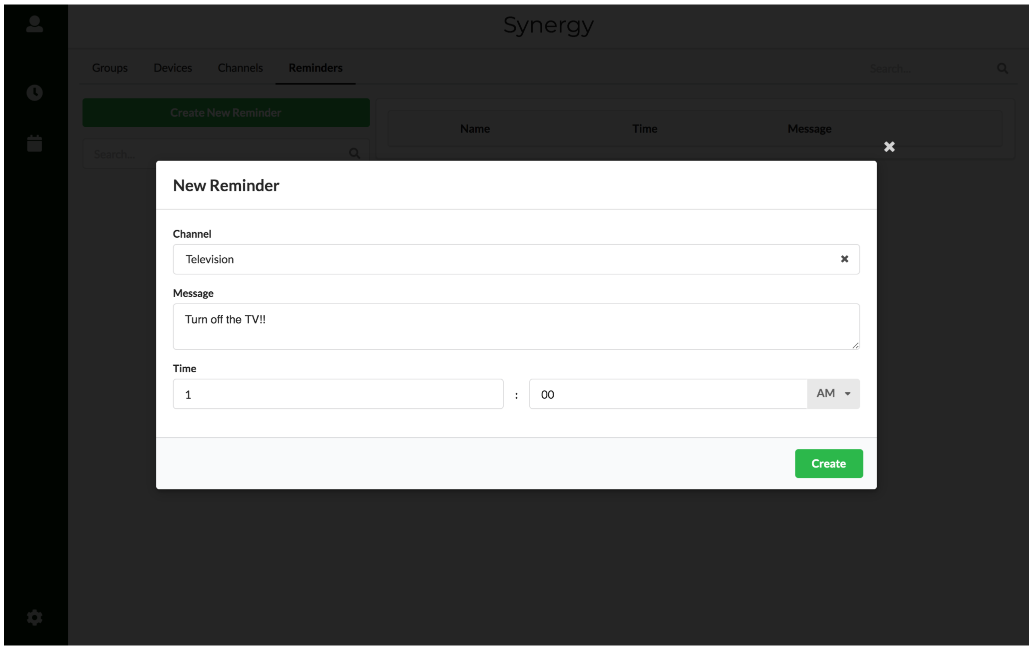


Figure 5.5: View and edit all groups that have been added to the system

Lastly, the settings pane allows users to create reminders. Figure 5.6 displays the create a reminder form. Users have the option to choose a channel to add the reminder to, set a message that will be sent to the user's phone which and a time to send the reminder at. The reminders panel displays the a list of created reminders as well as the time they will be sent.



New Reminder

Channel
Television

Message
Turn off the TV!!

Time
1 : 00 AM

Figure 5.6: View and edit all reminders that have been created for the system

5.2 Real-Time Energy Chart Pane

Energy usage can be viewed in multiple formats. When the user firsts loads the application they are presented with a pie chart that shows the cumulative energy usage of the entire system. Figure 5.7 shows the energy usage of 5 subsets that one might see in a standard household. In this chart, the Kitchen is using the most energy usage which makes sense given that appliances like the refrigerator are always using energy. Energy usage charts can be configured in a multitude of ways. Figure 5.7 shows the data as a pie chart where each of the segments represent the percentage of the total energy usage being used by each group. However, this same data could be shown as one of three other chart types including line and bar charts as well as a sunburst diagram. Furthermore, each chart type has additional options that can be toggled. For example, the pie chart can be configured to display as a donut or show the individual percentages of the members in each group.

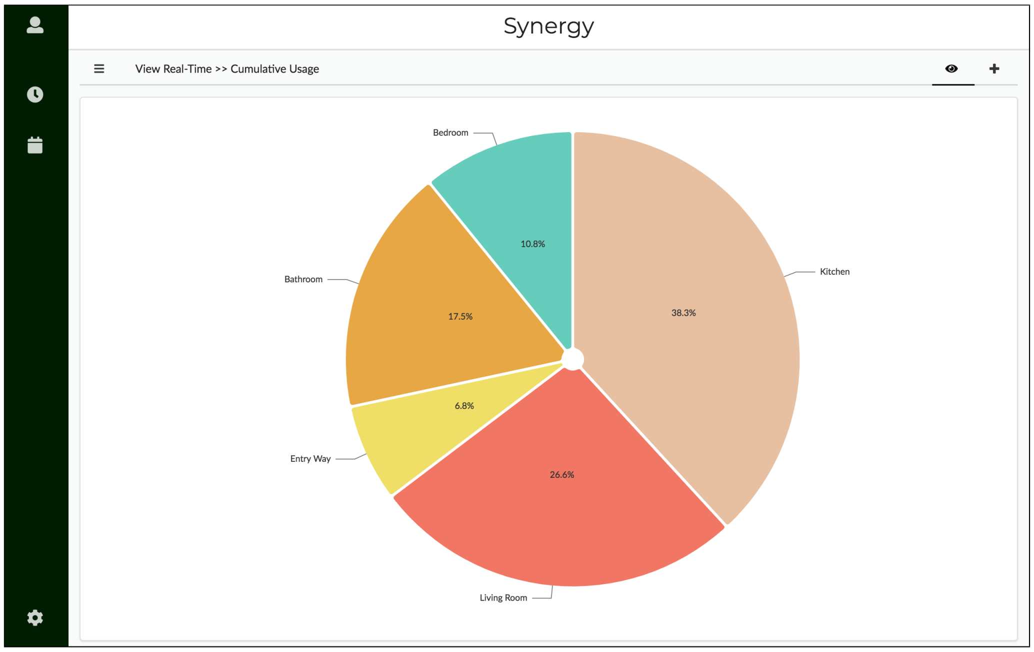
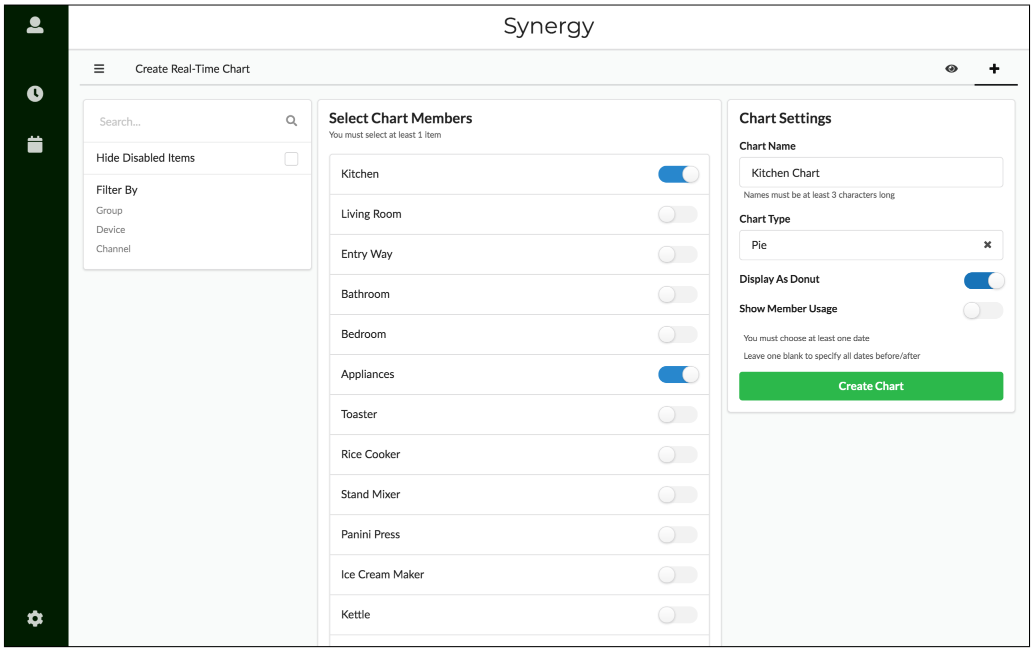


Figure 5.7: View the cumulative energy usage in real time

Real-time energy usage charts can be created and configured using the create a chart form. Similar to creating a group, Figure 5.8 allows users to choose any number of channels, monitoring devices and subsets. When creating a real-time chart, users can choose a chart type from the aforementioned chart types and enable any options that they would like to have enabled. Finally they can choose a chart name. Charts can be selected from a list using the burger menu found in the top left of the chart pane.



Select Chart Members
You must select at least 1 item

Kitchen	on
Living Room	off
Entry Way	off
Bathroom	off
Bedroom	off
Appliances	on
Toaster	off
Rice Cooker	off
Stand Mixer	off
Panini Press	off
Ice Cream Maker	off
Kettle	off

Chart Settings

Chart Name: Kitchen Chart
Names must be at least 3 characters long

Chart Type: Pie

Display As Donut: on

Show Member Usage: off

You must choose at least one date
Leave one blank to specify all dates before/after

Figure 5.8: Create a new chart to visualize a subset of energy usage data in real time

5.3 Historical Chart Pane

The historical chart pane is aesthetically very similar to the real-time chart pane. The default historical energy usage chart shows the cumulative energy usage of the system over the past week as a line chart as seen in Figure 5.9. The cumulative historical chart will automatically attempt to group data points in order to obtain the average for a day. If the historical usage was displayed for the period of a couple of months, then the algorithm will get the average over a week in order to reduce the network traffic and data processing.

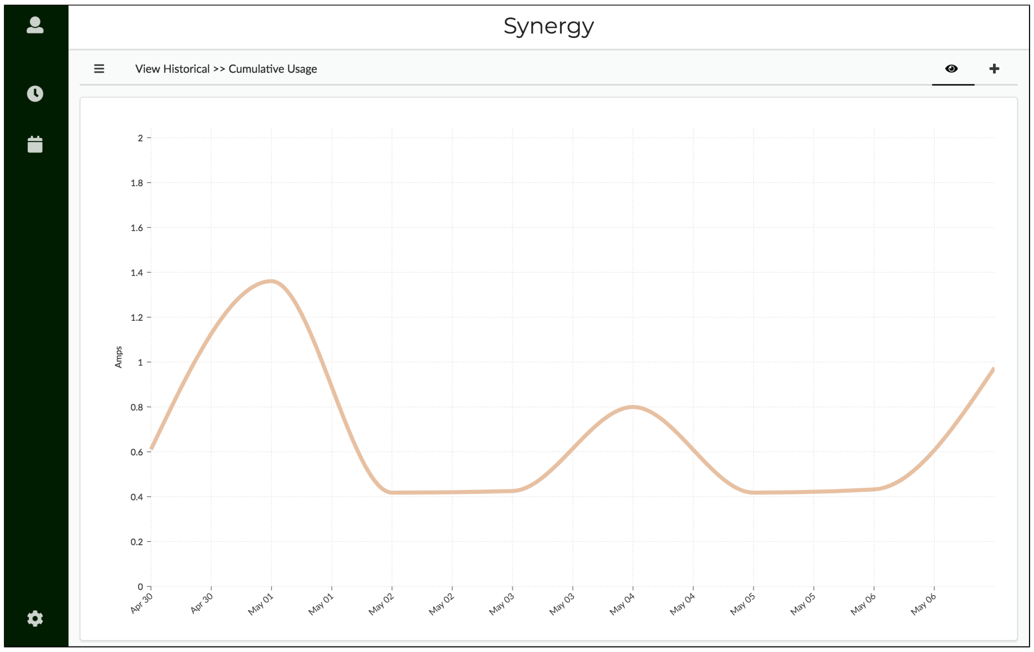
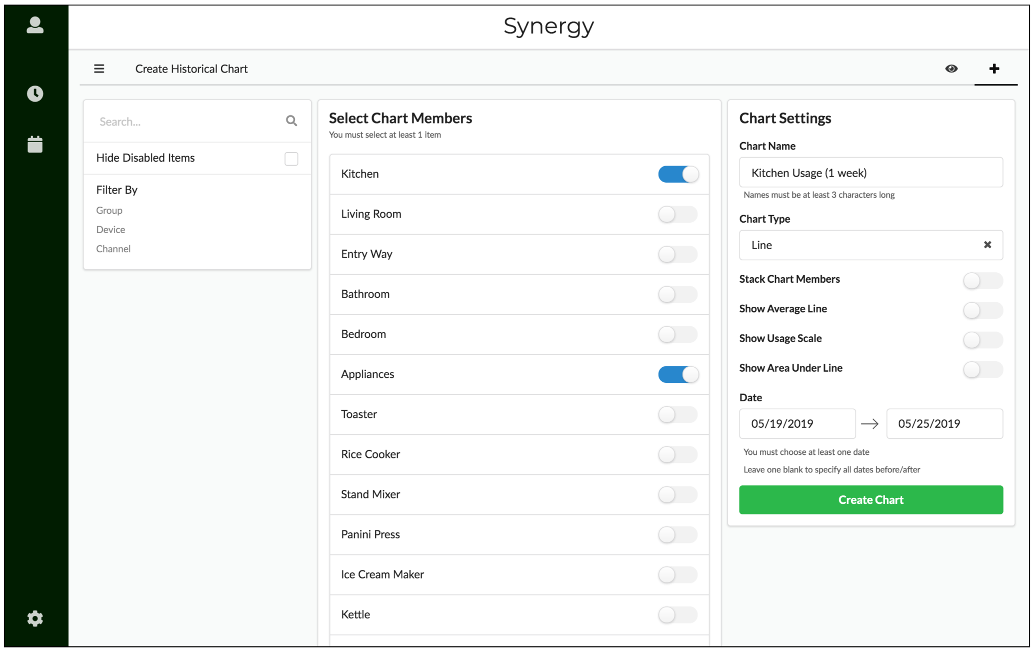


Figure 5.9: View the historical energy usage

Similar to creating a real-time chart, creating a historical chart has the same general layout. However, users are also able to choose a date range as seen in Figure 5.10.



Hide Disabled Items ☐

Select Chart Members

You must select at least 1 item

Kitchen	✓
Living Room	
Entry Way	
Bathroom	
Bedroom	
Appliances	✓
Toaster	
Rice Cooker	
Stand Mixer	
Panini Press	
Ice Cream Maker	
Kettle	

Chart Settings

Chart Name: Kitchen Usage (1 week)

Names must be at least 3 characters long

Chart Type: Line

Stack Chart Members

Show Average Line

Show Usage Scale

Show Area Under Line

Date: 05/19/2019 → 05/25/2019

You must choose at least one date

Leave one blank to specify all dates before/after

Figure 5.10: Create a new chart to visualize a subset of energy usage data over time

Chapter 6

System Design

Synergy is a combination of both frontend and backend components. The main system runs on an array of Raspberry Pis (RPi) [17]. We specifically used Rasberry Pi 3B+ for our system, however any Rasberry Pi or other Linux-compatible system would work as well. These mini computers have been modularized, creating an expandable system. Each RPi connects to an AC current monitor which can detect the energy usage of connected devices. *Synergy* also provides users with a web based visualization platform along with a web server to manager data flow.

6.1 Architecture Design

Synergy is a data-centric system. The energy usage is monitored by individual AC monitoring units which broadcast energy usage data to a central processing unit over a local area network through MQTT [11]. By using a centralized database, we are able to keep the data localized which provides increased protection of secure data and increases the speed at which we can provide updates and statistics. The database can also be configured to back up to an AWS MySQL database for increased historical usage storage [2]. The manager then processes the incoming data in order to respond to user requested notifications and determine whether a device is malfunctioning. Figure 6.1 illustrates how each current monitoring device wirelessly connects to the manager and how the manager interfaces with connected databases.

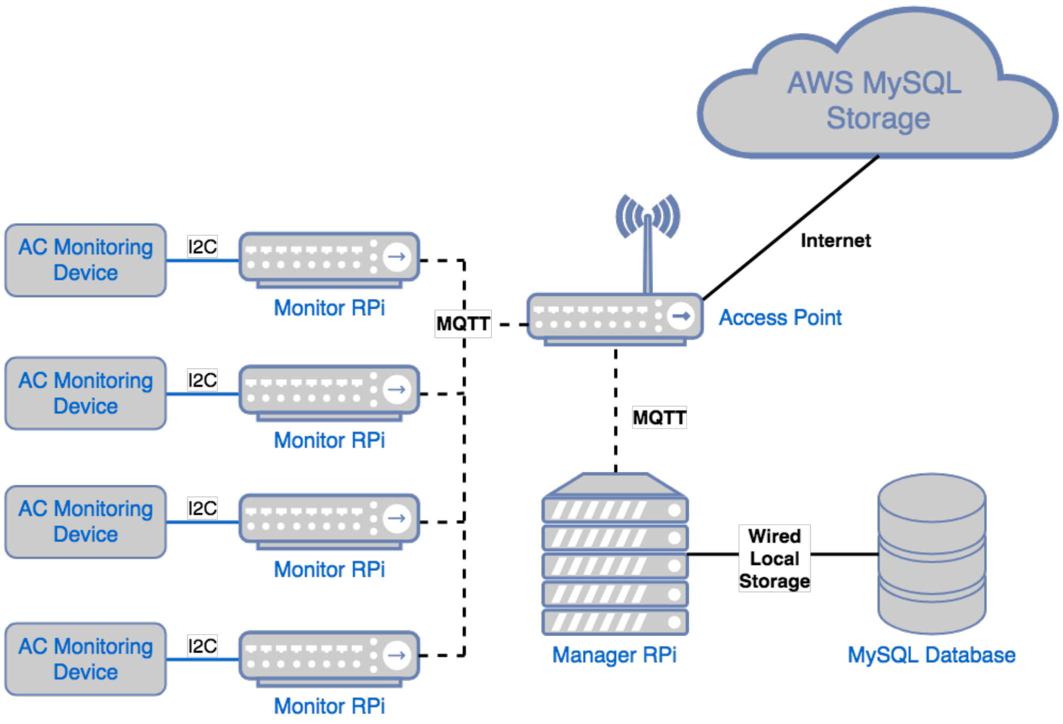


Figure 6.1: Data Centric Architecture

6.2 Hardware Design

Each AC monitor consists of two components; the energy monitor and a Raspberry Pi that publishes the data to the server. The AC current monitoring device we have chosen has a max error rate of 5%, and has a built in I^2C interface that allows it to be connected to a Raspberry Pi [1]. The AC current monitor shown in Figure 6.2 has the ability to monitor up to 2 channels, however, AC monitors are available in configurations up to 12 channels. to In order to ease the user experience, monitoring boards can be connected directly to individual outlets or into the electrical switch box of a building. This means that it could be used to monitor the energy usage of 12 devices, or just 12 connected lines, where each line has a maximum current throughput of 20 amps. While monitoring occurs passively within the sensor, the connected Raspberry Pi will query each connected device's current 2 times per second. This allows us to process the energy usage in real time.

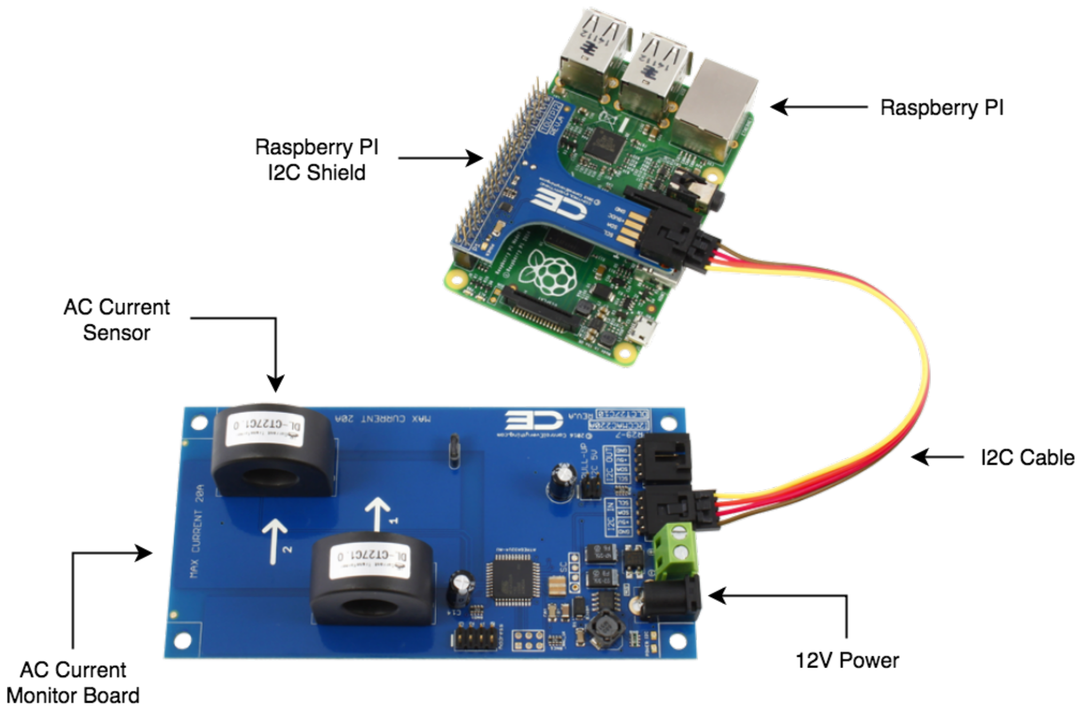


Figure 6.2: Hardware Configuration

6.2.1 Hardware Specifications

Synergy's monitoring system, as aforementioned, runs on an array of Raspberry Pis (RPi). Each mini-computer operates using a custom Unix based operating. Furthermore, in order to converse with the current monitors, the RPis are integrated with a multitude of IOT protocols that allow for further smart home integration.

Raspbian OS

Raspberry Pis come configured with Raspbian OS [18], a custom Unix based operating system. This OS has Python installed by default. Furthermore, it supports most Unix based commands which will allow for ease of development and a simple set up in production.

I^{2}C Protocol

Inter-Integrated Circuit (I^{2}C) [8] is a protocol designed to connect input-output interfaces in simple circuit installations. Each AC current monitor comes with I^{2}C built in and allows for seamless communication between it and the RPi.

6.2.2 MQTT

The energy usage is monitored by individual AC monitoring units which broadcast energy usage data to a central processing unit over a local area network through MQTT [11], which is a machine-to-machine communication protocol. The manager then processes the incoming data.

6.3 Backend Architecture

6.3.1 Overview

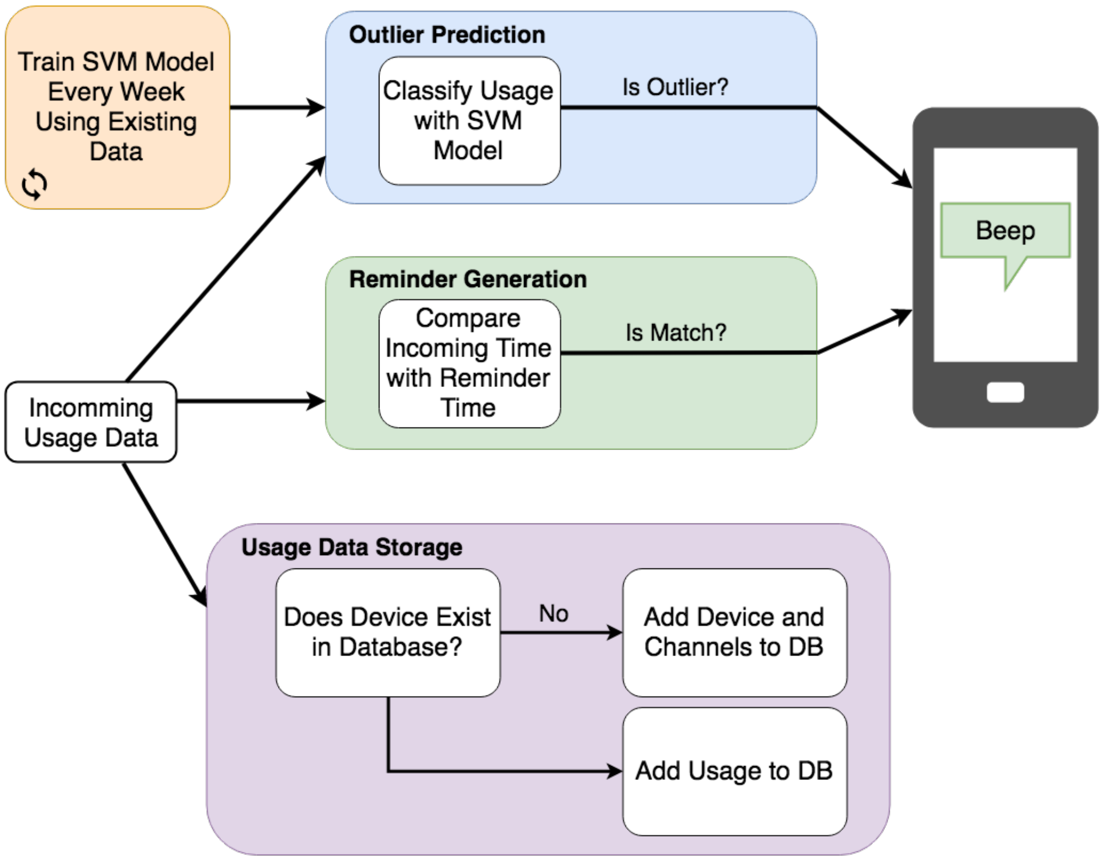

Train SVM Model Every Week Using Existing Data
Outlier Prediction
Classify Usage with SVM Model
Is Outlier?
Beep
Reminder Generation
Compare Incoming Time with Reminder Time
Is Match?
Incomming Usage Data
Usage Data Storage
Does Device Exist in Database?
No
Add Device and Channels to DB
Add Usage to DB

Figure 6.3: What happens to an energy usage data point as it is received

Synergy's frontend application has very little interaction with the backend. The backend consists of two components; the manager and the monitor. The manager processes incoming energy usage data from AC monitors not will access monitoring data from a Unix processing unit running on a Raspberry PI. This processor will enable queries to be forwarded from the React app so as to populate the data visualization graphs. As such the backend encompasses both the processing done on the data as well as the data storage. Furthermore, each individual Raspberry PI connected current monitor will be running a similar program designed to pull usage data and send it to the processing unit.

6.3.2 Technologies Used

Python

Python [16] is a scripting language known for its simple syntax and high readability. It enables developers to do more in fewer steps compared to other programming languages. Python comes pre-installed on Raspberry Pis and has little overhead. Synergy uses python in both the modular AC monitors as well as the manager which processes the data gathered from each monitor. Using the same language between backend components allowed for tighter coupling between modules and shared components.

Twilio

Twilio [42] is a cloud communications platform that was used in *Synergy* in order to enable SMS messaging. Through the messaging, we were able to send user-requested notifications, as well as alerts when energy abnormalities or inconsistencies were detected in a user's device.

Scikit-learn

Scikit-learn [34] is a machine learning library that was used to implement a one-class support vector machine (SVM). An SVM is used because the nature of our system does not easily allow for faulty data to train on. So instead, it is essentially able to generalize what a class of "good" data looks like at a certain time, and if a data point deviates too much from that, then Synergy sends a notification to the user which is possible due to the Twilio API. The SVM creates a new model for each channel once a week by training on all of the historical data gathered for that channel since the system was activated.

MySQL

MySQL [12] is the premiere SQL compliant relational database. A relational database is a table-based database which stores data in what looks like massive tables. This type of system is best used when you know what the structure of the data is going to look like. A system like this will allow us to store millions of data samples for both reference and visualization as well as using historical data samples for prediction.

Amazon Web Services: MySQL

At 2 samples per second, the system can generate store upwards of 170 thousand samples every day which will eventually turn into a massive database. Amazon Web Services (AWS) provides cloud MySQL database access to the same database structure without having to store the data on local drives [2]. This will enable *Synergy* to store all of its accumulated data which can then be accessed if a user would like to see their usage history.

6.4 Frontend Architecture

6.4.1 Overview

Synergy provides the user with a web-based visualization platform, which is built using React and Redux. Charts are rendered using a React-based charting library called Nivo charts.

In order to facilitate data transmission between the UI and the central database, we designed a web server that uses a combination of a RESTful API and web sockets, which provide a persistent connection between a client and a server. The web server was built using express.js in order to simplify route handling. The UI, server and database communicate openly through an NGINX reverse proxy.

Through the User Interface, a user is able to view the energy usage of any single device or subset of devices that they have manually grouped together. Energy usage can be view in numerous formats including individual statistics, and graphical charts. Charts can be chosen and configured on a per chart basis. Furthermore, the user can choose to view the real time usage or historical usage.

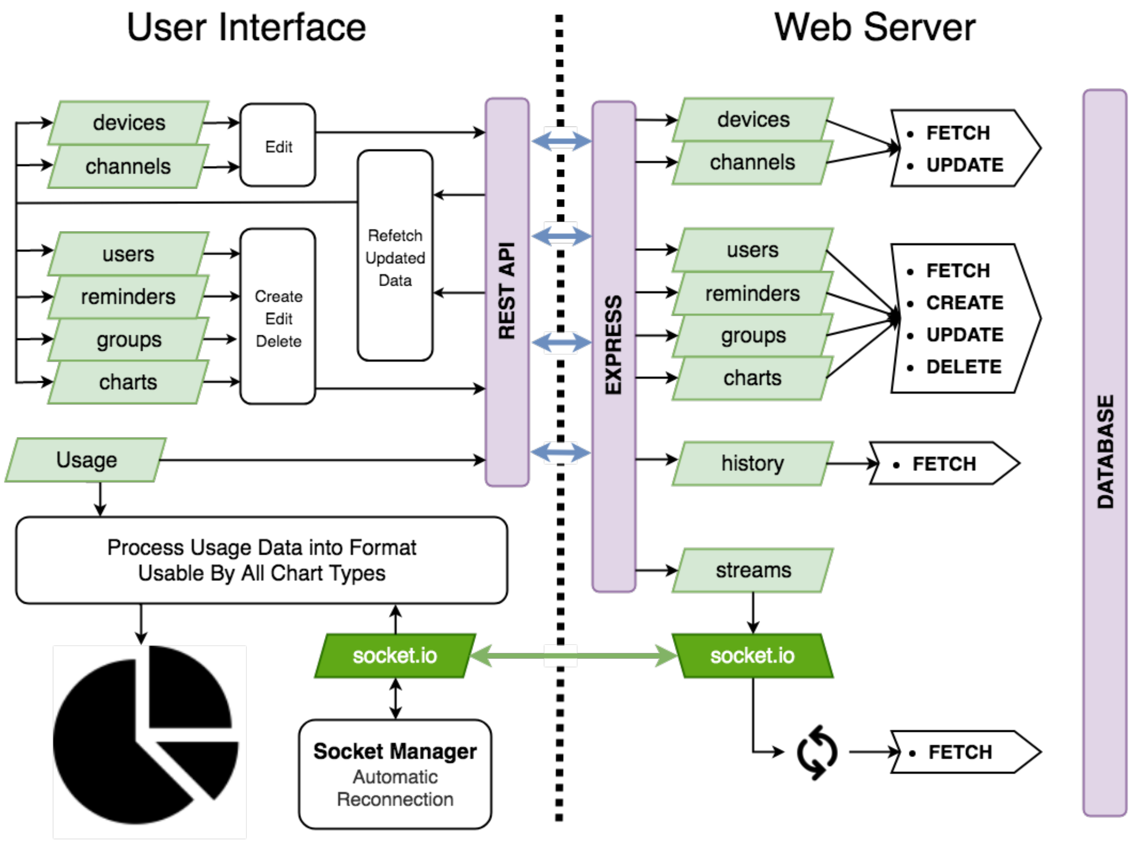


Figure 6.4: Interactions between the user interface and web server

The user interface works in conjunction with the web server in order to provide the user access the the data needed to operate the system. Whenever updates are made to the database, the UI performs a system sync in order to keep a record of the latest information available. The web server also manages any active sockets that the UI is using.

Furthermore, the UI attempts to keep further processing to a minimum by storing the real-time and historical usage data in a format that all of its charts can use.

6.4.2 Technologies Used

Synergy requires an internet enabled display for both configuration and visualization. The user facing application is be built for web browsers but future implementations can be modified for mobile devices. It is built on Javascript using a combination of NodeJS and ReactJS (referred to as Node and React respectively) frameworks to provide a functional yet simple interface.

Javascript

Javascript [9] is a scripting language that was initially developed for web browsers. This is what makes web browsers interactive. Not only does it provide the ability to develop interesting animations and interactions, it also allows for dynamic element generation and auto refreshing of page content. This is the core concept behind React.

ReactJS Framework

React [19] uses a virtual document object model (DOM) in order to identify what components of the web page have changed in an attempt to lessen the load of page updates. This framework enables *Synergy* to run continuously and in real-time without causing impacts to performance or inundating the web server.

NodeJS Framework

React is built on Node [14] which is a run-time environment for Javascript enabling it to run as a standalone system rather than purely browser based. Node gives us access to a multitude of additional libraries such as the ability to create dynamic charts or using pre-built methods to manipulate the date. These libraries can be used in conjunction in order to increase performance and provide stable visualization platform.

Socket.IO

Socket IO is a web socket implementation designed for NodeJS [38]. Sockets allow *Synergy* to stream real time data from the backend to the user interface. Multiple streams can be created in parallel. Sockets are typically used for messaging platforms and other types of systems that require realtime data transference.

ExpressJS Framework

In order to relay information from the backend to the frontend, we developed an Express web server [6]. Express allows the user facing application to request specified information such as the energy usage data between two dates as well as stream real-time data from the database. It also serves the visualization application by sending a specified data set to the user's browser. Although later versions of Node provide the ability to create a web server, Express simplifies route handling allowing us to access the *Synergy* database at different points with ease.

Development, Staging and Production

Node also enables environment based development. We used a combination of Babel, SASS and Webpack in order to build the user facing application. These libraries were used solely for development and are not required in the final version that is delivered to the customer allowing us increase the rate of development. Once the app has the minimal functionality, we used what is known as a staging environment to test it with the rest if the system. This is typically the final step before releasing the app to consumers. However, we also used this environment to test new features as they came out of development. Our final application fits into the last environment: production which is considered the most stable version available.

6.5 Design Rationale

The initial concept for this project was primarily an energy monitoring system however, upon researching similar systems on the market, we found that any brand name devices that could monitor energy usage also had an accompanying visualization software with few to no APIs to access said data. In discovering this, we quickly realized we would not be able to create an energy visualization tool without any data to visualize. Thus, we decided on building both the monitoring and visualization framework.

Because the initial system requirements were so broad, we had a large range of options to choose from regarding technologies, frameworks and libraries. In the end we decided to play to our strengths. Both of us had experience using both Javascript and Python and one of us had used React in a multitude of Projects. This led us to choose a project structure with these language constraints in mind.

The outcome of these experiences lead to *Synergy* which combines both the frontend Javascript experience with the python backend processing experience. The frontend app is designed to interface the system directly with users. It allows actors to configure, view, monitor and review their energy usage. The backend processes all of the data received from the AC current monitors into a usable format. It also identifies trends and patterns for the benefit of the user.

6.5.1 React vs. Other Javascript UI Frameworks

- **React**: The group has the most experience with this framework. It provides an easy, structured way to build state-based components with decreased overhead from the server.

- **Angular**: Uses a model-view-viewmodel system that has a steep learning curve and added complexity that was out of the scope of this project [3].

- **Vue**: Lack of support and English language documentation would make learning difficult. Furthermore, the community is small and it would be difficult to find sample use cases [43].

6.5.2 ExpressJS vs. Other Web Frameworks

- **ExpressJS**: Provided the perfect out of the box environment for a project of this size including a basic web server, routing and database functionality without restricting development flow.

- **Django**: Provides too much out of the box functionality and requires you to use all of their built-in functionality otherwise it won't work correctly [4].

- **Flask**: Provides too little out of the box functionality requiring a much of the simplicity provided by ExpressJS to be custom built for the project [7].

6.5.3 MySQL vs. Other Relational Databases

- **MySQL**: Easily allows storage of large data sets including both relational and object based data. Also has simple integration with AWS cloud services.

- **SQLite3**: All data is stored on local drives meaning we would have to use 2 different databases in order to use a cloud based database in conjunction with our local one [39].

- **PostgreSQL**: Although this is very similar to MySQL, we decided against it do it its lack of updates and extensibility [15].

Chapter 7

System Test Design

To ensure that our system behaves correctly in a multitude of environments, we tested the code in a controlled environment by setting up a differentiation between the the development realm and staging realm. This allows us to continue to run the system since early January of 2019, a period of approximately 5 months. This early testing phase was the staging environment and consisted of code that had been finished and was ready for testing as part of the larger system. Meanwhile, the development environment was used for code that was still in development.

7.1 System Analysis

Throughout our staging period of approximately 5 months, the project highlighted some potential issues that may need to be addressed in later iterations of the project past our projected completion date.

First, the system is only capable of measurements that use roughly a minimum of 100mA and a maximum of 20A. However, most house hold devices do not come close to using this maximum. The only issue would be if the system were to be installed in a large building's electrical panel. However, due to the scale that the system was tested at, this issue has a low risk.

Secondly we noticed that the system generates approximately 800 kilobytes per second that a device is power on. The data processor and user interface is designed to handle large data sets however, there is a limit to how much data should be being sent over the network. No one wants to have to wait for over 10 minutes while their browser receives over a gigabyte of data.

Lastly we recognize that this large amount of data can become extremely processor intensive. Although the system itself is modular, it is worth noting that in order to scale, the server than the manager runs on also needs to be scaled appropriately. In large setups, it might be prudent to also create system redundancies and fail-safes so that there is not a single point of failure.

7.2 Developmental Testing

In order to test our code we used a combination of white-box and black-box testing. Unit testing is a form of white-box testing that requires knowledge of how the system works on a programmatic level. Unit testing often involves a combination of testing the inner workings of functions and the return values of those functions. It also allows for snapshot testing which essentially stores a snapshot of the return value to match against. Snapshot testing is particularly good for large data sets and complex components, both of are commonplace in our project.

7.2.1 Javascript

For Javascript unit testing, we used Jest [10]. Jest is a unit testing framework developed for Javascript applications. It can test standard function calls, return values and is snapshot testing enabled. It also has features for module mocking which allows us to use fake components for predictable testing environments. Furthermore it comes with built in code coverage assessment tools which enabled us to check how well we are testing the system during development.

7.2.2 Python

For our Python development, we conducted unit testing using PyTest [26]. PyTest is a unit testing framework developed for Python applications. It is extremely similar to Jest with built in tools for standard unit testing, snapshot testing.

7.3 Usability Testing

User satisfaction is essential to the success of our system. Because of this, we generated test cases based on how a user will interact with the system in conjunction with our testing as the developers. To test the usability of our system, we used black box testing, in which the information about the code is ignored and the specification of the system is the driving force. We checked whether a select set of inputs produce the desired outputs, which we formulated using the use cases outlined in chapter 3.

7.3.1 Test Cases

- User can login to reach dashboard
- User can successfully activate a device
- User can view energy usage information from dashboard
- User can rename the channel of added device
- User can request notifications and receives notification when triggered

- User can receive notifications when a faulty channel is detected
- User can activate a second device
- User can group the two or more devices into one subset
- User can view the energy usage of the subset
- User can view the total energy usage of their devices
- User can view their past energy data gathered by system

7.4 System Testing

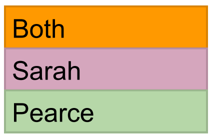
Although the use case and usability testing works well in the short run, the large amount of data that the system generates means that we needed to make sure that the system would be able to function as it was used on a larger scale. Using a selection of mock data, we ran a series of soak tests to see how the system would handle itself in high stress situations such as receiving energy monitoring data from over 20 devices at a time. Other types of soak tests included having many simultaneous real-time streams, increasing the amount of clients accessing the system at one time as well as requesting long periods of data for the historical charts.

Unfortunately, we were only able to use mock data for these soak tests due to having a limited amount of time to use the system, having a limit number of monitoring devices to test with as well as having a limited budget. In order to overcome some of these limitations, we collected user feedback about how the system operated under some of their custom test cases. As more and more features became available. The user feedback became a loop so that we could constantly adjust in order to provide the best user experience.

Chapter 8

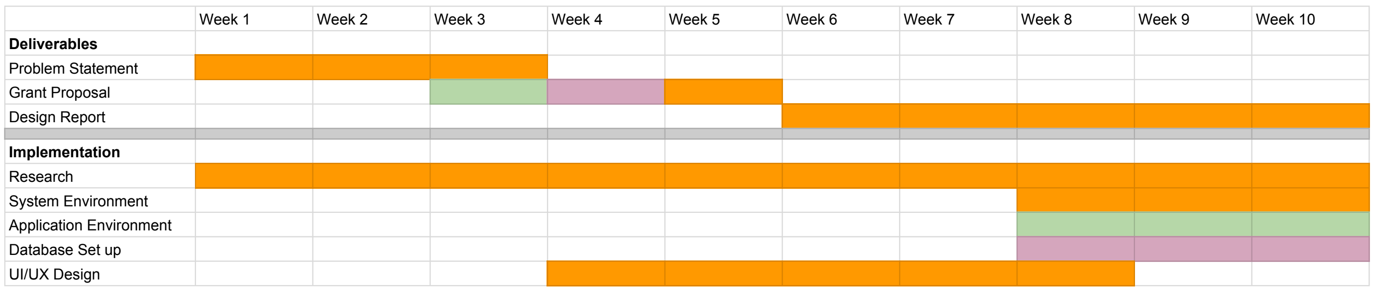
Project Timeline

In order for our development to run as smoothly as possible, we created a timeline to use as a reference point for what we should have accomplished throughout the three quarters we worked on this project. Each quarter had different deliverables that had to be completed, along with general progress on our system. By sticking to this layout, we had enough time to address any issues that arose. Deviating from this timeline and not meeting the deadlines we set for ourselves could have prevented us from creating the high quality product we aimed to produce. As such, it was important that we follow the project timeline as closely as possible.



🟧	Both
🟪	Sarah
🟩	Pearce

Figure 8.1: Legend for Development Timeline



	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Deliverables										
Problem Statement	🟧	🟧	🟧							
Grant Proposal			🟩	🟪	🟧					
Design Report						🟧	🟧	🟧	🟧	🟧
Implementation										
Research	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
System Environment								🟧	🟧	🟧
Application Environment								🟩	🟩	🟩
Database Set up								🟪	🟪	🟪
UI/UX Design				🟧	🟧	🟧	🟧	🟧		

Figure 8.2: Fall Development Timeline

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Deliverables										
Design Review	🟧	🟧	🟧							
Revised Design Report		🟧	🟧	🟧						
Operational System	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
Implementation										
Hardware										
Set up Environment	🟧	🟧								
Configuration		🟧								
AP Networking			🟧	🟧						
Frontend										
Settings	🟩	🟩								
Dashboard				🟩	🟩	🟩	🟩	🟩	🟩	
User Account					🟩	🟩				
Administrator Account									🟩	🟩
Backend										
Database integration			🟪	🟪						
Processing				🟪	🟪	🟪	🟪	🟪	🟪	
RPi Communication	🟪	🟪								
Networking				🟧	🟧	🟧				
Testing										
Unit Testing					🟧	🟧	🟧	🟧	🟧	🟧
Acceptance Testing									🟧	🟧

Figure 8.3: Winter Development Timeline

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Deliverables										
Design Conference	🟧	🟧	🟧	🟧	🟧					
Final Project Report						🟧	🟧	🟧	🟧	🟧
Completed Implementation	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
Implementation										
Frontend Refactoring			🟩	🟩	🟩	🟩				
Backend Refactoring			🟪	🟪	🟪	🟪				
Hardware Refactoring	🟧	🟧	🟧							
Testing										
Acceptance Testing	🟪	🟩					🟪	🟩		
Usability Testing	🟩	🟪					🟩	🟪		

Figure 8.4: Spring Development Timeline

Chapter 9

Risk Analysis

Before working on the development phase of this system, we identified and compiled possible risks related to the project. In Table 9.1 we consider how likely the risk was to occur, how severe the consequence would be, and the impact it would have in our implementation of the system along with the consequences they hold. The impact of each risk is calculated as the $probability * severity$. Finally, we identify strategies that could be used to mitigate the issue, either by lowering the probability of it occurring or by taking precautions to minimize the severity the risk poses.

Of the risks we identified, the largest risk that became a reality was bugs. Some of the bugs that arose significantly delayed our progress, especially since our source code ultimately ended up being more than 7,000 lines of code. Searching for bugs in this large of a code base was definitely an issue. However, we were able to overcome these bugs and complete our implementation on time.

Similarly, while we had enough funding to complete our system implementation, we would have preferred if we had more funding so that we could test our system on a larger scale, as we were only able to test the scalability of it with three devices connected to one central processing hub. However, we were able to accomplish our goals with the funding we did have.

Risk Name	Consequences	Probability	Severity (0-10)	Impact	Mitigation Strategies
Bugs	• Delays progress • May cause system to stop working	0.9	4	3.6	• Prioritize bugs that appear • Design the system in a modular way
Interfacing problems with hardware	• Entire system redesign • Project may not be completed	0.2	10	2	• Research compatibility prior to development • Research backup alternative options • Continue frontend development and use mockup database if problem persists
Not meeting deadlines	• Rush to catch up would lower overall quality of work • System may not be finished	0.5	9	4.5	• Keep teammate updated on progress • Avoid procrastinating • Cut less important features
Inexperience with technologies	• Delay in progress because of time spent learning technologies • Lower quality of work because lack knowledge of details	0.7	5	3.5	• Use technologies at least 1 team member has experience with • Assign tasks according to strengths • Reference documentation when encountering issues
Insufficient funding	• Unable to implement full functionality • Unable to finish project	0.3	9	2.7	• Purchase only as much is needed to be operational before expanding scope • Cut features • Scale down project

Figure 9.1: Risk Analysis

Chapter 10

Applications

Synergy has numerous applications across the board in terms of both environmental and economic impacts. Buildings are responsible for 40% of the energy usage on the planet and up to 30% of the carbon dioxide (CO_2) [20] released. Increased energy usage monitoring could lead to reductions in energy usage and the costs of energy usage, particularly by informing consumers about their energy consumption behaviors and informing producers of patterns in their devices' energy consumption that could be altered for a smaller carbon footprint. Furthermore, it could also influence smart grid systems in an effort to modernize the currently outdated electrical infrastructure.

10.1 Use Cases

Some of the potential use cases include:

- The ability to influence behavioral patterns to reduce energy consumption
- The ability to pinpoint faulty devices for replacement
- The ability to alert on energy spikes or short-circuits
- The ability to integrate with smart home technology to run automated routines based on energy usage patterns
- The ability to identify small changes that each of us can take to reduce our overall energy footprint

10.2 Energy and Cost Reductions

Synergy allows a clear view into the metrics of different energy-demanding appliances. For example, Synergy is able to reveal the difference in performance metrics when altering your lighting settings. As shown in Table 10.1, by dimming the lights of an overhead light source by 10% in an otherwise dark room, the illuminance is only reduced by 4.71% while the energy consumption is reduced by 10%. This results in a net decrease of 10% in terms of the cost of the electricity used (assuming \$0.1.488 per kilowatt hour (kWh)) over a 30 day period. The table only measures

Table 10.1: Decrease in light brightness to kWh/Cost Ratio

Brightness	Lux	Current	kW (1h)	kW (7h/day per month)	Cost
100%	170	0.442	0.05304	11.1384	$1.168
90%	162	0.398	0.04776	10.0296	$1.052
80%	152	0.355	0.04260	8.9460	$0.938
75%	146	0.323	0.03876	8.1396	$0.854
50%	107	0.204	0.02448	5.1408	$0.539
25%	50	0.086	0.01032	2.1672	$0.227

the consumption of a single room using smart LEDs. However, incandescent or florescent light bulbs would use more energy thus increasing the costs. If this metric were taken into account in a larger setting, such as an apartment (avg. 1200sqft), household (avg. 2600sqft), or office building (avg. 19,000sqft), potentially hundreds of dollars could be saved by just dimming the lights by 10%.

Furthermore many office buildings generally require lights to be left on at night, and without smart lighting, this means that despite vastly different internal lighting demands during the night and day, the same amount of energy is consumed at any time. By switching to smart lighting, managers of these buildings would be able to configure their lights specifically for the different demands of lighting throughout the day. Likewise, since Synergy relies on open protocols, it can be interfaced with different IoT devices such as cameras, Amazon Alexa, and thermostats to detect user activity and adjust light intensity automatically.

10.3 Smart Grid

Synergy's platform is focused on providing more information to the consumer, enabling better communication about how users can save on their energy usage as well as giving utility providers increased insight into how consumers are using the energy provided. This concept has been coined as *Smart Grid*, which consists of a series of new computing and automation technologies that monitor the energy grid in order to provide improvements in energy management [22], energy costs and increased integration of renewable energy sources like solar and wind power [36]. The current power grid in the United States was designed and built over 100 years ago during a time when most residences only needed enough electricity to power a few light bulbs and a radio. "Today, an electricity disruption such as a blackout can have a domino effect – a series of failures that can affect banking, communications, traffic, and security" [36].

The Smart Grid engages in two-way communication between a consumer's home and the energy grid. Synergy is designed to seamlessly integrate with existing systems so its wide breadth of communication technologies coupled with its modularity and scalability make it a prime target for usage in a Smart Grid setting. Energy monitoring by the consumer allows users to see exactly how much power is used and how much it costs. Furthermore, consumers will be more inclined to decrease power consumption when they can see how much the energy is costing them while using it instead of a final sum at the end of the month. Smart Grid technologies also help to significantly reduce the peak demand for energy usage which is a period of time when electricity is expected to be sustained at a higher than average level [44]. Many electricity providers charge consumers based on peak demand pricing. However, the introduction of smart monitoring systems like Synergy allows for dynamic pricing [37] because energy providers can more easily predict and provide for the energy requirements of peak demand [30]. Moreover, individual contributors can better manage their individual energy usage during peak times in order to cut down on their costs.

10.4 Denial-of-Service Detection

Another possible application of Synergy is denial-of-service detection. DoS attacks attempt to flood a network with an overwhelming amount of traffic with the aim of making the services or resources provided by the network unavailable. Early stage DoS detection is extremely difficult, despite an abundance of research into detection techniques [28]. Energy monitoring introduces new opportunities to DoS detection in IoT devices. Because of the increased load on the device, more power is consumed. While the increase is too small to be noticed on the AC monitor that Synergy currently uses, it is possible that a monitor designed for reading low-power devices could be substituted and accurately detect irregular increases in energy consumption, indicating a DoS attack. This is something we hope to develop in the future.

Chapter 11

Societal Considerations

11.1 Ethical

Beyond the financial gain that individuals and businesses can gain from employing an energy monitoring platform, there is also an ethical duty of stewardship to consider. In 2017, about 34% of the total U.S. energy-related carbon dioxide (CO2) emissions were emitted by the U.S. electric power sector [32]. CO2 emissions contribute to global warming, which has the potential to be extremely disruptive to our global climate [25]. It is essential that CO2 emissions are reduced, especially given that the "impact of carbon emissions persists longer than that of nuclear waste" [29]. If the issue of greenhouse gases is not addressed now, it may not be addressed in time. Energy monitoring can play a role in minimizing greenhouse gases by giving individuals the extra push to reduce their carbon footprint - even if their intentions are to save money. Seeing their usage in real-time may also influence more users to pursue alternative forms of energy - such as solar power - which has a smaller carbon footprint than natural gas or coal, which are the two largest energy sources[31].

11.2 Social

Who should have access to your energy data? Privacy is a major social concern that we discovered while implementing *Synergy*, primarily due to the different privacy expectations in work environment than high-density home environment. Our current system only has one type of user. This is suitable in single-resident home environment, where there would be a limited number of users wanting to see their data, as well as in work environments where someone like the facilities manager would be expected to see the details of usage. But in high-density home environments, like an apartment complex, you wouldn't want your landlord to see the details of your usage - but they would likely benefit from seeing the combined usage of all residents and historical trends. So with more time, a differentiation between account types would certainly be implemented.

11.3 Political

Our hope is that communities could come together to discuss how they could lower their carbon footprint based on the data gathered from *Synergy*. However, in order for that to be effective, *Synergy* would need to be implemented on a large scale, and communities would have to decide what actions to take based on the data.

11.4 Manufacturability

The system can easily be built, as we have implemented it fully. The largest cost factor is the AC monitoring boards that are used to measure each channel, especially when it is sized up to measure more channels. However, if we were to manufacture this for production, we would look into manufacturing the AC monitoring boards ourselves to cut down on the cost of the system. Likewise, while we utilized a Raspberry Pi 3B+, however to lower the cost, a Raspberry Pi Zero with Wi-Fi capabilities could be use instead, or any other Linux-compatible hardware.

11.5 Sustainability

Our system is sustainable in two ways - first, in being a sustainable, durable system. *Synergy* is open source, which allows the software to be modified and updated as needed. Likewise, it is not dependent on any specific hardware. It can use whatever is accessible to the user, and again, can be modified or updated as needed. In this way, the system will ideally not become easily outdated and can sustain over time. Secondly, our system supports sustainability in the traditional, environmental sense, which is discussed in the following Environmental Impact section.

11.6 Environmental Impact

The primary reasoning behind developing *Synergy* is to help users build habits to reduce their carbon footprint and live more sustainably. 34% of the total U.S. energy-related CO2 emissions were emitted by the electric power sector [32]. Placing energy management solutions in the hands of consumers enables them to actively institute measures for saving energy and serve their role in lowering global carbon emissions contributing to climate change.

11.7 Usability

Environmental stewardship is a global responsibility. Because of this, everyone needs to be conscientious of their electricity usage. In order to make this possible, we designed *Synergy* with usability in mind so that usage of the system could be widespread, not limited to only those extremely proficient in technology. Our hope is that anyone could customize their dashboard in a way that is meaningful to them - whether showing kWh or cost - and seamlessly integrate the system into their lifestyle to make non-invasive changes to their daily habits.

11.8 Lifelong Learning

Designing and implementing *Synergy* gave us an unmatched hands-on engineering experience. We had to understand the technologies we used for *Synergy* more than any undergraduate class prepared us for. However, it has inspired us to delve deeper into technologies we utilized, and to stay up to date on the latest technologies in order to continue to develop the best systems possible. Learning never stops, and *Synergy* has given us the courage to take our future learning into our own hands.

Chapter 12

Conclusion

Synergy is an energy monitoring solution that is designed to be scalable for any situation in which energy monitoring could be used to decrease energy usage and energy costs. Although this paper discusses how *Synergy* was implemented using Raspberry Pi board, it could be implemented using any Linux-based hardware components. Furthermore, its system is not locked to any one environment as it utilizes libraries and frameworks that are available across most programming languages. In addition to this, *Synergy* is open-source which allows any user to adapt it to their needs or contribute to improving it. As mentioned previously, *Synergy*'s reliance on open protocols enables it to be integrated with various IoT devices, which is an ability that open-source development could expand upon.

It is also worth noting that *Synergy* can be deployed with any number of monitors, managers, and servers, just like the majority of modern web-based platforms. It can be used in residential homes, but also in larger complexes such as university, campuses, and office buildings. It can also be used by appliance producers to reduce their per device power usage requirements. Likewise, it could be implemented by local utility companies to obtain a better understanding of how electrical energy is being used in their community. There are very few limits to where *Synergy* can be used the effect that it can have on the community that uses it. Despite its passive nature, its use attempts to change behavioural patterns so as to reduce the carbon footprint that is indicative of electrical energy usage across the globe.

Bibliography

[1] AC-Monitor 2018. 2-Channel AC Current Monitor with I2C Interface. https://store.ncd.io/product/2-channel-on-board-95-accuracy-20-amp-ac-current-monitor-with-i2c-interface/. (2018).

[2] Amazon MySQL 2019. Amazon RDS for MySQL. https://aws.amazon.com/rds/mysql/. (2019).

[3] Angular 2019. Angular. https://angular.io/. (2019).

[4] Django 2019. Django: The web framework for perfectionists with deadlines. https://www.djangoproject.com/. (2019).

[5] Electricity Sales 2019. *Electric Sales, Revenue, and Average Price*. Report. US Energy Information Administration.

[6] ExpressJS 2019. Express. https://expressjs.com/. (2019).

[7] Flask 2019. Flask: web development, one drop at a time. http://flask.pocoo.org/. (2019).

[8] I2C 2018. I2C Bus, Interface and Protocol. http://i2c.info. (2018).

[9] Javascript 2018. Javascript. https://developer.mozilla.org/en-US/docs/Web/JavaScript. (2018).

[10] Jest 2018. Jest: Delightful JavaScript Testing. https://jestjs.io/en. (2018).

[11] MQTT 2019. Message Queuing Telemetry Transport (MQTT). http://mqtt.org/. (2019).

[12] MySQL 2019. MySQL Relational Database. https://www.mysql.com. (2019).

[13] Neurio 2018. Neurio. https://www.neur.io/energy-monitor. (2018).

[14] Node 2018. Node.js. https://nodejs.org/en. (2018).

[15] PostgreSQL 2019. PostgreSQL: The World's Most Advanced Open Source Relational Database. https://www.postgresql.org/. (2019).

[16] Python 2018. Python. https://www.python.org. (2018).

[17] Raspberry Pi 2018. Raspberry Pi. `https://www.raspberrypi.org`. (2018).

[18] Raspbian OS 2018. Raspbian OS. `https://www.raspbian.org`. (2018).

[19] React 2018. React: A JavaScript Library for Building User Interfaces. `https://reactjs.org`. (2018).

[20] Ahmad, Muhammad Waseem and Mourshed, Monjur and Mundow, David and Sisinni, Mario and Rezgui, Yacine. 2016. Building Energy Metering and Environmental Monitoring - A State-of-the-Art Review and Directions For Future Research. *Energy and Buildings* 120 (2016), 85–102.

[21] Apthorpe, Noah and Reisman, Dillon and Sundaresan, Srikanth and Narayanan, Arvind and Feamster, Nick. 2017. Spying on the smart home: Privacy attacks and defenses on encrypted IoT traffic. *arXiv preprint arXiv:1708.05044* (2017).

[22] Blumsack, Seth and Fernandez, Alisha. 2012. Ready or Not, Here Comes the Smart Grid! 37, 1 (2012), 61–68.

[23] Copos, Bogdan and Levitt, Karl and Bishop, Matt and Rowe, Jeff. 2016. Is anybody home? inferring activity from smart home network traffic. In *IEEE Security and Privacy Workshops (SPW)*. IEEE, 245–251.

[24] Dezfouli, Behnam and Amirtharaj, Immanuel and Li, Chelsey. 2018. EMPIOT: An Energy Measurement Platform for Wireless IoT Devices. *Journal of Network and Computer Applications* 121 (2018), 135–148.

[25] Dietz, Thomas and Rosa, Eugene A. 1997. Effects of population and affluence on CO2 emissions. *Proceedings of the National Academy of Sciences* 94, 1 (1997), 175–179.

[26] Holger Krekel. 2018. PyTest: Helps you Write Better Programs. `https://docs.pytest.org/en/latest`. (2018).

[27] Jiang, Xiaofan and Dutta, Prabal and Culler, David and Stoica, Ion. 2007. Micro Power Meter for Energy Monitoring of Wireless Sensor Networks at Scale. In *2007 6th International Symposium on Information Processing in Sensor Networks*. IEEE, 186–195.

[28] Joshi, Bineet Kumar and Joshi, Nitin and Joshi, Mahesh Chandra. 2018. Early Detection of Distributed Denial of Service Attack in Era of Software-Defined Network. In *2018 Eleventh International Conference on Contemporary Computing (IC3)*. IEEE, 1–3.

[29] Keith, David W. 2009. Why capture CO2 From the Atmosphere? *Science* 325, 5948 (2009), 1654–1655.

[30] Peck, Nathan J. 2013. *Peak Power Control With an Energy Management System*. Technical Report. Naval Post–Graduate School, Monterey CA.

[31] US Energy Information Administration. 2018. Electricity in the United States is Produced with Diverse Energy Sources and Technologies. https://www.eia.gov/energyexplained/index.php?page=electricity_in_the_united_states. (2018).

[32] US Energy Information Administration. 2018. How much of U.S. Carbon Dioxide Emissions are Associated with Electricity Generation? https://www.eia.gov/tools/faqs/faq.php?id=77&t=11. (2018).

[33] Webb, Thomas L and Benn, Yael and Chang, Betty PI. 2014. Antecedents and consequences of monitoring domestic electricity consumption. *Journal of Environmental Psychology* 40 (2014), 228–238.

[34] Scikit-Learn [n. d.]. Scikit-learn: Machine Learning in Python. https://scikit-learn.org/stable/. ([n. d.]).

[35] Sense 2018. Sense. https://sense.com. (2018).

[36] Smart Grid 2019. What is the Smart Grid. https://www.smartgrid.gov/the_smart_grid/smart_grid.html. (2019).

[37] Smart Grid Pricing 2019. Time Based Rate Programs. https://www.smartgrid.gov/recovery_act/time_based_rate_programs.html. (2019).

[38] Socket.IO 2019. Socket.IO. https://socket.io/. (2019).

[39] SQLite 2019. SQLite. https://www.sqlite.org/index.html. (2019).

[40] Cameron Steel. 2017. Challenges of Energy Management. https://www.linkedin.com/pulse/challenges-energy-management-cameron-steel/. (2017).

[41] TP-Link 2018. TP-Link. https://www.tp-link.com/us/products/details/cat-5516_HS110.html. (2018).

[42] Twilio [n. d.]. Twilio: Programmable SMS. https://www.twilio.com/sms. ([n. d.]).

[43] Vue 2019. Vue. https://vuejs.org/. (2019).

[44] Energy Watch. 2017. Understanding Peak Load and Base Load Electricity. https://energywatch-inc.com/peak-load-base-load-electricity/. (2017).

[45] Na Zhao. 2018. *Half of US Homes Built before 1980.* Technical Report.

Appendix A

Source Code

Due to its size and complexity, all of the source code is available at `https://github.com/synergy-scu`.