Boston University
Electrical & Computer Engineering
EC463 Senior Design Project

First Semester Report

Greener Living

Submitted to

Professor Anna Swan
8 St. Mary's Street
Room 828
swan@bu.edu

by

Team 18
Greener Living

Team Members

Jovany Vazquez jvazquez@bu.edu
Jason Hu hujason@bu.edu
Brian Lin linb1@bu.edu
Ali Areiqat alikat@bu.edu
Yang Hang Liu hangliu@bu.edu

Submitted: 12/12/20

Table of Contents

Executive Summary

(Authored by Brian Lin, Jovany Vazquez)

Greener Living
Team 18– Team Greener Living

With environmental issues occuring at an uncomfortable rate, it is more important than ever to reduce our individual carbon footprint. By having access to one's utility usage at all times, it becomes easier to make energy-efficient decisions without relying on the monthly bill. The final deliverable for the project consists of sensors to log relevant data about the clients household, a database to store all the information being collected, and a web application that will visualize the data for the client. Our approach is to install a monitor and sensors to measure the home’s electricity, gas, temperature, and humidity at a constant rate throughout the day. Then, using the home’s local network, it will write the data to our database, which our web application and other tools will be able to access to provide an end-to-end product. The main feature is the system's ability to update at a constant rate to provide accurate information when the client uses our web application. This way, the client can see a more detailed analysis of their energy usage than they would get from their monthly bill.

1.0 Introduction (Authored by Brian Lin)

The customer's biggest problem is the **lack** of information that is available about their utility usages in the household. As the world pushes to fight climate change, there will be a need for information that is currently not being provided by the utility companies. To add on, some companies have the capability to do this, but they stick with the bare minimum when it comes to information. For example, Eversource only shows the annual electricity costs and helps identify each specific usage after. There is simply much more information utility companies can provide to their customers to help them understand their own contributions in terms of having a greener living. It also happens to be that utility companies never update in **real-time**; they do a monthly reading and update the customer through their bills.

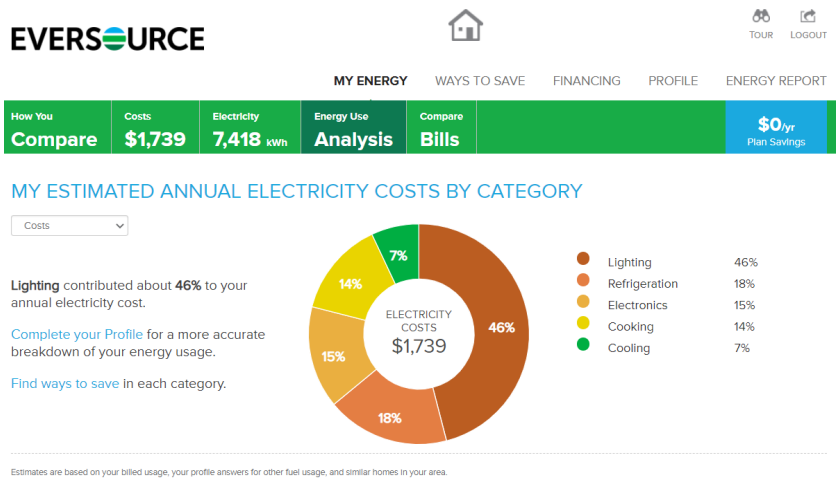


Figure 1. This is Eversource's Energy Use analysis that is provided to all customers.

Our team project's main purpose to combat this issue of the lack of utility usage information. We want to allow our customer to gain full access to their utility data in real-time through the implementation of our own sensors. We believe that there is a direct correlation between the benefit to the customer's greener living and the amount of information they will be able to get from our product.

Our general approach will be to use numerous types of sensors to grab the most information we can about the customer's house and utilities. We will help visualize all this data through our backend. The final product would be an end-to-end product from hardware to software, in which the customer will interact the most with our custom made dashboard. The customer would be able to see their utility usage in real-time and can make changes accordingly. The faster the customer is able to receive their data in real-time, the earlier they can make adjustments, which in turn can help lower their carbon footprint better as we push for a greater Greener Living.

2.0 Concept Development (Authored by Jason Hu)

Our customer's problem is regarding the fact that she is unable to know the specifics of how utilities are used within her home. The only frame of reference is the monthly utility bill, which only indicates general usage within the past month. This provides little information as to what she can do to change her behavior efficiently to save money and energy. This also makes it difficult to identify problems that may exist in her home (i.e the refrigerator using more electricity due to a broken cooling system).

From our understanding, our customer's problem can be broken down into a couple parts. First, she can only view her total energy consumption for the month, so we will need to create a method of measuring specific circuits and appliances on a regular basis. Second, she should be able to compare different statistics about her home, so our solution would need to be able to log the total electricity and gas usage as well as the solar production and display that information in an intuitive way. Last, she would need a platform to access all the information with ease, which would require some sort of application.

Our approach was to divide the solution into three parts. The first part would be the method of obtaining the necessary data, the second part would be the method of storing the data, and the last part would be the method of displaying the data. To display the data, the obvious choice would be to create a mobile or web application for which the client can simply access online to view the information. We decided to create a web application as it provided more tools and flexibility. For the most efficient method of visualizing data, we decided to display the sensor data on dashboards, which can be adjusted to a desired time period. This would require the use of programming languages such as JavaScript, HTML, CSS, as well as any additional programming frameworks to create our web application.

To store the data, we need to create a database that would be the best fit for our needs. When it comes to databases, there are two kinds to choose between: Relational and time series databases. The type of data our project will deal with will be data aggregate over time. Relational databases, although good, are inefficient with dealing with this kind of data. This is due to slower ingestion rates as the dataset size increases, which also can lead to a decrease in performance. Data retention can also become expensive. Time series datas on the other hand is optimized for data aggregated over time. Because of the higher ingestion rate and data retention policy that a time series database can provide, the approach was to select a database like InfluxDB. We also chose to store the data locally, due to additional costs that come with cloud hosting/services.

To obtain data, there were multiple approaches considered. The first approach was to create a completely software-based system where our web application would utilize services such as UtilityAPI to access our client's household information without the need to tamper with a circuit breaker or gas meter. However, this idea was abandoned due to its large financial costs implications and because it did not satisfy all the requirements regarding data collection. The second approach was to design a system where the client can manually upload her monthly bill as well as input additional information about her

time (i.e room dimensions, temperature, etc), and supplement that information with sensors for specific appliances. This idea was also abandoned because it would require too much work on the client's side, which went against the idea that the solution would be user-friendly. Thus, our final approach was to utilize a third party open-source hardware that could be installed into the client's circuit breaker to log the majority of the electrical data, with socket, temperature, and humidity sensors to provide additional information. This allows us to track all the information we need and would only require a one time installation for our client to be able to log the data at the rate we want.

The requirements for this project is to create an energy monitoring system that has sensors that logs data every five minutes which will be written to the database via WiFi. The databases and servers will be hosted on a Raspberry Pi which is connected to the local network. The database should be able to store up to a year's worth of sensor data. Data will then be read from the database and visualized on our web application.

3.0 System Description (Authored by all team members)

Our proposed solution will have three main parts; hardware, backend, and frontend. The gathering of information will happen through the hardware. The backend is responsible for storing the information and data in our chosen choice of InfluxDB and creating the representation of the data on Grafana graphs. The frontend is what the user will interact with the most; it will take those Grafana graphs and organize them into a more concise way for them to interact with.

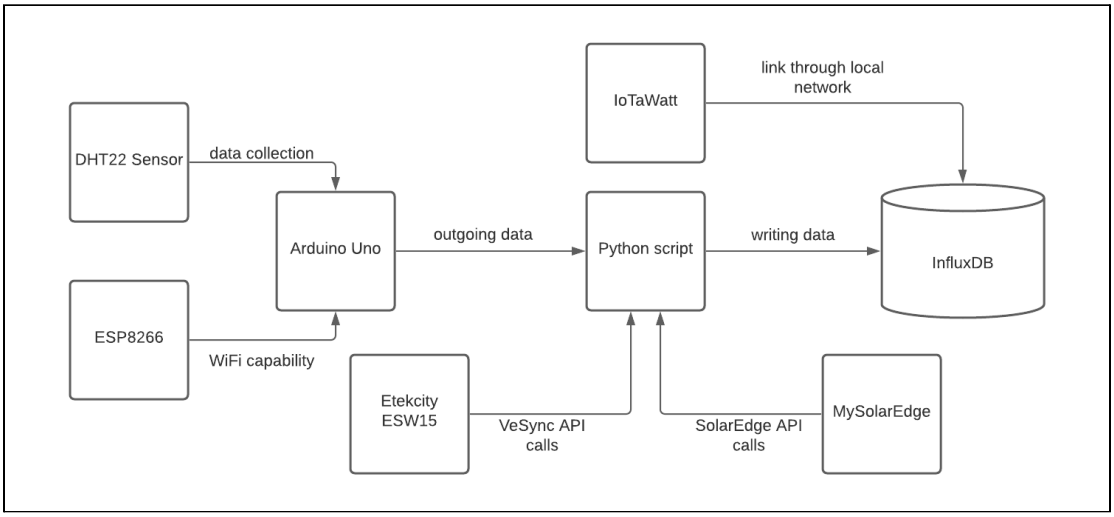


Figure 2. Hardware/Sensor system block diagram

The hardware will compose of mainly four streams being produced; temperature/humidity sensors, energy monitoring smart plugs, the IoTaWatt, and our customer's specific solar panels. The IoTaWatt will be placed on the house's circuit breaker to read the overall energy usage, while also reading the specific energy usage of each circuit, with a maximum of 14. We can also see the power of each appliance or room is being tracked on the IoTaWatt dashboard. This highlights the need for the customer to understand what each circuit in their circuit breaker composes of (e.g. Is this circuit for the whole room? Is this circuit for a single appliance?).

If the shortcomings of the customer's circuit layout do arise, this is where we would have to implement Etekcity ESW15 Wifi Energy Monitoring Smart Plug manually at each outlet for specific appliances. It has a specific python library called pyvesync, which was made for specifically VeSync compatible devices. The system's third hardware component of the temperature/humidity sensor will be a self-made combination Arduino UNO, the DHT22 temperature-humidity sensor and the ESP8266 WiFi microchip. This design will allow us to record the temperature/humidity every 2 seconds at minimum, while maintaining an accuracy of ±0.5°C for temperature and ±2% for humidity. The customer already has mySolarEdge, which has an API where we can call from. It also already has a graphical interface for public use if the customer allows it.

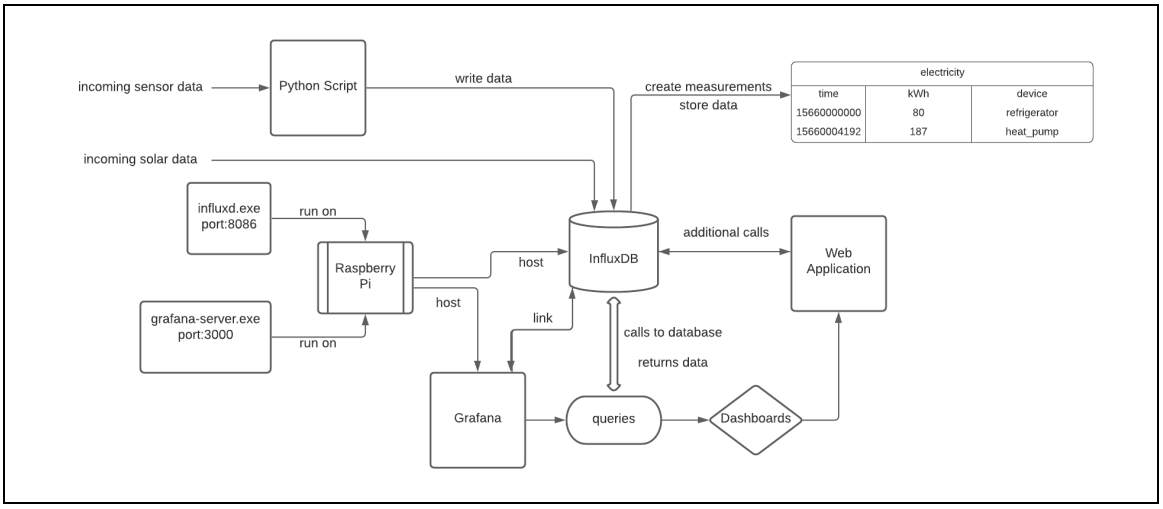


Figure 3. Backend system block diagram

The backend consists of two main components; InfluxDB and Grafana. Both components will have its server hosted on a Raspberry Pi which connects to the client's local network. These servers can run on local ports. The Raspberry Pi will have a microSD card with 128GB of storage, enough to store 365 days worth of data, up to 10,000 rows of data per day. All incoming sensor data can be written to the database using a combination of Python and influxQL. Grafana can then have its datasource be directly linked to the database, which can quickly be done since our servers are running on the same local network. Grafana's user interface allows us to easily query the database to change the measurement, values, tags, and other key fields that we want shown on a dashboard. By changing the custom.conf file for the Grafana to grant our web application permission and allowing embedding for the dashboards, we can implement the dashboards via embedding links within our code.

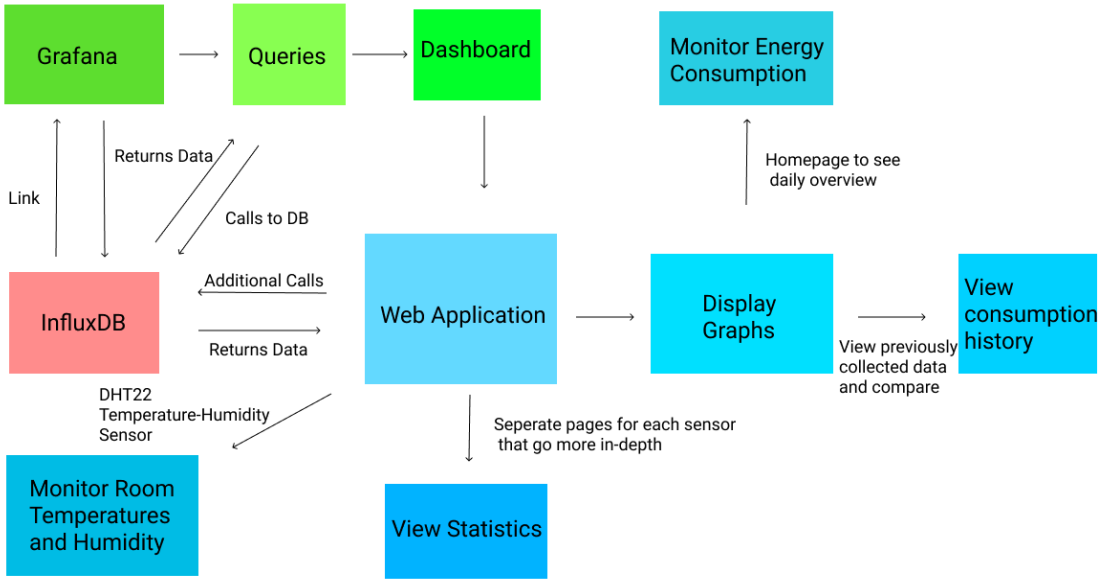


Figure 4. Front-end System Block Diagram

The front-end consists of HTML, CSS, and JS files that make up what the client will see on their side. From their point of view, they will start at the home page that shows all the other pages they can go to, in order to access the information they are seeking. This ranges from electricity, to gas, and even solar. All of these pages link to each other meaning they can be accessed from any other page. Each page will offer different information in several forms. For example, the Energy Usage page will show a graph in real time, while the Contact Us page will show simple text. All of the data information will be coming from the back-end side of the website meaning InfluxDB and Grafana. The website will be easy to navigate and very user friendly. The list of pages that will be included are as follows: Homepage, Energy, Gas, Solar, Contact Us, and Tips. All of the page setups, including text and images, will be done in HTML, while the styling of the pages and text will be done in CSS code. JavaScript will come in for the page to be more interactive and user-friendly.

```
//Example code for how one of our sensors will communicate with the pi

    const client = dgram.createSocket('udp4'); //communicates to pi through UDP protocol
    msg = reading;
    client.send(msg, 1131, '10.0.0.145', (err) => { //sends the reading to the pi with that ip address
        client.close();


//After receiving the message, the PI would parse the data and send it to a database

server.on('message', (msg, rinfo) => {
  msg = msg.toString();
  console.log(`server got: ${msg} from ${rinfo.address}:${rinfo.port}`);
    var parts = msg.split(",");
    myobj = { ID: parts[0], reading: parts[1], time: Date() }; //parsing for database
    db.collection(parts[0]).insert(myobj); //uses the ID to look for the database collection corresponding to what sensor it is and then inserts data

});

//Grafana will connect to the database to create graphs

dbo.collection("Electric").find{};//gets data from the database
var chart_elec = new chart(Grafana){
    title: {
        text: "Electricity"
    },
        data: [{
        name: "Electricity",
        type: "line",
        dataPoints: electric_data //creates graph with these parameters using the data
    }]
}

//After Grafana creates the craphs, we can display them onto our website using javascript

<iframe
    src="https://snapshot.raintank.io/dashboard-solo/snapshot/y7zwi2bZ7FcoT1B93WN7yWO4aMiz3pZb?from=1493369923321&to=1493377123321&panelId=4"
    width="650" height="300" frameborder="0">
</iframe>
```

Figure 5. Example Pseudocode

The first step of our system is for our sensors to send data to our Raspberry Pi. This can be done through UDP or TCP protocols. After receiving the data, our Pi can parse the data into our database. Grafana will access our database and use the data in order to create graphs. Our website can then use and display those charts.

4.0 First Semester Progress (Authored by all team members)

Due to problems with our discussion with our client, we finalized our deliverables a bit late into the semester, so we have mainly focused on setting up the front-end and back-end of our website. In addition, we have chosen and designed the sensors we will be using along with the IoTaWatt.

4.1 Front-End

The website's Front-End is still under development but it has some stylizing and structure that is similar to our desired goal. As designing a website is new to us, we are still learning and applying as we go. Our goal is to have it user friendly and easy on the eyes. We currently have our logo as well as the desired sections on the website including Electricity, Gas, Solar, Tips, and Contact Us. The rest is filler to get to know and understand the layout and process of building the website. For example, there is a long block of code that is irrelevant to the information that will be provided in the final product. The purpose of it was to understand the placement and capability of the functions (<head>,<body>, etc.) in HTML. We also have a green color block on the header for testing purposes to get familiar with it. Lasly, we have several images that are similar to our desired icons placed next to its respective section (i.e. a thunderbolt icon next to Electricity).The website only has HTML and CSS that makes up its code and how it looks currently. We are currently working on learning and implementing the interactive part of the website as well so that the client can browse through it easily.

4.2 Back-End

The backend of this project is mainly composed of InfluxDB (a times series database) and Grafana (an open source analytics and interactive visualization application). Brian was responsible for the InfluxDB and Jason was responsible for the Grafana part of the backend. We collaborated together to work on the integration of the link in between InfluxDB and Grafana because we were both interested in it. Initially, we had set up a local instance of InfluxDB on our own separate computers on localhost:8086. When we installed Grafana, it took over localhost:3000. At first, we used sample data that was provided to us because we wanted some substance in our local databases. This data could be found here at this link. To put this sample data into perspective, it recorded several water features (water depth, acidity levels, water temperature, average temperature, and etc.) of two different lakes. The frequency of the recording was every six seconds and is **historical** data from August 18th, 2019 to September 18th, 2019. We created the Grafana dashboards from all the sample data we received and showed the capabilities Grafana had in terms of customization.

For the first deliverable testing of the backend, we showed our local host of the database and our Grafana dashboards with sample data. The testing went smoothly and there were no hiccups that came along the way. However, Professor Pisano did have one major concern; our testing did not show that the backend was capable of accepting any sort of real time data. We understood that this was a justifiable issue as our final product did anticipate real-time data being displayed. To resolve the issue, Jason created a python

script that would generate a fake “real-time” data point (just used randint in the code), that would produce every 10 seconds. The code is posted below:

```
#!/usr/bin/env python3
from influxdb import InfluxDBClient
import uuid
import random
import time
client = InfluxDBClient(host='localhost',
port=8086)
client.create_database('test_data')
measurement_name = 'electricity_usage'
number_of_points = 1
location_tags = [ "fridge",
"washer",
"main_power",
"heat_pump",
"lights",
"stove"]

while True:
   time.sleep(10)
   data_end_time = int(time.time() *
1000) #milliseconds
   data = []

data.append("{measurement},location={l
ocation} voltage={voltage}i {timestamp}"

.format(measurement=measurement_na
me,

location=random.choice(location_tags),
            voltage=random.randint(0,50),
            timestamp=data_end_time))
   current_point_time = data_end_time
   for i in range(number_of_points-1):
      current_point_time =
current_point_time -
random.randint(1,100)

data.append("{measurement},location={l
ocation} voltage={voltage}i {timestamp}"

.format(measurement=measurement_na
me,

location=random.choice(location_tags),
```

```
                voltage=random.randint(0,50),

timestamp=current_point_time))
   client_write_start_time =
time.perf_counter()
   client.write_points(data,
database='test_data',
time_precision='ms', batch_size=10000,
protocol='line')
   client_write_end_time =
time.perf_counter()
   print("Client Library Write:
{time}s".format(time=client_write_end_ti
me -client_write_start_time))
```

We think that this does show we do in fact have the capability to produce real-time data and it solves one of Professor Pisano's concerns. We believe that it is a major accomplishment of the backend team to achieve the real-time feat, we have found a way to input real-time data through a python script.

4.3 Sensors

Since the IoTaWatt could not be used or tested until it is fully installed in the client's home, we focused on the additional sensors needed for the project.

First, the energy monitoring plugs. The different plugs we could use have been researched, and the Etekcity ESW15 has been settled on. This plug has been chosen because it is the easiest to use, as after connecting to it, the data can be collected using an api library pyvesync. Other good energy monitoring plugs like the Top Greener WiFi Smart plug have been considered; however, they have not been chosen because the process of setting them up requires certain device restrictions to be bypassed, by flashing the device, to get the data to the website. As such, the simpler approach with ESW15 was chosen.

Finally, the temperature and humidity sensor. Various temperature and humidity sensors have been researched and we have chosen to build our custom sensor which we have finished designing (its schematic is in the appendix). It's main components are an Arduino UNO, the DHT22 temperature-humidity sensor and the ESP8266 WiFi microchip. This design will allow us to record the temperature/humidity every 2 seconds at minimum, while maintaining an accuracy of ±0.5°C for temperature and ±2% for humidity.

5.0 Technical Plan (Authored by all team members)

Task 1. HTML
The plan is to have a good understanding of HTML, CSS, and JS by the start of the next semester on January 25, 2021. Although implementing the new skills over the winter is not required, it will be beneficial to have done so. The structure of the HTML portion, meaning structure, spacing and layout, should be completed by January 25, 2021 as well. This means that All of the information needed on the front-end of the website needs to be implemented in a correct format similar to that of our Figma Design. Lead: Jovany.

Task 2. Local database
A Raspberry Pi will be designed and tested to store a local instance of the Influx database, which will then be installed in the client's home. The database should meet the specifications for memory. The database should be tested with 400,000 rows of dummy data. Lead: Brian Lin; Assisting: Jason Hu.

Task 3. Temperature and Humidity Sensor
A temperature and humidity sensor with a +/- 1% accuracy and 5 second measurements interval has been designed and will be fabricated, and tested with reference to an electronic thermometer and humidity sensor. In addition, the sensor will be tested to make sure it sends the data properly to the website. Lead: Ali.

Task 4. CSS
All the CSS portions which include coloring and fonts should be implemented by February 20, 2021. Similarly to the HTML task, this should be implemented as close to our Figma Design as well. Of course, it is still subject to change if we believe any design changes necessary or complementary to the overall look of the website. Lead: Jovany.

Task 5. IoTaWatt
An IoTaWatt Open Wifi Electric Power Monitor will be installed in our client's home. The IoTaWatt has 14 input channels; therefore, allowing for the measurement of 14+ circuits. After installation, the IoTaWatt will need to be connected to our database and will need to be tested with large amounts of data sustained over a minimum of 12 hours. Lead: Yang Hang; Assisting: Ali.

Task 6: Energy Monitoring Plugs
The system to use 15A energy monitoring plugs has been designed, and will be fabricated and tested. It shall transmit energy usage information every 5 seconds. It will be tested for accuracy of data using an electricity usage monitor, for proper data sending time intervals by timing it and for proper integration with the website, by checking if the website receives the data. Lead: Ali.

Task 7. JavaScript
The JS portion should be implemented by March 15, 2021. The website should be interactive by the 15th and have any other JS implementations deemed necessary. Such implementations include, but are not limited to, widgets and animations. Lead: Jovany.

Task 8. Integration of Front-End and Back-End
The integration of Back-End and Front-End should be completed by April 1, 2021 after the rest of the Back-End is completed. This will require somewhat new knowledge but with experience in developing the website it will not impose an issue as we already know how we will try to implement it. We are planning on using iframe code to connect and show the graph from the Back-End to the Front-End. Lead: Jovany; Assisting: Brian Lin, Jason Hu.

Task 9. Grafana dashboards
A Raspberry Pi will be designed and tested to store a local instance of the Grafana, which will then be installed in the client's home. Grafana should be properly linked with the Influx database. Dashboards should be created for each utility/appliance tracked. Dashboards should also be adjusted to the proper time-scale and updated properly. This will be tested using dummy real-time-data being sent from a Python script for at least 12 hours continuously. Lead: Jason Hu; Assisting: Brian Lin

6.0 Budget Estimate (Authored by all team members)

In this section you discuss WHAT ARE THE BUDGET IMPLICATIONS AND CONSTRAINTS for your effort.

Item	Description	Quantity	Cost
1	Raspberry Pi Zero W	1	$15.00
2	MicroSD Card with NOOBS	1	$30.00
3	IoTaWatt	1	$139.00
4	IoTaWatt USB PowerSupply	1	$10.00
5	AccuCT 100A x 16mm split-core	6	$96.00
6	9V AC Reference Transformer 120V US Plug	1	$15.00
7	Breadboard and wires pack	2	$19.98
8	Arduino Uno Rev3	2	$46.00
9	DHT22 temperature-humidity sensor	2	$19.90
10	ESP8266 SMT Module - ESP-12F	2	$13.90
11	Sparkfun Logic Level Controller - Bi-Directional	2	$5.90
12	Etekcity ESW15 WiFi Energy Monitoring Smart Plug 2 pack	1	$21.99
	Total Cost		$432.67

Our biggest cost in our budget will be the IoTaWatt, where approximately is about half of our cost. This cost can be justified because it will solve many of the problems the customer faces and is open-source, so we can build upon it. As we can see, we end up with around $570 unspent on out of our full project. The biggest constraint will be having the right amount of current transformers and needing backup for those, so the cost may compile quickly.

7.0 Attachments (Authored by all team members)

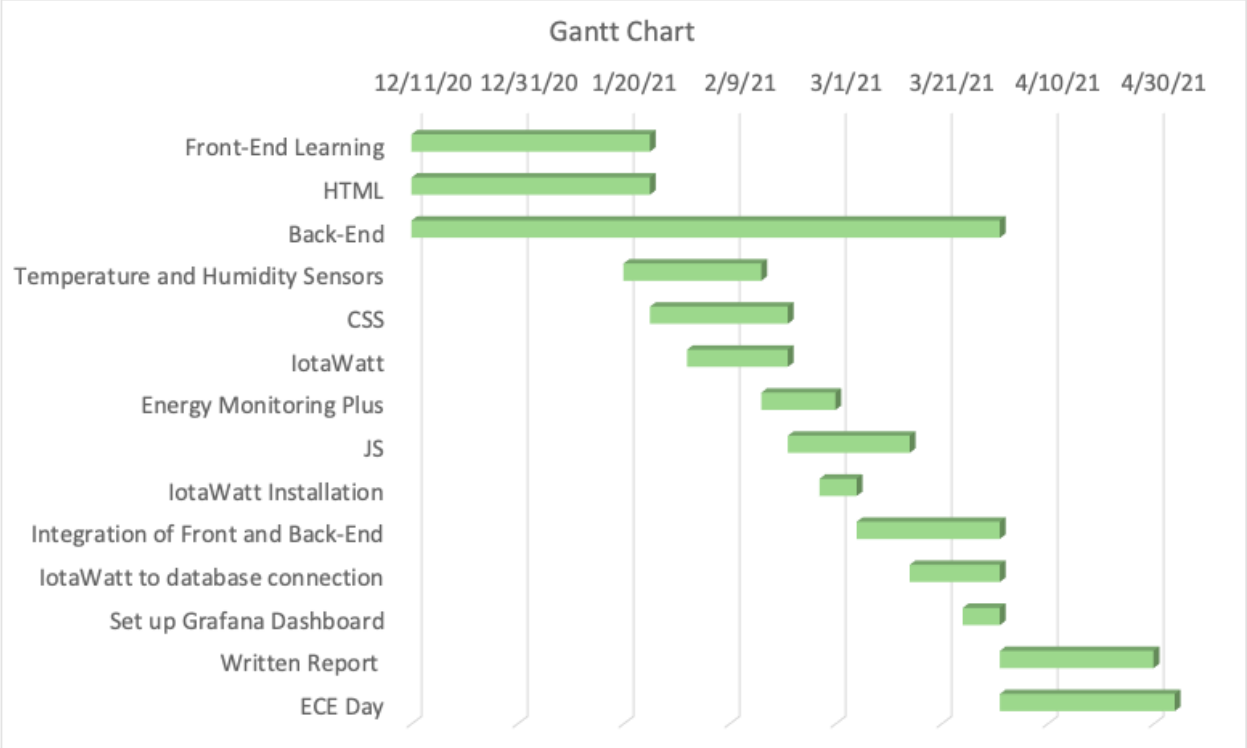
7.1 Appendix 1 – Engineering Requirements

Team #: 18 Team Name: Greener Living

Project Name: Greener Living

Requirement	Value, range, tolerance, units
Power	Input 100-240V AC Output +5 DC 600mA 120V Plug
Database Storage Duration	1 year, 365 days
Database Storage Size	128 GB
Ingestion Rate	at least 10,000 rows per day
Reading Frequency	every 5 minutes
Temperature Range	-40 to 125 ° C
Temperature Error	+- 0.5 ° C
Humidity Range	0 - 100%
Humidity Error	2 - 5%

7.2 Appendix 2 – Gantt Chart



7.3 Appendix 3 – Other Appendices

Ali Areiqat Email: alikat@bu.edu Phone:413-285-6593

Ali is a senior at Boston University from West Springfield Massachusetts. He is studying Computer Engineering while going through the pre-med pathway. He mainly spends his free time reading. He hopes to spend a gap year working before entering medical school after graduating.

Yang Hang Liu Email: hangliu@bu.edu Phone:508-524-6676

Yang Hang Liu was born in Fujian China on June 29, 1999. At the age of 3, he and his family moved to the United States. He first lived in Cape Cod, Massachusetts but moved to Tennessee in Middle School. He moved back to Massachusetts in high school, where he gained interest in the maths and sciences and wanted to become an engineer. He is currently pursuing that goal at Boston University.

Jovany Vazquez Email: jvazquez@bu.edu Phone: 857-249-9947

Jovany Vazquez is a senior at Boston University studying Computer Engineering. He grew up in Boston, MA where he attended Fenway High School while it was still located in front of Fenway Park, not too far from Boston University. When he is not studying, he spends his time gaming, watching TV, and drawing. His current goal is to graduate and get a job within the software engineering field while also starting a business on the side.

Brian Lin Email: linb1@bu.edu Phone: 347-552-2216

Brian Lin grew up in Queens, New York. He attended the Bronx High School of Science, and is currently a student at Boston University, pursuing a degree in Computer Engineering. His interests include music, cooking, and games. His career goal is to work in the software engineering field, eventually working as a full-stack developer.

Jason Hu Email: hujason@bu.edu Phone: 917-635-6099

Jason Hu is from Staten Island, New York. He attended Staten Island Technical High School. He is currently a senior Computer Engineering student at Boston University. His main interests include gaming and sleeping. His academic interests lie in software engineering, with a more specific focus on backend applications.

Crowley, Noah. “Getting Started with Python and InfluxDB.” *InfluxData*, 11 Apr. 2018, www.influxdata.com/blog/getting-started-python-influxdb/.

gigafide. “Create A Free Website From Scratch.” *YouTube*, YouTube, 13 Mar. 2009, www.youtube.com/watch?v=NG15HHwAtwU.

“InfluxDB 1.8 Documentation.” *InfluxDB OSS 1.8 Documentation*, docs.influxdata.com/influxdb/v1.8/.

“Install on MacOS.” *Grafana Labs*, grafana.com/docs/grafana/latest/installation/mac/.

“IoTaWatt Documentation¶.” *IoTaWatt Documentation - IoTaWatt 02_03_20 Documentation*, docs.iotawatt.com/en/02_05_11/.

LearnToProgramDotTV. “How To Build A Website From Scratch.” *YouTube*, YouTube, 18 Feb. 2020, www.youtube.com/watch?v=D9yAzfR3deQ.

Persen, Todd. “How to Send Sensor Data to InfluxDB from an Arduino Uno.” *InfluxData*, 21 Feb. 2018, www.influxdata.com/blog/how-to-send-sensor-data-to-influxdb-from-an-arduino-uno/.