First Semester Report
Greener Living

A.Areiqat, J.Hu, B.Lin, YH.Liu, J.Vazquez

Abstract (Authored by Brian) - With the rise of many environmental issues in the 21st century, 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 solely on the monthly bill. The final deliverable for the project consists of sensors that log relevant data about the client's household, a database that stores the information collected, and a web application that visualizes the data using graphs and charts. 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, they 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 of this project is the system's ability to update at a constant rate, providing accurate information whenever the client uses the web application. As a result, the client can see a detailed analysis of their energy usage than they would get from their monthly bill.

1 Introduction

(Authored by Brian)

With the concerns about climate change growing every day, the need for major changes in society is becoming more apparent as the frequency of natural disasters, droughts, and fires increases. Despite all the tangible evidence of dangers related to climate change, there is a surprising lack of action to address this issue, from world leaders to the average citizen alike. According to a survey conducted by the AP-NORC, 68% of Americans would not pay $10 more a month on their electricity bills, even if the $10 fee would go toward combating climate change. However, the average American is not fully to blame. Eco-friendly products and decisions are often more expensive than sticking to the status quo, and requires large changes in behavior that not everyone can afford.

Currently, around 80% of energy used by Americans comes from fossil fuels, a major contributor to the increase in greenhouse gas emissions. Furthermore, a large portion of energy use is wasted, which not only produces more greenhouse gases, but also raises the prices of utility bills each month.

Thus, rather than having average homeowners directly spend money to help combat climate change, a better approach would be to help homeowners make energy efficient decisions that will not only reduce their carbon footprint, but their utility bill as well. Research shows that without innovations in energy efficiency, energy consumption in the U.S would be 60% higher than it is today. By obtaining data at a faster frequency, Greener Living takes advantage of these innovations by providing an energy monitoring system that will give its user a more detailed insight into how energy is being used in their home rather than simply telling the user how much energy is used each month. By allowing homeoweners to check their updated information at their convenience, they can reduce energy wastage and save money at the same time.

The customer's biggest problem is the lack of information that is available about their utilities' usage in the household. As the world pushes to fight climate change, there will be a need for information that is not provided by utility companies. Additionally, even though some companies have the capability to do this, they stick to the bare minimum when it comes to providing information. For example, Eversource only shows the annual electricity costs and helps identify each specific usage after. There is much more information utility companies can provide to their customers to help them understand their energy usage and wastage, and their carbon footprint. It just so happens that utility companies never update provided information in real-time. Instead, they do a monthly reading and update the customer through their bills.

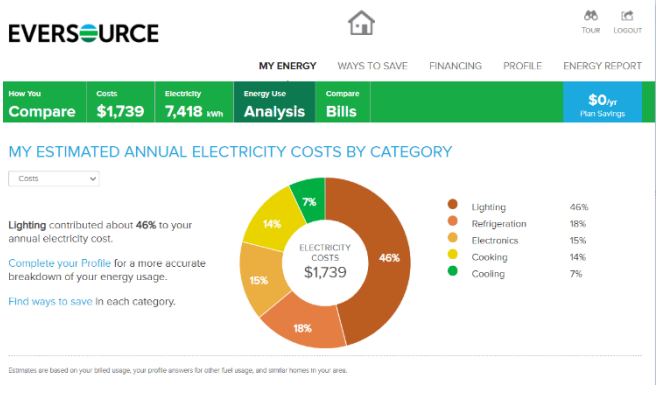


Fig. 1. This is Eversource's Energy Use analysis that is provided to all customers. It gives rudimentary analysis of the customer's annual energy usage and cost.

This project's purpose is to combat the lack of utility usage information. We want our clients to gain full access to their utility data in real-time through the implementation of our own energy monitoring system. We believe that the more information clients get from this project, the more they will attempt to live more greenly.

Our approach will be to use several 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 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, reduce their montly costs and help push for a greater, greener living.

2 Concept Development

(Authored by Jason)
Our customer's problem is that she is unable to know the specifics of how utilities are used within her home. The only frame of reference she has 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 (for example, the refrigerator using more electricity due to a broken cooling system).

Our customer's problem can be broken down into two 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's utility usage. Our solution would need to be able to log the total electricity, gas usage and solar production, and display that information intuitively on a platform where she can 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 would be obtaining the necessary data, the second would be storing it, and the last part would be 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 provides more tools and flexibility. To efficiently visualize data, we decided to display the sensor data on dashboards, which can be adjusted to a specific time period. This requires the use of programming languages such as JavaScript, HTML, CSS, as well as additional programming frameworks that would help create our web application.

To store the data, we need to create a database that would be best fit for our needs. There are two kinds of databases to choose from: relational and time series databases. The type of data our project will deal with will be data aggregated 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 can lead to decreases in performance. Additionally, data retention can become expensive. Time series databases on the other hand are optimized for data aggregated over time. Because of the higher ingestion rate and data retention policy a time series database provides, our approach was to select a database like InfluxDB. Moreover, chose to store the data locally, due to additional costs that come with cloud hosting/services.

To obtain the data, there were multiple approaches considered. The first 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 client's 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 house (i.e room dimensions, temperature, etc), and supplement that information with sensors for specific appliances. This idea was 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 access 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 log data every five minutes. The data collected will be written to a database via WiFi. The database and server 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. Finally, data will be read from the database and visualized on our web application.

3 System Description

(Authored by all team members)
Our proposed solution will have three main parts; the hardware, backend, and frontend. The hardware gathers the information. The backend stores the data in our chosen choice of InfluxDB and creates 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 the user to interact with.

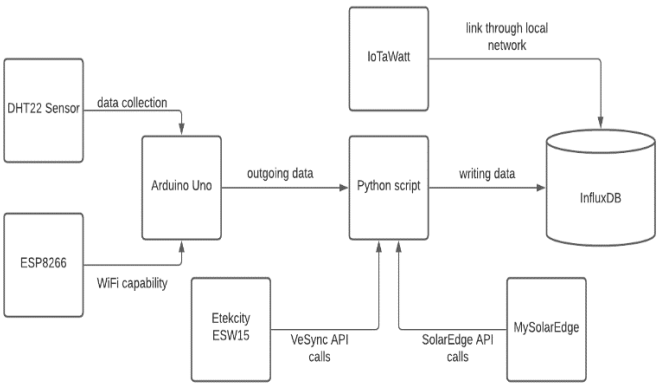


Fig. 2. Hardware/Sensor system block diagram. This shows the energy collected from our sensors and its flow into project's backend.

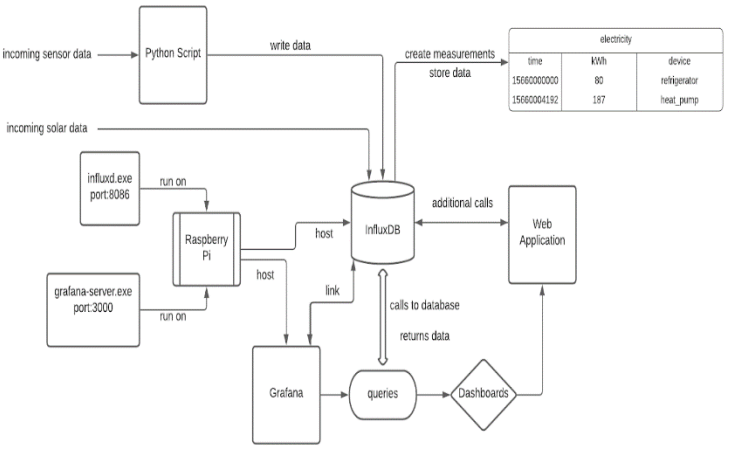


Fig. 3. Backend system block diagram. It shows where the data from our hardware components is intercepted, its flow and how it is processed, stored and manipulated.

The hardware is composed of four components that provide data: 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 a maximum of 14 circuits. We can also see the power of each appliance or room being tracked on the IoTaWatt dashboard. This highlights the need for the customer to understand what each circuit in the circuit breaker is composed of (e.g. Is this circuit for the whole room? Or is it for a single appliance?).

If there are shortcomings in the customer's circuit layout, we would implement the Etekcity ESW15 Wifi Energy Monitoring Smart Plug manually at each outlet for specific appliances to compensate. It has a python library called pyvesync, which is made for VeSync compatible devices. The system's third hardware component is the temperature/humidity sensor, which will be a self-made combination of 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 final hardware component is the client's solar panels. The client already has mySolarEdge (an app that collects and displays the solar panel's data), which has an API that we can call from. Also, the API has a graphical interface for public use if the customer allows it.

The backend consists of two main components; InfluxDB and Grafana. Both components will have their servers hosted on a Raspberry Pi, which connects to the client's local network. Additionally, these servers can run on local ports. Moreover, 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 data source 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.

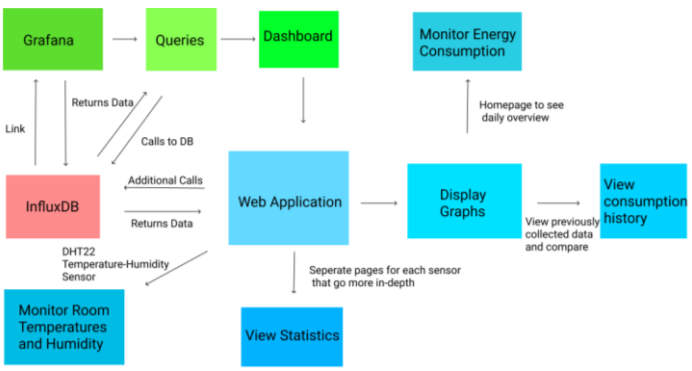


Fig. 3. Front-end system block diagram. It shows a general overview of the project's components with a focus on the displayed output

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. These pages will display electricity, gas, and even solar usage. 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 will be coming from InfluxDB and Grafana, the website's backend. 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 be used to make the page 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>
```

Fig. 5. Example Pseudocode: The first step of our system is for our sensors to send data to the Raspberry Pi. This can be done through UDP or TCP protocols. After receiving the data, our Pi can parse the data into the 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 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 a 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).Currently, the website's code is made up of only HTML and CSS. We are working on learning and implementing JS, the interactive part of the website 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 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 it's from August 18th, 2019 to September 18th, 2019. We created the Grafana dashboards from the sample data we received and showed the capabilities Grafana had in terms of customization.

For the first deliverable testing of the backend, we filled 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 (using radint) every 10 seconds.

```
#!/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) #millisec-
onds
   data = []
   data.append("{measurement},location={location}
voltage={voltage}i {timestamp}"
            .format(measurement=measurement_name,
            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 - ran-
dom.randint(1,100)
      data.append("{measurement},location={location}
voltage={voltage}i {timestamp}"
               .format(measurement=measurement_name,
               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=cli-
ent_write_end_time -client_write_start_time))
```

Fig. 6. This code was used to stimulate our website's functionality by generating fake real-time data, and testing whether it will be stored and displayed properly. The tested proceeded smoothly and we have a functionaing backend.

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 process the real-time data.

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 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 bypassing certain device restrictions 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 Technical plan

(Authored by all team members)
Each section here is a task we need to complete before we finish the project.

5.1 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.

5.2 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.

5.3 Temperature and Humidity Sensor

A temperature and humidity sensor with a +/- 1% accuracy and a minimum 5 second measurements interval has been designed and will be fabricated, and tested with reference to an electronic thermometer and humidity sensor by February 14, 2021. In addition, the sensor will be tested to make sure it sends the data properly to the website also by February 14.
Lead: Ali.

5.4 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.

5.5 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.

5.6 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. All of this should be done by February 28, 2021.
Lead: Ali.

5.7 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.

5.8 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.

5.9 Grafana Dashboards JavaScript

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. This will be completed by March 28, 2021.
Lead: Jason Hu; Assisting: Brian Lin

6 Budget Estimate

(Authored by all team members)

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. Additionally, we are estimating that we will need only two energy monitoring plugs; however, depending on the functionality of the IoTaWatt that number might change and we cannot know if it will until the IoTaWatt is installed at the client's home.

7 Attachments

(Authored by all team members)

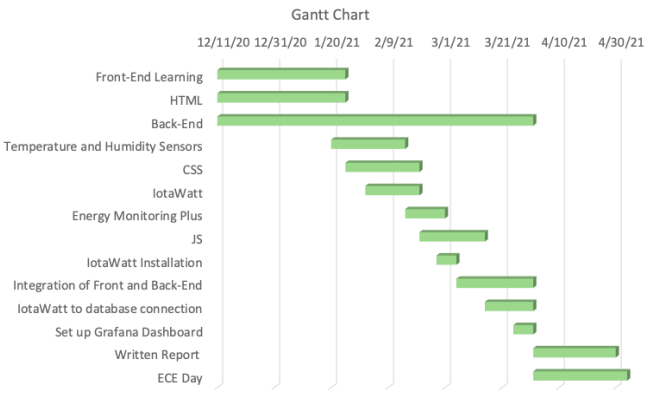
7.1 Appendices

Appendix A – Engineering Requirements

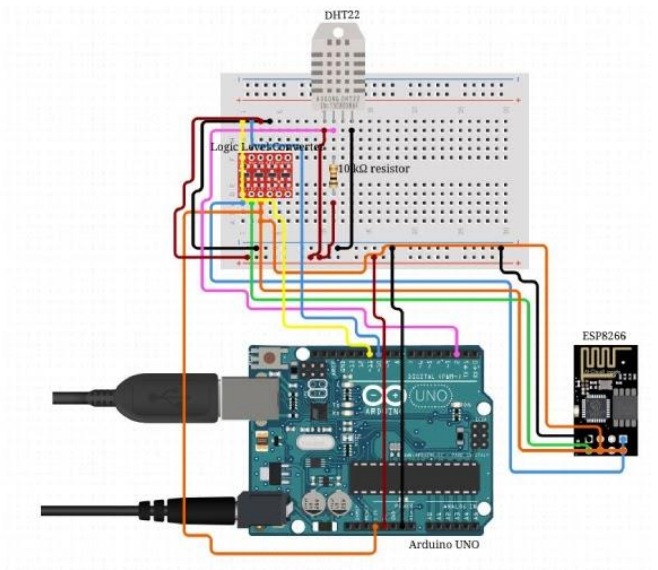
Requirement	Value, range, tolerance, units
Power	Input 100-240V AC Output +5 DC 600mA 120V Plug
Database Storage Duration	1 year, 365 days

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%

Appendix B – Gantt Chart



Appendix C – Temperature and Humidity Sensor Schematic



REFERENCES

[1] Crowley, Noah. "Getting Started with Python and InfluxDB." InfluxData, 11 Apr. 2018, www.influxdata.com/blog/getting-started-python-influxdb/.

[2] Gigafide. "Create A Free Website From Scratch." YouTube, 13 Mar. 2009, www.youtube.com/watch?v=NG15HIHwAtwU.

[3] "InfluxDB 1.8 Documentation." InfluxDB OSS 1.8 Documentation, docs.influxdata.com/influxdb/v1.8/.

[4] "Install on MacOS." Grafana Labs, grafana.com/docs/grafana/latest/installation/mac/.

[5] "IoTaWatt Documentation." IoTaWatt Documentation - IoTaWatt 02_03_20 Documentation, docs.iotawatt.com/en/02_05_11/.

[6] LearnToProgramDotTV. "How To Build A Website From Scratch." YouTube, 18 Feb. 2020, www.youtube.com/watch?v=D9yAzfR3deQ.

[7] 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/.

ACKNOWLEDGMENT

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. After graduating, he hopes to spend a gap year working before entering medical school.

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 during Middle School. He moved back to Massachusetts during 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.