Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

by
Team 18
Greener Living

Team Members

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

Submitted: 04/13/2021

User Manual

Table of Contents

Executive Summary

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 user'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, 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 user uses the web application. As a result, the user can see a detailed analysis of their energy usage than they would get from their monthly bill.

1 Introduction

The user's biggest issue is the **lack** of information that is available about their utility usages in the household. Utility companies give the bare minimum of information about energy consumption and never update in **real-time**; they do a monthly reading and update the user through their bills. Our project's purpose is to combat this issue of lack of utility usage information. We want to allow our user to gain full access to their utility data in real-time through the implementation of our device. Our user wants to be able to easily monitor their utility so that they can figure out what changes they need to make to reduce their energy consumption if need be.

From the hardware side, the device uses a Raspberry Pi to run the Raspberry Pi OS Lite. It utilizes a 16GB microSD card to be able to install InfluxDB and Grafana for transferring data. The IoTaWatt component can read and send data through the Raspberry Pi and update our database. The 2 DHT22 sensors can read temperature and humidity simultaneously while the Etekcity ESW15 smart plugs can read energy data from a wall socket. All of the data is funneled through the IoTaWatt and Raspberry Pi to the web application. (For troubleshooting purposes, the hardware can be reset and fixed by unplugging and reconnecting the components in the order which is described in Section 3.2)

As for the software, the web application utilizes InfluxDB and Grafana, which store and retrieve data respectively, and then produces and updates several graphs throughout the web application in real time. With the help of queries to store the graphs, the web application uses iframes to display the graphs. The web application is developed in React, which utilizes HTML, Javascript, and CSS all together to make it a seamless and easy to use web application. We included a navigation bar, drop down menus, and buttons for the user's freedom to modify and view the information how they want to, such as hourly, daily, weekly, and even monthly data.

Our approach helps the user view the most information they can about their utility usage. The user is able to see their utility usage in real-time and can make changes to their lifestyle accordingly. The faster the user 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.

This User Manual describes the specific implementation of all the components, as well as how they work together to produce reliable and readable data. The rest of the manual includes an installation guide, as well as information regarding the data being displayed and what the user can do on the web application. Aside from how to use the device and application, the user manual will also touch upon any issues that may arise from the use of our product and warnings about safety and usage. Lastly, the manual will provide information about engineering standards and the cost of the devices that were used in order to make this device. There is also a bit of information about the team that made this device, so go ahead and read on to learn more about the project and us!

2 System Overview and Installation

2.1 Overview block diagram

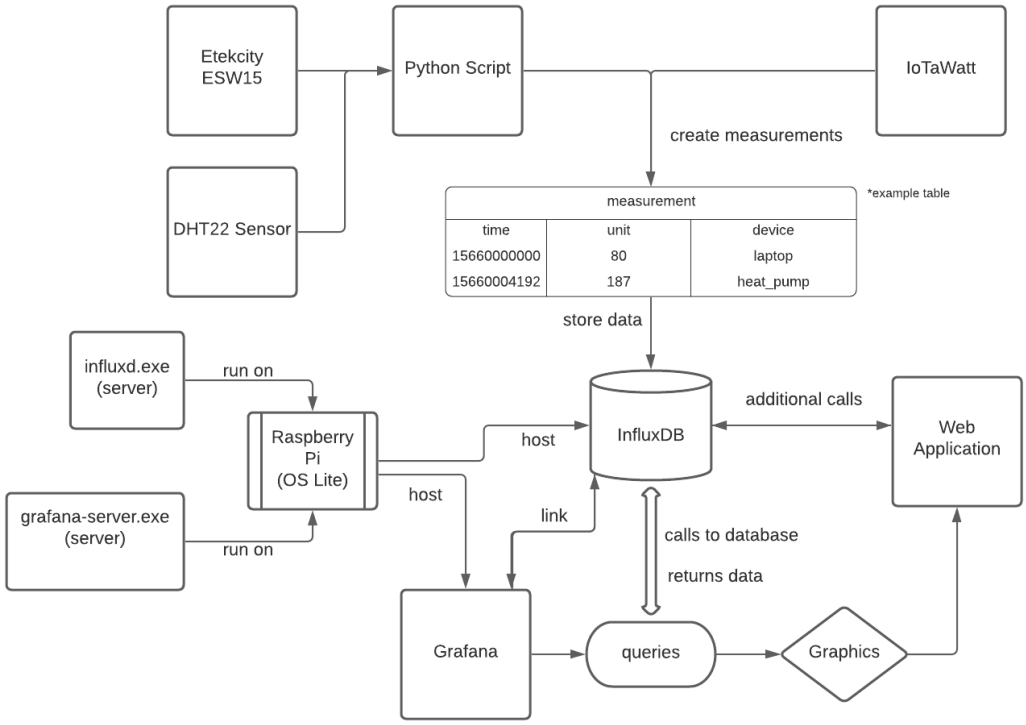


Figure 2.1 System Overview Block Diagram

2.2 User interface.

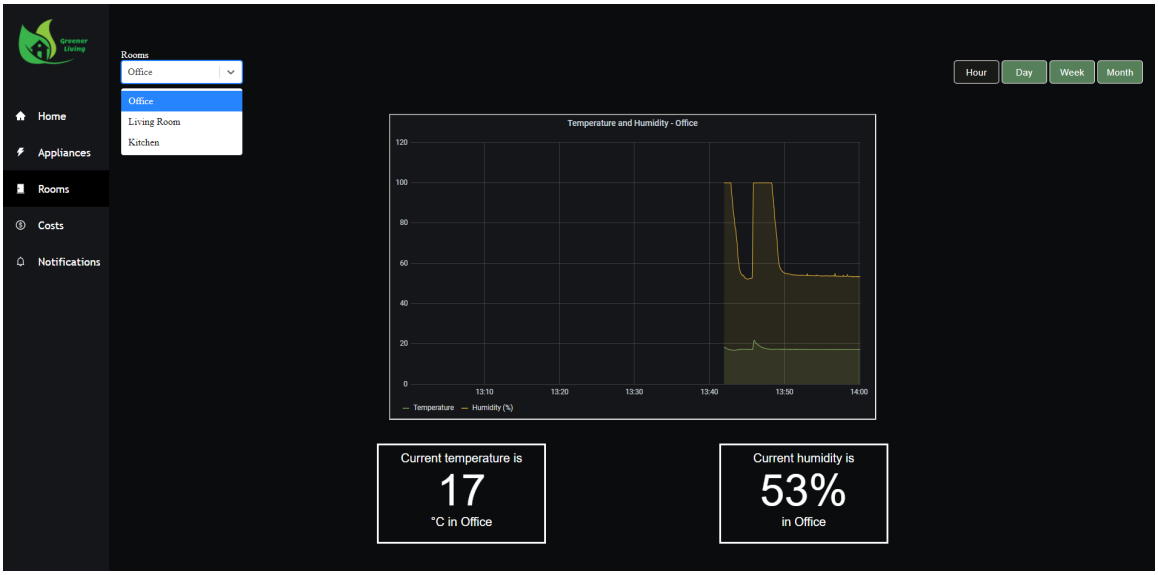


Figure 2.2 Frontend: Rooms UI
Displays the current temperature and humidity where the DHT22 sensor is placed. The user is able to swap between locations and timeframes easily.

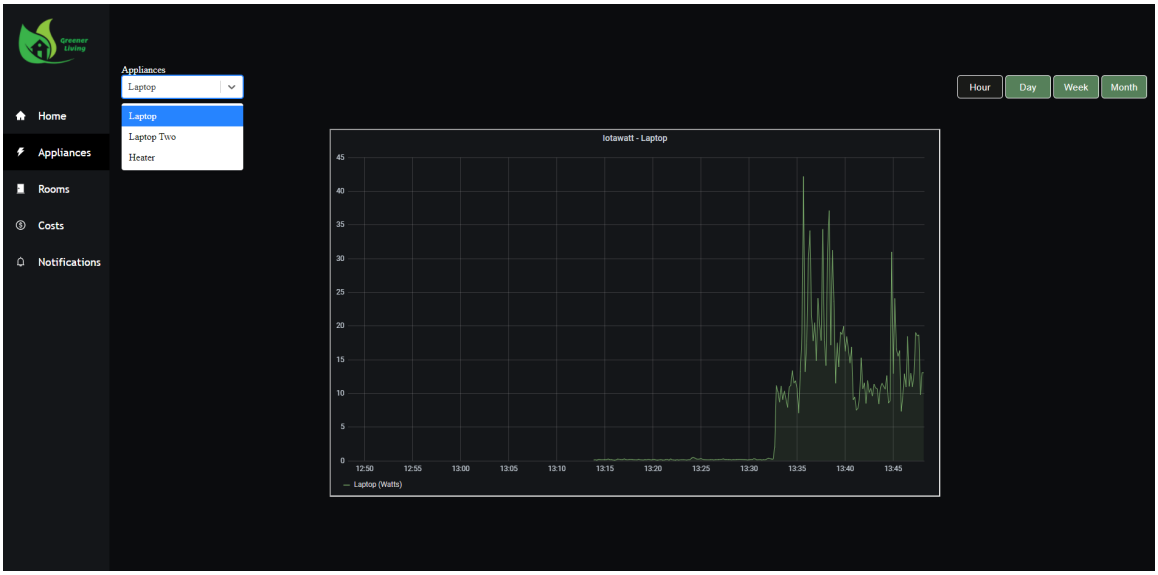


Figure 2.3 Frontend: Appliances UI
Displays the electricity usage of appliances. The user is able to swap between appliances and timeframes easily.

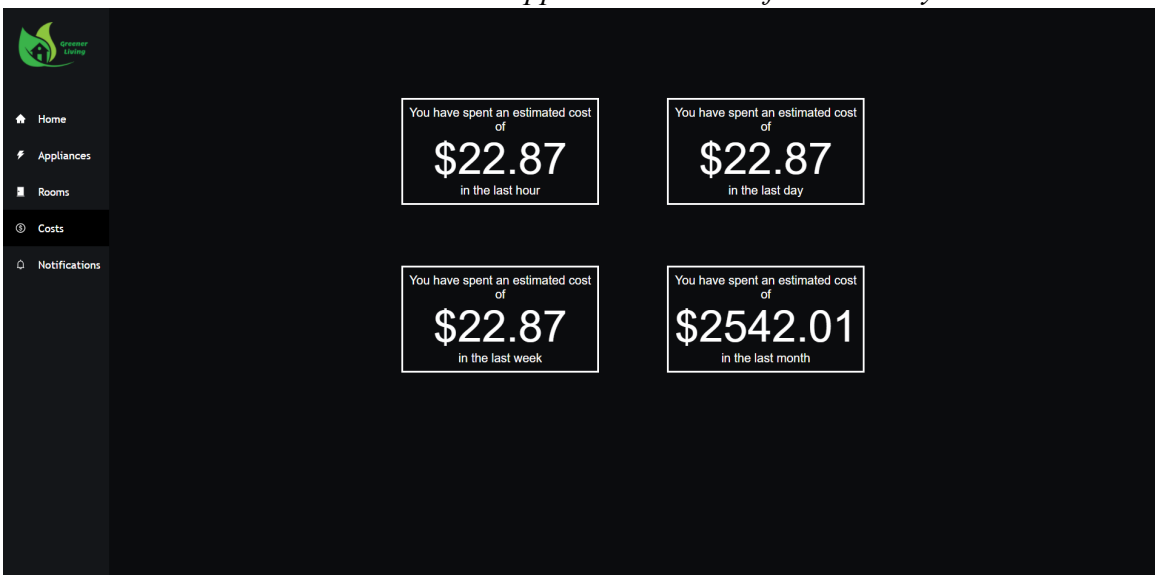


Figure 2.4 Frontend: Costs UI
Displays the estimated cost using a rate obtained by an API call. The call returns the yearly average Electricity rate for Massachusetts.

2.3 Physical description.

Figure 2.5 IoTaWatt connected to current transformers (left) and smart plugs (right)

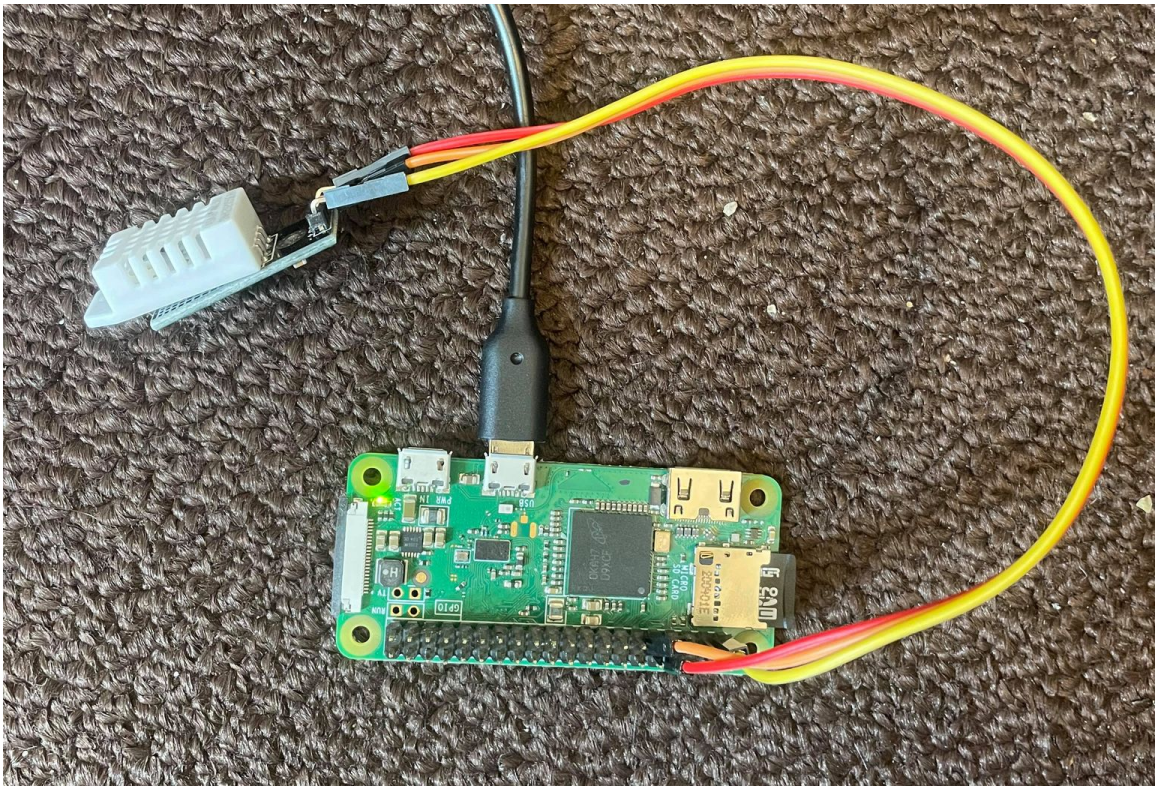


Figure 2.6 DHT22 Sensor Connected to Raspberry Pi Zero

2.4 Installation, setup, and support

Installation: All parts of the system (the Raspberry Pi, IotaWatt, and sensors) come prebuilt and ready to go. All that needs to be done is to plug them into the appropriate power supply and set them up.

While installing, there are several factors that the user should take into consideration. First, the Etekcity plugs require a 2.4GHz network to connect to, so one should be present near where they are set up. Secondly, the Raspberry Pi and IotaWatt require a WiFi connection to function properly, so they should also be placed near a WiFi router. Finally, the Raspberry Pi should be placed in a location central to the other sensors, and the WiFi router, and one where other people will not be able to interfere with. It is the core of the project, so if it breaks the entire system breaks, and so it should be treated with care.

Setup: Note that in order to set up the system properly, it needs to be done in person by our team. This is because the data recording scripts require the Raspberry Pi's IP address to function, so that must be changed in the code in order to function. Additionally, the Raspberry Pi needs to be set up to connect to the user's WiFi and that has to be done by the team. Similarly, the IoTaWatt needs to be set up and connected to WiFi by the team. Finally, the Etekcity plugs use a preset account, however, if the user wants to create their own, that needs to be created and the corresponding data recording script needs to be adjusted accordingly.
Other than these, software like InfluxDB and Grafana comes pre-installed on the Raspberry Pi, so there is no need for the user to do anything.

Assuming that the above staff-involved setup steps have been completed, what the user needs to do is trivial:

1. Connect the Raspberry Pi to its power supply cable and plug that into an outlet. The Raspberry Pi should turn on, and will start running its data collecting code automatically. Nothing further is needed for the temperature and humidity sensor.
2. Connect the IotaWatt to an outlet and connect the cable of the device you want its energy measured to it
3. Download the VeSync app and log into the preset account.
4. Plug the Etekcity plugs into outlets, and follow the instruction manual provided to connect them to your 2.4GHz network.
5. When the setup is done, connect the devices you want their energy measured to the plugs, and turn the Raspberry Pi off and on by removing it from its power supply
6. Wait 2-3 minutes for the Raspberry Pi to set up, then open the website "localhost:3000" while being connected to the same network that the Raspberry Pi is connected to. You should be able to see the website fully functioning.

Support: All these different sensors and devices might seem intimidating, however, once setup properly, and as long as no one changes anything, it can run unattended for at least a year. The code and the devices have been set up with enough redundancies that most issues the user faces can be solved by simply turning the sensors and/or Raspberry Pi off and on again. The team can be contacted at any of their emails for further assistance.

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation

The user will interact with the website as with any other normal website:

1. As shown in the User Interface section, there are multiple tabs on the left side of the page that allow the user to navigate through different pages of the website. Each tab displays different information. The appliances tab shows the energy data collected from the various devices. The rooms tab shows the temperature and humidity in specific rooms.The costs tab shows the accrued cost of energy used.
2. The drop down menu allows the user to swap between devices, changing the corresponding graph to show the collected data of the selected device.
3. The timescale buttons in the top right corner allow the user to adjust the graphs timescale, allowing them to see the data collected from the devices in the past hour, day, week and month

3.2 Operating Mode 2: Abnormal Operations

All of these operation fixes should be followed by a reset of the system by disconnecting and reconnecting the Raspberry Pi from its power supply. Abnormal operations include lack of power to the Raspberry Pi, the energy monitoring plugs not turning on, and any of the sensors not generating readings. All of these could be solved by first removing them from their power supply, then connecting them back while ensuring proper, firm connection of all components.

One abnormal operation that the user will have to fix is if the DHT22 sensor is somehow disconnected from the Raspberry Pi. In such a case the user has to intervene to reconnect it to the 3 needed pins, which may seem intimidating, but is actually very easy.

First, remove the Raspberry Pi from power and make sure nothing is connected to it. Then connect the DHT22's + wire to pin 1 (the top leftmost pin, located to the right of the SD Card). The out wire should be connected to pin 7 (the third pin below pin 1). Finally, the - wire should be connected to pin 6 (the one directly to the top right of pin 7). Make sure that the connections are firm. At this point, it is safe to reconnect the Raspberry Pi to power, and it will function properly.

3.3 Safety Issues

The biggest safety issue in this project is adding current transformers or adjusting the IoTaWatt physically in any way shape or form, if the user decides to use the IoTaWatt on their electrical panel. In such cases, we highly recommend that the user install the IoTaWatt with an electrician or extensive knowledge about circuit panels. For reference, the average voltage in a circuit panel is about 120 Volts, which is enough to seriously

injure or kill the user. The user should make sure that the circuit panel is grounded properly so they will not be subject to any shocks from loose wiring.

Although a minor issue, the Raspberry Pi may be susceptible to battery overheating if exposed to high temperatures. This would greatly reduce the Raspberry Pi's lifetime, so we recommend that it should be placed at room temperature. If the Raspberry Pi fails, it will erase the InfluxDB database and Grafana instance on it, which means the whole system would fail because there would be no data transfer, so we highly encourage you to place the Raspberry Pi in a safe spot.

Overall, our project relies on electricity to function, so any safety precautions associated with electricity should naturally be taken when working with it. For example, the user should always be mindful of their safety and surroundings, when connecting the sensors to their needed power supply. Spilling liquids on any component of the sensory system is incredibly dangerous, so the user should be mindful of avoiding that.

One final note: The main components of our system are generally separate from each other, with the exception of the Raspberry Pi. If the plugs, IoTaWatt or DHT22 are disconnected or malfunction, that shouldn't stop the functioning of the rest of the system, and only that part will no longer show values on the website. If the Raspberry Pi malfunctions however, everything breaks as the website and the data collection scripts will no longer function, so the utmost care should be taken when handling it.

4 Technical Background

4.1 Hardware Component

Greener Living uses a Raspberry Pi Zero W along with additional sensors in order to fulfill hardware and backend functionality. The Raspberry Pi utilizes a 16GB microSD card to run using Raspberry Pi OS Lite, as well as host InfluxDB and Grafana servers. The Raspberry Pi was selected because it functions as a normal computer with Linux OS and can host and store things on a local network, avoiding the need to spend on cloud services. Influx and Grafana servers were installed onto the Raspberry Pi via ssh using the terminal.

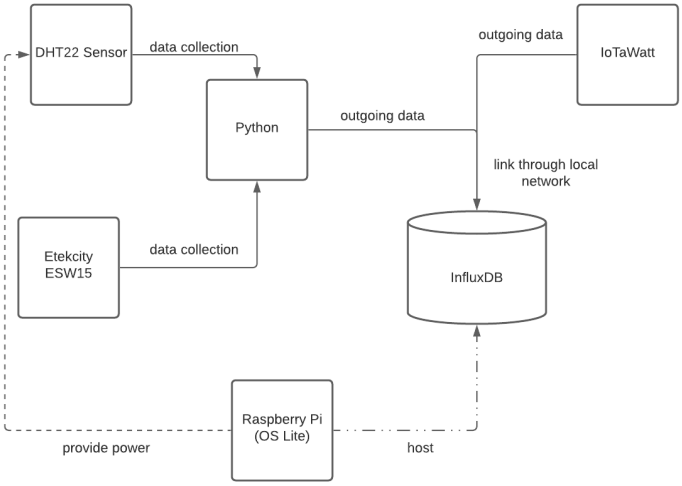
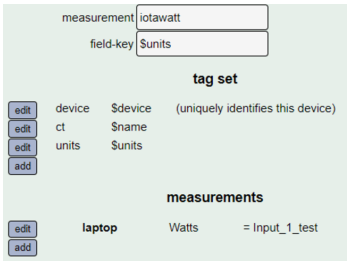


Figure 4.1 Hardware Overview

The IoTaWatt is used to measure energy consumption using current transformers and store that data in the database using the same network. The IoTaWatt writes data to the database using the IP address of the RaspPi according to the preset tags and measurements.



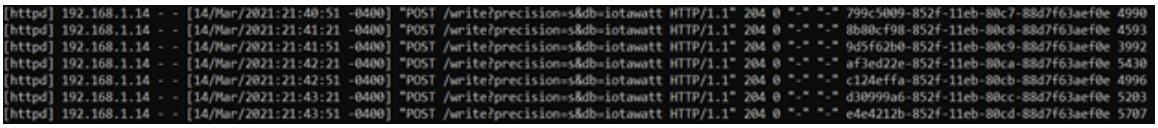


Figure 4.1 IoTaWatt interface with preset tags and terminal showing data being written to the database from the IoTaWatt

The sensors used are the DHT22 temperature and humidity sensor, and the Etekcity ESW15 energy monitoring plug. The DHT22 is connected directly to a Raspberry Pi Zero W via a female to female jumper wire. The Etekcity ESW15 is connected to a power outlet, and the device is in turn connected to it. The Raspberry Pi collects data from these two sensors, stores them in an Influx database and graphs them using Grafana. Python was the main programming language used for the data collection scripts.

4.2 Software Component

Greener Living's software components can be divided into parts: backend and frontend. The backend consists of an Influx database that will store all information received from the sensors and organize the data accordingly with the use of measurements, key-fields, and tag-sets as seen in the example below:

```
name: iotawatt
time                Watts  ct     device   units
----                -----  --     ------   -----
1615772000000000000 213.53 laptop IotaWatt Watts
1615771990000000000 181.48 laptop IotaWatt Watts
1615771980000000000 184.41 laptop IotaWatt Watts
1615771970000000000 212.67 laptop IotaWatt Watts
1615771960000000000 195.66 laptop IotaWatt Watts
```

Figure 4.2 Example of InfluxDB Database

The backend also consists of a Grafana server, which connects to the database using it's server URL and allows the data to be visualized using different tools which are used to generate the graphs and other graphics that are used on the frontend. This is done by setting queries for each graph and storing all of those graphs on the server which then can be used in the frontend using embedded links generated by each graph.

The second part of the software component is the frontend. This includes the web application developed with React, using JavaScript, HTML, and CSS. The project is organized in a hierarchical structure, with the entire application at the highest level and the components and functions at the lower levels.

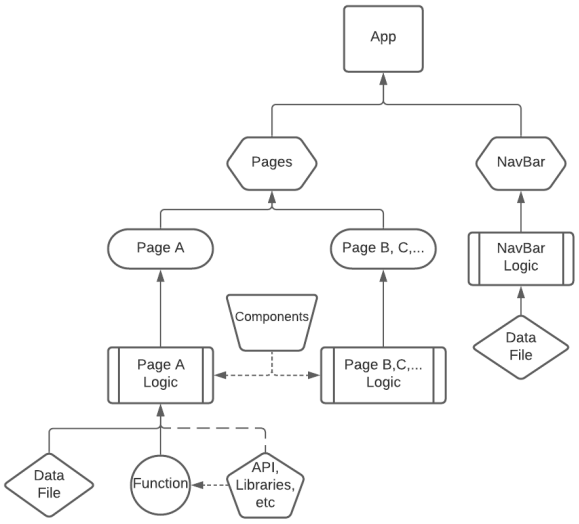


Figure 4.3 Front-end System Block Diagram

At the top, App.js displays a navigation bar and a page, which can be changed depending on which tab is active on the navigation bar. The tabs are created by mapping a list using an array of objects that contain the title, path, and icon of the page. Then, React-Router is used to link each tab with the associated page.

Each page contains its own data file, specific to the intended content to be displayed. The pages also import components such as a dropdown menu or a button group. Pages that import these components pass each other props which are then used to determine which graph should be displayed. Some pages also utilize APIs to perform certain functions, such as to query the database directly from the web application using influx-api. Influx-api allows InfluxQL, InfluxDB's query language, to be used in javascript much easier.

5 Relevant Engineering Standards

Our project follows several engineering standards.

1- National Electrical Code, which sets the foundation for electrical safety in the US. Although this code mostly guides large scale projects, it is important to apply it on the small-scale too, to maintain consistency of electrical standards, and so that people who need to work on the smaller project later on will know what sort of rules and standards it followed.

Our project follows article 300.3 section A, where our conductors must be installed in specific methods and grouped together. The main conductors we are using are wires that connect the Raspberry Pi Zero W to the DHT22 sensor. These wires are pre-made female-to-female jumper wires, enclosed within a plastic cable and are grouped together. This has been installed properly, and it follows the electrical code.

Another important part is article 250.1 section A, which sets the grounding rules for electrical equipment. The jumper wires mentioned above are connected to the ground via the grounding pin of the Raspberry Pi. Additionally, these wires and Raspberry Pi are connected in such a way that establishes an effective ground-fault current path, further adhering to the electrical code.

Finally, our project follows article 110, which gives the requirements of electrical installation. Our wiring was organized in a neat manner, with the Raspberry Pi organized well with its connection with the temperature sensor. This follows section 12 of article 110, ensuring the safety of our product even if a layman somehow gets their hands on it.

2- HTML standards: HTML is the standard markup language for web pages on the internet. It is what we used to create our website, in conjunction with CSS. The website follows the standards for HTML syntax, for example, using a DOCTYPE and formatting document elements in the form <html> element. This standardizes our website code, allowing any future user to understand it easily, and be able to edit it as they see fit.

3- Time and Date ISO standards: This standard (specifically ISO8601) sets an internationally recognized format for dates and times. Our graphs that display temperature, humidity, and energy over time follow this standard, for example, displaying time in the notation hours-minutes-seconds. This allows all users, regardless of location, to immediately be able to read the time on the graph.

6 Cost Breakdown

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	Quantity	Description	Unit Cost	Extended Cost
1 IoTaWatt and Current transformers	1	IoTaWatt and three current transformers	$208.30	$208.30
2 16GB MC Micro SDHC	1	Micro SD card for the Raspberry Pi	$4.25	$4.25
3 Vilros PI Zero W Starter Pack	1	Raspberry Pi Zero W	$31.86	$31.86
4 Etekcity Smart Plug	4	Electricity Smart Plug	$11.68	$46.72
5 Gowoops 2 pcs DHT22 Sensor	1	Temperature and Humidity Sensor	$12.73	$12.73
			Beta Version-Total Cost	$303.86

The budget encompasses only our hardware components. The quantity of smart plugs depends on how many the user wants, so it changes with that. Additionally, if the user wants another to measure temperature in a second room, then a second Raspberry Pi needs to be bought. Finally, for our usage, we bought three current transformers for the IoTaWatt, however, that might go up or down depending on the user. For the software side, all of it was free and open source, removing a potential additional cost from the budget.

7 Appendices

7.1 Appendix A - Specifications

Team #: 18
Team Name: Greener Living
Project Name: Greener Living _____

Requirement	Value, range, tolerance, units
Power Supply	• IoTaWatt - 5 V • PI Zero - 3.3V • Smart Plug - 120V • DHT22 - 3 to 5 V
Database Storage Size	16 GB
Reading Frequency	• IotaWatt - 5 seconds • Smart Plug - 15 seconds • DHT22 - 2 seconds
Dimensions	• IoTaWatt - 1.06 x 3.35 x4.92 in • PI Zero - 2.56 x 1.18 in • Smart Plug - 2. x 1.18 x 2.2 in • DHT22 - 1.05 x 2.32 x 0.53 in
Temperature Range	-40 to 125 ° C
Temperature Error	+- 0.5 ° C
Humidity Range	0 - 100%
Humidity Error	2 - 5%
Overall System Cost	$303.86

7.2 Appendix B – Team Information

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.