Senior Design
ENG EC 463

Team 18 Second Prototype Testing Report

To: Professor Pisano

From: Greener Living

Team: 18

Date: 2/25/21

Subject: Team 18 Second Prototype Testing Report

1.0 Introduction

Compared to our last prototype testing, we have been able to obtain some of our hardware components, such as some energy monitoring plugs and a Raspberry Pi. During our last prototype testing, we were only able to use sample data to represent our energy monitors, but now we are able to start implementing our database with our hardware. In our demo, we show the implementation of our backend with our hardware and also the progress on our web application.

2.0 Equipment

2.1.0 Hardware

2.1.1 Raspberry Pi Zero W
It records the measurements taken by the temperature and humidity sensor and the energy plug in an Influx database, and visualizes them locally using Grafana. All data recorded is updated in real time and that is reflected in the graph visualization.

2.1.2 DHT22 Temperature and Humidity Sensor
It measures the surrounding temperature with ±0.5°C accuracy and humidity with 2-5% accuracy. It is connected to a Raspberry Pi Zero W which records the measurements taken.

2.1.3 Etekcity ESW15 WiFi Energy Monitoring Plug
It measures the power usage of any device connected to it at an outlet. It is used to measure the power consumption of any device the client wants to know more about

2.2.0 Software:

2.2.1 Python 2 scripts

First collects temperature and humidity data, and stores them in an influx database in the raspberry pi. The second does the same but with the energy monitoring plug.

2.2.2 Web Application

It shows our HTML and CSS website implementation as well as the integration of the Grafana dashboards into the website.

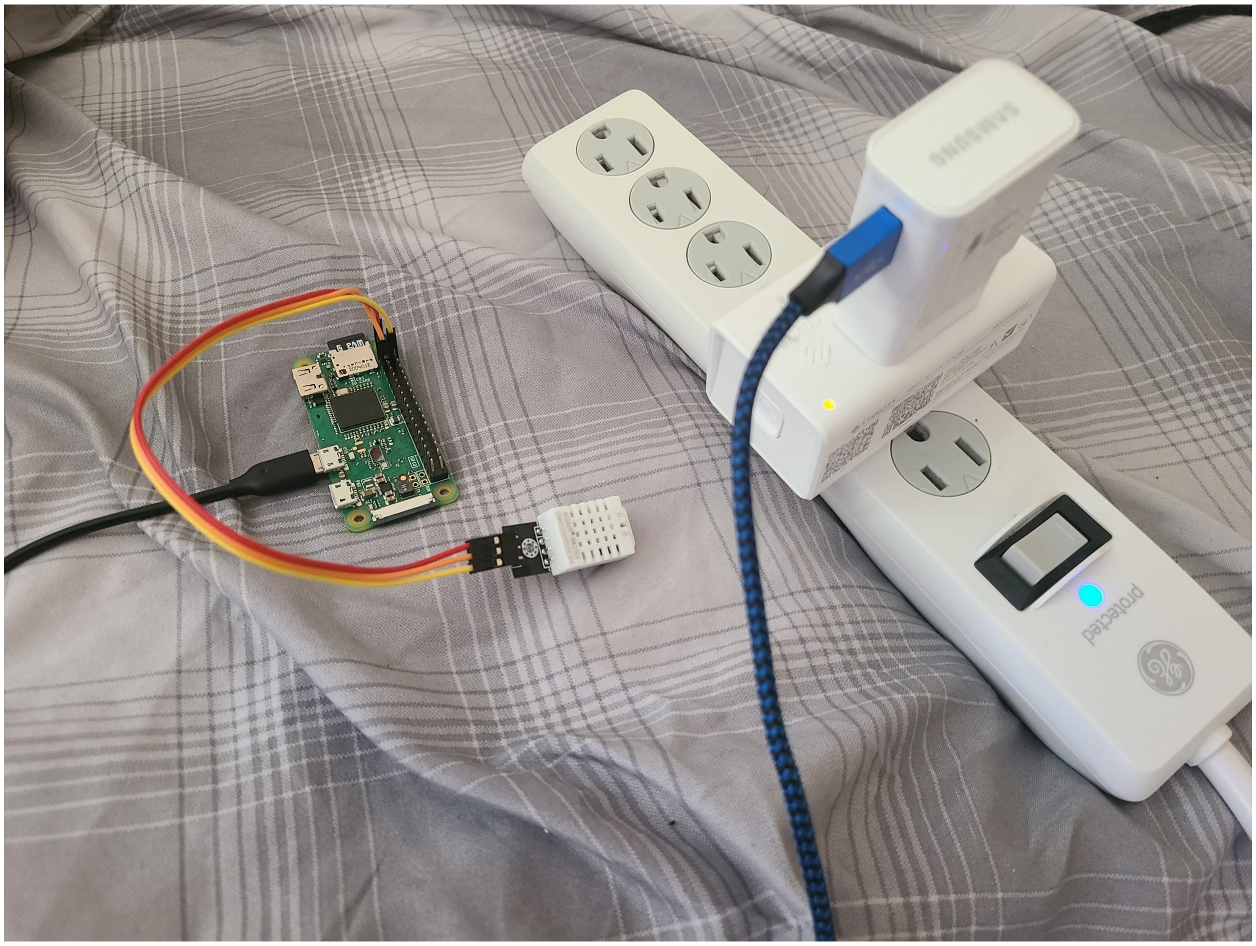


Figure 1: The DHT22 sensor connected to the Raspberry Pi Zero W and the ESW15 energy plug connected to a phone charger.

3.0 Setup

1. The Raspberry Pi Zero W and the energy plug are connected to the same WiFi network.

2. The DHT22 sensor is properly connected to the Raspberry Pi. The energy plug is connected to a device that is actively consuming power.
3. Two scripts, which store the data measured by each of the two devices in an Influx database, are ready to use.
4. The local instances of InfluxDB and Grafana are enabled on the Raspberry pi. The local instance of Grafana is checked to see that all previously created dashboards and graphs are in order.

4.0 Testing Procedure

1. Run the two scripts and show that they are taking measurements live using print statements. Show that changes in the environment (like breathing on the DHT22 sensor) can result in a change in the measurements (the humidity recorded increases).
2. Run a query on the two databases of the two sensor devices to show that collected data is being stored properly.
3. Access Grafana's local interface
    a. Show the integration of the local database and Grafana (data sources tab)
    b. Show that the power, temperature and humidity are being collected and the graphs are updating in real time
    c. Show customization of the panels and all the options it allows
    d. Show time range manipulations
    e. Show that the user can take a link, snapshot, or embed a iFrame onto their own website.
4. Show website designs and current state.

5.0 Measurements Taken

1. Data collected by the two sensor devices is properly stored in a database.
2. Breathing hot air on the DHT22 sensor gives higher temperature humidity values showing that it is recording data and responding properly.
3. The energy plug is taking power usage in real time and the recorded power is shown.
4. The local instance of Grafana is displaying the stored data (temperature, humidity, power) in three separate graphs, which get updated in real time to show the current measurements being taken.
5. The website layout was responsive with the different tabs operational. Grafana was shown to be integrated onto the website, and graphs were shown to update in real time to reflect the current measurements being taken.

6.0 Conclusion

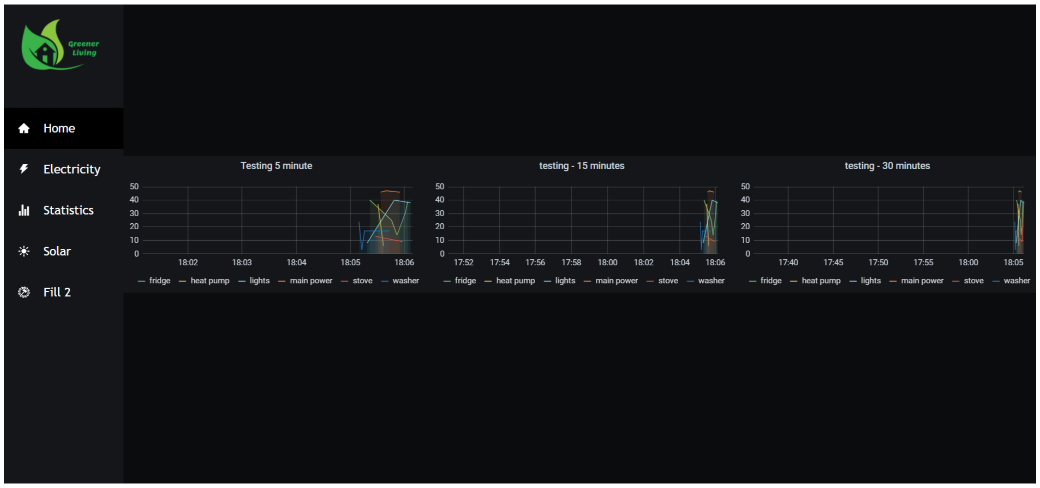
Our second prototype testing was very successful. The sensor devices and scripts worked perfectly, with all data being stored in the proper database and displayed in real time using Grafana. Adding more energy plugs should pose no issue and will be easily integrated.

Additionally, due to a lack of communication from our client, we were unable to showcase the Solar Energy tracking aspect of our project.

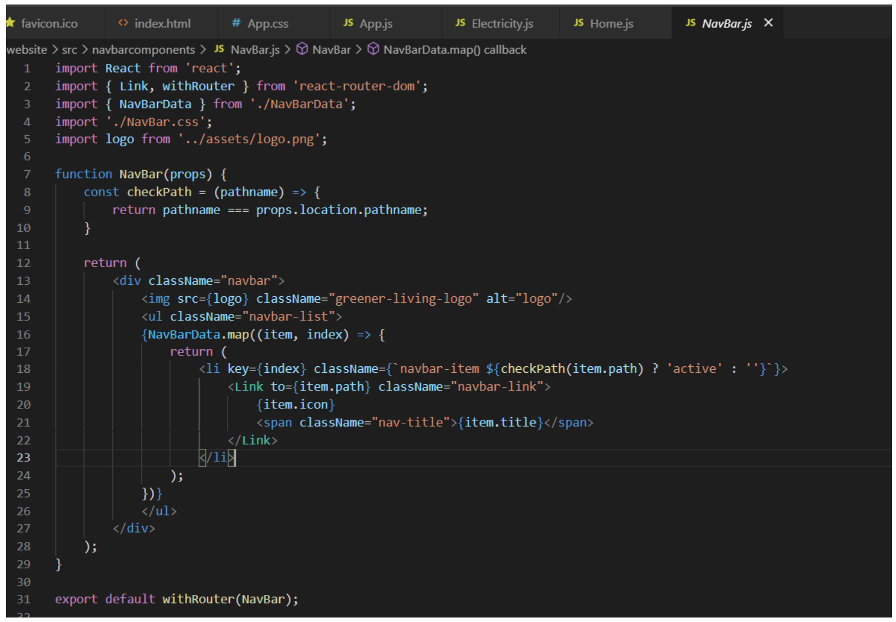
Following up from our first prototype testing where we wished to generate fake data to showcase influxDB and Grafana, we used actual data gathered from sensors this time. We are happy to be able to prove that our product functions end-to-end. Overall, we were able to achieve all the measurable criteria we were hoping to showcase for this second prototype testing. Looking forward, we hope to test using all our different components, and potentially adding several more energy plugs, in the same location so that we can ultimately set up our project in our client's home.

7.0 Addendum - Website Development

Last time, we decided to switch from an HTML/CSS project to a React.js project. The reason for this is that we have the benefits of Javascript and the react framework, as well as HTML/CSS, which will make building the website easier due to the amount of tools provided by React. The design of our web application is currenting being developed to look as similar as possible to the design we create on Figma.



Currently, there is a navigation bar on the left side of the screen with different tabs to organize the information we want to display. This was done by creating components for the navigation bar as well as pages for each tab. Then, by implementing react-router, we were able to show different content when different tabs are selected. To show that Grafana can be used, we embedded the graphs we have created onto our React project and were able to display them and have them update in real time.


favicon.ico
index.html
App.css
App.js
Electricity.js
Home.js
NavBar.js
website > src > navbarcomponents > NavBar.js > NavBar > NavBarData.map() callback

```
import React from 'react';
import { Link, withRouter } from 'react-router-dom';
import { NavBarData } from './NavBarData';
import './NavBar.css';
import logo from '../assets/logo.png';

function NavBar(props) {
    const checkPath = (pathname) => {
        return pathname === props.location.pathname;
    }

    return (
        <div className="navbar">
            <img src={logo} className="greener-living-logo" alt="logo"/>
            <ul className="navbar-list">
            {NavBarData.map((item, index) => {
                return (
                    <li key={index} className={`navbar-item ${checkPath(item.path) ? 'active' : ''}`}>
                        <Link to={item.path} className="navbar-link">
                            {item.icon}
                            <span className="nav-title">{item.title}</span>
                        </Link>
                    </li>
                );
            })}
            </ul>
        </div>
    );
}

export default withRouter(NavBar);
```

We also wanted to note that the hardware can only be used by one member due to not everyone being on campus. Thus, when showing the real-time update for our graphs, a python script extremely similar to our set-up was used to generate “sensor data”. This should not be an issue however, and the script was used for demo purposes only. We will be using the sensor data primarily going forward.

While our web application is still in development, we want to make sure that all of our data collection is working first before trying to display all the content. What we have done so far is create the foundation (blank pages) for which we can add/customize in the future.