Remote Area Monitoring

Wyatt Vining, Abhijeet Malviya, Nicholas Gonzalez, Johan Castillo

DEPT. OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, UNIVERSITY OF CENTRAL FLORIDA, ORLANDO, FLORIDA, 32816-2450

***Abstract* – An innovative and original project consisting of hardware and software designs, the Remote Access Monitoring system or (RAM), is an effective system that can interact in real time and retrieve data of different components: wind speed, humidity, temperature, air pressure, air quality, and soil moisture content through the articulation of a mesh network. The mesh network consists of nodes which represent a geological area; these nodes comprise of unique sensor data recorded in real time by our hardware design. The motivation behind this project is to give the users a better understanding of the area they may use without fearing any catastrophic damage. This project can help as a purpose to stopping wildfires in the forest soon.**

I. INTRODUCTION

In recent years, we have seen the devastating consequences of global warming. One such consequence was the measurable uptick in wildfires. In 2020 alone over 10 million acres of land burned destroying nearly 18,000 structures. Our motivation originated from one event in 2016, the Chimney Tops 2 fire. One group member was present at the origin of this wildfire. The short of it is, wildfires are terrifying and have a lifelong impact on those affected. So therefore, we investigated methods of gathering information about what causes those wildfires. Some of the variable that we took into account included the fuel load, which is the number of combustible materials present such as dried brush and dead trees. We realized that reliably detecting fires is an engineering challenge that may quickly exceed the capacity of two semesters, which is why we didn't create a fire warning system. With consideration of the fire investigator's insights, we created an open-source monitoring system. We focused on monitoring the environmental factors leading up to wildfires. Some of these factors include fuel load, moisture, temperature, volatile compounds in air and wind speed/direction. Our system consisted of a resilient mesh network of nodes. Each node reports its data back to an aggregator node where the data is stored and displayed. All nodes are modular allowing only the necessary sensors be equipped for the specific application of the system. The system is fault tolerant, maintaining functionality with a minimal number of nodes remaining. The information gathered by our system aides those responsible for forest management in the prevention of forest fires by prompting preventative actions as well as informing movement of assets when actively fighting a fire by placing the mesh near populated areas may aid emergency management in issuing earlier warnings to residents in the event an evacuation is required.

II. SYSTEM COMPONENTS

Our Remote Area Monitoring device is composed of many different types of components and modules that work together in order to gather the data for our nodes. This section introduces and explains the different components within our system.

A. Microcontroller

The microcontroller unit or MCU will be the brains of each node. After researching different types, we decided to use an ESP32 system on a chip from Espressif Systems. We made sure that it had a good selection of both digital and analog general purpose input output (GPIO) pins and also that it was low-cost and has a versatile system. The chip also executes a 32-bit application and the clock frequency of this board reaches 240 megahertz, and it has 512 kilobytes with 30 or 36 rounds, 15 in each row. The ESP32 is also able to withstand high temperatures and also included a built-in Wi-Fi antenna so that the user can interact with the device portably with good connection.

B. Photovoltaic Panel and Battery

We decided to use two components to power the Remote Area Monitoring device. One of these components is by using 3 Watt Minimum 6 Volt Photovoltaic Panels which gathers energy from sunlight to power the system. The photovoltaic panels charge the lithium-ion foil battery in each node and helps the device to operate off-grid for extended periods of time. The lithium-ion foil battery is a type of rechargeable battery made up of cells in which lithium ions travel from the negative electrode to the positive electrode via an electrolyte during discharge and back again during charging. To make sure that these components are safe, we used a TP4056 battery charger chip that protects the cell from over and under charging. The TP4056 chip features two status outputs, one for charging in process and one for charging completed. It also has a charge current of up to 1A and accepts a wide range of voltages while charging the battery.

C. Temperature, Humidity, and Pressure Sensor

We needed to use a BME280 sensor in order to gather data for three different types of environmental nodes which are temperature, humidity, and atmospheric pressure. It combines high linearity and high precision sensors, making it ideal for low current consumption, long-term stability, and strong EMC resilience. It also has a fast reaction time and meets the requirements for developing applications such as context awareness and high accuracy across a wide temperature range. The BME280 was created with low current consumption (3.6 pA @1Hz), long term stability, and good EMC robustness. BME280 was sensitive to waste heat from nearby devices so we then also added a DS18B20 temperature sensor in order to offset the temperature of the BME280. A capacitive soil moisture sensor was used to measure the moisture in the soil by using the soil as a dielectric.

D. Anemometer

An anemometer is needed to measure the wind speed and direction. The anemometer that we used is self-built with two different assemblies both using magnets in order to determine the wind characteristics. The wind speed sensor part uses an A3144 hall effect magnetic sensor fixed in place and produces active high pulses as the magnet passes through the sensor. The number of pulses is counted by the microcontroller which it then determines the wind speed. The wind direction sensor part uses a Melexis MLX90316 absolute rotary position sensor with a magnet and wind vane attached to the end of the axle that gathers and returns a PWM signal when it rotates that can then be used to get the direction.

E. CO2 and TVOC Sensor

We needed to use a CCS811 air quality sensor along with a Total Volatile Organic Compounds (TVOC) sensor to measure the CO2 level from the amount of smoke present in the area. The TVOC sensor that we used is able to detect 1000ppm change in TVOCs. These sensors were also placed on our product while activating the mesh network. As the mesh network is activated, the air quality detects the air and obtains all the data it can and displays it on the web interface when the user hovers over the different nodes as they please.

F. Compass

A Honeywell HMC5883L digital compass is used to find the wind direction and determine the node's position relative to the Earth. The measurement for wind direction comes from a compass heading from 0 to 360 degrees. This serves as a reference point to base the wind direction measurements from the anemometer. It has a magnetometer that is used to measure the compass azimuth, which also helps to add minimal cost and complexity to the system while reducing human error. The magnetometer's X-axis is aligned with the zero point of the wind direction to measure the direction and the degrees.

G. Camera

We needed to use a camera for the device so that it could take pictures of the environment around it so that it would report data about it in order to prevent future fires. The type of camera that we used 2MP Sensor SPI Interface. The images from the camera extracts details from the environment and identifies sources that can cause forest fires. Based on the output signal, there are two types of cameras which are analog and digital.

III. SYSTEM CONCEPT

Our system is superlatively displayed in terms of system components which are placed on our circuit board to gather data in real time. These components; mostly purchased; are used fully to produce the final product. The section provides a technical aspect of the components that we have used in our project.

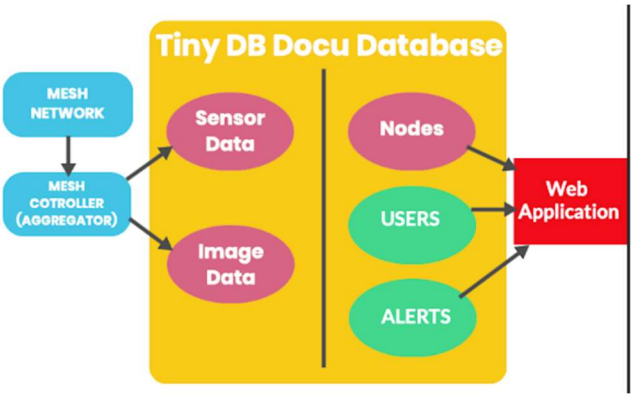

Fig. 1. System overview of software and hardware

Here, we have nodes in a mesh network connected to the mesh controller (aggregator) which connects to the server running parser which connects to the database and then the web application.

IV. HARDWARE DETAILS

Excluding the microcontroller due mainly to probable redundancy, each of the primary system components described in previous sections of system Components, will now be discussed in even greater detail with test results provided.

A. Soil Moisture

Our testing technique will consist of three extreme-to-extreme cycles. The first extreme is to leave the dry sensor out in the open on a paper towel. This is an example of severely dry dirt. The sensor may also be submerged in seawater. This saltwater solution will consist of 3/4 cup water and 1 teaspoon table salt. The seawater will stand in for thoroughly wet soil. This circumstance is not natural, but it does indicate the sensor's functioning range. The sensor will be measured dry before being immersed in a saltwater solution. When the sensor is taken from the fluid, it will be totally dried. For each sensor, the submersion-drying cycle will be done Two times.

The results of our testing may be observed for both samples in Table 1 and Table 2 below.

Table 1
Soil Sensor 1 Test Results

Continuous Current Draw (mA)		5.64	
Cycle Number	Dry Output (Counts)	Submerged Output (Counts)	Delta (Counts)
1	2870	953	1917
2	2909	961	1948
3	2903	946	1957

Table 2
Soil Sensor 2 Test Results

Continuous Current Draw (mA)		5.66	
Cycle Number	Dry Output (Counts)	Submerged Output (Counts)	Delta (Counts)
1	2863	942	1921
2	2858	954	1904
3	2857	945	1912

This data allows us to derive three major conclusions. First and foremost, the sensors have a wide range of counts. The difference from low to high measurement is larger than 1900 counts over all 6 trials and 2 sensors. This is nearly half of the total 4096 counts that the microcontroller can measure. This is a significant enough delta to detect changes in soil moisture. The measurements are precise, as we can see in the second observation. After being immersed, the sensors returned to their dry measurement in all experiments. This is also true for submerged measurements. This offers us assurance that the measurements will be constant when the environment is stable.

B. Photovoltaic Panel and Battery

The engineering team is planning to use a 5x5 polycrystalline solar panel. They give a panel with a relatively small area of roughly 5 square feet, a peak power output of 3W, and a peak output spec (500mA) at 6 volts, which is enough to run the 200mA system and charge during peak daytime hours. Our engineers anticipated that this would be adequate, and it was proven by testing, as explained more in the following sections.

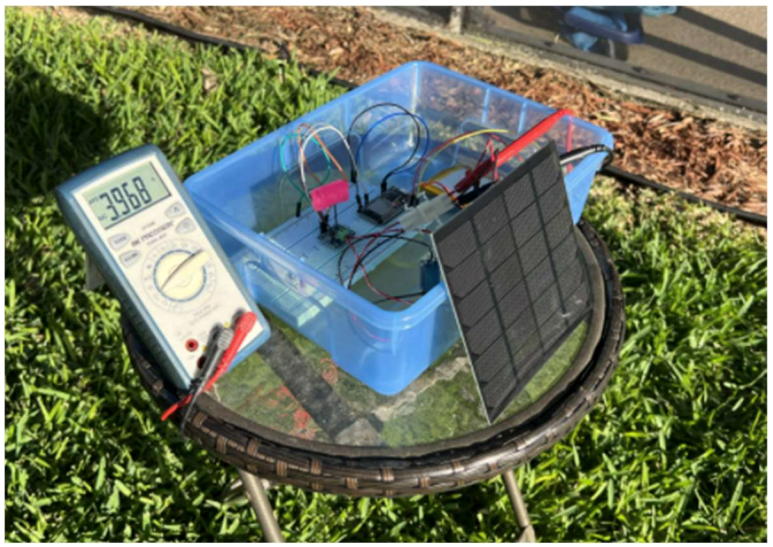

Fig. 2. Solar power system testing apparatus

During the early testing phase for our design, we utilized the testing apparatus depicted in Fig. 2. This apparatus consisted of the major components in our power system designs. The continuous draw of from the battery with the photovoltaic panel disconnected was approximately 200mA to represent our expected power budget.

During a 3.5-day period, the battery regulator, battery pack, and solar panel were all assessed. We created a multi-axis graph with Minutes on the horizontal axis and a double vertical axis with Current (mA) on the left vertical axis and Voltage on the right horizontal axis. The battery voltage periodically rises but remains constant over the day, and the current swings to the negative, suggesting that the battery is being charged.

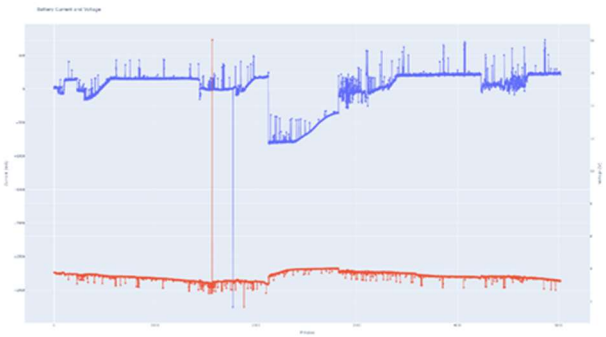

Fig. 3. Battery voltage and current over time.

C. Temperature, Humidity, and Pressure Sensor (BME280)

The BME280 supports the I2C and SPI (3-wire/4-wire) digital serial interfaces. The sensor has three power settings: sleep, regular, and forced. The sensor alternates between measurements and idle intervals in normal mode. When using the BME280's built-in IIR filter to filter short-term disturbances, this mode is suggested (e.g., blowing into the sensor). In forced mode, the sensor performs a single measurement on demand before returning to sleep. This mode is suitable for applications requiring a low sample rate or host-based synchronization. A variety of oversampling settings, filter modes, and data rates may be used to tailor data rate, noise, reaction time, and current consumption to the user's needs.

D. CO2 and TVOC Sensor (CCS811 sensor)

We are measuring carbon dioxide and total volatile organic compounds to provide an overall picture of air quality in a region when there are fires nearby. With this in mind, we will design the subsystem's testing technique to emphasize stability and the ability to monitor deltas. We may begin by measuring the known atmospheric baselines for both measures. The average carbon dioxide concentration in the atmosphere is 400 parts per million. Total volatile organic compound content is close to 0 parts per billion. When put in a well-ventilated environment, we may anticipate our sensor to read 0 parts per billion total volatile organic compounds.

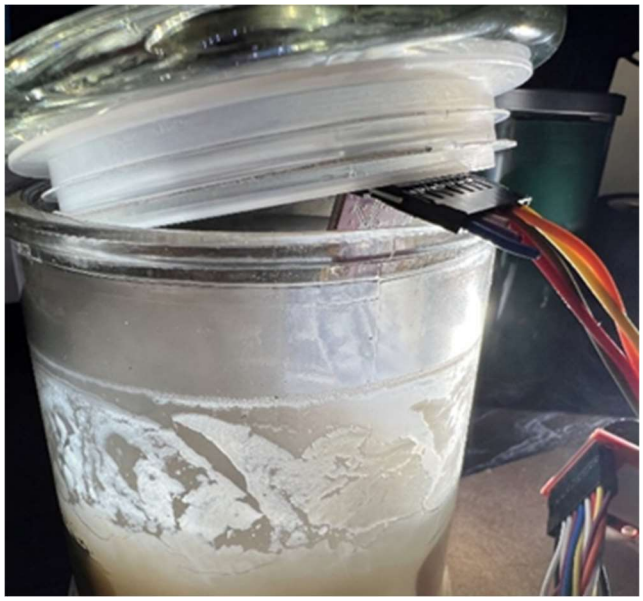

Fig. 4. Testing of CCS811 CO2 and TVOC sensor

The first condition for our testing is that the sensor reports no more than 10 parts per billion total volatile organic compounds but no more than 420 parts per million carbon dioxides when put outdoors. The first prerequisite is that the sensor properly measures our predicted values. From here, we may experiment with different values. These new values will be more of a test for accuracy than accuracy. The sensor must also read the value and provide a reliable reading for five measurement cycles. Stability is defined as a variation in either measurement of no more than 10 parts from cycle to cycle. The results of our testing for the air quality sensor may be observed in Table 3.

Table 3
Air Quality Sensor Test Results

Test Condition	Expected Max CO2 (PPM)	Actual CO2 (PPM)	Expected Max TVOC (PPB)	Actual TVOC (PPB)	Passing
Outdoors	400	400	0	0	PASS
Indoors	400 - 1000	949	500	83	PASS
Candle	Greater than 1000	8224	Greater than 500	14011	PASS

E. Camera

The majority of the camera module testing will be done at the system level. Separate from the functioning of the camera module itself, the flow of pixel data via the node and across the mesh network offers a problem. Two photos will be captured as part of the testing approach for the subsystem level camera module. For this test to be regarded passing, two conditions must be met. The first need is that the Python control program can both order the camera to take a shot and save the captured data to a jpeg file. The second criteria is that both photographs be somewhat eye-focused. We shall leave this criteria at the test agent's discretion.

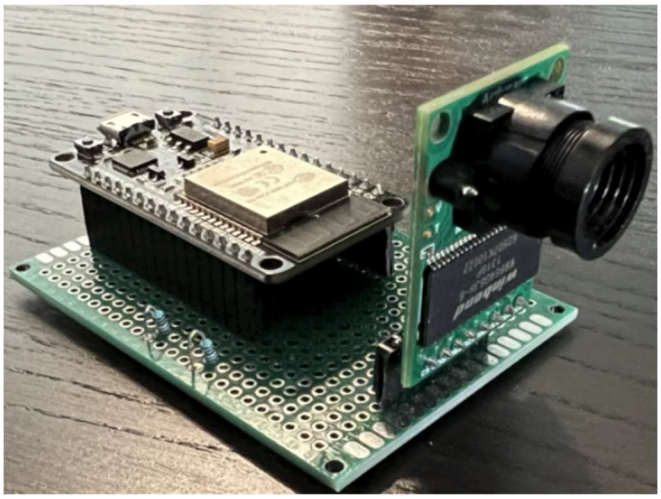

Fig. 5. Prototype used for testing the camera module.

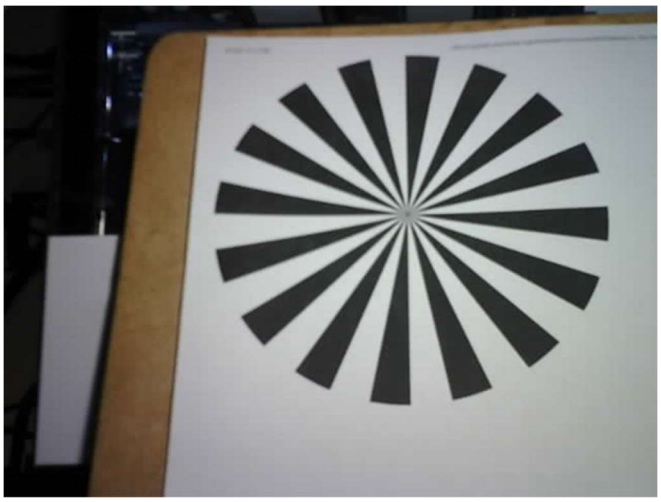

Fig. 6. Spoke target test image

Both prerequisites for a successful test are met by the image in Figure The python command script was used to capture and save this image. The picture subject is a spoke target for evaluating camera optics and resolution. This provides us with a clear metric for evaluating our image. Given the image's resolution, the capture may be judged passable.

While testing our camera, we encountered an issue where the data processing requirement for image capture was outside the capabilities of our microcontroller. The ESP32 does not contain enough memory to store and then transmit the image data. As a result, we selected the ArduCam Mini module depicted in Fig.4. This module has a built in first in first out cache. The microcontroller may now read a single packet worth of data and transmit the data over the mesh network before retrieving the next data packet from the camera module.

F. Main PCB Layout

The actual placement and connectivity of each component in the electrical design is represented by the printed circuit board layout. Our system's physical mechanical design is directly influenced by the layout. This layout may be observed in Fig. 6. We must consider the completed circuit board's dimensions as well as the component placement. We'll also have to take into account any manufacturer suggestions for the physical arrangement of the pieces. The main system PCB houses the CPU, power systems, and all sensors, except for wind sensors. All supporting small components are likewise housed on the main PCB. The primary system PCB layout was created with prototyping and testing in mind. In addition, to save money, we chose a 100mm square dimension. We discovered that restricting the size and selecting a standard size lowers the cost of the PCB from the vendors.

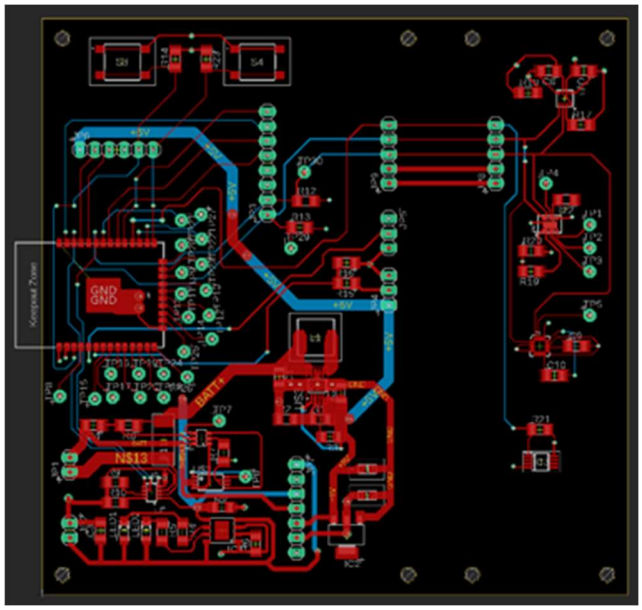

Fig. 7. Main PCB layout

V. SOFTWARE DETAIL (ABHIJEET)

A. Arduino Framework

The Arduino IDE is very simple and reliable. Additionally, the Arduino family of boards had an extensive range of documentation and libraries along with a wide user base. All the factors discussed aligned to address the stress points during the early stage of project development and the end stage of the project where we deploy it. We have chosen the EASP32 microprocessor to connect all our sensors which are collecting data in real time. The board runs software which is made for the Arduino platform. The Arduino interacts with the module in a two-way fashion with the usage of the Arduino core, which contains the main code for the IDE while the web application shares its dependency modules with the external processing component.

Our design board had some essential sensors and other devices installed such as: temperature, humidity and pressure sensor, anemometer, CO2 and TVOC sensor, compass, and camera. Another key benefit of the Arduino framework is the wide variety of software libraries available for interacting with these modules. These libraries often encapsulate the lower level register reads and writes to the sensors themselves. Allowing a developer to access base functionality of each sensor using a higher level function call. These libraries may also incorporate advanced functionality defined in the data sheets for each of the sensors. An example of this is the ability to select advanced running modes for the BME280.

B. Python Programming language

An important standard we followed while using python is PEP 8. We used this to properly style the code we wrote. In order to make sure our functions ran properly, we installed several packages such as dash, plotly, source, pandas, numpy, etc. We used PyCharm as our chosen IDE to construct all the code because it is very python friendly and has more add-on functionalities for important libraries. The purpose of our application was to showcase the real time and historical data of the node (area) where the sensors collect its data from. There is a local area network and a Linux command and a control center as well.

With the chosen Arduino board and MSP 32 microcontroller, python and Arduino are such software tools that should work with our device. Our database holds collections or data from the sensors which we have on the board which gets stored in the backend of the software side and then it sends the data to the frontend of the application to its specific parts of the webapp, for example, the data retrieved from the sensor data would be sent to the section displaying temperature, humidity, and wind speed. We also chose python because it is quick for prototyping and testing. For our GPS system, the nodes are meant to be pinpoints on the map which cites the different nodes. The GPS system we have tracks all the information that is gathered on that specific node and then all the information will be transferred to the data server which is also made through python.

C. Plotly and Dash

Dash is a python framework made by Plotly for developing interactive web applications. Dash is written through the use of Flask, Plotly.js and React.js. In order to make our web application interactive, we just needed python. It is open source, and the application was developed using the framework viewed on the web browser. The Dash core components were used to build our elements displayed on the web application such as graphs, sliders, drop-down menus, etc. There are also callbacks which are used to bring interactivity into the web application. The function helps in defining the activity during where a user can click a button to interact with something or he or she can click a drop-down menu for more options. In order for all the functions to execute, we had to install important packages such as dash-html components, dash-core components, and Plotly. These packages help in creating graphs, drop-down menus, etc. On the other hand, Plotly packages help in creating graphs and plots for reading various datasets which was recorded through our sensor data stored in the backend side of the software application. Callbacks in dash are used to make the web application or any application interactive. Firstly, a callback is initialized using "@app.callback()" and then this is followed by a function. We have used plenty of callback features in our software application to make our web application interactive for the users. We have input functions and output functions. The output functions in our web application were developed to display the wind speed, temperature, humidity, etc.

Fig. 8. Example of our interactive graph

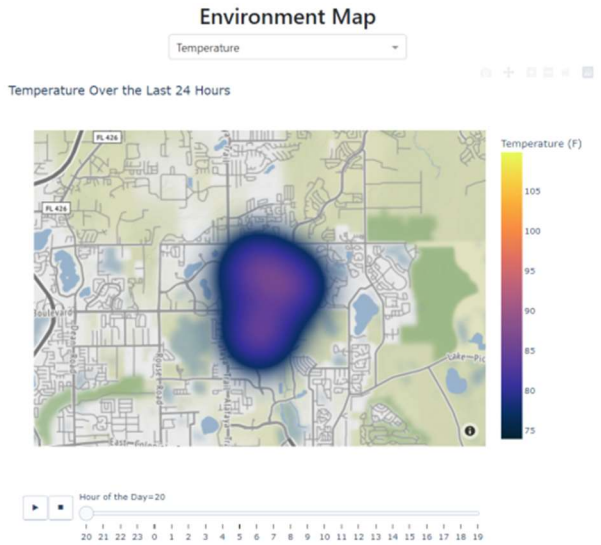

Fig. 9. Example of our density heatmap used to display environmental data over physical locations.

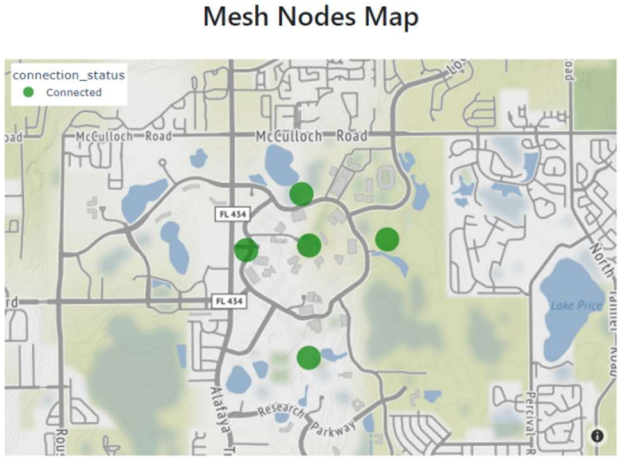

Fig. 10. Map showing each of the nodes physical location and status.

The images in Fig. 8, 9, and 10 represent the different visuals on our web application through the uses of Dash and Plotly. The graphs generated from the data in our system allow the user to select a time scale starting from the last hour and ending with all time. This time scale may also be adjusted using the interactive slide selector at the bottom of the graph. These features give the user the flexibility to view the data in a way that best serves their needs. The density heatmap, like the graphs, allows the user to select the type of data they would like to view with a dropdown menu. The selected datatype is then displayed on a map of the area being monitored. The heatmap is also animated with 24 frames representing the previous 24 hours of time.

D. TinyDB

TinyDB is a document-oriented database written purely in python with no other external dependencies. TinyDB does everything using JSON. We have a JSON file in our IDE, PyCharm, that has stored all the data through our sensors from our board. It is not only run locally to reduce potential costs for remote hosting, but also allows for storage of unstructured data as each node may have a user selected assortment of sensors.

VI. Acknowledgment

The engineers of this senior design project wish to acknowledge the assistance, guidance and support of Dr Samuel Richie and Dr Lei Wei and Dr Aravinda Kar; University of Central Florida.

References

[1] Management of National Park Service Programs. National Park Service, 2006, https://www.nps.gov/subjects/policy/upload/MP_2006.pdf.

[2] "National Fire Danger Rating System." Cibola National Forest and National Grasslands - Resource Management, https://www.fs.usda.gov/detail/cibola/landmanagement/resourcemanagement/?cid=stelprdb5368839

[3] Schlobohm, Paul, and Jim Brain. Gaining an Understanding of the National Fire ... - NWCG. NWCG Fire Danger Working Team, July 2002, https://www.nwcg.gov/sites/default/files/products/pms932.pdf.

[4] Cohen, Jack D., and John E. Deeming. United States Agriculture the National Fire ... - Fs.fed.us. May 1985, https://www.fs.fed.us/psw/publications/documents/psw_gtr082/psw_gtr082.pdf

[5] "Understanding Fire Danger (U.S. National Park Service)." National Parks Service, U.S. Department of the Interior, 2021, https://www.nps.gov/articles/understanding-fire-danger.htm

[6] "IPC Standards for Printed Circuit Boards: PCB Design." Mcl, Millenium Circuits Limited, 10 Mar. 2022, https://www.mclpcb.com/blog/ipc-standards-for-pcbs/#:~:text=IPC%2D2221%20is%20the%20standard,rigid%20and%20MCM%2DL%20PCBs

[7] van Rossum, Guido, et al. "Python Enhancement Proposals." PEP 8 – Style Guide for Python Code, 1 Aug. 2013, https://peps.python.org/pep-0008/#:~:text=The%20Python%20standard%20library%20is,inside%20parentheses%2C%20brackets%20and%20braces.

[8] Smith, Diane M. Sustainability and Wildland Fire The Origins of Forest Service Wildland Fire Research. USDA, May 2017, https://www.fs.usda.gov/sites/default/files/fs_media/fs_document/sustainability-wildlandfire-508.pdf.

[9] National Fire Incident Reporting System Complete Reference ... FEMA, Jan. 2015, https://www.usfa.fema.gov/downloads/pdf/nfirs/nfirs_complete_reference_guide_2015.pdf.

[10] "Anemometer." University of Technology Sydney, 11 Aug. 2017, https://www.uts.edu.au/partners-and-community/initiatives/after-da-vinci/models/anemometer#:~:text=The%20four%2Dcup%20velocity%20anemometer,at%20Armagh%20Observatory%2C%20Northern%20Ireland.

[11] Ouellette, Veronica. "Types of Anemometer." Sciencing, 2 Mar. 2019, https://sciencing.com/units-anemometer-measure-8146408.html.

[12] Max, et al. "The Complete Library of Types of Anemometer - Industrial Manufacturing Blog." Linquip, 20 Aug. 2021, https://www.linquip.com/blog/the-complete-library-of-types-of-anemometer/.

[13] National Geographic Society. "Anemometer Functionality." National Geographic Society, 9 Oct. 2012, https://www.nationalgeographic.org/encyclopedia/anemometer/#:~:text=The%20arms%20are%20attached%20to,used%20to%20calculate%20wind%20speed.

[14] Beach, Emily. "Differences between a Wind Vane and an Anemometer." Sciencing, 2 Mar. 2019, https://sciencing.com/differences-between-wind-vane-anemometer-4801.html.

55–172. 1.