Logo%20Main%20200

**Boston University**

**Electrical & Computer Engineering**

**EC464 Capstone Senior Design Project**

User's Manual



Plants are Neat

Submitted to:

Eugene Kolodenker

eugenek@bu.edu

by

Team # 27

Plants are Neat

Team Members

Ben Livney [blivney@bu.edu](mailto:blivney@bu.edu)

Maxine Loebs [mloebs@bu.edu](mailto:mloebs@bu.edu)

Sergio Pareja [sergio98@bu.edu](mailto:sergio98@bu.edu)

Emanuel Perez [emperez@bu.edu](mailto:emperez@bu.edu)

Noah Spahn [nspahn@bu.edu](mailto:nspahn@bu.edu)

Submitted: 04/13/2021

#### Plants are Neat

#### Table of Contents

[Plants are Neat](#_heading=h.rjea5n1p5ug3) 2

[Table of Contents](#_heading=h.3v7gp4gaqwbo) 2

[**Executive Summary**](#_heading=h.sq5q1tan0oj0) **3**

[**Introduction**](#_heading=h.1fob9te) **4**

[**System Overview and Installation**](#_heading=h.3znysh7) **5**

[Overview block diagram](#_heading=h.2et92p0) 5

[User interface](#_heading=h.tyjcwt) 6

[Physical description](#_heading=h.3dy6vkm) 6

[Installation, setup, and support](#_heading=h.1t3h5sf) 7

[Hardware](#_heading=h.qfp7iu4th8ze) 7

[Software](#_heading=h.f09mqujrdr5m) 7

[**Operation of the Project**](#_heading=h.4d34og8) **9**

[Operating Mode 1: Normal Operation](#_heading=h.2s8eyo1) 9

[Operating Mode 2: Abnormal Operations](#_heading=h.17dp8vu) 10

[Safety Issues](#_heading=h.3rdcrjn) 11

[**Technical Background**](#_heading=h.26in1rg) **12**

[**Relevant Engineering Standards**](#_heading=h.lnxbz9) **15**

[**Cost Breakdown**](#_heading=h.35nkun2) **16**

[**Appendices**](#_heading=h.1ksv4uv) **18**

[Appendix A - Specifications](#_heading=h.44sinio) 18

[Appendix B - Team Overview](#_heading=h.iht5idurlq0f) 18

[Appendix C - Additional Figures](#_heading=h.6awhdndcp8fu) 19

# 

# 

# 

# 

# Executive Summary

Farming is paramount to society. Throughout history, as global populations have increased, farming methods have had to improve to keep up with growing demand. Today, the situation is no different – with more and more mouths to feed, farmers are looking to modernize their practice and adopt technologies that can help them boost crop yields, reduce waste, save money, and stay afloat among industrial-scale agricultural corporations. To address this need, we will be developing a fully-functional

agricultural sensor network, which reports data about specific locations on the farm back to the user in the comfort of their farmhouse. We have come up with a detailed plan to realize our design. The backbone of our final product is our sensor nodes. These will contain our selected sensors, a microcontroller equipped with a radio transceiver, and the required hardware to supply power to all of the electronics inside. All of the components will be put into a weatherproof enclosure. All of these nodes will be connected together in a scalable mesh network. The data from all of the nodes will be pooled into a single gateway node, which will then send compiled data to our web console. Our web console will present this live data in a neat, readable format. There are other agricultural sensor networks on the market, but our product will include a number of innovative features that make it stand out, such as energy harvesting, and the use of a wide variety of sensors to paint a full picture of many conditions out in the field.

# Introduction

In recent years, farmers have been embracing technology to supplement and/or replace more traditional farming methods in order to increase crop yields, lower the cost of production, reduce waste, and – in the case of smaller, local farmers – remain profitable and competitive in a market that is dominated by large-scale agricultural corporations.

As such, we have created an agricultural sensor network to help accomplish all of these things. The network allows the farmer to be aware of the conditions at any location on their farm at which an individual sensor node is placed, all without having to actually set foot out into the field, as the data is available to view on a web console. This saves time and money. The farmer doesn’t have to drive around the farm themself to check on things, or pay people to do the same. Some farms – even those owned by small farmers – are massive in size, so checking up on areas of the farm could become quite time-consuming, even when done only once a day. Having insight into the current conditions around the farm additionally allows the farmer to save money by being able to better distribute resources. For example, the device is equipped with a soil moisture sensor. By monitoring soil moisture, the farmer can in turn adjust how much they are watering different areas of their fields, preventing overwatering and in hand saving money. With knowledge of field conditions, the farmer can make adjustments to ensure ideal conditions for achieving the most successful crop, and thus achieve higher profits.

The sensor nodes feature solar energy harvesting electronics to maintain power for up to a year continuously. Intelligent power path management will maintain power to the sensor node by managing available solar power and power from a backup li-ion battery. The sensor nodes are capable of measuring ambient temperature, humidity, ambient light, soil moisture, and soil temperature, and can transmit data wirelessly using the LoRa RF protocol. The network, when installed correctly, has indefinite range if enough nodes are available, and does not require cell service or Wi-Fi to operate. The network is self-healing, so if a node malfunctions, the network will discover this and notify the user. A gateway node collects all of the data and displays it on a website to easily view data from all nodes.

# System Overview and Installation

## Overview block diagram

Each sensor node is outfitted with a solar energy harvesting system incorporating a small photovoltaic (PV) module and power electronics to maintain power for at least one year. Nodes are equipped with a 3W 6.5V PV module capable of providing a maximum of 460mA in high light conditions. In order to not drive the module too hard, a maximum battery charging current of 400mA is set by the battery charger. For low light conditions, a low-power boost converter increases the PV voltages above 1V up to 5V to operate the microcontroller correctly and to power the sensors.

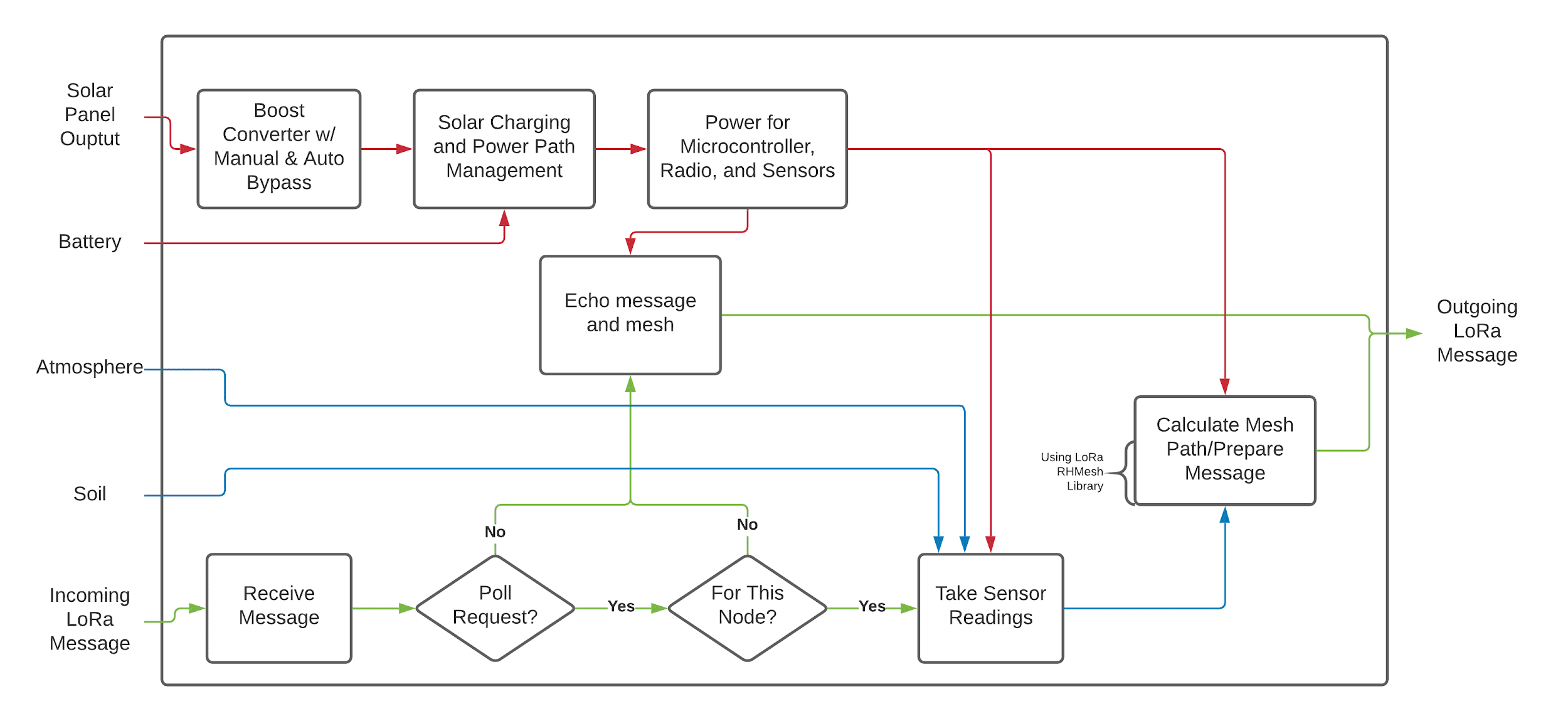


Figure 1: A functional analysis of a sensor node describing power flow, sensor operation, and LoRa mesh networking.

The LoRa communication protocol allows for wireless communication and mesh networking between LoRa-enabled devices using a tunable 900MHz carrier signal without the need for cell service, Wi-Fi, or Bluetooth capabilities. As shown in Figure 1, each sensor node listens for incoming LoRa messages, either from the gateway node or from other sensor nodes. If the incoming message is not a poll request, meaning the message contains sensor readings, the sensor node uses the established mesh protocol using LoRa to repeat that message, sending it closer to the gateway node for analysis. If the incoming message is a poll request, the sensor node checks if it is polling itself, in which case it takes sensor readings and sends them to the mesh network. Otherwise, it repeats the poll request so that other nodes out of range of the gateway can receive it.

The gateway node sends poll requests at regular intervals to the mesh network for data from specific sensor nodes. As described above, the sensor nodes echo the poll request until it reaches its destination, at which point the data is sent and echoed back. If the gateway node receives a response before a timeout, it pushes it to the web console for processing using a Node.js batch file. If no response is received before the timeout, the gateway node pushes a notification to check the status of the requested node. Since the network should be installed such that every node is in range of multiple other nodes, there will be many paths from the gateway to any requested node. Thus, if a timeout occurs, the gateway will know that the requested node is at fault and not some other sensor node.

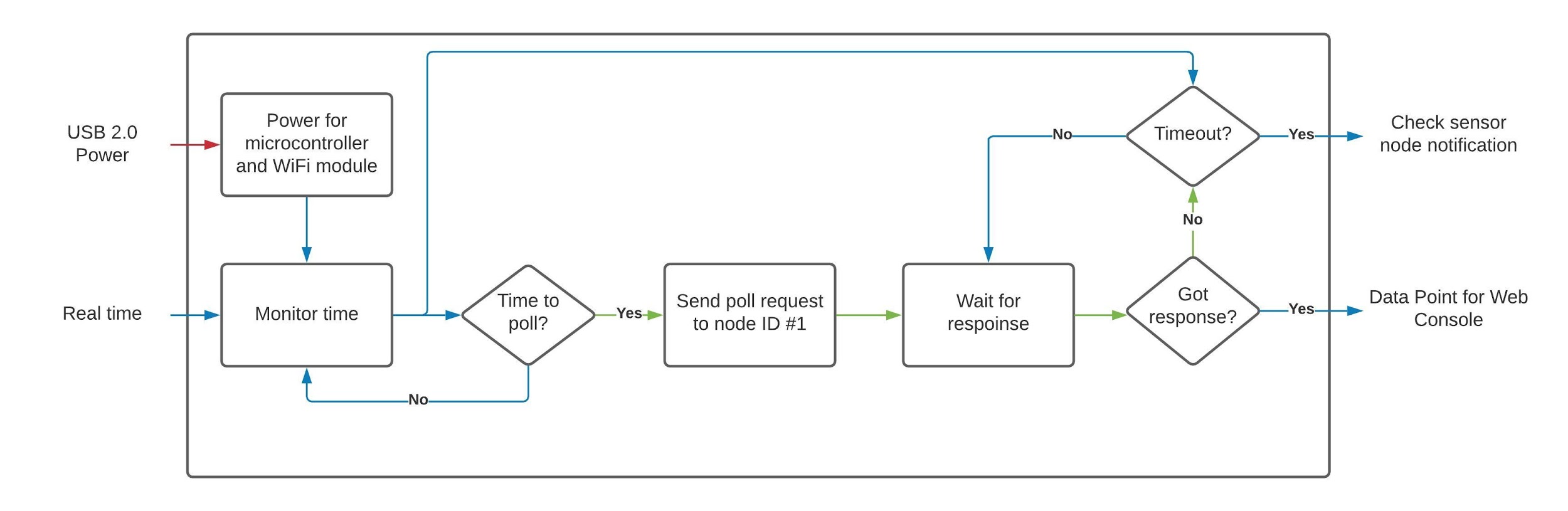


Figure 2: A functional analysis of the gateway node describing poll requests, responses, and interaction with the web console.

## User interface

The user interface of our product is made up of a webpage based client that reports data and notification thresholds to the user. The system uses a webpage integrated with javascript to pull data from the thingspeak channel and display the data for requested nodes at the given time. The nodes can be toggled between on the page to display different graphs. Thresholds for notifications can be viewed within the gauge values.

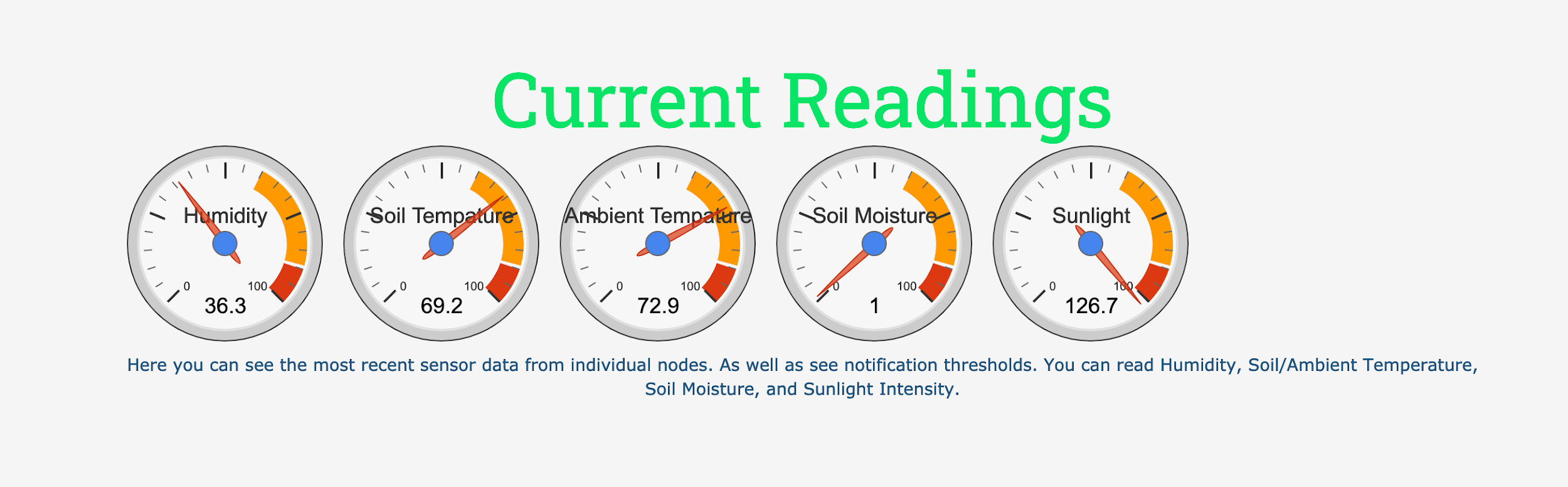
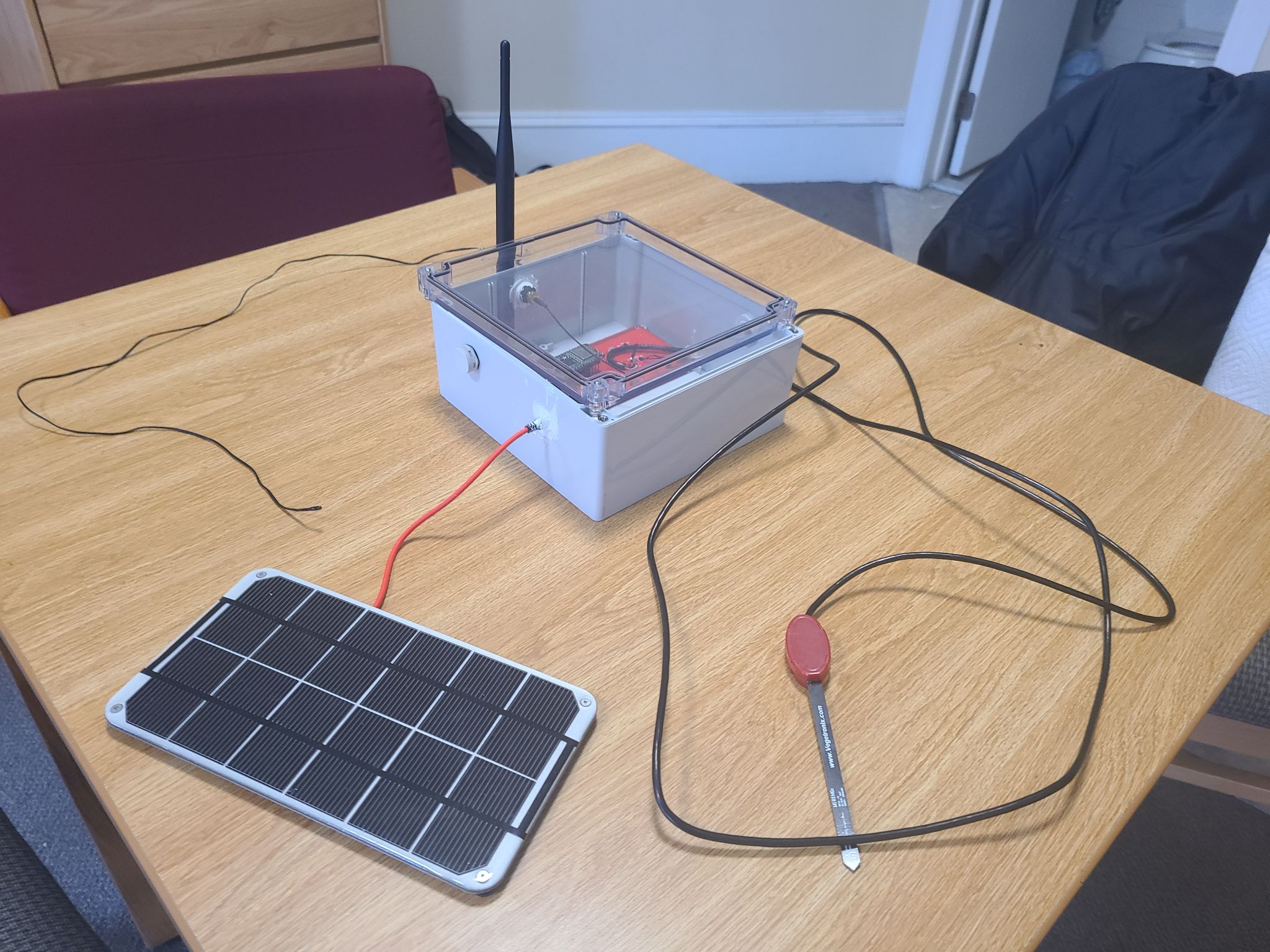


Figure 3: A screenshot of the automatically updating dials on the web console that the user can view to see the data in real time.

## Physical description

The electronics of our sensor node are housed in a NEMA 4X compliant enclosure. The enclosure measures 7.1 x 7.1 x 3.54 inches. It has a removable, clear plastic cover attached with screws. A gasket provides a waterproof seal between the cover and the rest of the enclosure. Attached are two vent plugs which allow for pressure equalization. Sensors for soil moisture and soil temperature extend out of the enclosure to be placed into the ground. The solar panel cable also feeds into the enclosure. The solar panel should be placed to the side of the enclosure, so it doesn’t cast a shadow and prevent accurate readings of the ambient light sensor. The antenna also is external, to prevent any sort of attenuating of the signal.The through holes drilled into the sides have all been sealed with a marine-grade sealant to keep the enclosure completely weatherproof.

The node is secured to the ground with simple ground stakes, similar to what you’d use to secure a tent. They are places in holes that extend through to the bottom of the enclosure, external to the part that is weatherproof.

Figure 4a and b: Pictures of the enclosure of the sensor node.

## Installation, setup, and support

## Hardware

Installation of hardware will require as many nodes as needed, a micro-usb to usb cable, and any windows computer. The nodes will require four (4) metal spikes that come with each node. Place the enclosure in the desired spot and drive each spike through the slots on the edges of the enclosure and through the ground with a hammer. Drive the soil moisture/temperature sensor into the ground and ensure the humidity sensor is above ground. Make sure that the solar panel is facing up towards the sun. Repeat the process for each node, noting that each node should be no further away than three (3) kilometers away from any other node. If the nodes lack line of sight it is important to place them much closer. The on-board microprocessors already have the code flashed on to them, therefore once nodes are installed they will automatically start-up and work on their own. Then proceed to connect the mothernode with the micro-usb and the computer with the USB. If the on-board LED’s light up, then the installation of the mothernode was successful.

## Software

The user interface can be found on <https://plants-are-neat.github.io/>, no set-up required. Email automatically sends upon reaching thresholds.

A stable internet connection and a computer are both required to run the network.

***Windows***

Installing with a Windows machine will require all relevant files (PANinstall.bat, runMESH.bat, Serial.js) and NODE js to be fully installed. After all files have been downloaded from the git, simply make sure the three files are in the same directory and double click the PANinstall.bat file to run it. This batch file is responsible for downloading the dependencies needed to run the Serial.js file. Then running the runMESH.bat file will initialize the connection between mothernode and the computer. It is important to note that the computer needs to be on to be updating the information on the website. Lastly, make sure that no other programs are using the serial connection, the serial connection cannot be shared between programs and may cause the software to crash and exit execution.

***Other***

Installing with a machine other than a Windows operating system will require all relevant files (Serial.js) and NODE js to be fully installed. Open the command prompt and input the following commands:

npm install thingspeakclient

npm install serialport

npm install prompt-sync

These commands will download all NODE.js dependencies needed to run the Serial.js. To run initialize the connection between the mothernode and the computer input the command:

node Serial.js

It is important to note that the command line needs to be in the same directory as the Serial file. Also important to note that the computer needs to be on to be updating the information on the website. Lastly, make sure that no other programs are using the serial connection, the serial connection cannot be shared between programs and may cause the software to crash and exit execution.

# Operation of the Project

## Operating Mode 1: Normal Operation

Operating Mode:

The following instructions are for the mother node to work properly.

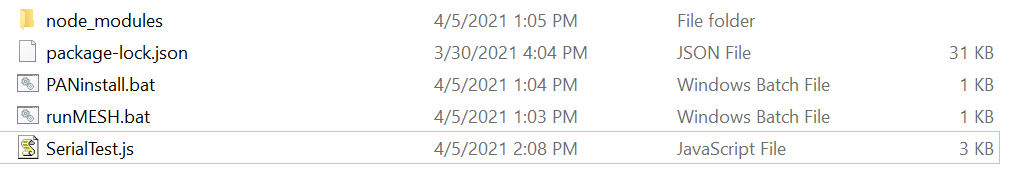


Figure 5: The relevant installation files.

After setup and installation is finished. The next step will be running the runMESH.bat (as shown above) via double clicking the file. Make sure everything is located in the same directory. The user interface would be displayed on screen (as shown below). The program will display all serial connected devices. Please select the correct COM option corresponding to COM that is used for the mother node. After the selection, the program will start running the SerialTest.js.

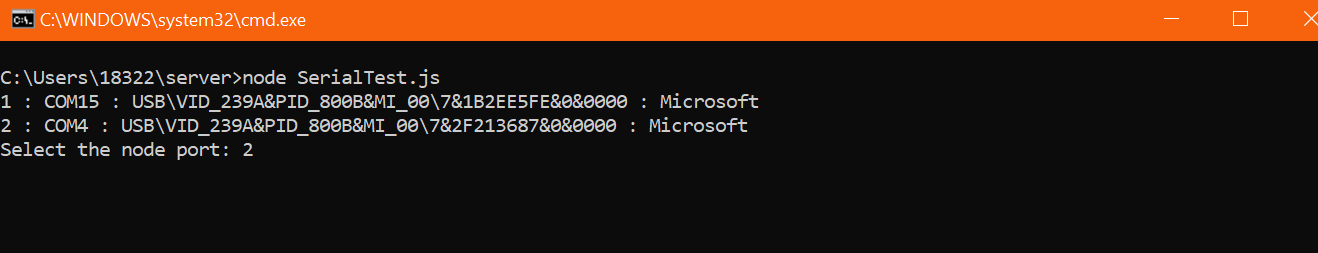


Figure 6: The command prompt while setting up the Node.js connection

Once the outer nodes are installed correctly, the console will start receiving messages (data) from them. If the data received is valid, a message of being sent would show up (as shown below). If the console does not print, “update successfully. Entry number was: #”, check abnormal results section for troubleshooting solutions. If the frequency of the outer node is high (every five seconds), the program will update the server sequentially via first-in first-out behavior. The remote server (website) will display the changes every ten seconds.



Figure 7: A screenshot of the command prompt while data is streaming in.

After the console message, the data will be sorted out in the website to match it’s format like the following:



Figure 8: Live plots of data streaming in for a sensor node.

Other than running the program in the background, no further actions are needed to properly request data, send data, and update the server.

Outer Node Operating Mode:

After setup and installation is finished for the outer microcontroller (node), the outer node will need to be powered via battery, usb, or wall. Afterwards, the node will not need any further steps to work properly. Set the microcontroller in the desired location to collect sensor data. During installation, the device’s data collection frequency can be customized (how long it stays asleep) and sent to the mother node.

Exit (Stop) Operating Mode:

To exit and stop the program, press Ctrl + C (at the same time). The program will ask if you would like to terminate the batch job, type Y and press enter to complete the exit. The image below illustrates the procedure.

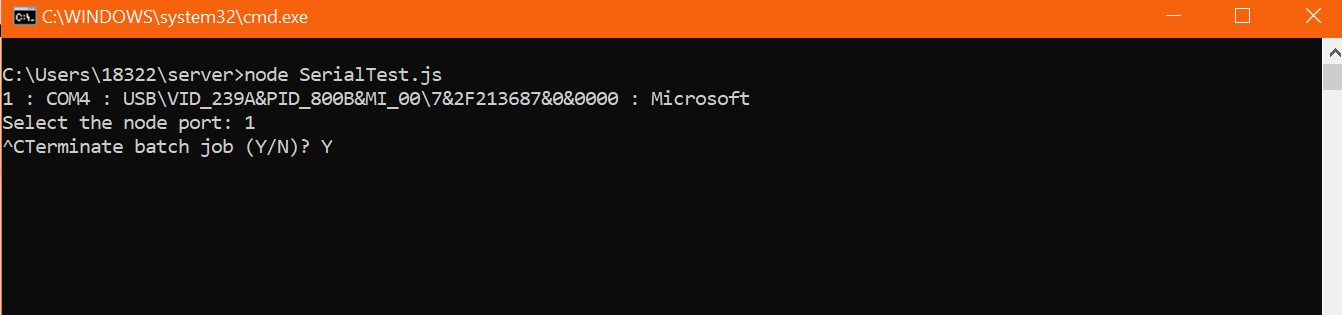


Figure 9: The command prompt when exiting the data streaming program.

## Operating Mode 2: Abnormal Operations

Abnormal results:



Figure 10: Abnormal data.

Above is an example of abnormal data caused by the outer node having broken sensors (or installed incorrectly). The mother node will receive and display the message to the console, but will not send the data to the website. The threshold of what the program considers abnormal can be changed in the SerialTest.js file to match the commonality of the

The problem is rooted either by a faulty sensor or a faulty device. The difference can be seen in the data collected. The example above illustrates incorrect data for all sensors, but if only one is the issue, then reinstalling a new sensor will be the optimal fix. The problem can be solved by installing the sensors correctly or replacing the broken node with a new node.

## Safety Issues

There are no serious safety issues inherent to our project. The greatest concern, which we have put in a lot of effort to prevent, is the chance of water getting in contact with active electronics. We have completely waterproofed our enclosure to avoid any chance of this happening. Of course, it is the responsibility of the user to make sure that water does not enter the enclosure when the cover is off. The only other possible hazard that comes to mind is the possibility to trip over the nodes when they are in the ground! It may be a good idea to mark the locations with some sort of flag to make the placement a bit more conspicuous.

# Technical Background

**4.1 Hardware Component**

The physical electronics for this project consist of sensors to gather meaningful data, a LoRa-equipped microcontroller to poll the sensors and access the mesh network, and solar charging and power path management electronics to effectively harvest solar energy to power the project for upwards of one year continuously.

**4.1.1 Sensors**

The project features a mix of analog and digital sensors. The analog sensors operate by changing an electronic property in response to varying nonelectronic conditions. For example, the soil temperature sensor is a thermistor, whose resistance changes according to temperature, and the soil moisture sensor produces a varying output voltage in response to changing volumetric water content. The digital sensors use the I2C protocol, and are addressed by the LoRa-equipped microcontroller using two data lines. The chosen microcontroller is a FeatherBoard with LoRa from Adafruit.

**4.1.2 Power**

A solar panel is used to harvest solar energy in order to power the electronics. The BQ24074 IC is responsible for managing power delivery and regulation. When solar power is available, it delivers power to the electronics while also charging a backup battery, a 4-cell li-ion 12000mAh battery. When solar power is available but weak, it will not charge the battery but it will power the electronics. When solar power is not available, the battery will start to power the electronics until solar power becomes available again.

**4.2 Software Component**

For software components we chose to use Arduino’s IDE for its simplicity judged against the C++/C the microprocessors would normally use. For libraries we used RadioHead.h, a library built around these LoRa equipped microprocessors that facilitate communication between boards. We also decided to use NODE js because of its event-driven and non-blocking nature. Meaning that the code does not run from line to line but is constantly executing different commands from different functions simultaneously which is useful when it requires to react to real time events. Lastly, we made batch files for ease-of-use for the user. The batch files run commands based on the Windows operating system to install all NODE js dependencies and another batch file to execute the NODE js code.

**4.2.1 LoRa & Mesh Network**

We chose to use LoRa for our RF protocol. Operating on the 900MHz ISM band, it does not require a license. It is very long range with line-of-sight and consumes very little power. We set up our nodes to communicate in what is known as a mesh network. Our network consists of several sensor nodes as well as a gateway node which polls individual sensor nodes for data. In this mesh network, the sensor nodes all communicate with each other as well as the gateway, rather than simple two-way communication between each individual sensor node and the gateway. This has a number of benefits. Poll requests and data to be sent back to the gateway can travel much further distances. Messages can bounce from node to node until reaching the target destination, so as long as intermediate nodes are available, a message can get to a very distant node from the gateway and vice versa. Moreover, the routes from source to target are automatically routed. This means that if a certain node acting as an intermediary were to break and could not receive a signal, a new path could be found automatically if other nodes were in range. This level of robustness is important, especially if farms were going to utilize many nodes, as one node breaking would not knock out the entire network. The LoRa protocol and mesh network were established using Arduino libraries compatible with our Adafruit Feather M0 boards.

**4.2.2 Sleep Protocol**

To minimize energy consumption, the outer node periodically goes to sleep every X (determined by the user during installation) minutes. The use of Adafruit SleepyDog library to robustly put the device into lowest power consuming mode. The library uses the device’s power mode (specifically SLEEP\_MODE\_PWR\_DOWN) to shut down all unessential functions. The library uses the device’s watchdog timer functions to wake up every 8 seconds. As a result, the device repeatedly goes to sleep till it matches the user’s input. Afterwards, the device carries on to its main function.

**4.2.3 Networking to Web Console**

Since different microprocessors have different architectures that complicates board-to-board communication, we decided it would be best to communicate directly to the computer using a micro-usb to usb cable. The cable establishes a serial connection between the computer and the mothernode. The microprocessor then can send bits of information using its built in serial commands to display data. The NODE js code reads from the port in order to display what was picked up and uses ThingSpeak’s own library to send that information to the relevant channels on the website.

**4.2.4 Web Console and User Interface**

In order to demonstrate the WebApp visualization portion of the client’s needs, we utilize a Matlab analysis script, and the ThingSpeak IoT tool to upload data and analyze the uploaded data into a visualization. Data is uploaded from the gateway node via a USB serial connection, and is analyzed by a Matlab script operating within ThingSpeak. The frequency of data collection is programmable and currently set to poll the sensor nodes every 15 minutes. The visualization is done through ThingSpeak with a server-based Matlab script. The live data is displayed within the plants-are-neat.github.io page. Live-updating dials and graphs allow the user to view live data in real time using different formats.

**4.3 Physical Component**

The most important consideration for choosing an enclosure to house our electronics was weatherproofing. The sensor nodes will be out in a field, so depending on the local climate they can be subjected to hazards such as heavy rains, windblown dust, mud, or ice. As such, we decided on an enclosure that is NEMA 4X compliant, meaning that it is watertight, protects against fallen and windblown dirt and dust, undamaged by the formation of ice, and corrosion-proof. We inserted pressure-equalizing vent plugs on opposite sides of the enclosure, which maintain this NEMA 4X compliance. This allows for accurate readings from out sensors, as well as added protection against condensation accumulation. Through-holes were drilled for our external antenna, the solar panel cable, and the soil moisture and temperature sensors. These holes were sealed with a marine-grade polyurethane sealant that keeps any water from entering the enclosure. To secure the node to ground, four holes were drilled next to the cover screws, which extend through to the bottom. These holes exist outside of the part of the enclosure that is weatherproof. We chose to use simple garden stakes, which are easily inserted into these holes.

# Relevant Engineering Standards

This project uses the feather M0 LoRa equipped model from Adafruit. This product already is compliant with many IEEE (Institute of Electrical and Electronics Engineers) and United States government agencies like the FCC (Federal Communications Commission). The FCC controls communications for the radio, the FCC ID number for the product is 2ASEORFM95C. The tests constructed by Adafruit shows that they comply with the following FCC standards:

FCC 47 CFR Part 2, Subpart J

FCC 47 CFR Part 15, Subpart C

KDB 174176 D01 Line Conducted FAQ v01r01

ANSI C63.10-2013

KDB 558074 D01 15.247 Meas Guidance v05

The frequency the radio can reach is 945MHz so it’s within the 0-3GHz set by the IEEE C95.2-2018 standard. The frequency we used, 915MHz, is in compliance with the NTIA and should not interfere with any other radio signals that may be used. Important to note, these standards are, for the most part, for the United States and not liable to be used in other places that are not part of the mainland United States or any of its territories.

Our enclosure has a NEMA 4X waterproofing standard, meaning that absolutely no circulating particles such as dust or lint may get into the enclosure. Dust or lint settling on top of the box is also no issue for the enclosure. It can also resist being ‘hosed-down’ without letting any liquid in. Lastly, it can also easily resist objects thrown or dropped onto it such as dirt. The equivalent for the product using the IP standards (Ingress Protection Code) would be IP64 since it allows no dust to enter the enclosure and no splashes of water to get in, but no tests were conducted using water jets so a higher rating cannot be guaranteed.

The solar panel and the battery both use the IEEE 1562-2007 standard. Both the size and working currents and voltages don’t go above 1 ampere or 5 volts. They both work on powering the microcontroller with adequate power and timing and do not interfere with each other thanks to the on PCB wiring.

The PCB itself was measured against IPC standards (Institute for Interconnecting and Packaging Electronic Circuits). The PCB falls under class-1 of the standard since it is a consumer product with no ‘hard’ requirements to be running without fail. The PCBs are not meant to be tampered with after sale and coupled with the resistant enclosure the only standard necessary to meet was IPC-2221, IPC-2221B standard and the IPC-9592 , it is a class-1 pcb with a level-1 wiring.

The software was done using standard coding practices. Meaning there are no variable names that do not directly relate with the purpose of the variable. All lines are indented using the ‘tab’ key. Each line that requires explanation has either a comment or a block comment explaining the process. Lastly, the general code structure goes library or dependencies inclusions, global variable declarations, functions, and then user defined functions. Each function also starts off with variables unless it is declared later on for the sake of efficiency running the program.

# Cost Breakdown

The cost of each sensor node may seem daunting at first glance. This proposed cost breakdown is for the next design revision of the product, the beta version. The cost can be significantly reduced for the mass-manufactured version by selecting less expensive sensors and by integrating the LoRa capabilities and microcontroller onto a single PCB with all other electronics. The sensors chosen for the alpha version, and the sensors listed here in the beta version, are some of the easiest to use sensors on the market. As such, their price is very high. Selecting less expensive, but less intuitive sensors could bring the price down. Additionally, the LoRa FeatherBoard microcontroller board has many additional unnecessary features for this product. Taking only the LoRa module and a more bare-bones microcontroller can bring the cost down significantly. This price estimate is also for single units, and the price will decrease for bulk ordering.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Project Costs for Production of Sensor Node Beta | | | | |
| Item | Quantity | Description | Unit Cost | Extended Cost |
| 1 | 1 | LoRa FeatherBoard microcontroller | $34.95 | $34.95 |
| 2 | 1 | Large 6.5V 3W solar panel | $45.00 | $45.00 |
| 3 | 4 | Sensors | $12.41 | $49.62 |
| 4 | 1 | All electronics (i.e ICs and passives) | $9.74 | $9.74 |
| 5 | 1 | LoRa antenna | $12.75 | $12.75 |
| 6 | 4 | Li-ion battery cells | $4.28 | $17.12 |
| 7 | 1 | PCB | $4.00 | $4.00 |
| 8 | 1 | Nema 4x enclosure and vents | $31.43 | $31.43 |
| Sensor Node Beta Total Cost | | | | $204.61 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Project Costs for Production of Gateway Node Beta | | | | |
| Item | Quantity | Description | Unit Cost | Extended Cost |
| 1 | 1 | LoRa FeatherBoard Microcontroller | $34.95 | $34.95 |
| 2 | 1 | LoRa Antenna | $12.75 | $12.75 |
| 3 | 1 | Enclosure | $5 | $5 |
| Gateway Node Beta Total Cost | | | | $52.70 |

# Appendices

*.*

## Appendix A - Specifications

|  |  |  |
| --- | --- | --- |
| **User Requirements** | **Solution Rating (0-10)** | **Qualitative** |
| View Weather Data from anywhere | 9 | Website has mobile as well as well as desktop compatibility |
| Modularity of Solution | 10 | System communicates sensor data with modular data center |
| Ease of Setup | 7 | Semi-automated however some technical setup |
| Notifications | 7 | Email notifications including node |
| Enclosure Weatherproofing | 10 | IP43 compliant |
| Node Self Sufficiency | 9 | Hypothetical ability to self sustain indefinitely |
| Communication Range | 8 | Testing suggested a range of 1km between each node |

## Appendix B - Team Overview

Ben Livney

845-642-0837

[blivney@bu.edu](mailto:blivney@bu.edu)

Maxine Loebs

631-702-5813

[mloebs@bu.edu](mailto:mloebs@bu.edu)

Sergio Pareja

787-587-7420

[sergio98@bu.edu](mailto:sergio98@bu.edu)

Emanuel Perez

832-267-8474

[emperez@bu.edu](mailto:emperez@bu.edu)

Noah Spahn

617-970-8255

[nspahn@bu.edu](mailto:nspahn@bu.edu)

History of Team and Company:

The team started in mid September. We wanted to help one of society’s most important sectors, agriculture. A testament of our core values is in our name, Plants are Neat. Our lead market strategist, Ben, wanted our company’s name to follow our values of down to earth customer service while using simple solutions to solve modern problems. The goal of our company is to give farmers a reliable, economical way to survey their farmers without needing to traverse a multi thousand acre landmass.

## Appendix C - Additional Figures

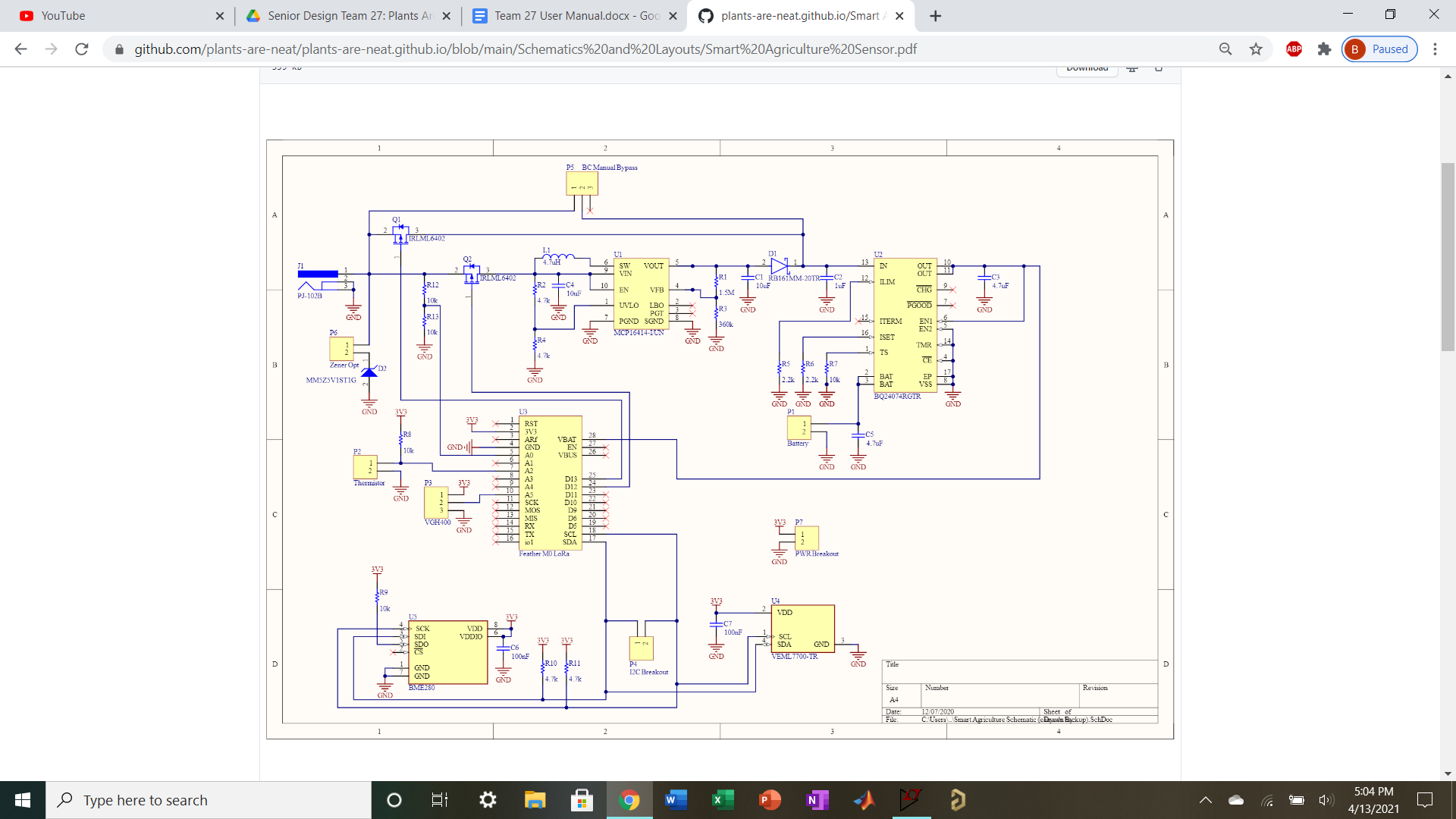


Figure 11: A full schematic.

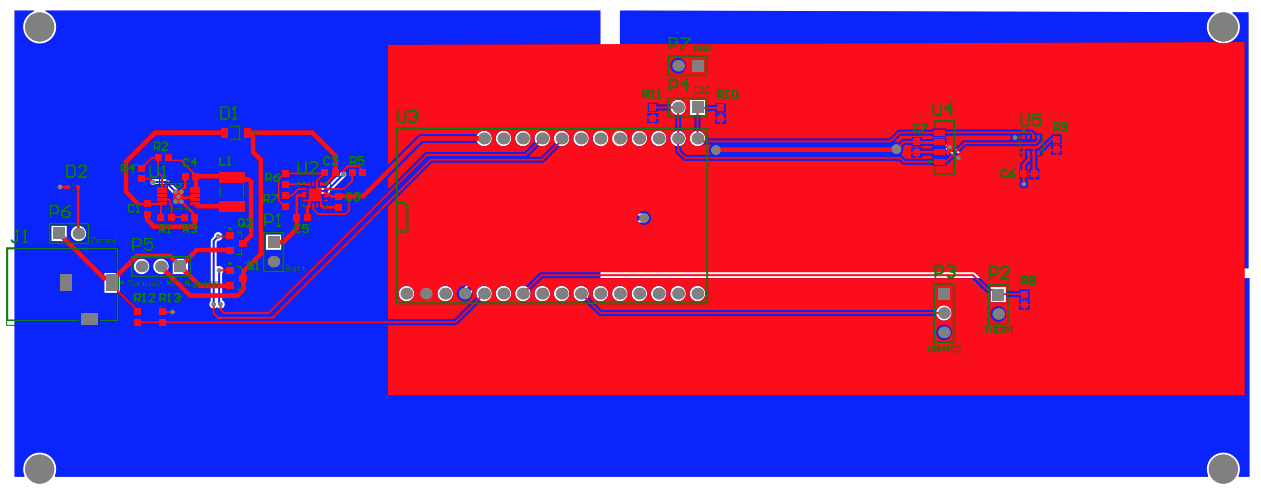


Figure 2: A rendering of the PCB layout.