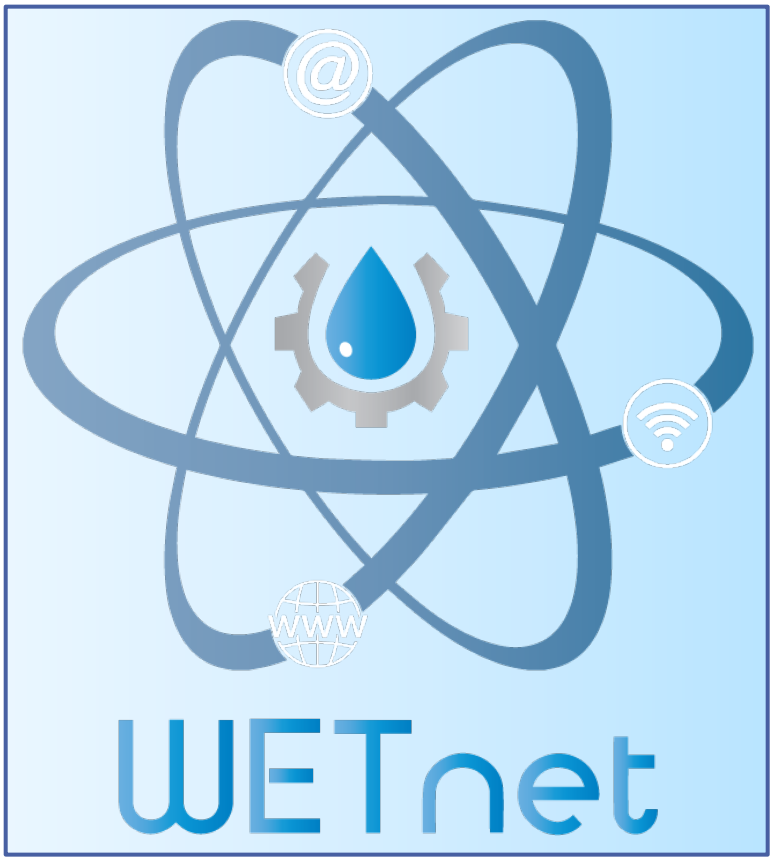


WET (Weather Engineering Team)

Visit us @ WETNet.sdsu.edu

Our Team:

Nathan Vujovich

Alven Eusantos

Joseph Dantay

Yuanzhi Li

Jeremey Lee

Johnny Briseno

Jonathan Acuario

Noel Garcia

Longjie Chen

Matthew Salvino

Stephen West

Kevin Sahagun

Brent Knickerbocker

Ankit Verma

Benjamin Hunt

Philippe Grepo

Brandon Ma

Yusuf Shaikh

Sponsors:

Institution:

Table of Contents:

Abstract:

The WETNet provides a systematic platform for granular resolution of microclimate analysis all while using geospatial deployment techniques. With emphasis on both hardware and software our weather network integrates a sensing system supported by backend servers, and maintained using real time updates in a single consolidated web development environment. Our nodal network is custom designed from the ground up while maintaining Super and Sub Node relationships. This sensitivity allows for the effective regression analysis of data in respect to energy demand expectations and allows for predictive and reactive analytics. With microclimate research as our top priority, our system has the capacity to evolve and expand into a wide integration.

Introduction:

San Diego is known for its pristine beaches and weather conditions, however the "619" is not always considered paradise. Wildfire, as generated by Northern Santa Ana winds, often destroys infrastructure and poses a major threat to the local utility, San Diego Gas & Electric. Wildfire in extreme cases can cause explosions at substations consequently removing customers from the grid. As electric energy continues to be a necessity for daily life, the absence of electricity, at times, could be the difference between life and death.

Our project exits initially as a research solution, and evolves into a network integration to combat extreme microclimate conditions while maintaining high precision weather measurements. Our team uses state of the art engineering practices while considering costs, challenges and benefits of pursuing this microelectronic solution for the county and beyond. This project also serves as as a grooming period for potential SDG&E employees.

The significance of our project over SDG&E's current weather systems is granularity. This lends the opportunity to determine weather over a wide spectrum from 10-minute increments to 10-month intervals. Other advantages include low cost, durability, reliability, micro dimensional, mobility and ease of repair/maintenance. Our design is inspired from existing hardware but unique from the previous mentioned capabilities.

Our network is deployed in a star topology sensor array that receives updates from branches supported by the local Wi-Fi. These branches of the star can be better

defined as Sub Nodes. The Sub Nodes then collect Temperature, Humidity, UV index, and Air Quality. The mother nodes known as Super Nodes will digest this data along with Air Pollutants, Wind Speed, Wind Direction, and Photos by pushing to a hosted Jason Server. The secured server will format the data, and forward the information to our website for user interface. The user can then interact with our hosted environment by clicking specific nodes, panning data, and visualizing microclimate trends. In all, our product enhances the utility, educates the customer and establishes a predictive mechanism for management.

Project Description:

Overview

In order to meet the needs of SDG&E we provided a microclimate weather system consisting of a website to display weather data, a server to store weather data, and a weather network to collect data. What is unique about our system is its small deployment radius and high accuracy. One of the many applications of our system is in determining where the best place is to install long term, high cost equipment. Through the user's analysis of our system's data, the user can determine the best place to deploy high cost, long use equipment based on how the equipment is expected to wear due to weather conditions. Through the analysis of weather systems data, the user can extend the life of their long use equipment and save money in the upkeep of that equipment.

Our weather sensing network is comprised of Sub Nodes and super nodes where super nodes have a full weather sensor package and Sub Nodes have a limited weather sensor package. The Sub Nodes collect data and send it to one of two super nodes which in turn send data to the server where data is displayed by the website for the user.

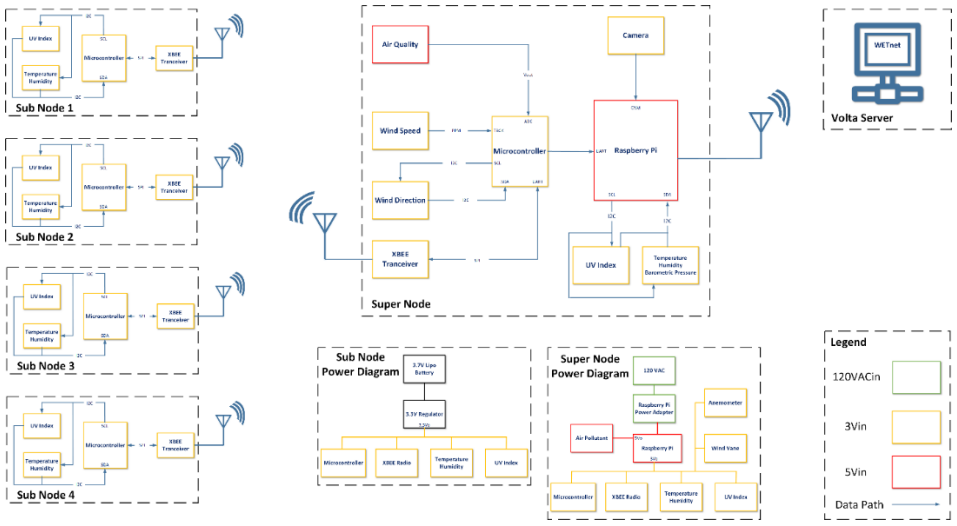


FIGURE 1: OVERALL BLOCK DIAGRAM

Website:

The website template for WETNet began by exploring the requirements of the system. It was created with HTML, CSS, and JavaScript. Using the JQuery Library due to its prebuilt libraries specifically for graphics as well as functionality for data pulls from JavaScript Object Notation or .json files. The Bootstrap framework provides the website with an organized and refined aesthetic. The essential necessity for the system was to be able to compare weather criteria such as temperature, humidity, UV…etc at various locations across the campus of SDSU. The hope is that locations in the shade will be cooler, locations away from construction sites will have better AQI, locations higher up will have more wind interference…etc. A website was required that would be able to show where each node was placed, while showing the various data simultaneously.

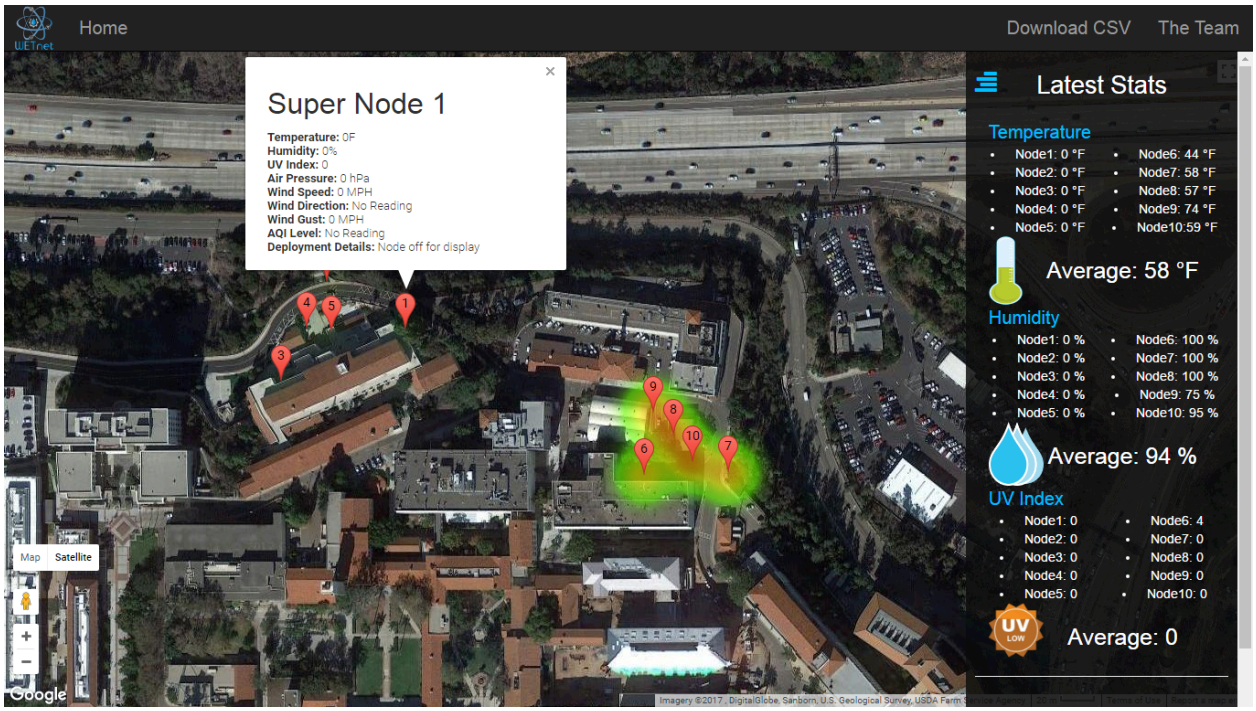


FIGURE 2: WEB SITE WITH POP-UP, SIDE PANE, AND HEAT MAP

WETNet implements a pop out side-panel feature allowing for a more detailed view of the data when the user requires it, otherwise showing the minimal necessities. The side panel is created with a light transparency overlapping the Google Map, creating the illusion of having 50% more of the screen available. The WETNet heatmap utilizes a Google Maps API. Which was chosen because it is used on several applications and websites in addition to Google. Another benefit of the Google Maps API is that licensing is free for up to 25,000 refreshes daily. At its current scale WETNet is

unlikely to need a premium license for more refreshes. This allows WETNet to maximize functionality without incurring additional license fees.

FIGURE 3: WEB SITE WITH SUPER NODE SELECTED AND DISPLAYING IMAGES AT SLIDER POINTS

WETNet's heat map displays the average of all temperature readings for the specified time range chosen on the time slider. When a node is clicked, a pop up window containing the most recent data read from that node is displayed. The popup window also provides deployment details that describe the physical placement of each node. All of the graphs, sliders, and radio buttons are placed on the side-panel while it's expanded. During the time when the side panel is collapsed the user can view the full map and the most current stats of the nodes. This can help give users insight on any sensor data discrepancies such as being mounted in a high elevation area or placed in the shade or indoors.

When the side panel is in the collapsed view, WETNet displays the most recent readings of each node for temperature, humidity, and UV index as well as an average of each respective value. The expanded view displays node selection options, a time slider, various historical data options, and a graph. Selecting the compare button adds the current node selected to the list of nodes to be displayed on the graph. The user can select up to 10 nodes to compare temperature, humidity, or UV Index on the graph. Pressing the clear button will erase all selected nodes from the graph.

Data markers 1-10 are displayed on the map to represent the nodes at their respective GPS locations. Clicking on any one of those nodes will open an info window displaying sensor data from that specific node. This data is an average of data across the time frame specified by the time slider, in the side panel, the name of the specific node and its deployment details are also provided in the info window. This allows the user to understand why there would be some data discrepancies between nodes. For example, a node that is placed atop the SDSU engineering building may have higher temperature averages than a node that is placed in the shade. The map also has a heatmap layer. Each node has a heat point on it that reflects the same average temperature indicated by the time slider. This means that if a node has a higher average temperature, its heat point will appear more red as compared to a node that has a lower average temperature, its heat point will appear more green. This functionality means WETNet's map is s powerful tool able to represent data both visually and numerically.

A uniquely functional aspect of the WETNet tool is the time slider. The JQRangeSlider JQuery plugin provides the user with a responsive and intuitive slider. This slider determines the time frame in which data will be displayed on the website's graph and heat map. Minimum and maximum parameters can be changed in 10 minute increments which allows for very granular examination of historical sensor data. The slider has two views in which the user can view data. The "Today" and "Last 7 Days" buttons allow the user to select a timeframe from the last 24 hours and the last 7 days respectively. As the slider is changed, two still images taken by the Super Nodes are displayed correlating to the start and end of the slider's selected timeframe.

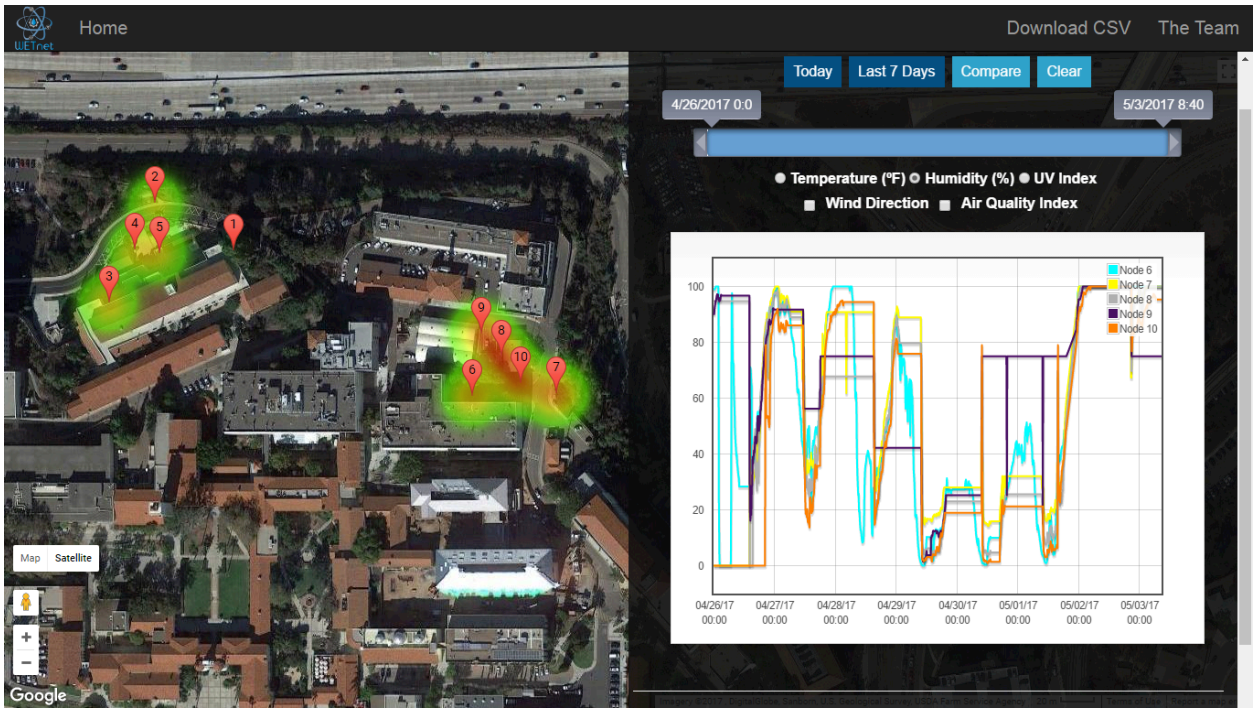


FIGURE 4: WEB SITE DISPLAYING COMPARISON BETWEEN NODES

Nodes 1 and 6 are Super Nodes, the remaining eight are all Sub Nodes. Sub Nodes are more compact and provide sensor readings such as temperature, humidity, and UV index. The Super Nodes are larger and contain extra sensors that allow them to display additional readings such as: wind speed, wind gust, wind direction, air pressure, and air quality. All of these additional options are only displayed when a Super Node is currently selected. When the wind direction or air quality index options are checked, the web page will display the most frequent wind direction and air quality in the specified time frame for Nodes 1 and 6.

WETNet allows the user to download a CSV file containing all historical sensor data. This CSV file contains the same data set as the master list used to populate both the graph and heat map. This file allows visitors to analyze the microclimate of the SDSU campus in greater detail.

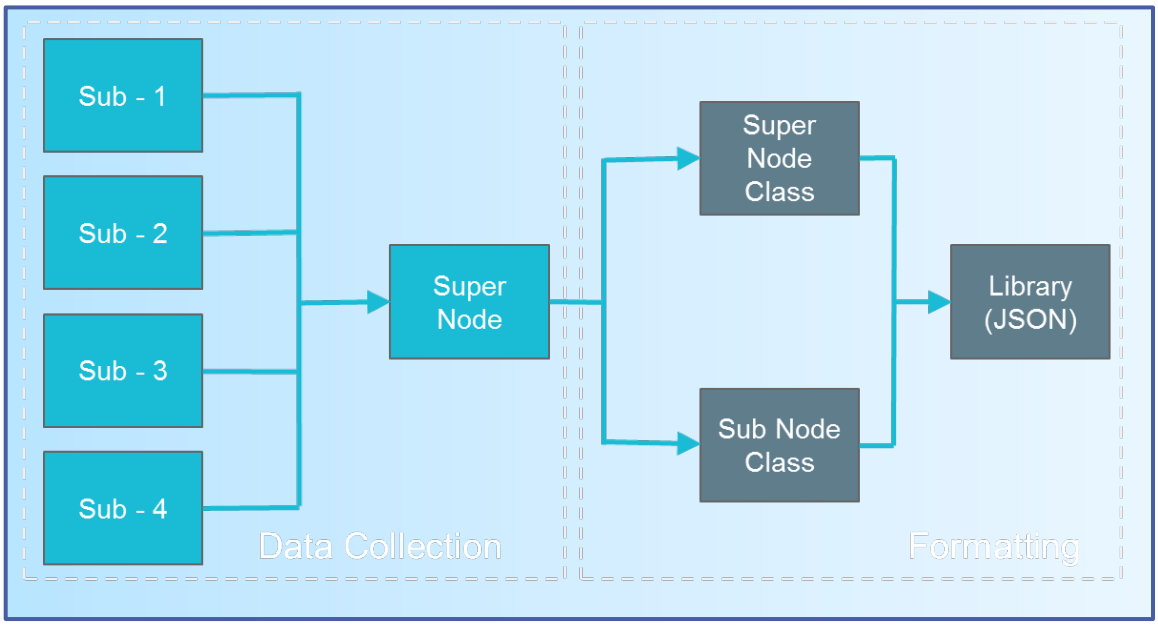


FIGURE 5: DATA FLOW DIAGRAM

This is then used by WETNet to pull current & previous data for its various visual representation and graphing functions. However, while this JSON file is very useful for website integration, it is actually untranslatable. Therefore, another executable was developed & setup on the server. Each occasion where the Master.JSON is updated also generates a CSV file based on the newest data. This CSV contains all the data up to the current point in time and overwrites the previous file. A CSV file is a simple excel representation of a large amount of data, it is easy to read & analyze. The development of this executable was done by taking the keys, node names, from the Master.JSON dictionary and using them as "rows" for the excel table. These node names were dictionaries themselves, and by taking the keys of each node, time stamps, WETNet developers were able to obtain the "columns" for the excel table. The actual data is then cross referenced between the node names and time stamps within the JSON dictionary and formulated onto the excel table.

Server:

Every few minutes the Sub Nodes sends a single instance of measured data to its corresponding Super Node as a hexadecimal value. This single instance of data that the Super Node receives along with the data measured from the Super Node itself is then concatenated into a Java Script Notation file (.JSON). JSON is a lightweight data-interchange format that allows for flexibility and ease of processing by various

programing languages. After a Ten-minute span, this JSON file is transferred over Wi-Fi to the server and is ready to be processed by backend executables.

A backend executable was setup on the server to concatenate the JSON files before they are overwritten. Every interval the JSON file was overwritten, the executable would begin processing & concatenating the new JSON file. The script began by decoding the base64 string, from the Super Node, into a JPG image. Once the image had been decoded and saved onto the server, it would be removed from the JSON dictionary thereby reducing the size of the overall JSON file. The timestamps, representing the keys, and the data, representing the values, from the Super Node.JSON files were then concatenated into a larger Master.JSON file on the server. The server would then wait until the two Super Nodes transferred & overwrote the next Super Node.JSON files and then repeat this process.

The network also builds a management line. This allows users to login and maintain the system over the internet. Both the data and management connections are automatically built when the system connections to the local available Wi-Fi connection. If the system stops transmitting data it will still be able to be logged into and upon reboot will be brought back into operational conditions.

Super Nodes:

The Raspberry Pi 3 is WETNet's hardware connection, consolidating the data in order to be further processed for the network. The code on the raspberry pi is designed to get sub node and local super node data to the server in a clean and efficient manner while also allowing the system's operator to monitor the status of the weather sensing network. The super node is in the center of all the hardware devices and will be the primary device bringing the data values to the database library. The library contains all the values for the respective sub node at a specific time. It is constantly updated so that multiple data can create a large data set.

FIGURE 6: SUPER NODE BLOCK DIAGRAM

Two Python object classes were created to organize the program, a sub node class, and a super node class. The super node class stores the data collected by the super node. The sub node class stores the data collected by the sub node. The sub node class also stores data associated with specific Sub Nodes in the previous ten minute read cycle.

Upon startup the program initializes the super node class instance and local sensors connected to the raspberry pi. The program then initializes multiple instances of the sub

node class and assigns each instance its node number. There is one sub node instance per sub node deployed in the field. After class instances, the main program initializes and starts the raspberry pi's UART serial communication with the super node PIC microcontroller. The program waits for pairing messages from all Sub Nodes before moving on to its steady state operation.

In steady state operation, the raspberry pi awaits data from the super node pic. It checks the data packet for corruption and logs the incoming message in the log text file along with a time stamp in relation to the total read cycle.

- If the packet is not corrupted, the program checks to see if the data corresponds with local sensor readings or sub node readings.
- If the data came from a sub node, the program will forward the data to the sub node class were data is stored as an attribute of the sub node object.
- The sub node object will also set a flag notating that that particular sub node sent in data. This is necessary in determining if any sub node has stopped operation for any reason.
- If the incoming data is from local sensors, the program will store this information in the super node class, read sensors connected to the raspberry pi, and end the read cycle.
- When this occurs the program checks to see if all of the Sub Nodes have been read.
- If a sub node has not been read, this is noted in the error log so an operator can repair the sub node. Once the check is done, the node class instances return data vectors from their respective classes which are then stored in a time stamped dictionary.
- Once the dictionary has been built, it is transferred into a JavaScript Object Notation (JSON) file. JSON is compatible with many languages such as Java, JavaScript, C, Python and others. JSON is built on collections of data, or can be referred as an object, which is ideal for our dictionary. This makes it easier to transfer the JSON file through the server and eventually into the network, where the data will be analyzed.
- Once data has been saved, the file is then copied onto San Diego State's Volta server where data is consolidated into a master data set and read by the website to be displayed and analyzed by the user.

FIGURE 7: NODE PLACEMENT

The sub node data is converted from I2C to floating value in the microcontroller alone. During the first ten minute interval, the first set of floating values will be transmitted as a string which is transmitted through the serial port of the Raspberry Pi and will then need to be separated. Incoming messages are parsed by the Python command split(). We have implemented the split function by calling it anytime a space is found in the string or split(' '). Once the line of string has been separated the floating values will be stored as variables that are assigned to each data. These variables will be called in the sub node class, providing an organized format. The sub node class makes it easier to link the data to the specific sub node and later into an array. The array will contain the node number, the temperature, the humidity and the UV index. The end purpose is to store the array of data, associating with the appropriate time stamp in the library. For instance, a string "X 2 78.65 53.4 1.25U" is received in the Raspberry Pi. "X" is the start character, "2" is the node number, "78.65" is the temperature, "53.4" is the humidity, "1.25" is the UV index and "U" represents the end of the line. As seen in the string, there are spaces separating each data. the split function will separate the data accordingly, which then passes through the sub node class and populates the formatted array in the library. The necessity of "X" and "U" is to differentiate the data values for the respective sub node, since the data will be coming in from the four Sub Nodes for each 10-minute time interval.

The super node sends the data through the microcontroller and also from direct connection of the sensors to the Raspberry PI. The air quality, wind speed and direction

are attained from the microcontroller. However, the temperature, humidity, barometric pressure, UV index and the pictures taken from PI Camera will be sent directly to the Raspberry PI. These data values will be consolidated in the super node class. The super node class contains function that converts the sensors from I2C value to float values. The Pi camera will store a picture as a JPEG image and convert it to a string. All these data are formatted in an array and populated in the library in the following order, temperature, humidity, UV, pressure, wind speed, wind direction, wind gust, picture and air quality. The super node class provides an organized format to make it easily accessible when needed to be processed for analytical data.

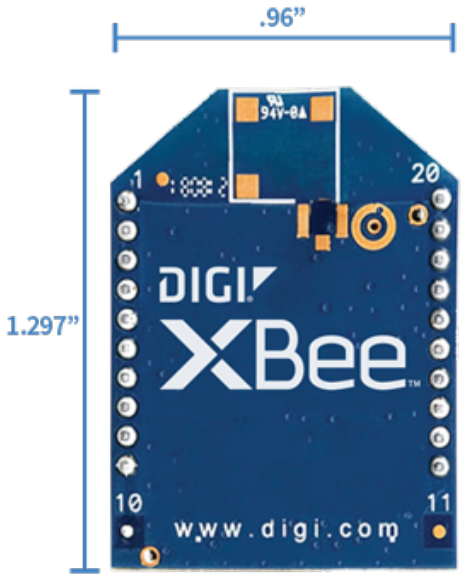


FIGURE 8: XBEE RADIO

WETNet uses the Xbee S2C radio, which utilizes the Zigbee protocol. Zigbee is used to create personal area networks which implement small, low power radios. This protocol fits the design perfectly, as it aims to measure the weather data of microclimates. Other applications include home automation and medical device data collection, which are similar in scale.

WETNet's communication protocol is designed to be as user friendly and easy to set up as possible. To achieve this, dynamic node assignment is utilized. Based on the current model, there is one super node and four Sub Nodes per cluster. As the super node gets turned on, it looks to pair with the Sub Nodes by sending a pairing message. As each node is turned on, it responds to the pair message sent out by the super node, and assigns itself a node number. This node number corresponds to its time slot, or bin, and will send periodically over the length of the total period, which is ten minutes by default. The design also uses a feedback control message, sent by the super node, to confirm receipt of a message and to adjust the timing of the sub node. This adjustment is used to help each sub node stay in its bin, which helps avoid collisions between messages, and corrects for oscillator drift in the microcontroller over the course of a deployment.

WETNet is extremely scalable and customizable for future revisions. The number of Sub Nodes in each cluster can be adjusted to meet various applications, which will help to increase the range of cluster and resolution. The total period, or frequency at which the Sub Nodes send their data, can also be adjusted to increase or decrease the rate at which data flows into the super node. These changes can easily be made by the user, making our system superior in terms of granularity and scalability.

The XBEE S2C has a line of site range of 120 meters or 400 feet, therefore placement of the sub-nodes may vary. Obstructions such as trees and buildings can create packet loss as distance increases. So, by mapping out the positioning of the super-node WETNet can expand from that given position to areas around the campus that will give us 90 to 100 percent confirmed packet transfer. Using the XCTU program developers were able to preview the connection between each XBEE using the range test as well as viewing the packet transfer between XBEE's. To get the best connectivity an open line of site is essential between XBEE's. If there is an obstruction the distance of placement for sub-nodes must be decreased in order to get better packet transfer between the two radio's. One of the sights used was the roof of the SDSU Engineering building, having a high positioning ensures better readings as well as a larger range for the XBEE S2C, from there the sub-nodes are deployed as far as 200 feet, on top of a light post, the automotive building, and GMC building.

SolidWorks was originally used to design two enclosures to house the electrical components to protect against ambient weather conditions. The first draft of the Sub Node and Super Node were determined to be too large. The design needed to shrink to allow for mobility and compactness to be key features. After some feedback, another design was constructed but and additional key features added such as the visual representation of parts laid out with the enclosure designed on the 3D CAD software. The choice was made to buy the enclosures. Hammond offered their SolidWorks model files on their website; saving many hours of modeling. The final draft 3D model was constructed with everything to scale to make the visual representation as accurate as possible; furthermore, assisting the design team to visualize new concepts and placements of openings for sensors

An enclosure was created to protect the sensors and PCB board from environmental changes that may occur while deployed. The PCB board consists of a XBEE S2C, Si1145 UV Sensor, and BME 280 Temperature/Humidity sensor fits inside a 1.4x2.2x3.3 Hammond box. There are openings on top of the box to allow for better readings for each sensor. A battery casing of basswood was created for the A 3.7 LiPo battery and to hold the PCB in place. Adding Velcro onto the enclosure, the battery and PCB board can be attached to a variety of surfaces to ensure the best reading and positioning for temperature, humidity, and UV light. With such a small enclosure the WETNet hardware can be easily placed in areas that are the most effective.

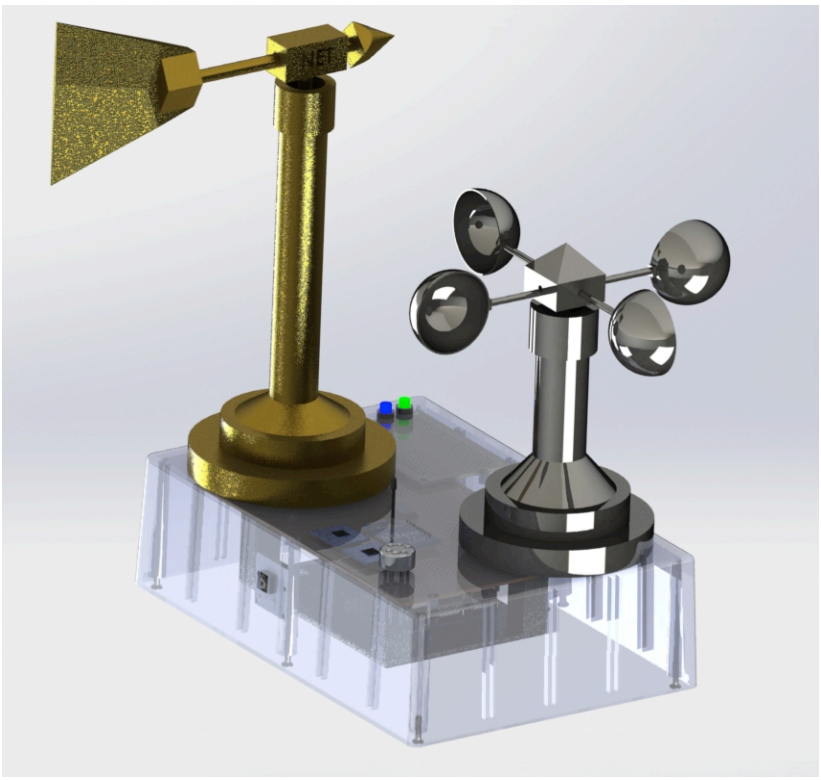


FIGURE 9: SUPER NODE

Wind Vane Overview

The wind vane on the Super Node device outputs the current wind direction in the designated area. There are eight compass directions that the device outputs (i.e., N, SW, E) these will be shown on the GUI with an arrow, pointing in the current wind direction. Instead of purchasing a pre-existing wind vane, WETNet uses a custom purpose built vane in order to get a better understanding of the stability and fluidity that must be considered for accurate data. After researching the standard design for the device, a wind vane base and shaft were designed on SolidWorks then 3D printed. The 3D parts include: the outer-pivot base, the shaft, and the triangular tail. The largest triangular tail possible without interfering with the anemometer was used. For stability purposes two bearings where used, one flanged for the top and a light load ball bearing for the bottom. This compensates for any misalignment or bouncing from the wind vane.

Wind Vane Sensor/Code

The sensor utilized for the wind vane was the AS5048B, a 14-bit angular rotary sensor it has an i2c interface and senses the magnetism from the south pole of a magnet and changes values as the magnet changes its angle. The magnet is attached to the shaft of the wind vane and hovers above the sensor at a quarter-inch. The wind vane is calibrated by pointing the sensor relative to North. With a miniature compass attached to the Super Node, the Node is properly adjusted to its corresponding direction. The Pic24 writes to the angle address, 0xFF, and the most significant bit (MSB)

reads out hexadecimal values from 0x00 to 0xFF. Each direction is correlated with eight hexadecimal values. To get a common mode of direction, over a 10-minute period, eight flags were assigned to each direction. The microcontroller is constantly collecting output values every 5 seconds. The flag with the most increments after 10 minutes is sent as the common direction and is inserted into a string with the air quality and wind speed.

Wind tunnel/Outdoor Testing

To test the fluid mechanics of the wind vane, a wind tunnel was created using a cylindrical case and a fan. The wind vane and anemometer were set at the positions where they will be placed on the super node and checked to see if the wind vortex, created from the anemometer, would have any drastic interference with the accuracy of the wind vane. One test was with the anemometer and the wind vane attached. The second test was just the wind vane. The final results concluded that the wind vane with the anemometer was only off by five degrees, on average, this was found to be sufficient data to keep the positions and structure of the two devices. For practical data, tests were run outdoors on top of the SDSU Music building to correlate wind vane data, by utilizing Putty via UART serial cable, with the actual wind direction in the area. The wind vane's top base was adjusted to implement a long length on the tail for more torque and to counterbalance the weight on both ends of the arrow and tail. With adjustments came more fluidity and stability from the overall device.

Anemometer

The anemometer featured on the super node consist of three major 3D printed components, the outer base, the inner shaft, and the cups. There are a total of three cups seen on the anemometer to provide better fluidity. In addition, at each opening of the outer base there are bearings in order to ensure that the inner shaft is able to spin freely without any friction. At the end of the inner shaft, there is a permanent magnet. When the cups catch the wind they rotate the inner shaft, rotating the magnet. At the bottom edge of the outer base, there is a hall effect sensor. Each time the magnet rotates and passes by the hall effect sensor, the sensor outputs a digital pulse. The digital pulse is sent into the PIC as an external clock source for Timer 3. Timer 3 is being used to count the number of pulses within a certain time interval, in this case every five seconds. After five seconds is passed, the value of the counter is recorded and converted into MPH with the following equation:

$$MPH = \frac{Pulses \times 30 \times 60 \times Circumference}{12 \times 5280}$$

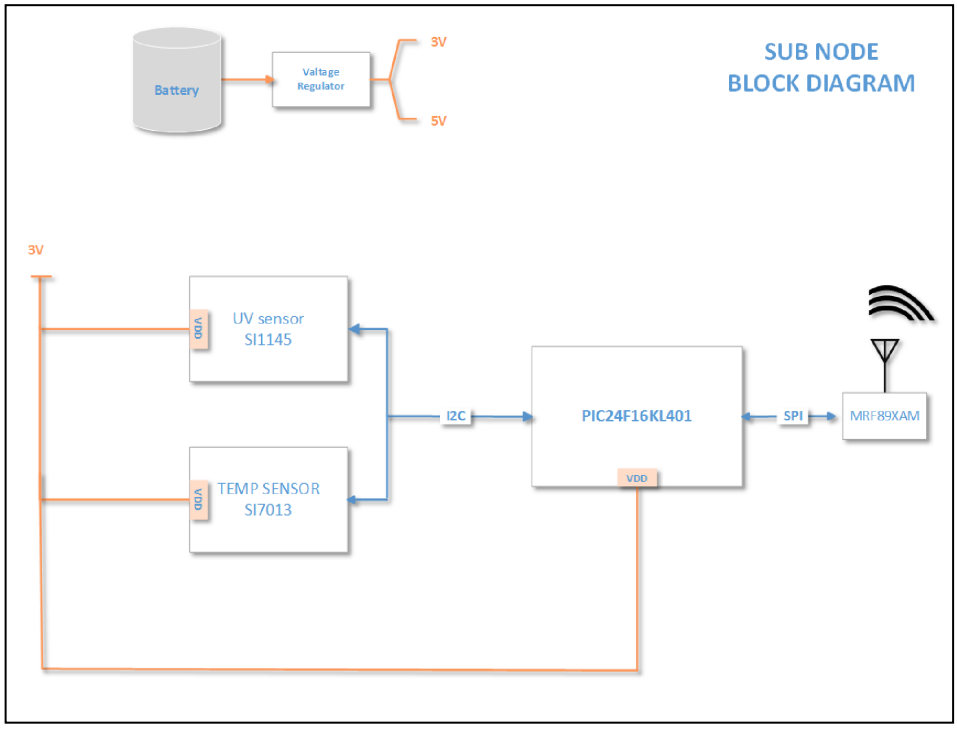
EQUATION 1: MILE PER HOUR

The circumference used is in inches and consists of the inner shaft and

the magnet. The MPH value is then stored in an array. When the super node is ready to send data to the Raspberry Pi, the average of the array is taken. In addition, the code goes through the entire MPH array and looks for values that exceed a certain threshold. If the values are not outliers, it saves these values as gust speed.

Sub-Node

The brain of the Sub Node is the PIC24F16kl401. This pic was chosen due to its two I2C ports, its low power demands, and required peripherals. Both the temperature and UV sensors on the Sub Node use the I2C, and for communication the pic also has a UART port which makes interfacing with the radio and sending data packets very easy. The low power RC oscillator is the clock source for the timer and is 31K Hz which allows WETNet to count to a maximum of 8.5 minutes before overflowing. Fully charged the lithium polymer battery used to power the Sub Node is 4.2 volts and the minimum voltage the battery can go down to before it becomes damaged is 3.7 volts. The pic's I/O pins can only receive a maximum of 3.6 volts. A voltage divider is used to ensure the voltage coming from the battery is never over 3.6 volts. The ADC converts the voltage coming from the battery to a digital number and once the number is below the value for 3.7 volts it sends a message to the error log to change the battery of the specific node. The largest hurdle with the microcontroller is the lack of available code on this specific model, but the pic24F series is very prevalent allowing for the code of similar microcontrollers as a reference. The pic is used to power the temperature and UV sensors equipped on the Super Node so the same code didn't have to be written twice.



The creation of the Printed Circuit Board was vital to WETNet systems functionality. It was necessary to realize conceptual block diagrams into specific schematics and then compile them into a working PCB. Using state of the art software and circuit theory, local libraries with custom parts, partitions and cells were created to address the constraints of a unique design. It

FIGURE 10: SUB NODE BLOCK DIAGRAM

was imperative to integrate two separate PCB boards tailored for specific functionality within the Super and Sub Node models. The Sub Node required the most attention due to compact design constraints and necessary repeatability. The Super Node had more sensors with added surface area allowing for easier routing and an overall less extensive reproduction process.

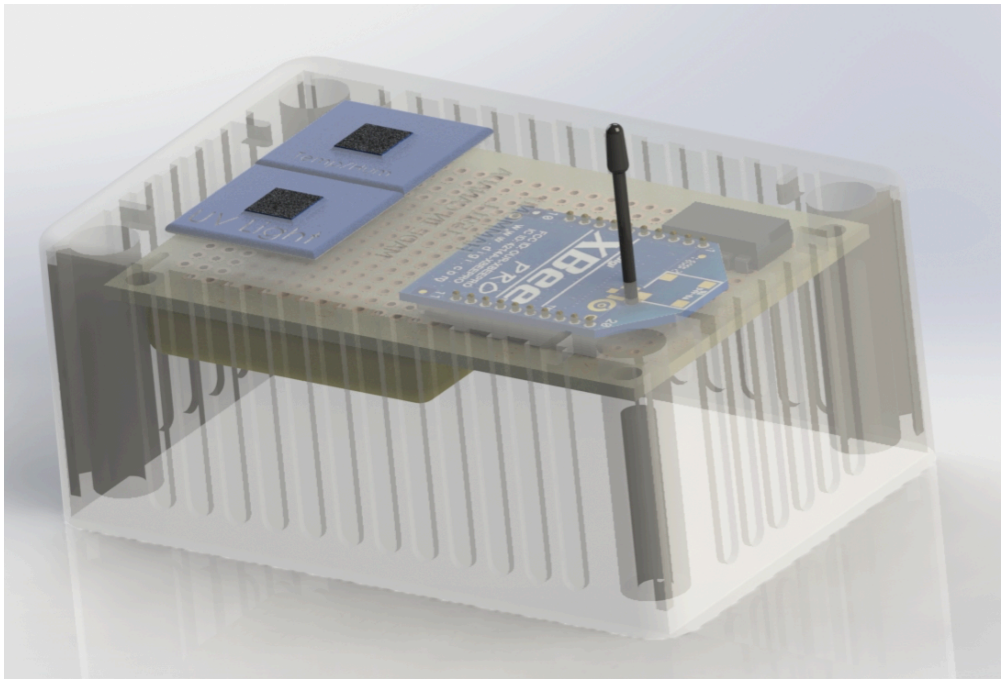


FIGURE 11: SUB NODE

Air Quality

One of the functions that is required of WETNet is the ability to measure air quality. To meet this specification, pollutant sensing capabilities of the TGS2600 are integrated into the system. This sensor was chosen due to its sensitivity to the changes in the amount of pollutants present in the air. The sensor can measure as little as 15 parts per million of a selected pollutant. The air quality sensor can also measure multiple types of pollutants such as methane, carbon dioxide, carbon monoxide, and sulfur monoxide. The final factor in choosing this sensor was the cost. While many other sensors had similar capabilities, this sensor was the best value at half the price of other comparable sensors with more functionality. The TGS2600 Air Pollutant Sensor works by measuring the conductance of pollutants in the air and outputting a corresponding voltage. To calibrate our sensor, various sites were selected and measured using the Air Quality Index scale. Two of the sites used were Del Mar and El Cajon. Del Mar was chosen to represent an area understood to have a “Good” AQI level of 41 and El

Cajon to measure the output in a "Moderate" air quality area. The El Cajon site had an AQI level of 57 when tested resulting in a close-to borderline sample of moderate air quality.

The driver for the air pollutant sensor utilizes the ADC of the PIC24F16KL401 microcontroller. The ADC samples the voltage outputs of the sensor once every 6 seconds and outputs the average of these values to the Raspberry Pi every 10 minutes. The driver finds the average of samples to prevent any outliers from skewing the data significantly. The air quality is then sent to the Raspberry Pi in the form of a hexadecimal number. This hexadecimal number represents the air quality on the Air Quality Index scale where a 0x0001 represents "good", 0x0010 represents "moderate", and 0x0100 represents "unhealthy for sensitive groups". The data will be formatted and displayed on the WETNet website and updated every 10 minutes alone with data from other sensors in the network.

Sub Node	Standby Current (mA)	Standby Time (s)	Active Current (mA)	Active Time (s)	1 Hour Consumption (mAh)
Microcontroller	0.0003	5999.994	0.15	0.006	0.0003
UV Sensor	0.0005	599	0.009	1	0.0005
Temp/Humidity	0.0000005	599	0.00035	1	0.00000108
XBEE Transmit	0.00036	99%	45	1%	0.4504
XBEE Receive	0.00036	99%	31	1%	0.3104
Total Consumption					0.762

TABLE 1: POWER USAGE

The power consumption for the Sub Node is calculated and illustrated in the table above. There have sub temperature sensor BME280, UV Index sensor SI1145, PIC24F and Xbee in the Sub Node. And the table above which are the active current and standby current which we can find from the datasheet. By multiply by the time for each of them, we can get the total consumption which is 0.762mAh. And we choose a 3.7V 380mAh battery which can provide 4.0 voltage once it's fully charged. And this battery we use right now can let the Sub Node works for 498 hours which equal 20.7 days. In order to get the voltage work for each sensor and pic, we need a voltage regulator KY5033 to regulate the 4v voltage to 3.3v which we need. In addition, we use the capacitors both Vout and Vin to make sure there have stable current through the devices at anytime.

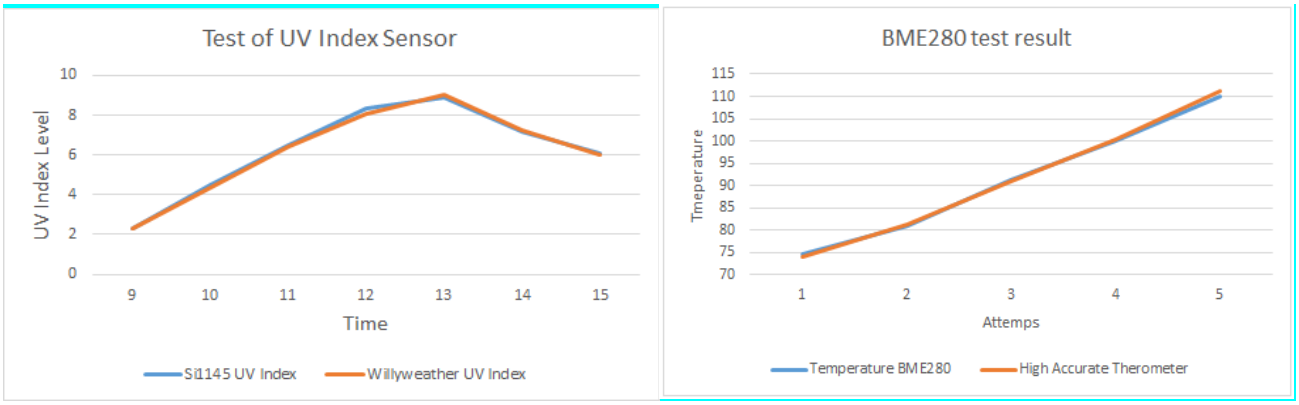


FIGURE 12: SENSOR TEST RESULTS

Comparing the testing result from the Si1145 UV Index sensor with data from Willyweather online. Results in a table as above. In this instance showing data between 9AM and 3PM. the UV Index is collected in the same location at each hour and the average value for each hour will result in a more accurate comparison. A thermometer for the BME280, is used to measure the temperature by using dryer. This testing shows that the equation from the datasheet from WETNet does work accurately and efficiently and the I2C also works efficiently.

Data Connections:

I2C protocol allows communication of data between I2C devices over two wires, which are SCL and SDA respectively. A master-slave protocol was used, the microcontroller (pic24f16kl40) initiates communication and drives the clock. Slaves in this instance are the temperature and UV sensors. Each device has its own address. For example, the 7-bit device address of temperature sensor is 0x76 and UV sensor is 0x60.

I2C will only read an 8-bit address. Thus, every device address used need to left shift for one bit. The master will send the address of the respective slaves and R/W bit, followed by other data. Then the particular address will be activated and respond to the master. It is crucial that resistors are pulled-up on those two lines, to make sure that when SDA is not operating, the SDA line is ideal, which needs to be pulled up. There are several functions of I2C that are required:

Functions	**Behavior**
Start condition	SCL=1, SDA falling edge
Stop condition	SCL=1, SDA rising edge
Acknowledge	SCK=0, SDA=0

The Graph below illustrates I2C waveforms

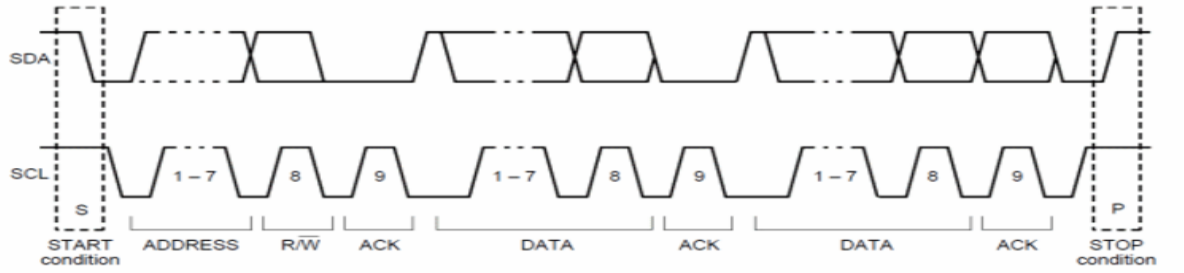


FIGURE 13: I2C CHART

There are two different situations used in the I2C communications, respectively, reading and writing. Reading is used to read values or parameters of devices and writing is used to pass command to register in some particular cases. Here is sequences of writing and reading.

Writing	Reading
Send a start sequence	Send a start sequence
Send the device address with R/W bit low	Send the device address with R/W bit low
Send the internal address	Send the internal address
Send the data byte	Send a repeated start sequence
Send the stop sequence	Send the device address with R/W bit high
	Send the stop sequence

TABLE 2: I2C WRITE AND READ SEQUENCE OF COMMANDS

MILESTONES

The overall budget for the Microclimate Weather Network project was $1200. Illustrated in the table below, WETNet was accomplished with an approximately $70 surplus. Components (sensors, cameras, and radios) were a major factor in monetary assignment. The remainder covered testing, batteries, microcontroller devices, etc.

Final Cost:

Category	Budget Spent	Estimated Cost	Amount Remaining
Sensors/Camera	$ 427.23	$ 430.00	$ 2.77
Radios	$ 205.03	$ 240.00	$ 34.97
Microcontrollers	$ 145.00	$ 145.00	$ -
Battery/Power	$ 70.00	$ 70.00	$ -
Testing	$ 102.98	$ 102.98	$ -
Misc/Tax	$ 157.69	$ 162.02	$ 4.33
Box Enclosures	$ 49.66	$ 50.00	$ 0.34
Total:	$ 1,157.59	$ 1,200.00	$ 42.41

TABLE 3: BUDGET

WETNet we intended to accomplish the following milestones. Shown in green are the specific tasks which were completed on the dates predicted while the tasks shown in red were ones that were more difficult and needed more time to achieve. Tasks that were not finished on time such as working prototypes of the sub node and super node and the testing of the data transference between the two devices were a result of miscalculation which were initially thought to be needed. For example, in order to have built custom parts using the Mentor Graphics program it was required for the designers to learn to create individual symbols and cells which could then combine together to form the single component. Ultimately, the design team was able to learn about the process and the perseverance required for the design and production of fully functional devices, from concept to deployment.

Date:	Milestone:
March 15, 2017	Volta server uses secure copy to receive .json file from the Super Node

March 17, 2017	Successful test data transfer from Super Node to server
March 19, 2017	Working prototype of Sub Node using break out boards.
March 22, 2017	Hello World sent from Sub Node to Super Node
March 24, 2017	Successful test data set sent from Sub Node to Super Node
April 1, 2017	PCB prototype of the Sub Node
April 7, 2017	Hello World sent from ALL Sub Nodes to Super Node
April 13, 2017	Day of field data displayed on website

TABLE 4: MILESTONES

Volta server uses secure copy to receive .json file from the super node – March 15, 2017

The purpose of this milestone was to ensure that we would be able to connect super node to the server and then store data on the server. This milestone was important because the volta server is the connection between our weather sensing network and our website. If we were not able to achieve this milestone it would mean that we would have to redesign our system so that we were not dependent on SDSU's volta server. We were able to meet this milestone on time and therefore did not have to redesign our network.

Successful test data transfer from super node to server– March 17, 2017

The purpose of this milestone was to ensure that we would be able to transfer a data set to the server in a timely manner. If the file transfer from the super node to the server took too long due to the size of the file being sent we would have to find a more efficient way to send data to the server. Since we were able to meet this milestone we

did not have to spend resources researching more efficient ways to transfer our data set to the server since our current method was sufficient.

Working prototype of sub Node using break out boards– March 19, 2017

The purpose of this milestone was to ensure that the sensors, microcontroller, and radio were all compatible with each other and to start debugging the errors associated with implementing all of the code. Two issues were the main reason this milestone was not met. The first issue was the temperature sensor. We could read the slave address of the sensor with the pic but could not read the data coming from the sensor with the pic. We tested the sensor with arduino and got everything to work, but kept facing the same problem when using the pic. After a couple days of unsuccessful testing we switched from the SI7013 sensor to the BME280, which is used by the super node. We switched to the BME280 because even though it is more expensive we couldn't dedicate anymore time to debugging the issues associated with the SI7013. The next major issue was sending the data through the original radio. Issues with the hardware on the radio's breakout board and overly complicated communication protocol delayed the completion of the sub node prototype.

Hello world sent from sub node to super node– March 22, 2017

The purpose of this milestone was to ensure that we could get basic communication working between a sub node and a super node. We were not able to meet this milestone because the original radios we chose for our system used a communication protocol that proved to be overly complicated for our use. We were able to complete this milestone by switching to radios that used the zigbee communication protocol. This protocol proved to be much easier to work with and we were able to get hello world up and running within two days of having the radios in hand.

Successful test data set sent from sub node to super node– March 24, 2017

The purpose of this milestone was to ensure that we had all of the sensors on the sub node programmed and working and that we could parse the data on the super node. We were not able to meet this milestone because the drivers for the sensors were not yet written and because we needed to change the radios. We were eventually able to meet this milestone and get a complete data set on to the super node from the sub node.

PCB prototype of the sub node complete – April 1, 2017

The purpose of this milestone was to give us a goal to complete while on spring break and give us a couple of weeks to focus on the super node PCB. This milestone

was not completed on time because there were more issues with the PCB and we couldn't print a new revision until school started. The biggest issue that we didn't think about was not being able to print further revisions until school started back up on April 3rd. There were three issues with the first PCB that was created, but both were easy to fix. The first issue was the voltage divider that is supposed to be connected between the battery and the ADC of the pic was instead two resistors wired in series. This issue just required the PCB to be retraced in mentor graphics. A more crippling issue was the pins to program the pic from the pickit were incorrect. This issue was not caught when we reviewed the schematic before the PCB was laid out, but it was also an easy problem to fix. The last issue was no ground plane was added to the PCB and we had to wait until April 3rd to learn how to add the ground plane. The issue with the first revised board was the pic wasn't properly grounded. We got the third revision to work properly but we had to add a power capacitor to keep the voltage line stable. The fourth revision worked when properly soldered but sloppy soldering jobs caused components to get shorted. We were eventually able to complete this milestone, but we are still having issues with some of the PCB's not working correctly, the biggest issue is getting the radio to pair with the super node.

Hello world sent from all Sub Nodes to the super node – April 7, 2017

The purpose of this milestone was to test our communication protocol and ensure that we could get data from multiple Sub Nodes without collision in the network. We were able to meet this milestone and were able to get a sting from each sub node to ensure that we could get data to the super node. We were unable to send real data to the super node because at the time we did not have enough sensors to make a complete system.

Day of field data displayed on website – April 13, 2017

The purpose of this milestone was to ensure that our entire system was operational from Sub Nodes to website. We were unable to meet this milestone because the Sub Nodes were not built. At the time of this milestone we were however to display multiple days of fake test data on the website and were also able to build data sets on the server.

Conclusions and Recommendations:

Conclusion:

Our design aims to ease utility decision-making by focusing on the granular analysis of San Diego State University's campus weather environment. Although the implementation is limited to short-term deployment, our network has the capacity to evolve into a wide operational weather-sensing network that is easy to deploy near a Wi-Fi setting. Our network also demonstrates the challenges associated with a

microelectronic approach and highlights where dedicated efforts should be allocated to make this a viable option for companies similar to SDG&E.

Recommendations:

When building our system there were many considerations and challenges in terms of design priority. Our biggest limitations came from budget and scope. Since the project targets the university we were constrained to functionality and ultimately a working prototype. This means that the fixation on a wide component list is not feasible in the 15-week time frame, especially for the number of team members. The best expression of this particular design constraint is the radio and micro-controller selection. Our initial radio selection, while better suited for our conceptual implementation, proved to be very un-user friendly. This led to wasted hours with many people unable to initialize the system to an operational state. This led to the redesign of a more robust, user friendly, and power hungry solution.

Our biggest recommendation is to focus highly on more specific problems or concepts. For example, if the project had focused entirely on a radio, antenna, and protocol optimization for the start topology we could have found very specific research as to solutions for SDGE's potential use cases. This would allow for either better exploration of potential SDG&E test cases or for a more general solution to be used by other universities or businesses.

References:

Micro-Controller Notes:

http://www.robot-electronics.co.uk/i2c-tutorial

http://www.best-microcontroller-projects.com/i2c-tutorial.html

http://www.best-microcontroller-projects.com/i2c-tutorial.html

http://www.best-microcontroller-projects.com/i2c-tutorial.html

Secure Copy and SSH Notes:

https://www.raspberrypi.org/documentation/linux/usage/cron.md

http://stackoverflow.com/questions/50096/how-to-pass-password-to-scp

https://unix.stackexchange.com/questions/132326/initiate-ssh-connection-from-server-to-client

https://www.howtoforge.com/reverse-ssh-tunneling

Appendices:

See digital drive