Mobile Data Harvester
Senior Design Project

Prepared By:

Taylor Jones
Abraham Palmerin
Joshua Papanicolas

Faculty Advisor: Dr. Farid Farahmand
Industry Advisor: Sean Headrick
Client: Dr. Chris Halle

May 3, 2017

www.dataharvester.weebly.com

Abstract

A key component in preserving and understanding the environment and its ecosystems is the ability to collect environmental data continuously and reliably. Collecting environmental data using passive data loggers and manual downloading can be a daunting task. Today, for many scientists the application of various environmental sensor networks has become the standard research tools for data collection. However, these networks are often unable to cover a wide geographical area. Furthermore, implementing massive environmental sensor networks without damaging the ecosystems in order to provide connectivity and power to the sensors can be challenging.

Over the last several years the Unmanned Aerial Vehicle (UAV) technology has been widely proposed and utilized in commercial and civilian activities, particularly in the areas of remote environmental monitoring, sensing, and mapping. For example, UAVs are being used for acquisition of remotely sensed data and imagery. In such applications the UAV carries a variety of small airborne sensors such as cameras and Lidars.

In this work we discuss the hardware and software design of our proposed UAV-based Mobile Data Harvester and its ability to reliably and continuously collect and transfer environmental data.

The Mobile Data Harvester consists of multiple solar-powered ground-based stationary sensor nodes and a mobile node that can be mounted on a UAV. Each low-cost low-power stationary sensor node is capable of logging the environmental data (e.g., temperature, soil moistures, humidity). Once the mobile node reaches its pre-programed GPS location via the UAV, it requests data transmission from the stationary node. Consequently, the logged data is transferred via ZigBee wireless protocol and eventually removed from the stationary node.

In this paper we also report on key performance factors of the Mobile Data Harvester, including aerial position accuracy, data transmission time, power consumption, and the vertical operating range of the wireless communication. We conclude the paper by presenting potential ideas to improve and enhance the overall system performance.

Contents

List of Figures

List of Tables

1 Introduction

In many fields that make use of sensors to gather data, there is a need to place sensors in remote areas. In some cases these places are hard to get to, dangerous, or cause a hassle to the person who needs to get there by foot or vehicle due to its remote location. A few examples include, gathering data from tree tops, mountainous regions, and getting data from deep forests. There must be a way to access these secluded areas more than once without the hassle of having to continuously go back just to get some data. We are proposing a system that will allow the user to gather data from such sensors in a much more convenient and safer manner. This system will employ low power remote sensors and will be durable enough to withstand wilderness and weather conditions. These sensors will gather a multitude of data like temperature, precipitation, air quality, etc. The data will be logged and then extracted by the user wirelessly through a mobile platform. This platform could be applied to a multitude of different modes of transportation, but in the case of this project we will be mounting it onto a unmanned areal vehicle (UAV).

1.1 Literature Review

Similar Products and Ideas:
For comparisons between these and our project see methodology.

1. **Remote Sensing Using UAVs**

   Similar concept except they use the drone as a wifi relay to stream live data from the sensors. [4]
   https://people.kth.se/ gonga/remoteuav.html
2. **RFID-Reading Drone Tracks Structural Steel Products in Storage Yard**

   Steel company uses drones and RFID to track their stock. [2]
   http://www.rfidjournal.com/articles/view?12209
3. **Drone Scan**

   Company makes drone systems involving RFID and barcode scanners (mounted on the drone) to help warehouses track their stock. [1]
   http://www.dronescan.co/
4. **Satellite Connected Sensors**

   Alternative idea where instead of using drones to gather the data from the sensors they use a satellite link to directly gather the data from remote sensors. [9]
   https://www.technologyreview.com/s/538726/nano-satellites-work-with-ground-sensors-to-offer-new-eye-on-disaster-relief-and/
5. **Sensor Buddy**

   Similar project done at this school. Our project is essentially an extension of this. [10]
   spott79.wixsite.com/spott-weiss

1.2 Problem Statement

As stated before, a major issue with placing sensors in remote areas is ease of access. If one wanted to, for example, measure average temperatures deep in a forest, on a glacier, or near a volcano, gathering said data could not only create unnecessary extra travel time for the user, but could even be dangerous. A better method than walking to each sensor and manually gathering the data would be to have it done automatically from a distance. One could have the sensors connected to a network that is connected to the internet allowing for constant access, but if the sensors are in a very remote location that might not be an option due to cost, lack of access, or the slow speed of satellite connections. One could also create their own network through the use of multiple relays that eventually connect to a central station, but if the sensors are placed in a very wide spread configuration and in a very remote location, the number of relays required could be make the system too costly and complicated to set up and maintain. Lastly, if one simply used a long range radio transmitter, while the network would be simple, power required would be too much for a small battery and solar panel.

1.3 Methodology

We propose to solve these problems through the use of a short distance connection on a local network. The sensors will be connected to a network of wireless nodes (we'll call these "sensor nodes"). Each of these sensor nodes will be taking samples of data over a user defined interval (we will be doing 30 minute intervals for the demonstration) and saving the data onto a SD card mounted to the device. The data will include a variety of data from different sensors mounted to the node, a time stamp for each sample, and a unique label corresponding to that node. A UAV will be controlled remotely and fly over each of these nodes. A node mounted on the UAV (we'll call this the "mobile node") will send a radio signal (via the ZigBee protocol) that will activate the sensor node which will in turn send its data back to the mobile node via another ZigBee radio signal and then deactivate. The mobile node will use GPS to tell its location so that it can know if it is at the right location before starting the transmission to help ensure a strong connection is available. After obtaining the desired data, the UAV can then travel back to its starting location where the user can retrieve the data from an SD card and upload it onto their computer via a program that we will provide.

The project is different than its competition for a variety of reasons. One major advantage of our system is that the modular nature of it allows for it to be used with platforms aside from UAVs and could be attached to any mobile system whether it be a car or even a person. Most UAV data gathering systems of this kind simply use visual techniques (eg various IR or hyperspectral cameras that are simply strapped to the UAV itself) [2][3][6], while our system is compatible with a variety of different sensors (ideally we would like this to be a modular system that could be implemented with any kind of remote sensor)

and uses sensors that are always on site. It is also possible to make the system completely autonomous (aside from the user's initial placement of the sensors).

To address the question of why not just use a satellite connection on each sensor? Such systems exist, but as said before, tend to be costly due to the need to access a satellite (where one would likely need to pay a fee to use). The implementation of low orbit nanosatellites have been used before as well, but such systems can be even more costly to implement and even more so it were to fail (the nanosatellite breaks)[9]. Our system would be a one time purchase that could be used many times over and will be easier to repair if something were to go wrong (a sensor goes down or the UAV crashes), thus it would be a cheaper alternative. It is also independent of any outside networks and thus can be more reliable since the user is using their own network.

As mentioned previously, the use of wireless relays are a common method to gather remote data. One of the examples from above even expands on this concept by having a UAV that acts as a temporary relay [4]. Unfortunately, these systems still have the issue of becoming overtly complicated and costly when a large number of sensors are placed in wide spread and very remote areas.

Lastly, in comparison to a previous project done at our school that heavily inspired our project, there are many key differences and improvements that we have performed. One is that their system was designed around the use of having their equivalent to the mobile node in a backpack to be carried around, while our system is designed around being mounted onto a UAV and thus has to be much more compact[10]. Then, their system used only one sensor per node, while ours works with multiple, different kinds of sensors. Lastly, our system has been tested for multiple sensor nodes (3), has autonomous capabilities, and uses GPS proximity detection to better ensure a good connection is possible before transmission[10].

1.4 Challenges

We must keep our circuit board to a limited dimension so it can be attached to the UAV without interrupting its balance. Our board must be able to distinguish which sensor is which and where that specific sensor is located. We must maintain power efficiency throughout the process so our sensors will only be activated when the UAV is nearby and when taking a data sample. There will also be a system that will allow the UAV mounted system to make note of unresponsive sensor nodes and report this information to the user. We must maintain a distance of under 300 ft between our board and nodes. The nodes will also be about 100 ft. from each other. The packets being sent should not be so large that it makes the travel time to gather the data too long, since for the case of using a UAV we are limited by its battery life. The implementation of an SD card to a microcontroller system can be difficult. Also, the communication between the Xbees in the manner we are using them can be complicated.

1.5 Marketing Requirements

(MR1) Guaranteed battery life up to 30 minutes for the mobile node on the UAV. **(See test 2.2.5 and test 2.2.4)**

(MR2) User will need an FAA license to operate the UAV. **(see SSU UAV policy)**

(MR3) Sensor nodes will support multiple sensors (soil, humidity, and temperature). **(See test 2.3.14 and test 2.3.18)**

(MR4) Can reach areas that are difficult to reach by foot. Up to a 90 meter radius. **(see test 2.2.7 and test 2.3.17)**

(MR5) Flag is initiated when a certain sensor is unresponsive. **(See test 2.2.13)**

(MR6) User can receive data directly from the mobile node SD card. **(See test 2.2.8)**

(MR7) Overall data information will be uploaded through a Matlab program displaying: node ID, sensor data, time, date, GPS coordinates. **(see test 2.2.15)**

(MR8) Sensor node batteries do need to be manually recharged or replaced. **(see test 2.2.11)**

(MR9) Node will be placed in weather-proof casings to protect them from damage and will be durable enough to withstand outdoors conditions. eg rain, etc., (won't necessarily withstand tampering from wildlife such as bears, etc.). **(See future work)**

(MR10) Both sensor and mobile nodes (without the sensors) will cost under $60 per node. **(see cost calculations)**

(MR11) Data accurate to three significant figures. **(See test 2.2.6)**

(MR12) The system must be able to function over long periods of time without maintenance (months) **(See test 2.2.16)**

1.6 Engineering Requirements

(ER1) 5 volts supplied by the power supply of the UAV to the mobile node. **(See test 2.2.5 and test 2.3.17)**

(ER2) Data will be transmitted and received up to 300 feet. **(See test 2.2.1 and 2.2.2)**

(ER3) Data is transfered to the user's computer by use of an SD card. **(See test 2.2.8)**

(ER4) Will use ZigBee wireless protocol for low power consumption (will have low data rate of 250 kbps). **(See test 2.2.9, test 2.3.17 and test 2.3.18)**

(ER5) Can support at least 3 sensors at a time on each sensor node. **(See test 2.2.14 and test 2.3.18)**

(ER6) Mobile node PCB will be 3x 3 so it can be placed easily on a medium to large sized UAV. **(see test 2.2.10)**

(ER7) Mobile node will weigh no more than 5 ounces.**(see test 2.2.10)**

(ER8) Sensor nodes will be powered by solar charged batteries. **(see test 2.2.11)**

(ER9) Mobile node will use a GPS module to know when to receive data and also check to see if a sensor node is being unresponsive. **(See test 2.2.12, test 2.2.13, and test 2.3.18)**

(ER10) Each Xbee module will have attached 900 Mhz band rubber duck antenna (RPSMA connector). **(see test 2.2.7)**

(ER11) The size of the data packet size must be restricted so that transmission doesn't take too long or the data takes up too much space in the sensor node's flash memory (around 32 bytes per data set) **(See test 2.2.3 and 2.2.5)**

2 Implementation

Our Mobile Node consisted of a Digilent WF32 board with an on board micro SD card module. The mobile node is attached to a UAV traveling to each Sensor Node in order from Sensor Node 1 to Sensor Node 3, it must travel in this order in order to prevent crosstalk and confusing the Mobile Node. Data transmission will not begin until the Mobile Node is within the specified GPS coordinate radius of the specific Sensor Node. Once within range, the Mobile Node will transmit a flag to the Sensor Node telling it to transmit all of its data saved onto its SD card. The Sensor Node consists of a Digilent Uc32, which is a lower end variation of the WF32 chosen to help conserve power. It is powered by a solar panel and battery. Once the data has been confirmed to have been received, the Mobile Node sends another flag to the Sensor Node telling it to clear its old data. Once the Mobile Node receives data from all three Sensor Nodes, the UAV then return to the user who can retrieve the SD card and put it into their computer where a provided Matlab program will organize the data and tell them if anything went wrong. The program also allows them to program the GPS coordinates of each Sensor Node into the Mobile Node.

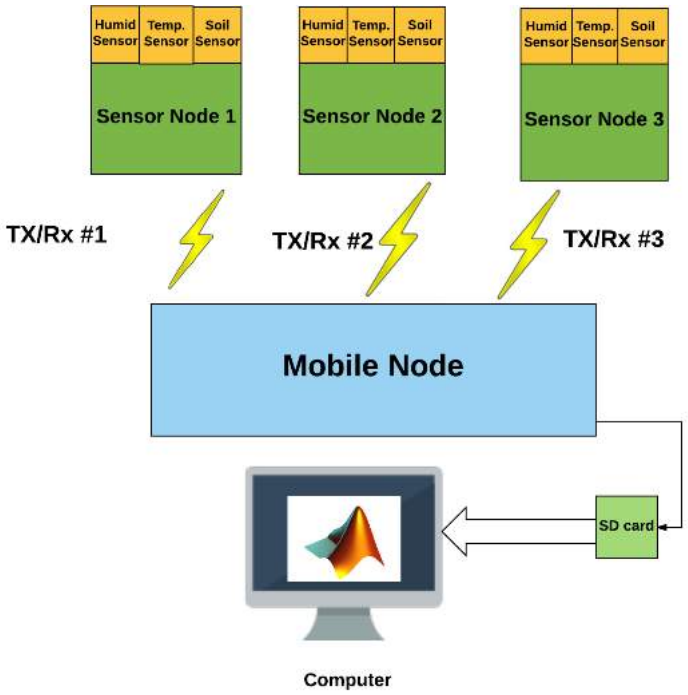


Figure 1: Overall Functional Diagram

Node Number	Date/Time Number	Temperature	Soil Moisture	Humidity
1	2017/3/14 11:18:25	64.32	322.43	55

Table 1: Example of the packet being sent by the sensor node

Node Number	Date	Time	Temperature (C)	Soil Moisture (mm)	Humidity (%)	Latitude	Longitude
1	11/27/2016	11:45 PM	64.32	322.43	55	51.434491	-3.789107
1	11/27/2016	12:45 PM	64.12	321.1	55	51.434491	-3.789107
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
2	11/27/2016	11:40 pm	63.57	302.67	53	51.425672	-3.760387

Table 2: Example Excel layout the client will be able to view from the collected data the mobile node received

2.1 Design Approach

Each node was programed using C++ using a modified variation of the Arduino IDE for Digilent products.

The Mobile Node starts by waiting until it is within the preset GPS radius of the first Sensor Node. Once it is, it broadcasts a flag until it starts receiving a data packet. If it doesn't get anything for over a minute, it starts looking for the GPS location of the next Sensor Node. Once it starts getting data, and there are no interruptions (if there is a gap in the data transmission it starts sending flags again and will continue where it left off in keeping track of the one minute time limit), it will keep reading data until it receives a flag indicating that the transmission is complete. It will then send one last flag back indicating a successful transmission, save the data onto its SD card, and start looking for the GPS coordinates of the next Sensor Node until it has done so for all three nodes. After which it will go into an idle state.

The Sensor Node idles until a half hour has passed, after which it will read a sample from each of its sensors and record those values onto its SD card. It will continue to do this indefinitely. At the same time, it is constantly looking for a flag from the Mobile Node to begin the transmission. If it does, it will start sending all of the data it has stored and send a flag at the end once it has finished. It will then wait until it either receives the previous flag again for retransmission, or the flag indicating a successful transmission after which it will delete its old data and go back into an idle state taking samples every half hour again.

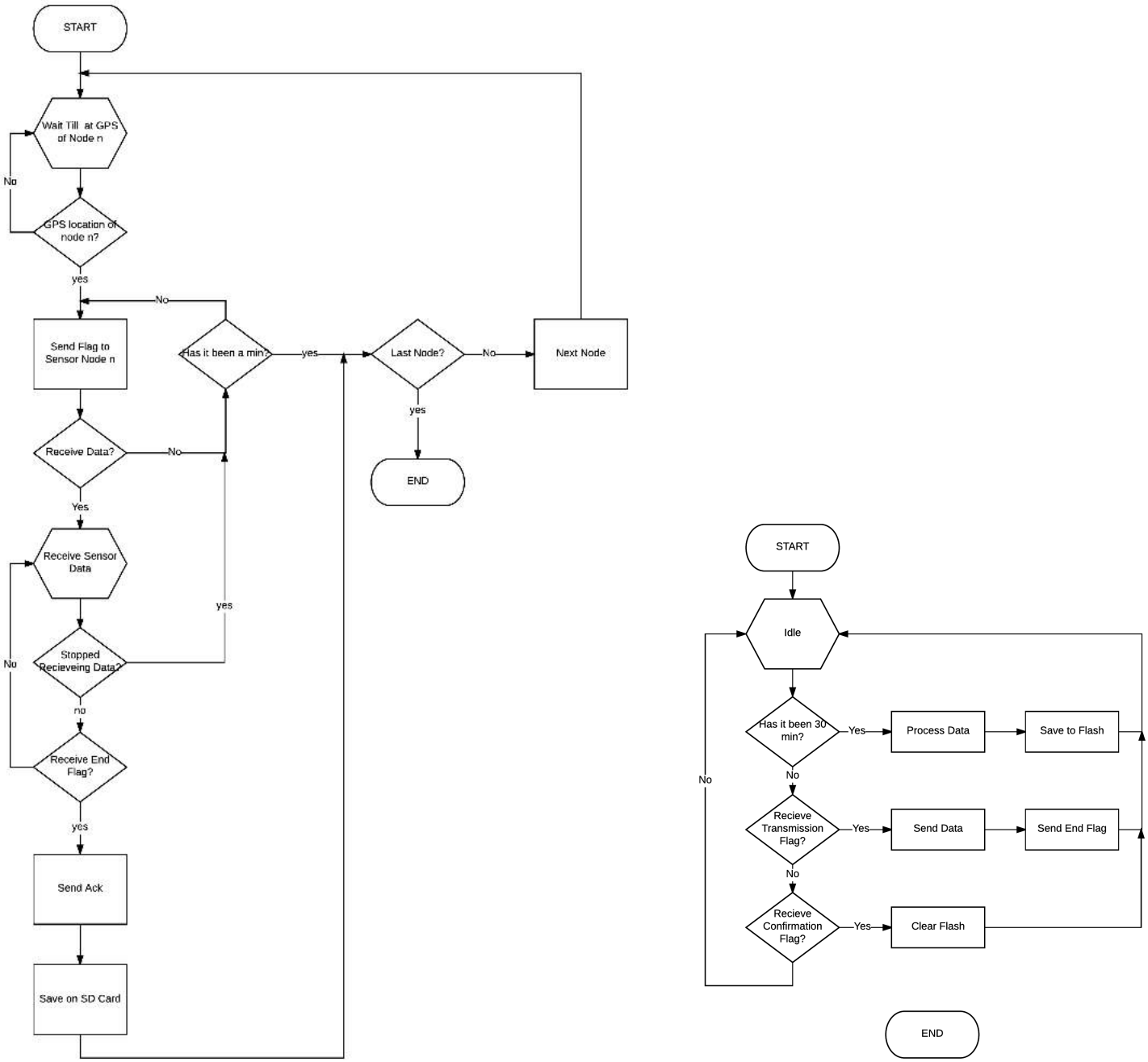


Figure 2: Programming Flowcharts: Left: Mobile Node. Right: Sensor Node.

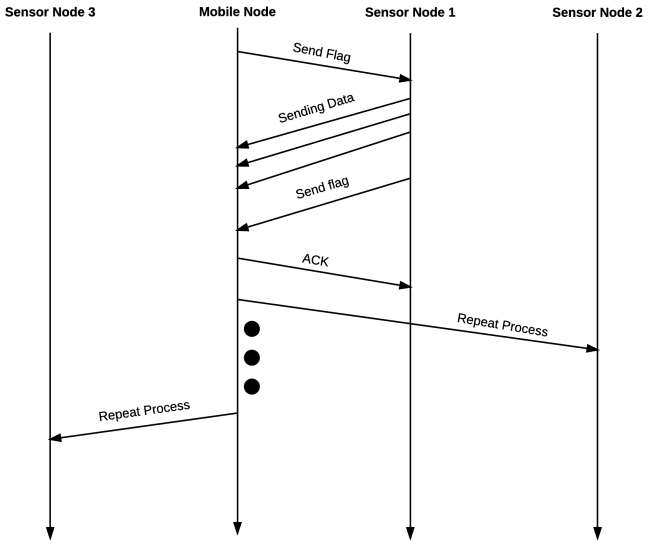
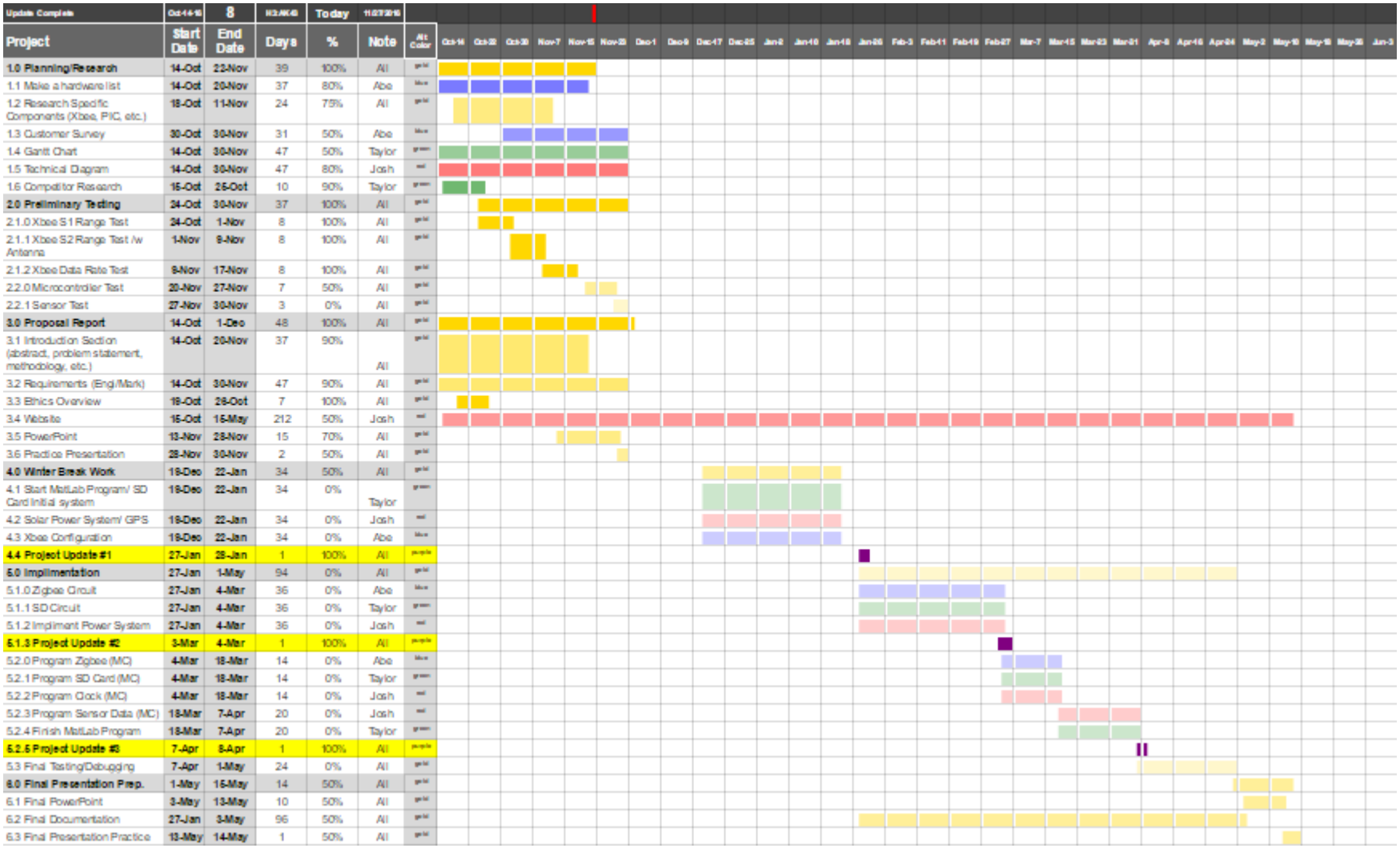


Figure 3: Wireless Protocol Timing Diagram between each Sensor Node and the Mobile Node

This Timing Diagram takes place as soon as the Mobile Node is within the defined GPS coordinates of the Sensor Node. The mobile node sends a transmission flag to the Sensor Node, then the sensor node will send its data with an acknowledgement. The mobile node will then send another acknowledgement and travel to the next Sensor Node and repeat the process.

2.1.1 Schedule

Due to the time limit constraints of this project we need to plan out our schedule carefully. Below a gantt chart summarizes our planned schedule. Each person did a portion of the project individually, with assistance from their team members. Taylor Jones worked mostly on designing, building, and programming the Mobile Node. Joshua Papanicolas and Abe Palmerin worked together on different parts of the Sensor Node with Joshua mostly working on the power system and Abe mostly working on implementing the sensors and Xbee interface.



Project	Start Date	End Date	Days	%	Note
1.0 Planning/Research	14-Oct	22-Nov	39	100%	All
1.1 Make a hardware list	14-Oct	20-Nov	37	80%	Abe
1.2 Research Specific Components (Xbee, PIC, etc.)	18-Oct	11-Nov	24	75%	All
1.3 Customer Survey	30-Oct	30-Nov	31	50%	Abe
1.4 Gantt Chart	14-Oct	30-Nov	47	50%	Taylor
1.5 Technical Diagram	14-Oct	30-Nov	47	80%	Josh
1.6 Competitor Research	15-Oct	25-Oct	10	90%	Taylor
2.0 Preliminary Testing	24-Oct	30-Nov	37	100%	All
2.1.0 Xbee S1 Range Test	24-Oct	1-Nov	8	100%	All
2.1.1 Xbee S2 Range Test /w Antenna	1-Nov	9-Nov	8	100%	All
2.1.2 Xbee Data Rate Test	9-Nov	17-Nov	8	100%	All
2.2.0 Microcontroller Test	20-Nov	27-Nov	7	50%	All
2.2.1 Sensor Test	27-Nov	30-Nov	3	0%	All
3.0 Proposal Report	14-Oct	1-Dec	48	100%	All
3.1 Introduction Section (abstract, problem statement, methodology, etc.)	14-Oct	20-Nov	37	90%	All
3.2 Requirements (Engi/Mark)	14-Oct	30-Nov	47	90%	All
3.3 Ethics Overview	18-Oct	25-Oct	7	100%	All
3.4 Website	15-Oct	15-May	212	50%	Josh
3.5 PowerPoint	13-Nov	28-Nov	15	70%	All
3.6 Practice Presentation	28-Nov	30-Nov	2	50%	All
4.0 Winter Break Work	18-Dec	22-Jan	34	50%	All
4.1 Start MatLab Program/ SD Card Initial system	18-Dec	22-Jan	34	0%	Taylor
4.2 Solar Power System/ GPS	18-Dec	22-Jan	34	0%	Josh
4.3 Xbee Configuration	18-Dec	22-Jan	34	0%	Abe
4.4 Project Update #1	27-Jan	28-Jan	1	100%	All
5.0 Implimentation	27-Jan	1-May	94	0%	All
5.1.0 Zigbee Circuit	27-Jan	4-Mar	36	0%	Abe
5.1.1 SD Circuit	27-Jan	4-Mar	36	0%	Taylor
5.1.2 Implimant Power System	27-Jan	4-Mar	36	0%	Josh
5.1.3 Project Update #2	3-Mar	4-Mar	1	100%	All
5.2.0 Program Zigbee (MC)	4-Mar	18-Mar	14	0%	Abe
5.2.1 Program SD Card (MC)	4-Mar	18-Mar	14	0%	Taylor
5.2.2 Program Clock (MC)	4-Mar	18-Mar	14	0%	Josh
5.2.3 Program Sensor Data (MC)	18-Mar	7-Apr	20	0%	Josh
5.2.4 Finish MatLab Program	18-Mar	7-Apr	20	0%	Taylor
5.2.5 Project Update #3	7-Apr	8-Apr	1	100%	All
5.3 Final Testing/Debugging	7-Apr	1-May	24	0%	All
6.0 Final Presentation Prep.	1-May	15-May	14	50%	All
6.1 Final PowerPoint	3-May	13-May	10	50%	All
6.2 Final Documentation	27-Jan	3-May	96	50%	All
6.3 Final Presentation Practice	13-May	14-May	1	50%	All

Figure 4: Project Schedule

2.1.2 Key Components

1. 3.7 volt batteries (4)
2. Pic32 Microcontrollers (PIC32MX695F512L) (1)
3. Pic18 Microcontrollers (PIC18F45K22) (3)
4. Printed Circuit Boards (10)
5. Xbee Radio Module (S3 Pro) (4)
6. Xbee breakout boards (4)
7. Sensors (9) (3 temperature, 3 humidity, 3 soil moisture)
8. electrical components (wire, capacitors, resistors)
9. 32 GB micro SD card(4)
10. Device casings(4)
11. GPS Module (1)
12. Real Time Clock Module (4)
13. Solar Panels (3)
14. 900 Mhz Duck Antenna (4)

2.2 Challenges

2.2.1 Resolved Problems

1. **Data duplication:** This problem occurs when the data on the SD card is not erased after the data has already been sent. However, since we have programed the sensor node to erase the data after it has been sent, the user will not have this issue.

2. **Lost data:** Data from the sensor node may be lost if bytes in the buffer overflows. However, by adding a small delay while the buffer is being processed, this will not be a problem for the user.

3. **Sensor node unresponsive:** If the sensor node is unresponsive, it can be one of several problems. One possible case is that the sensor node is not being powered by the power circuit, in which the battery has probably died. If this is true, then the user will just need to wait until the battery has been recharged by the solar panel. Another possible issue is either the Xbee on the mobile node is not finding the Xbee on the sensor node or vice versa. Therefore, the Xbees may need to be reset with the proper ID or replaced if they have broken.

4. **Solar Panel Circuit not providing enough power/ Battery dies:** In this case, the battery has died because the solar panel is not receiving enough sun, and therefore is unable to charge the battery. The user will need to wait until the solar panel has recharged the battery.

5. **Data reliability:** If the the sensors are not calibrated properly before using out in the field, the data will not be reliable. Therefore, the sensors should be calibrated.

6. **Single sensor breaks on node:** If data from a sensor is giving back all zeros or NAN, then it is likely the sensor is broken and will need to be replaced.

2.2.2 Common Problems

1. Possible interference from other x-bees in the field.

2. GPS returns all zeros for longitude and latitude for the first few minutes of it being powered

3. Real Time Clock becomes inaccurate when backup battery runs out

4. Some data corruption occurs when device is first powered up and immediately tries to do a transmission

2.3 Testing

An important part of product development is initial testing. In order to properly design the initial prototype we must know what kind of limitations we might encounter and what exactly we need to use in our product ahead of time before we settle on exact specifications.

2.3.1 ER2: Range testing w/ improved modules and antennas

In this testing state we attempted to test the maximum range of the Xbee. We used an Xbee Series 2 at 2.5 Ghz with larger antennas. We found that we were able to transmit packages from over 120 meters (393 ft) away without errors. That is over 90 feet further than our required 300 ft. We then conclude that we will be using Series 2 or better Xbees with these antennas, or something similar, in our product.

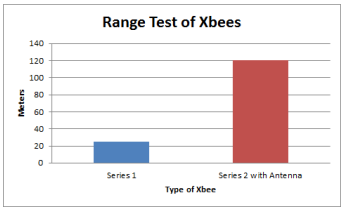


Figure 5: Range Test of Series 1 Xbee vs Series 2 Xbee with antenna

2.3.2 ER11: Packet Size Test

In this test, we wanted to see how long it took for a Xbee to transfer large amounts of data. We found that in API mode it sent max 255 byte packets maximum taking about 2 minutes to transfer about 30 kilobytes of data. Which is a rate of 2.09 Kbps (kilobits per second). We also tested AT mode where it was able to transfer 15 kilobytes in 18 seconds. We plan to use at least 3 different sensor modules.

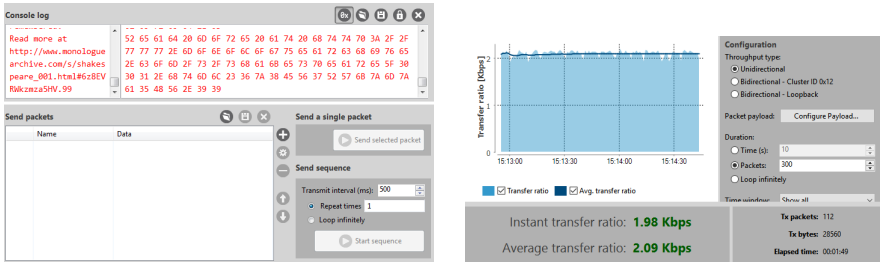


Figure 6: Left: AT Test Right: API Test

2.3.3 MR1: Transmission Time Minimum

For this test we calculated the possible amount of time the mobile node would have to receive the data from a sensor node and the amount of data that would be transferred. We found that based on the Xbee range, the device could have between 12 and 30 seconds to receive the data at an altitude of 50m and would receive between 3.9 - 9.4 kilobytes of data respectively.

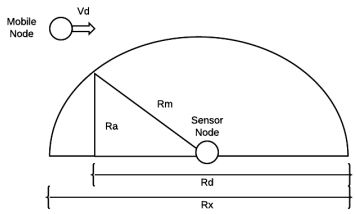


Figure 7: Model of a Moving Mobile Node Communicating with Sensor Node

- Vd = average speed of the UAV (m/sec)
- Rd = range radius (m)
- Rx = max range of Xbee
- Rm = distance between mobile and sensor nodes (m)
- Ra = flight altitude = 50 m
- Tz = transmission range of the ZigBee = 20 Kbps - 250 Kbps

$$Rd = 2 * (Rm^2 - Ra^2)$$

$$ContactTime = \frac{Rd}{Vd}$$

$$DataTransmitted = \frac{TransmissionRate * ContactTime}{8}$$

10	<--Drone Speed	25000	<--Zigbee Transmission rate				
Range Radius (m)		Flight Altitute			Contact Range	Contact Time	Data (byte)
80	6400	2500	3900	62.44997998	124.89996	12.489996	39031.23749
90	8100	2500	5600	74.83314774	149.6662955	14.96662955	46770.71733
100	10000	2500	7500	86.60254038	173.2050808	17.32050808	54126.58774
110	12100	2500	9600	97.97958971	195.9591794	19.59591794	61237.24357
120	14400	2500	11900	109.0871211	218.1742423	21.81742423	68179.45072
130	16900	2500	14400	120	240	24	75000
140	19600	2500	17100	130.7669683	261.5339366	26.15339366	81729.35519
150	22500	2500	20000	141.4213562	282.8427125	28.28427125	88388.34765
160	25600	2500	23100	151.9868415	303.9736831	30.39736831	94991.77596

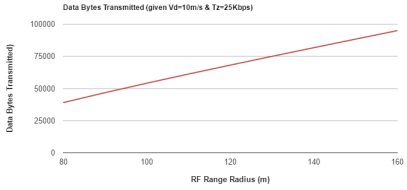


Figure 8: Top: Table of Simulated Values, Bottom: Plot of Simulation

2.3.4 MR1: Power Tests

In these tests we measured the power consumption of both the mobile and sensor node's power consumption so that we could be sure that we could provide enough power for a long enough time for our devices to be effective. The sensor node must be powered indefinitely via a solar charged battery, while the mobile node is powered by the battery of the UAV. The sensor node must not drain the battery faster than it can be charged, while the mobile node must not use so much power that it prevents the UAV from completing its flight.

To measure the current and power of the sensor node device, we connected it to a 3.3 volt power supply. Once the sensor node was powered on and working, we read the current of the device from the power supply unit for when the device was transmitting and when it was not. We found the current when the sensor node is transmitting to be 0.04 Amps. When it is not transmitting, that is, idle, it produces a current of 0.03 Amps. Using the power equation, P=VI, the power of the device while transmitting is 0.1328 Watts and 0.0996 Watts while not transmitting.

	Idle	Transmission
Mobile Node	0.16 Amps, 0.8 Watts	0.2 Amps, 1 Watt
Sensor Node	0.03 Amps, 0.0996 Watts	0.04 Amps, 0.1328 Watts

Table 3: Power Usage of Each using 5V Supply (in Amps and Watts)

2.3.5 ER11/MR11: Packet Size Calculation

In order to ensure that the micro-controller's built in flash can hold the data that the sensor node will be saving onto it and to help us calculate the transmission time of the data packets we need to know the exact size that these packets will be. There are two ways in which we can store each data set that is taken, one is to store them as a single string and the other is to store each value individually as floating point numbers. For each packet, we are storing and transmitting 4 different numbers.

For strings we will assume that we will be accurate to up to 6 significant figures for the sensor data and the time is stored as a 10 digit number (UNIX time format). Strings store each digit as a single character. Thus, we would have 28 characters for the data, plus 3 more for each comma separating them, and 1 more that is added invisibly by the string data structure in C++. Thus, each data set will be 32 bytes (1 byte per character). This number will go up by 3 for each significant figure that we would add if desired.

For floating point, no matter how big the number is (as long as it is smaller than 2,147,483,647) each number will take up 4 bytes. Thus, using the same data set of 4 numbers (this time with no commas, we would store each number separately) each data set would also use 32 bytes. Except in this case each value could be accurate up to 10 significant figures (again as long as it doesn't go over the size limit).

In conclusion, both of these methods of storing the sensor data are viable with each having benefits and drawbacks. Using strings would be easier to organize and transmit and would leave less work for the time sensitive mobile node since it wouldn't have to organize the data itself. But if we wanted to have extra accuracy in our data, it would take up exponentially more space on the flash. For floating point, we can be more accurate while using the same space as a 6 significant figure accurate string data set, but it would be more complicated to store, organize, and transmit.

We do not plan on being accurate over 6 significant figures and the flash memory of the micro-controller can hold 1000 (or more if we use the programming memory) 32 byte data sets on a a PIC32MX340F512, or about 500 (if we use part of the programming memory) 32 byte data sets on a PIC18F45K22. Which would equate to having to recover the data every 20 days for the PIC32 and every 10 days for the PIC18. Though both of these values can go up depending on how much programming memory is left after we burn the chip.

Thus, we will be using strings to store our data, since it will be easier to organize and transmit and we do not need the extra accuracy of full floating point numbers.

Date/Time Number (in UNIX time)	Temperature	Soil Moisture	Humidity	Commas	Overhead	Total
1486077022	64.3257	322.432	55.5437	3 Commas	String Structure in C++	N/A
10 bytes	6 bytes	6 bytes	6 bytes	3 bytes	1 byte	32 bytes

Table 4: Break Down of Size of 1 Data Packet

2.3.6 ER10/MR4: Range Test 3

In this test we used the program XCTU to measure the RSSI (receive signal strength indicator) between two nodes over distance and over time using Xbee pro. In our test we were able to go over 500 feet without losing the signal. We could have gone even further, but ran out of unobstructed space.

In another test, we measured the loss over range instead of time over 1000ft. As seen from figure 11, there noticeable loss the further we got from the sensor node. At the beginning and the end, the signal doesn't change which is just a discrepancy caused by the testing software we used (XCTU) where there wasn't a noticeable change in loss for the first few feet and at the end, the signal timed out which appears as a flat line on the graph.

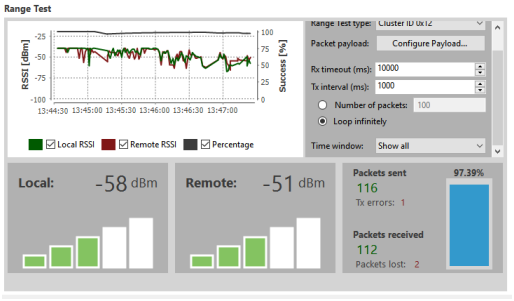


Figure 9: Range Test with Loss Over Time for 500ft with Xbee Pro S3B

Distance (ft)	Distance (m	Local (dBm)	Remote (dBm)
0	0	-40	-40
20	6.1	-40	-40
40	12.2	-40	-40
60	18.3	-40	-42
80	24.4	-45	-47
100	30.5	-47	-48
120	36.6	-49	-51
140	42.7	-48	-52
160	48.8	-53	-55
180	54.9	-62	-67
200	61	-61	-62
220	67.1	-65	-65
240	73.2	-66	-64
260	79.3	-62	-61
280	85.4	-65	-68
300	91.5	-64	-75
320	97.6	-67	-66
340	103.7	-69	-77
360	109.8	-73	-76
380	115.9	-65	-66
400	122	-71	-79
420	128.1	-79	-82
440	134.2	-76	-80
460	140.3	-76	-80
480	146.4	-76	-84
500	152.5	-75	-76
520	158.6	-75	-76
540	164.7	-76	-76
560	170.8	-78	-77
580	176.9	-78	-77
600	183	-78	-77
620	189.1	-78	-77
640	195.2	-78	-77
660	201.3	-78	-77
680	207.4	-78	-77
700	213.5	-78	-77
1000	305	-78	-78

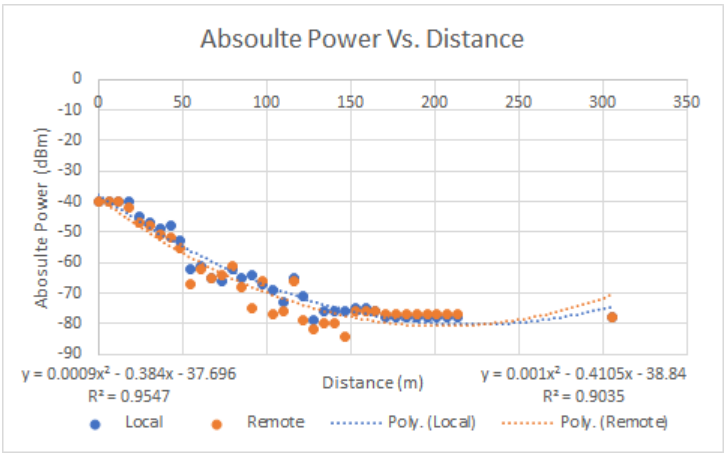


Figure 10: Range Test with Loss Over Distance for 1000ft with Xbee Pro S3B

2.3.7 ER3/MR6: SD Card Test

The original plan for this portion of the device was going to involve a PIC18F45K22 microcontroller and an external SD mount. Programming the system to save files on an SD card can be challenging and is beyond the scope of our knowledge, so we must use a library that will do all of this for us. First we set up the device with the previously mentioned PIC and tried using a library provided by Microchip, but found that it only officially supported certain controllers. We found a third party library (FatFs by elm-chan) that was universal for any controller, but was unable to get even that to function. We tried again with an external oscillator as the clock (once with a crystal and again with signal generator) thinking that the clock wasn't a nearly perfect square wave which is required for SD cards to function, but was still unable to save anything on the card.

There is obviously a way to make this work, but due to time constraints we had to switch to the WF32 board by Digilent (see appendix). This board has a built on SD card mount and a controller with a bootloader that has a predefined SD card library explicitly made for its controller. We were able to easily get it to function and saved a text file with "hello world" written in it. After that we were able to save a fake data (ie defined in the code, not actually measured) as a csv (comma-separated value) file which can easily be converted into an excel file for the user (see MATLAB test (2.2.15)).

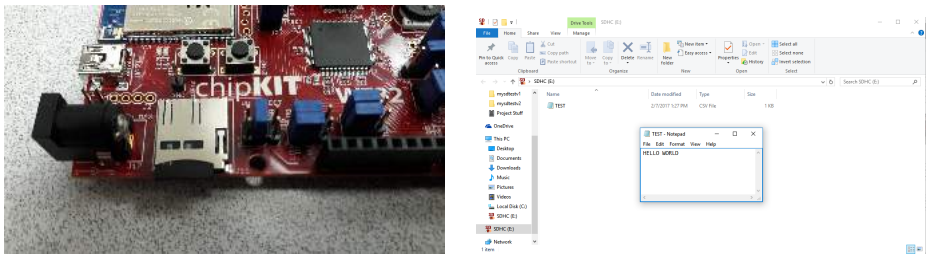


Figure 11: Left: SD Card on a WF32 Board. Right: CSV File Generated.

2.3.8 ER4: Xbee General Test

Our devices (the mobile and sensor nodes) will be communicating via the ZigBee protocol via a transmitter called an Xbee. Our initial testing with these device can be seen in the test 2.2.1, 2.2.2, and 2.2.3, but these test simply tested range and transmission rate. We need to be able to send actual data via the circuitry we will be using for the actual devices and not with the testing boards and laptops we were using for these early tests. We set up preliminary circuits to allow us to do these tests (see appendix: Circuits) and coded the controllers on them to use the Xbees to transmit a packet of fake data from one sensor node to the mobile node. In the test, we simply had the sensor node broadcast the packet continuously every 2 seconds and had the mobile node receive the

packet and save it to the SD card continuously. We were able to successfully do this test repeatedly (for a more vigorous example of this procedure see test 2.2.16).

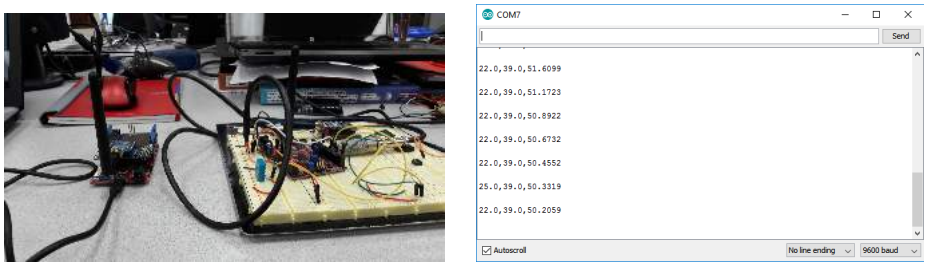


Figure 12: Left: Mobile Node (left) and Sensor Node (right) Communicating. Right: Serial Monitor displaying packets received by Mobile Node.

2.3.9 ER6/ER7: Dimensional Measurements

The Mobile Node will be placed on a UAV to be carried to each Sensor Node due to them being placed far out in the wilderness. UAVs can be very sensitive about the dimensions and weight of their payloads and thus we must ensure that the mobile node isn't too big or too heavy to be carried by a UAV.

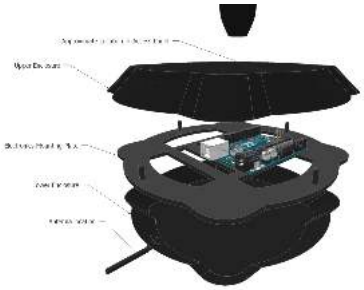


Figure 13: CAD drawing of Preliminary Shell Design (by Sean Headrick)

Length	Width	Height	Weight
8.9 cm	5.3 cm	2.9 cm	90.17g

Table 5: Mobile Node Dimensions

We simply measured the dimensions with a ruler due to not needing exact precision and measured the weight with a digital scale (provided by the SSU Chemistry Department).

The device will be placed in a custom shell provided and designed by Sean Headrick of Aerotestra.

2.3.10 ER8/MR8: Solar Charger Test

To power our sensor nodes at each station, we will use a solar panel and lithium ion batteries. For our initial tests, we used a 2.5W 5V/500mAh solar panel and two 3.7/1200mAH lithium ion cell batteries in parallel. In between the solar panel and batteries is a 5V charger board, used to charge the batteries from the solar panel. From the charger board, we connect it to a 3.3V step-down converter, which powers the sensor node.

When the solar panel is in direct sunlight it produces over 6V. In our first test we got 6.08V. It also has a current of 0.430 Amps in sunlight. Knowing the voltage and the power, we find that the solar panel produces a little over 2.5 Watts, exactly what we should expect. Having our power circuit connected correctly, we were able to power the sensor node device with no issues while having the batteries charging at the same time. This test indicates that as long as there is sunlight, the sensor node will always receive enough power to transmit and remain on.

To make sure the sensor node can be powered all through the night when there is no sun, we had to check the batteries to make sure they would last at least 12 hours with no power coming from the solar panel. The first batteries we tested were two 3.7V UltraFire lithium ion cell batteries in parallel rated at 1200mAH each. However, after leaving the device running for a few hours after the batteries were fully charged, the batteries stopped producing enough current to power the device after only a few hours. Next, we tested a lithium ion polymer battery also rated 1200mAH. This time, the battery lasted longer than 8 hours while still providing enough current to the sensor node. While this did not last the 12 hours we were hoping for, we did not have the funds to purchase batteries rated at a higher level. If we had the budget, we would have gone with a battery rated at 2000 mAH LiPo battery for each node, so we could be sure it would last 12 hours.

We tested the voltage of the 1200mAH battery and saw how voltage changed over time in the sun. We tested how the voltage increased with the solar panel charging it, and also how it decreased when the solar panel was not charging it. Below are graphs that show the voltage of the battery over time, where we took measurements every ten seconds.

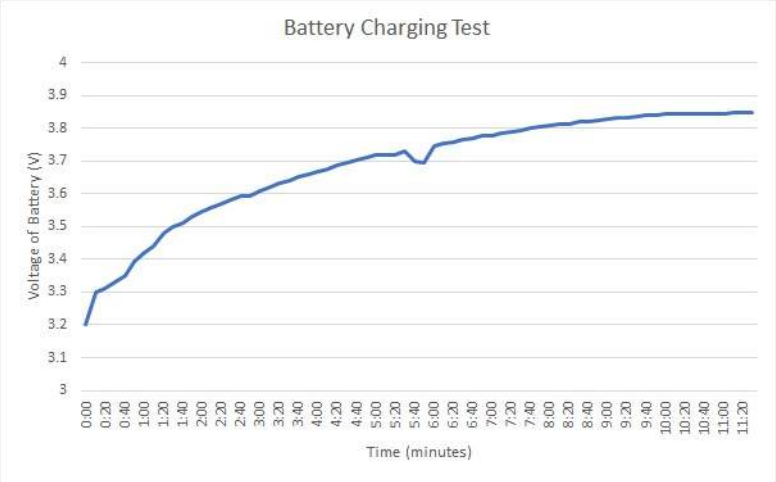


Figure 14: Battery Charge Test Over Time

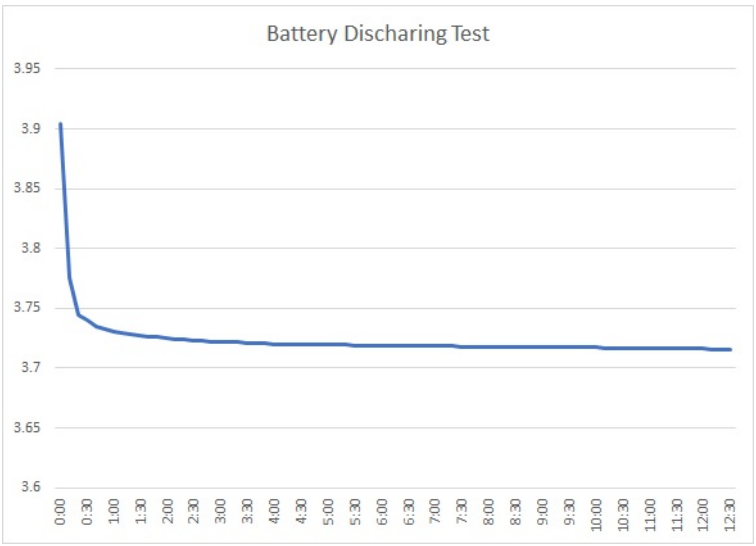


Figure 15: Battery Discharge Test Over Time

2.3.11 ER9: GPS Test

The GPS module we chose for our project is the GP-20U7. The first test we conducted with the GPS module was to test if we could get NMEA data from it and display it on a serial monitor. We found that while we were inside, the NMEA data would be incomplete as it could not acquire the satellites. However, when we took it outside, we were able to see the full NMEA data we were expecting. Using several GPS libraries in our code, we were able to get latitude and longitude coordinates.

For the second test, we needed to be able to ensure that the GPS coordinates were accurate. We tested this by comparing the value read from the module with the coordinates from a hand held GPS device. (results here)

For the third test, we needed to measure out the radius required for the mobile node to be within with sensor node for transmission. From previous tests (see test 7), we know that the Xbees can transmit out to over 1000ft, thus we have a lot of room to work with when it comes to the radius around the sensor node we need to make with the GPS. We wanted to test out a 300ft radius (see figure 14), but found it difficult to find a large enough open space for this test. So instead we did a 100ft radius and scaled the measurements to 300ft.

Distance	Latitude (GPS Module)	Longitude (GPS Module)	Latitude (Control)	Longitude (Control)
Origin	38.339182	-122.669337	38.339200	-122.669212
50ft (N)	38.339324	-122.669335	38.339412	-122.668528
100ft (N)	38.339459	-122.669325	38.339564	-122.669252
Difference (50ft)	-0.000142	-2E-06	-0.000212	-0.000684
Difference (100ft)	-0.000277	-1.2E-05	-0.000364	-0.00096
50ft (S)	38.339006	-122.669325	38.339120	-122.069401
100ft (S)	38.339952	-122.669308	38.339021	-122.069530
Difference (50ft)	0.000116	-1.2E-05	8E-05	0.000189
Difference (100ft)	0.00023	-2.9E-05	0.000179	0.000318
50ft (E)	38.339210	-122.669506	38.339187	-122.069565
100ft (E)	38.339207	-122.669653	38.339242	-122.069633
Difference (50ft)	-2.8E-05	0.000169	1.3E-05	0.000353
Difference (100ft)	-2.5E-05	0.000316	-4.2E-05	0.000421
50ft (W)	38.339234	-122.669150	38.339234	-122.069138
100ft (W)	38.339218	-122.669985	38.339200	-122.069011
Difference (50ft)	-6.2E-05	-0.000187	-3.4E-05	-7.4E-05
Difference (100ft)	-3.6E-05	-0.000352	0	-0.000201

Table 6: GPS Value Table

For the final test, we used a preliminary 50ft radius and tested the mobile node's ability to determine its location. When the mobile node entered into the 50ft radius around the sensor node, it began transmitting the flag to tell the

sensor node to begin transmitting its data.

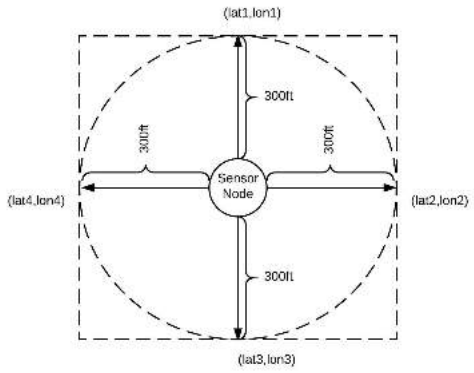


Figure 16: Plot of proximity Mobile Node must be within Sensor Node to communicate

2.3.12 ER9/MR5: Downed Node Test

In this test, we needed to be sure that the mobile node could handle a situation where a sensor node didn't respond for whatever reason. What it does is once it is within proximity of the sensor node, after a period of 1 minute, if no data is received, the mobile node will save a message onto the SD card indicating something went wrong. When the user tries to save the data onto their computer via the program we provided, said program will display a message saying which node is being unresponsive.

2.3.13 ER5/MR3: Sensor Tests

All three sensor nodes will be using the same exact sensors and same number of sensors. In this project, each sensor node will have a temperature sensor, humidity sensor, and soil moisture sensor. This totals nine different sensor readings for 3 different kinds of sensors.

Our first sensor is a DS18B20 waterproof digital temperature sensor. It requires a 1-wire interface plus Vcc and ground. It has a -55 to 125 degree Celsius temperature range. The probe is 7mm in diameter and 26mm long. The complete length of the temperature sensor is 6 feet. We calibrated this sensor by conducted a simple ice bath where we stirred its probe in a cup of the correct ice to water ratio until 0 degrees Celsius was reached. We were able to run a simple program where we were able to see the current degree values on our serial monitor. According to the DS18B20 data sheet, this temperature sensor has a 0.5 degree accuracy within the -10 to 85 degree Celsius range.

Figure 17: DS18B20 Weatherproof Temperature Sensor

Figure 18: Temperature Probe in Ice Bath

```
ROM = 28 8E C0 A9 8 0 0 DA
  Chip = DS18B20
  Data = 1 6 0 4B 46 7F FF A 10 F1  CRC=F1
  Temperature = 0.37 Celsius, 32.67 Fahrenheit
No more addresses.
```

Figure 19: Temperature Sensor Serial Monitor Results

Our second sensor is a SHT10 digital soil moisture sensor by Sensirion. It is a 4-wire Sensirion chip inside a sinter metal mesh encasing to be placed in soil. The casing is weatherproof and meant to be placed in soil over long periods of

time. According to the SHT10 data sheet, it has a 4.5 percent accuracy relative humidity reading, 14mm diameter and 50mm long sensor. It is interfaced using I2C and the cable itself is a meter long. We tested our soil moisture sensors by comparing them with a high-quality Hydrosense II Soil Moisture Measurement System by Campbell-Scientific. The Hydrosense II is a portable, handheld device for easily obtaining soil measurements. It contains a 12cm probe and a large LCD display. It has a 3 percent volumetric water content accuracy and outputs percentages from 0-50 percent. We conducted this test at the Pepperwood Preserve in Santa Rosa.

Figure 20: Left: Our SHT10 Soil Sensor by Sensirion Right: The AM2315 Humidity Sensor by AOSONG

Figure 21: Left: Soil Moisture Test Location Right: Patch of soil

Our third Sensor is a AM2315 Digital Humidity Sensor by AOSONG. Similar to the Soil Moisture Sensor, it requires an I2C communication and only uses 4 wires. The body size is 98mm by 16mm and the cable itself is 20 inches long. It has a 0-100 percent humidity reading according to the AM2315 datasheet. We compared our humidity results using the Dwight Weather Station at the Pepperwood Preserve. Values were adjusted accordingly.

Figure 22: Left: Hydrosense II screenshot Right: Hydrosense probe in soil

Figure 23: Humidity at Pepperwood Preserve Dwight Center Weather Station

2.3.14 ER5/ER6/MR9: Solar-Powered Ground-Based Sensor Node

After placing all of our sensors onto one board, we placed the complete sensor circuit, which is composed of the Uc32 board, Xbee Shield, Xbee, our 3 sensors, a real-time clock module, and the micro SD card module, into the Sensor Node case.

Here we show our complete sensor and power circuit in our weatherproof casing. Our case is made of PVC with 4 holes fitted for conduit tubes. Holes that are not used are covered with stoppers. This casing is originally made to provide power outlets from a concrete floor. We placed our case onto a wooden board then hung onto an outdoors U-post.We also placed the solar panel right above the case.

Figure 24: A look inside a sensor node case

Figure 25: An outdoor photo of a complete sensor node with solar panel

2.3.15 MR7: MATLAB Test

For this test, we are testing the code and capabilities of the Matlab program we created for our project. The program is used to pull the data off of the SD card from the mobile node and converts the CSV file to a more palatable excel file with titles on the data sets and save it all on the user's computer. It also removes the old data from the SD card for the user to prevent repeated data. Lastly, the program allows the user to program the GPS coordinates of the sensor nodes for the mobile node's GPS proximity functionality.

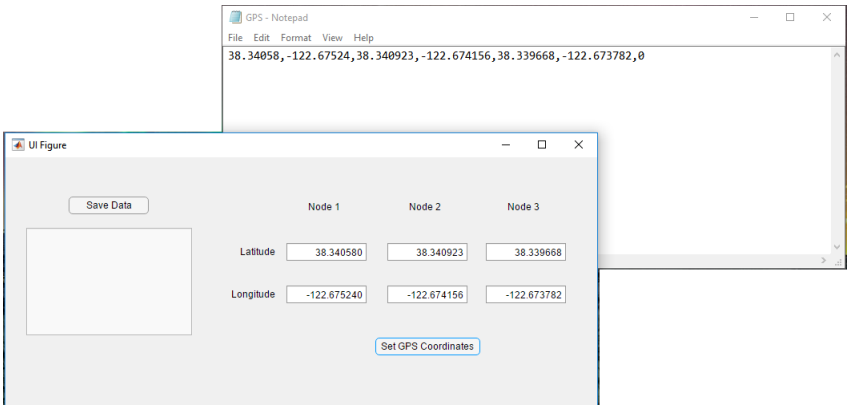


Figure 26: Program Successfully Saving GPS Coordinates

The program was able to save the GPS coordinates into a CSV file that the mobile node can read off of its SD card to know what coordinates each sensor node is at.

Along with this, we were able to successfully pull data saved by the mobile node onto the same SD card and save it onto the user's computer as an excel file in a more readable format and delete the old data.

2.3.16 MR12: Long Term Test

It is important to know how long our system can run before receiving errors or missing data. In our preliminary long term test we left our sensor and mobile nodes on all night by plugging them in the computers in the design lab. Our sensor node was transmitting whatever was in its serial port every half hour. We started the test around 7:30 pm on Tuesday February 7. The next day we checked our data around 10:00 am Wednesday February 8 and were pleased that no data was missing or corrupted. Below is a screen shot of the data from the SD card. This preliminary test lasted for about 15 hours.

Starting up					
22	38	55.6264	0	0	Node is Active
22	38	55.7825	0	0	Node is Active
22	38	56.6615	0	0	Node is Active
22	38	56.5436	0	0	Node is Active
22	38	55.4649	0	0	Node is Active
22	38	56.2614	0	0	Node is Active
22	38	57.5325	0	0	Node is Active
21	38	57.5614	0	0	Node is Active
21	38	57.3254	0	0	Node is Active
22	38	56.9831	0	0	Node is Active
22	38	56.7124	0	0	Node is Active
22	38	56.4777	0	0	Node is Active
22	38	56.2353	0	0	Node is Active
22	38	56.4412	0	0	Node is Active
22	38	56.1168	0	0	Node is Active
22	38	56.2031	0	0	Node is Active
22	38	57.2466	0	0	Node is Active
22	38	59.7549	0	0	Node is Active
22	38	60.6128	0	0	Node is Active
22	39	61.3021	0	0	Node is Active
21	39	61.8522	0	0	Node is Active
22	39	62.3979	0	0	Node is Active

Figure 27: Screen shot of 15 hour long term data in thirty minute intervals

After seeing how the data was collected inside, we wanted to make sure the data would be collected just as well outside. When we got a weatherproof case for each device, we took all three sensor nodes we made and put them outside to test. Like stated previously in the Solar Charger Test, the batteries did not last the whole night. Since this was the case, we also had to ensure that the sensor node device would start running again as soon as the batteries charged up, which they did. The data was correctly stored onto the micro SD card when left outside. Here are the results of one of the tests we ran:

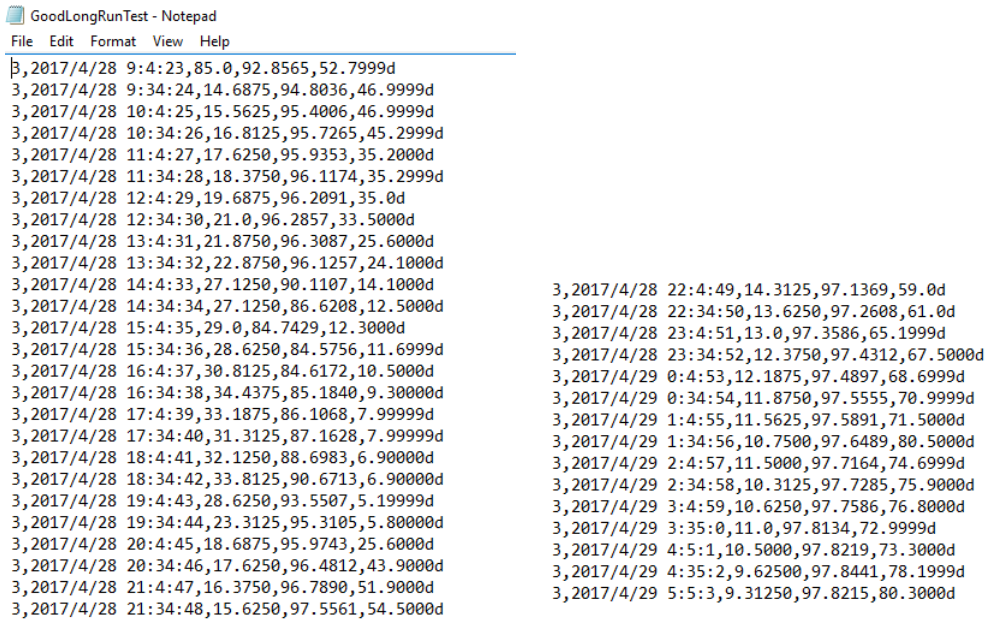


Figure 28: Sensor node device ran from around 9:00 am to around 5:00 am the next day

2.3.17 MR4/ER1/ER4: UAV Test

One of the main goals of our project is to have a mobile node attached to a UAV fly over the sensor nodes and collect data wirelessly from the air. Thus, we had to test to make sure this was possible. For this test, we attached our mobile node to an Aerotestra UAV. Once the sensor node was placed on the ground, the UAV was programmed to fly over the sensor node, hover there, and come back. We did this test three times, with each time checking the SD card for the data. The first time we did it, we had the UAV hover over the sensor node for five minutes at 90 feet. The second time we had it hover for three minutes at 50 feet and the last one for one minute at 50 feet. We found that the mobile device was able to collect all the data within a minute. We were able to collect the data from the sensor node and efficiently and as planned. However, there was a several data strings that were corrupted from the data we collected. We later resolved this issue by changing how the data is transmitted. Instead of transmitting a single packet of data, the sensor node transmits one data string at a time.

Figure 29: Picture of Abraham Palmerin with Sean Headrick piloting the UAV

2.3.18 MR3/ER4/ER5/ER9: Moving Mobile Node Test

For this test, we simulated what our project is hoping to accomplish. We set up three sensor nodes around Sonoma State University, each with a specified GPS location. The mobile node was moved around by car, and was driven by each of the nodes on the ground. When the mobile node was done collecting the data, a LED would light up to let the driver know to move on to the next sensor node and see how long it takes to collect data. The test was successful, with the environmental data being collected as soon as the mobile node got in range. However, one of the sensor node's battery was not charged fully and thus did not transmit any data. After powering the device to a computer, we got the data. This test showed that our project is fully working.

In this test we used a 50 ft. radius around the sensor nodes, so they would transmit as soon as the mobile node got close enough. We also calculated what a 200 ft. radius would look like, which is closer to what we will be using in the final product.

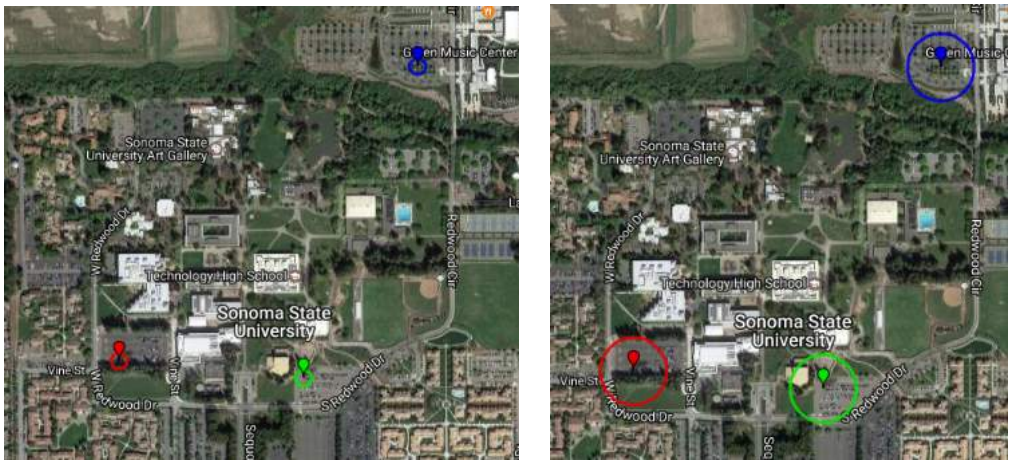
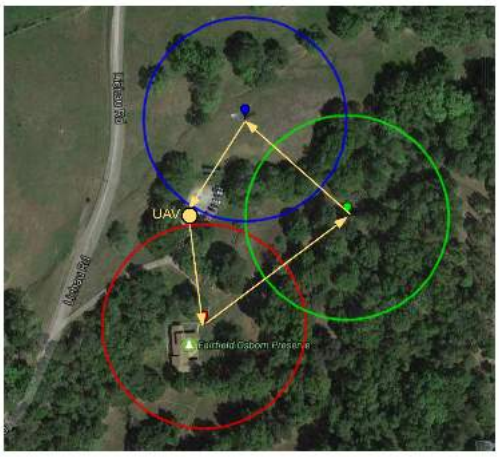


Figure 30: Left: 50 ft Radius. Right: 200 ft Radius.

2.3.19 Final Test

In our final test, we mounted the mobile node to a UAV and set it to autonomously fly to the three nodes. We set the mobile node to begin transmission at 200 ft. Each sensor node was placed at various locations at the Fairfield Osborn Preserve in Rohnert Park, Ca.

The UAV was able to successfully fly to each, it hovered for 10 seconds at each node to ensure a complete transmission. We ran into few problems involving power to the third sensor node and receiving power from the UAV to the mobile node. After a few hours of troubling shooting the problems, we were able to do a successful test, gathering real sensor data from each node. The figure below shows the flight path of the UAV along with the 200 ft GPS radius that the mobile node did the transmission in along with the data from the test after being processed by our Matlab program.



	A	B	C	D	E
1	Node_Number	Date_Time	Temperature	Soil_Moisture	Humidity
2	1	4/21/2017 17:35	29.125	66.6151	35
3	1	4/21/2017 18:05	21.75	45.7066	36
4	1	4/21/2017 18:35	22.75	39.7379	36.2999
5	1	4/21/2017 19:05	26.3125	38.4683	35.7999
6	1	4/21/2017 19:35	26.5625	37.8203	35
7	1	4/21/2017 20:05	25.5	37.2149	34
8	GPS Coord	38.343465	-122.594609	0	0
9	2	4/21/2017 20:35	23.125	37.7486	33.5999
10	2	4/21/2017 21:05	21	38.4864	33.7
11	2	4/21/2017 21:35	19.1875	39.1799	33.7999
12	2	4/21/2017 22:05	17.75	39.8167	34.2999
13	2	4/21/2017 22:35	16.5625	40.5953	35.2999
14	2	4/21/2017 23:05	15.625	41.1727	35.9
15	GPS Coord	38.34405	-122.593627	0	0
16	3	4/21/2017 23:35	14.8125	41.6583	36.2999
17	3	4/22/2017 0:05	14	42.5551	36.9
18	3	4/22/2017 0:35	13.3125	53.671	37.2999
19	3	4/22/2017 1:05	15.625	47.7884	47.2999
20	3	4/22/2017 1:35	16.625	46.1274	45.9
21	3	4/22/2017 2:05	15	45.8864	41.7
22	GPS Coord	38.344578	-122.594327	0	0

Figure 31: Fairfield Osborn Preserve: Left: UAV Flight Path with 200 ft Radii Right: Data in Excel

2.4 Hardware Design

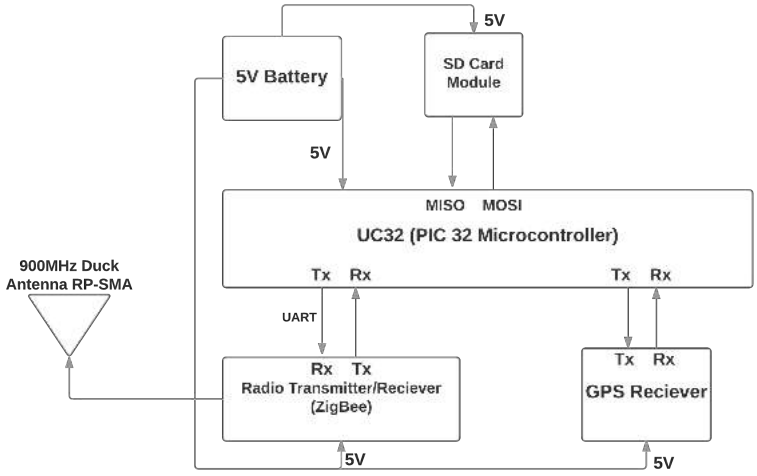


Figure 32: System Diagram of Mobile Node

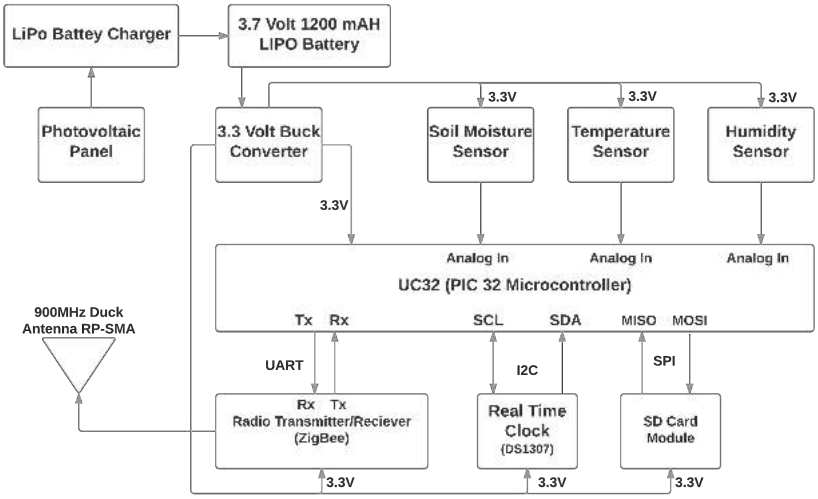


Figure 33: System Diagram of Sensor Node

3 Ethics of the Engineering Profession and Our Project

Like any field, ethics is important to engineering. There are a variety of things to consider when it comes to being ethical in our field. In the case of our project here are some things we must consider to ensure it was done in an ethical manner. One issue is when a team doesnt make due on the promises that they make on their product. We will ensure that we complete the aspects of our project that we promised we would complete. If any mistake is made during development we will take responsibility for our actions. Any reference used will be appropriately credited and we will not take credit for anything that we didnt do ourselves. Members of the group will not disclose information that is not intended for the public whether it be just among the group or on behalf of Sonoma State and its preserves or Aerotestra Inc.

In our case, we have been allotted a UAV shell by Aerotestra Inc. to work with because we will be using their UAV for the final product. Thus it would be prudent to use the donated materials respectfully and to make good use of them. Going further on the subject of UAVs, the FAA requires that a licensed personnel is present at all times while a UAV is being flown and that said UAV must been within view of said person. Thus, we will ensure that we will have such a person present while we are testing with the UAV. We will also ensure that our product is not left on the site where it is being used once it is no longer being used and thus not pollute the preserve that it is located at. When it comes to the funding that we will receive we will make good of use of it. We will spend it only on materials needed for the product and nothing else.

When it comes to the use of the device itself we will make sure that it will follow a set of key guidelines. The devices will provide reliable data and the devices themselves will be safe to use and do no harm to the user or the environment that it is placed in. The devices will work as promised and follow all IEEE standards that must be followed.

4 Future Work

Our project is a complete working product. However, that is not to say there could be improvements made to our design for future works, or new ideas using our existing project. There are a few features and ideas that could be looked at in the future:

1. Make the system completely autonomous. The UAV will be scheduled to go at different times and collect the data without having the user be present.
2. Using our sensor nodes, make a network where each sensor node is connected together and sends the data to a central node when asked. That way the UAV only has to go to one location instead of going node to node.
3. Have the data from the sensor nodes display data in real time
4. Make sensor nodes more modular for different sensors
5. Include some form a data checking to the transmission handshake (eg a checksum, parity bit, etc) to further improve data transmission reliability.
6. Change over from uC32 and WF32 boards to independent microcontrollers to further reduce size and cost of the nodes.
7. More user friendly GPS coordinate set up. (eg user could carry mobile node around while setting up sensor nodes and it keeps track of locations automatically).
8. Implement a data archiving system, where sensor nodes store old data in separate files in their memory and the user can set the mobile node to ask for old data if they want.

5 References

[1] "Airborne data system," DroneScan, 2016. [Online]. Available: http://www.dronescan.co/. Accessed: Oct. 4, 2016.

[2] R. Journal, "RFID-Reading drone tracks structural steel products in storage yard - page 1," 2014. [Online]. Available: http://www.rfidjournal.com/articles/view?12209. Accessed: Oct. 4, 2016.

[3] J. Gabay, "RF links for civilian drones," 2015. [Online]. Available: http://www.digikey.com/en/articles/techzone/2015/jan/rf-links-for-civilian-drones. Accessed: Oct. 4, 2016.

[4] "Remote sensing using UAVs," 2016. [Online]. Available: https://people.kth.se/ gonga/remoteuav.html. Accessed: Oct. 4, 2016.

[5] H. Xiang and L. Tian, "Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (UAV)," Biosystems Engineering, vol. 108, no. 2, pp. 174190, Feb. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1537511010002436. Accessed: Oct. 4, 2016.

[6] L. Reich, "Breakdown of drone remote sensing sensors," in Drones, Geoawesomeness, 2016. [Online]. Available: http://geoawesomeness.com/breakdown-drone-remote-sensing-sensors/. Accessed: Oct. 4, 2016.

[7] E. Marris, "Drones in science: Fly, and bring me data," News Feature, vol. 498, no. 7453, p. 156, Jun. 2013. [Online]. Available: http://www.nature.com/news/drones-in-science-fly-and-bring-me-data-1.13161. Accessed: Oct. 4, 2016.

[8] "UAV and GISAn emerging dynamic duo," 2014. [Online]. Available: http://www.esri.com/esri-news/arcuser/spring-2014/uav-and-gis-an-emerging-dyn amic-duo. Accessed: Oct. 4, 2016.

[9] A. Rosenblum, "Nanosatellites will stop the Internet of things from ever going Offline," MIT Technology Review, 2015. [Online]. Available: https://www.technologyreview.com/s/538726/nano-satellites-work-with-ground-s ensors-to-offer-new-eye-on-disaster-relief-and/. Accessed: Oct. 4, 2016.

[10]]"Spott-weiss," spott-weiss. [Online]. Available: http://spott79.wixsite.com/spott-weiss. Accessed: Nov. 27, 2016.

6 Appendix

6.1 Detailed Budget

Quantity	Part	Price	Description	Link
3	1200 mAh LiPo Battery	$9.95 each	Batteries used to power sensor nodes	https://www.adafruit.com/product/258
3	Digilent chipKIT uC32	$29.95 each	Boards used for sensor nodes	http://tinyurl.com/kpkkdr2
1	Digilent chipKIT WF32	$69.99	Board used for mobile node	http://tinyurl.com/le8cgf9
4	Xbee S3 Pro Radio Module	$39.00 each	Modules used for wireless communication	http://tinyurl.com/kpfw7n5
4	Xbee Breakout Board	$10.00 each	Board used to mount Xbees on	https://www.adafruit.com/product/126
3	Temperature Sensors	$9.95 each	Sensors to measure temperature	https://www.sparkfun.com/products/11050
3	Humidity Sensors	$29.99 each	Sensors to measure humidity	https://www.adafruit.com/product/1293
3	Soil Moisture Sensors	$49.95 each	Sensors to measure soil moisture	https://www.adafruit.com/product/1298
3	Weather Proof Casings	$14.97 each	Casings for sesnor nodes	http://tinyurl.com/m5lzzbw
1	GPS Module	$4.99 each	Module to get GPS coordiantes	https://www.sparkfun.com/products/13740
3	Real Time Clocks	$5.99 each	Record the time when data is collected	http://tinyurl.com/lnbfhau
3	Solar Panels	$8.99 each	Solar Panels to charge batteries	http://tinyurl.com/mgcq2yg
4	900 MHz Duck Antennas	$7.95 each	Increase the range of transmission	https://www.sparkfun.com/products/9143
4	Xbee Shield	$6.00 each	Shield for the UC32 board	http://tinyurl.com/kewud4e
4	16 GB Mirco SD Card	$6.95 each	SD card to save data	http://tinyurl.com/ktenoxh
3	Micro SD Card Module	$7.95 each	SD card Module for UC32	https://www.adafruit.com/product/254
3	3.3V Buck Converter	$4.95 each	Regulate voltage to UC32	https://www.adafruit.com/product/2745
3	LiPo Battery Charger Board	$5.99 each	Charger board	http://tinyurl.com/l8czpoj
	Total Cost	$ **860.53**		

Table 7: Complete Budget

6.2 Example Customer Survey

Customer Survey:
Data Harvester

Customer Name:_____

Please fill out this survey and send it back to us as soon as possible.

	Answer (yes/no)	Comments
Don't have the time to walk/hike to remotely placed sensors	no	Depends on how often I have to
Need to collect remote sensor data regularly.	no	
Want to be able to collect remote data automatically.	yes	
Do you have sensors gathering data in hard to reach locations?	yes	
If it were easy, would you like to have remote sensors?	yes	
Would you like to learn or are you interested in exploring the possibilities of drone technology in the environment?	yes	

If you are interested in this project, please fill out our second survey on the next page, if not simply fill out this first one as best you can.

Otherwise, thank you for your time, we appreciate your input.

Please fill out this survey if you are interested in our project:

	Answer (yes/no)	comments
Are you interested in gather environmental data? (temperature, humidity, soil moisture, etc)	**yes**	
Would you be willing to get a drone flying license from the FAA?	**yes**	**If it isn't too expensive in time or money**
Would you be interested in having a modular system that allows for the use of any kind of sensor?	**yes**	
Would you want to fly the drone yourself?		
Would you rather the drone be autonomous?	**no**	**Depends but not likely**
You would like the data to be stored on a removable SD card?	**yes**	
You would like the data to be uploaded to a computer automatically via bluetooth?	**yes**	
Is there a specific period of time that you want to gather your data? (daily, weekly, monthly, etc)	**Don't know**	

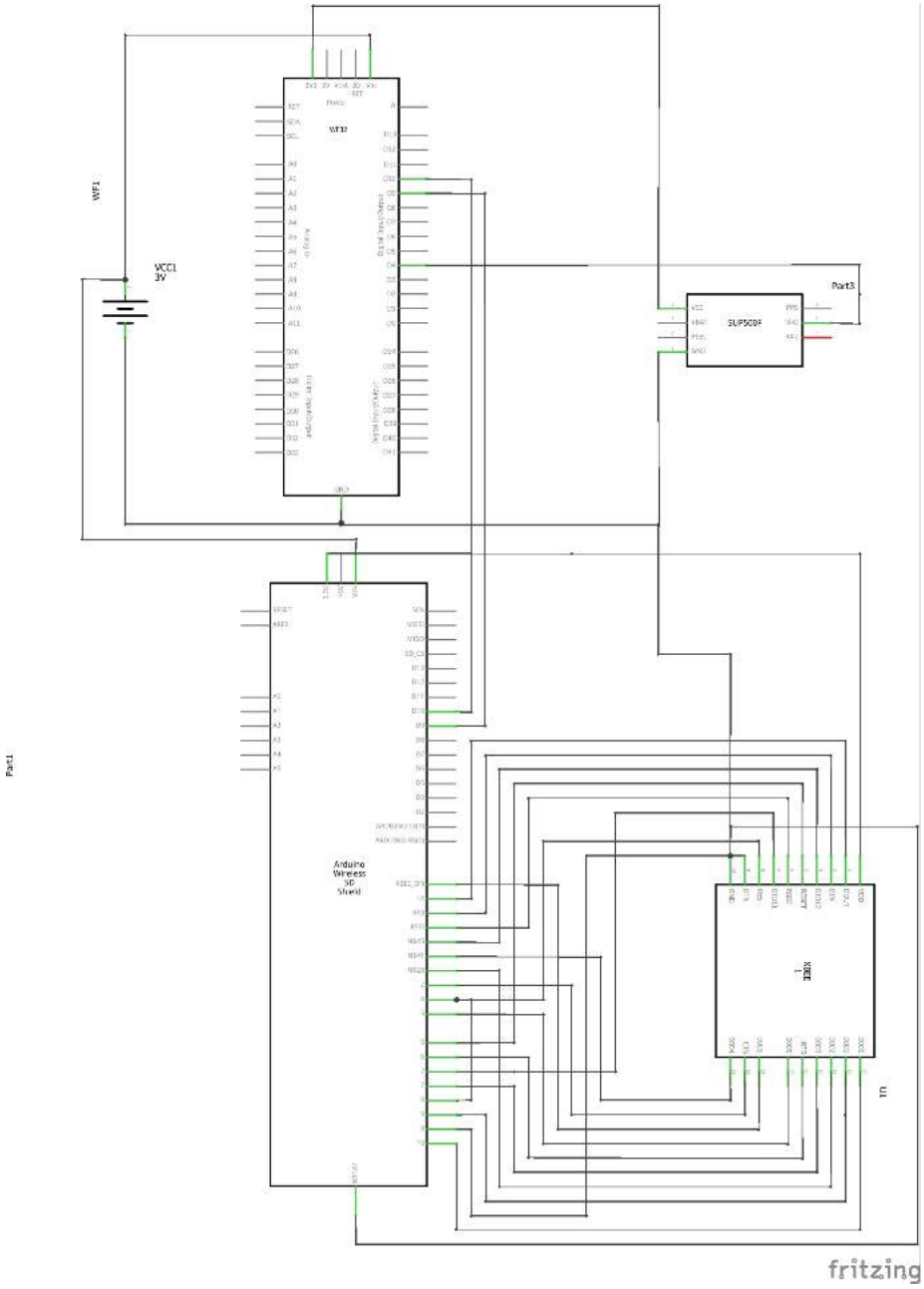
Thank you for your input, we appreciate you spending the time to help us in our endeavors.

The Data Harvester Team:
Taylor Jones
Abe Palmerin
Josh Papanicolas

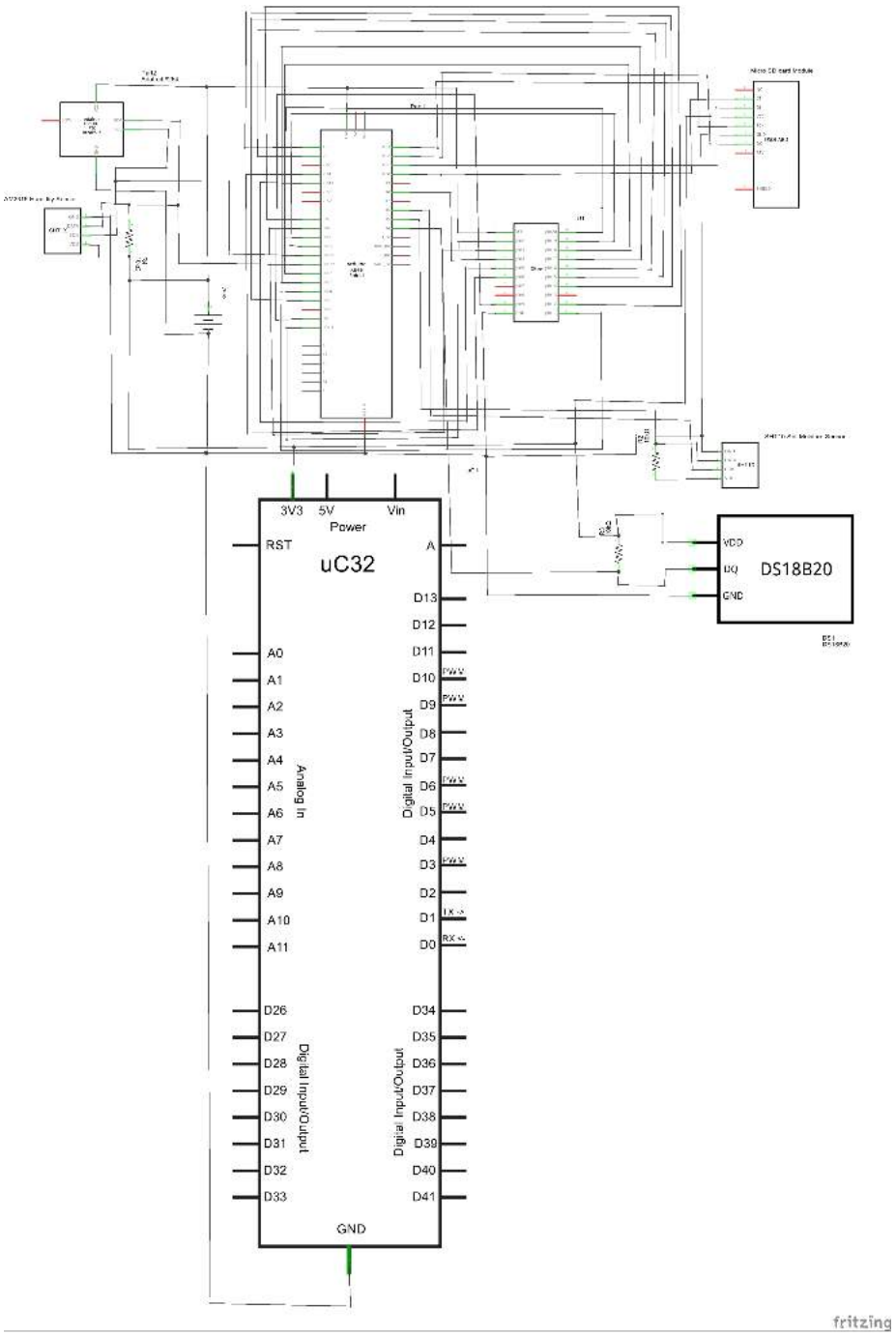
It was difficult to find people applicable to this project. Many students and faculty were not interested in taking our surveys since they had no purpose of collecting data or using a UAV. Environmentalists and UAV hobbyist were most interested in the application of our project. We also found out that some environmentalists preferred to walk far distances to collect the Sensor Node data themselves rather than a UAV doing it for them. On the other hand, some people enjoyed the fact that they did not have to go to isolated areas everyday to get their data.

6.3 Circuit Diagrams

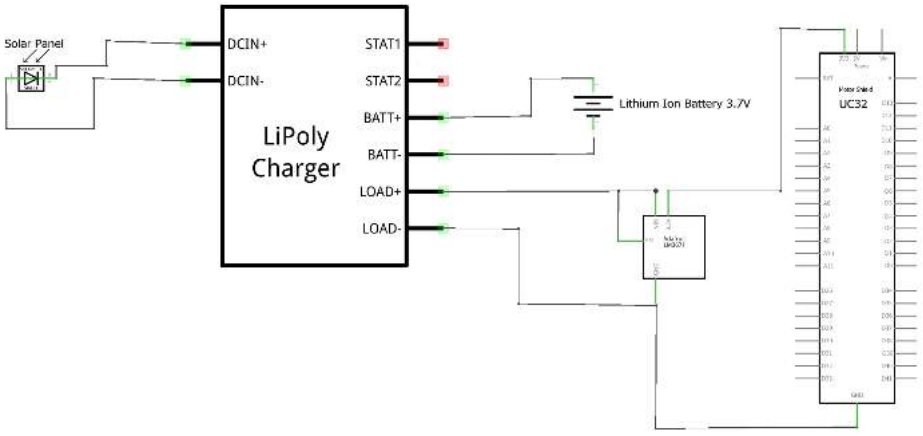
6.3.1 Mobile Node



6.3.2 Sensor Node



6.3.3 Sensor Power Supply



7 A Special Thanks

We would like to thank our faculty advisors Dr. Chris Halle and Dr. Farid Farahmand for their help and support. Our industry advisor, Mr. Sean Headrick and Aerotestra for making our final test possible.The SSU SOURCE award for funding this project and making it possible. MESA coordinator, Dr. Carolyn Peruta, for donating soil moisture sensors for this project. Our intern, Micaela Bush of MESA, at Santa Rosa Junior College for helping us with necessary testing at the Pepperwood Preserve and project documentation. Suzanne Decoursey, reservations manager at the Fairfield Osborne Preserve, for letting us test this project at the preserve. Lastly, our friends and family for their support.