# tree hugger and the Hug Net Final Report For ECE 445, Senior Design

By

Jinge Li

Ruiqi Zhu

Yangwentao Fang

# Abstract

Because climate change is becoming a more important issue, environmental scientists need to collect data about carbon. tree hugger is such a device to monitor and record carbon amount in every tree that it's attached to. Our project focuses on creating a base station and wirelessly collect the data from the tree huggers.  This paper discusses the design for our mesh network and our base station.

# Contents

# 1. Introduction

## 1. 1. Statement of Purpose

Currently retrieving data from tree huggers has to be done by collecting each SD card from tree to tree. This problem was presented to us by Professor DeLucia and we are implementing a solution for him. Our project aims to develop wireless communications between tree hugger devices and creating a mesh network for further data transfer.

## 1.2. Objectives

### 1.2.1. Goals and Benefits

- Implement wireless data transfer for each tree hugger device.
- Design and construct a base station to collect data wirelessly from nearby tree hugger devices.
- Increasing wireless area coverage through mesh network.
- Make data collecting easier and more efficient.
- Protects the tree hugger device from weather and exposure, since SD card does not need to be manually taken out.

### 1.2.2. Functions and Features

- tree huggers being able to send and receive data.
- Send data wirelessly from tree hugger to the base station.
- Store collected data on both the tree hugger's SD card and the base station's SD card.

# 2. Design

Our whole system includes two main parts: the base station and the tree huggers. The following figure 1 shows our overall design of our system.

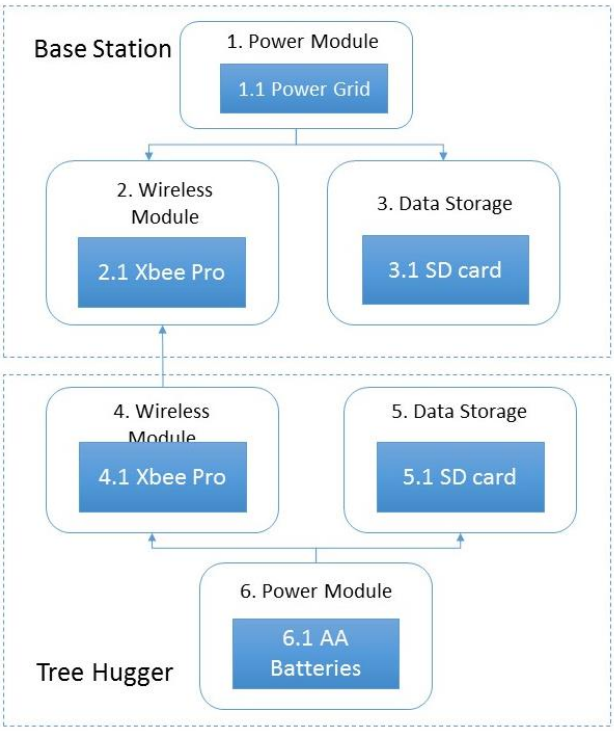



Figure 1. Overall system block diagram.

## 2.1. base station Hardware

### 2.1.1. Design Procedure

The base station acts as the main hub of the mesh network. All the data from tree huggers in the mesh network is eventually transmitted to the base station and stored in the SD card on the base station.

The base station is implemented to fulfill the following requirements:

1. Wirelessly collect data constantly
2. Store data onto the SD card
3. Stable and reliable power supply

The figure 2 shows the block diagram for the base station. The figure 3 shows the actual hardware product of the base station.

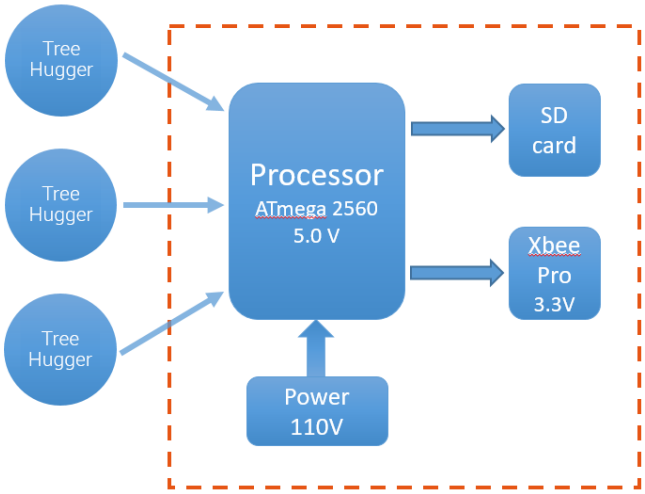



Figure 2. Block diagram of the base station.

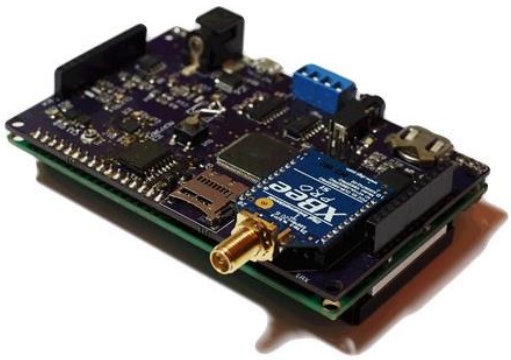


Figure 3. Base station final product.

## 2.1.2. Design Details

### *Processor -- ATmega 2560*

The processor controls and coordinates every other chip on the base station board. Hence, the processor needs to be easily programmable, and expandable. In the initial design process, we have considered using a Raspberry Pi as the foundation of the base station board. However, there are several disadvantages. The first disadvantage is the cost of Raspberry Pi, which is very high. Secondly, Raspberry Pi is very powerful, but we will not utilize most of its functions so this make the whole design inefficient. We finally have decided on the ATmega 2560 chip because we would like the base station to be more powerful than the tree hugger. The ATmega chip has more I/O pins, bigger flash memory, larger SRAM. The main advantage of the ATmega 2560 chip is its large number of I/O pins, which could enable us to expand its functionalities in the future easily. The ATmega chip operates between the voltage input of 4.5V and 5.5V, while the Xbee Pro chip operates at 3.3 V. The difference between their operating voltages require us to design two different voltage inputs on the base station board. [1]

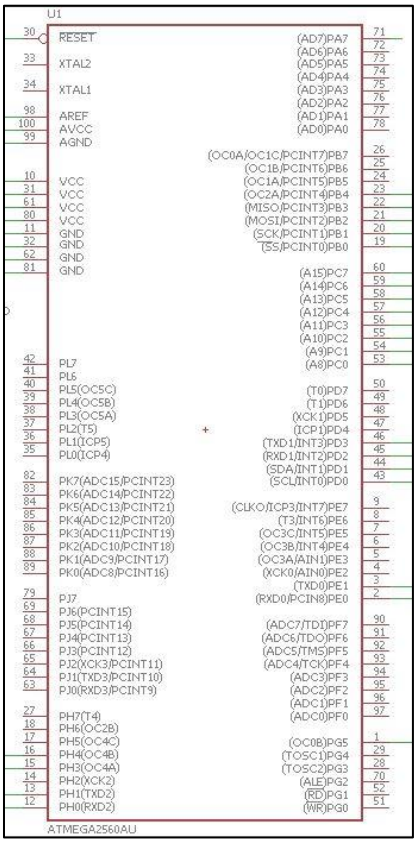



Figure 4. Eagle CAD schematic of the ATmega 2560.

### *Power Supply Board*

The base station constantly receives large amount of data packets. Therefore, the base station consumes more power than the tree hugger device. Moreover, the power of the base station needs to be stable and reliable. Having considered these factors, we have decided to use the 110 V from power outlet. The 110 V is too high for most of the on-board regulators, so we first draw the power from the power grid by using a AC power adapter, which would output 5V. Then we chose two power regulators to regulate power to 3.3V and 5V. To achieve this purpose, we have chosen SPX29300T and 29302U5 chips with two resistors. [2]

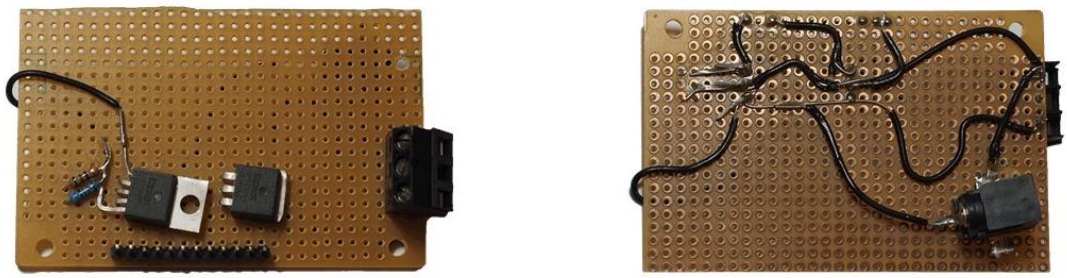


Figure 5. Power supply board upside (right) and bottom side (left).

### *Xbee Pro Series 1 and Configuration*

The Xbee model we chose for our project is Xbee Series1 Pro and there are three reasons. First, this specific model of Xbee has the capability of supporting the 802.15.4 firmware. The 802.15.4 firmware allows each device in the network to sleep when it's not transmitting so that a more energy efficient network can be implemented. The second reason is that this firmware also has an open sourced library that allows users to easily program the Xbees through Arduino. Lastly, this model of Xbee has sufficient transmission range to be deployed in a forest environment. [3]

The first step during configuration is to make sure that all Xbees have the same 802.15.4 firmware since this firmware allows each device in the network to be configured as an end device. This ensures that each device can go to sleep when they are not transmitting. The second step is that each device within the same mesh network must have identical channel ID. Next, the destination high and low are respectively set to 0 and 0xFFFF so that they can communicate with every other device within the network. Lastly, each Xbees are configured to API mode with escape so that we can utilize the open sourced library to program them.

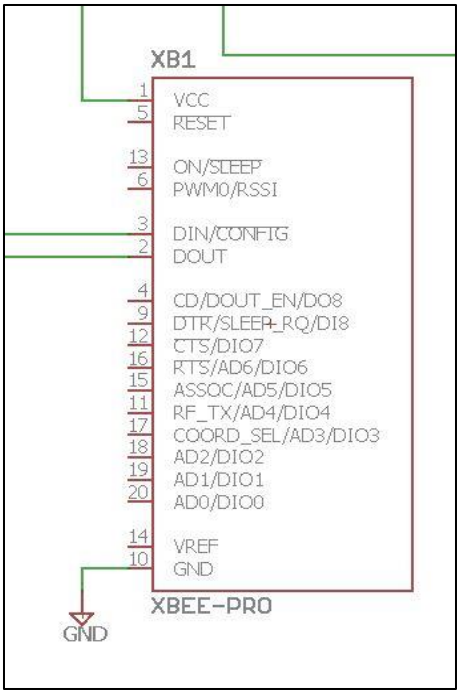



Figure 6. Eagle CAD schematic of the Xbee Pro Series 1.

### *Voltage Protection*

We have striven to design the base station as stable as possible. We do not wish to see any overvoltage, undervoltage, and reverse voltage damaging the board and sabotaging all the data. Therefore, it is essential to include voltage protection as part of our design. Since the power for the base station must always pass the AC adapter, which outputs 5V. We would only a voltage protector that spans from 5v to -5v. We have chosen the LTC4365 as our voltage protector. One of its advantage of the LTC4365 is its wide operating range, between 2.5V and 34V. Secondly, it has overvoltage protection up to 60V and reverse supply protection up to -40V. It also blocks 50Hz and 60Hz AC power, which are two main power frequency used around globe. The LTC 4365 works by controlling the gate voltages of a pair of external N-channel MOSFETs to restrain output in a safe range. The LTC4365 also consumes very little power, only 125uA in normal operation. [4]

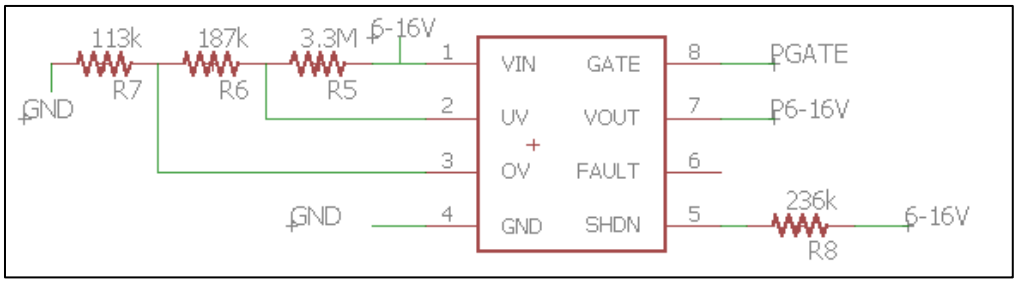



Figure 7. Eagle CAD schematic of the LTC4365 and corresponding resistors.

### *I/O Expansion*

To make the board easily expandable, we have also decided to expand on the existing I/O pins of the ATmega 2560 in our designing PCB process. We have chosen to use the PCF8574 chip as the I/O expansion. [5] The PCF8574 operates between 2.5 V and 6V Vcc power supply. We are supplying it 5.0 V Vcc, shared with the ATmega 2560 chip. The device features an 8-bit quasi-bidirectional I/O port (P0–P7), which are enough for expansion purpose.

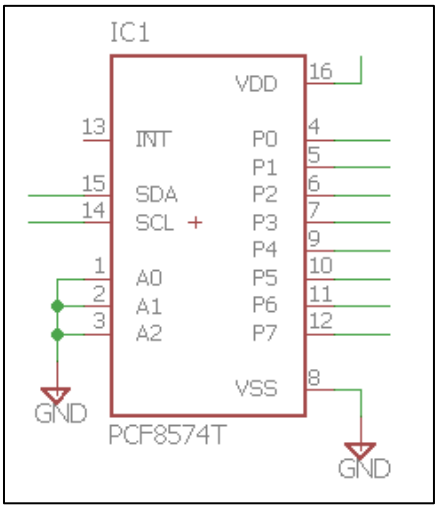



Figure 8. Eagle CAD schematic of the I/O expansion board PCD8574.

### *Line Driver/Receiver*

To overcome the capacitance caused by too many devices on the bus, we have decided to use a line driver. The line driver behaves like a buffer and amplifier. For this purpose, we have chosen the MAX232SOIC16. [6] This chip operates at 5.0 Vcc power supply, same as the ATmega2560 and PCF8574 chips. Hence, they can share the same Vcc supply.

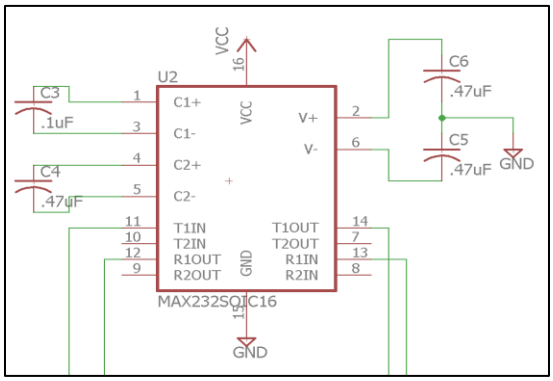



Figure 9. Eagle CAD schematic of the line driver/receiver MAX232SOIC16.

### *GSM/GPRS module*

We would like to expand the wireless function in the future, so we have chosen to include a GSM/GPRS module in our PCB design. This could enable users to fetch data from the base station at home if there is cellular coverage in the forest.

For this purpose, we have chosen the SM5100B module. It operates between 3.3V to 4.2V, so it can share the same voltage with the Xbee Pro chip. When the chip enters sleep mode, it only consumes less than 2mA, which is very power efficient. [7]

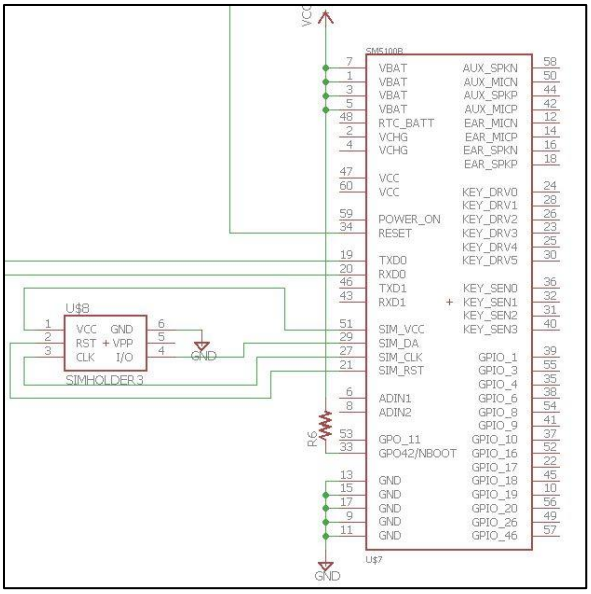



Figure 10. Eagle CAD schematic of the GSM/GPRS module SM5100B.

### *Real Time Clock*

We also need a real-time clock to be able to deliver time stamp to tree huggers for our future work. For this purpose, we have chosen the $I^2C$ RTC with integrated crystal and SRAM. This unit has a battery power input in case of interruption of main power. It also operates at 3.3V, which is supplied by our SPX29300. [8]

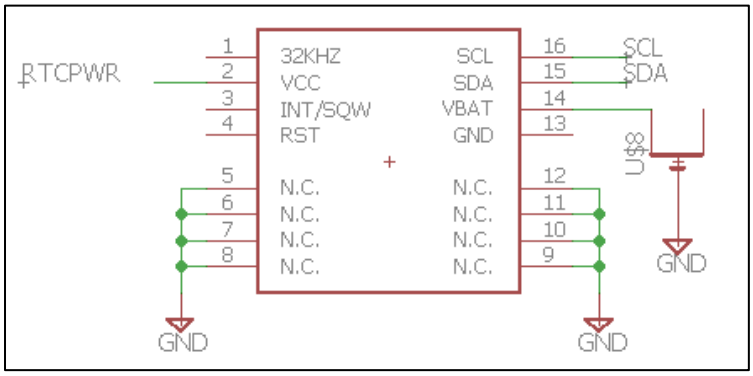



Figure 11. Eagle CAD schematic of the real-time clock DS3232.

### 2.1.3 Path Loss

The Xbee module attached to the tree hugger device is XBee Pro S1 (data sheet can be found in the citation). It has an indoor/urban range of 100m, frequency of 2.4 GHz, and a transmitter and receiver gain of 15 dBi. With this information, we can calculate a Free Space Path Loss of 50 dB by using the following equation:

$$FSPL = 20\log_{10}(d) + 20\log_{10}(f) + 20\log_{10}\left(\frac{4\pi}{c}\right) - G_t - G_r$$

Eq. 1. Free space path loss equation. [9]

Where:

$$\mathrm{d} = 100\mathrm{m}$$

$$\mathrm{f} = 2.4\ \mathrm{G}Hz$$

$$G_t = G_r = 15\ dBi$$

We get:

$$\boldsymbol{FSPL = 50\ dB}$$

### 2.1.4. Link Budget

According to the data sheet, XBee Pro S1 has a transmit power of 18 dBm. So, we can calculate a received power of -17 dBi by using the following equation:

Transmit power – Path loss + Receiver Antenna Gain = Received Power

Eq. 2. Received power equation. [9]

$$18 - 50 + 15 = -17\ \mathrm{dB}\boldsymbol{i}$$

Once we have the Received Power, the Link Margin is the difference between the receiver's sensitivity, in this case -100 dBm according to the data sheet, and the received power, -17 dBi. Thus, we have a Link Margin of **83 dBi**.

Finally, we can calculate the range of the Xbee module by applying the following Link Budget equation:

$$range = antilog(\frac{P_{TX} + G_{TX} - L_M + G_{RX} - P_{RX} - 104}{20}) \text{ miles}$$

Eq. 3. Xbee Pro communication range equation. [9]

Where:

$$\mathrm{P_{tx}} = 18\ \mathrm{d}B$$

$$\mathrm{G_{t}}x = \mathrm{G_{r}}x = -15\ \mathrm{d}B\mathrm{i}$$

$$\mathrm{L_m} = 83\mathrm{d}B\mathrm{i}$$

$$\mathrm{P_{rx}} = -100\ \mathrm{d}B\mathrm{m}$$

In our case instead of 104 dBi, we have a Path Loss of 50 dB, a transmit power (PTX) of 18 dBm, a transmitter and receiver gain (GTX/GRX) of -15 dBi, a Link Margin (LM) of 83 dBi, and a receiver sensitivity of -100 dBm. So, we can calculate a final range of **0.0056 miles or 9.012 meters** for the XBee module. (XBee Pro 60mW U.FL Connection - Series 1)

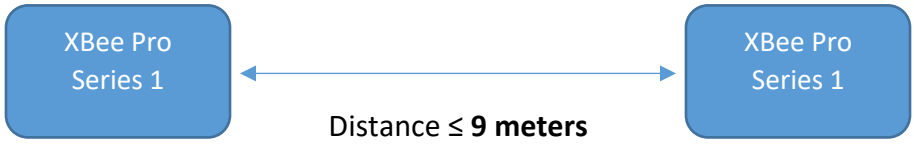



Figure 13. Maximum communication distance for Xbee Pro Series 1 chips in the forest condition.

## 2.2. Software

### 2.2.1. Data Transferring System

Figure # provides a flowchart for the main logic process for our data transferring system. Flowcharts of subroutines are provided in figures # and #. First, we do power control initialization so that SD cards are sufficiently powered to be read and written. Then we would set the destination address, so this Xbee module knows where to be sending its data to. In each main loop, we would call receive to check if there are data to log. Then we would check to see if time constraint is meet. If time constraint is met then we would call sent function and finish this loop cycle, else if not met we would just finish this loop cycle.

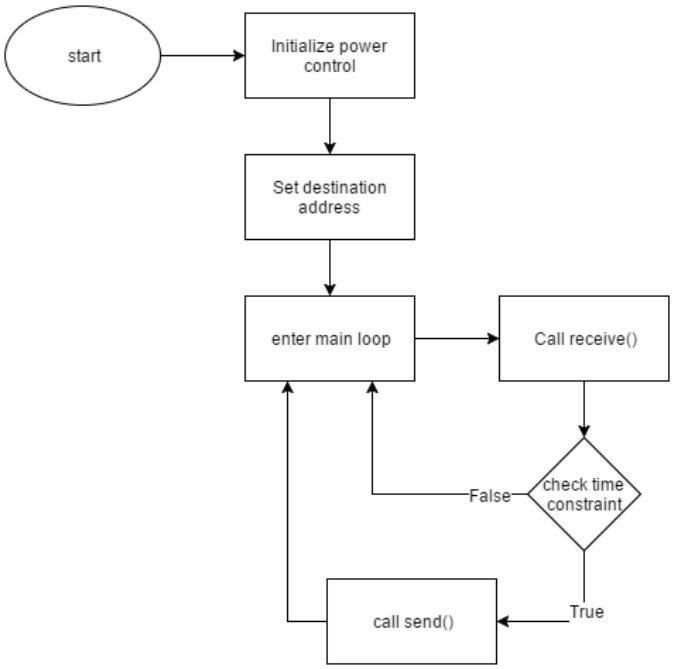



Figure 14. Atmega328p main loop flowchart.

The receiving procedure is a mechanism that happens every main loop cycle which follows Figure # below. While the device is running it will keep scanning, and attempt to read packets through the Xbee module. When it is determined that a packet is available it will check whether the packet type is RX16 or RX64 and get the corresponding data. Finally, the local CSV file is opened and the data retrieved through the packet will be printed to the local CSV file.

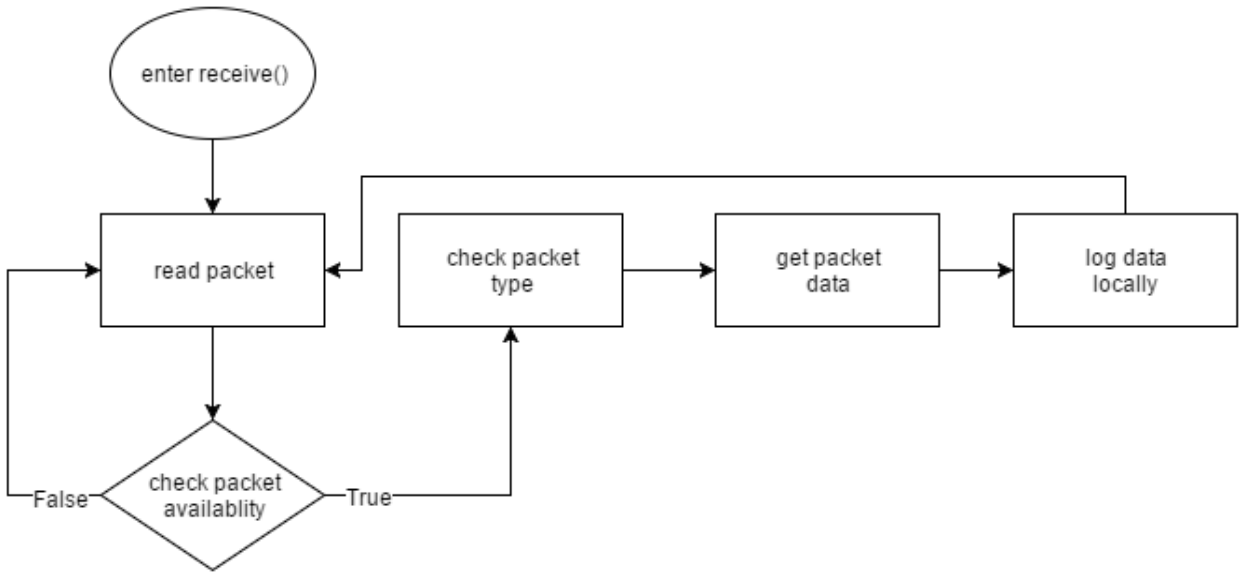



Figure 15. Receive() function flowchart.

The send function starts by opening the data log file that's locally saved on the SD Card. After that it follows a periodic procedure in which it parses one line of the log file in each cycle, converts the line into a packet and sends the packet to a destination XBee. The procedure cycle will run until all lines in the file are send.

The packet creation procedure utilizes a payload. It has a capacity that goes up to 100 bytes since each line of data in the local log will not exceed 100 characters. Once a line in the local log is finished parsing each character is stored into the payload for transmission. The transmission procedure make use of a Tx64Request since in our XBee configuration its address is 64 bits. The Tx64Request then loads the payload and the destination address of the tree hugger the data wants to be sent to. And finally the Tx64Request is sent through the xbee.send() function to complete the sending procedure for one line.

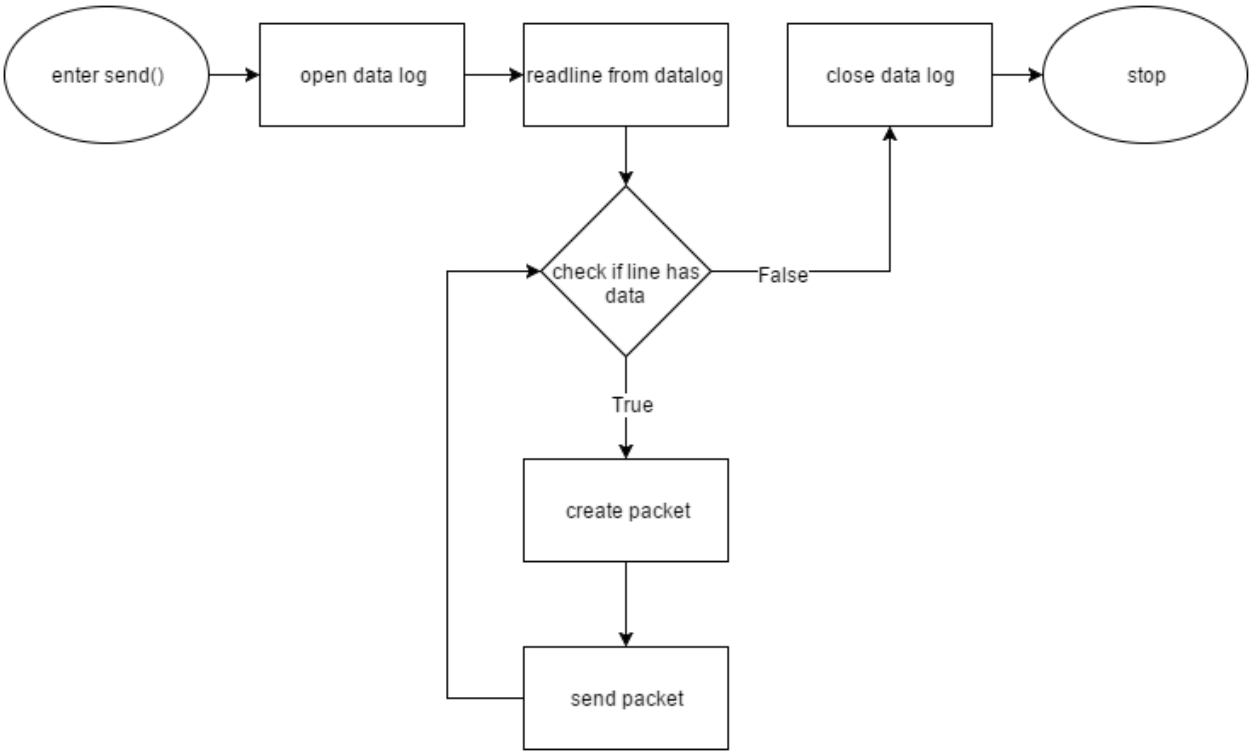



Figure 16. Send() function flowchart.

### 2.2.2. Mesh network

#### *Motivation*

We decided to create a mesh network for our tree huggers, because XBees have limited communication range and we want to collect data from tree huggers that are distant from the base station. From our linked budget, we estimated the Xbees on the tree huggers that are in a dense forest is able to reliably transmit data within nine meters' radius. So, by creating a mesh network and layering the tree huggers, we can extend the range of devices which the base station could receive data from.

#### *Network Description*

The Xbee Pro S1 we utilize in this project has the capability to set it's sending destination address in arduino code using the open source xbee.h library. Each Xbee is assigned with a unique serial address which we use as an identification when sending and receiving. To create the mesh network, we individually coded each tree huggers' destination address so that they are sending data to the correct receiving Xbee.

#### *Network Design*

Since the Xbees only have a range that goes up to nine meters according to our link budget, multi-hop is required during data transmission to cover a larger area. So, the mesh network structure we chose to complete this task is a tree type of network with different layers based

on each node's distance to the base station. So, for example the tree huggers that are closest to the base station while not exceeding the range of nine meters are the first layer of the mesh network and they don't require any type of multi-hop during data transmission. However, they do act as intermediate nodes for the nodes that are in the outer layer of the network so that those tree huggers can multi-hop and transmit data to the base station from a further distance.

An example of Mesh Network is below. The top red node is the base station and all the nodes below are child tree huggers that relay data upwards towards the base station. Nodes of the same color signifies they are in the same layer.

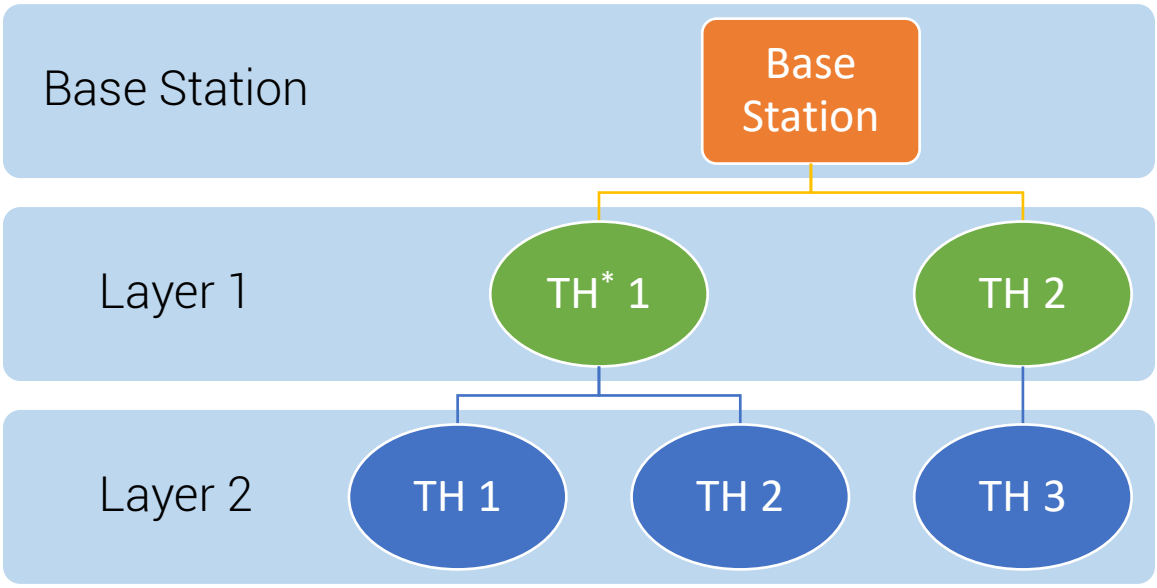



Figure 17. Example of Mesh Network (TH* = tree hugger).

### *Synchronization*

The synchronization of this network is done via a scheduling mechanism. Each tree hugger is programmed such that it has a layer order and a serial order. The layer order determines which layer the tree hugger is in the network and the serial order is the tree hugger's id number within that layer of tree huggers. During data transmission, each tree hugger is assigned with a delay time based on its layer order and serial order. The final delay each tree hugger receives will be calculated based on the following equation:

Scheduling Delay = Layer Order * Layer Delay + Serial Order * Serial Delay

The more outer layer the tree hugger is the longer it would wait to transmit its data. When comparing tree huggers within the same layer, the larger the tree huggers' serial number the

longer it will wait. This synchronization method insures that outer layer tree huggers will send first to inner layer tree huggers and tree huggers within the same layer would not interfere with each other by sending data at the same time.

According to our verification, this design does exactly what we intend it to do. With the proper destination setup and scheduling management, we can create a mesh network which extend to multiple layers and allow tree huggers from distance transferring data sequentially towards intermediate node and base station. So, this design can handle a large coverage area. Thus, allows data collection to be much easier.

There are two major constraints with this design. First, this entire mesh network is hard-coded. Each Xbee's sending destination must be preemptively configured before putting them into the network. This means even though this design works, it is extremely inflexible in terms of adapting network dynamics. For example, if a non-leaf tree hugger is lost due to hardware failure, all its children tree huggers will fail as well simply because they have lost their destination/intermediate node to the base station that's hard coded to them.

The second constraint is power consumption. The intermediate tree huggers that are closer to the base station will have a much intenser power consumption than the leaf tree huggers because as intermediate nodes they must transmit data that's equivalent to all their children's data. This problem is less observative in a smaller scaled mesh network that contains low number of layers. However, the battery lifes on the tree huggers are shortened at an exponential rate the closer they are to the base station.

# 3. Design Verification

Design Verification: This section details all the tests performed to ensure the functionality of all the software and hardware implementations of the system.

## 3.1. SD card

We verified the functionality of SD Card by having a tree hugger read and write data to the SD card. For more details on the specific requirements and testing procedures, please see Appendix B.

## 3.2. Wireless Module:

We used two programmed tree huggers to test the wireless communication, range and data transmission. To test the wireless communication, we attached two Xbees to the computer using Xbee explorers. After configuring them we turned them on and try to send packets from one to the other. The observation through XCTU indicates that the correct packet is received by the receiving Xbees. So it's verified that the basic communication is working between configured Xbees.

The test of range is built based on the previous verification. Instead of attaching both Xbees on the same computer, we attached two Xbees on two different computers and have them placed nine meters apart from each other. We then sent packets from one to the other and was able to confirm that the packets are received with a nine meter range between the two.

Finally we verify the data transmission between tree huggers by programming the sending and receiving code to two different tree huggers. We then power both of them up and have one send data to the other until the entire local data log on the sending tree hugger is transmitted to the other one. We then took the SD card on both tree huggers out and compared the data received against data sent and found out they are identical. This verifies that the data transmission between Xbees on tree huggers works. For details on the specific requirements and testing procedures, please see Appendix B.

## 3.4. Power Regulator Board Verification

The first verification we must do is to test the power regulator board is supplying the correct 3.3 V and 5.0 V to the base station board. Without this step, we cannot test the connections and functions of the components on the base station. To verify the power regulator board is supplying the correct voltage, we measure the 3.3 V output and 5.0 V output from both power regulator chips by using a multimeter. The results are shown in the table 1.

| Power Regulator Model | Theoretical Output Voltage | Measured Output Voltage |
|---|---|---|
| **SPX29300T** | 3.3 V | 3.285 V |
| **29302U5** | 5.0 V | 4.891 V |

Table 1. Verification data for power regulator board.

The results match our theoretical data, which prove we have the correct power supply for our base station board.

## 3.5. Power Verification

After having the correct power supply to the base station board, we need to ensure every component is receiving the correct power supply. We measured the pin on each component and compare with our theoretical data. The results are shown in the table 2.

| Chip | | Measured |
|---|---|---|
| **ATmega 2560** | 5.0 V | 3.294 V |
| Xbee Pro Series 1 | 3.3 V | 3.287 V |
| LTC4365 | 5.0 V | 4.856 V |
| PCF8574 | 5.0 V | 4.956 V |
| MAX232SOIC16 | 5.0 V | 4.926 V |
| **SM5100B** | 3.3 V | 3.293 V |
| **DS3232** | 3.3 V | 3.296 V |

Table 2. Power supply voltage verification data for the base station's components.

The ATmega 2560 chip is receiving incorrect voltage supply of 3.294 V instead of 5.0 V. We believe this error has caused our ATmega 2560 unable to receive data from the Xbee Pro chip and store onto the SD card. The voltage has led to the malfunction of the software. We tested the software on tree hugger at the correct operating voltage, and everything works correctly. But, when we lower the voltage, the tree hugger becomes also unable to receive data and store onto the SD card.

## 3.6. Wiring Verification

For our project, the main functions that we wish to achieve is to be able to receive data from the Xbee Pro chip and store data onto the SD card. To verify that the three modules have successful data connection, we set the Data Out pin on the ATmega 2560 to high (3.3V) and measure the voltage of the correspondent pins on the SD card slot and the Xbee Pro chip. The results are shown in table 3.

| Chip | Din | Measured Din |
|---|---|---|
| Xbee Pro Series 1 | 3.3 V | 3.287 V |
| SD card slot | 3.3 V | 3.268 V |

Table 3. Power supply voltage verification data for the base station's components.

As we could see from the above table, the ATmega 2560 chip, Xbee Pro chip, and the SD card slot have successful data connections.

# 4. Costs

## 4.1. Labor Cost

| Name | Hourly Rate | Total Hours | Invested Total |
|---|---|---|---|
| Ruiqi Zhu | $50 | 250 | $12500 |
| Jinge Li | $50 | 250 | $12500 |
| Yangwentao Fang | $50 | 250 | $12500 |

## 4.2. Part Cost

| Name | Quality | Cost/Unit | Cost |
|---|---|---|---|
| Battery | 2 | $2.5 | $5 |
| Flash memory | 1 | $11.95 | $11.95 |
| Arduino | 1 | $25 | $25 |
| Wireless Adapter | 1 | $10 | $10 |
| Xbee | 2 | $40 | 80 |
| Various resistors | many | $10 | $10 |
| ATMEGA2560AU | 1 | $15 | $15 |
| GSM/GPRS SM5100B | 1 | $50 | $50 |

## 4.3. Grand Total: $37706.9

# 5. Conclusions

## 5.1. Accomplishments

We conclude this report by evaluating our project as a whole. We were able to implement the basic functionalities of our requirements. The tree huggers have been successfully prototyped such that their wireless capabilities are utilized for remote data transmission. To utilize wireless transmission further we were able to create a mesh network that covers a large enough area and allows data from distanced tree huggers to be collected.

## 5.2. Uncertainties and Future Work

However, there are still some challenges that we struggled and were not able to implement. First of all, despite the fact that the mesh network is implemented and functions exactly as how we intend it to be, it is extremely inflexible to network dynamics. This will be a common theme when the devices are deployed into forests. For example, if an intermediate tree hugger node within the network is lost due to hardware failure, our current mesh network algorithm will not allow all its children nodes to switch to a new intermediate destination. This means that once an intermediate tree hugger is lost, all its children's data will also be lost. So the first step of future work that can be done to improve this project is the implementation of a more dynamic and self-healing type of mesh network so that it can diagnostic itself when encountering changes.

The next major challenge also has to do with our mesh network design. Although our design allows it to cover a large area of forests by making the mesh network multi-layered and multi-hopped, it causes an inevitable power consumption problem. For the tree huggers that are within the closer layer of the base station, they carry a much heavier burden in terms of power consumption when compared against the leaf nodes. The reason is that not only do they have to stay awaken longer to receive all their children node's data, but also they have to transmits all those data on top of their own data to the next intermediate tree hugger node. This phenomenal is a general attribute of multi-hopping mesh network. For each mesh network layer that's closer to the base station, the battery life on those tree huggers decreases exponentially. There are two solutions that can help improve the power consumption issue. First one is to simply support these devices with stronger power supplies since power consumption will always be more intense on intermediate nodes. On top of that, the larger the network is the more power intense intermediate nodes are since they will be responsible for more data reception and transmission. By lowering the scale of the mesh network we are able to relieve some power consumption issue with the price of reducing coverage area.

Our final challenge revolves around our base station. Despite the software designed for the base station functions as intended, we were not able to get the hardware of the base station to work due to wiring and power supply issue. One of the future work that can be done to finish up the base station design is to diagnostic the base station wiring so that it will have proper power supply to run. However, a better solution is to simply replacing the ATmega 2560 micro-controller on the base station with the ATmega 328p micro-controller from the tree hugger. During our diagnostic for the base station we found out that the tree hugger is completely sufficient for the functionality of the base station. By replacing the micro-controller not only can we supply the base station with proper power supply so that it functions, but also, we can share the same software archetype on both the base station and the tree hugger to ensure consistency.

## 5.3. Ethical consideration

The purpose of this project is to develop a wireless communications and mesh network for tree hugger devices so that scientist can safely and efficiently. Scientist would not need to trek through tough conditions in rain forests. We also had to sure. This follows the first code of the IEEE Code of Ethics [10]: “accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment.”

We will make sure that our wireless devices will not be negatively the environment and other people. This follows the ninth code of the IEEE Code of Ethics: “To avoid injuring others, their property, reputation, or employment by false or malicious action.”

# References


[1] "ATmega2560," Atmel, Feb 2014. [Online]. Available: http://www.atmel.com/devices/atmega2560.aspx.

[2] "SPX29300," EXAR, 2019. [Online]. Available: https://www.exar.com/product/power-management/power-conversion/ldos-and-regulators/linear-ldos/spx29300.

[3] "XBee® 802.15.4," DIGI, [Online]. Available: https://www.digi.com/products/xbee-rf-solutions/modules/xbee-802-15-4.

[4] "LTC4365 - Overvoltage, Undervoltage and Reverse Supply Protection Controller," Linear Technology, 2016. [Online]. Available: http://www.linear.com/product/LTC4365.

[5] "PCF8574," TEXAS INSTRUMENT, [Online]. Available: http://www.ti.com/product/PCF8574.

[6] "Texas Instruments MAX232DR," MOUSER ELECTRONICS, [Online]. Available: http://www.mouser.com/ProductDetail/Texas-Instruments/MAX232DR/?qs=8Pd2FuFSoMFWYM0h5A6gNw%3D%3D.

[7] "GSM/GPRS Module - SM5100B," Sparkfun, [Online]. Available: https://www.sparkfun.com/products/9533.

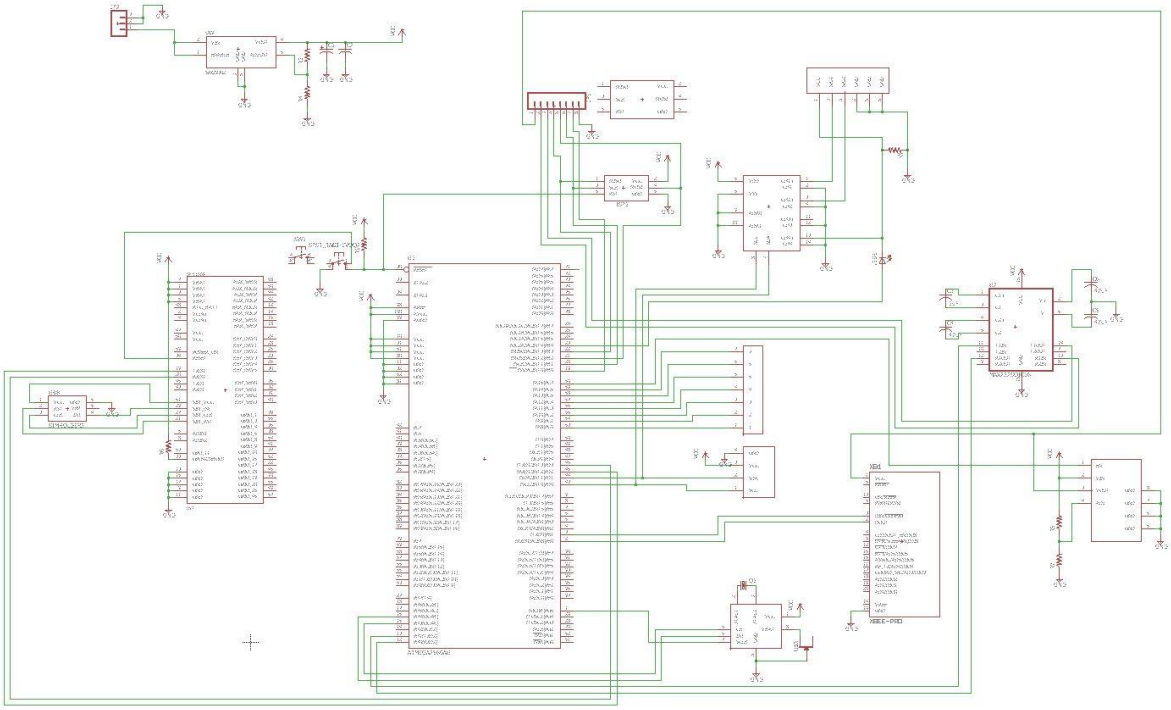
[8] "Extremely Accurate I²C RTC with Integrated Crystal and SRAM," Maxim Integrated, [Online]. Available: https://www.maximintegrated.com/en/products/digital/real-time-clocks/DS3232.html.

[9] T. Fagerness, "Estimating Wireless Range," All About Circuits, [Online]. Available: http://www.allaboutcircuits.com/technical-articles/wireless-range/.

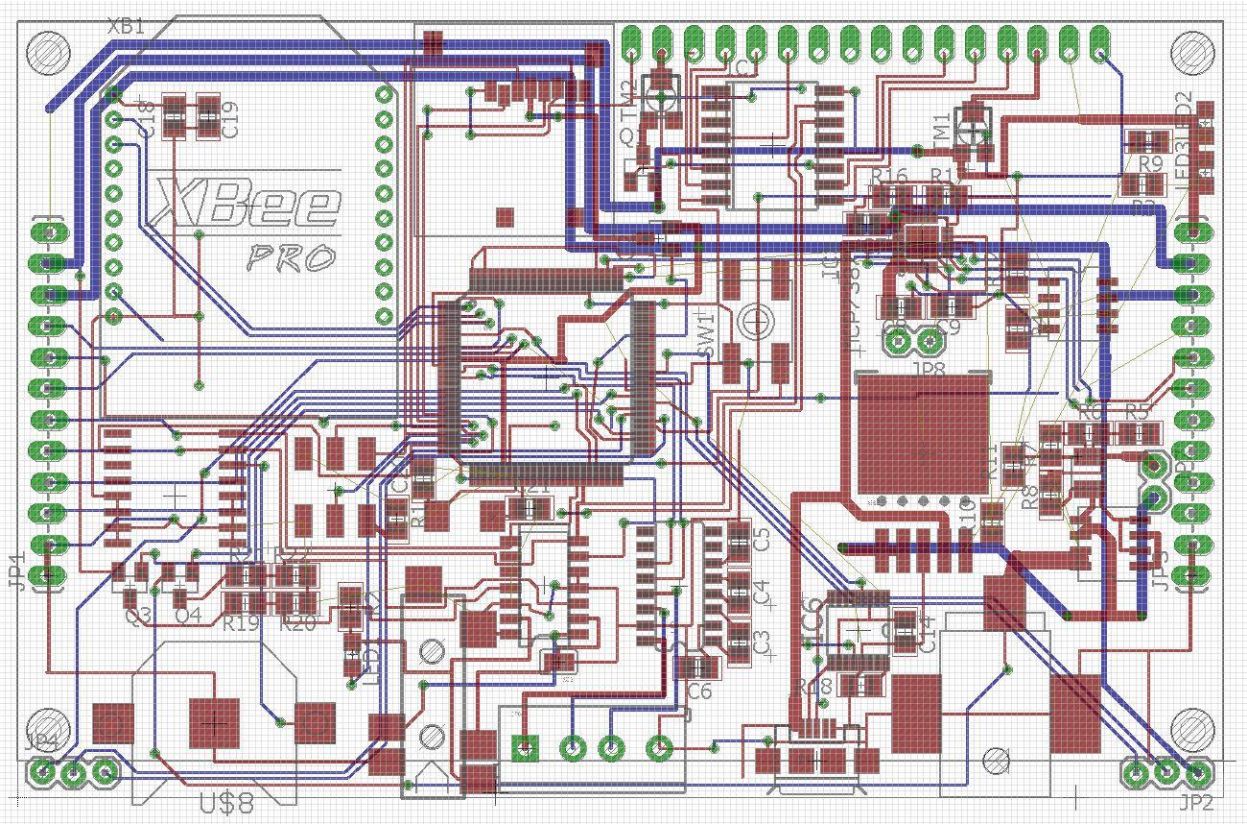
[10] "IEEE Code of Ethics," IEEE, 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html.

# Appendix A

## base station Schematics



## base station PCB Design



# Appendix B

## Requirements and Verification Table

As mentioned in Chapter 3, Table 5 in Appendix A contains our full Requirements and Verifications Table.

| Components | Requirements | Verification | Points (Total :100) |
|---|---|---|---|
| tree hugger Code | 1. Able to log data to the SD card in the tree hugger without stop logging | 1. The tree hugger device needs to be able to log data into the SD card for at least 24 hours without stopping. | 10 |
| tree hugger/Base station: Xbee | 1. Able to wirelessly transmit up to a range of 9m. And be able to receive data with a receiver<br>2. Transmit data up to 65 bytes per packet.<br>3. Data transmission rate is up to 20 bits/s. | 1. a) Measure 9 meters of distance using measuring tape and put two xbee modules that distance apart.<br>b) Check that one xbee module can sent packets to the other xbee.<br>2. a) Preload the SD Card with 65 bytes of data.<br>b) Use the xbee to send those data to a laptop that receives with a wireless adapter<br>c) Check that the received data is the same as preloaded data on the SD card.<br>3. a) Detect the latency of the data transmission<br>b) Divide 65 bytes by that to verify the transmission rate. | 30 |
| tree hugger: Li-ion battery | 1. Battery provides 3.3V ± .1V to the xbee and at least 300mA when fully charged.<br>2. Operation time of 1 year | 1. Using a multimeter, measure the voltage of across a 10ohm resistor while discharging 300mA current.<br>2. Discharge the battery by drawing 1000x power of the tree hugger device for half a day. | 15 |

| | | | |
|---|---|---|---|
| Base station: SD card | 1. Storage capability of 1GB.<br>2. Able to read and write to the SD card at 4MB/s | 1. Plug the SD card into a card reader then into a computer and check its storage size.<br>2. Use a USB benchmark software to test SD card's R/W speed. | 15 |
| Base station: power | 1. Standard 110V power supply which is then drew by the Voltage regulator. | 1. Use a multimeter to check the voltage going into the voltage regulator is at 110V. | 20 |
| Base station: Voltage Regulator (SPX 29302) | 1. Must be able to provide current from the power supply for the SM5100B to continuously run at 3.3V with maximum power consumption | 1. Use a multimeter to check that the voltage regulator draws Draw current at 3.3V from the regulator which is regulated to 300mA with a 10ohm resistor. | 10 |