Final Paper for Fall 2018 EE 396
2nd Generation Relay Node: Team Bumblebee

Authors: Xiao Jing Chen, Mizpah Mansanao, Sharmaine Javier

Date: 14 December 2018

Abstract: The Smart Campus Energy Lab's (SCEL) sensor node 'Bumblebee' wirelessly collects and relay meteorological data collected via a weatherbox Xbee device to the gateway in Holmes Hall 493. The current sensor nodes used in SCEL weather boxes have limited ranges in populated areas like the University of Hawai'i at Manoa campus. Therefore, the relay module 'Bumblebee' was developed to extend the range of the sensor node network. 'Bumblebee' utilizes an Xbee Pro S2B for wireless communication, an Atmega328P for its microprocessor, and a solar panel with a rechargeable battery for power. Along with these components a PCB, firmware, and housing needed to be developed to create a working relay communication node. Preliminary range testing with the module has shown that range can extend beyond 400ft and package send/ receive effectiveness can be affected by "movements" between the two communication nodes. More testing will be conducted to test the exact limitations of the module and how to make a more robust network.

Table of Contents

I. Introduction

The Smart Campus Energy Lab (SCEL) is one of the many research laboratories within the Center of Renewable Energy and Island Sustainability (REIS). The objective of SCEL is to develop technologies that will promote sustainability and renewable energy usage. The motivation of SCEL is the development of low-cost and reliable environmental sensor nodes to collect meteorological data such as temperature, humidity, and solar irradiance. Bumblebee's ambition was to create and design a relay to extend the range of current sensor node networks. These sensor nodes are to be placed on the rooftops of the University of Hawaii at Manoa.

Bumblebee does not involve any sensors. Therefore, the ideal location for Bumblebee weatherboxes would be in between distant sensors and the lab gateway. Bumblebee is based off of the third generation of sensor nodes which is Cranberry. Bumblebee has the simplified circuit with its Atmega328P, Xbee Pro S2B, solar panel and rechargeable battery of 3.6V. Bumblebee is interested in performing more range testing to improve weather collecting data in the nearest future and to also network with multiple sensor nodes.

II. Relay Node: Bumblebee Overview

Block Diagram

The block diagrams shown below are the overall design of both power system and communications of Bumblebee. Figure 1 below represents the power system.

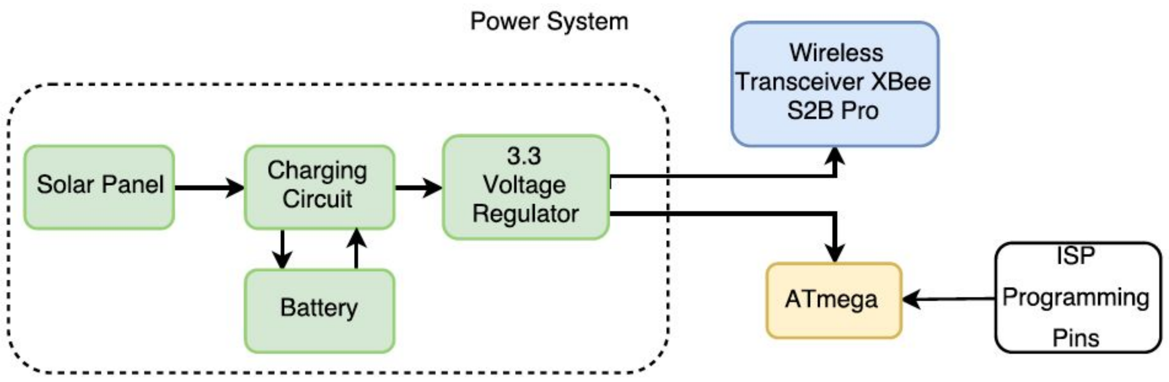


Figure 1: Power Block Diagram

The power block diagram describes the functionality of each of the hardware components that are connected. To help power the Bumblebee relay node, we incorporated a charging circuit and a solar panel into the board. Since Bumblebee does not have any sensors, a 5V booster converter is not necessary because the Atmega and Xbee can operate at 3.3V. Only one voltage regulator was used to supply enough current to the circuit because there was not much current.

Figure 2 represents the different sensor nodes will communicate wirelessly. The signal/communication block diagram illustrates the basic paths where the data will travel from the sensor node to the gateway computer. For instance, Cranberry's weatherbox will collect data from its sensors, construct a packet with that collected data and send the packet to the relay node. Next, Bumblebee receives that data and then forward it to the gateway Xbee which that data is sent to the lab gateway. Overall, the diagram illustrates that in order for the lab gateway to receive a packet, data has to go through multiple Bumblebee nodes.

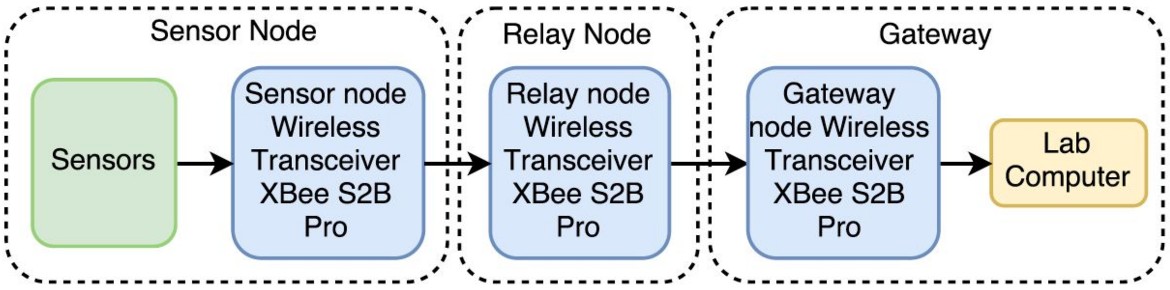
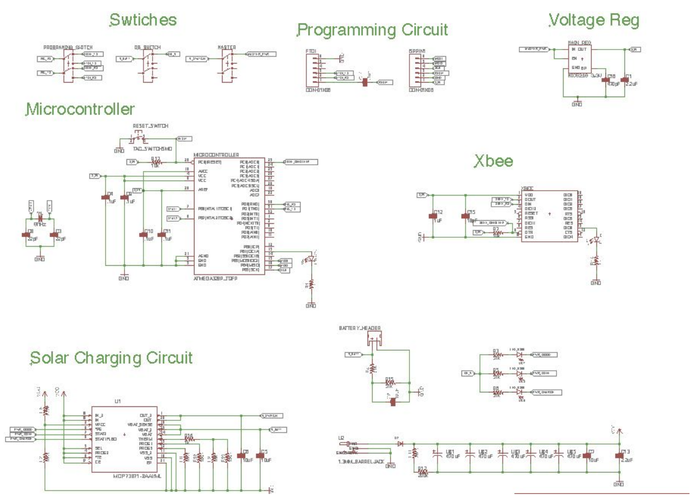
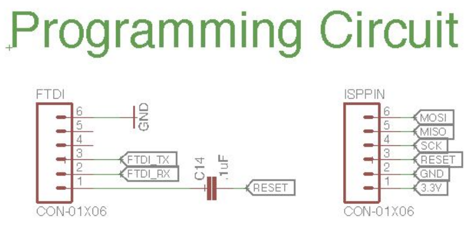

Sensor Node
Sensors
Sensor node Wireless Transceiver XBee S2B Pro
Relay Node
Relay node Wireless Transceiver XBee S2B Pro
Gateway
Gateway node Wireless Transceiver XBee S2B Pro
Lab Computer

Figure 2: Signal Block Diagram

Design: Schematic and PCB Layout

Referring to the previous team's meeting minutes, one of the main modifications that needed to be fixed for this semester is modifying the schematic and PCB layout to include ISP Programming pins and a capacitor in between the FTDI and Reset pin. Since all members of this semester's team are all new, we knew that there will be a lot of learning curve. Because of that to save us time, instead of making our own PCB layout, we decided to use the previous team's design and made the necessary adjustments. The main design for the schematic and PCB layout is based off the Cranberry Weatherbox. The previous team decided that the design for the board will consist of all surface mount components with the exceptions of the FTDI and ISP Programming pins.





(a) *(b)*

Figure 3: Schematic with ISP Programming Pins and Capacitor between FTDI and Reset

One of the main addition to the schematic from the previous semester was an Xbee sleep line, which allows the Xbee to remotely put into sleep mode. This implementation will save power consumption in the board because the Xbee will be in sleep mode when there is no data to be transmitted. However, since we run out of time this semester, we were not able to implement and test this addition to our board.

Another component is the programming switch. This is one of the very important parts of the board because it allows the connection of the RX and TX of the Atmega to be switched between the TX and RX of FTDI and Xbee. When programming the board, the RX and TX of the Atmega should be connected to the TX and RX of the FTDI, respectively. When the board is ready to receive and transmit packets, the programming switch should be flipped so that the RX and TX of the Atmega are connected to the TX and RX of the Xbee.

Just like the Cranberry Weatherbox, the Bumblebee board uses PPM 20 QFN IC microchip as a charger. This component was implemented along with other passive parts using Cranberry’s schematic. The charging chip will help keep the relay node self-sustaining. The solar panel will collect energy that will then be used to charge the battery during the day. Three LEDs are also connected to the charging chip to tell the status of the charger. The three statuses are on, charging and done charging. A debug switch is used to turn the LEDs on and off to control the power consumption.

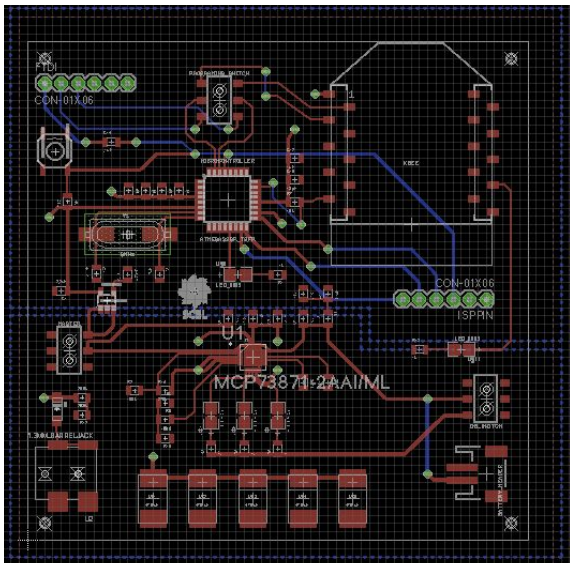


Figure 4: PCB Layout

The current PCB design, shown in Figure 4 has 3 inches by 3 inches dimension. The top layer of the board is ground, while the bottom layer is split into 3.3V and input voltage from the charging chip. All of the components are in the top layer. The top half of the top layer of the board consists of components that require 3.3V. These components include the microprocessor Atmega328P, Xbee Pro S2B, FTDI, and ISP Programming pins. On the other hand, the bottom half of the top layer of the board is the charging chip. The components are placed strategically to met certain criteria. For instance, all capacitors are closed to a respective component to clean the signals going through. The antenna of the Xbee is also placed where there are no metals nearby. Metals could possibly interfere with the communication signal and can cause data packet errors and loss packets.

The current layout of the PCB will produce a bigger housing because of how the wiring for the battery and solar panel are oriented. For next semester, we are planning to redo the whole layout of our PCB so that all wires are going in the same direction as the Xbee. This way it will be easier to come up with the design, and at the same time minimize the size of the housing.

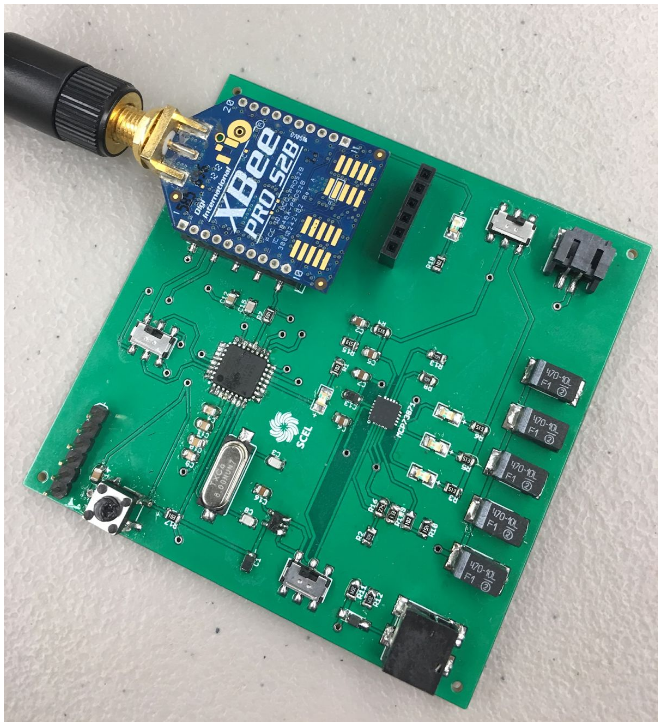


Figure 5: Populated PCB

Bare Arduino Board

At the beginning of the semester, we decided to build the bare Arduino board so that we can understand the functionality of each component and the relay node itself. For our bare Arduino, we stick with the 16MHz external clock instead of the 8MHz as it shows in the schematic. This is to avoid bootloading our Atmega168 to run on 3.3V since this is a different microprocessor than the one in the schematic. We've also used a breakout board for the solar charging chip and a step down 3.3 voltage regulator. There are two debug LEDs to show whether the board is getting power and if the Xbee is connected.

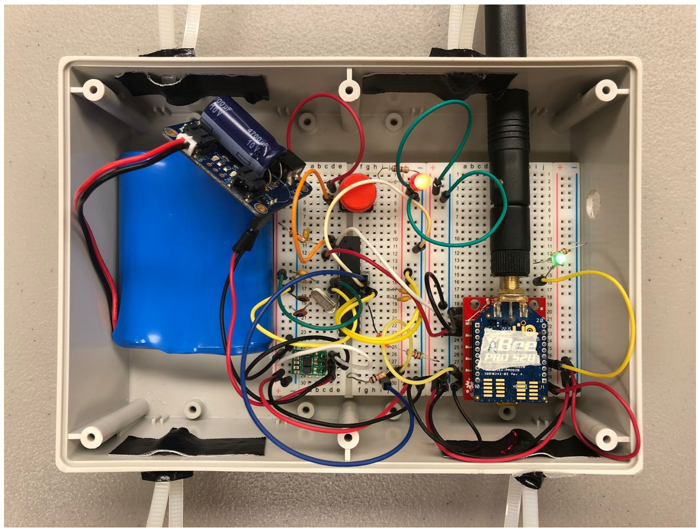


Figure 6: Bumblebee Bare Arduino Board Ready to be Deployed

We programmed our board using the relay code from previous semester team. Since the microprocessors are different, we had to make small adjustments on the code. The PCB has the Atmega328P microprocessor, while our bare Arduino board has the Atmega168. We had to change the pin numbers on the code so that it corresponds to the pin numbers of the microprocessor in our bare Arduino board.

After completing our bare Arduino board, we conducted packet relay testing using the XCTU to test whether our code and Xbee connections work. After making sure the Xbees can communicate with each other, we also conducted range (field) tests. We did not have much time to work on our PCB and complete it to be ready for deployment. Because of that, for the first deployment, we used our bare Arduino board to deploy a relay node. Since we need to put our board in a house, the housing team made a temporary housing for our board as shown in Figure 5 above. On the deployment day, our Bumblebee board is able to receive packets from Cranberry weatherbox and transmit it to the gateway. However, when it is already at the roof of Holmes Hall building, there were no packets being transmitted to the gateway. This problem is addressed in the problems encountered and solutions section of this report.

Packet Relay Testing

The XCTU and Arduino IDE programs were used to test the abilities of the Xbee to send and relay packets. The first step is to configure the Xbees into either AT or API mode using the XCTU program. For this project, we configured three Xbees: one as an API coordinator and two as an API router. The Xbee configured as a coordinator acts a the gateway or receiving-end. The two routers are used for the sending-end or sensor node and for the relay node. Xbees that are configured as routers can transmit packets to other Xbees, while coordinators can only transmit to itself. Xbees are also identified through their address, which can be divided into two parts — serial high and serial low.

In able for the Xbees to communicate with each other, their PAN IDs have to be the same. This can be changed in the XCTU console. When the destination addresses are not specified, the sending Xbee can transmit to any Xbee with the same PAN ID. To specify which Xbee to transmit to, the destination address of the sending Xbee should be the address of the receiving Xbee. For the Xbee that is transmitting packets, make sure to change the API mode from 1(default) to 2. If this is not changed, the Xbee can still be recognized by other Xbees, however, it won't be able to transmit packets.

After configuring the Xbees, attach one Xbee to an Arduino board and program it using the Arduino IDE program. A program to send data packets in certain time interval can be written using Andrew Rapp’s Xbee library for Arduino IDE that can found online. When programming the Xbee in the Arduino board, make sure to change the switch into DLINE on the Xbee shield and switch it back after. If not, a sync error could occur while trying to upload the code onto the board. To see whether the sending Xbee is transmitting data or there is a communication between the Xbees, go back to the XCTU console monitor and close the port. Once the port is closed, there should be packets received if all the connections are correct.

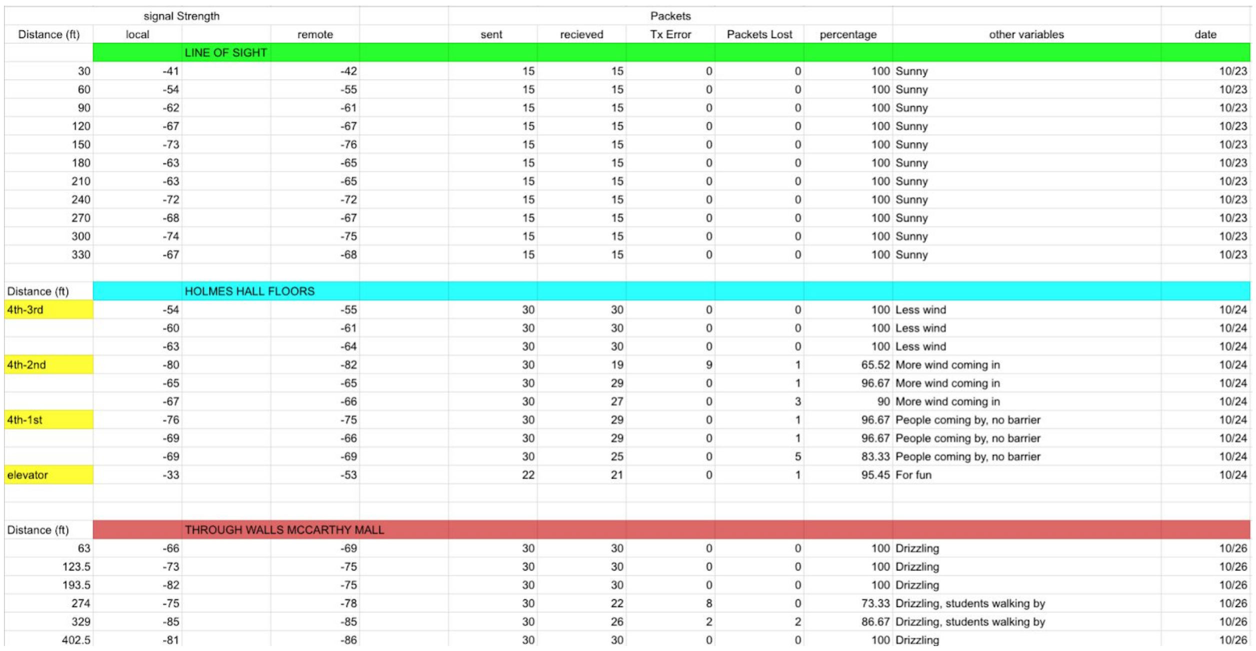
Range (Field) Testing

To determine the performance of the Xbee, we performed range testings in three different ways. The purpose of range testing is to take into account as many variables as possible and to gather data on how far the Xbee can implement certain distances and obstacles. To conduct this testing, we used XCTU since it has a built-in range test software program. The data values that were included were local strength, remote strength, packets sent and received, TX errors, packets lost, and percentage of packets received. The results of all three range testings are shown in Figure 7.

The first range test conducted was the straight line-of-sight. This was conducted in Holmes Hall 4th floor. One Xbee is stationed at one end of the building, while the other Xbee moves away in increments of 30 feet, keeping it line-of-sight. No packets were lost in this time of testing. Ideally, the signal strengths should have been around -32 dBm, no TX errors, and 100% packets received. For the line-of-sight test, the results show that at 330 ft the signal strengths were around -67 dBm and -68 dBm with 100% packets received.

The second range testing done was not line-of-sight which was also conducted in Holmes Hall floors from 4th to 1st floor. The same set up is performed as the straight line-of-sight, but the other Xbee moved to different floors. As the distance go farther for the Xbees, the two increased in number of TX errors and packets lost. In addition, the signal strengths were significantly lower due to variables like the weather, walls and people passing by.

The third range testing conducted was the through hall at McCarthy Mall at campus. The same set up was done with this testing just like the straight line-of-sight. Based on Figure 7, it was a gloomy, windy weather and more students were passing by—this being the reason why more TX errors showed up in the overall results. There are still more useful variables to test to determine the performance of the Xbees. Since we are using Xbee S2B, we plan to conduct more range testings with Xbee S2C.



	signal Strength				Packets				
Distance (ft)	local	remote	sent	recieved	Tx Error	Packets Lost	percentage	other variables	date
	LINE OF SIGHT								
30	-41	-42	15	15	0	0	100	Sunny	10/23
60	-54	-55	15	15	0	0	100	Sunny	10/23
90	-62	-61	15	15	0	0	100	Sunny	10/23
120	-67	-67	15	15	0	0	100	Sunny	10/23
150	-73	-76	15	15	0	0	100	Sunny	10/23
180	-63	-65	15	15	0	0	100	Sunny	10/23
210	-63	-65	15	15	0	0	100	Sunny	10/23
240	-72	-72	15	15	0	0	100	Sunny	10/23
270	-68	-67	15	15	0	0	100	Sunny	10/23
300	-74	-75	15	15	0	0	100	Sunny	10/23
330	-67	-68	15	15	0	0	100	Sunny	10/23
Distance (ft)	HOLMES HALL FLOORS								
4th-3rd	-54	-55	30	30	0	0	100	Less wind	10/24
	-60	-61	30	30	0	0	100	Less wind	10/24
	-63	-64	30	30	0	0	100	Less wind	10/24
4th-2nd	-80	-82	30	19	9	1	65.52	More wind coming in	10/24
	-65	-65	30	29	0	1	96.67	More wind coming in	10/24
	-67	-66	30	27	0	3	90	More wind coming in	10/24
4th-1st	-76	-75	30	29	0	1	96.67	People coming by, no barrier	10/24
	-69	-66	30	29	0	1	96.67	People coming by, no barrier	10/24
	-69	-69	30	25	0	5	83.33	People coming by, no barrier	10/24
elevator	-33	-53	22	21	0	1	95.45	For fun	10/24
Distance (ft)	THROUGH WALLS MCCARTHY MALL								
63	-66	-69	30	30	0	0	100	Drizzling	10/26
123.5	-73	-75	30	30	0	0	100	Drizzling	10/26
193.5	-82	-75	30	30	0	0	100	Drizzling	10/26
274	-75	-78	30	22	8	0	73.33	Drizzling, students walking by	10/26
329	-85	-85	30	26	2	2	86.67	Drizzling, students walking by	10/26
402.5	-81	-86	30	30	0	0	100	Drizzling	10/26

Figure 7: Range Testing Results

III. Problems and Solutions

The initial problem was getting ourselves familiarized with the project 'Bumblebee' as we were all new members to this project. We were unsure of which direction to take when we first began the lab hours. Luckily for us, the past bumblebee team has done excellent documentation on their purpose, process, and the problems they ran into along with the solutions.

After we familiarized ourselves and began the bare Arduino building process, we encountered several problems when uploading the code into our board. One of the reasons for that error was caused by the 8MHz clock. Since our microprocessor is not configured to run on 8MHz, we changed it back to the 16MHz clock and we were able to upload the necessary code as well as uploading the relay and sending codes into the two Xbees.

After configuring the Xbees, we tested to see if the coordinator was receiving data, but it wasn't receiving anything. We asked from the mentors, Andrew, and Kenneth for help. However, they were also struggling. Furthermore, after three weeks of trying to figure out the cause of this problem, we, with the help from Kenny and past team members, finally reached to a resolution that the main problem was the API mode setting on the XCTU. After changing the API mode from 1 (which was the default) to 2, we tested our board and our Xbees started to send and receive data on both ends.

With our completed Bumblebee board that can successfully relay Cranberry packets to SCEL's gateway, we deployed at the end of October. To our surprise, gateway did not receive any packets from bumblebee. We later found out that the wiggle room on our housing for the antenna caused our Xbee to be slightly lifted and loosened, therefore causing a bad connection to the board. We decided to deploy next with our PCB instead of attempting to deploy again with our Bumblebee board.

The following week, we populated our PCB board. However, we soon discovered that we were not able to receive a package. We discovered that certain components were not soldered on accurately and therefore not connected. We immediately resoldered those components. Unfortunately, our PCB still wasn't able to receive a package. It could be a software or hardware problem that we are working on figuring out.

IV. Future Work

At this moment, we have a populated PCB with unknown hardware and/or software problem. In the following semester, we plan to debug our PCB as we are unable to receive packets as of now. Unfortunately, we did not have a working PCB this semester, we did not have a chance to deploy Bumblebee. However, next semester we hope to be able to deploy. To achieve that goal, we also need to communicate with Team Strawberry to create a housing for our PCB and components. The current design of our PCB will produce a bigger housing compared to the other teams. It is because of the orientation of the wirings, which is addressed above. For next semester, we plan to modify or create a new PCB design that will take into account the possible way to minimize the size of the housing.

Earlier this semester, we found out that XBee S2B Pro, the wireless communication component we are currently using, has been discontinued. We will do research on the new wireless communication component XBee Pro S2C and integrate it to bumblebee so all the components are easily accessible and updated.

For this semester, we worked with Team Cranberry during the deployment. However, because of the error that occurred, we were unsuccessful to deploy a relay node. During the communication testing of the Xbees in our Bumblebee board and Cranberry board, we noticed that Bumblebee is not able to handle more than one packets at the same time. Based on what Andrew suggested this should be a small modification on the relay code. In addition, we hope to work on the ability to relay information for multiple sensor nodes instead of just one so that Bumblebee can relay collected meteorological data packets of Cranberry, Guava, and Apple.

V. Conclusion

At the beginning of the Fall 2018 semester, there was a big learning curve to catch up with the progress of past members since we are the new members of Bumblebee. We were able to slowly adjust to the new materials and concepts with this project after referring to previous team's work such as reports and meeting minutes as well as their schematics and PCB layout. The first approach to improve Bumblebee is to do some modification on the PCB layout just as the past member suggested which is to add the ISP programming pins and putting capacitor between the reset and FTDI. Moreover, building a bare Arduino was necessary to do to have a more understanding of the functionality of Bumblebee and to see if the schematic is accurate. With this process, we ran into multiple problems, but we were able to resolve them with the help of the leadership people and exchange emails from past members.

The downcast of this semester's Bumblebee is the undeployment. Due to the late arrival of our board, we were not able to meet the deadline and also encountered problems that we have yet to resolve. Nonetheless, we hope to achieve of deploying two Bumblebee weatherboxes on the roof of Holmes Hall. To reach this goal, we hope to debug our PCB and deploy with a customized housing, integrate XBee S2C Pro to our code and layout, potentially work with team Guava and Apple to relay their packets, and lastly work on the ability to relay information for multiple sensor nodes instead of just one as well as conducting more range testings in different conditions.

Overall, this project has been a great learning experience. We had a chance to gain more knowledge and enhanced our skills in XBee communications and networking, PCB design, and soldering. This project allowed us to visualize what we learned in class and strengthen our knowledge of it. Because we are the new members of Bumblebee, we encountered numerous obstacles in knowing the new materials and concepts on how Bumblebee really works. However, we were able to gradually adjust to the new environment and work through the process. \

References

1. Gammon, Nick "Atmega Board Programmer" Mar 29, https://github.com/nickgammon/arduino_sketches/tree/master/Atmega_Board_Programmer
2. Rapp, Andrew "Arduino library for communicating with Xbee radios in API mode" Dec. 2016, https://github.com/andrewrapp/Xbee-arduino
3. Xbee / Xbee-PRO ZigBee RF nodes" User Guide, Digi International Inc., April 2008, revised July 2016, http://www.digi.com/resources/documentation/digidocs/PDFs /90000976.pdf.