Final Paper for Spring 2019 EE 496
2nd Generation Relay Node: Team Bumblebee

Authors: Xiao Jing Chen, Mizpah Mansanao, Sharmaine Javier

Date: 12 May 2019

Abstract: The Smart Campus Energy Lab's (SCEL) relay node 'Bumblebee' wirelessly receive and transmit 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 S2C 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 node 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 S2C, 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 Diagrams

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

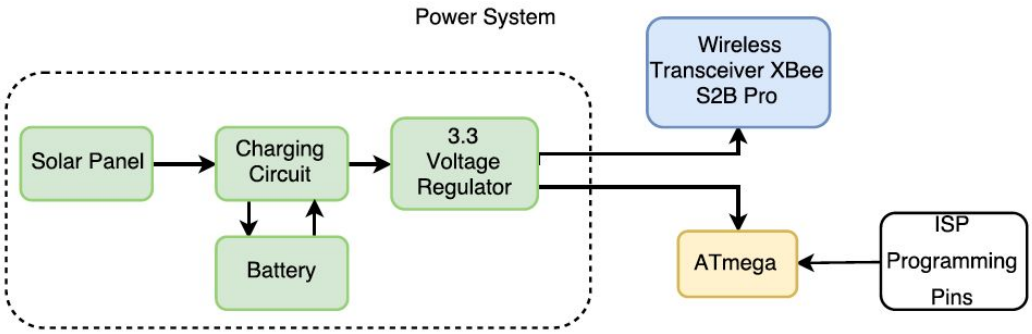

Figure 1: Power Block Diagram of Old Design

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. As shown in the old block diagram, a 5V boost converter was 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.

However, since there was an issue with how the code works, we incorporated a 5V regulator so that it properly functions with the 16 MHz clock to prevent some problems from happening. Referring to the data sheet, it does not say it should not work, but it is not *qualified* for operation at 16 MHz below some voltage higher. This means that a considered chip may have a potential to fail and considered it low quality.These failures may be obvious or understated.

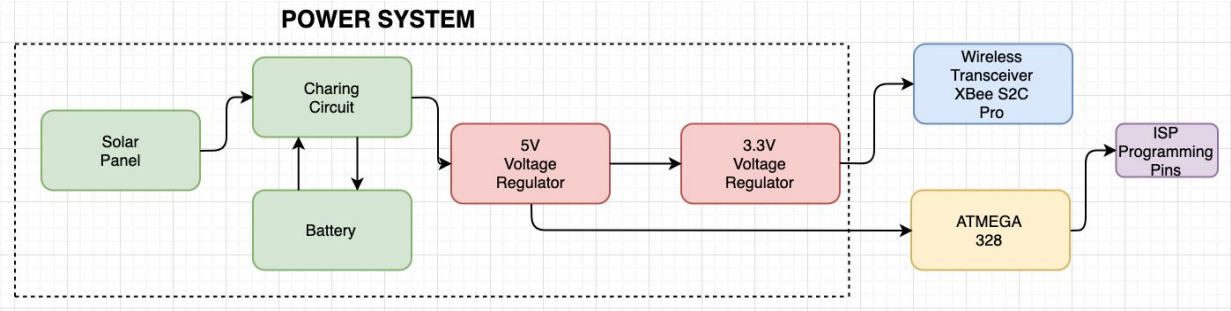

Figure 2: Power Block Diagram of New Design

The signal/communication block diagram shown in Figure 3 illustrates the basic paths where the data will travel from the sensor node to the lab gateway computer. For example,

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.

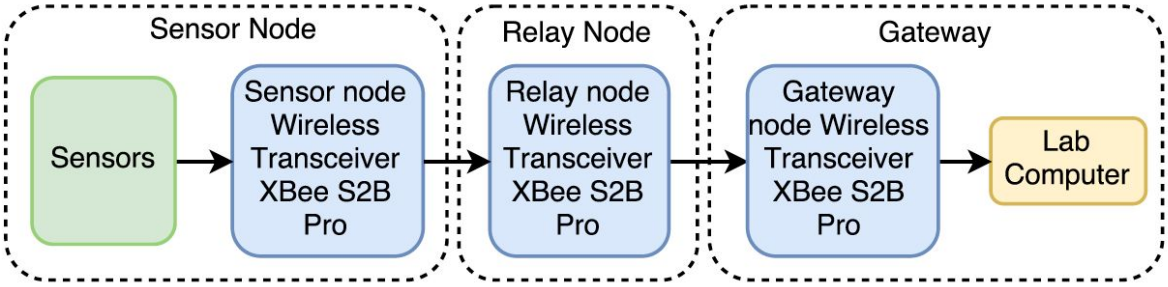

Figure 3: Signal Block Diagram

Design: Schematic and PCB Layout

Last semester, the main addition to the schematic and PCB design were the ISP Programming pins and a capacitor in between the FTDI and reset button. The PCB layout from previous team was modified to include these additions—keeping the same size and design with minimal changes. However, for this semester, a new board was fabricated with two different versions. The changes made were to minimize the source of possible errors in the printed circuit board (PCB) design and to simplify its complexity in terms of soldering.

The schematic and PCB layout for the first version is shown in Figure 4. This version eliminates the programming switch and the solar charging circuit built into the previous board. While trying to debug the fully populated PCB from the previous semester, we found out that there might have been voltage leak caused by the programming switch between the FTDI and the XBee RX and TX lines. This may have caused the untimely TX response for the XBee. For that

being the reason, we decided to remove the programming switch and used two digital pins of Atmega328P. By doing so, we were able to connect the TX and RX lines of the FTDI and XBee to the Atmega without the switch.

After populating another PCB from the previous semester, there was a problem in the output voltage from the charging circuit. This may have been caused by the badly soldered microchip. Since the microchip is a QFN package, it was difficult to test whether the component was soldered on properly. After several tries of resoldering the microchip, there was still no improvement in the power going to the board. At this point, we decided to use a breakout board for the solar charging circuit.

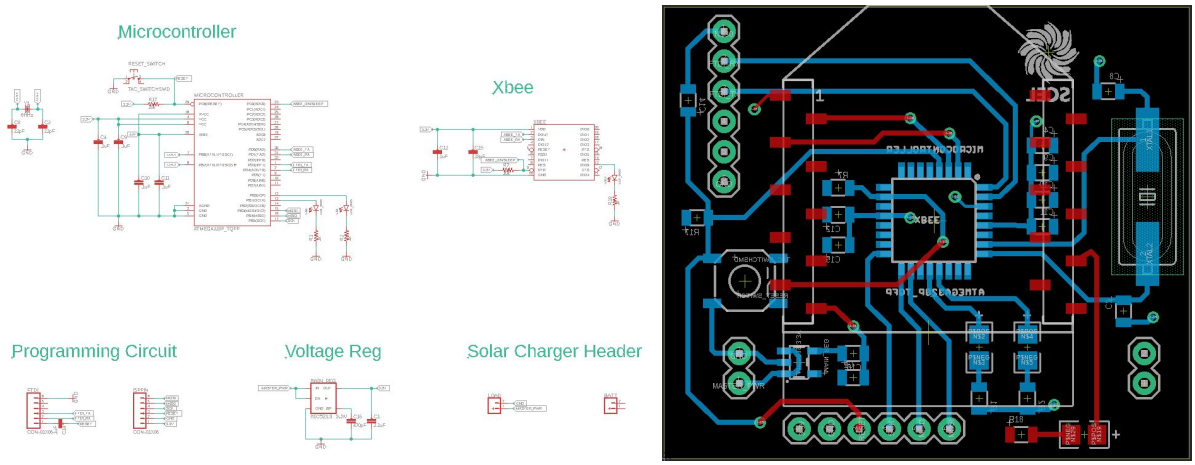

Figure 4: Schematic and PCB Layout for Version 1 (3.3V and 8MHz)

For this version, the board is still running at 3.3V with 8MHz clock as shown in Figure 4. After some revisions in the design, the size of the new board is reduced to almost a quarter of the old PCB. The new design has dimension of 1.5 inches by 1.75 inches. The top layer of the board is ground and it consists of the XBee Pro S2C, headers for the solar charging circuit breakout board, and the LED to indicates the XBee is connected. The bottom layer is 3.3V, which consists of the rest of the components. These includes the Atmega328P, 8MHz clock, and headers for FTDI and ISP Programming pins.

The second version shown in Figure 5 is very similar to the design of the first version. The main difference for this version is the 5V voltage regulator added to have the Atmega run on 5V with 16 MHz clock. While debugging the first version, we compared it to our Bare Bumblebee and noticed that we have been using a 16 MHz on our bare board. We tried changing the clock of our first version to 16MHz and it works properly as a sending node. However, it does not function properly when we programmed it using our relay code. This may have been because the board was being powered with 3.3V, which is not compatible with a 16MHz clock.

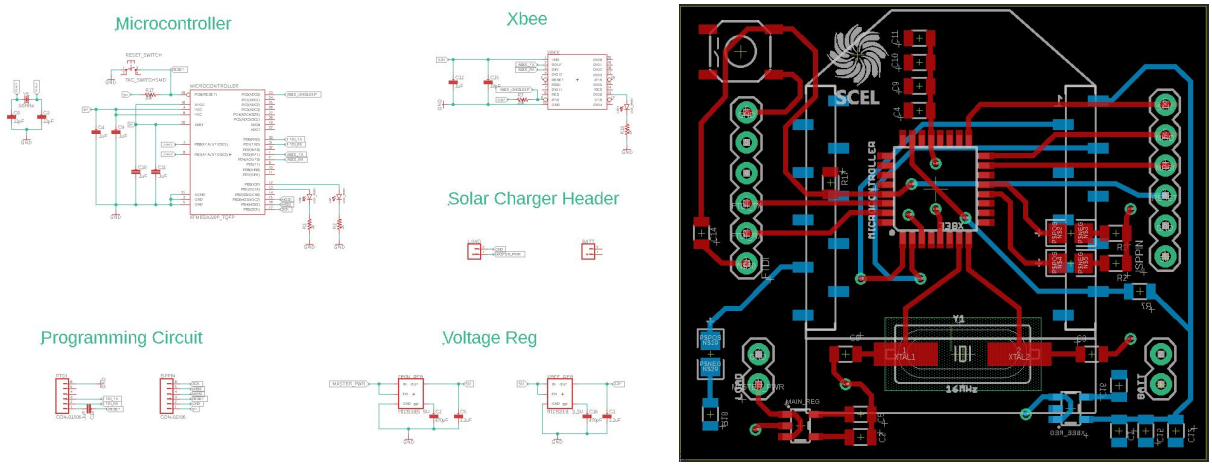

Figure 5: Schematic and PCB Layout for Version 2 (5V and 16 MHz)

The size and most of the components of this version are the same as the first version. However, the placement of components were slightly different. For instance, the top layer is now 5V consisting of the Atmega328P, 16MHz clock, 5V voltage regulator and the headers for the FTDI and ISP Programming pins. On the other hand, the bottom layer is ground with the 3.3V voltage regulator, XBee Pro S2C, and headers solar charging circuit breakout board. Another significant difference is the connection of the RX and TX lines of the FTDI and XBee. In the first version, the FTDI TX and RX are the ones connected to the digital pins. However, it should have been the XBee TX and RX lines connected to the digital pins. For this being the reason, the

code was implemented to have the digital pins as another RX and TX lines for the XBee. Due to confusion, we switched the connection pins for the TX and RX of the XBee and FTDI. This changes are taken into account in our relay code.

In addition, some tips of improving the connections of each component was the design were documented. For instance, we made sure that there were no ‘Y’ or ‘T’ splits in our traces. We also made sure that each component has equal trace lengths from the power to acquire even distribution. Another important improvement to point out is to make sure that the traces of the clock are symmetric. Figure 6 below shows the side by side comparison of fully populated printed circuit boards from last semester and this semester.

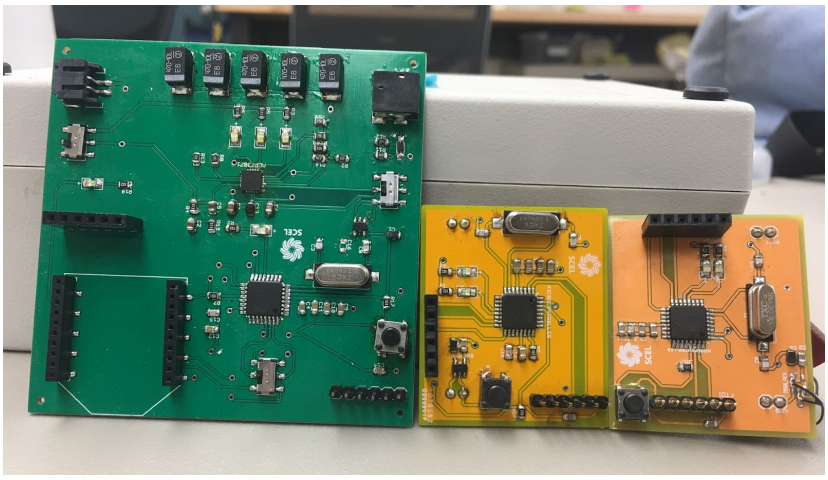

Figure 6: Side by Side comparison of PCB designs from previous and current semester

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 API router. The XBee configured as a coordinator acts as 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 order 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

Since the bumblebee relay node is updated to use the XBee Pro S2C instead of the XBee Pro S2B from last semester, we determined the performance of the XBee Pro S2C by performing range testings on our bare Bumblebee board in four 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 such as obstacles and weather. To conduct this testing, we used a built-in range test software program in XCTU. The data values that were included were local strength, remote strength, packets sent and received, TX errors, packets lost and percentage of packets received.

Distance (ft)	local	remote	sent	recieved	Tx Error	Packets Lost	percentage	other variables	date
30	-65	-63	15	15	0	0	100	Sunny, Windy	3/8
	-67	-64	15	15	0	0	100	Sunny/windy	
	-65	-65	15	15	0	0	100	Sunny/windy	
60	-75	-71	15	15	0	0	100	Sunny, Windy	
	-82	-79	15	15	0	0	100	Sunny/windy	
	-75	-71	15	15	0	0	100	Sunny/windy	
90	-80	-78	15	15	0	0	100	Sunny, Windy	
	-80	-77	15	15	0	0	100	Sunny/windy	
	-82	-78	15	15	0	0	100	Sunny/windy	
120	-82	-79	15	15	0	0	100	Sunny, Windy	
	-83	-86	15	15	0	0	100	Sunny/windy	
	-86	-87	15	14	0	1	93.33	Sunny/windy	
150	-92	-92	15	15	0	0	100	Sunny, Windy	
	-9	-92	15	15	0	0	100	Sunny/windy	
	-9	-90	15	15	0	0	100	Sunny/windy	
180	-10	-91	16	14	1	0	93.33	Sunny, Windy	
	-89	-88	15	15	0	0	100	Sunny/windy	
	-89	-86	15	15	0	0	100	Sunny/windy	
210	-10	-93	15	12	2	1	80	Sunny, Windy	
	-10	-86	15	15	0	0	100	Sunny/windy	
	-10	-90	16	13	2	0	86.67	Sunny/windy	
240	-95	-93	16	7	8	0	46.67	Sunny, Windy	
	-94	-94	16	4	11	0	26.67	Sunny/windy	
(a)	0	0	15	0	15	0	0	(DIDNT RECEIVE ANY PACKETS	
(b)	-9	-92	16	7	8	0	46.67	Sunny/windy	
270	-93	-92	15	13	2	0	86.67	Sunny, Windy	
270	-94	-93	16	14	1	0	93.33	Sunny/windy	
270	-9	-94	16	8	6	1	53.33	Sunny/windy	
300	-9	-89	15	15	0	0	100	Sunny, Windy	
	-9	-88	15	15	0	0	100	Sunny/windy	
	-9	-89	15	15	0	0	100	Sunny/windy	
330	-9	-93	15	15	0	0	100	Sunny, Windy	
	-9	-89	15	15	0	0	100	Sunny/windy	
	-91	-90	15	11	4	0	73.33	Sunny/windy	
360	-12	-89	16	5	10	0	33.33	Sunny, Windy	
	-12	-91	15	15	0	0	100	Sunny/windy	
	-12	-89	15	15	0	0	100	Sunny/windy	
390	-9	-91	15	15	0	0	100	Sunny, Windy	
	-10	-87	15	15	0	0	100	Sunny/windy	
	-10	-91	15	15	0	0	100	Sunny/windy	

Figure 7: Line of Sight Range Testing Results

The first range test conducted was the line-of-sight testing completed in Holmes Hall 4th floor where there were no obstacles between the local signal and remote relay module. We placed the local signal on one end of the building and moved the Bumblebee relay module away in increments of 30 feet until we hit the other end of the building. The weather condition of the first range test was sunny and very windy. We received most of the packets sent in the beginning

and as the remote relay module moved further away, the signal got weaker and we began to see more variants of packet lost. A graph representation of our results can be seen in blue in Figure 10. As compared to the range testing results obtained from last semester, more packets were lost as the distance between remote and local signal furthered. We believe this may be largely due to the weather condition. In previous testings, it was in a sunny and calm environment whereas this semester's range test was in a much more windy environment.

Distance (ft)	local	remote	sent	recieved	Tx Error	Packets Lost	percentage	other variables
4th-3rd	-58	-59	15	15	0	0	100	WINDY, placed under bench
	-63	-63	15	15	0	0	100	WINDY, placed under bench
	-62	-63	15	15	0	0	100	WINDY, placed under bench
4th-2nd	-43	-90	15	13	2	0	86.7	not too windy compared to 3rd floor, under bench
	-42	-87	15	15	0	0	100	not too windy compared to 3rd floor, under bench
	-87	-87	16	12	3	0	80	not too windy compared to 3rd floor, under bench
4th-1st	-78	-86	15	15	0	0	100	less wind, less obstacles
	-80	-80	15	15	0	0	100	less wind, less obstacles
	-81	-78	15	15	0	0	100	less wind, less obstacles

Figure 8: Not Line of Sight Range Testing Through Floors Result

The second range testing done was a not line-of-sight testing conducted in Holmes Hall floors. For this test, we set the local signal on Holmes Hall 4th floor and varied the remote relay module to 3rd, 2nd, and 1st floor. We received all of our packets on the 3rd and 1st floor with no TX errors, however, we had an average of 2 TX errors on the 2nd floor. We believed that from 4th to 3rd floor, the distance and obstacle between the signals were fairly small therefore we received all the packets. Although the distance and obstacles were more prominent from 4th to 1st floor, all of the packets were received because 1st floor is a much more open area in comparison to 2nd floor. On the 2nd floor, the combination of the benches and floors may have caused us to have more TX erros. A graph representation of our results can be seen in gray in Figure 10.

Distance (ft)	local	remote	sent	recieved	Tx Error	Packets Lost	percentage	other variables
90								
Trial 1	-81	-79	15	15	0	0	100	Sunny, little wind
Trial 2	-82	-79	15	15	0	0	100	Sunny, little wind
Trial 3	-81	-79	15	15	0	0	100	Sunny, little wind
175								
Trial 1	-72	-69	15	15	0	0	100	Sunny, little wind
Trial 2	-91	-89	15	15	0	0	100	Sunny, little wind
Trial 3	-91	-88	15	13	0	2	86.67	Sunny, little wind
244								
Trial 1	-91	-90	15	15	0	0	100	Sunny, little wind
Trial 2	-90	-89	15	15	0	0	100	Sunny, little wind
Trial 3	-91	-90	15	15	0	0	100	Sunny, little wind
324								
Trial 1	-93	-93	15	7	8	0	47.67	Students in front 1st tree
Trial 2	-94	-93	15	6	9	0	40	Crowd people walking/passing by
Trial 3	-94	-110	15	3	12	0	20	Crowd people walking/passing by
403								
Trial 1	-6	-94	15	3	11	1	20	
Trial 2	-6	-110	15	1	14	0	6.67	
Trial 3	0	0	15	0	15	0	0	
477								
Trial 1	0	0	15	0	15	0	0	
Trial 2	0	0	15	0	15	0	0	
Trial 3	0	0	15	0	15	0	0	

Figure 9: Not Line of Sight Range Testing Through Wall McCarthy Mall Result

The third range test done was a not line-of-sight testing through the trees in McCarthy Mall. The set up of this test was similar to the straight line-of-sight testing. Here we had the local signal behind the first tree and varied the remote relay module behind each tree along McCarthy Mall. As shown in Figure 9, almost all packets were received and sent in the beginning. As the distance and number of obstacles between the local and remote signal increased, we started to lose packets gradually until all packets was lost at 477 feet. A graph representation of our results can be seen in orange in Figure 9.

Figure 10: Range Testing Results

The fourth range test done was a not line of sight testing through walls in Holmes Hall. The purpose of this particular range test is to see the effective range of the XBee Pro S2C through wider obstacles as well as in an enclosed environment. From inside HH492 (SCEL break room) to inside HH493 (SCEL main room), there was one wall in between and we received all of the packets sent. From inside HH 492 to outside HH 491, there was two walls in between and we received 93% of the packets sent. From outside HH493 and outside HH486 (opposite room to SCEL), there was three walls in between and we received none of the packets.

Now that we tested with variables such as weather, distances, obstacles, walls, and combination of the mentioned, we gained an idea of the effective range of XBee. However, there are still other useful variables to test to determine the performance of the XBee. In the following semester, we plan to repeat and conduct more new range testings with XBee Pro S2C with more trials and set of tests for each location.

III. Problems and Solutions

Last semester, we left off with a populated PCB that was not able to receive a packet. This may be a software or hardware problem that we were not able to figure out. We began the semester debugging the PCB from the previous semester. We worked with Kenneth on the problem and was informed that it was a possibility that the three switches on the PCB may have caused leaked voltage into the XBee as mentioned above. In addition, we worked with Firmware to check the functionality of the debug LEDs and discovered that the code doesn't seem to go through the loop.

After accidentally removing the XBee headers and ripped out pads on our PCB, we immediately populated a new PCB. With the new PCB, we noticed that the debug LEDs would

not light up when the battery is plugged in. We tested the voltages going through each components in our board and found that there is not enough voltage going to the voltage regulator. In the end we came to the conclusion that the solar chip was either not soldered on properly or not working properly. Following that, we populated another PCB to see perhaps it was a problem with the first PCB. The same problem persisted.

Due to the many problems of the 1st PCB, we decided to fabricate a new PCB design. On the new PCB design, we replaced the solar charging chip with a breakout board to minimize soldering issues, removed switches to prevent voltage leakage, added in more debug LEDs and testing vias, as well as, changing wiring orientations.

While waiting for the PCB to be manufactured and delivered, we performed range testing with XBee Pro S2C and our bare arduino board. Our bare board suddenly shut off. After changing the battery and checked the output voltage from the regulator, the regulator was not functioning properly and outputted 5V. Following that, we decided to temporally go without a regulator since our battery was only outputting 3.4V. During this time, we have not noticed that we were using a 16MHz clock while powering the board with around 3.3V. Since the Atmega used in the Bare Bumblebee is not bootloaded to work with 8MHz, we kept the clock to 16MHz. Because of that, we did not took into account the inaccuracy it may have caused in our range testing data.

After soldering the 2nd Bumblebee design, we noticed that the PCB is neither sending nor receiving packets. After changing the clock from 16MHz to 8MHz, the PCB was able to function as sending node and was able send packets to the coordinator. However, it does not function properly as relay node. Our debug LED for receiving packet light up once indicating that it was able to receive from the sending node. However, the sending LED did not light up and

no packets were received at the coordinator. We consulted with firmware and learned that 8MHz clock runs more ideally with 3.3V while 16MHz clock runs more ideally with 5V. Originally, we wanted to keep the 8MHz clock on the PCB, however we remembered we had trouble bootloading with 8MHz clock in the past. To resolve this problem, we fabricated a 3rd PCB design to have a 16MHz clock running on 5V and the XBee on 3.3V.

We received the 3rd PCB design very late into the semester and did not have enough time to debug and work on it as much as we'd like to. The initial problem of the 3rd PCB design was that the 5V regulator was not a step up regulator and therefore the clock was not receiving enough power. In the future, we need to debug the 3rd PCB design as well as the relay code to work on 3.3V with 8MHz.

IV. Future Work

At this moment, we have three versions of populated PCB design. Each version of our designs were improved in a way to minimize errors as seen in the previous versions. For the coming semesters, we plan to debug our second board by designing a bare Bumblebee with Atmega328P bootloaded to run at 3.3V with 8MHz clock. While researching for ways to bootload an Atmega328P, we found out that there were available Atmega that are already bootloaded. We plan on purchasing some components to use for our bare Bumblebee design. Using this design, we will try to debug our board to make a way that will allow our relay code to work at 8MHz.

The third PCB design currently acquires a 5V regulator that does not step up they input voltage which means that it is not supplying the right amount of voltage to our microcontroller. In the following semester, we plan to look for a component that will properly step up the voltage

input of the board. For now, we plan to manually solder a 5V step up voltage regulator. We were not able to test whether this design can function as a relay node properly because of the problem with the voltage regulator. Unfortunately, we did not have a working PCB this semester and did not have a chance to deploy Bumblebee. Next semester, we will be working on Bumblebee for no credits in hopes to debug and have a working prototype to deploy and improve on for Spring 2020.

The current housing design was designed with the first PCB design in mind. Since then, we have updated our PCB design to be a quarter of the size it used to be. In the future we hope to communicate with Team Strawberry to create a more compact and space efficient housing for our PCB and components.

Since this semester is the first semester that Bumblebee has used XBee Pro S2C following the discontinuation of the XBee Pro S2B, we hope to collect even more data of the XBee Pro S2C through range testing. We were able to gather some information on the XBee Pro S2C's effective range this semester but there are still more weather, distance, and obstacle variations and trials that we wanted to perform in order to gather more information and further understand the effectiveness of the XBee Pro S2C.

From the previous semester of working with Team Cranberry for communication, we notice that the Bumblebee relay module wasn't able to handle more than one packets at a time. Andrew suggested that it would be possible to handle more than one packets at a time with a small modification on the relay code. In the following semesters, we hope to be able to look into that and work with firmware to integrate that into our code. In addition, we hope to look into 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 Spring 2019 semester, we mainly worked with the first version of the PCB design made last semester. The first half of the semester we worked to debug the problems we had in that PCB. Through looking at past team's work such as reports and meeting minutes as well as working with the leadership team, we found out what may be happening on our PCB for it to not receive and relay packets. With help from the mentors we were able to fabricate a new design to move forward and minimize errors from the last PCB design.

Although we had a PCB design ready for manufacture, we unfortunately was not able to send it in to Seeed Studio due to Ron Ho funding being down for the beginning half of the semester. This held us back for a bit since we weren't able to know if our newly fabricated design is free of all errors. Fortunately, towards the second half of the semester, Ron Ho funding was back and we were able to quickly order the PCB. In the meantime of waiting for Ron Ho funding and PCB manufacture/delivery, we were able to perform a few range testings to understand the effective range of the XBee Pro S2C.

After the arrival of the second PCB design, we were able to quickly populate and test the PCB. We worked largely with firmware to debug the code and in process found out that the 16MHz clock we're using ideally should be ran at 5V instead of the 3.3V that we were supplying at the time. With about 3 weeks left to the semester, we immediately fabricated a third PCB design to fix the errors we've seen in the second design. Although the third PCB design came in and we populated it, we did not have enough time to fully analyze and debug this design. We only know that the regulator we have is not a 5V regulator that steps up voltage.

Due to the delay of ordering PCB with Ron Ho funding being down, we were not able to have a working Bumblebee relay module this semester. We, unfortunately, were not able to meet the deadline and also encountered problems that we have yet to resolve. Nonetheless, we currently have three populated PCB designs with each an improvement from the previous designs. We hope to work on and debug the Bumblebee relay module next semester to achieve a deployable Bumblebee weatherbox on the roof of Holmes Hall in Spring 2020. After that, we hope to potentially work with other sensor weatherbox teams such as Guava and Apple to relay their packets, and perhaps look into 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 learned in the previous semesters. This semester, we ran into problems beyond our control such as Ron Ho funding. From that, we learned how to utilize our time and shift our focus to minimize the effect of problems we cannot control. This project allowed us to visualize what we learned in class and strengthen our knowledge of it, as well as, learning how to deal with outside factors that we may encounter in the real world.

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.