EE 496 Final Paper for Fall 2020 2nd Generation Relay Node: Team Bumblebee

Authors: Arnold Flores, Raellis Young

Department of Electrical Engineering

University of Hawaii at Manoa

Renewable Energy and Island Sustainability Program

Smart Campus Energy Lab

18 December 2020

Abstract: **‘**Bumblebee’ is Smart Campus Energy Lab’s (SCEL) 2nd generation communications module designed to relay meteorological data collected from weather boxes on Holmes Hall. Meteorological data that is collected from the weatherboxes on top of Holmes Hall is sent wirelessly using an XBee device inside the weatherbox to the gateway in Holmes Hall. Due to the limited range of the weatherbox sensor node, ‘Bumblebee’ was developed to extend the range of the sensor node network.

Table of Contents

I. Introduction

In 2008, the state of Hawaii and the Department of Energy decided to collaborate to reduce the dependence on imported fossil fuels. Due to Hawaii being the most fossil fuel-dependent state in the United States, the goal of the Hawaii Clean Energy Initiative is to become 100 percent energy clean by 2045[1]. Smart Campus Energy Lab (SCEL) was made in effort under the University of Hawaii's Renewable Energy and Island Sustainability (REIS) program to support the goals of the Hawaii Clean Energy Initiative. The REIS program's goal is to help UH Manoa to run on its own microgrid that is powered by 100 percent renewable energy.

SCEL's goal is to help make this microgrid by designing, building, and deploying "weather boxes" that can collect and send meteorological data such as temperature, humidity, and solar irradiance. These weather boxes should be designed and developed so as to be low cost so that they may be mass produced. Meteorological data that is gathered from these weather boxes will provide help in planning for future renewable energy installations and data for forecasting algorithms[2].

Bumblebee does not include any sensors, so it does not collect any data. The ideal location for Bumblebee boxes would be between distant sensors and the lab gateway. Bumblebee is based on the third generation of sensor nodes, Cranberry, and uses many of the same components including the Atmega328P microcontroller, the XBee Pro S2C, and is powered by a solar panel and rechargeable battery. The Bumblebee box has a simplified circuit and its own PCB design and housing. Our team also does range testing and networking with XBees.

II. Relay Node: Bumblebee Overview

Block Diagrams

The figures shown below are the overall designs of Bumblebee's power and communication distribution. Figure 1 shows our signal/communication block diagram. While Figure 2 illustrates the power block diagram.

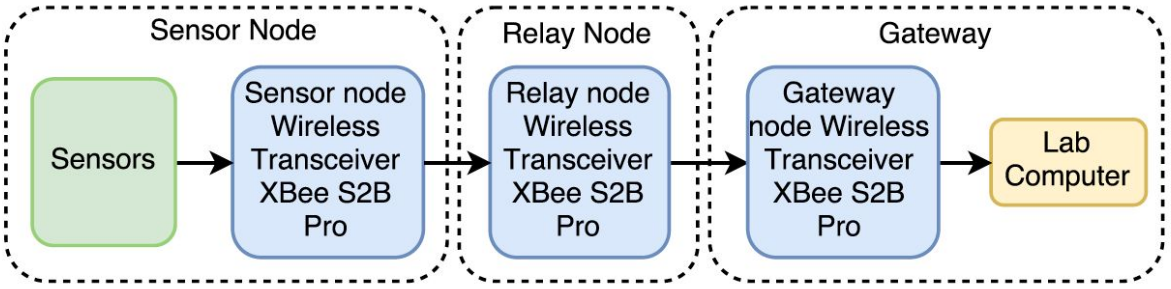

Sensor Node
Relay Node
Gateway
Sensors
Sensor node Wireless Transceiver XBee S2B Pro
Relay node Wireless Transceiver XBee S2B Pro
Gateway node Wireless Transceiver XBee S2B Pro
Lab Computer

Figure 1: Signal/Communication Block Diagram

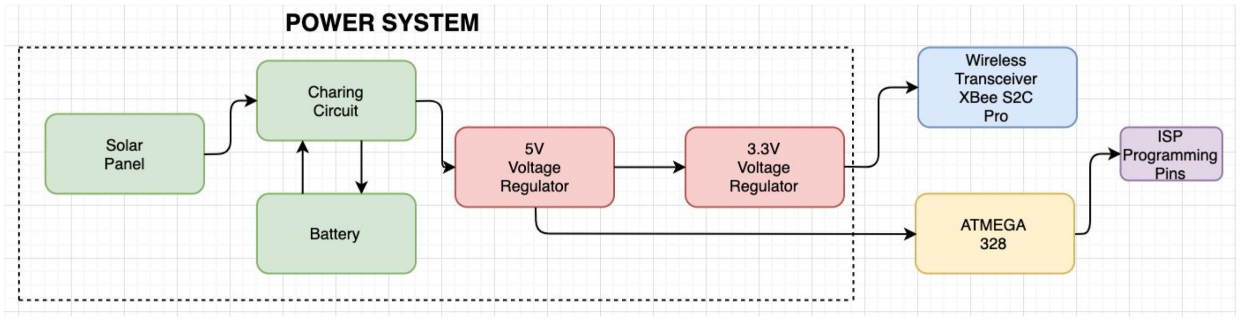


Figure 2: Power Block Diagram

Figure 1, the signal/communication block diagram shows how the different nodes will communicate wirelessly. The signal block diagram illustrates the basic path of the data traveling from the sensor node and eventually to the gateway computer. Since DigiKey has discontinued XBee S2B Pro, we have switched and currently use the XBee S2C Pro. This block diagram shows that from the weatherboxes that collect data from its sensors, it will construct a data packet and send it to the relay node. Bumblebee will receive that data and send it to the gateway

XBee to be sent to the lab gateway. Overall, this diagram illustrates that data has to go through multiple Bumblebee nodes in order for the lab gateway to receive a packet.

In Figure 2, the power block diagram describes the functionality of each of the hardware components connected. To help power the Bumblebee relay node, a charging circuit and solar panel are incorporated into the board. A 5V boost converter was implemented in previous semesters to have the Atmega run at 5V with a 16MHz clock. Previously, only one 3.3V voltage regulator was used to supply enough voltage to the circuit, considering that no other sensors are connected to Bumblebee. Implementing a 5V boost converter will improve power battery life and will allow Bumblebee to function properly as needed.

Designs: Schematic and PCB Layouts

Originally, the main design for the schematic of the relay node is based on the Cranberry weatherbox. However, over the past few semesters, the design has been modified to fix connection mistakes and to minimize debugging errors. One of the main challenges that the team always ran into was determining whether the error was from software or hardware. Because of this, to eliminate some possible errors on the hardware, the team changed the design of the board by replacing some parts. One of the main changes was to use a breakout board for the solar charging circuit instead of keeping the implementation from Cranberry’s design. Another change in the design was making the Atmega run on 5V, rather than 3.3V. Unfortunately, right now, we are still in the process of implementing the part successfully.

Referring to last semester’s team, the problem that needed to be fixed was that we needed a working 5V regulator to boost the current 3.7V to power the Atmega and 16MHz clock. We modified the schematic and PCB layout from last semester to implement two different 5V regulator parts. Version 4.0 used a 5V step-up breakout board that we use on our bare Arduino board to minimize debugging errors. Version 4.1 is implemented with an SMD 5V boost regulator that we found online. Luckily, previous team members had rejoined the team and were able to be more productive and get more things done in the first half of the semester.

For version 4.0, shown in Figure 3 below, the team decided to implement the 5V step-up breakout board from Sparkfun. Since the main reason why we wanted to make the Atmega run on 5V is to minimize debugging errors, we've kept the bare Bumblebee and PCB consistent by using the same part. However, as we were trying to order this part, we found out that it has been retired. We still pushed through with design because we want to see if the board will be able to successfully transmit data. Just like the most recent design from the previous semester, the size of the PCB is kept the same. It is 1.5 inches by 1.75 inches. Since the relay node does not have sensors, it is much smaller compared to other weather boxes.

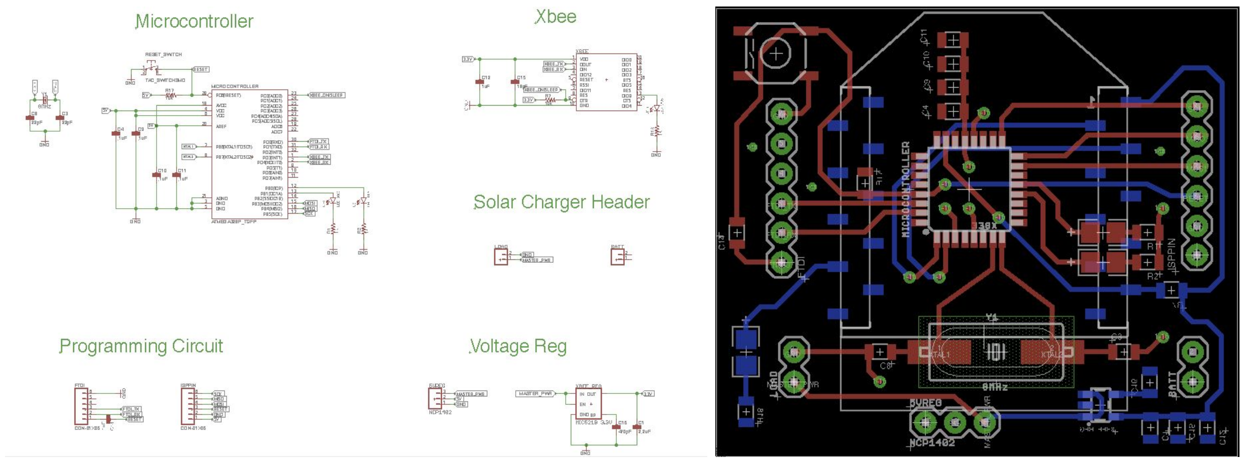


Figure 3: Schematic and PCB Layout of Ver. 4.0

As mentioned earlier, the 5V breakout board has been a retired product and is no longer for sale. Because of that, the team designed another board, which is version 4.1 shown in Figure 4. For this version, we searched for a boost converter that has the same specifications as the part from Sparkfun. The part that we decided to use is the TPS61222DCKR from DigiKey. Similarly, this is a boost switching regulator with a fixed output of 5V and 200 mA. The current members do not have experience in finding a suitable part to use. We had a hard time looking for a good replacement for the voltage regulator looking at the different specifications of the parts. With the part that we settled with, we are not sure if this part is a good match and will have to do testing in order to find out.

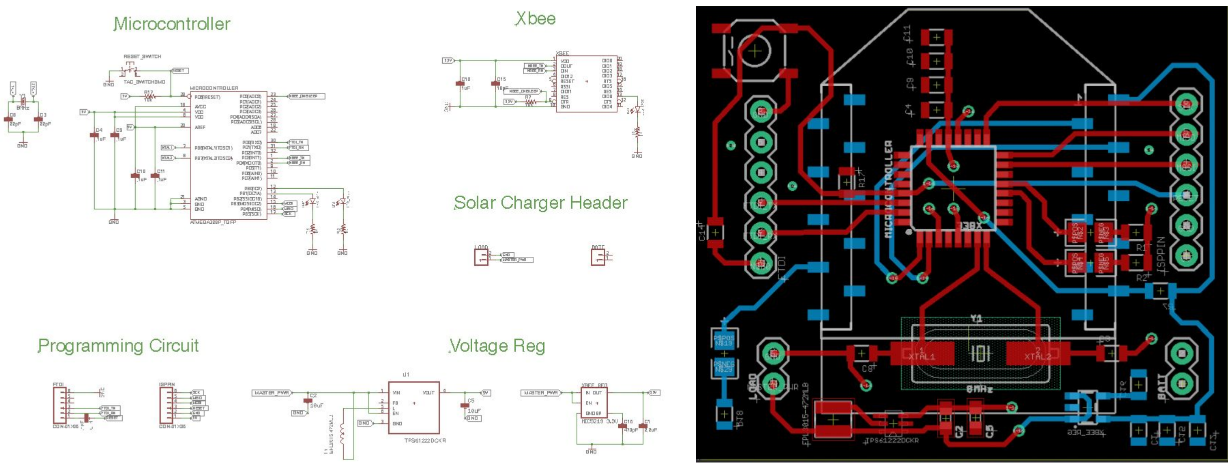


Figure 4: Schematic and PCB Layout Ver. 4.1

Bare Arduino Board

The bare Arduino board or also known as the bare Bumblebee is a working communications relay device and a prototype of the PCB Bumblebee. The bare Bumblebee is essentially the design in Figure 3 but built on a breadboard using breakout boards and DIP chips. The main use of the bare Bumblebee is to test our design before it is implemented on a PCB. We also used the bare Bumblebee for range testing of the XBee S2C Pro.

We rebuilt the bare Bumblebee from last semester because there was a sync error that came up before. Therefore, we decided to reference a picture from the previous semester and build it exactly as shown in Figure 5. This bare Bumblebee board was used for range testing and was successful in receiving data packets from the coordinator XBee.

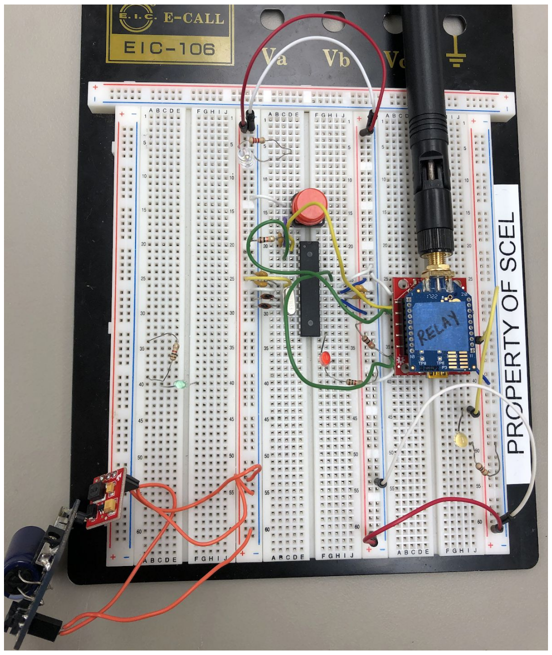


Figure 5: Bare Bumblebee

Packet Range Testing

The XCTU and Arduino IDE programs were used to test the abilities of the XBee to transmit and receive 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 coordinator XBee acts as the gateway or receiving-end. The two routers are used for the sending-end (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 intervals can be written using Andrew Rapp's XBee library for Arduino IDE that can be found online[3]. 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.

Field Range Testing

The purpose of range testing is to take into account as many variables as possible, such as obstacles and weather, and to gather data on how far the XBee can reach certain distances . To conduct this testing, a built-in range test software program in XCTU should be used. The data values include local strength, remote strength, packets sent and received, TX errors, packets lost and percentage of packets received. By determining those data values, we can conclude how the signals would behave with different variables. These variables could be based on the weather and the more wind there is, the more likely the signal will be interrupted. This variable also includes when students are passing by between the two XBee modules and will result in some disruption in the data.

Set-up

As you can see in Figure 6, the line-of-sight testing provides a clear pathway between the local and remote XBees while increasing the distance in between after each test. With the not line-of-sight testing, there would be obstructions in between the local and remote XBee. As the tests progress, the number of obstructions in between and the distance between would increase.

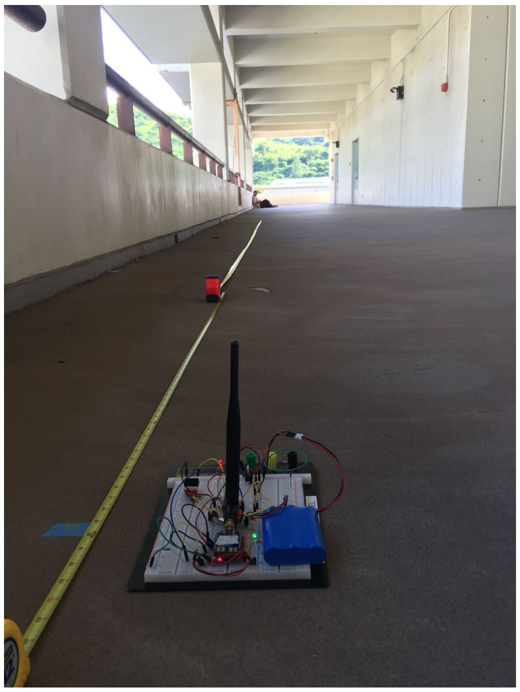
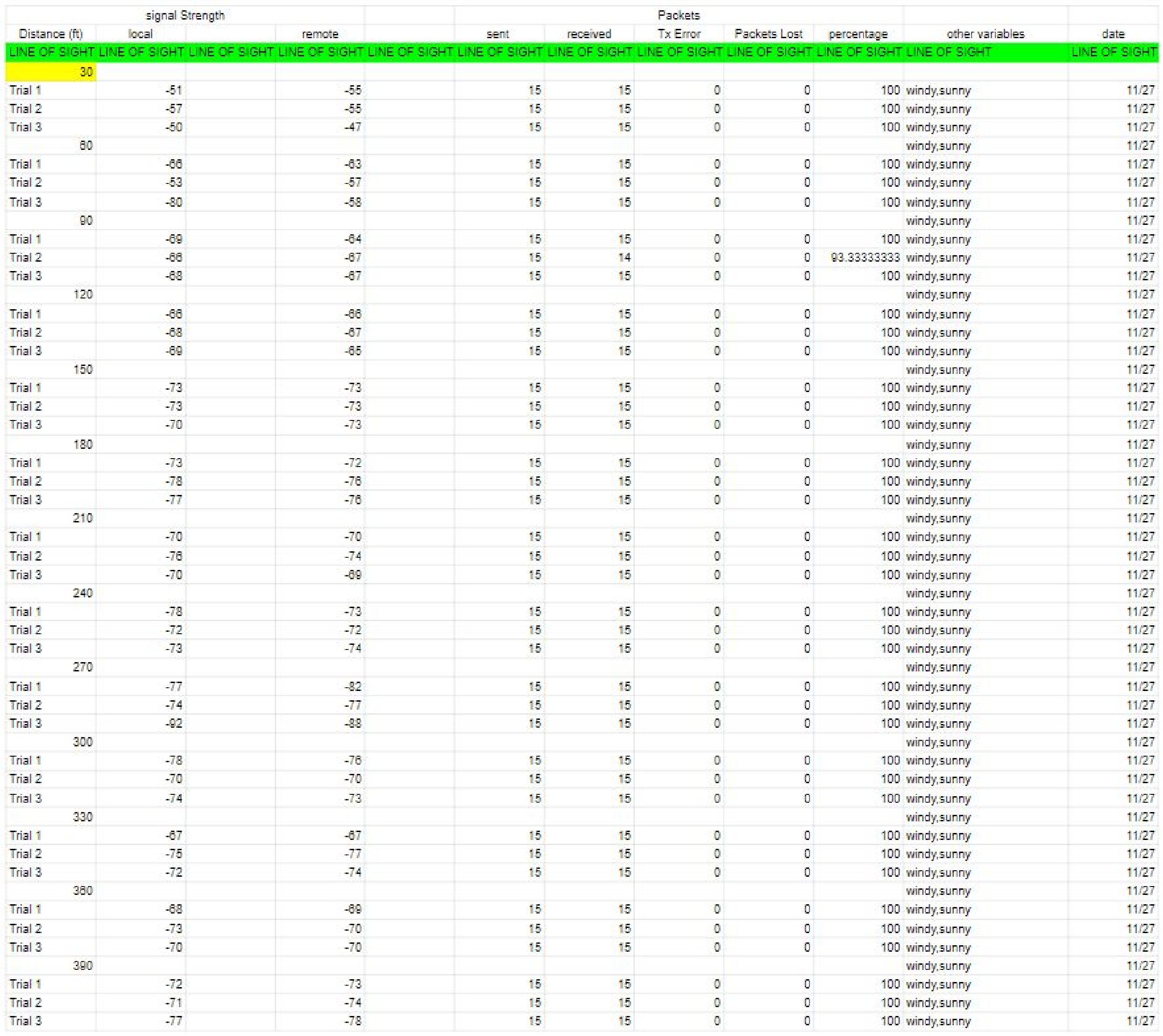


Figure 6: Line of Sight Set-up

Range (Field) Testing Results

Using the XBee S2C Pro, we determined the performance of it by completing range-testing on our bare Bumblebee board in three different ways; line-of-sight, not line-of-sight, and through different floors.

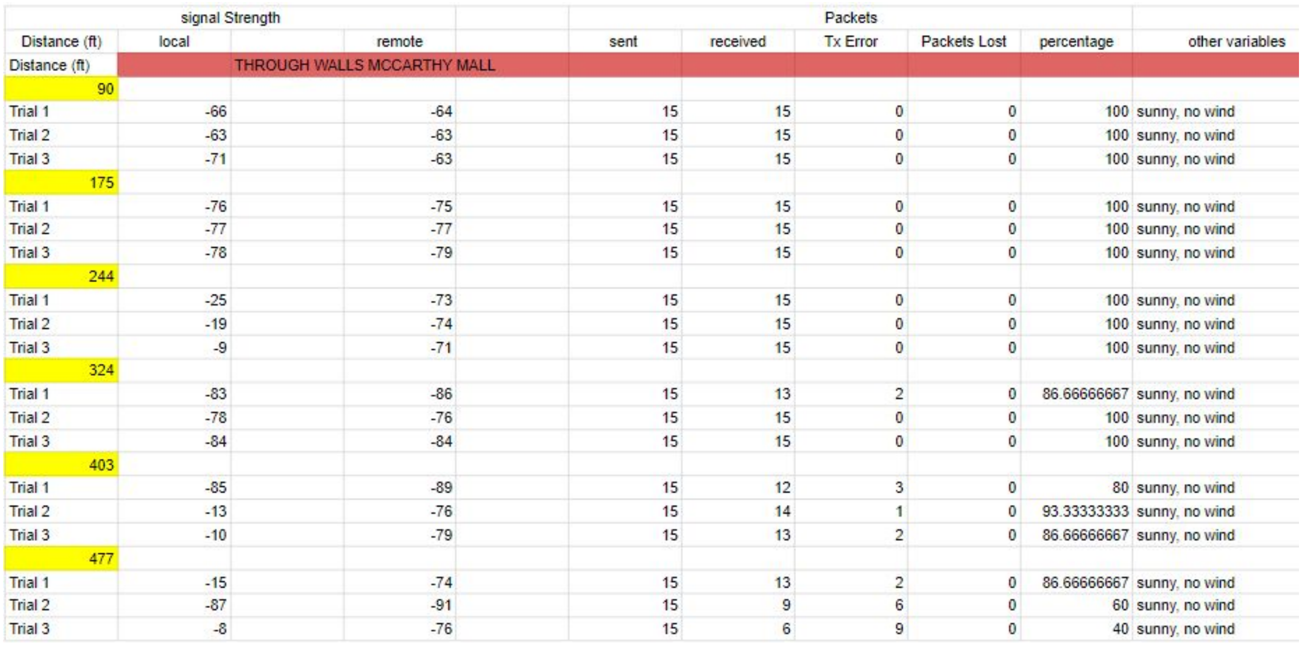
The purpose of range testing is to take into account as many variables as possible and to gather data on how far the XBee can implement certain distances such as obstacles and weather. To conduct this testing, we used a built-in range test software program in XCTU. The data values included were local strength, remote strength, packets sent and received, TX errors, packets lost and percentage of packets received. A good signal strength would be greater than -80.



	signal Strength						Packets				
Distance (ft)	local		remote		sent	received	Tx Error	Packets Lost	percentage	other variables	date
LINE OF SIGHT	LINE OF SIGHT	LINE OF SIGHT	LINE OF SIGHT	LINE OF SIGHT	LINE OF SIGHT	LINE OF SIGHT	LINE OF SIGHT	LINE OF SIGHT	LINE OF SIGHT	LINE OF SIGHT	LINE OF SIGHT
30											
Trial 1	-51		-55		15	15	0	0	100	windy,sunny	11/27
Trial 2	-57		-55		15	15	0	0	100	windy,sunny	11/27
Trial 3	-50		-47		15	15	0	0	100	windy,sunny	11/27
60										windy,sunny	11/27
Trial 1	-66		-63		15	15	0	0	100	windy,sunny	11/27
Trial 2	-53		-57		15	15	0	0	100	windy,sunny	11/27
Trial 3	-80		-58		15	15	0	0	100	windy,sunny	11/27
90										windy,sunny	11/27
Trial 1	-69		-64		15	15	0	0	100	windy,sunny	11/27
Trial 2	-66		-67		15	14	0	0	93.33333333	windy,sunny	11/27
Trial 3	-68		-67		15	15	0	0	100	windy,sunny	11/27
120										windy,sunny	11/27
Trial 1	-66		-66		15	15	0	0	100	windy,sunny	11/27
Trial 2	-68		-67		15	15	0	0	100	windy,sunny	11/27
Trial 3	-69		-65		15	15	0	0	100	windy,sunny	11/27
150										windy,sunny	11/27
Trial 1	-73		-73		15	15	0	0	100	windy,sunny	11/27
Trial 2	-73		-73		15	15	0	0	100	windy,sunny	11/27
Trial 3	-70		-73		15	15	0	0	100	windy,sunny	11/27
180										windy,sunny	11/27
Trial 1	-73		-72		15	15	0	0	100	windy,sunny	11/27
Trial 2	-78		-76		15	15	0	0	100	windy,sunny	11/27
Trial 3	-77		-76		15	15	0	0	100	windy,sunny	11/27
210										windy,sunny	11/27
Trial 1	-70		-70		15	15	0	0	100	windy,sunny	11/27
Trial 2	-76		-74		15	15	0	0	100	windy,sunny	11/27
Trial 3	-70		-69		15	15	0	0	100	windy,sunny	11/27
240										windy,sunny	11/27
Trial 1	-78		-73		15	15	0	0	100	windy,sunny	11/27
Trial 2	-72		-72		15	15	0	0	100	windy,sunny	11/27
Trial 3	-73		-74		15	15	0	0	100	windy,sunny	11/27
270										windy,sunny	11/27
Trial 1	-77		-82		15	15	0	0	100	windy,sunny	11/27
Trial 2	-74		-77		15	15	0	0	100	windy,sunny	11/27
Trial 3	-92		-88		15	15	0	0	100	windy,sunny	11/27
300										windy,sunny	11/27
Trial 1	-78		-76		15	15	0	0	100	windy,sunny	11/27
Trial 2	-70		-70		15	15	0	0	100	windy,sunny	11/27
Trial 3	-74		-73		15	15	0	0	100	windy,sunny	11/27
330										windy,sunny	11/27
Trial 1	-67		-67		15	15	0	0	100	windy,sunny	11/27
Trial 2	-75		-77		15	15	0	0	100	windy,sunny	11/27
Trial 3	-72		-74		15	15	0	0	100	windy,sunny	11/27
360										windy,sunny	11/27
Trial 1	-68		-69		15	15	0	0	100	windy,sunny	11/27
Trial 2	-73		-70		15	15	0	0	100	windy,sunny	11/27
Trial 3	-70		-70		15	15	0	0	100	windy,sunny	11/27
390										windy,sunny	11/27
Trial 1	-72		-73		15	15	0	0	100	windy,sunny	11/27
Trial 2	-71		-74		15	15	0	0	100	windy,sunny	11/27
Trial 3	-77		-78		15	15	0	0	100	windy,sunny	11/27

Figure 7: Line-of-Sight Range Testing Results

The first range test conducted was the line-of-sight testing completed on 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 farther away in increments of 30 feet until we reached the other end of the building. The weather condition of the first range test was sunny with light wind. We received most of the packets sent throughout the entire testing procedure as shown in Figure 7. As expected, the signal strength weakened as the distance increased. We are not sure why there are slight variations where one data packet got lost, but as a whole, this range-testing was a success. There was one moment at around 270ft when the antenna of the local XBee fell and was oriented parallel to the ground when it should have been perpendicular. The signal at that point was not good. Now we know that the orientation of the antennas are important and affects the signal strength of the XBees. When we were testing, we also got random gusts of wind which had weakened the signal, but didn't result in the loss of data packets. Since we only lost one packet and the signal wasn't too bad, we would consider this range test a success.



	signal Strength					Packets			
Distance (ft)	local	remote		sent	received	Tx Error	Packets Lost	percentage	other variables
Distance (ft)	THROUGH WALLS MCCARTHY MALL								
90									
Trial 1	-66	-64		15	15	0	0	100	sunny, no wind
Trial 2	-63	-63		15	15	0	0	100	sunny, no wind
Trial 3	-71	-63		15	15	0	0	100	sunny, no wind
175									
Trial 1	-76	-75		15	15	0	0	100	sunny, no wind
Trial 2	-77	-77		15	15	0	0	100	sunny, no wind
Trial 3	-78	-79		15	15	0	0	100	sunny, no wind
244									
Trial 1	-25	-73		15	15	0	0	100	sunny, no wind
Trial 2	-19	-74		15	15	0	0	100	sunny, no wind
Trial 3	-9	-71		15	15	0	0	100	sunny, no wind
324									
Trial 1	-83	-86		15	13	2	0	86.66666667	sunny, no wind
Trial 2	-78	-76		15	15	0	0	100	sunny, no wind
Trial 3	-84	-84		15	15	0	0	100	sunny, no wind
403									
Trial 1	-85	-89		15	12	3	0	80	sunny, no wind
Trial 2	-13	-76		15	14	1	0	93.33333333	sunny, no wind
Trial 3	-10	-79		15	13	2	0	86.66666667	sunny, no wind
477									
Trial 1	-15	-74		15	13	2	0	86.66666667	sunny, no wind
Trial 2	-87	-91		15	9	6	0	60	sunny, no wind
Trial 3	-8	-76		15	6	9	0	40	sunny, no wind

Figure 8: Not Line-of-Sight (Through Trees at McCarthy Mall)

The second range test done was 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, increasing the distance and the number of obstructions. As shown in Figure 8, all packets were received and sent for the first three distances. After that, as the variables increased, there were a few packets lost. The greatest number of packets lost were in the last trial of 477 feet with seven trees between the two XBees. We were shocked by how the signal strength numbers differed from last semester's tests. Last semester, at farther distances, the signal was around -7 to -8 which was odd because they represent a strong signal. We suspect that the XBees lost connection at those distances and would send to the other local XBee present instead of the remote XBee. There weren't many people who passed by while testing. Again, the results may be more accurate than last time because our XBees didn't lose connection. We conclude that a good distance to have our XBees at, without losing packets, would be around 244ft with 4 or less obstructions in between.

The third range testing that we usually do is not line-of-sight testing conducted in Holmes Hall floors. For this test, we would set the local signal on Holmes Hall 4th floor and vary the remote relay module to 3rd, 2nd, and 1st floor. We felt that this test was not very intuitive because we don't know how tall the floors are in Holmes Hall.

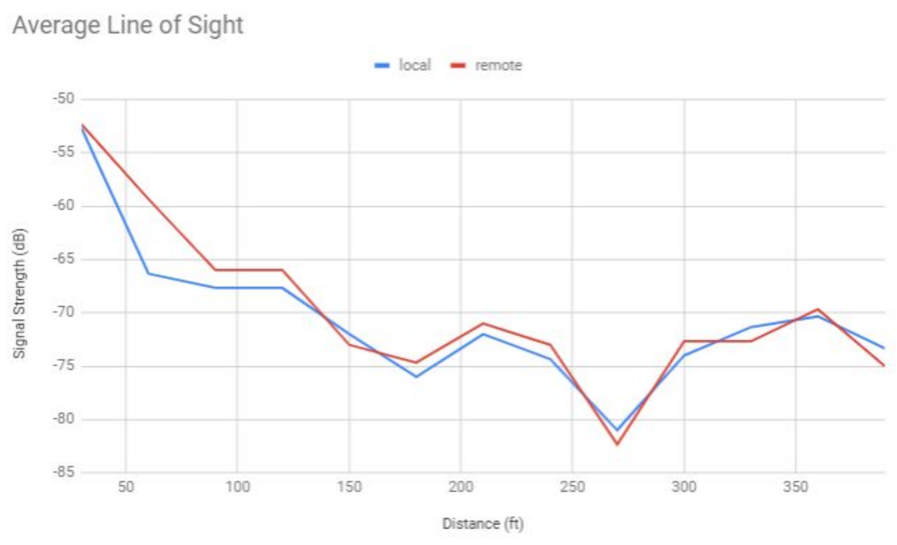


Figure 9. Line of Sight Results

As shown again, most of the packets were received for the entire testing process. By looking at the numbers in Figure 9, we can see that the local and remote signal strength gets weaker as we increase the distance and dipped down at 270 feet due to the antenna's orientation. A graph representation of our not line-of-sight results can be seen in Figure 10. The signal goes down for the remote XBee as the number of obstructions and distance increases, whereas the signal oddly improves for the local XBee.

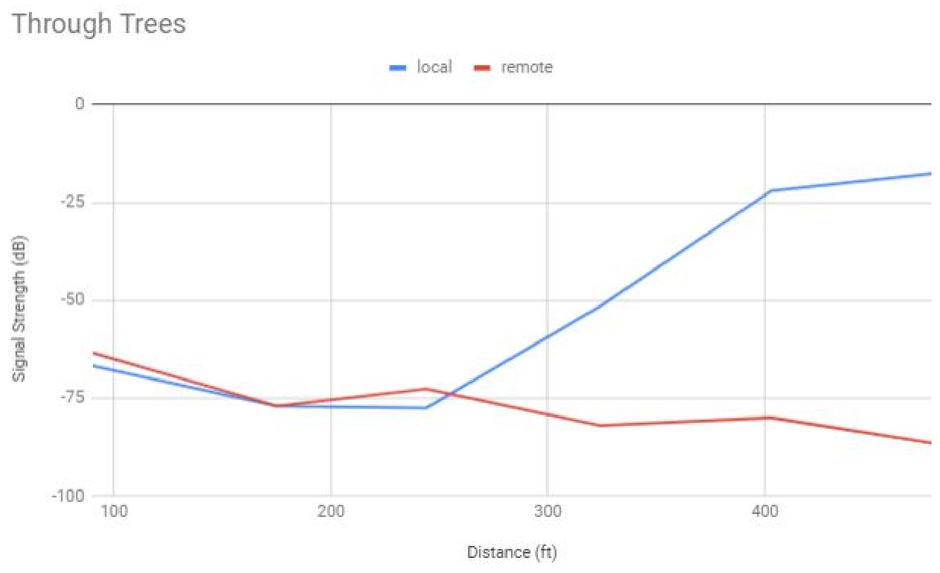


Figure 10: McCarthy Mall Results

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 the XBee S2C Pro. 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 range testings with the XBee S2C Pro and conduct more trials and sets of tests for each location. We also plan to debug the two fabricated PCBs so it works and attempt to range test with those as well.

III. Research - Networking of XBee

During the beginning of the semester, we were faced with lockdowns due to the COVID-19 pandemic. Because of the situation, we were forced to stay at home and not be able to work in the lab for the first month of school. Since Bumblebee is a hardware team, it was difficult to continue testing and debugging the new boards from last semester. To keep the progress of the project, the team decided to spend the remainder of the semester doing research on the networking of XBee modules and other types of technology similar to the XBee. During the first few weeks, we've learned about one-to-many and many-to-one routing configurations and introduced the team to our new members.

One-to-many routing is also known as point-to-multipoint communication. In this network, a coordinator acts as a central node that can communicate with one or multiple end devices. The central node starts the network by a broadcast to all end devices. This will then allow the end devices to join the network in the selected frequency channel. Since there are multiple devices communicating with the central node, the coordinator also provides network synchronization. By doing so, this will eliminate delay in data transmission and reception. In this network, the synchronization is done by polling nodes[4]. Polling is an access control method that allows all the end devices in the network to send data. The end devices check for the readiness or state of the central host first, then one will be chosen to send data[5]. The diagram shown in Figure 11 illustrates the arrangement of the devices in this network. This network topology is called a star. In this type of network topology, there can only be one coordinator.

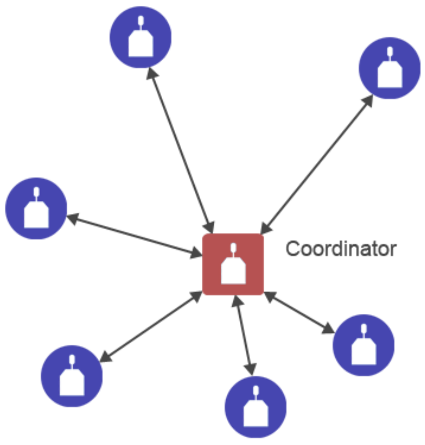


Figure 11: One-to-Many Network Topology called Star [4].

Another configuration we learned about is the many-to-one routing. In this configuration, multiple nodes need to communicate with a single node. This single node performs a centralized function and is commonly known as a collector or concentrator. In figure 12 below, it shows that there are more than two roles unlike in the one-to-many routing. The advantage of this network is the collector sends a many-to-one broadcast transmission to form reverse routes on all other devices. When the other XBee modules receive this request, the module creates a path back to the collector by storing a reverse many-to-one routing table entry. This way these devices do not need to discover a route to send the data to the collector. When every XBee discovers a route to the collector, it will generate a broadcast route discovery message. If there are multiple XBee, it will cause the network to be overloaded with these messages. However, in this configuration, there are no responses generated so the network traffic congestion is minimized[6].

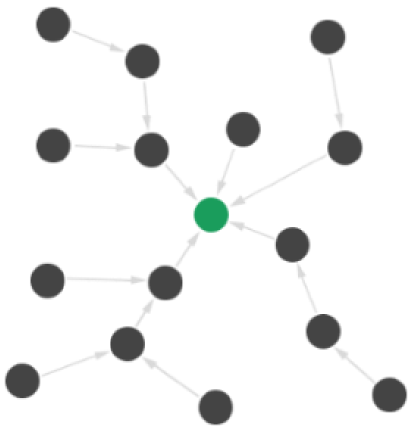


Figure 12: One-to-Many Network Topology called Star [6].

To enable this routing in a network, the configuration of the data collector has to be set using the XCTU software. The Many-to-one Route Broadcast Timeout (AR) parameter sets a 10 second time interval for sending the many-to-one broadcast transmission. When configuring the XBee module, this parameter needs to be set to 0 to enable this option. Once this is assigned, it will cause the XBee to immediately send a single many-to-one broadcast to other devices in the network. There are two ways to disable this option in a network. The first one is to explicitly set the AR parameter of the collector to 0xFF. When this is set, the many-to-one broadcast by the collector will end. The other way is to reform the network by broadcasting a System Reset (SR) command. By doing this, it will remove the data collector status as an aggregator from the routing tables[6].

IV. Problems and Solutions

This semester we ran into various problems while working on Bumblebee. The biggest problem we faced was the COVID-19 pandemic. This really reduced the amount of work we could complete in the lab this semester. We couldn't get to do range testing with the two PCB relay nodes we designed and fabricated. We were forced to delay debugging our boards and mainly focus on obtaining results for our range testing.

Early in the semester, we had researched different networking configurations and other alternatives that we could consider besides the XBee. An ongoing problem we had from the previous semesters to fix the powering problem on our PCB. After populating new boards and debugging soldering problems on our version 4.0 PCB, we were finally able to get an LED light on. From last semester, we ordered from a manufacturer in San Francisco called OSHPark. Although we eventually obtained our PCBs it did take a lot longer than we expected because we weren't able to pick up parts from the EE Office. We spent our time debugging version 4.0 and understanding the relay and sending code. We also taught our new members how to bootload and connect the boards together on XCTU.

When range-testing with the bare Arduino board, we had a problem with how fragile it was. All the parts were just plugged in using wires which resulted in the parts falling off easily and causing us to fix it every single time we carried it somewhere. This resulted in delays where range-testing that could've been completed within an hour, became a two-hour task.

We also ran into problems with our code. Uploading code from Arduino onto our relay and sending XBees sometimes didn't work so we had to call Sharmaine to help us. We also couldn't detect the XBees from the coordinator so we had to look at the code again to see what was wrong.

V. Future Work

Next semester, we plan to debug version 4.0 and repopulate around two or three more boards. We also want to populate more version 4.1 and debug it if any problems arise. We want to range-test with both versions to see which works better to deploy. We also plan to fabricate a new board, debugging, and testing as well. We want this new version to have an LED to indicate that our board is being reset. This idea was taken from Team Guava when they were teaching us how to bootload our boards.

We want to work with the housing team to make a new box for the smaller PCB board we now have compared to the size we had three semesters ago. We will have to come up with a new design that will be smaller and possibly cheaper as well.

Something that we may not have realized while range-testing is that the signal of the XBee could be affected by the EM waves that are present in the labs we are conducting range testing next to on Holmes Hall 4th floor. That is something that we need to look further into next semester. We were thinking of doing line-of-sight range-testing on the ground floor of Holmes Hall where there would be fewer variables to measure. We also want to try range testing from one building to another.

Also, if time permits, we would like to experiment with either the one-to-many or many-to-one configuration to see which network would be efficient in the future. For instance, having multiple sensor nodes communicate to one relay node, which would then transmit to the next relay or to the gateway.

VI. Conclusion

At the beginning of the semester, Team Bumblebee was prepared to make progress in creating a working communications relay node with returning members Arnold and Raellis, and new members Francis and Lauryn. From the previous semester we had already received our newly designed versions 4.0 and 4.1 PCB's, but due to the COVID-19 pandemic we were not able to get into the lab the first month of the semester. As we got more direction from the school we were able to finally get into the lab once a week on Sundays and we used the rest of our lab hours to do research and plan on weekdays. This made it very difficult to physically work on our board with each other in Team Bumblebee and to collaborate with other lab teams.

Although we were limited on how we worked on Bumblebee we didn't let that stop us from working on the project. Through the use of video conference calls and text, we were able to communicate effectively with each other and collaborate on ideas. We were able to teach Lauryn and Francis about how Xbee works and we walked them through the needed tutorials. This semester we were able to finally populate and power both versions 4.0 and 4.1after several sessions of debugging. We were also able to take a deeper look into the software aspects of Team Bumblebee with the sending and relay codes, and the bootloading code. As for range testing, we were able to gather more range testing results toward the end of the semester. The new data that was gathered this semester was then compiled with data from previous semesters.

Overall, Team Bumblebee was able to make good progress in developing a deployable relay node even with the complications of the COVID-19 pandemic. Now that the two returning members are leaving, we believe that Francis and Lauryn are well prepared to continue the progress of Team Bumblebee and are capable of teaching new incoming members. In the upcoming semester, team Bumblebee will be ready to again pick up where we left off. Next semester we will bootload and program both versions 4.0 and 4.1and debug if needed. We also hope to do more extensive range testing under variables such as rain. Another concern that we would like to address next semester is to implement a reset LED on a new version 4.2 PCB. This LED will help Team Bumblebee in efforts of debugging and deployment. As we finish up work with SCEL we are happy to leave the future work of Team Bumblebee in the hands of Francis and Lauryn as we strive to help the University of Hawaii at Manoa become a 100% renewable energy institution.

References

[1] Hawaii Clean Energy Initiative. (n.d.). Retrieved May 08, 2020, from http://www.hawaiicleanenergyinitiative.org/.

[2] (n.d.). Retrieved May 08, 2020, from http://scel-hawaii.org/research/.

[3] Rapp, Andrew "Arduino library for communicating with XBee radios in API mode" Dec. 2016, https://github.com/andrewrapp/XBee-arduino.

[4] "Point-to-multipoint Communication," DIGI, 09-Sep-2019. [Online]. Available: https://www.digi.com/resources/documentation/Digidocs/90001456-13/concepts/c_point_to_multipoint.htm.

[5] A. Singhal, "Polling in Networking | Access Control Method," Gate Vidyalay, 26-Apr-2019. [Online]. Available: https://www.gatevidyalay.com/polling-access-control-in-networking/.

[6] "Many-to-one Routing," DIGI, 29-Aug-2019. [Online]. Available: https://www.digi.com/resources/documentation/Digidocs/90001942-13/concepts/c_many_to_one_routing.htm. [Accessed: 16-May-2020].