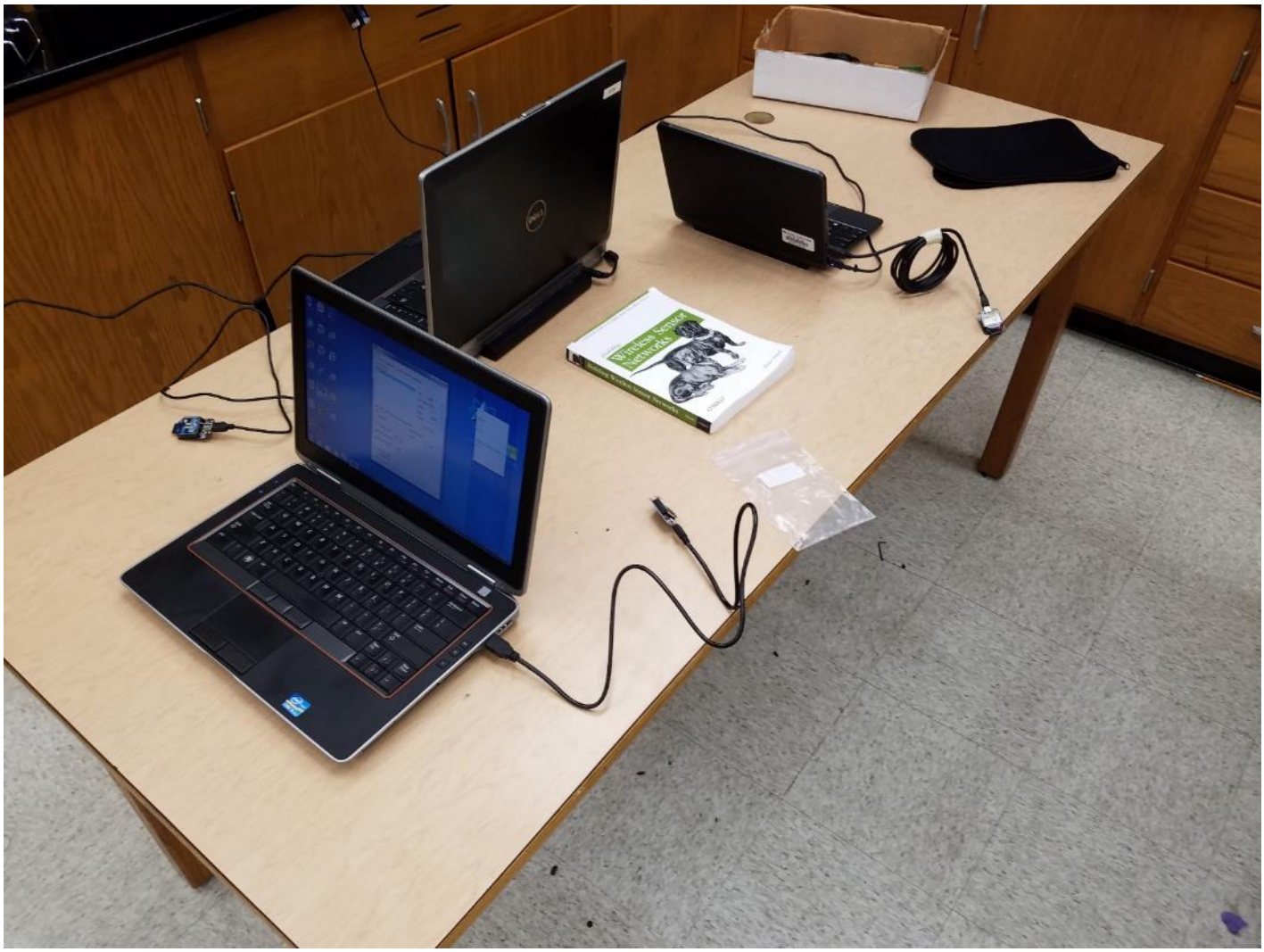



# Individual Project Portfolio

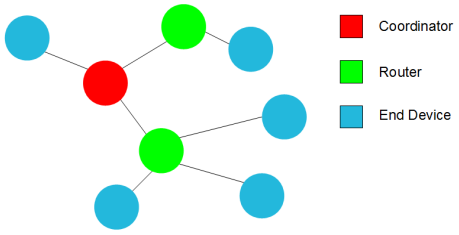
XBEE ZIGBEE MESH NETWORK

**Nikhil Sharma** | Robotics 4 | 2016-2017

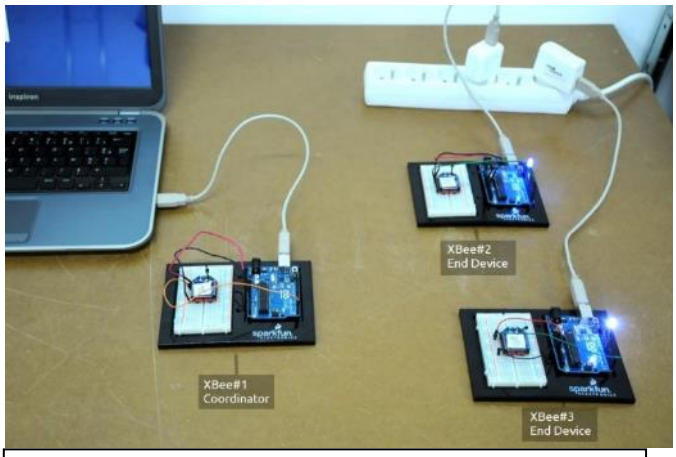
# Table of Contents

# Metric 1- Design, Drawings and Specifications

## DRAWINGS, PHOTOS, SCANS




This is the **mesh network**. It utilizes a series of routers, end devices, and a coordinator to communicate messages across the devices (Like a game of whisper).




This is an example of what our initial set up may look like, with the 1 coordinator and 2 end-devices.

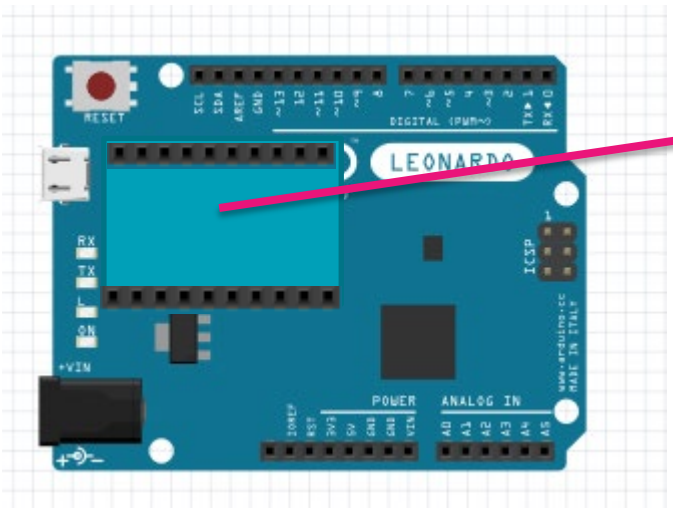
## DESCRIPTIONS OF PROJECT

The goal of this project will be to successfully create a mesh network between at least 3 Xbee devices and utilize the network created to integrate with Arduino Leonardo boards to communicate sensor data across the devices.

## THE "WHY" OF THE PROJECT

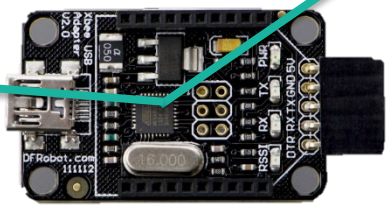
Throughout this project, I will continue learning about open source platforms and developing my Arduino skills as well as learn the basics of networking and the Internet of Things. The first few steps will involve learning about the basics of networking, and configuring the various network addresses correctly between the Xbee devices to set up the mesh network. This will be valuable not only in this situation, but in any computer networking classes that I might take. Additionally, I will also be working with the open source platform Arduino, furthering their knowledge of the platform. Throughout the project, I will be learning the basics of the Internet of Things concept. This concept states that everyday devices will have Internet connectivity, allowing them to send and receive data wirelessly. This concept is similar to the proof of concept that will be demonstrated, with the robots sending and receiving sensor data, albeit at a smaller scale.

## SENSOR AND ACTUATOR MAPPING





This pink line represents how the Xbee module will attach to the Leonard socket, bypassing any bread-boarding. However, we cannot use XTCU (the program for working with the Xbee) with the Leonardo.


This green-blue line represents how I will be programming the actual Xbee, setting up the Mesh network. It requires a Xbee-to-USB device so that XTCU can read/write to the Xbee. Initially, I thought the Leonardo would work for the USB connection, but XTCU doesn't seem to be reading the Xbee device's settings.

## GENERAL PARTS LIST

In the course of this project, I will be using the following components:

- Xbee 2mW PCB Antenna- Series 2
- Arduino Leonardo with Xbee Socket
- Pre-built Learning Robots (which I will be interfacing the Xbees with)
- Dell Venue 10
- Xbee USB Adaptor

# Metric 2- Goals, Tasks, and Rationale

## PRIMARY GOALS

1. Establish a network between 1 Xbee API Cooridinator and 2 Xbee API End Devices
2. Successfully send a string of text using the created network
3. Use the created Mesh Network to send sensor data from one device to a coordinator then to another end device

## SECONDARY GOALS

1. Integrate the Xbee devices with pre-built Learning Robots and successfully send sensor data across the robots
2. Using the established network, coordinate multiple Learning Robots to move in a square path (based of colored tape placed on ground) without touching each other.

## ROAD BLOCK TO GOALS

**Primary Goal Potential:**

1. API vs AT Mode. AT mode is simple point to point communication between two Xbee modules, building off of the Xbee S1 devices. API Mode is the one necessary for this challenge, and it is also much harder to configure. API mode allows you to build the mesh network between the different Xbee devices. Only one of the Xbee devices can be a coordinator, the others can be either routers or end devices. To achieve my goal, I would set two Xbees as end devices, meaning they would be the ones transmitting the data, and one Xbee as the coordinator to relay the message between the two end devices.
2. Once the network is established, this goal requires the knowledge of basic XTCU commands to obtain the Serial High, Serial Low, Destination High, and Destination Low as well as the PAN ID. Learning these commands should take a week, but then this goal becomes easily obtainable.
3. The challenge with this goal will be to interface sensors with the Arduino Leonardo board that I'm using. Bread-boarding will have to be involved, I'm thinking of just interfacing one ultrasonic sensor and then sending those values, as this will help with the first secondary goal.

**Secondary Goal Potential:**

1. The difficulty with this will be integrating the Xbee/Leonardo with the existing microcontroller on the Learning Robots. Since they already have a Arduino Mega with a Mega shield, I would have to figure out some way of obtaining the sensor data from the Mega.

2. The challenge to this goal will be strictly programming, having to program the Arduino to read the Xbee data instantly and respond to that. It may not be as smooth as I envision, but it would be a great advanced proof of concept.

## STEPS TO OVERCOME ROADBLOCKS

1. *Building Wireless Sensor Networks* is a book provided to me for work with the Xbee devices, and I believe it will help me in establishing the mesh network with the coordinator and end devices. If the book doesn't help, then there are examples I've found online where hobbyists have detailed the steps in building a point-to-point network, and I can expand from there. This is the biggest component of my project, and the one I believe will take the longest. Setting up the actual mesh network shouldn't take too long with XTCU, but I will have difficulty interfacing the Xbee with the Arduino Leonardo. I believe it treats the Xbee as a serial output, so I could just do Serial1.write() but I'm unsure about that.
2. *Building Wireless Sensor Networks* also contains a section about working with XTCU, so I should be able to learn the relevant information from reading the book, otherwise I can check the Digi site for their documentation and go from there.
3. There are online resources I can use to easily breadboard an ultrasonic sensor to connect to the Arduino Leonardo and then once I can gather the ultrasonic data, then output it to the Xbee using Serial1.write() (hopefully).

## TIMELINE FOR PROJECT

- Build a point-to-point network between two Xbees
- Expand the point-to-point to contain 3 Xbees in a mesh network
- Successfully send a string of text between the built network
- Connect an ultrasonic sensor to the Arduino and send sensor values to another Arduino
- Log sensor values (either in Serial Monitor or connected LCD screen) on the receiving Arduino.
- Integrate Xbee modules with the Learning Robots
- Demonstration of all primary goals
- Determine if second secondary goal is achievable
- Demonstrate Proof of Concept (either with the Arduinos sending the ultrasonic data or the secondary goals)

# Metric 3- Budget Planning, Proposal, and Report

## DETAILED PARTS LIST

- Xbee Series 2 Modules
- DFRobot Xbee-USB Adaptors
- HC-SR04 Ultrasonic Sensor
- Arduino Leonardo/Mega with Xbee Socket
- Dell Venue Tablet
- Solderless Breadboards and Male-Male wires
- I/O Expansion Shield V7.1
- USB-to-MiniUSB Cables
- Pre-made robots with motors connected to Arduino Board

## COST OF PARTS, ITEMIZED

| Item | Quantity | Unit Cost/Total Cost |
|---|---|---|
| XBee 2mW PCB Antenna - Series 2 | 3 | $25.95/$77.85 |
| DFRobot Leonardo with Xbee Socket (Arduino Compatible) | 3 | $19.90/$59.70 |
| DFRobot Xbee-USB Adaptors | 3 | $29.99/$89.97 |
| Ultrasonic Sensor - HC-SR04 | 3 | $3.95/$11.85 |
| DFRobot I/O Expansion Shield V 7.1 | 3 | $29.99/$89.97 |
| Dell Venue Tablet/Laptop | 1 | $145/$145 |
| Breadboard Classic | 3 | $9.95/$29.85 |
| Total | | $504.19 |

## MONEY SOURCE PLANNING

Most of these resources, especially the Arduino Leonardo and Mega boards, were available in class. At the start of the project, there were 3 existing Series 2 Xbees and more modules and USB Adaptors will be purchased throughout the course of the project. The tablet and laptops were already part of the student or classroom as was all the breadboard materials. The Learning Robots used for the final demonstration were also pre-built and available in the classroom.

## BUDGET TIMELINE

The Xbee Series 2 Modules, USB Adaptors, Arduino Leonardo and Mega, Ultrasonic Sensor, and tablet/laptop were all available starting in mid-September. Additional Xbee radios and USB adaptors were ordered and delivered at the start of December. The

Ultrasonic sensor became part of the project in mid-November and the pre-existing Learning Robots will be used in the start of January.

## EXPENSE TRACKING

As 95% of the parts were already existing in the classroom, there is no expense tracking for them. The additional Xbee radios and USB adaptors ordered were from DFRobot and Sparkfun and were delivered at the start of December.

# Metric 4 – Daily Activities and Progress

## SEPTEMBER 22ND, 2016

### Goals

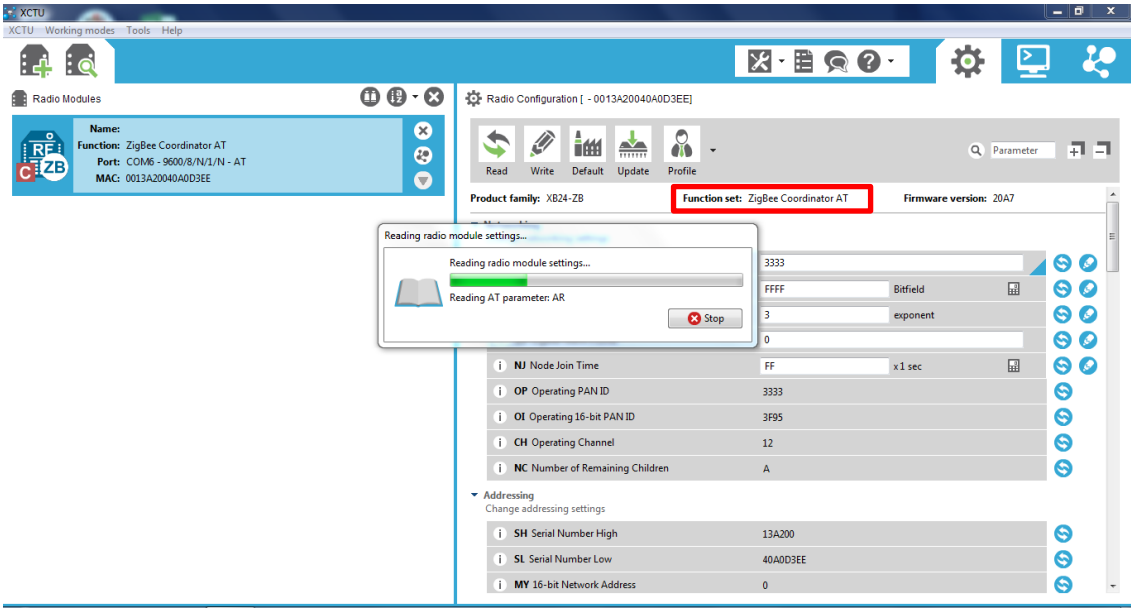
- Update the Xbee devices firmware using XTCU
- Successfully read the parameters from the device

### Accomplished Today

Today I used the Xbee to USB device to connect the Xbee to XTCU. Initially I was running into errors where it prompted me to reset the Xbee, however the DFRobot Xbee-to-USB did not come with a reset button. Disconnecting and reconnecting the Xbee seemed to have solved this problem however. One problem I ran into right off the start was that XTCU was not working with Windows 10. I had to use a Windows 7 laptop to connect XTCU with the Xbee. I believe this is a driver issue from Digi's end, but I will keep looking for other solution, as I cannot work on this from home then. I also successfully read the parameters and variables from an Xbee device and performed a full-reset so that I could reconfigure the PAN ID.

### Next Step

The next step will be to check what mode (AT or API) the Xbee are in, and then go from there. I believe I will initially work in AT mode, although it is just point-to-point. Establishing a point-to-point communication between two Xbees and sending text between the two devices will help me for when I have to repeat the process, except in API mode.




This is the initial configuration for the Xbee device. In this screenshot, I've connected the Xbee-to-USB to the Windows 7 Laptop and I'm reading the initial parameters for the device. The function set is to AT mode, for point-to-point.

**Daily Picture/Screenshot**

## SEPTEMBER 27TH, 2016

### Goals

- Configure the Xbee using Legacy XTCU
- Read *Building Wireless Sensor Networks* to figure out AT/API mode
- Obtain PAN ID and SH/SL and DH/DL from the Xbee module

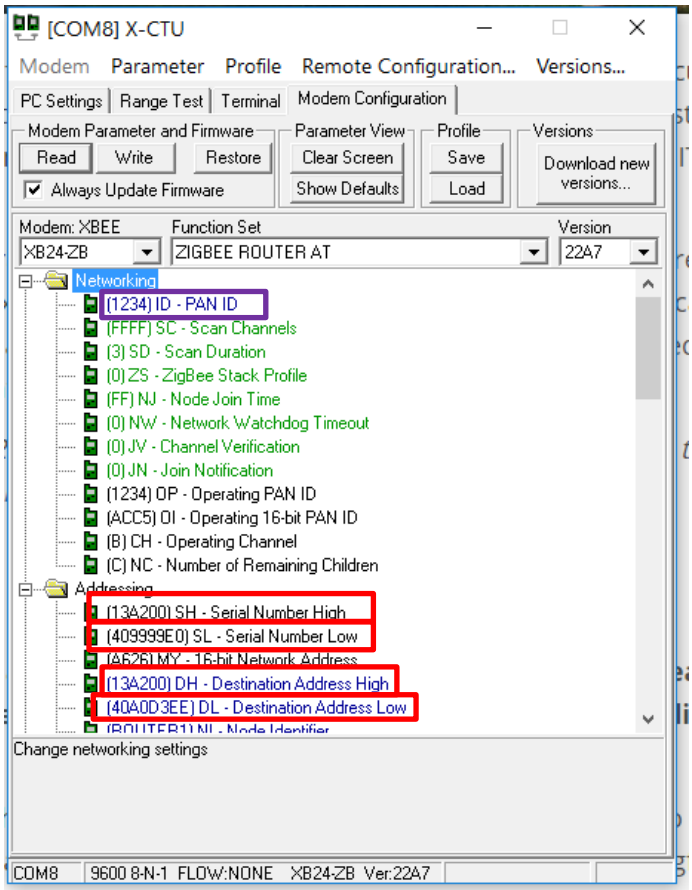
### Accomplished Today

Today I read up more about the difference between AT and API modes and how to create the network connection. As previously mentioned, AT mode is for point-to-point, between two Xbees. Both Xbees have to have the same PAN ID, and you must record each individual Xbee's Serial High and Serial Low, the values for which they can receive data from. On the second Xbee, you have to put the Destination High as the first Xbee's Serial High, and the Destination Low as the first Xbee's Serial Low. Then repeat this process for the other Xbee, so that the Serial High of one Xbee is the Destination High of the other and Serial Low of one Xbee is the Destination Low for the other. This will create the AT/point-point network and then you can use the built-in terminal to send messages. I also figured out that if I use the XTCU Legacy version, with a minimalistic GUI but same features, then I can get it to work on Windows 10, so that's what I'll be using in all my screenshots now.

### Next Step

The next step is to research how to interface the Xbee module with the Arduino Leonardo board. If I had an Sparkfun Xbee shield for Arduino, then it would be as simple as just mounting the shield on the Arduino. The Arduino Leonardo cannot be used for interfacing the Xbee with XTCU, as XTCU cannot pick up the Xbee module within the Arduino. I believe there might be a library for Xbee that I could download and use for Arduino.

## Daily Picture/Screenshot




This is the screenshot for the legacy XTCU. As you can see, it's not as aesthetically-pleasing as the previous screenshot, but it still contains all the features and parameters for the Xbee device. You can see the PAN ID, Serial Number High, Serial Number Low, and Destination High/Low are all parameters within the Xbee. Calibrating these values between the devices is important in building the network.

## Useful Links (not part of Daily Journal but convenient Location)

https://www.dfrobot.com/wiki/index.php/DFRobot_Leonardo_with_Xbee_R3(SKU:DFR0221) – Arduino Leonardo

Xbee Arduino Library- https://github.com/andrewrapp/xbee-arduino

https://www.youtube.com/watch?v=YloKoSebqjE – S2 API Mode

## SEPTEMBER 29TH, 2016

### Goals

- Explore Xbee libraries for Arduino
- Figure out how to output data to Xbee from Arduino

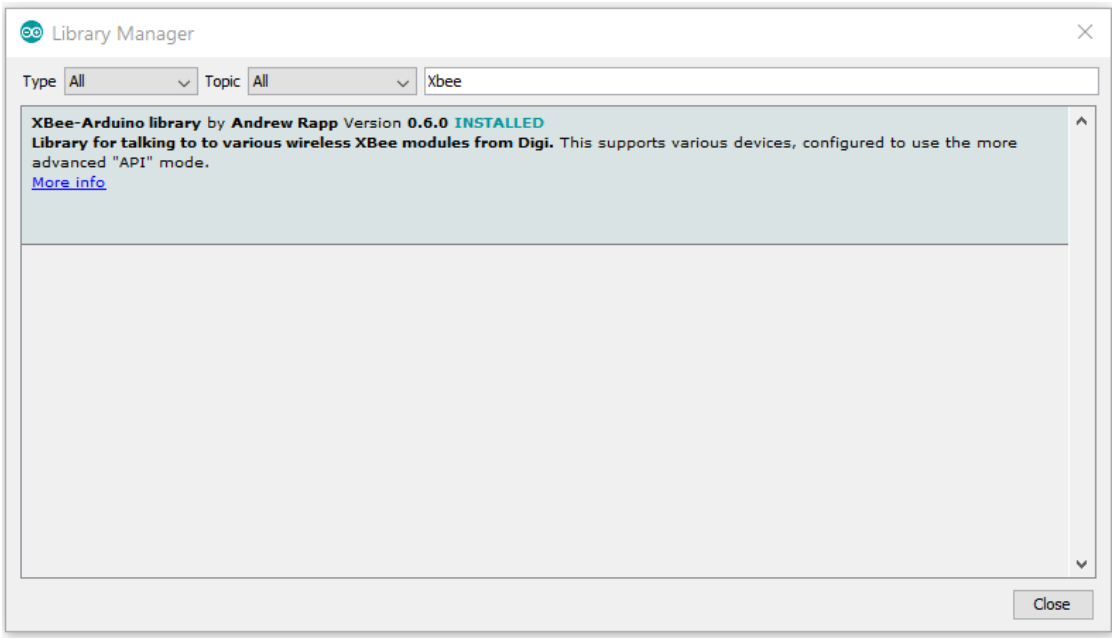
### Accomplished Today

Today I did some research on interfacing Xbee with Arduino. There are two potential possible ways to output data to the Xbee from the Arduino, both of which I will have to test. The first method I found was an Arduino Library designed for Xbees for API mode, which would take care of the mesh network requirement. The documentation for this library is on GitHub, so I would have to look into that to figure out the syntax. The second method I found was from the DFRobot Arduino Leonardo with Xbee socket Wiki. It included a screenshot of Arduino code where the code just used the Xbee as Serial1, but this is unverified code as of yet.

### Next Step

The next step will be to create the point-to-point network and then send a string of text from one computer to another using the Xbee and XTCU terminal. This will give me a good starting point to transition over to the Mesh network, as AT and API have similar commands and parameters.

### Daily Picture/Screenshot




This shows the Xbee-Arduino Library that I will further investigate through the documentation on the GitHub(Link to GitHub is above this page)

sketch_oct07ab §

```
// # Description:
// It is just used to test Leonardo module
// Input  "p" once time the led will be turn on;twice to turn off.
// It is controled by Serial1 Xbee side only. but you can check the status from serial port.

int ledPin=13;
int val;
int count=0;
void setup()
{
  pinMode(ledPin,OUTPUT);
  Serial.begin(9600);
  Serial1.begin(115200);  // Xbee Serial1 port baud rate should be set the same to the xbee baud rate
}
void loop()
{
  val=Serial1.read();    // read xbee Serial1 data
  if(-1!=val)           // "-1" means no data
  {

    if('p'==val){
      if(count==0)count=1;
      else if(count==1)count=0;
    }


    if(count==0){
      digitalWrite(ledPin,HIGH);  // Turn on
      Serial.println("LED ON");  // you can check the status from serial port.
      Serial1.println("LED ON"); //you can check the status from serial1 port(xbee).
    }
    else if(count==1)
    {
      digitalWrite(ledPin,LOW);    // Turn off
      Serial.println("LED OFF");  //you can check the status from serial port.
      Serial1.println("LED OFF"); //you can check the status from serial1 port(xbee).
    }
  }
}
```

This shows the snippet of code that provided from the DFRobot Wiki, showing how the user sets the Xbee as Serial1, initiates it with the Baud rate determined from XTCU, and can then read and print to the Xbee with the Serial1.read and Serial1.print commands.

# END OF 1st NINE WEEKS

## OCTOBER 18TH, 2016

**Goals:**

- Interface two Xbees for a AT network
- Communicate from one XTCU Terminal to the other

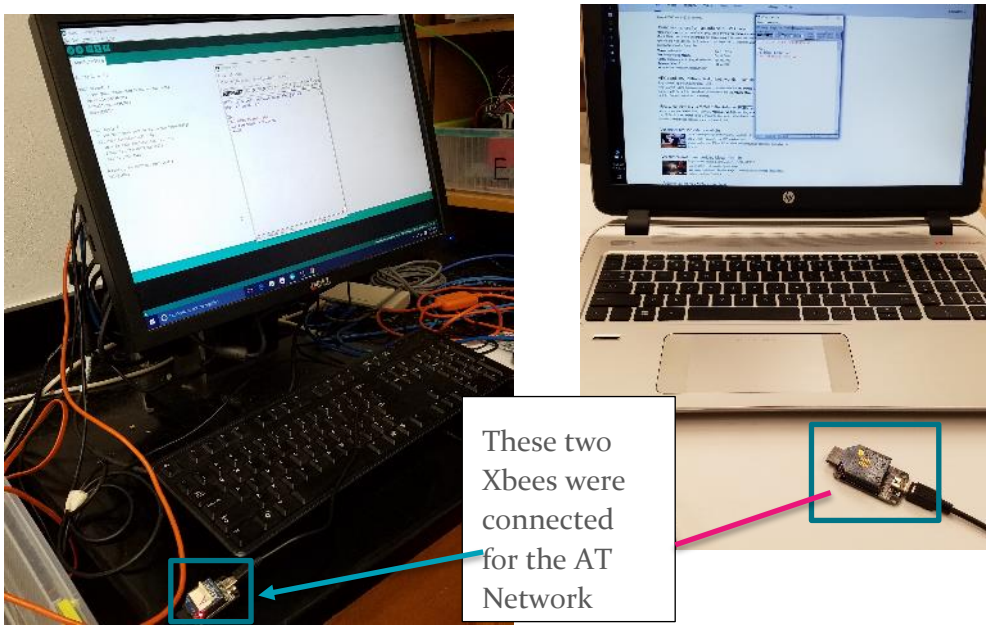
**Accomplished Today:**

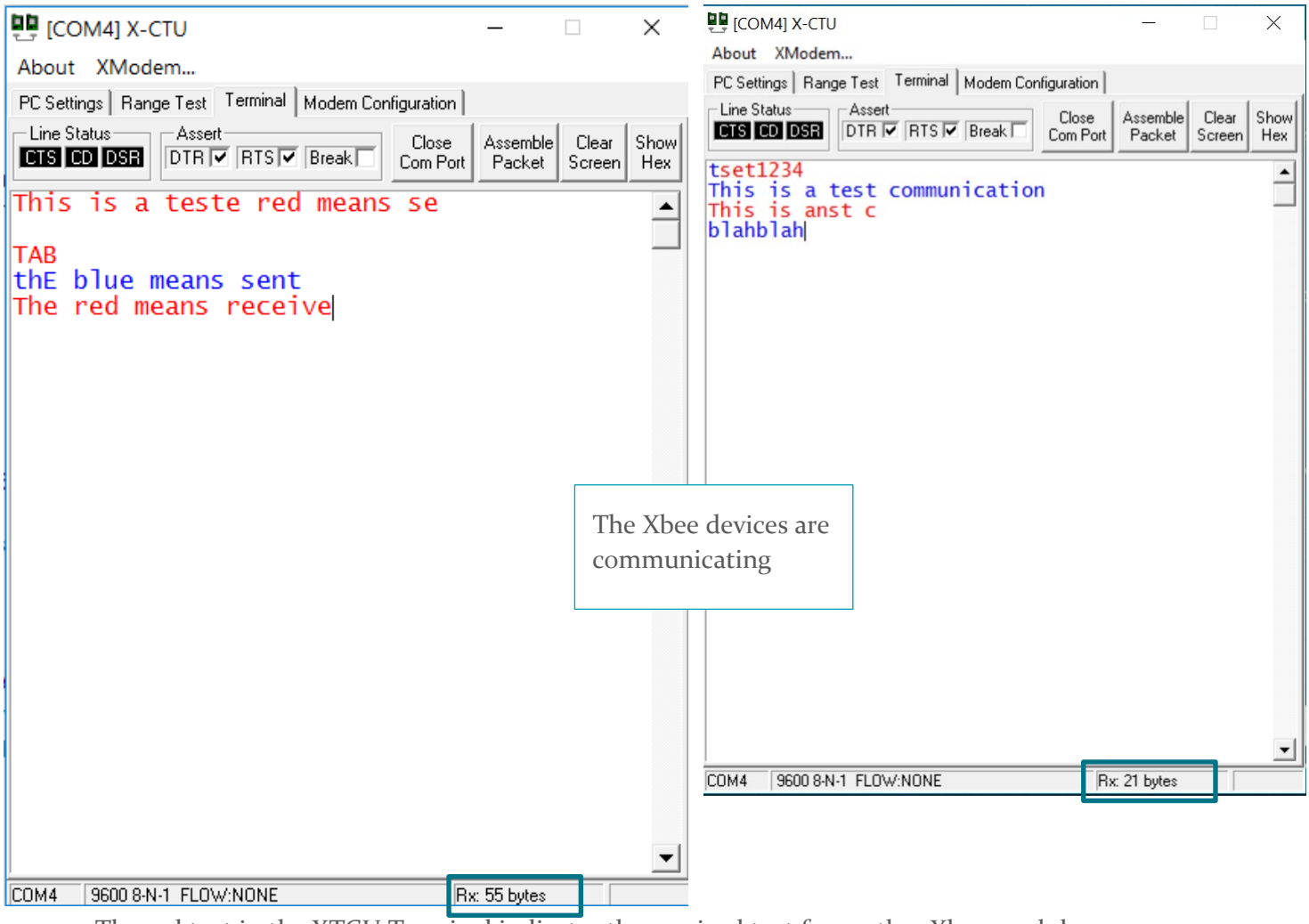
Today, I was able to successfully connect two Xbee devices into an AT Network. Despite my end goal being API mode, it will be helpful for me to learn the AT Commands and how to output data from Xbee to Arduino using AT mode. Once I've figured it out in AT mode, I should have an easier time converting it into API Mode. To connect it into AT Mode, I had to set the two PAN IDs to the same address (7779) and match the Serial Low with the Destination Low and the Serial High with the Destination High for both Xbees. This step is crucial in the pairing process, as it determines what ranges the Xbees can detect the messages and convert it into text. After successful connection, I opened the Terminal for both instances of XTCU and could send text messages between the two Xbees as well as view the Tx and Rx information. After sending a few test messages, I moved onto researching interfacing the Xbee with Arduino through Serial1 on the Arduino Leonardo.

**Next Step:**

The next step in the process is to transfer one Xbee device from the Xbee-USB receiver to the Xbee socket on the Leonardo and send/receive messages from the network. XTCU does not recognize the Arduino Leonardo as a viable Xbee-USB interface, therefore I will have to print the incoming values through the Arduino Serial Monitor. I still have to figure out the difference between the previously found Xbee Arduino Library and using the Xbee as a Serial1 device.

**Daily Picture/Screenshot**






The red text in the XTCU Terminal indicates the received text from other Xbee modules within the network. Since this was only point-to-point, that means there were only two Xbee devices connected. The blue text represents the text that is sent from the specific XTCU terminal. The pictures above show two instances of a AT network between 2 Xbee radios.

## OCTOBER 24TH, 2016

### Goals

- Interface Xbee AT Network with Arduino
- Communicate between Arduino Serial Monitor and XTCU Terminal

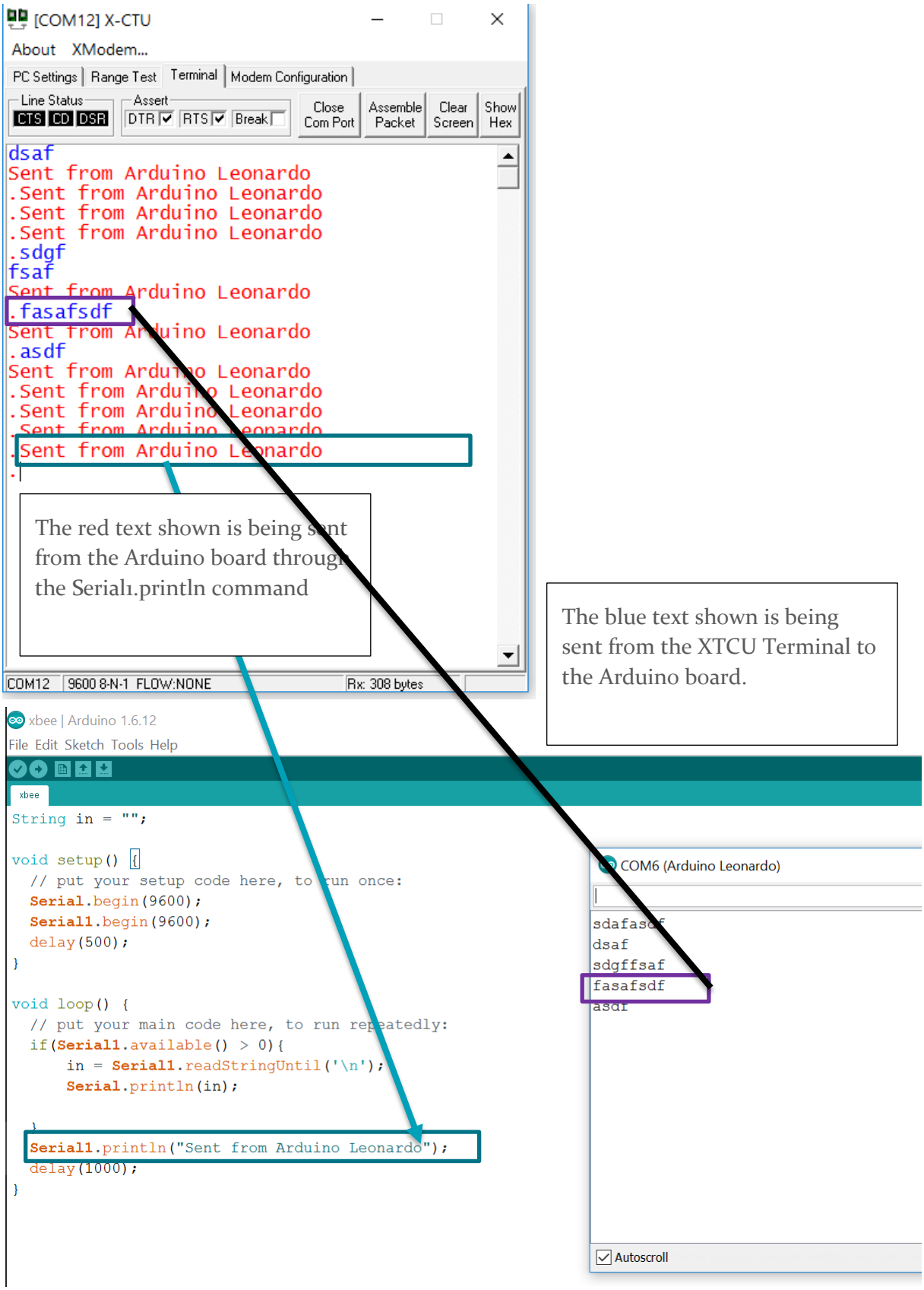
### Accomplished Today

After exploring the GitHub library previously discovered for the Xbee/Arduino interface, I realized that library was intended for users who did not have an Arduino with a Xbee socket or are using API Mode. However, I already had a few Arduino Leonardo boards and Mega Sensor Shields with Xbee socket and I'm currently working with AT commands. Therefore, I followed the Serial and Serial1 path as suggested by the Arduino Leonardo Wiki. I declared my Serial Monitor as Serial and the Xbee as Serial1 and surely enough, Serial1.println resulted in the Xbee printing text to XTCU while Seria1.readStringUntil resulted in the Xbee reading from the XTCU Terminal. However, this is still an intermediate step as my end goal is to communicate through Arduino in API Mode. For this, I do not think it will be as easy as using Serial1.print and Serial1.read to transmit/receive data.

### Next Step:

The next step is to transfer the second Xbee device from the Xbee-USB device to the second Arduino Leonardo. This way, I'll create an Arduino to Arduino network using Xbee and send/receive text using the Serial Monitor. In addition, I've been noticing that there are uncommon yet present instances where the text received will be different from the text transmitted and often one or two letters are misplaced. I believe this is a problem with the SH/SL and DH/DL commands and would like to use the XTCU Terminal to obtain the Xbee information.

### Daily Picture/Screenshot:


[COM12] X-CTU
About XModem...
PC Settings | Range Test | Terminal | Modem Configuration
Line Status
CTS CD DSR
Assert
DTR RTS Break
Close Com Port
Assemble Packet
Clear Screen
Show Hex
dsaf
Sent from Arduino Leonardo
.Sent from Arduino Leonardo
.Sent from Arduino Leonardo
.Sent from Arduino Leonardo
.sdgf
fsaf
Sent from Arduino Leonardo
.fasafsdf
Sent from Arduino Leonardo
.asdf
Sent from Arduino Leonardo
.Sent from Arduino Leonardo
.Sent from Arduino Leonardo
.Sent from Arduino Leonardo
.Sent from Arduino Leonardo
.
COM12 9600 8-N-1 FLOW:NONE Rx: 308 bytes
The red text shown is being sent from the Arduino board through the Serial1.println command
The blue text shown is being sent from the XTCU Terminal to the Arduino board.
xbee | Arduino 1.6.12
File Edit Sketch Tools Help
xbee
String in = "";
void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
Serial1.begin(9600);
delay(500);
}
void loop() {
// put your main code here, to run repeatedly:
if(Serial1.available() > 0){
in = Serial1.readStringUntil('\n');
Serial.println(in);
}
Serial1.println("Sent from Arduino Leonardo");
delay(1000);
}
COM6 (Arduino Leonardo)
sdafasdf
dsaf
sdgffsaf
fasafsdf
asdf
Autoscroll

## NOVEMBER 8TH, 2016

### Goals

- Interface an Arduino to Arduino Xbee AT Network
- Use AT Commands to extract the Serial High, Serial Low, Destination High, and Destination Low of a Xbee

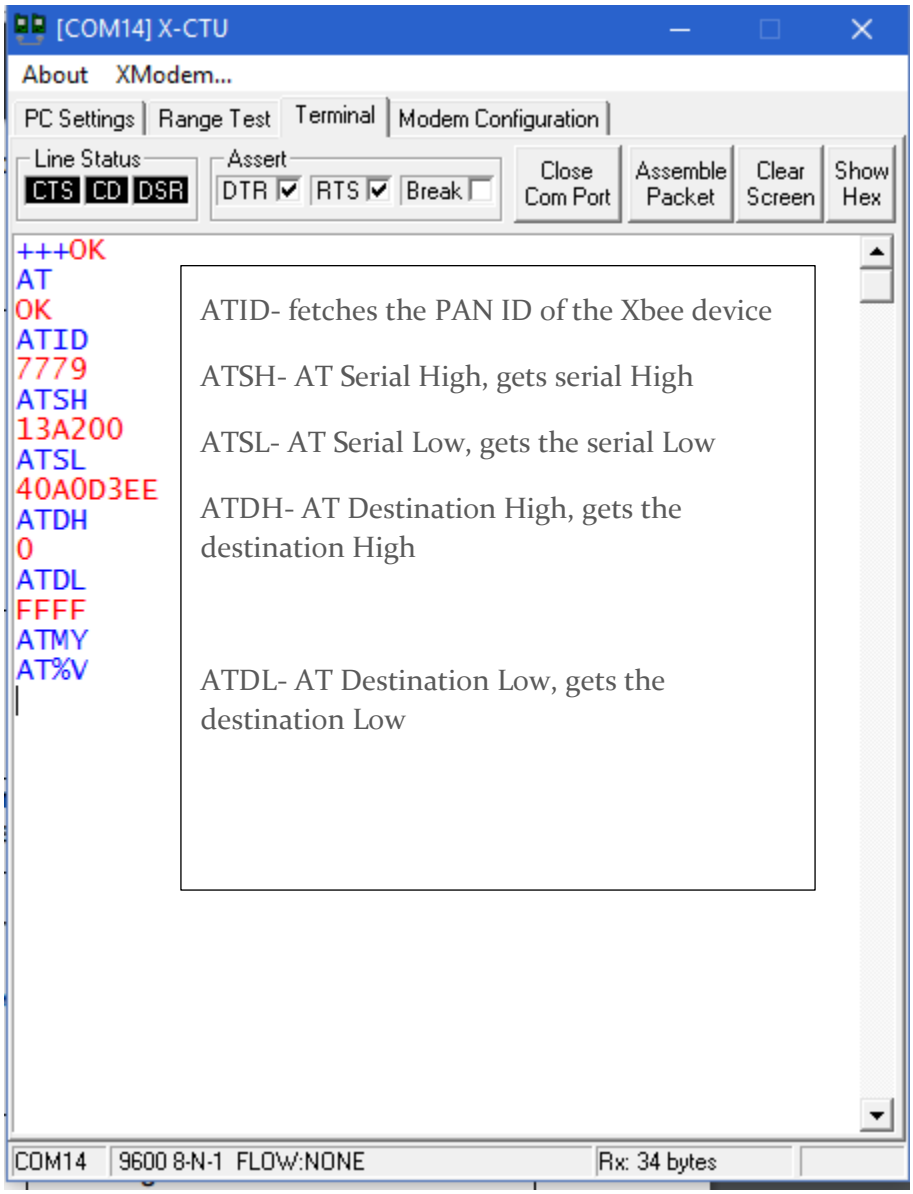
### Accomplished Today:

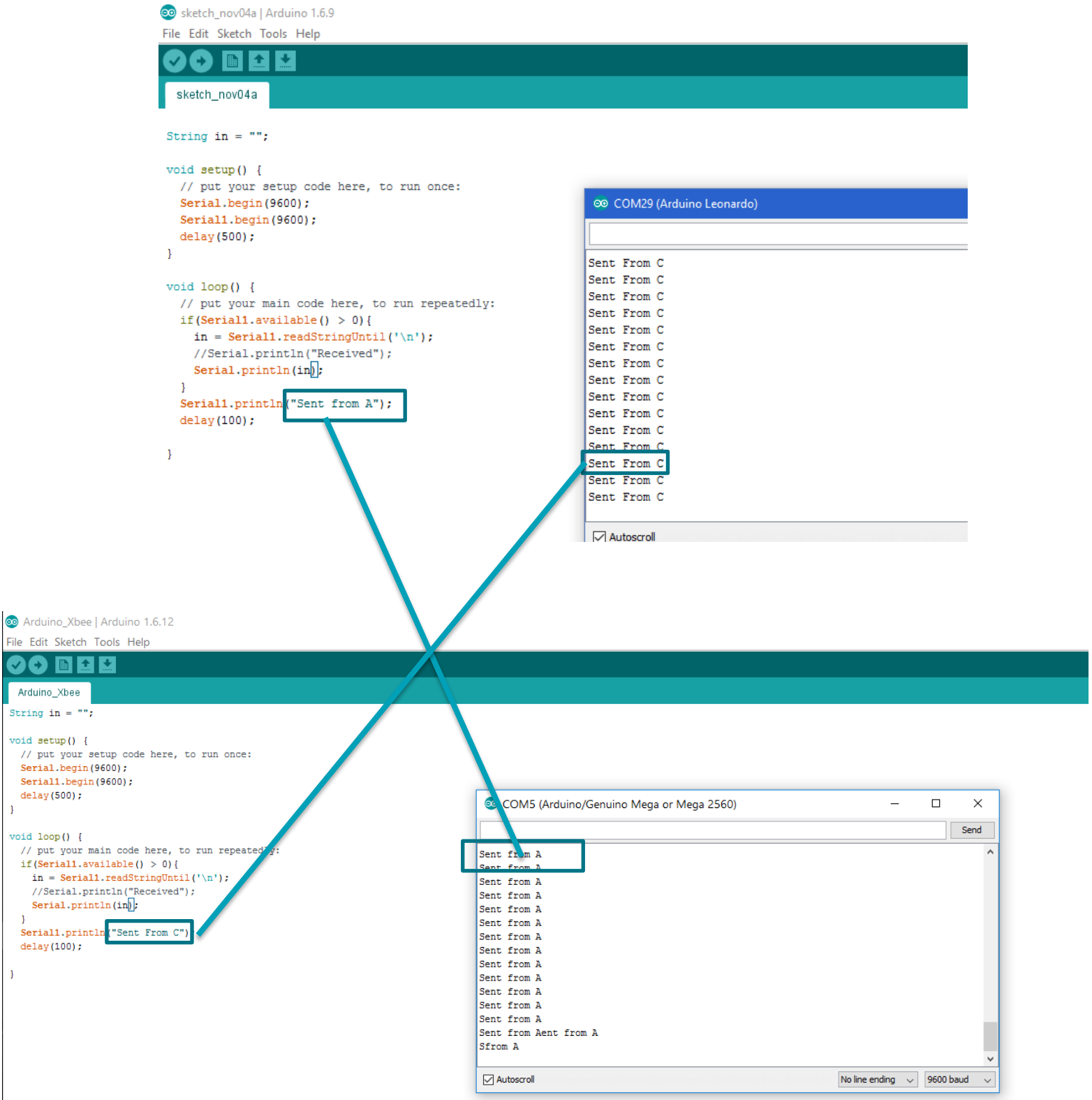
After having successfully interfaced an Arduino Leonardo with the XTCU terminal, I expanded so that two Leonardo boards have Xbees now and can communicate through the serial monitor. However, through the communication, I noticed that some of the incoming data was out of order. Some of the incoming characters were shifted, so I used the AT Commands to obtain the Serial High, Serial Low, Destination High, and Destination Low of the two Xbee devices. Ideally, one would want the Xbee A's SL and SH to match with Xbee B's DL and DH. Additionally, Xbee B's SL and SH would match Xbee A's DH and DL. Using these AT commands, I was able to easily extract this information and set the appropriate Serial Lows and Highs as well as Destination Lows and Highs.

### Next Step:

Now that I have been able to successfully send strings across the Arduinos with the Xbee, I would like to move to sending and manipulating sensor data. My end goal is to interact with the pre-built Learning Bots and coordinate their movements with only one ultrasonic sensor. Therefore, I should interface an Ultrasonic sensor with the Mega Shield and then send the data to the Leonardo board.

### Daily Picture/ Screenshot:


[COM14] X-CTU
About
XModem...
PC Settings
Range Test
Terminal
Modem Configuration
Line Status
CTS
CD
DSR
Assert
DTR
RTS
Break
Close Com Port
Assemble Packet
Clear Screen
Show Hex
+++OK
AT
OK
ATID
7779
ATSH
13A200
ATSL
40A0D3EE
ATDH
0
ATDL
FFFF
ATMY
AT%V
ATID- fetches the PAN ID of the Xbee device
ATSH- AT Serial High, gets serial High
ATSL- AT Serial Low, gets the serial Low
ATDH- AT Destination High, gets the destination High
ATDL- AT Destination Low, gets the destination Low
COM14
9600 8-N-1 FLOW:NONE
Rx: 34 bytes


sketch_nov04a | Arduino 1.6.9
File Edit Sketch Tools Help
sketch_nov04a
String in = "";
void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
Serial1.begin(9600);
delay(500);
}
void loop() {
// put your main code here, to run repeatedly:
if(Serial1.available() > 0){
in = Serial1.readStringUntil('\n');
//Serial.println("Received");
Serial.println(in);
}
Serial1.println("Sent from A");
delay(100);
}
COM29 (Arduino Leonardo)
Sent From C
Sent From C
Sent From C
Sent From C
Sent From C
Sent From C
Sent From C
Sent From C
Sent From C
Sent From C
Sent From C
Sent From C
Sent From C
Sent From C
Autoscroll
Arduino_Xbee | Arduino 1.6.12
File Edit Sketch Tools Help
Arduino_Xbee
String in = "";
void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
Serial1.begin(9600);
delay(500);
}
void loop() {
// put your main code here, to run repeatedly:
if(Serial1.available() > 0){
in = Serial1.readStringUntil('\n');
//Serial.println("Received");
Serial.println(in);
}
Serial1.println("Sent From C");
delay(100);
}
COM5 (Arduino/Genuino Mega or Mega 2560)
Send
Sent from A
Sent from A
Sent from A
Sent from A
Sent from A
Sent from A
Sent from A
Sent from A
Sent from A
Sent from A
Sent from A
Sent from A
Sent from A
Sent from Aent from A
Sfrom A
Autoscroll
No line ending
9600 baud

## NOVEMBER 10TH, 2016

### Goals

- Breadboard a HC-SR04 Ultrasonic sensor to an Arduino Mega with Sensor Shield
- Move the Xbee from the Arduino Leonardo to the Mega Sensor Shied

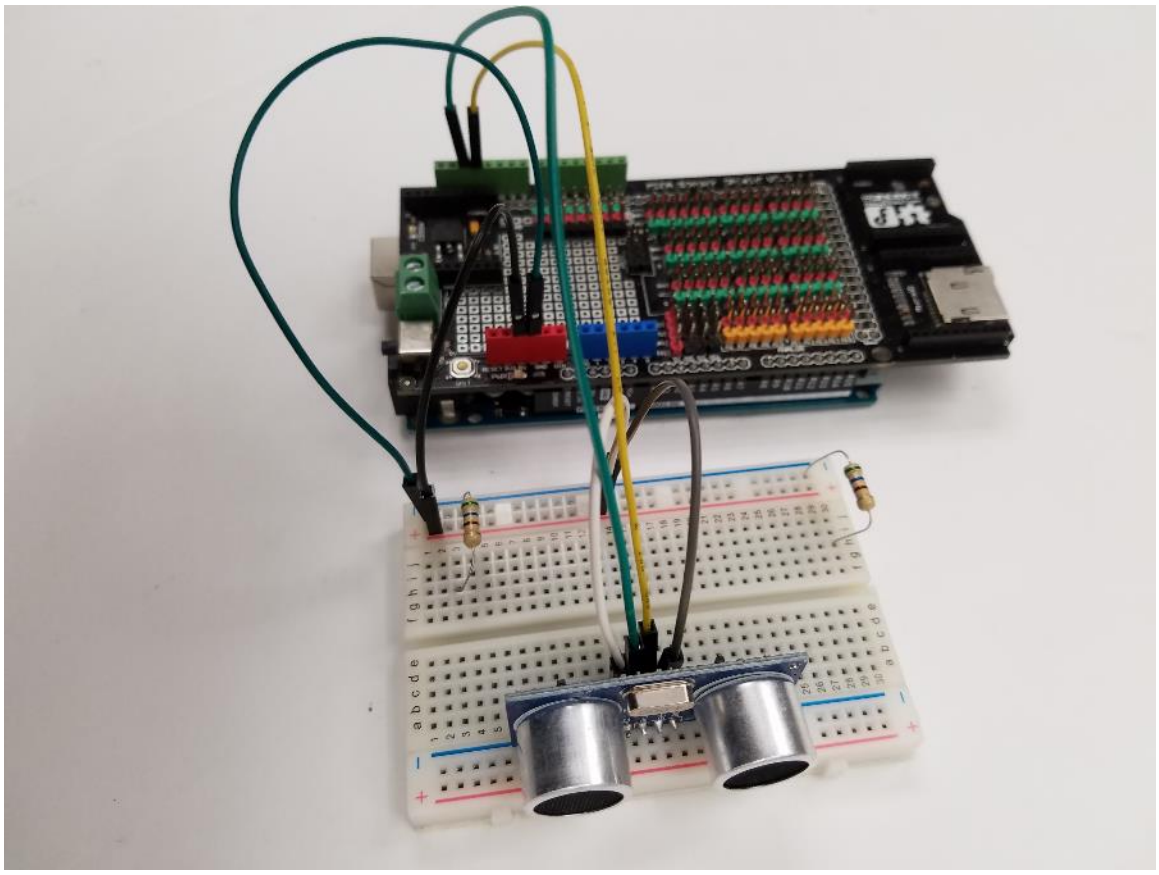
### Accomplished Today:

I was able to breadboard out an HC-SRO4 Ultrasonic sensor into an Arduino Mega Sensor Shield. Initially, I believed I could just directly attach the sensor into the shield but then I realized bread-boarding would be required. This was accomplished using 5 male-to-male wires, a solderless breadboard, and the ultrasonic sensor. While initially I attached LEDs to visually display the incoming data, I ultimately ended up removing them so that there wouldn't be two extra wires and for simplicity. After the Mega Sensor Shield was set up with the bread-board and ultrasonic sensor, I moved one of the Xbees from the Leonardo to Mega shield. This way, the Mega can process the sensor data and then output it through Serial1.

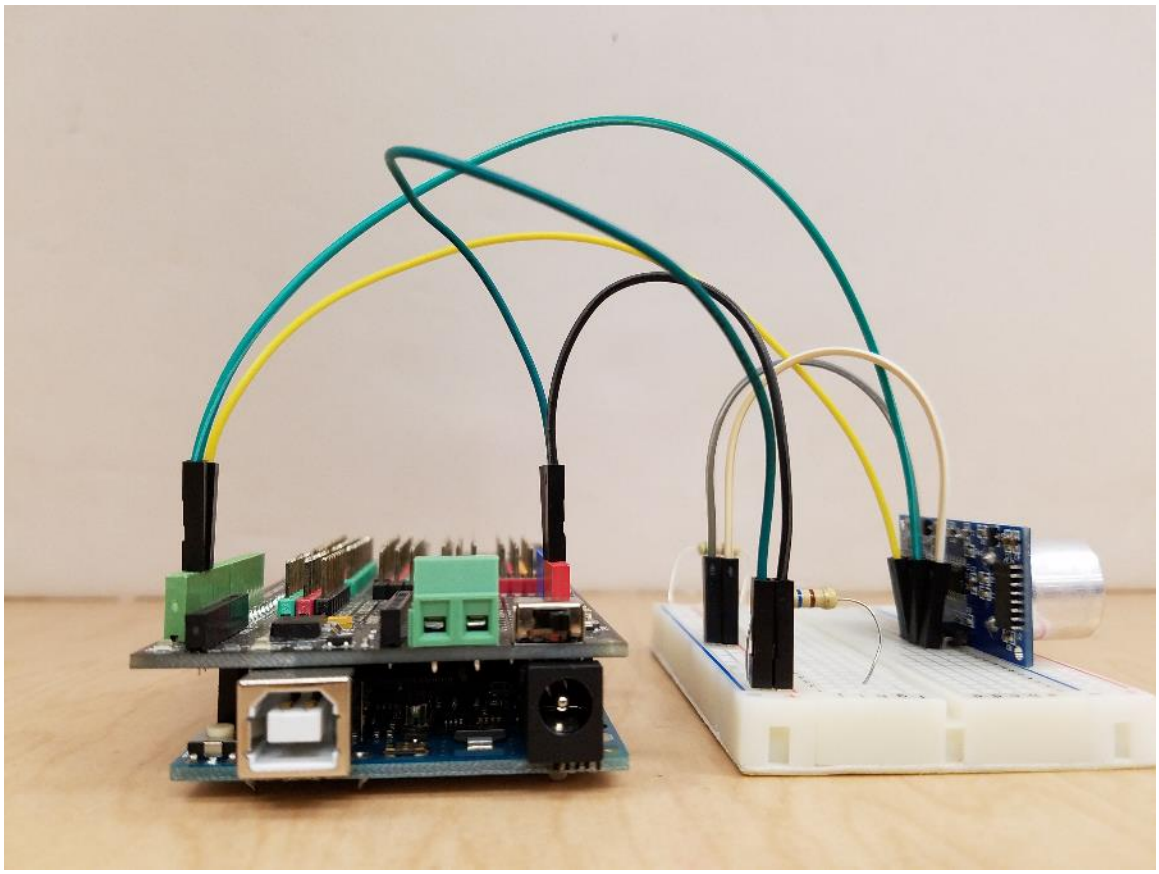
### Next Step:

After the hardware components of the sensor are finished, I would like to work on the code to process distance in centimeters and printing it to the Serial Monitor. This will allow me to verify the echo and trig pins are correctly bread-boarded and will eventually allow for the data to be sent through Xbees.

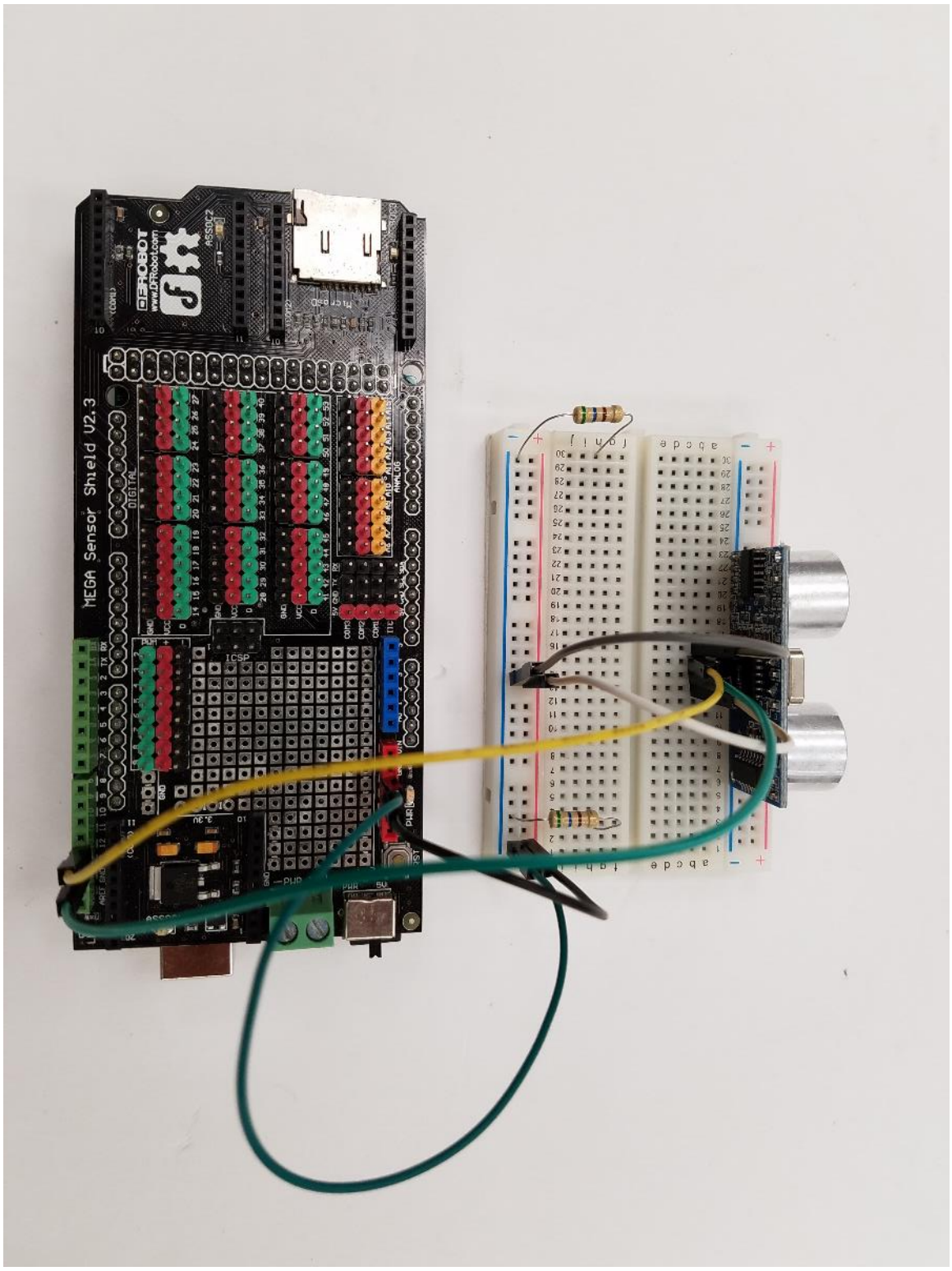
### Daily Picture/Screenshot:

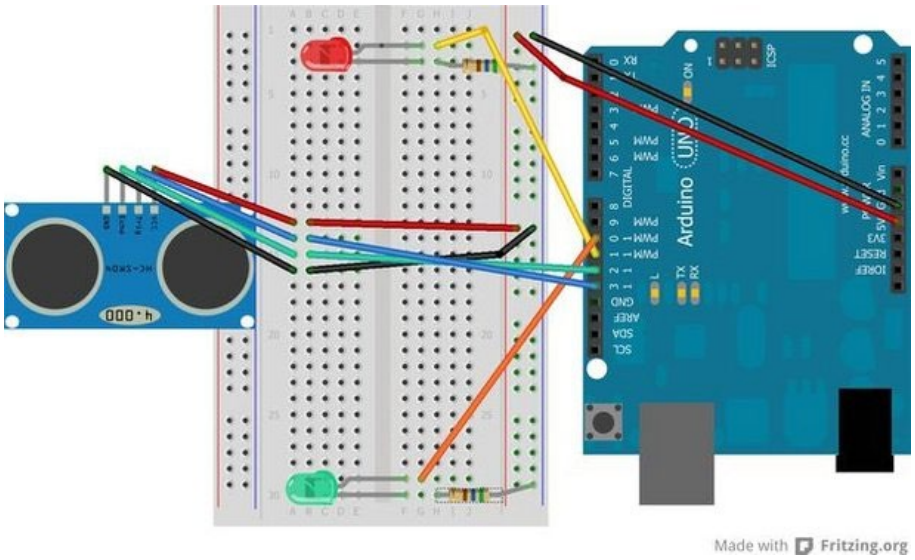


Top View



Side View


MEGA Sensor Shield V2.3
DFROBOT
www.DFRobot.com
DIGITAL
ANALOG
ICSP
PWR
5V




Although not for the Mega specifically, this pin-out diagram helped me with the bread-boarding so I thought to include it.

## NOVEMBER 14TH, 2016

### Goal:

- Gather data from the Ultrasonic sensor in Arduino

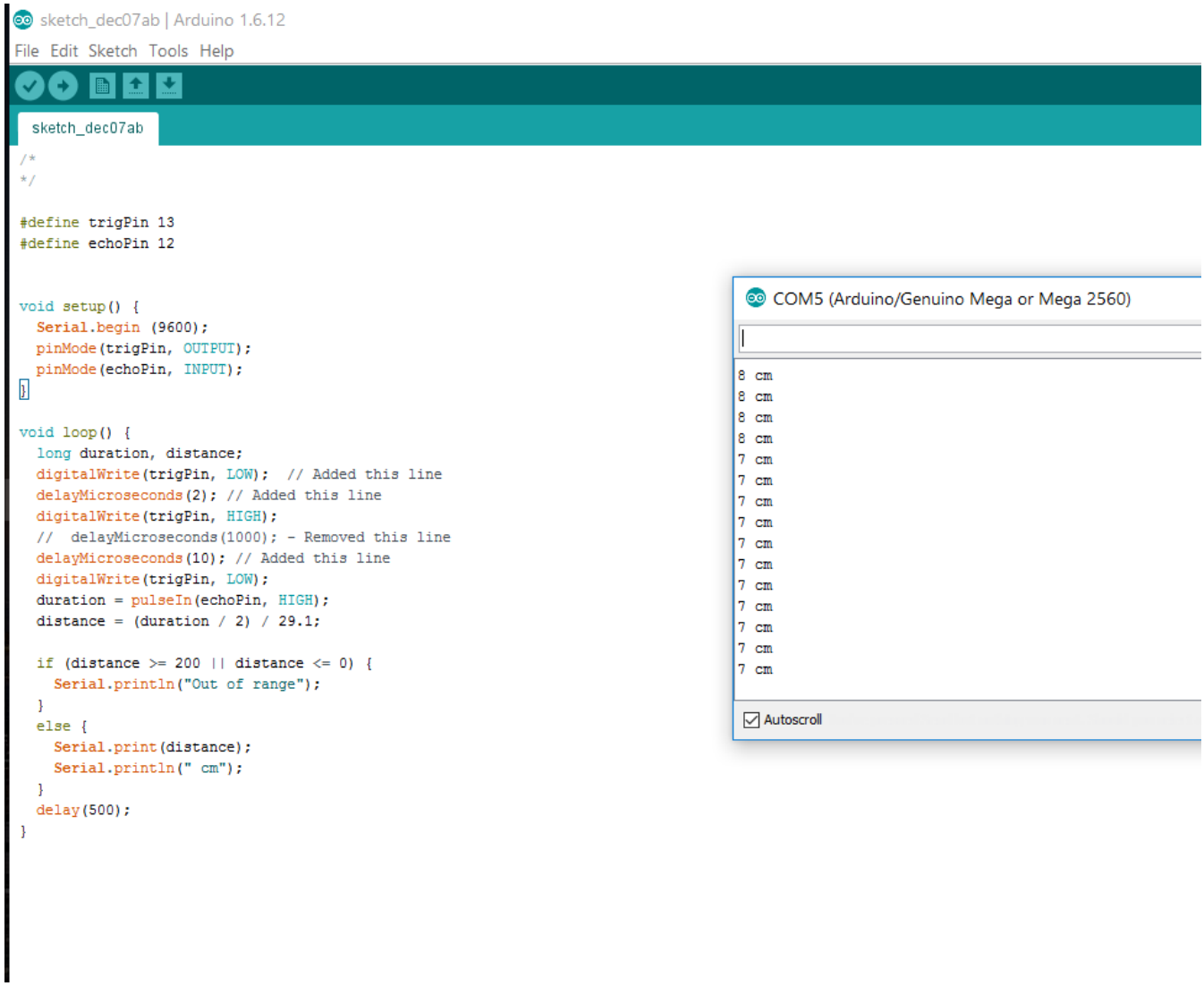
### Accomplished Today:

Today I implemented the ultrasonic sensor code that will display distance in centimeters. I needed help with programmatically setting up the trigPin and echoPin so I asked a classmate who is also working with this sensor to help out. Even after setting up the trigPin, I couldn't figure out how to convert the data I was receiving into centimeter values. Ultimately, the Arduino forum helped show me that I would have to use the pulseIn method and divide by 29.1. I was able to print the centimeter values out to the Serial Monitor as well. I also labeled the Xbee module on the Mega sensor shield "Xbee A" and the module on the Arduino Leonardo "Xbee B".

### Next Step:

Now that I can successfully print to the Serial Monitor, I also should be able to print to Serial1 and send the data to the Arduino Leonardo with Xbee B. After I parse in the incoming serial string, I can use the distance as determinants for activities performed on the Leonardo, similar to how I will implement only one ultrasonic sensor with the Learning Robots.

## Daily Screenshot/Picture:



## NOVEMBER 18TH, 2016

### Goals:

- Transmit Ultrasonic data from Mega to Leonardo using Xbee
- Use the incoming data within an if statement and make decisions based off incoming data

### Accomplished Today:

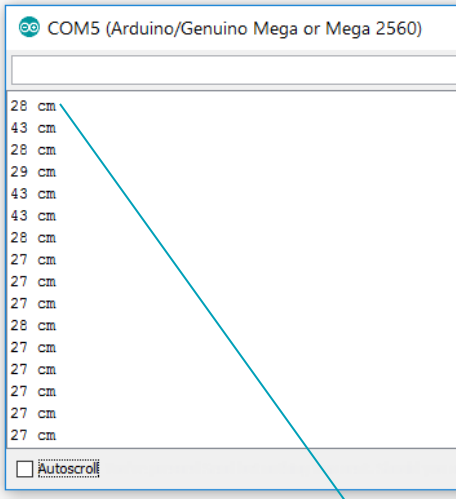
I was able to successfully send the ultrasonic data from Xbee A and receive on Xbee B using Serial1.println. Initially, I set the delay to be 100, so it would send the data every tenth of a second. However, I found that there was a tradeoff between accuracy and time. The quicker I sent the data, the more likely the incoming serial data on Xbee B would be out of order or skipped. I found that setting a delay of 5 seconds between each data reading and transmission was optimal for both time and accuracy. I did run into a problem when I tried to compare the incoming string to another string using an if statement. After doing research, I discovered that serial data sent through Xbee cannot easily be compared to as strings (to determine if two strings match) and will have to find another way.

### Next Step:

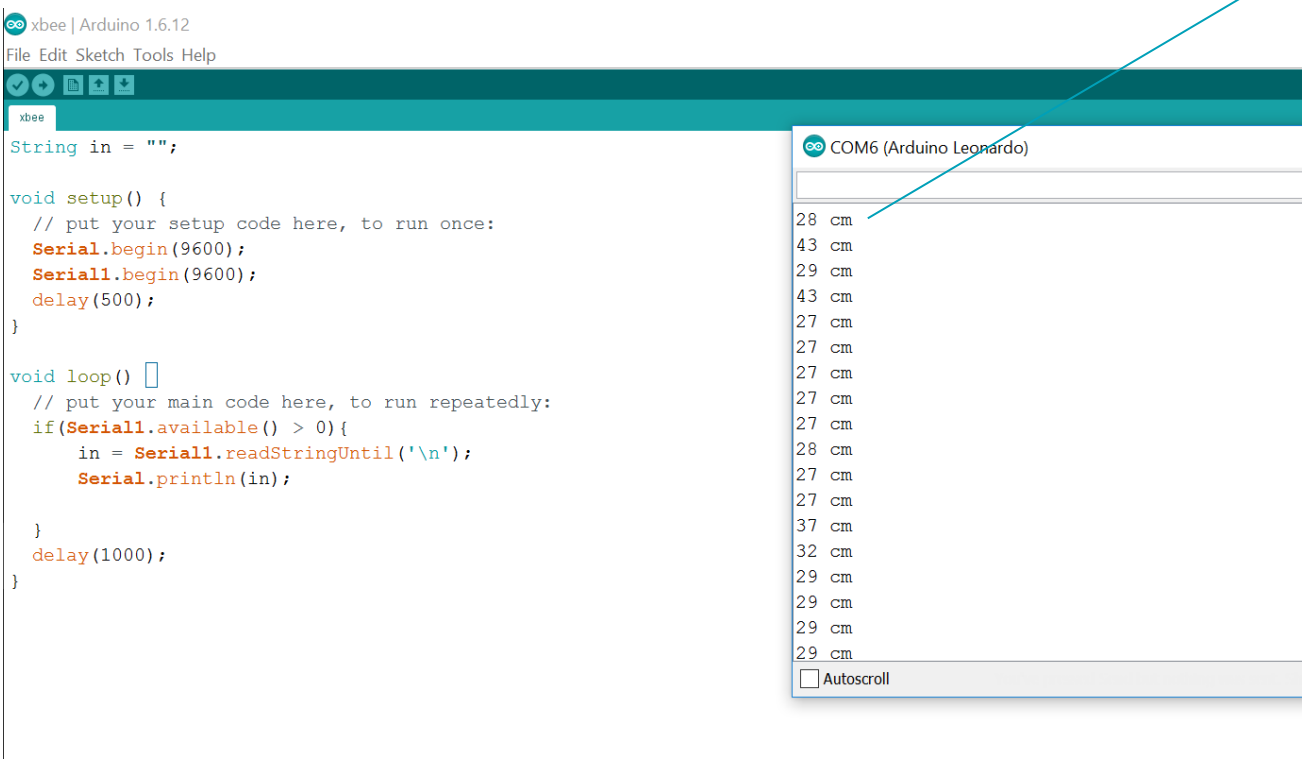
Find a way to compare the incoming string value to other strings. I also want to use the incoming distance data to influence some aspect of the Leonardo, such as the internal LED. That way I can demonstrate that sending data through Xbee can allow for decision statements to be executed efficiently on the receiving end.

### Daily Screenshot/ Picture:

```
void setup() {
  Serial.begin (9600);
  Serial1.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
    delay(500);
}

void loop() {
  long duration, distance;
  digitalWrite(trigPin, LOW);  // Added this line
  delayMicroseconds(2); // Added this line
  digitalWrite(trigPin, HIGH);
  //  delayMicroseconds(1000); - Removed this line
  delayMicroseconds(10); // Added this line
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration / 2) / 29.1;

  if (distance >= 200 || distance <= 0) {
    Serial.println("Out of range");
  }
  else {

    Serial.print(distance);
    Serial.println(" cm");
    Serial1.print(distance);
    Serial1.println(" cm");

  }
  delay(500);
}
```




Notice how the data values are matching and being sent from the Mega to the Leonardo.



## NOVEMBER 29TH, 2016

**Goals:**

- Correct the serial string error discovered last time
- Expand upon the previous day's progress and use the incoming ultrasonic data to manipulate the internal Leonardo LED

**Accomplished Today:**

After looking through the Arduino Forum, I found that the easiest way to parse incoming serial data was to make a character array of the incoming data and then combine it to create a string at the end of each data transmission. I have yet to implement this yet, as I focused on just single character data packets for now. I also managed to set up an if statement so that if the ultrasonic sensor was less than 5 cm away from an object, that the internal LED light would turn on the Arduino Leonardo. This proves that incoming data-based loops and statements can be efficiently used on the receiving Arduinos.

**Next Step:**

Figuring out how to read Serial string data properly using character arrays. In addition, I also want to start working with API mode. The past few weeks I've been only doing point-to-point, which is easy and has been done numerous times before. The AT mode only served as an introduction into Xbee and I believe I'm ready to start working with the challenging API mode now.

**Daily Screenshot/Picture:**

This is the code from the Ultrasonic Mega Shield being sent to the Leonardo. You can see that it can either turn the LED on or off according to the character that is sent.

temporary

```
  pinMode(LED_BUILTIN, OUTPUT);
  delay(500);
}

void loop() {
  // put your main code here, to run repeatedly:
  while(Serial1.available() > 0){
      delay(1);
      char c = Serial1.read();
      if(c == 'O'){
        Serial.println("Turn LED on");
         digitalWrite(LED_BUILTIN, HIGH);
      }
      if(c == 'N'){
        Serial.println("Turn LED Off");
          digitalWrite(LED_BUILTIN, LOW);
      }
      if(c == ""){
        Serial.println("No data coming");
      }
      c = "";
      //in = Serial1.readStringUntil('\n');
   /*   if(){
          digitalWrite(LED_BUILTIN, LOW);
          Serial.println(" LED Off");
      }
```

COM6 (Arduino Leonardo)

```
Turn LED Off
Turn LED Off
Turn LED Off
Turn LED Off
Turn LED Off
Turn LED Off
Turn LED Off
Turn LED Off
Turn LED Off
Turn LED Off
Turn LED on
Turn LED on
Turn LED on
Turn LED Off
Turn LED Off
Turn LED Off
Turn LED Off
```

Autoscroll

sketch_dec07ab | Arduino 1.6.12

File Edit Sketch Tools Help

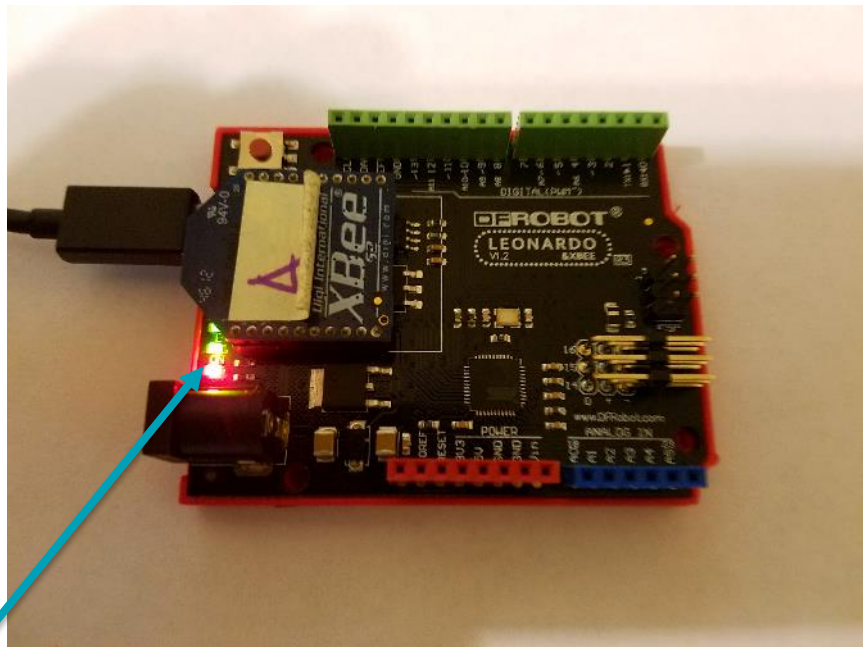
sketch_dec07ab

```
#define trigPin 13  //Defines Ultrasonic trigpin to 13
#define echoPin 12  //Defines Ultrasonic echoPin to 12
String in = "";

void setup() {
  Serial.begin (9600); //Serial Monitor
  Serial1.begin(9600); //Xbee
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
    delay(500);
}

void loop() {
  long duration, distance;
  digitalWrite(trigPin, LOW);  // Added this line
  delayMicroseconds(2); // Added this line
  digitalWrite(trigPin, HIGH);
  //  delayMicroseconds(1000); - Removed this line
  delayMicroseconds(10); // Added this line
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration / 2) / 29.1;

  if (distance >= 200 || distance <= 0) {
    Serial.println("Out of range");
  }
  else {
    if(distance >= 5){   // IF THE DISTANCE IS GREATER THAN 5 CM, LIGHT OFF
      Serial.print(distance); //prints cm value
      Serial.print(" cm ");
      Serial.println("Light off"); //indicates light should be off
      Serial1.println("N");       // Sends a value of "N" to Xbee on Leonardo
    }
    else{                //DISTANCE IS LESS THAN 5, LIGHT ON
        Serial.print(distance);   //prints distance
        Serial.print(" cm ");
        Serial.println("Light on"); //indicates that the Leonardo light should be on
        Serial1.println("O");      //Sends a value of "Y" to Xbee on Leonardo
    }

   /* Serial1.print(distance);
    Serial1.println(" cm");*/

  }
  delay(5000);
}
```

COM5 (Arduino/Genuino Mega or Mega 2560)

```
44 cm Light off
44 cm Light off
44 cm Light off
45 cm Light off
3 cm Light on
2 cm Light on
2 cm Light on
45 cm Light off
45 cm Light off
44 cm Light off
44 cm Light off
44 cm Light off
44 cm Light off
44 cm Light off
21 cm Light off
```

Autoscroll




This is the LED being turned off and on by the ultrasonic sensor.

## DECEMBER 5TH, 2016

### Goals:

- Convert from AT Mode into API Mode
- Obtain and set the Destination High, Destination Low for the Coordinator and Router
- Send a message from all 3 radios through the XTCU terminals

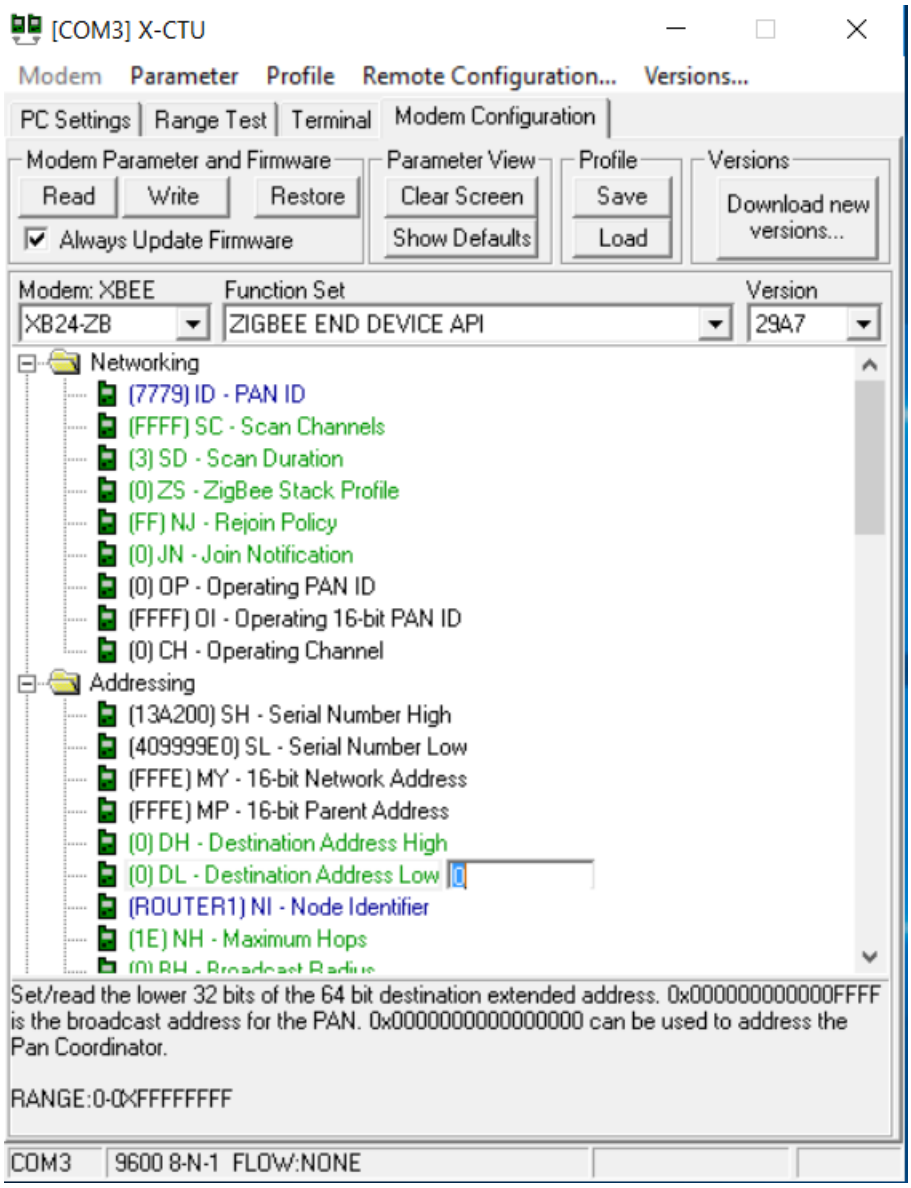
### Accomplished Today:

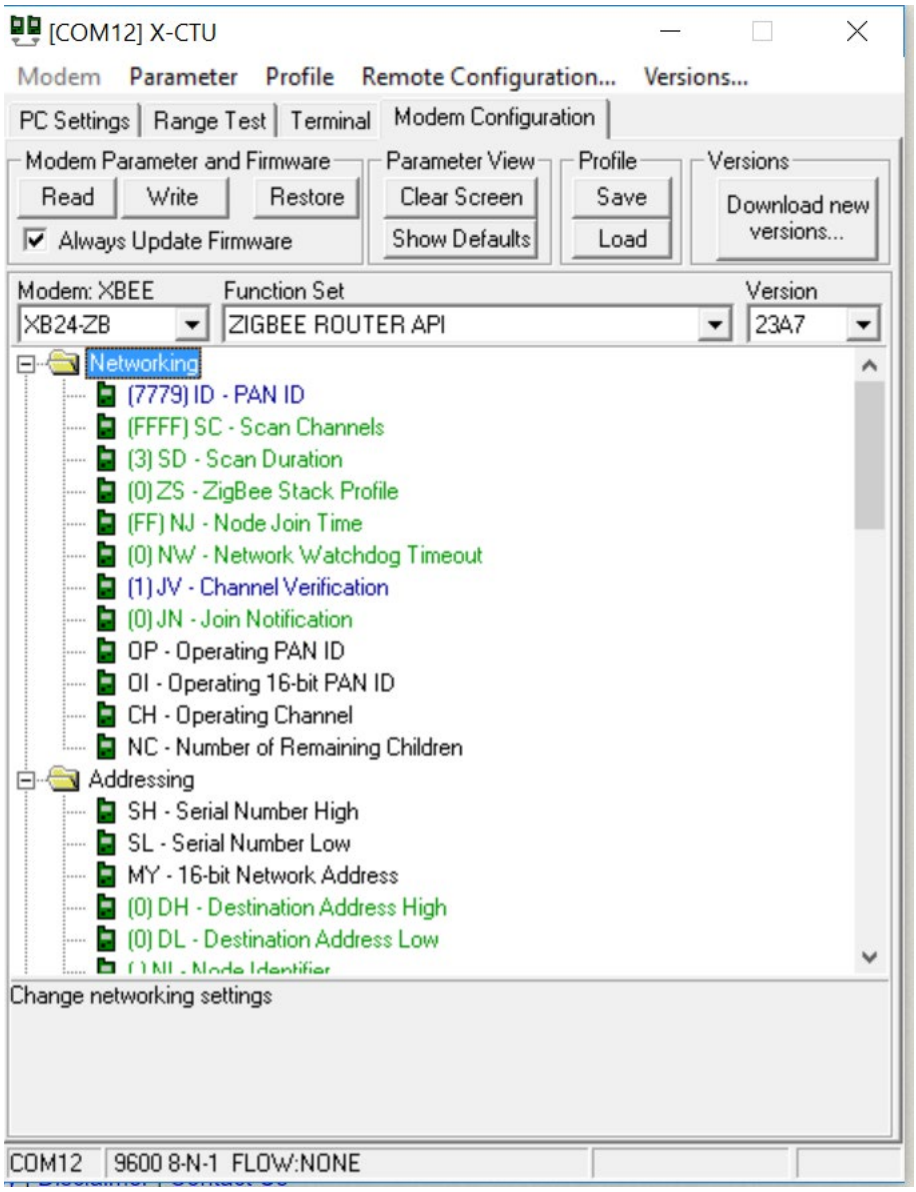
I converted from the AT mode into the API mode. Point-to-point only allowed me to work with 2 Xbees at a time but now API mode should allow multiple Xbees in a network. Currently, I only have three Series 2 Xbees so I set one up at the API Coordinator (which you can only have one of in a network), one as a API Router (can have multiple) and one as a API End Device (also multiple). The eventual goal is to create a mesh network with the Series 2 modules, but that can only be achieved with 4+ modules. In addition to the conversion, I also obtained the SH/SL & DH/DL for the 3 modules. With AT mode, as long as the PAN IDs match, the serial highs and lows/destination highs and lows didn't have to be exact. API Mode requires that these values are exact for all the modules within the network so this configuration was essential. When I opened the XTCU Terminal and tried to send simultaneous messages across the terminal, the data packets would not send.

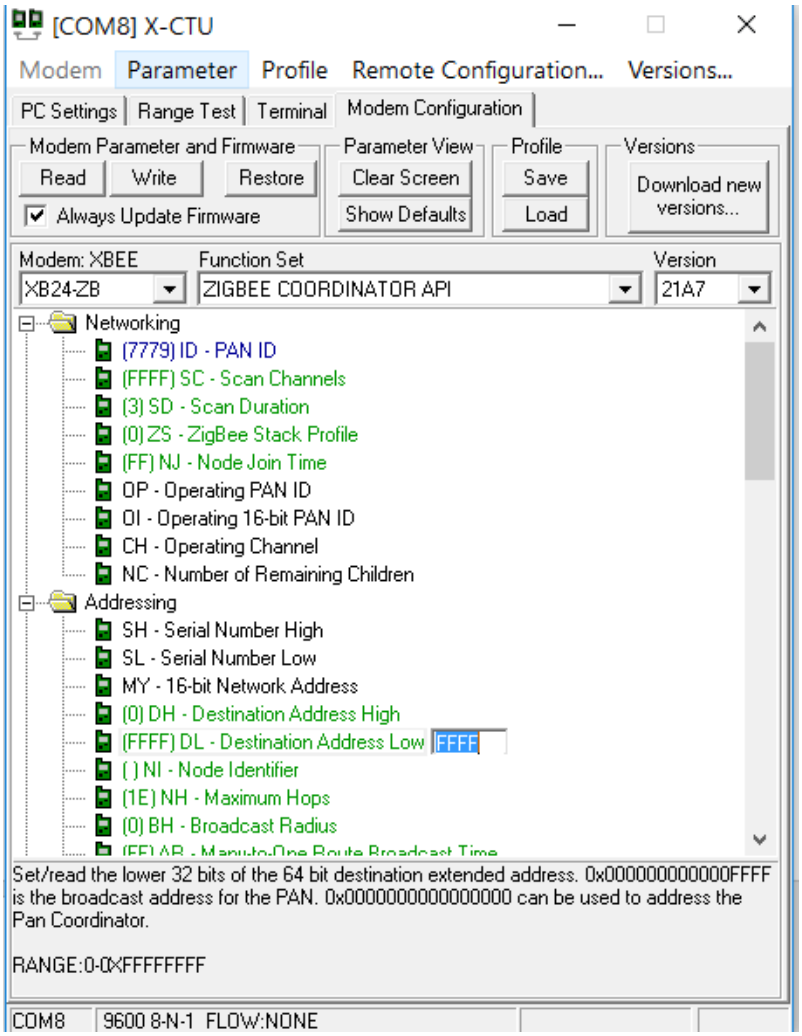
### Next Step:

The next step is to figure out why the terminal was not showing the messages even though all the configuration is apparently correct. I believe I can reference the *Building Wireless Sensor Networks* book to accurately construct the API network and discover the error. Additionally, I want to start working with the Learning Robots and setting up the Xbee Arduino Library with API-enabled Series 2 Modules.

### Daily Screenshot/Picture:


[COM3] X-CTU
Modem
Parameter
Profile
Remote Configuration...
Versions...
PC Settings
Range Test
Terminal
Modem Configuration
Modem Parameter and Firmware
Read
Write
Restore
Always Update Firmware
Parameter View
Clear Screen
Show Defaults
Profile
Save
Load
Versions
Download new versions...
Modem: XBEE
XB24-ZB
Function Set
ZIGBEE END DEVICE API
Version
29A7
Networking
(7779) ID - PAN ID
(FFFF) SC - Scan Channels
(3) SD - Scan Duration
(0) ZS - ZigBee Stack Profile
(FF) NJ - Rejoin Policy
(0) JN - Join Notification
(0) OP - Operating PAN ID
(FFFF) OI - Operating 16-bit PAN ID
(0) CH - Operating Channel
Addressing
(13A200) SH - Serial Number High
(409999E0) SL - Serial Number Low
(FFFE) MY - 16-bit Network Address
(FFFE) MP - 16-bit Parent Address
(0) DH - Destination Address High
(0) DL - Destination Address Low
(ROUTER1) NI - Node Identifier
(1E) NH - Maximum Hops
Set/read the lower 32 bits of the 64 bit destination extended address. 0x000000000000FFFF is the broadcast address for the PAN. 0x0000000000000000 can be used to address the Pan Coordinator.
RANGE:0-0XFFFFFFFF
COM3
9600 8-N-1 FLOW:NONE


[COM12] X-CTU
Modem
Parameter
Profile
Remote Configuration...
Versions...
PC Settings
Range Test
Terminal
Modem Configuration
Modem Parameter and Firmware
Read
Write
Restore
Always Update Firmware
Parameter View
Clear Screen
Show Defaults
Profile
Save
Load
Versions
Download new versions...
Modem: XBEE
XB24-ZB
Function Set
ZIGBEE ROUTER API
Version
23A7
Networking
(7779) ID - PAN ID
(FFFF) SC - Scan Channels
(3) SD - Scan Duration
(0) ZS - ZigBee Stack Profile
(FF) NJ - Node Join Time
(0) NW - Network Watchdog Timeout
(1) JV - Channel Verification
(0) JN - Join Notification
OP - Operating PAN ID
OI - Operating 16-bit PAN ID
CH - Operating Channel
NC - Number of Remaining Children
Addressing
SH - Serial Number High
SL - Serial Number Low
MY - 16-bit Network Address
(0) DH - Destination Address High
(0) DL - Destination Address Low
Change networking settings
COM12
9600 8-N-1 FLOW:NONE


[COM8] X-CTU
Modem
Parameter
Profile
Remote Configuration...
Versions...
PC Settings
Range Test
Terminal
Modem Configuration
Modem Parameter and Firmware
Read
Write
Restore
Always Update Firmware
Parameter View
Clear Screen
Show Defaults
Profile
Save
Load
Versions
Download new versions...
Modem: XBEE
XB24-ZB
Function Set
ZIGBEE COORDINATOR API
Version
21A7
Networking
(7779) ID - PAN ID
(FFFF) SC - Scan Channels
(3) SD - Scan Duration
(0) ZS - ZigBee Stack Profile
(FF) NJ - Node Join Time
OP - Operating PAN ID
OI - Operating 16-bit PAN ID
CH - Operating Channel
NC - Number of Remaining Children
Addressing
SH - Serial Number High
SL - Serial Number Low
MY - 16-bit Network Address
(0) DH - Destination Address High
(FFFF) DL - Destination Address Low
FFFF
( ) NI - Node Identifier
(1E) NH - Maximum Hops
(0) BH - Broadcast Radius
Set/read the lower 32 bits of the 64 bit destination extended address. 0x000000000000FFFF is the broadcast address for the PAN. 0x0000000000000000 can be used to address the Pan Coordinator.
RANGE:0-0XFFFFFFFF
COM8
9600 8-N-1 FLOW:NONE