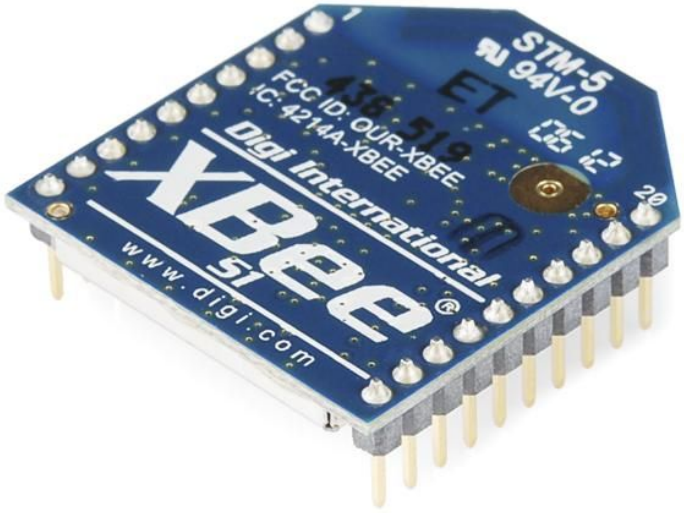
# Xbee Guide

**With Robotics Challenge Application**





## By Adam St. Amand

## Revised By Jack Maydan, Alexis Deukam, Robert Belter

(12.17.2015)

## Revised By Thomas Horning, Sean Lambert, Priyanka Makin, Chase Pellazar

(02.07.2017)

Colorado Space Grant Consortium

# Table of Contents

# 1 Back to Basics

**Covered in this section**

This document discusses what an Xbee is, how to interface with it, and demonstrates some of the Xbee's capabilities with a specific application. This guide is by no means a thorough explanation, but rather a starting place to build a fundamental understating. The reader is highly encouraged not only to follow the guide, but also experiment with the modules in order to gain a deeper understanding. Even so, this guide just scratches the surface of what the Xbee is capable of.

## 1.1 Foundations

### 1.1.1 What is an XBee?

It can get confusing when you start learning about Xbee's, but don't let this deter you! Xbee's are extremely popular and it's because they are very powerful and simple to use. Here is a link to a quick guide that has all the information you need to get started. It's well written and certainly worth a quick look through.

https://www.sparkfun.com/pages/xbee_guide

If you're like me (slow to learn) then you may need to read that guide several times before things start to click. To help out, here's a quick summary of the key points.

**IEEE 802.15.4** is a standard protocol for low rate wireless personal area networks.

**ZigBee** is sort of an extension of this standard which adds capability and complexity.

The **Digimesh** protocol is similar to Zigbee in that is it another extension of the IEEE 802.15.4 standard, but it offers different functionality than that of ZigBee.

---

**Frequency** – 2.4GHz, 900MHz, 868MHz, and 865MHz. The lower frequencies can penetrate better and therefore potentially have greater range, but they require a larger antenna to be effective.

**Power** – Regular (Low Power/LP) or Pro (High Power). The Pro version functions the same as the regular except that it uses more power (costs more money) and consequently has a much greater range.

**Wireless Interface** – 802.15.4 (Series 1), Zigbee (ZB/Series 2), DigiMesh, and Wi-Fi (802.11).

**Antenna** – Chip Antenna, Wire Antenna, u.FL Antenna, RPSMA Antenna, and the Trace Antenna.

An **Xbee** is an RF (Radio Frequency) module created by Digi. The Xbee comes in plenty of different varieties. Typical characteristics of the different models are listed below.

Here's a **very useful** visual reference to sum it all up:
http://www.digi.com/pdf/chart_xbee_rf_features.pdf

**Resources/Information**

Brief Xbee Introduction
https://www.youtube.com/watch?v=Wyvpjy9MPD4

Zigbee Basics
https://www.youtube.com/watch?v=dn4631u2Zxg

Xbee Overview
https://www.youtube.com/watch?v=RYQpZWij2-M

WLPAN & IEEE 802.15.4 Tutorial
http://dkc1.digikey.com/us/en/tod/Atmel/LowPowerWireless/LowPowerWireless.html

Xbee vs. Xbee Pro
https://www.youtube.com/watch?v=G_ScqGDY2eoLnk

Xbee Buying Guide
https://www.sparkfun.com/pages/xbee_guide

Xbee (Series 1) Product Manual
http://ftp1.digi.com/support/documentation/90000982_S.pdf

Xbee Reference Guide
http://www.digi.com/pdf/chart_xbee_rf_features.pdf

### 1.1.2 What is a Fio?

The Fio, just like an Arduino Uno or Arduino Mega, is a microcontroller development board. This board is special however because it is optimized to interact with Xbee modules. The Fio has an on-board socket which allows an Xbee to be attached to the board directly! The Fio has two versions, both of which are described below.

**Arduino Fio**

This board has the Xbee socket, **but** the microcontroller on the board is an ATmega328P. Here's how SparkFun describes it.

*"The Arduino Funnel I/O (Fio) is a board designed by Shigeru Kobayashi, based on the original design from HYPERLINK "http://www.sparkfun.com/commerce/categories.php?c=135" LilyPad.*

*Funnel is a toolkit to sketch your idea physically, and consists of software libraries and hardware. By using Funnel, the user can interface to sensors and/or actuators with various programming languages such as ActionScript 3, Processing, and Ruby.*

*Arduino Fio is compatible with Funnel. It has connections for a Lithium Polymer battery and includes a charge circuit over USB. An XBee socket is available on the bottom of the board. The Fio has been designed to be wirelessly reprogrammable"*

**Fio v3**

The Fio v3 has an ATmega32U4 microcontroller on it. It's similar to the Arduino Fio, but they have a few key differences. Below is a description from SparkFun.

*"The Fio v3 is a new spin on the Arduino Fio hardware powered by the ATmega32U4.Not only is it small and LiPo-ready, it's a very capable XBee-ready development board.*

*The JST-connector and 3.3v system voltage make this a great development tool for portable devices, simply plug in a Li-Poly battery and you're ready to go. Wireless sensor networks and communication are made easy by the on-board XBee socket."*

Due to the differences in microcontrollers, the Arduino Fio **requires** an external serial connection over an FTDI Basic, FTDI cable, or other serial connection in order to program the microcontroller via a **wire connection**. The Fio v3, however, can be programed easily by using an on-board USB jack. Also the Fio v3 can function as an HID. An HID is a human interface device, such as a mouse or keyboard.

| | Arduino Fio | Fio v3 |
|---|---|---|
| **Microcontroller** | ATmega328P | ATmega32U4 |
| **HID Capability** | No | Yes |
| **Programming** | Requires external Serial Connection | On-Board USB Jack |
| **Xbee Socket** | Yes | Yes |
| **Li-Poly Battery** | Yes | Yes |

**Lithium Polymer Batteries**

Another benefit of the Arduino Fio and the Fio v3 is that they consume relatively little power. This allows them to be battery powered. Each comes with a connection for a lithium polymer battery and includes a charging circuit. In other words, you can charge the Li-Po by plugging both the board into a power source via USB as well as the Li-Po into the board.

***Note:*** *From here on, "Fio" will refer to the Fio v3*

**Resources/Information**

Fio v3 Product Page
https://www.sparkfun.com/products/11520

Arduino Fio Product Page
https://www.sparkfun.com/products/10116

Fio v3 Hookup Guide
https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide

Arduino Fio Overview
http://arduino.cc/en/Main/ArduinoBoardFio

## 1.2 Application

Although the Xbee module is extremely flexible and can be used for a wide variety of applications, in this section we will focus on a particular application; building a beacon-tracking receiver as well as the beacon itself.

### 1.2.1 The General Idea

Imagine you're in an open field looking straight down at a compass placed in your hand. Also in the field with you is a friend. Your friend is similarly looking straight down at a compass placed in their hand, but they are also spinning in place. Your goal is to navigate to your friend, but ***without*** either of you looking up from your compasses.

To help you out, your friend is constantly yelling out the direction he is facing (his heading). With some careful reasoning, you deduce that when his voice is the loudest, he must be facing directly towards you.

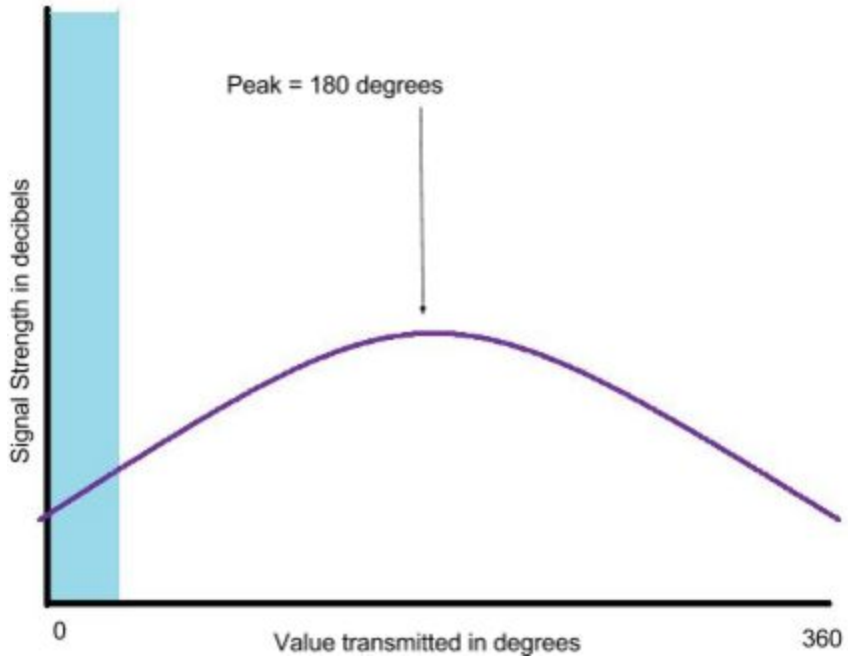
Now, you listen for a little while, keeping track of what direction he yells out when his voice is the loudest. Then, you use your compass to navigate to the opposite of that direction. (For example, if during your friend's loudest yell he said "270 degrees!", then you would travel 90 degrees to navigate towards him.)


The Xbee beacon and receiver are much the same, except instead of listening for the loudest yell, the Xbee receiver will watch for the **strongest signal**.

Relating the analogy to the actual system, the Xbee beacon will constantly transmit the direction it is facing (its heading) while the Xbee receiver will constantly receive those headings. The Xbee receiver will then check to see which of those headings was received with the greatest signal strength. The heading with the most signal strength is the heading that the beacon was transmitting when it was facing the receiver. This is how we'll navigate.

**How do you know when the signal is strongest?**

The Xbee modules emit a field of signal that is generally the same strength in all directions. In our application we will use a **patch antenna**. The patch antenna transmits data in a directional beam. The further the receiver is from the directional beam from the transmitter, the weaker its signal is. In this example the strongest signal (peak) is received when the beacon is pointing approximately 180 degrees.




### 1.2.2 Hardware/Software List

The application mentioned above can be accomplished in many different ways using the same hardware and even more ways by changing the hardware. This guide will take you step by step on one approach, but feel free to skew from the guide if you're feeling adventurous or feel that you can optimize with your setup. With that said, below is a list of hardware and software that will be **required for our approach**.

| Software | Hardware |
|---|---|
| Arduino IDE (code verified for V1.0.6)<br>http://arduino.cc/en/main/software | Fio v3 Atmega32U3 (2)<br>https://www.sparkfun.com/products/11520 |
| | Xbee (Series 1; Trace Antenna) (1)<br>https://www.sparkfun.com/products/11215 |
| XCTU<br>http://www.digi.com/xctu | Xbee (Series 1; u.FL connection) (1)<br>https://www.sparkfun.com/products/8666 |
| | Interface Cable SMA to u.FL (1)<br>https://www.sparkfun.com/products/9145 |

| | |
|---|---|
| Robotics Challenge Code<br>https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge | Lithium Polymer Battery (2)<br>https://www.sparkfun.com/products/8483 |
| | Patch antenna (Male SMA) (1)<br>http://www.l-com.com/wireless-antenna-24-ghz-8-dbi-flat-patch-antenna-4ft-sma-male-connector# |
| Xbee-Arduino API Library<br>https://code.google.com/p/xbee-arduino/ | Magnetometer (LSM9DS1, SparkFun 9dof stick) (1)<br>https://www.sparkfun.com/products/13944 |
| | Xbee explorer (optional)<br>https://www.sparkfun.com/products/11812?_ga=1.16 49458.2009305648.1417585592 |

### 1.2.3 Configuring the Xbee using XCTU (Video Tutorial)

One of the many strengths of the Xbee (the Series 1 in particular) is that it is relatively simple to configure. DigiKey has written a user friendly software called **XCTU** that allows a user to change a bunch of parameters and settings on the Xbee. But before we can use XCTU to configure the Xbee, we need to find out how to connect the Xbee to the computer so that XCTU can recognize it.

The trick is to connect the Xbee to the computer via USB so that XCTU can communicate with the module. At this point, there are **many** approaches that can be taken to accomplish the same task.

You can configure Xbee's **wirelessly**, using an **Xbee explorer**, using a **Fio**, etc.

This guide will cover how to use the **Fio**, but below is a link to the SparkFun tutorial explaining how to configure Xbee's using an explorer. The approaches are only slightly different, and when it comes to using XCTU SparkFun has an excellent explanation. Regardless of the approach you take, it's worth reading through the SparkFun tutorial.

https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu?_ga=1.203678611.2009305648.1417585592

## Configuring Arduino for Sparkfun's custom boards

Before using the Fio to configure the Xbee, make sure you have all the necessary software/drivers in place so that you can program the Fio. The SparkFun Hookup Tutorial covers how to do all of this. The whole guide is worth reading, but you specifically need to read the Installation section.

https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide

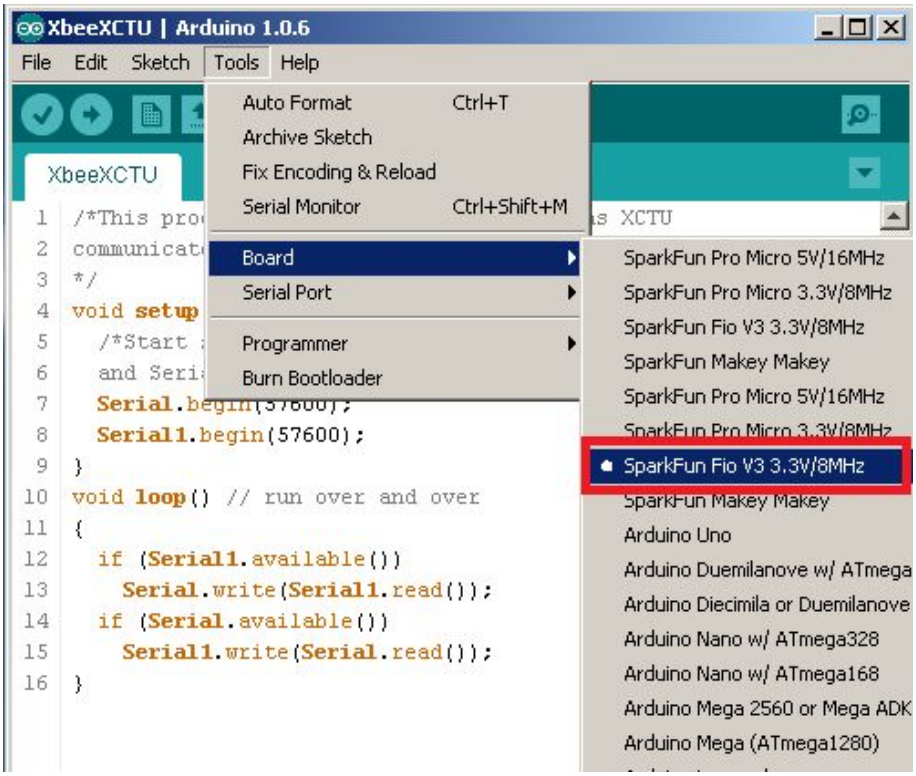
**Using the Fio to Configure Xbee**

In order to interface with the Xbee we will essentially use the Fio as an Xbee explorer. The problem is that if you connect the Fio via USB the computer will read data from the Fio, not the Xbee.

The way around this is a simple sketch that tells the Fio to send data from the Xbee to the computer and vice versa. That way the Fio sort of becomes a transparent medium through which data travels to and from the Xbee and Fio. The steps to accomplish this are as follows:

**a.** Attach the Xbee to the Fio. (Make sure the Xbee is facing the right way! There's an outline on the Fio so you can easily tell.)

**b.** Plug the Fio into a computer via USB.




**IMPORTANT**

Before proceeding to the next step, in the Arduino IDE you must have the correct board selected! If you upload any sketch to the Fio while the incorrect board is selected, you will most likely brick the Fio. To ensure you have the correct board selected, go to Tools>Board>SparkFun Fio V3 3.3/8MHz.

**c.** Upload the "XbeeXCTU" sketch to the Fio, found here:

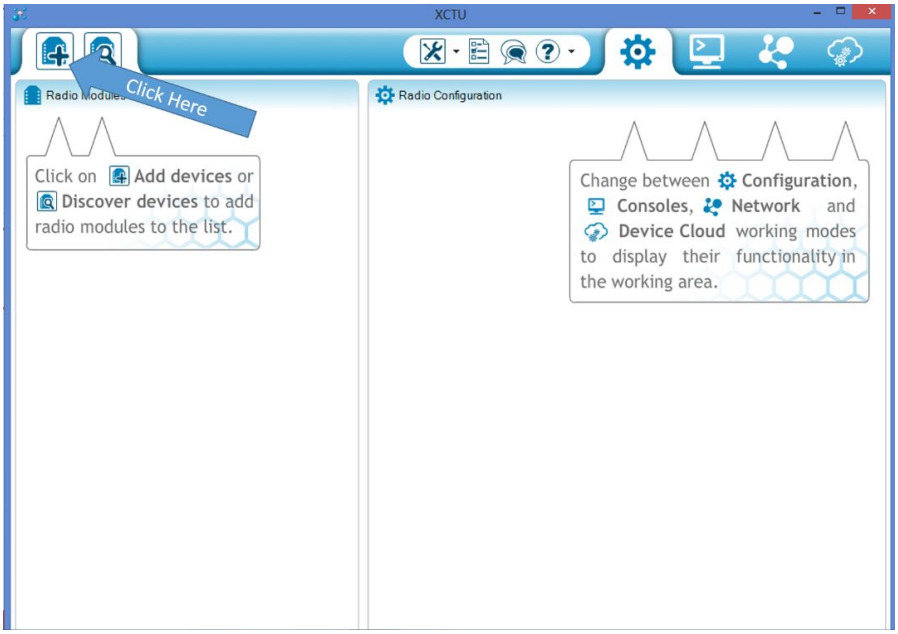
https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2015_Robotics_Challenge/XbeeDiagnostics/XbeeXCTU

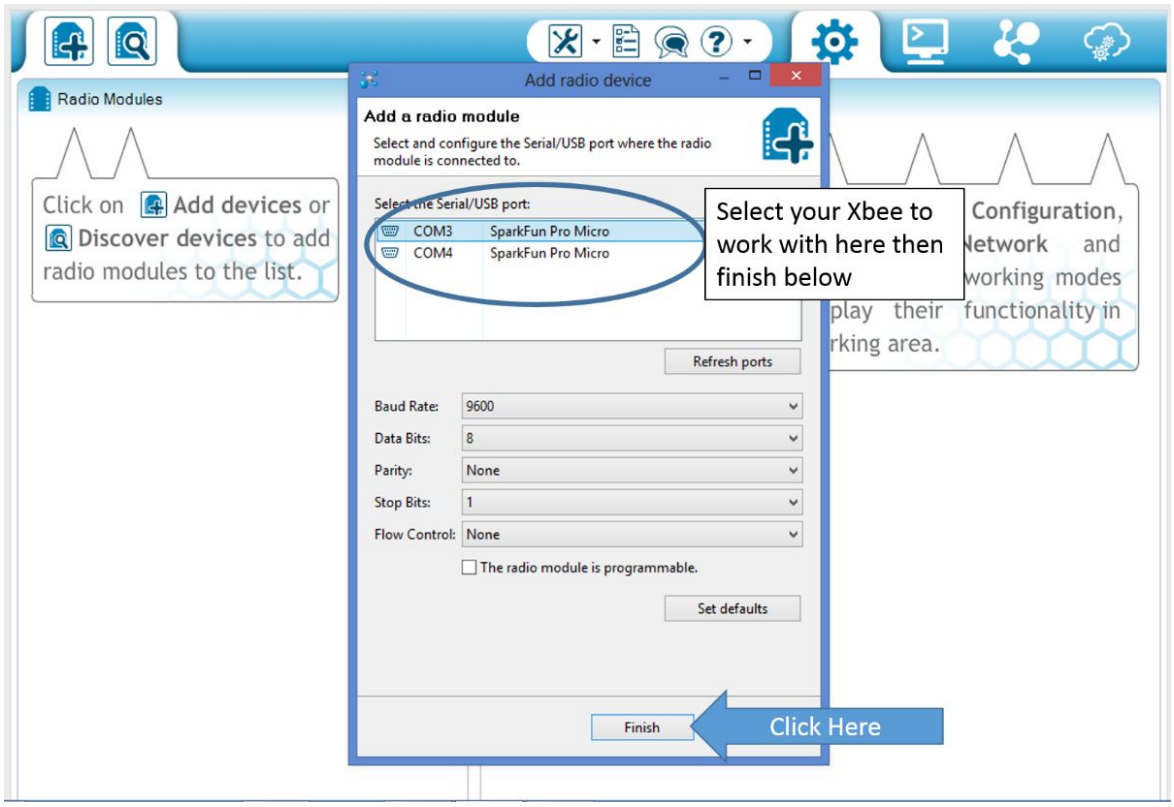
As mentioned before, XCTU will not recognize your Xbee unless this code is run on your Fio first. This sketch allows XCTU software to talk to the Xbee through the Fio.

Note: Before uploading XB_XCTU sketch to the Fio for the first time change the baud rate in the sketch to 57600.

**d.** Open XCTU/Add Devices

Click on the *Add Devices* button in the top left corner. You should see the “SparkFun Fio v3” on the list. Select it and hit *Finish*.






**e.** Configure the Xbee

Double click on the device under the *Radio Modules* section. From here you have access to configure the Xbee in many different ways. The recommended configuration is listed below, but these configurations can be changed or adjusted to meet your specific needs. Repeat steps (a-e) for the transmitter Xbee. All other settings in XCTU should remain as default.

| Receiver Xbee | Transmitter Xbee (Beacon) |
|---|---|
| **ID** PAN ID = 1111<br>**DL** Destination Address Low = 1234<br>**MY** 16-bit Source Address = 5678<br>**BD** Interface Data Rate = 57600[6]<br>**AP** API Enable = API enabled w/PPP [2] | **ID** PAN ID = 1111<br>**DL** Destination Address Low = FFFF<br>**MY** 16-bit Source Address = 1234<br>**PL** Power Level = Lowest [0]<br>(PL can be adjusted up or down to improve signal strength at distance)<br>**BD** Interface Data Rate = 57600[6]<br>**AP** API Enable = API enabled w/PPP [2] |

***Remember!*** After making changes to the Xbee you must write the changes. Click the pencil icon at the top of the Radio Configuration pane to write all changes made.

**Note:** Depending on what type of Xbee's you're using, it is really easy to lose track of which

Xbee is configured as the receiver and which is the transmitter. It may be wise to make some kind of distinguishing mark on the Xbee's so that you don't confuse the two. This applies to the Fio's as well. Sharpie works fine.

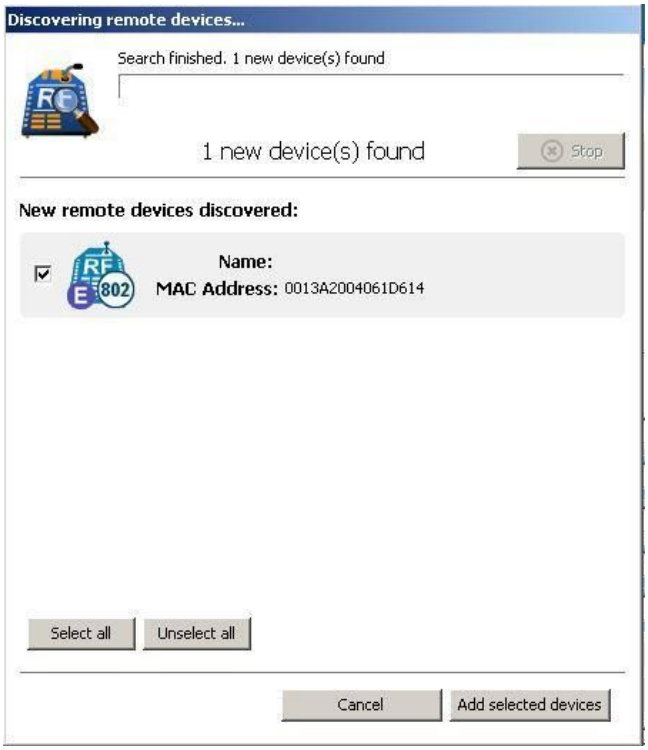
**f.** Exit XCTU

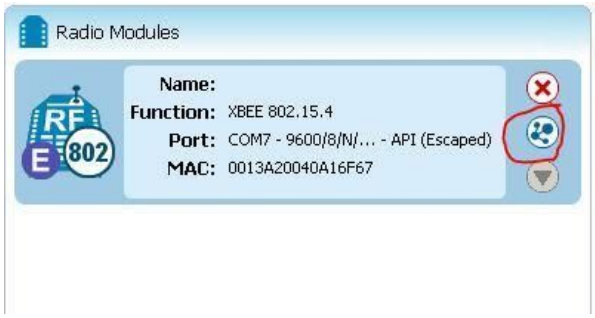
Each time you need to configure the Xbee modules, you will need to run through steps a-e again. In other words, you will have to upload the "XbeeXCTU" sketch to the Fio each time you want to make changes to the Xbee configuration. This sketch is what essentially turns the Fio into an Xbee explorer. Alternatively, you could just use Xbee explorers, eliminating the need to upload the sketch each time.

Note: Reopen the XB_XCTU sketch and change the baud rate back to 9600. Upload it to the Fio.

***Bonus*** **(Video Tutorial)**

After you get the Xbee's communicating with each other initially, you can easily configure multiple Xbee's at once with just one plugged in via USB. With one Xbee open in XCTU, click the blue networking icon on the device's image. All other Xbee's on the same network will be displayed. Select which ones you would like to configure and click on *Add selected devices*.

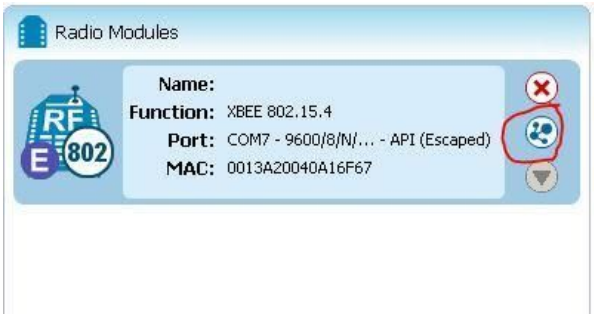





**Test Run (Video Tutorial)**

With both Xbee's configured, they ought to be communicating with each other. Now is a good time to make sure of this working that we don't run into issues later.

This time, plug **both** Xbee's into the computer. This means you will need to upload the "XbeeXCTU" sketch to both Fio's. Run XCTU. You should be able to discover both Xbee's by clicking here:



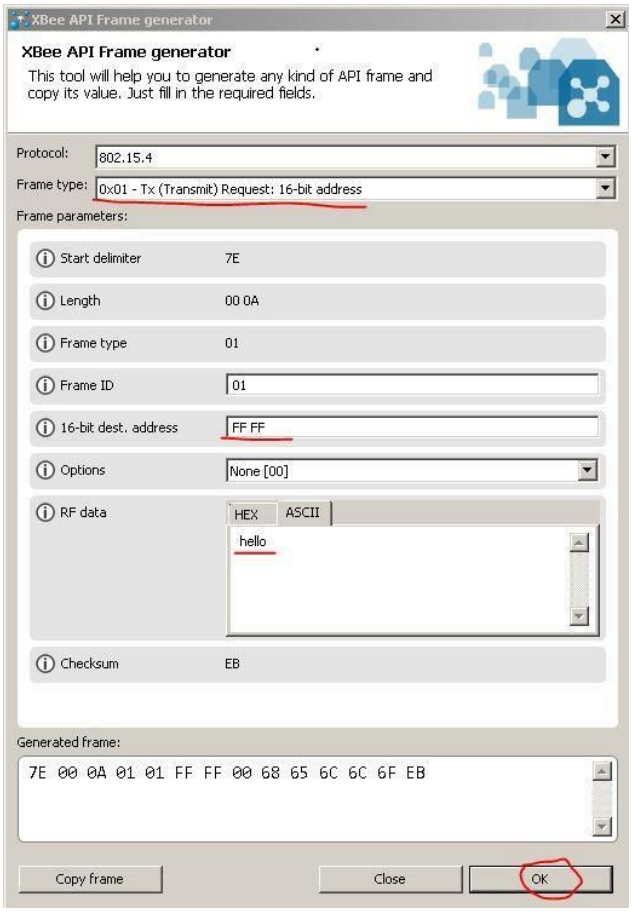


Click the terminal icon in the top right corner.

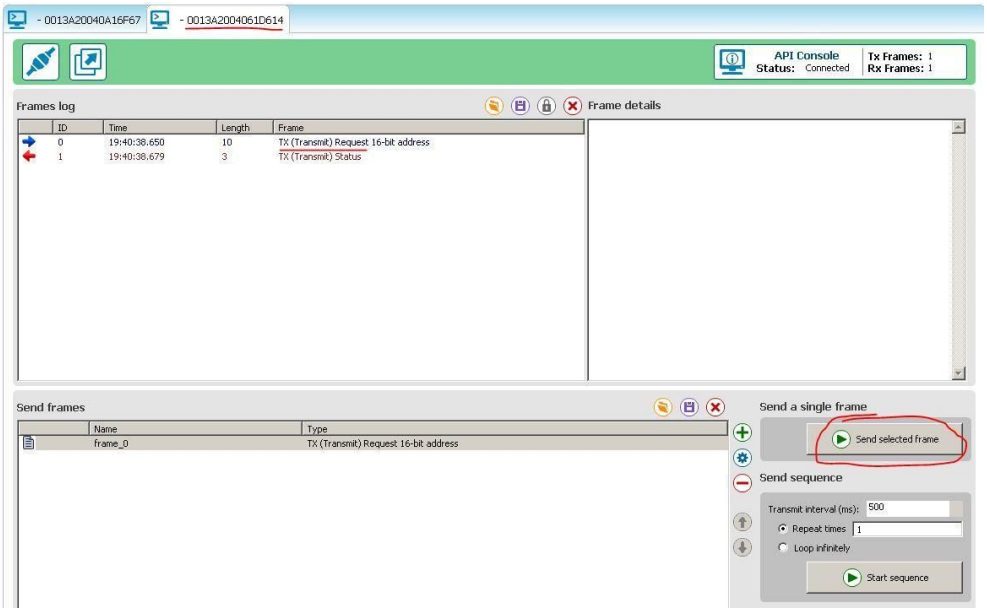
Open serial communication by clicking the plug icon. You will have to do this for both Xbee's

With the Transmitter Xbee selected, click the 'add new frame to list' button in the lower right. A window will pop-up. Click "create frame using 'Frames Generator' tool".

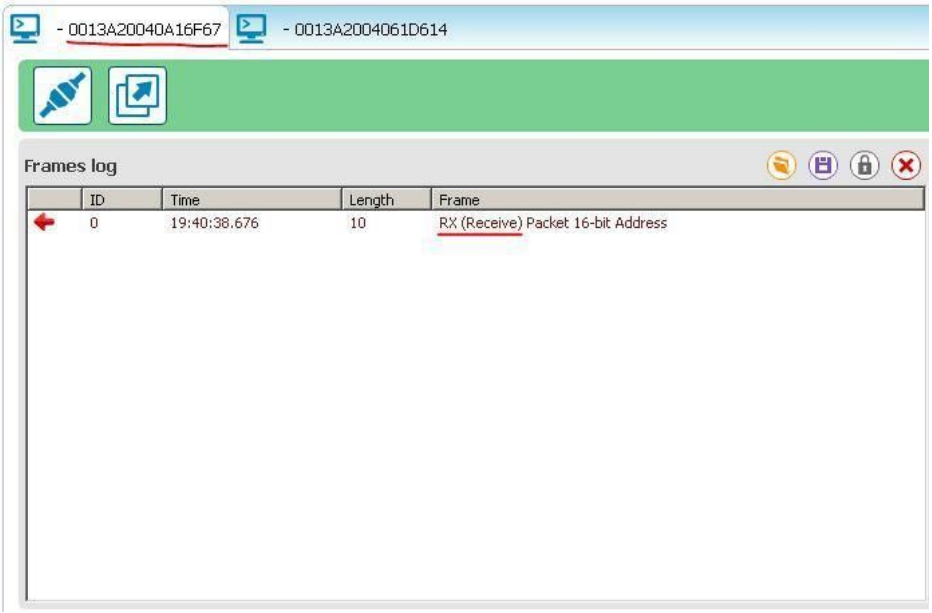
Select the proper settings as illustrated in the image below. The RF data doesn't really matter, but make sure you put something there. Hit OK when you are finished.




Click 'Send selected frame' in the bottom right. You should notice a message appear in the Frames log that says a packet has been transmitted.




Now select the Receiving Xbee by clicking on the tab at the top. You should notice a message in the Frames log that says a packet has been received. Congratulations! The Xbee's are now communicating with each other in API mode.



**Resources/Information**

| | |
|---|---|
| XCTU Overview<br>https://www.youtube.com/watch?v=EA-2Xa5OAY8 | Xbee Explorer<br>https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu?_ga=1.203678611.2009305648.1417585592 |
| Fio-V3 Hookup Guide<br>https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide | SparkFun 9DOF Sensor Stick Guide<br>https://learn.sparkfun.com/tutorials/9dof-sensor-stick-hookup-guide?_ga=1.229623132.1340058446.1441581116 |

### 1.2.4 Programming Arduino for Xbee (Video Tutorial)

Now that the Xbee's are configured to communicate with each other, we need to take a look at how we'll send and receive information to and from the Xbee's. There is a powerful library for Arduino that has simplified the process of sending/receiving information to the Xbee's greatly.

https://code.google.com/p/xbee-arduino/

You just need to download the library and place it in your Arduino IDE Library folder. If you'd like to know more about how the code works and what's going on behind the scenes, there's some information in the link above. With the library in place, we can now upload code to the Fio's that will interact with the Xbee's.

**Note:** The Xbee-Arduino library is hosted on Google Code, which will no longer be supported in the near future. The project can also be found on GitHub and will likely be available there for a much longer time.

https://github.com/andrewrapp/xbee-arduino

## 1.3 Usage Guide and Fio Instructions: Robotics Application

Now that you have two Xbee's communicating you can apply this wireless data transmission to your navigating robot. The Transmitter Xbee will send out a compass heading that it obtains from a triple axis magnetometer.

Once you have the XBee properly configured, download the receiver code below to the fio.
https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2017_Robotics_Challenge/xb_Receiver_System_Code
and the transmitter code to the beacon Fio (if needed):
https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2017_Robotics_Challenge/xb_Transmitter_System_Code

When the receiver code is uploaded to the Fio, it will use the output from the transmitter XBee to find the direction from the beacon to the receiver. It will then output this as a value from 0 - 359 degrees, with north being 0, increasing clockwise. If no signal can be received, or is otherwise invalid, it will output -1. This output can be received in a few different ways.

This is table of various methods for obtaining data from the Fio.

| | |
|---|---|
| Serial (Over USB) | The output from the Fio can be read over serial monitor on a computer (Baud rate 57600). It will print a newline for each reading, which will occur approximately every 5 seconds. |
| Serial (Over DIO) | The output will also be sent over serial on DIO9 on the Fio (Baud rate 57600). The output will appear the same as to the computer. |
| I2C | The output can also be accessed over I2C. The I2C pins on the fio are<br>- SDA - DIO2<br>- SCK - DIO3<br>The address of the Fio will be 0x08 and the value will be sent as a two byte signed short (MSB first). |

For examples of how to do the aforementioned read types through another arduino see:

I2C:

https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2017_Robotics_Challenge/Arduino_Read_using_i2c

UART:

https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2017_Robotics_Challenge/Arduino_Read_using_UART

**Note:** You should familiarize yourself with the data that the transmitter sends, and the receiver gets. However, using Xbee's alone is not accurate to the systems that will be used for the challenge. In the challenge the transmitter will use a patch antenna and transmit a directional beam of signal. You will use the receiver code to filter signal strength so that you receive only the strongest (most direct) data, which is what gives you the correct direction. Without a patch antenna your Xbee's will receive your compass reading no matter what.

# 2 Build a Diagnostics Unit

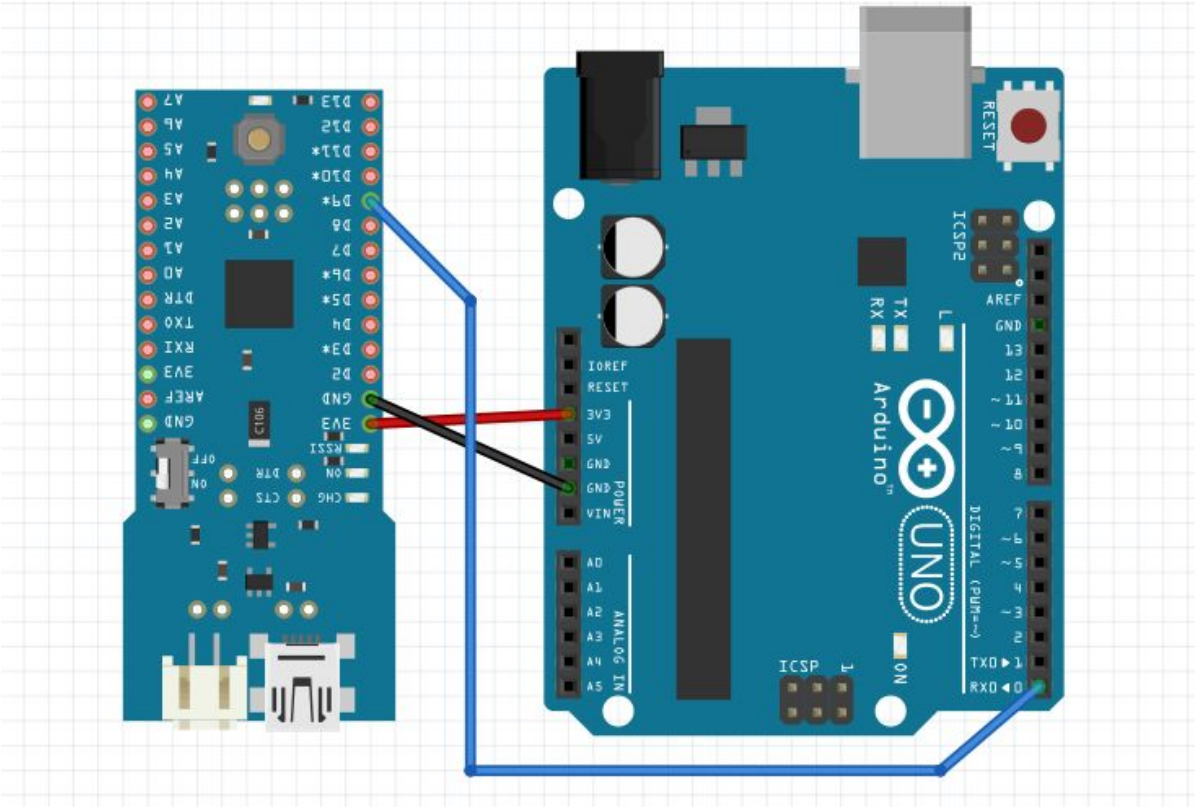
A diagnostic unit is useful because it allows you to test your data transmission and reception from the beacon without having to involve your robot. This helps isolate issues and helps make sure your robot is receiving the correct data.

## 2.1 Parts list

| Description | Price | Link | Notes |
|---|---|---|---|
| **Reciever Module** | | | |
| Fio v3 | $34.95 | https://www.sparkfun.com/products/11520 | |
| XBee 1mW Trace S1 | $24.95 | https://www.sparkfun.com/products/11215 | |
| **Remaining Mat.** | | | |
| Arduino Uno | $24.95 | https://www.sparkfun.com/products/11021 | Any Arduino may be used here |
| Project Box | ~ $5.00 | | The box just needs to be large enough to hold all components |
| Protoshield | $9.95 | https://www.sparkfun.com/products/7914 | Optional |
| LCD Display | $17.95 | https://www.sparkfun.com/products/256 | Any HD44780 LCD is OK. |
| 10K potentiometer | ~ $0.95 | | Any 10k Potentiometer will work |
| Toggle Switch | ~ $0.95 | | Any toggle switch |
| 9V Battery connector | ~ $1.15 | | Other options for power can be used |
| | | | |
| **Total** | $112.75 | | |

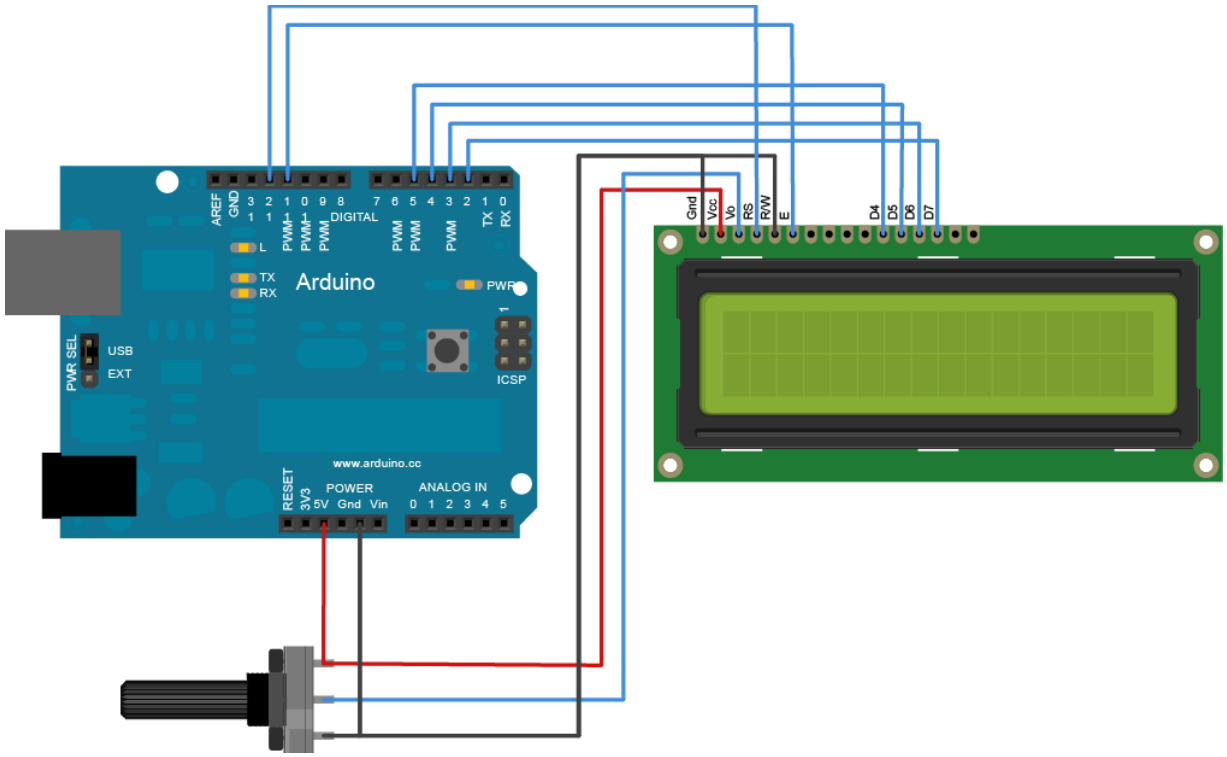
## 2.2 Electrical Wiring / Connections

The electrical wiring for the RF Debug Unit is very straight forward.
To start, connect the Fio to the Uno as below:




- Fio 3v3 to Arduino 3v3
- Fio GND to Arduino GND
- Fio D9 to Arduino D0

Next, connect the LCD and potentiometer to the Uno:




- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2

Finally, you will need to provide power to your Arduino. We did this by using a 9V battery, but anything that will power an Arduino Uno is fine. We also recommend that you use a toggle switch, to allow you to turn the unit on and off from the outside.

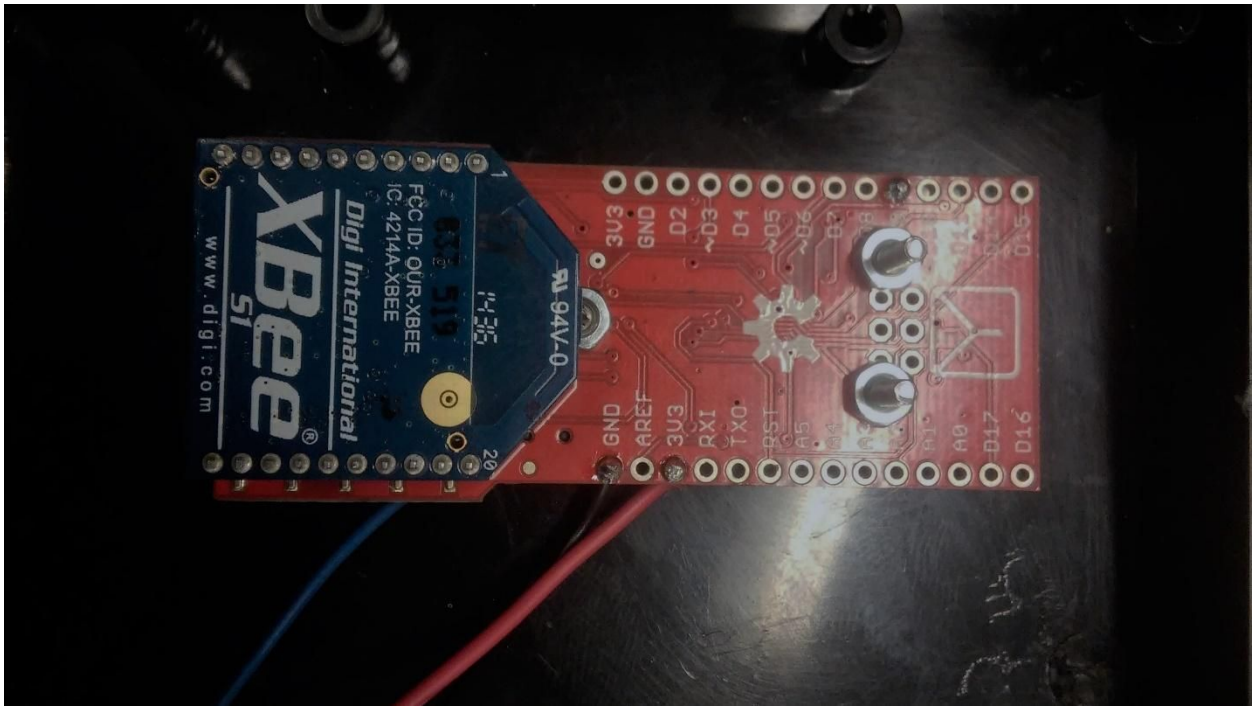
## 2.3 Software

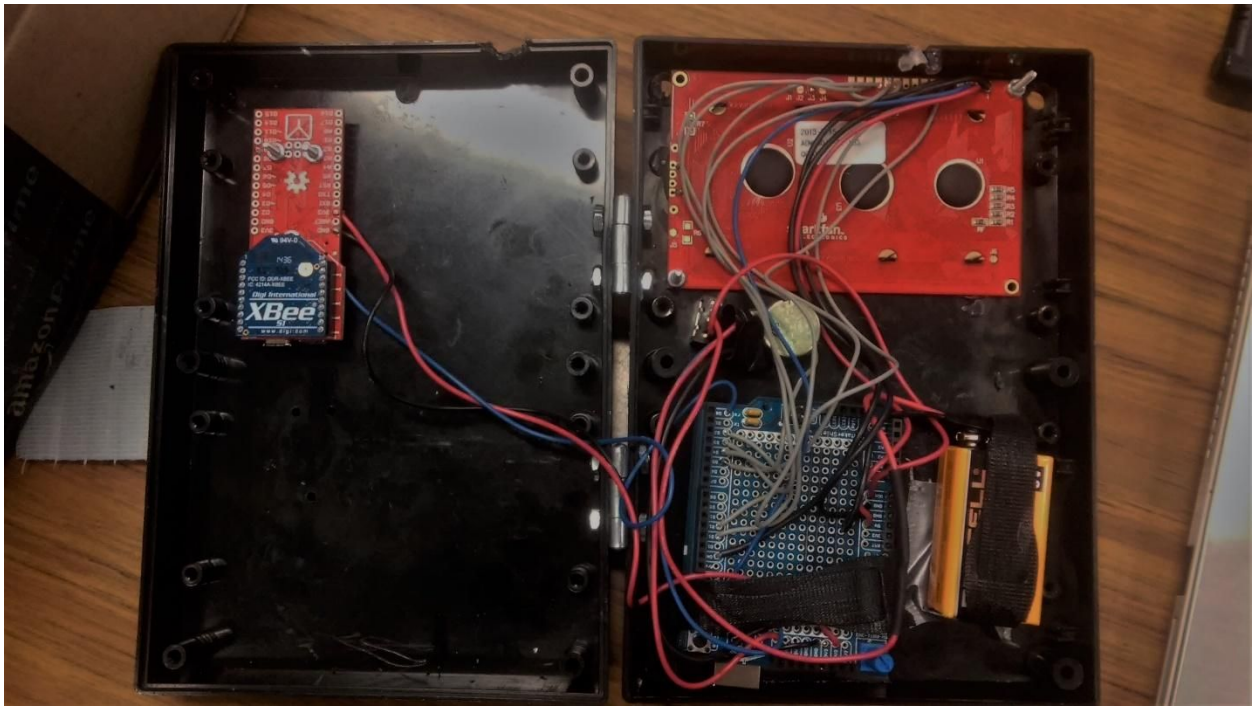
Before mounting all of the components into the project box, configure the xBee as in the *Instructions for Configuring XBee System* guide, upload the receiver software to the Fio, and the processing code to the Arduino Uno.

| | |
|---|---|
| *Fio Receiver Code* | *https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2017_Robotics_Challenge/xb_Receiver_System_Code* |
| *Uno Code* | https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2017_Robotics_Challenge/Arduino_Debug_or_Diagnostic_Unit_Code |

## 2.4 Mounting

The final step is to mount all of the items into the project box. There are many different ways to do this, and it is largely up to personal preference. We have provided pictures of our setup below:

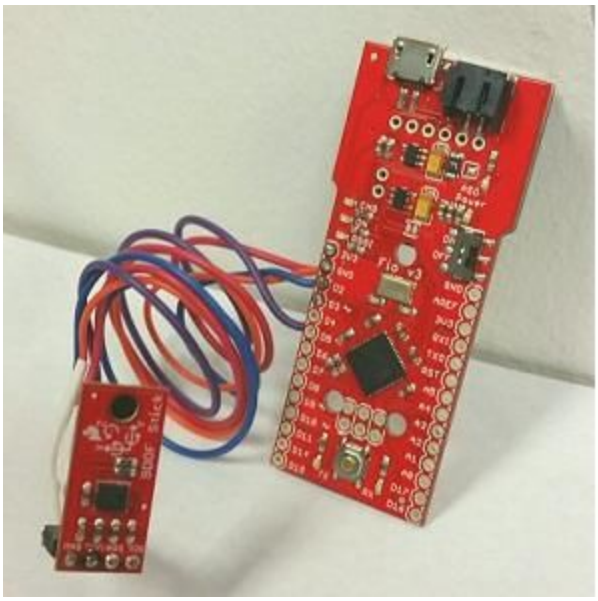

XBee
S1
Digi International
FCC ID: OUR-XBEE
IC: 4214A-XBEE
www.digi.com
94V-0
3V3
GND
D2
D4
D15
GND
AREF
3V3
RXI
TXO
D17
D16

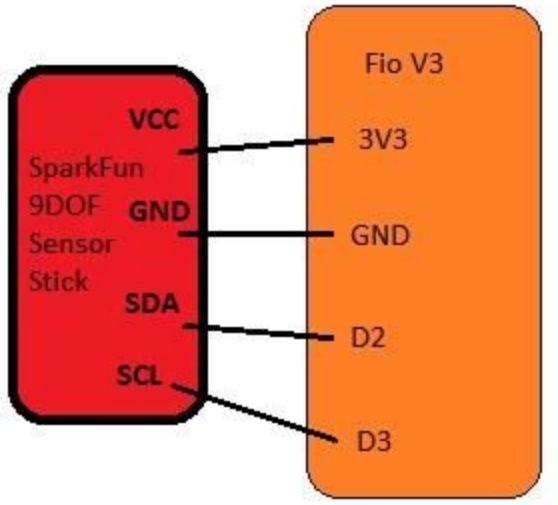


## 2.5 Transmitter/Beacon

Now that we have the Xbee's communicating with each other, we have to tell the Transmitter Xbee what to send. In this application, we want the Transmitter to transmit the direction it is facing (its heading) to any receivers in the area. This means the transmitting Fio simply takes the heading from a compass and sends it to the transmitting Xbee which in turn sends it to a receiving Xbee. We will use the LSM9DS1 SparkFun 9DOF Sensor Stick as a digital compass.

First attach the compass to the transmitting Fio. This step is simple, and only requires some basic soldering. Below is a graphic that shows which pins go where.






Now we can look at the software necessary to make it run. Since the Xbee is already configured, all we need to do is setup the compass.

### 2.5.1 Compass Setup

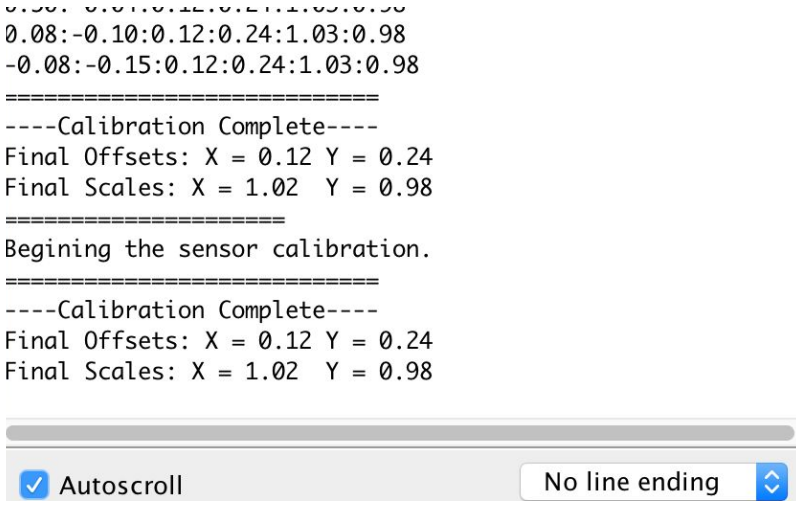
Digital compasses are easily affected by their surroundings. Magnetic influences in the immediate environment and sensor bias will cause a digital compass's readings to be inaccurate. We need to calibrate the compass by correcting for the magnetometer's **offset** and **scaling** error. Traditionally there are two ways to do this; apply offset and scale factor to the registers of the compass, or adjust our code to "fix" the data coming back from the compass. We will do the latter.

We need to find the compass's offset and scaling factors and apply it to all code used in

conjunction with the compass.

We have provided code that you can upload to the Fio that will give back the compass's offset and scale factors after a series of instructions. The sketch is called **'MagnetometerCalibration9DOF.ino'**. It can be found on the Robotics challenge GitHub page.

Once you have completed the calibration, you should now have four values similar to the image below.



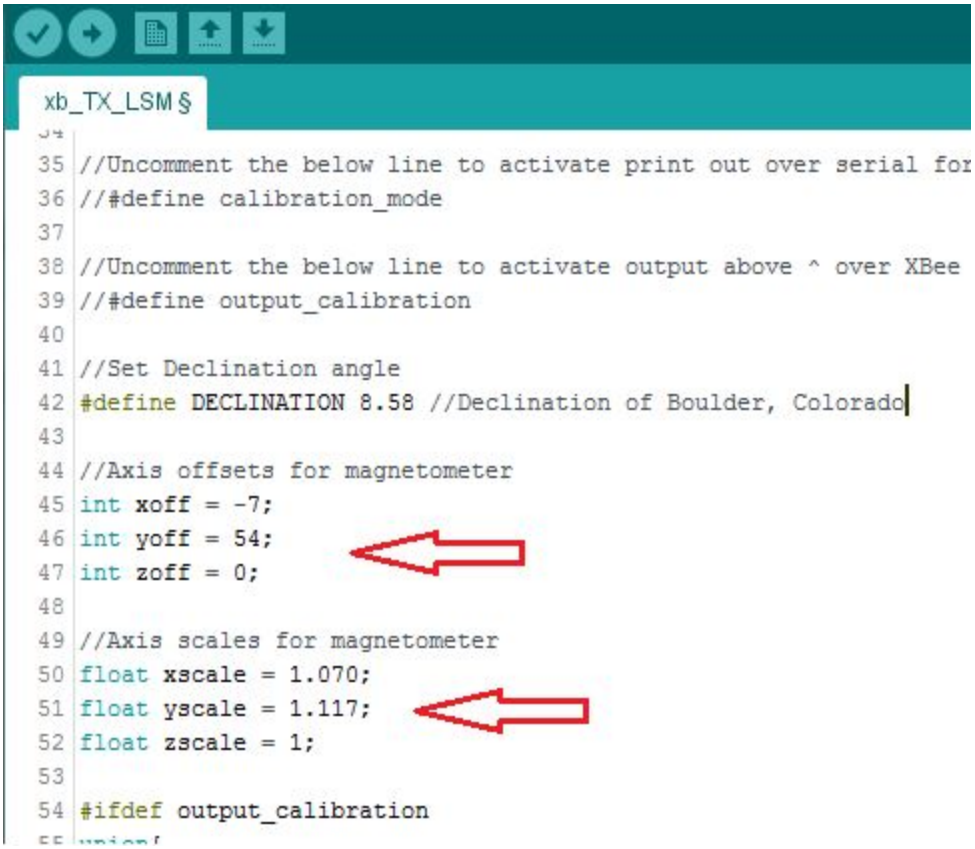

Now, that we have the offset and scale factors, we need one more value for calibration. **Declination.**

Due to variations in the Earth's magnetic field, we need to account for the local declination. You can simply look up the declination of your area:

www.magnetic-declination.com
or
www.ngdc.noaa.gov/geomag-web/#declination

Next, we need to apply these values to our code, which means were are ready to complete set-up of the Fio.

Now we will upload the final code to the Fio. This code will do exactly what we described earlier; take a reading from the compass and send it to the transmitting Xbee. The sketch is called '**Xb_TX_LSM**'. It can be found on the Robotics challenge GitHub page. Open the sketch and add the offset and scale factors, and declination you just found into the code, as shown below. Then upload!




**Note:** The transmitter Xbee must be attached to the **patch antenna** in order to transmit reliably. It's as simple as plugging in the male SMA from the antenna to the transmitter Xbee. You may need to recalibrate your compass when you travel a large distance due to a change in environmental conditions.

At this point you should have a beacon that is constantly spitting out its current heading.

**Resources/Information**

| Compass guide<br>https://learn.sparkfun.com/tutorials/9dof-sensor-stick-hookup-guide | Compass Code<br>https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2017_Robotics_Challenge/LSM9DS1MagnetometerCalibration | XbeeTX Code<br>https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2017_Robotics_Challenge/xb_Transmitter_System_Code/xb_TX_LSM |
|---|---|---|
| Declination<br>www.magnetic-declination.com<br>www.ngdc.noaa.gov/geomag-web/#declination | **Magnetometer Calibration**<br>**https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2017_Robotics_Challenge/LSM9DS1MagnetometerCalibration** | |

## 2.6 Receiver Xbee

The last step is to receive the data we started transmitting earlier. This time the receiving Xbee receives the data (a compass heading) and sends it to the receiving Fio. The Fio reads and processes the data and sends it through serial to whatever device needs that data (typically some master Arduino board).

Receiving the heading doesn't actually require building any hardware at this point. In fact, all we need to do is upload code that tells the Fio to request data from the Xbee at some regular interval and then process/send that data. The sketch is called '**xb_RX**'. It can be found on the Robotics challenge GitHub page. You should now be able to **receive from the beacon!**

**Note:** As mentioned in the Diagnostics section, it's always a good idea to upload the "XbeeCalibration" sketch to the receiver Xbee **before** "XbeeRX". The **Calibration** sketch just outputs exactly what is received, while the **RX** sketch actually processes the data. It is difficult to tell if you are receiving the correct data initially unless you run the Calibration first.

Whenever you're about to test/run the beacon/receiver system, it is best to start with something simple. Specifically, with the transmitter Xbee transmitting, upload the "XbeeCalibration" sketch to the receiver Xbee. This sketch simply prints to the serial monitor the headings being transmitted by the transmitter Xbee.

If you don't see the headings you'd expect, or if you don't see anything at all, you know that something needs to be fixed before going any further.

**Resources/Information**

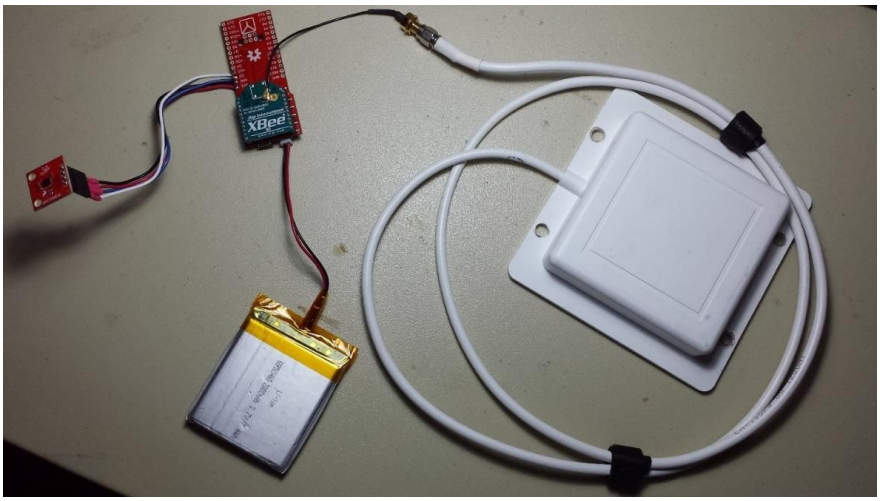
| | |
|---|---|
| | XbeeRX Code<br>https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge |

## 2.7 Putting it All Together

At last! All of the necessary steps have been taken, and all we have left to do is put all the parts together. Below is a table that summarizes what has been done and what you should be

expecting from your newly created project.

### 2.7.1 Transmitting System




**Hardware/Software List**

| Hardware | Software |
|---|---|
| Patch antenna (1)<br><br>Magnetometer (LSM9DS1 SparkFun Sensor Stick) (1)<br><br>Lithium Polymer Battery (1)<br><br>Interface Cable SMA to u.FL (1)<br><br>Xbee (Series 1; u.FL connection) (1)<br><br>Fio v3 Atmega32U3 (1) | xb_TX_LSM.ino<br><br>Arduino/Xbee Library |

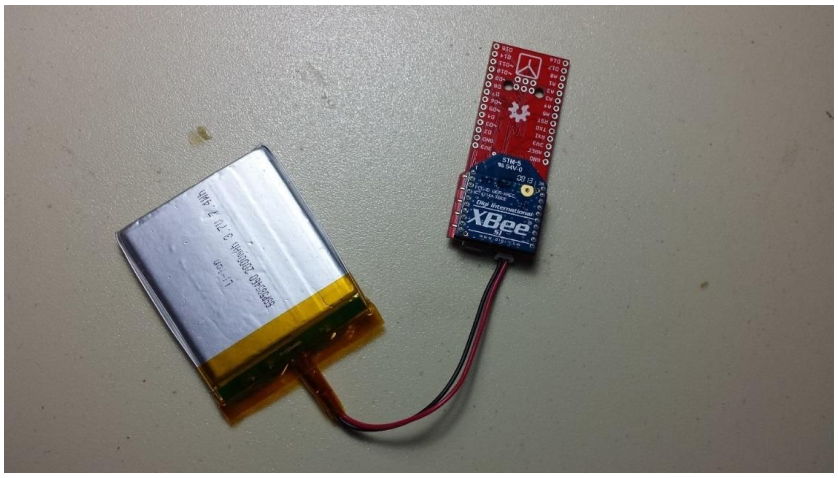
**Summary**

The compass (magnetometer) operates in continuous mode. The Fio reads the data (heading) from the compass and sends it to the Xbee. Note that the data is divided by two, making the max value 180 instead of 360. This is to decrease the amount of bits required for the transmission. The Xbee then sends the data through the patch antenna.

It is **vital** that the antenna rotates horizontally, similar to a lighthouse. This is because the

receiver will determine the heading of the beacon relative to the receiver based on which heading is received with the greatest signal strength. Without varying the signal strength (rotating the antenna) the receiver would not be able to determine which way the beacon was transmitting from.

### 2.7.2 Receiving System




**Hardware/Software List**

| Hardware | Software |
|---|---|
| Lithium Polymer Battery (1)<br><br>Fio v3 Atmega32U3 (1)<br><br>Xbee (Series 1; Trace Antenna) (1) | Xb_RX.ino |

**Summary**

The data is received by the Xbee. The Xbee then sends the data to the Fio, which collects the data and passes it through a digital low pass filter. The average maximum value after filtering is then taken as the result. This result is the heading of the beacon **relative to the receiver**. For a robot to navigate to the beacon, it would have to take the opposite of the received heading.

Also note that the received headings are between 0 and 180 degrees. You will need to **multiply by 2** in order to find the actual heading.

---

**Receiver and Transmitter Demo Video**

---

https://drive.google.com/open?id=0BwmuhZeuzCL4RXN3c05tQ2R6Ulk

# 3 Diagnostics, Debugging, Troubleshooting

When dealing with Xbee's and Fio's, even slight mistakes/discrepancies can prevent the entire system from functioning. This section outlines the steps necessary to resolve some of the more common issues.

## 3.1 Fio not visible in Arduino IDE

The Fio can be a real finicky device when it comes to making a connection with the Arduino IDE. More often than not, if the Fio isn't visible to the IDE, then the Fio is what is called **'Bricked'**. A Fio can become 'Bricked' for many reasons. Below is what SparkFun has to say about it.

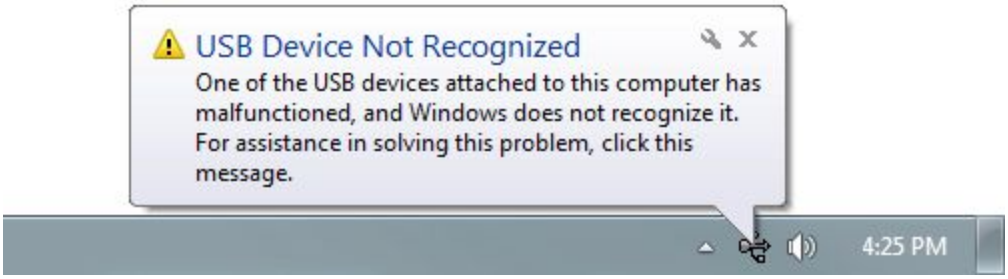
> *"The most common source of Pro Micro "bricking" is uploading code to it with an incorrectly set board (e.g. programming a 16MHz/5V Pro Micro with the board set to 8MHz/3.3V). Also, make sure your sketch doesn't mess with the ATmega32U4's PLLCSR register, or any other register that sets up USB functionality on the ATmega32U4."*

It can be really hard to predict what exactly caused the bricking. The good news is that the solution is pretty much the same regardless of what caused it to brick.

**How to Revive a "Bricked" Pro Micro (from SparkFun)**

Incorporating all of the USB tasks on a single chip is an awesome feature that makes the Pro Micro and boards like it truly unique. But it also places more stress on a single chip, and if anything goes wrong with that chip, the board becomes nearly unusable. It's not uncommon for Pro Micro's to become "bricked" and unprogrammable. But, in most cases, the bricking is reversible!

The Pro Micro will actually take code compiled for the wrong operating speed, but when it tries to re-enumerate, you'll be greeted with a notification like this:




To revive the Pro Micro, you'll need to find a way to upload a sketch to it with the board option

correctly set. We can do this with a little help from the bootloader.

First, you'll need to **set the serial port to the bootloader**. But that port is only visible when the board is in bootloader mode, so pull the reset line low twice quickly to invoke the bootloader reset feature discussed above. On Pro Micro's, or other devices which don't have a reset button, you can either use a wire to quickly short 'RST' to 'GND' twice, or wire up a temporary reset button. **While the Pro Micro is in the bootloader** change the 'Tools > Serial Port' menu to the bootloader COM port. Quick! You've only got eight seconds. On Windows, the bootloader's COM port number is usually one number higher than the Pro Micro's regular port number.

With the serial port set, we're just about ready to re-upload our sketch. But first, **double check that the board is correctly set**. Then **reset to bootloader again**, and quickly upload your sketch. Again, you'll have to be quick…you've only got eight seconds. It may help to press the Upload keybind – CTRL+U / CMD+U – immediately after resetting.

It can take a few tries to get the timing right. Since the code has to compile first, it may help to **hit upload first** and then reset.

> In my personal experience, if the computer doesn't recognize the Fio, just assume it is bricked. Follow the steps above to upload a sketch to the Fio and the issue may resolve itself. There have been times I've uploaded exclusively using the method above, just because I couldn't get the computer to see the Fio.

If the sketch uploads properly, but the Fio remains bricked, it may be an error in the code. For example, trying to access elements beyond the allotted size for an array, if the error isn't found by the compiler, will likely brick the Fio. Try uploading something like the 'Blink' example sketch to make sure it isn't your code.

## 3.2 Xbee not visible in XCTU

Another connection issue that crops up from time to time is that the Xbee will not be visible in XCTU. Luckily, this problem is usually an easy fix.

**Solution 1** – Try uploading the XbeeXCTU sketch to the Fio again. If the correct sketch isn't uploaded, XCTU will not read from the Xbee. (Uploading the code will also ensure that the Fio is not bricked)

**Solution 2** – If the **Baud Rate** configured on the Xbee is different than the Baud Rate set in the "XbeeXCTU" sketch, then the Fio will not be able to communicate with the Xbee, hence XCTU will not be able to communicate with the Xbee either. This is kind of a **catch 22**. In order to change the Baud Rate you need to reconfigure the Xbee using XCTU, but you can only get XCTU to recognize the Xbee if you have the Baud Rate configure on the Xbee **matches** the Baud Rate in the "XbeeXCTU" sketch. If you don't remember what the Baud Rate on the Xbee is, the good news is there are only a few values that it can be configured to. You may have to **guess** until

you find which value works.

## 3.3 Xbee's Not Communicating with Each Other

If you have ensured that the Fio's are functioning properly (not Bricked) and that the Xbee's are communicating with their respective Fio's, but the Xbee's are still not transmitting and/or receiving, then typically the configuration is incorrect.

Double check with section 1.3 that the configuration of each Xbee is **exactly** the same as was said. Even slight variations can keep the Xbee's from communicating with each other. If the problems persist, it is a good idea to try using different Xbee modules, just to ensure that the Xbee modules themselves are functioning properly.

## 3.4 Matlab Script for Diagnostics/Analysis (Video Tutorial)

As you may have seen already, there is a lot of data being transmitted/received between the Fio's. Although the data is simply a heading and signal strength, with so many headings being received it can be difficult to make sense of all the data. I've written a Matlab script that will read the received values from the Fio and plot them in real time. This allows you to see the data coming in graphically as it is being received. The script also supports input from multiple sources. This means you can graph both raw data as well as filtered data. This can allow you to see the effect that a particular filter is having on your raw data while seeing the raw data as well. You can find the Matlab script in the 'XbeeDiagnostics' folder linked below.
https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2015_Robotics_Challenge/XbeeDiagnostics

**IMPORTANT**

The Fio outputs data faster than Matlab can process it. Consequently, a delay is used in the Matlab code to slow down the transmission. This will result in a lower sample rate of the compass heading, which means you will ultimately lose data. Usually, this isn't an issue. If you notice that you aren't receiving as much data as fast as you'd expect on Matlab, chances are the delay is the cause. If you need every data point, then you may have to serial.print the data and plug in the raw data into a program like Excel or plot it in Matlab.

## 3.5 Misc Diagnosis and other problems

1. Xbee's have a function where they can update code on the Arduino

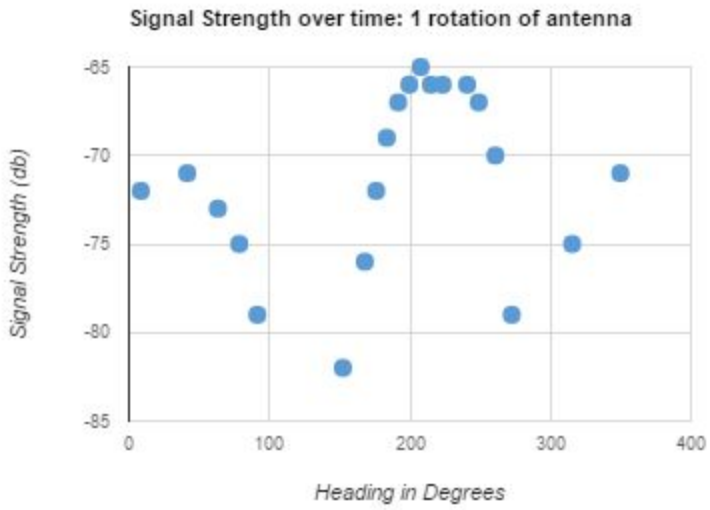
wirelessly. This means if you have two Xbees plugged in to two separate boards and powered on, code you upload from one Xbee can be transmitted to the other. Best solution is to power off one board while uploading to the other.

2. Interference. Compasses and magnetometers are extremely sensitive to conductive metals and magnetic fields. Both are found EVERYWHERE inside. Best testing conditions are outside in an open field. Make sure your robot is not causing interference as well. (Once, for example, we used steel bolts to affix the compass and were receiving incorrect readings). Breadboards can cause issues too. Avoiding interference is best done by attaching compasses with plastic bits and elevating them above your robot or breadboard far from electric/metal components or motors. See magnetometer testing code in 2016 code dump (page 9).

# 4 XBee Data Sheet

http://www.digi.com/pdf/ds_xbeemultipointmodules.pdf

# 5 Transmitter Receiver Test Data




This graph represents what the receiver Xbee reads during one rotation of the antenna. The peak (approximately 200 degrees) is the heading that the xb_RX code should filter and write as your official heading.

The range of our system at full power was capable of a consistent signal from over 100 feet using the patch antenna on a 3.7 volt Li-Po battery.. The transmitter **PL** setting in XCTU can be pushed up or down to adjust available range.

*The entirety of our data obtained from testing is here:
https://docs.google.com/spreadsheets/d/1WsT3GYtCndcYqy4XCja2pAqTdEteAyzcTjxSbw0MiOk/edit?usp=sharing

It contains multiple rotation data, range testing etc.

# 6 Parts to Purchase

**List of items to buy to make your own Xbee-Beacon system**

Transmitting System:

| Item | Purpose | Link |
|---|---|---|

| | | |
|---|---|---|
| Directional Patch Antenna | In order to transmit reliably | http://www.l-com.com/wireless-antenna-24-ghz-8-dbi-flat-patch-antenna-4ft-sma-male-connector |
| Magnetometer (LSM9DS1 SparkFun 9DOF Sensor Stick) | It is a 3-axis magnetometer, use as a digital compass to sense the direction | https://www.sparkfun.com/products/13944?_ga=1.267265742.1340058446.1441581116 |
| XBee 1mW U.FL Connection - Series 1 | The transmitter Xbee will send the direction it is facing | https://www.sparkfun.com/products/8666 |
| Interface Cable SMA to U.FL | Interface cable from XBee to antenna | https://www.sparkfun.com/products/9145 |
| Lithium Polymer Battery | Power source | https://www.sparkfun.com/products/8483 |
| Arduino Fio v3 Atmega 32U 3 | Board used to house XBee and interpret Data from compass | https://www.sparkfun.com/products/11520 |
| Project Box | | |

Receiver System:

| Item | Purpose | Link |
|---|---|---|
| Arduino Fio v3 | Board to connect the Xbee receiver. However, you can hook up Xbee's w/ a breadboard. | https://www.sparkfun.com/products/11520 |
| XBee 1mW Trace Antenna - Series 1 | Works as receiver and transmitter | https://www.sparkfun.com/products/11215 |
| Optional LCD Screen | Display data received | https://www.sparkfun.com/products/9053 |
| Lithium Polymer Batteries | Power Source | https://www.sparkfun.com/products/8483 |

# 7 2016 Beacon System Code Repository

https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2017_Robotics_Challenge

Note: This GitHub contains all of the code we used to develop the beacon and receiver systems. The RX and TX codes are for the receiver and transmitter (linked previously). The LSM9DS1MagnetometerCalibration is code that can be used to get the calibration values for your compass and to make sure your compass works properly.

# 8 Old Code Dump 2015 and before

https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2016_Robotics_Challenge

https://github.com/Colorado-Space-Grant-Consortium/Robotics_Challenge/tree/2016_master/2015_Robotics_Challenge

***Does not apply to 2017 challenge. This is old code! It is included because it could potentially be used retroactively.

# 9 Change log

| Version | Date | Changes |
|---|---|---|
| V1 | -- | Initial Release |
| V2 | -- | Added changelog. Changed cover-art from Series 2 XBee to Series 1. Fixed links in parts list and added U.FL to SMA connector. |
| V3 | 11/16/2016 | Reorganized the document to have a more logical flow. Also reformatted the document to be more centered. |
| V4 | 02/07/2017 | Reworked for LSM9DS1 SparkFun 9DOF Sensor Stick. Added more information about compass |