# Aerosmith Maze

Texas A&M University

HRG

# TABLE OF CONTENTS

# Overview

For this project, the goal was to build a maze that a small metal bb pellet could navigate. This was done mainly through the use of teensy microcontrollers, ultrasonic sensors, ESC motors and xbee radio modules.

The maze outline, resembling a smith chart, was carved into a large, round piece of plexiglass with a CNC machine. Six ESC motors are attached to this piece of plexiglass, which tilt the maze in various directions, based on how fast each motor is spinning. This is how the bb navigates the maze, as each motor receives data from an ultrasonic sensor which tells it how fast or slow to spin. Each ultrasonic sensor will output a distance measurement based on how close or far an object is to it. The distance value that each sensor outputs is sent to a teensy microcontroller, which is programmed to send the data through an XBee module. The data is sent wirelessly to another XBee on the receiving end, which is connected to another teensy. This teensy is attached to the plexiglass maze, and is programmed to control all of the 6 motors, based on the different values it receives from each of the 6 sensors.

The maze is designed to be controlled by 6 different people at one time, with each person controlling one motor by moving their hands in front of an ultrasonic sensor. Based on how the teensys were programmed, the closer your hand is to the ultrasonic sensor, the faster the motor will spin. Therefore, to tilt the maze up in one direction, one person will have to move their hand closer to the ultrasonic sensor that corresponds to that specific motor. The maze is attached to a ball and socket joint which is tethered to a tripod, which allows it the freedom to rotate in any direction, without the maze flying away due to the lift of the motors. This project was designed to encourage people to work together to complete a task, and the maze is best navigated when the speed of one motor is changed at a time.

This document will outline the materials needed and the steps that should be taken in order to build the Aerosmith maze, and a list of references that was used will be included at the end.

# Materials

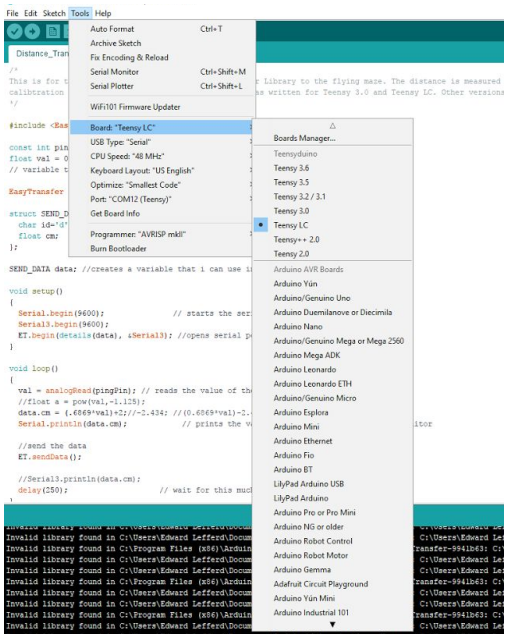
- 1 Round piece of plexiglass
- 6 Turnigy Aerodrives motors (D3536/9 910KV)
- 6 Electronic Speed controllers (Hobby King 30A UBEC)
- 6 Propellers: 3 1045R, 3 1045
- 6 Metal Channels
- 3 3D printed legs
- 6 3D printed motor mounts
- 6 Metal cross brackets for motor mounts
- 1 power supply (BK precision 1900)
- 7 breadboards 16.5 cm x 5.5 cm
- 7 XBee PRO S1 With Antennas
- 7 Teensy LC or Teensy 3.0
- 7 Teensy to XBee Adaptor kits
- 6 parallax 28015 ping ultrasonic range finder or XL-MaxSonar –EZ/AE Ultrasonic range finder
- XBee Explorer Dongle
- 1 ball and socket joint
- Assorted nuts and screws

# Software

- Arduino 1.8.3 (https://www.arduino.cc/en/main/software)
- Teensyduino (https://www.pjrc.com/teensy/teensyduino.html)
- EasyTransfer Library (https://github.com/madsci1016/Arduino-EasyTransfer)
- XCTU (Source 8)
- AutoCAD Fusion 360 (https://www.autodesk.com/products/fusion-360/overview)
- Anaconda
- Pycharm
- PySerial

# Setting up the Software

You will need to download and install Arduino 1.8.3 onto your computer if you do not already have Arduino installed. Leave all settings as default. Open up Arduino and try uploading a blank code to your device, or maybe a simple blink program. Make sure the right board is selected by going to Tools->Board:-> and select the device you are trying to upload the code to, such as Arduino Genuino/Uno, Teensy LC, or Teensy 3.0. Also under Tools->USB Type:-> make sure that “Serial is selected and under Tools->Port-> make sure the specific device that is being uploaded to is selected. When using the Arduino it is necessary for the Rx1 and Tx1 pin to be clear and not connected to anything when uploading code to the device or an error will appear.




Pictured Above: Tool Bar Settings in Arduino

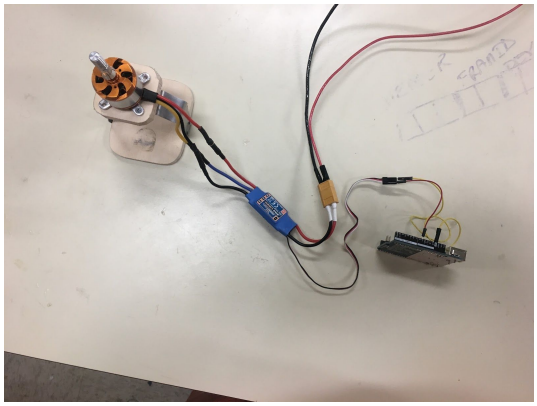
For Teensy 3.0 and Teensy LC it is necessary to have the Teensyduino downloaded. Do not change any of the settings or file locations when downloading Teensyduino, everything will be automatically setup for you. When using Teensy 3.0 it is necessary to change the frequency from 96 MHz (overclocked) to 48 MHz to avoid damaging the device by going to Tools->CPU Speed:->

The library EasyTransfer will need to be downloaded for this project and installed in the correct file. To do this find your Arduino folder stored in your hard drive at a location such as c/Program Files (x86)/Arduino/library and copy the unzipped EasyTransfer library into the library folder. If an error occurs about the library when trying to run/compile the code it may be necessary to remove the individual EasyTransfer libraries from the main unzipped file and place them in the library file separately.

To configure the XBees you will need to have XCTU downloaded and installed on your computer. Download the latest version of XCTU and install it on your computer without changing any of the default settings.

## Motor Testing

To test motors, wire pin 9 of the Arduino to the signal wire of the ESC and the ground wire of the Arduino to the ground wire of the ESC. Also, attach the servomotor and the power supply to the ESC.



*Pictured Above: Wiring of Servo Motor for testing*

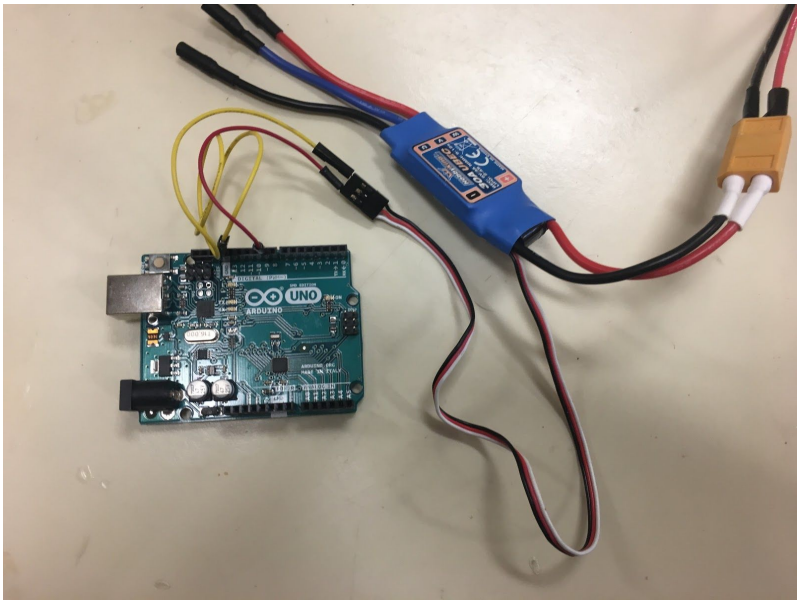
Upload the code to the Arduino and open the serial monitor. It can be found under the filepath Arduino Code->Test Code->Motor_Test->Motor_Test.ino. Type in a

value and your motor's speed will gradually rise or fall to match your entered value.

## Programming the ESC's

The arduino file underneath the filepath: Arduino Code->Program ESC->ESCProgrammer->ESCProgrammer.ino, is designed to program the electronic speed controllers so that when the ESC's are controlling the speed of the motors, the motors will turn on at roughly the same value and spin at similar rates.

1. Plug in an arduino to your computer and attach the ground port on the arduino to the ground pin (black wire) on the ESC PWM wires and port 9 on the arduino to the signal pin (white wire) on the ESC PWM wires. Also, attach power wires to the ESC.

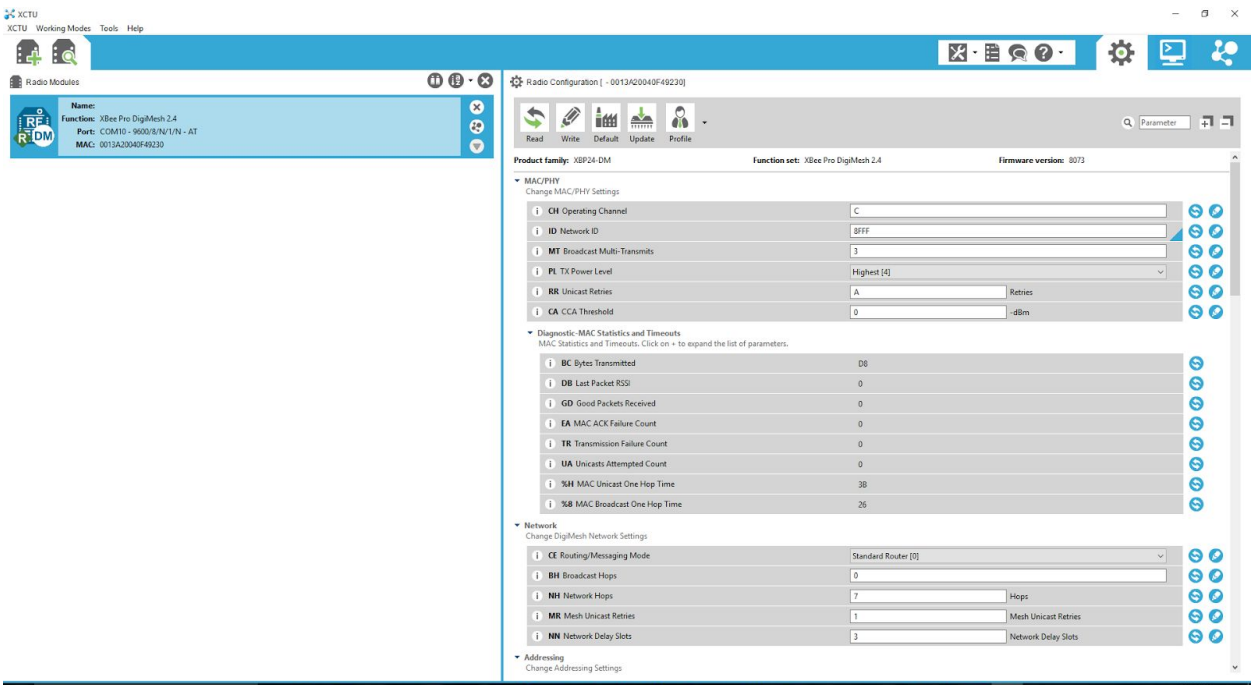



Pictured Above: how to configure wires to ESC to program

2. Before uploading the code, disconnect and reconnect the power supply to the ESC, this will allow the ESC to be programmed. Upload the code located in the aforementioned file path and upload this onto the board. Open the serial monitor and wait for the text "Done" to appear.
3. Repeat these steps on all the ESC's until all are programmed.

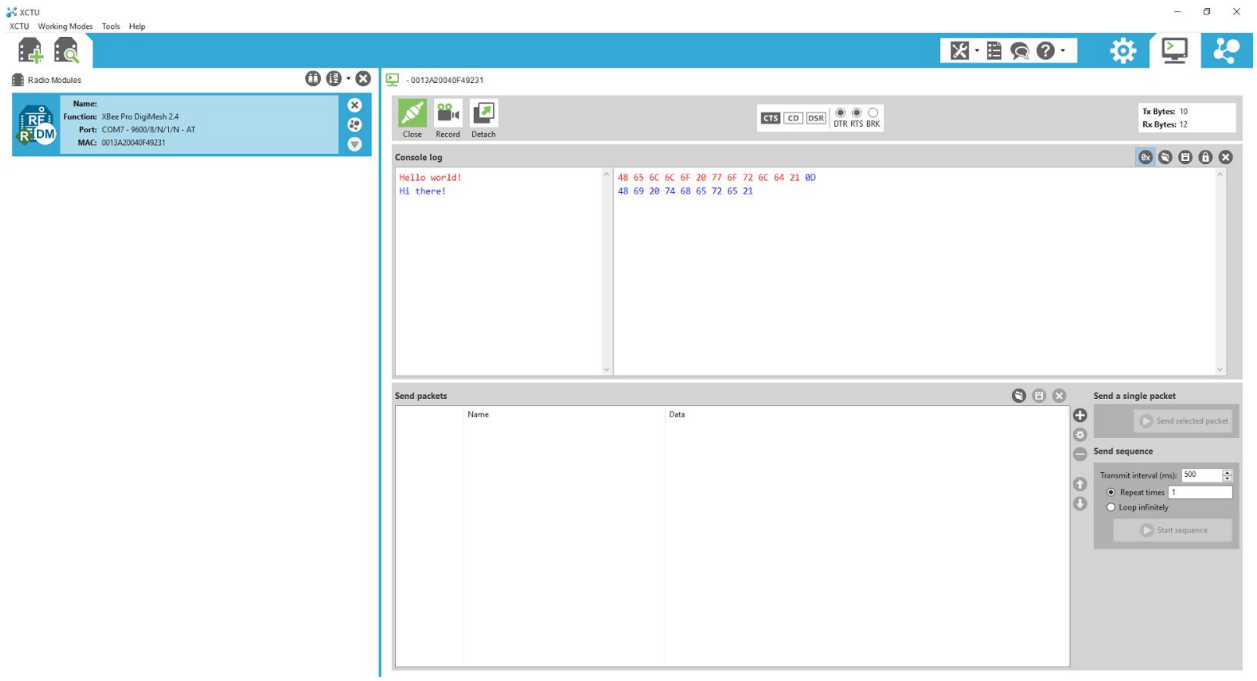
# XBee Configuration/ Setup

Once XCTU installed connect the XBee using the XBee Explorer Dongle and open XCTU. Click on “Discover Radio Modules connected to your machine” and select the USB serial ports that show up. Click Next and then Click finish. If the device does not show up the first time you may need to select more Baud rates before clicking finish. Click “Add selected devices”, and then click on the selected device to configure it and access the console. A window my appear that says that the device needs to be reset, in which case the pin in the bottom of the board marked “RST” must be connect to the pin marked “GND”. If the XBee does not show up when searching for devices, and resetting it does not work the XBee may be broken and it would be best to try a different one. The configuration settings used are pictured below.




Pictured Above: The Settings window for XBee module displaying the settings we used

The only thing that was changed from the default was changing the Network ID from 7FFF to 8FFF. To communicate between two XBees select the console tab located next to the settings tab and click the “Open/Close” button. This allows for you to see the messages sent by other networks on your console screen and send messages from the console screen to other XBees in the same network.



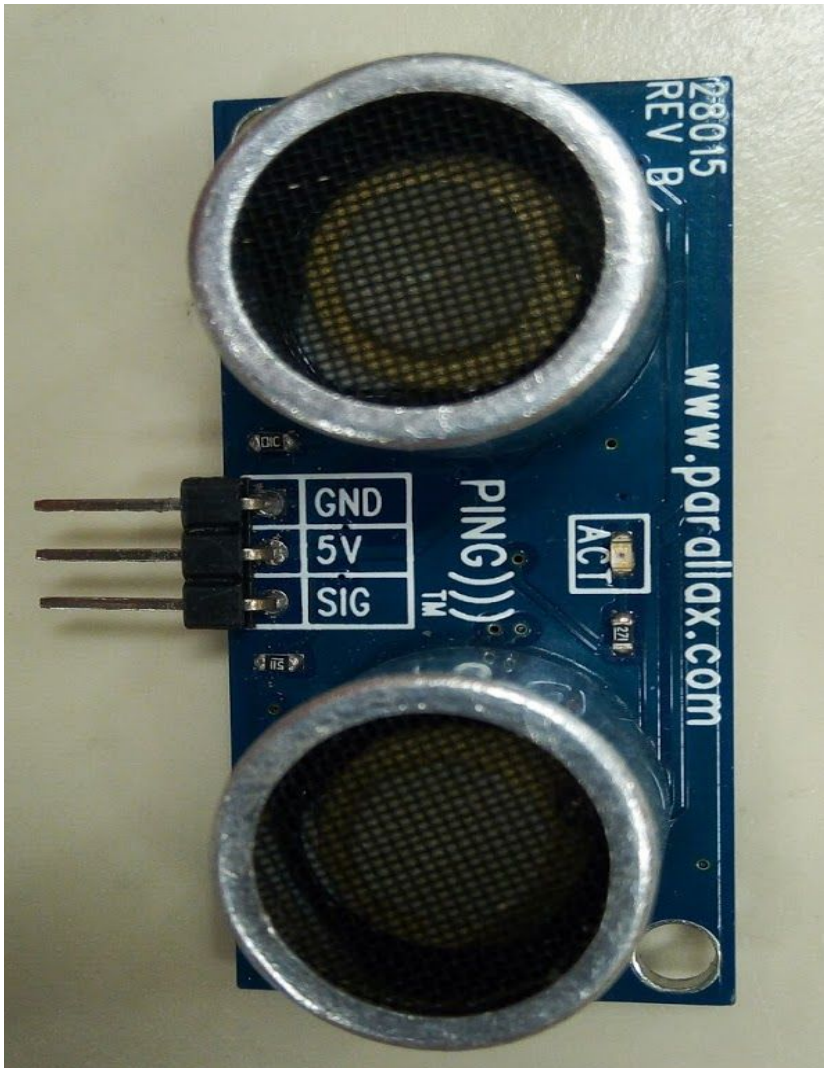

Pictured Above: Console window showing two XBees communicating with each other. The red text is from the other XBee, while the blue text is from the XBee that is connected to your device.

# Distance Transmission Configuration/ Setup

## Distance Sensor Setup

Using a Teensy 3.0 or Teensy LC connect the distance sensor that is being used according to the following setup: Connect the GND pins to each other, the 5V pin to Vin on Teensy 3.0 or 5V on Teensy LC or Arduino Uno. Connect the signal pin (for the parallax 28015 ping ultrasonic range finder it is labeled SIG and for the XL-MaxSonar –EZ/AE Ultrasonic range finder it is pin 3) to an analog pin (In the code used in the project it is pin 14 for Teensy 3.0 and Teensy LC).

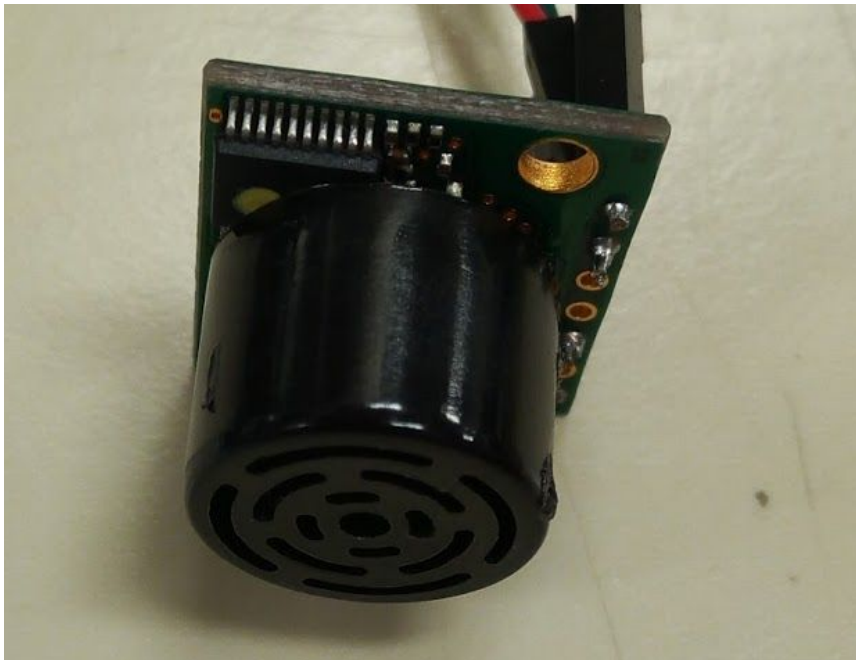
For the parallax 28015 ping ultrasonic range finder we found code for measuring the distance at this link https://www.arduino.cc/en/Tutorial/Ping . This code relies on sending a pulse signal through the sensor and measuring the delay of the returning signal, and converting the time using the speed of sound to convert it into inches and centimeters. The code used in the project used the function to send out the pulse signal and converted it to centimeters. Pictured below is the sensor used.




The code used in the project can be found at the filepath Aero_Maze_Ouija->Arduino Code->Drone Code->Distance_Transmission_28015->Distance_Transmission_28015.ino. This code provides accurate readings from 5cm to 80cm. If you are using this code for a

board other than Teensy 3.0 or Teensy LC you may need to comment out lines involving Serial3 as this will cause an error when compiling. The distances will be displayed on the serial monitor.

For the XL-MaxSonar –EZ/AE Ultrasonic range finder we created our own distance conversion code by taking in the signal using analogRead() to read in data from the pin connected to the sensor and printing the value to the serial monitor. We recorded the values produced by analogRead() along with the distances that we placed an object at from 95 cm to 20 cm every 5 cm (a limitation of this sensor is that it can not measure anything below 20 cm) plotted it to determine an equation that fit it, coming up with distance = (.6869*val)+2 where val is the integer value that is read in from analogRead(). This mapping function was created for the Teensy LC and Teensy 3.0. When used with a different device, such as an Arduino Uno, the values produced by analogRead() are different and require a new mapping function to be created. Below is a picture of this sensor.

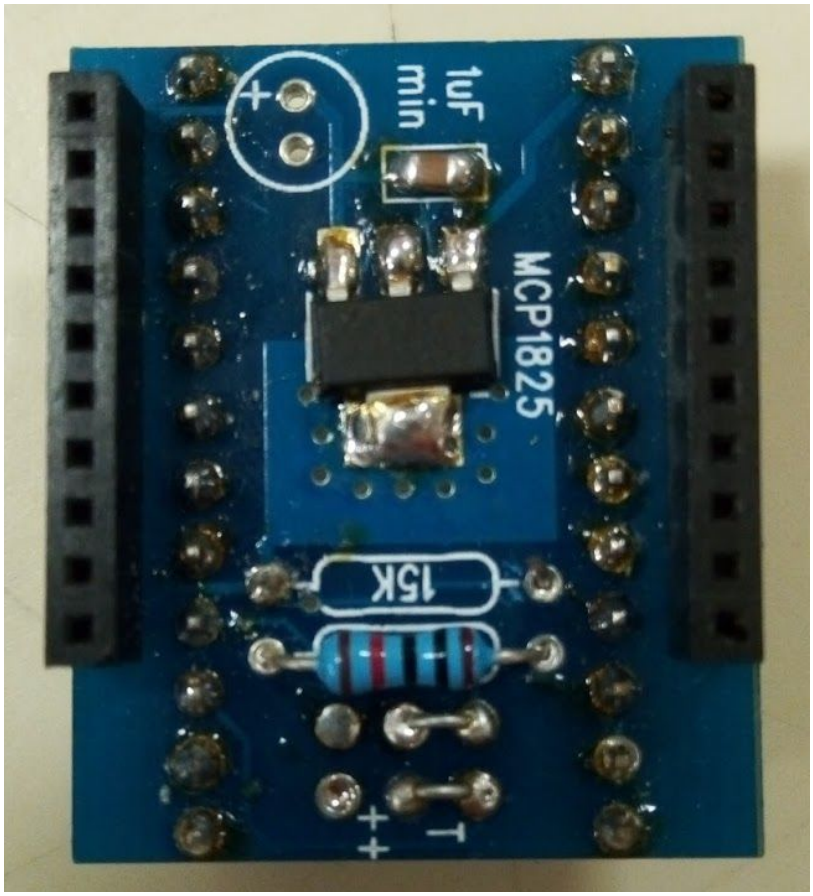


The code can be found at Aero_Maze_Ouija->Arduino Code->Drone Code->Distance_Transmission_MAXSONAR->Distance_Transmission_MAXSONAR.ino. This code will provide fairly accurate readings (+- 3 cm) from 20 cm to

95 cm. If this code does not seem as accurate as you would like you will need to go through the mapping process described above.

XBee Adaptor Connection

This part requires an XBee adaptor board to connect the XBee to the Teensy. The one used in this project can be found here https://www.pjrc.com/store/xbee_adaptor.html. (Note: this page recommends that you do not use these boards for any new projects, but we used them in this project as we already had them). Follow the steps outlined in this page to configure these for Teensy 2.0. It is important to follow the instructions outlined in the page for using it with Teensy LC or 3.0 as the boards were designed for the regulator chip to handle the 5V from the Teensy 2.0 or 2.0++, which means that for using the Teensy LC or 3.0 requires the 15k resistor to not be connected. Below is an image an adaptor showing the configuration used in this project. The 15k resistor is removed and the two connections below the 10k resistor go from the middle pins to the left pins (without the ++).

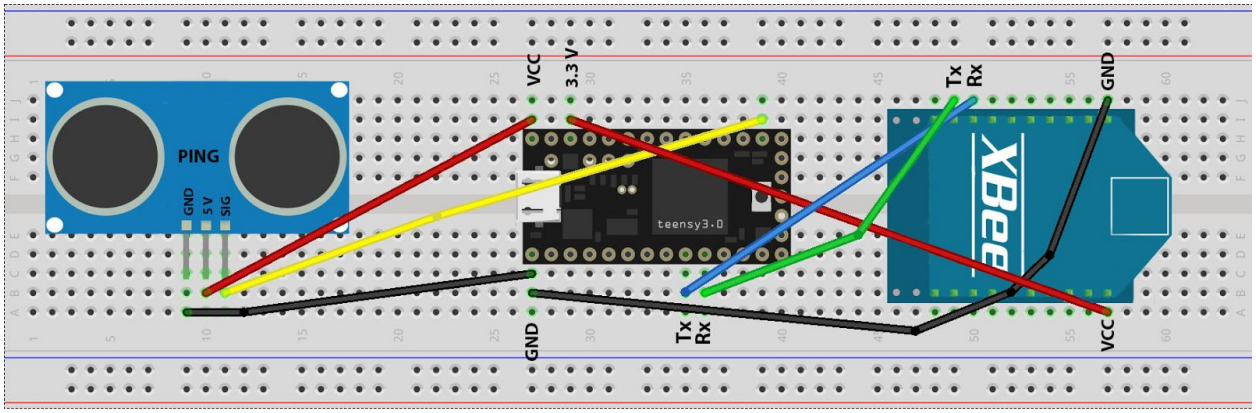



In the code you will also need to include the lines XBee xbee = XBee(); and Serial3.begin(9600). A test code we used to verify that we could communicate through the Teensys and XBee setup can be found at the file-path Aero_Maze_Ouija->Arduino Code->Test Code->communication_Test->communication_Test.ino. This code allows you to send messages to and from the serial monitor. It serves as a simple test and does not require the sensors to be setup as it sends and receives strings of text.

When you examine this code you may notice that it includes both Serial and Serial3. Serial is connected to the serial monitor on the Teensy/Arduino that is being used. Serial3 is used for Rx3 and Tx3 which are the data that is being sent out through the XBee and received through the XBee. Rx3 and Tx3 are used with the Teensies because of the specific adaptor board used requires that Rx3 and Tx3 be used if using anything higher than a Teesny 2.0 or 2.0++. If using a different

type of Adaptor board which allows for the use of Rx1 and Tx1, then use Serial instead of Serial3.

To connect the adaptor board-XBee setup to the Teensy 3.0 or Teensy LC follow the diagram below:




To complete the distance transmission setup simply use the code from the Distance Sensor Setup section and make sure that the XBee and distance sensor are connected as outlined above.

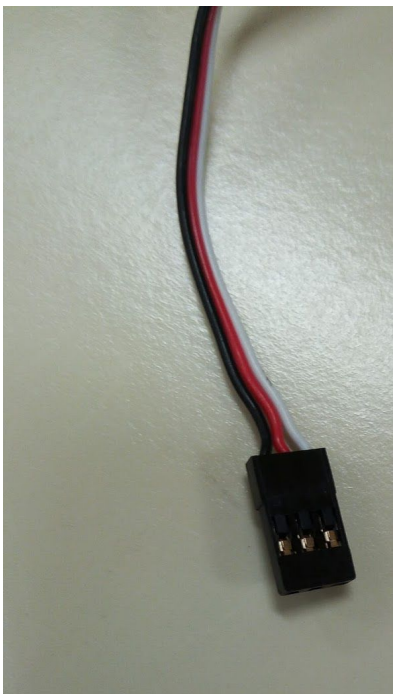
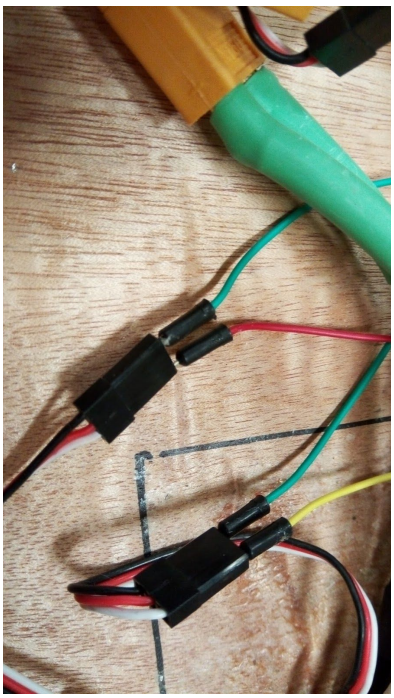
# Configure Distance Receiving/Motor Control Setup

## XBee Adaptor Connection

See XBee Adaptor Connection under Configure Distance Transmission Setup

## Motor Connection Guide

Each esc has a 3 input connection: leaving the middle red wire empty, the black wire is connected to the ground of the Teensy and the white wire is connected to the signal pin on the Teensy (in our code we used pins 2-6 and 9).

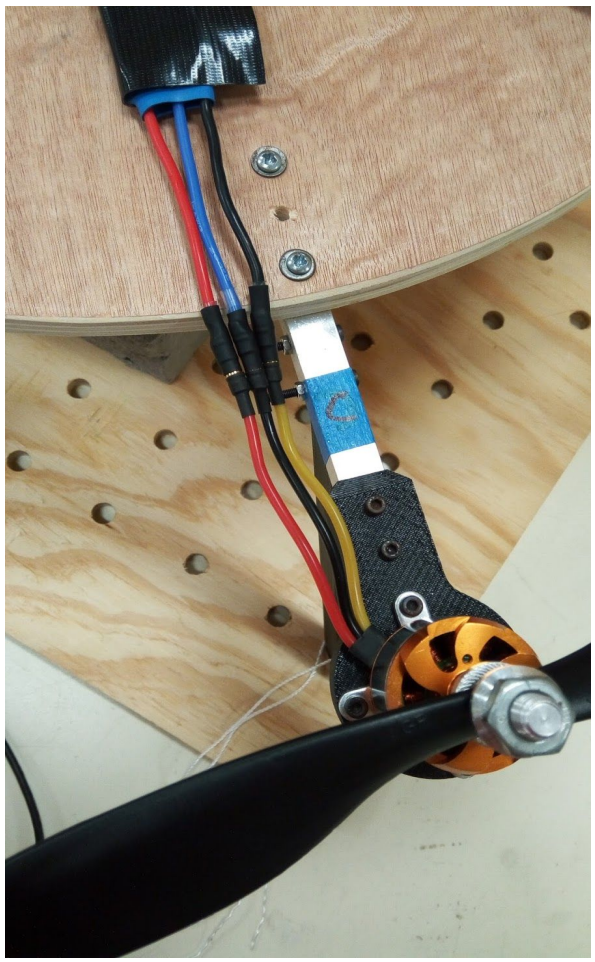



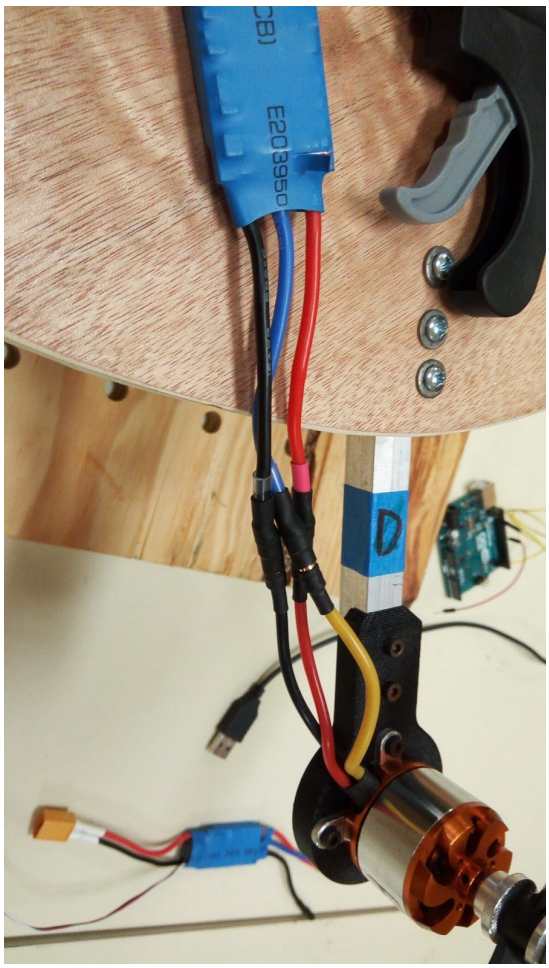

***Pictured Left****: the three input connection: This is used in programming the esc and will be connected to the Teensy.*
***Middle****: This shows the wires from the Teensy connected to the three input connection (note that the green wires in this image represent the ground wire).*
***Right****: The esc with the input side (top) and output side to be connected to the motors (bottom)*

The code for receiving the packets of data and sending out a signal to the motors can be found under the file-path Aero_Maze_Ouija->Arduino Code->Drone Code-> _6_Motors_Recieving->_6_Motors_Recieving.ino.

The red and black power cables on the input side of the esc will be connect to power and ground of the power supply respectively. On the output side of the esc the connection cables should alternate the connection between the black and colored wires, with red still going to red.

So the setup would be red-red,blue-yellow,black-black; and for the next esc-motor setup it should be red-red,black-yellow,blue,black. This can be seen in the picture below.




**Pictured Left**: Showing the connection red-red,blue-black,black-yellow.
**Pictured Right**: Showing the connection red-red,blue-yellow,black-black.

It incorporates the easy transfer library to receive the distances and an identifying character from each setup, and then case structures to select between the different characters and control individual motors.

# Designing The Maze

Using the software Autodesk Fusion 360 we designed a marble maze that is based upon a Smith chart. Using the parameters menu, one can easily reconfigure the size and dimensions of the maze to fit the almost any size board. One can easily find the software underneath Autodesk's official website and receive a one month free trial to quickly generate an STL underneath any size they wish. The STL we have included in our files is a maze designed for a 14 inch diameter round piece of fiberglass.

## Modifying the Maze

In Autodesk Fusion 360, open the parameters menu and the main parameters the user will want to modify will be:

- body_diameter - Set the diameter of your wood
- trough_width - Set the width of the guiding structures of the maze
- trough_depth - Set the depth of the guiding structures
- end_hole_depth - Set the depth of the center hole in the maze
- distance_guide - Is approximately the distance between the center of the neighboring guiding structure and the guiding structure that the former cuts off. One should tinker with this parameter to get the correct dimensions for their maze.

Mounting the Maze

Mounting the maze to the tripod requires a lot of different parts that attach to each other. First, a 3D printed tripod mount will attach to the tripod, which will have the ball and socket joint attached to it. A metal rod screws into the ball and socket joint, which will be attached to the bottom of the maze. A 3D printed funnel will then go onto the metal rod, and have holes for 6 different wooden dowels, one for each motor. Each motor is connected to the maze with a metal bar, and to this bar there will be attached a dowel holder. This will hold the wooden dowel for each motor that attaches to the funnel. Each part has been pictured below with the included dimensions. The dowel holder, funnel, and tripod mount was 3D printed, while the ball and socket joint and wooden dowels were purchased at a store.

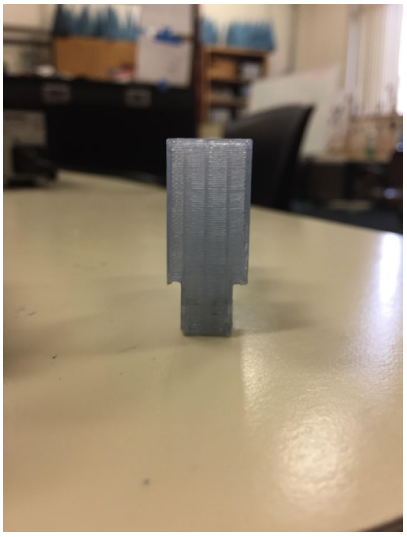
Dowel Holder:

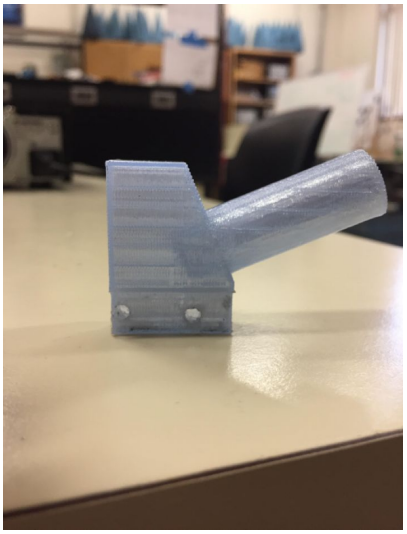
Inner Diameter: 1cm

Height: 3.6cm

Lower Width: 11.5 cm

Angle of tube = 23˚ from horizontal





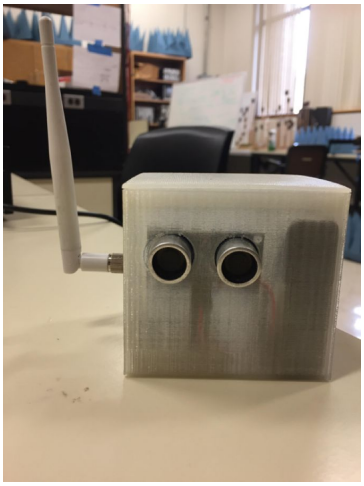
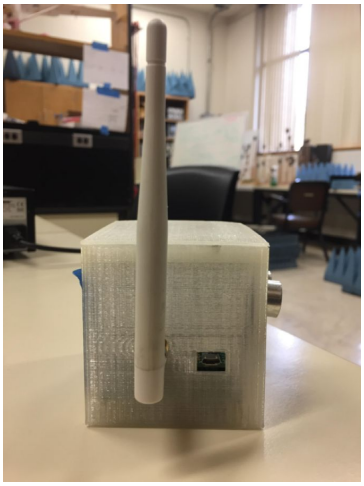
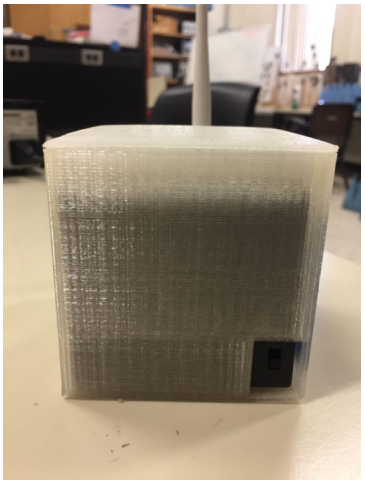
# Final Product

Sensor Boxes

Length: 8.5 cm

Width: 7.4 cm

Height: 7.5 cm

*6 Sensor boxes were 3D printed. Each box contains an ultrasonic sensor and an XBee with an antenna, both of which are connected to a teensy.*

# References

Websites for purchasing materials:
6 Turnigy Aerodrives motors (D3536/9 910KV)
(https://hobbyking.com/en_us/turnigy-d3536-9-910kv-brushless-outrunner-motor.html)
· 6 Electronic Speed controllers (Hobby King 30A UBEC)
(https://hobbyking.com/en_us/hobby-king-30a-esc-3a-ubec.html)
· 6 Propellors: 3 1045R, 3 1045
(https://www.newegg.com/Product/Product.aspx?Item=9SIA2RP0XW3805)
· 6 Metal Channels
(http://www.johnsonrollforming.com/display.php/display/A1/category/2)
6 Metal cross brackets for motor mounts
(http://www.crlaurence.com/crlapps/showline/offerpage.aspx?ProductID=11909&GroupID=14911&History=39325:22043:7146:14894:14906:14894&ModelID=14911&pom=0)
· 1 power supply (BK precision 1900)
(http://www.bkprecision.com/products/power-supplies/1900-1-16v-60a-switching-dc-power-supply.html)
· 7 breadboards 16.5 cm x 5.5 cm
(https://www.itead.cc/breadboard-16-5-x-5-5cm.html)
· 7 XBee PRO S1 With Antennas (https://www.sparkfun.com/products/8742)
· 7 Teensy LC or Teensy 3.0 (https://www.pjrc.com/teensy/teensyLC.html)
· 7 Teensy to XBee Adaptor kits (https://www.sparkfun.com/products/13311)
· 6 parallax 28015 ping ultrasonic range finder or XL-MaxSonar –EZ/AE
Ultrasonic range finder (https://www.parallax.com/product/28015)
· XBee Explorer Dongle (https://www.sparkfun.com/products/11697)