Created by Anthony Scopazzi

Spring 2020

**Contents**

What is a CTD? Page 1

Why CTD’s are used

Example plots with explanation

Getting Started Page 2

Equipment and Software

Troubleshooting Arduino IDE - COM port, other problems

Intro Breadboard and Wiring

Intro Soldering

Intro Coding

Temperature Sensor Page 4

Wiring

Collecting temp calibration data

Parsing Data in Excel

Example final graphs Salinity Sensor Page 9

Explanation

Wiring

Calibrating salinity

Pressure Sensor Page 10

Explanation

Wiring and Testing

Micro SD Card Breakout Board Page 11

Explanation

Wiring

Waterproof Housing Page 12

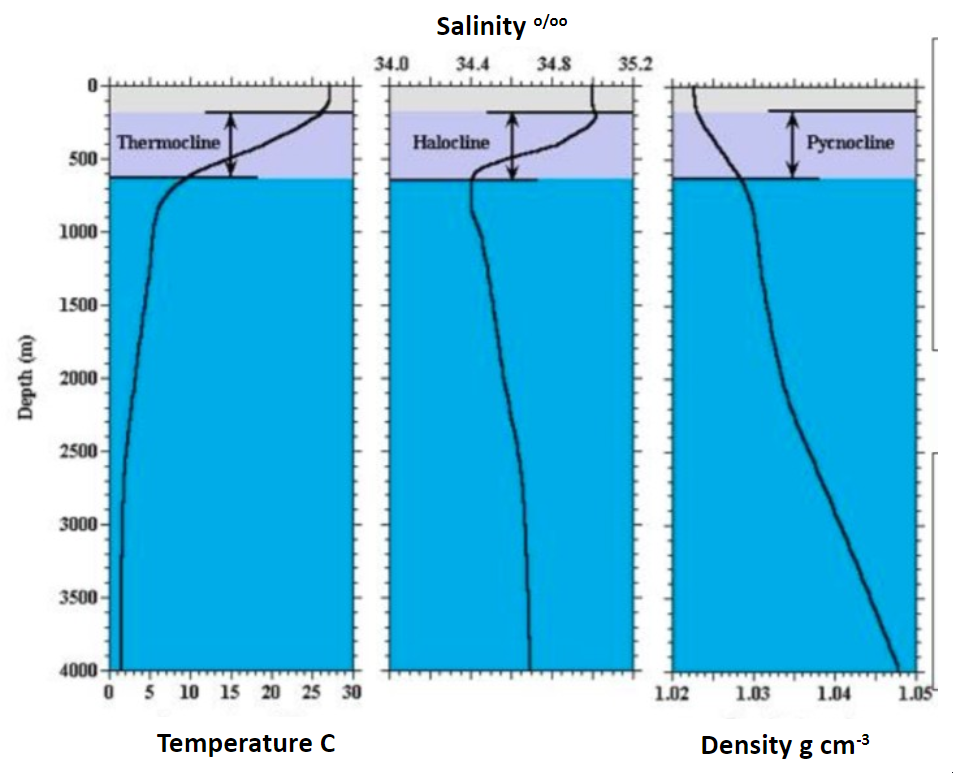
3D printed equipment tray

Making the housing

Arduino Programs Page 13

**What is a CTD?**

CTD stands for Conductivity, Temperature, and Depth as that is what a CTD measures. Basically, what the instrument does is measure the salinity and temperature at a depth, which can be displayed in graphs such as ones below.



This instrument, which supplies what may seem like simple datapoints, can help answer a lot of questions about the water column deployed in. Using the information from a CTD it is possible to infer further and answer many questions such as: Is the water column stable and not mixing, or unstable and prone to mixing? What is the depth of the thermocline or halocline? How would that effect the life living in the epipelagic (the top photic zone)?

Helping to answer questions like that enables scientists to better understand the entire system they are studying with a relatively simple device. Usually, these instruments cost $5,000 or more, but through this manual we will build one for less than $500.

**Getting Started**

Board and Sensors

Arduino Uno R3 ($20)

Dallas Temp Sensor DS18B20 Temperature Sensor (~$10 each = $30 total)

Atlas Scientific EZO-EC Embedded Conductivity Circuit ($240 for Conductivity K Kit)

SparkFun Pressure Sensor Breakout - MS5803-14BA ($60)

Total Cost = $350 plus cost of wires, etc.

Misc Equipment

Soldering iron and accoutrement

Heatshrink

Solid core wires

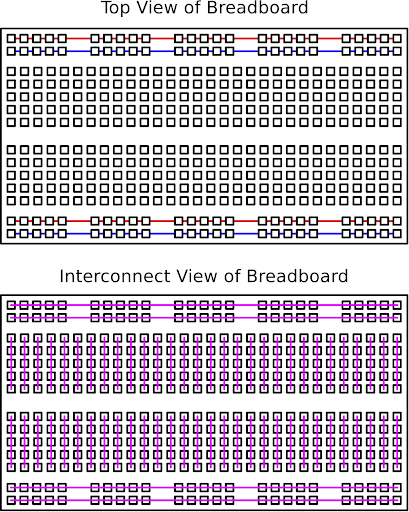
Wire strippers / cutters  
Safety glasses

Software

Arduino IDE software - only works on PC

<https://www.arduino.cc/en/main/software>

Unable to connect to board - COM PORT error

1 - make sure USB is plugged into computer

2 - Navigate to ‘Tools’ --> ‘Port’ --> select ‘Arduino UNO’

3 - Try a different cable, restarting the program/computer, unplug/replug

4 - Ask teacher

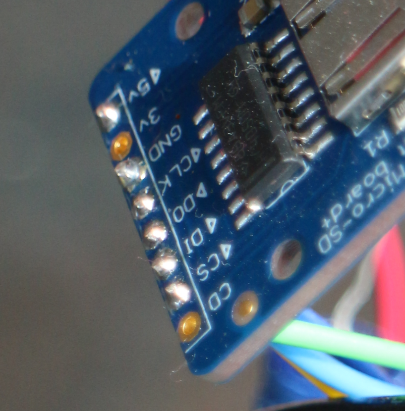
**Intro Breadboards**

Breadboards are for temporary testing of wiring and are connected as shown right. The holes down the sides with red and blue colors go all the way down the board, to be used as hot (red) and ground (blue). The inside is connected as shown and is where components go. Common errors are not putting the wires into the row with a component and mixing up hot and ground connections. Being consistent with colors is also incredibly helpful.

**Intro Soldering**

Safety – Always wear safety glasses when soldering! Small pieces of flux or solder can fly off and get in your eyes. Just wear them – it’s not worth the potential for eye problems. Try not to breath in the solder smoke. It’s the flux on the solder burning. It won’t kill you but is not good for you either. Just don’t breathe it the best you can.

General – Given the choice (and you have a choice) but consistent with wire colors. Future you will thank past you. Red = hot, power, voltage | Black (or blue if don’t have black) = ground

Stripping and Soldering – Use wire strippers of the correct size and expose only about 5mm of wire and before attaching the wires slip on some heat shrink. Put into the board and solder. It should look something like the photo on the right. Clip the extra wire and inspect to be sure you did not bridge any connections. If you did unsolder and redo.

**Intro Coding**

Common errors

Not having a “;” at the end of lines

Baud rate in code different than in serial monitor

Missing a capital letter, or having one where there shouldn’t be

Helpful commands

//To leave a line in, but not have it be ran code, comment out the line with two “//” before it

#define rx 2 //define what pin rx is going to be, “tx” would be transmit

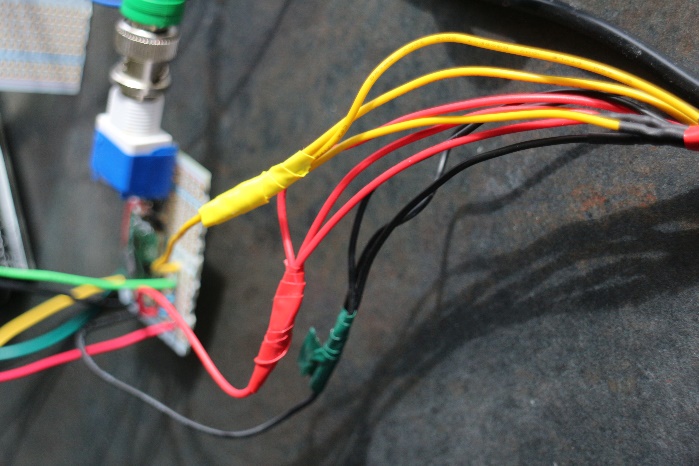
Serial.begin(9600); //sets baud rate for the hardware serial

//leave concise notes in your code, ideally in the same line, to make sure you can go back and know what it’s doing. Do it in the same line to keep it readable.

Use the code in the appendix or downloaded zip file for individual sensors and final code

**Dallas Temp Sensor (DS18B20 Temperature Sensor)**

<https://lastminuteengineers.com/ds18b20-arduino-tutorial/>

**Explanation**

While these are cheap, they do work well, especially with three averaged together. By using three in this manner we can get a very accurate picture of the real temperature – the downside being they do not have the fastest response rate. This is compensated for by lowering slowly into the water when deploying.

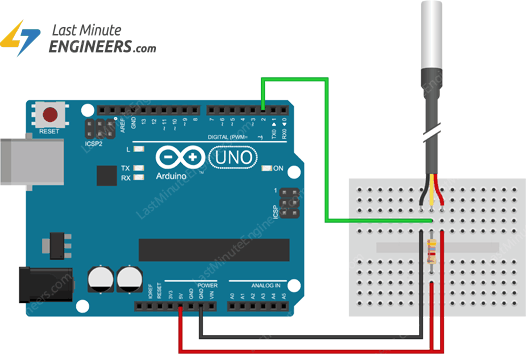
**Getting Sensor Ready to Wire**

Three individual sensors for an array – each has three wires. Strip and solder each wire from the sensor to a solid core wire.

As you have three sensors and one pin on the Uno you must connect the matching colors of each sensor into one wire that gets plugged into the board, as shown to the right. Be sure to color each wire consistently and to heat shrink the exposed wire. If you don’t have heat shrink big enough use electrical tape as in the photo (heat shrink is preferable.

**Wiring Temp Sensors**

Be sure to use a 5k ohm resistor (4.7k-5.2k)



**Testing Temp Sensors**

Wire them as in the diagram above, then use the code “tempSensor\_test.ino” to make sure they work. Troubleshoot with each other, or the teacher, if they do not work/display correctly. One sensor might display -127 as its reading; this mostly likely means there is a problem with one of your wired connections. One of the sensors may have a wire soldered poorly or it may not be fully connected in the breadboard. Once everything’s working, start calibrating!

**Collecting Temp Calibration Data**

The idea is to heat all the sensors at the same heat and see how they react in comparison to each other.Gather the following materials:

Arduino and sensors Bunsen burner with stand

Yellow temp sensor Beaker to fill with water

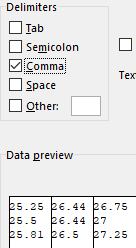
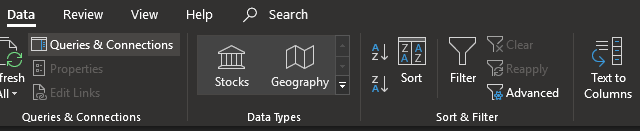
1. In the room temperature tap water (about 20C) place the three sensors so they *do not touch the bottom* of the glass. Do the same with the yellow temp sensor.
2. Once you turn on the bunsen burner, *clear the serial monitor* and record a reading of the yellow every 10 seconds until the water reaches about 90C. It should take at least a few minutes.
3. As soon as the water reached about 90C unplug the Arduino from the computer. This will stop the serial monitor from moving so you can copy all the information to Excel.
4. Put the yellow temperature sensor data at the appropriate ten second intervals in your data.

**Parsing Temp Calibration Data in Excel**

Copy and paste the data into an Excel column. It should look something like this:

|  |  |  |
| --- | --- | --- |
| 25.25,26.44,26.75 |  |  |
| 25.5,26.44,27 |  |  |
| 25.81,26.5,27.25 |  |  |

Select the above column, then go to “Data” and select “Text to Columns”



Make sure “Delimited” is selected. It should be by default. Click Next.

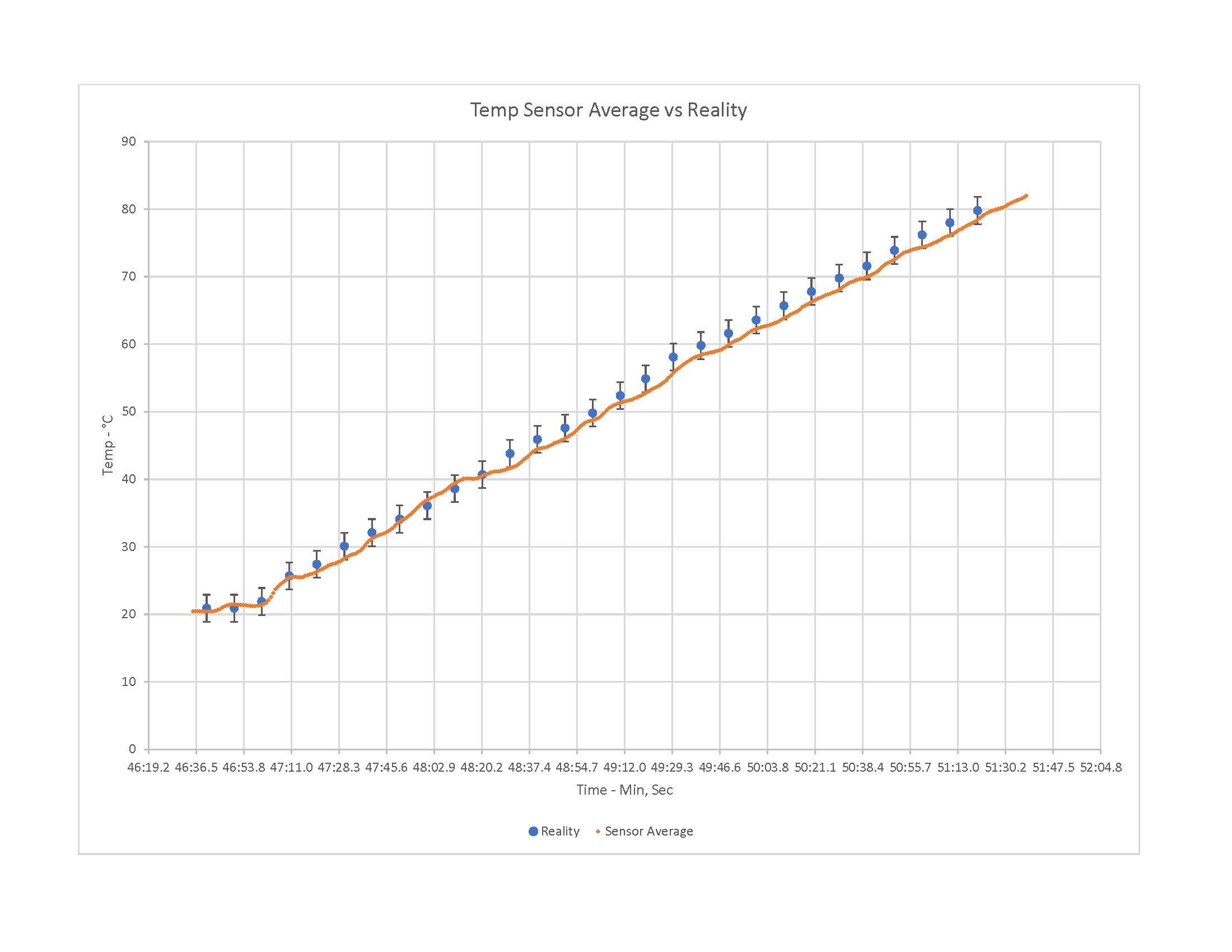
Deselect Tab and check Comma. The data below should look like this image.

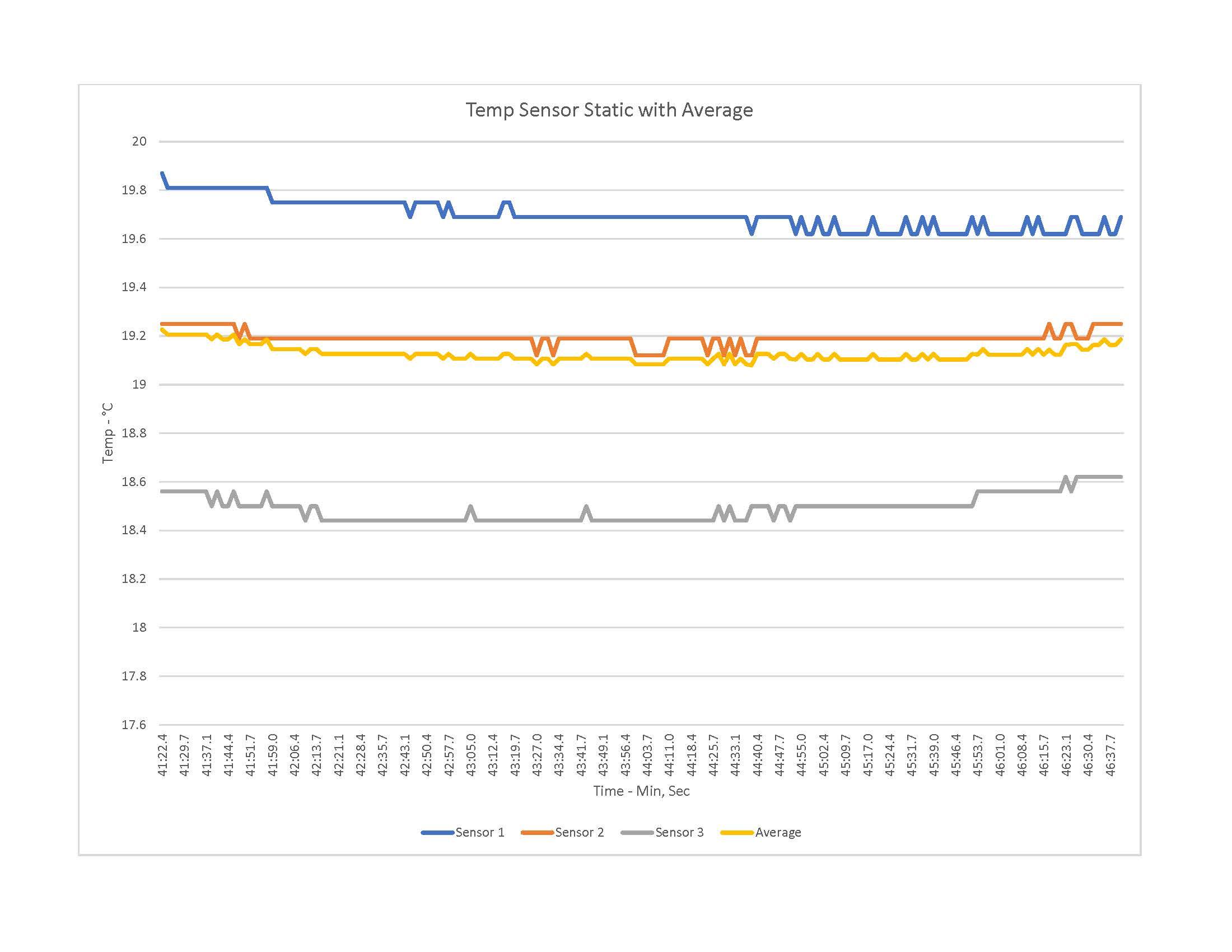
Click Next then Finish

The data should look like this table in Excel. Go about graphing!

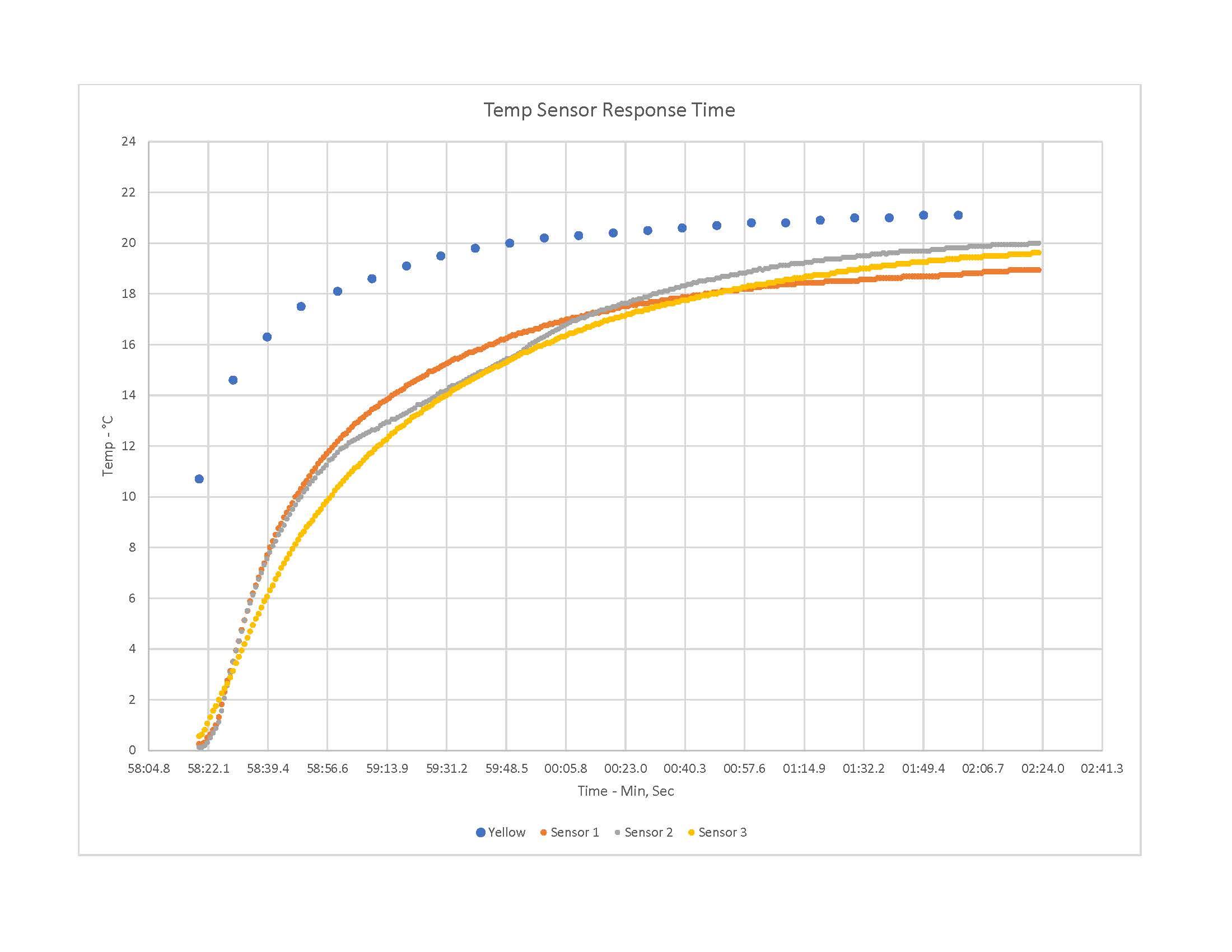
|  |  |  |
| --- | --- | --- |
| 25.25 | 26.44 | 26.75 |
| 25.5 | 26.44 | 27 |
| 25.81 | 26.5 | 27.25 |

**Final Temp Calibration Graphs**

Using a standard error of +- one you can see the average of the three sensors is within the acceptable margin of “reality” from the yellow temperature sensors. The time is in minutes and seconds. As you will see in the next graph, the individual sensors are not exactly on the average, but together they match reality well. 



The variance in a static temperature of the sensors over 5 minutes. There is about a one degree variance between the gray and blue, but on average they are accurate to the real temperature.



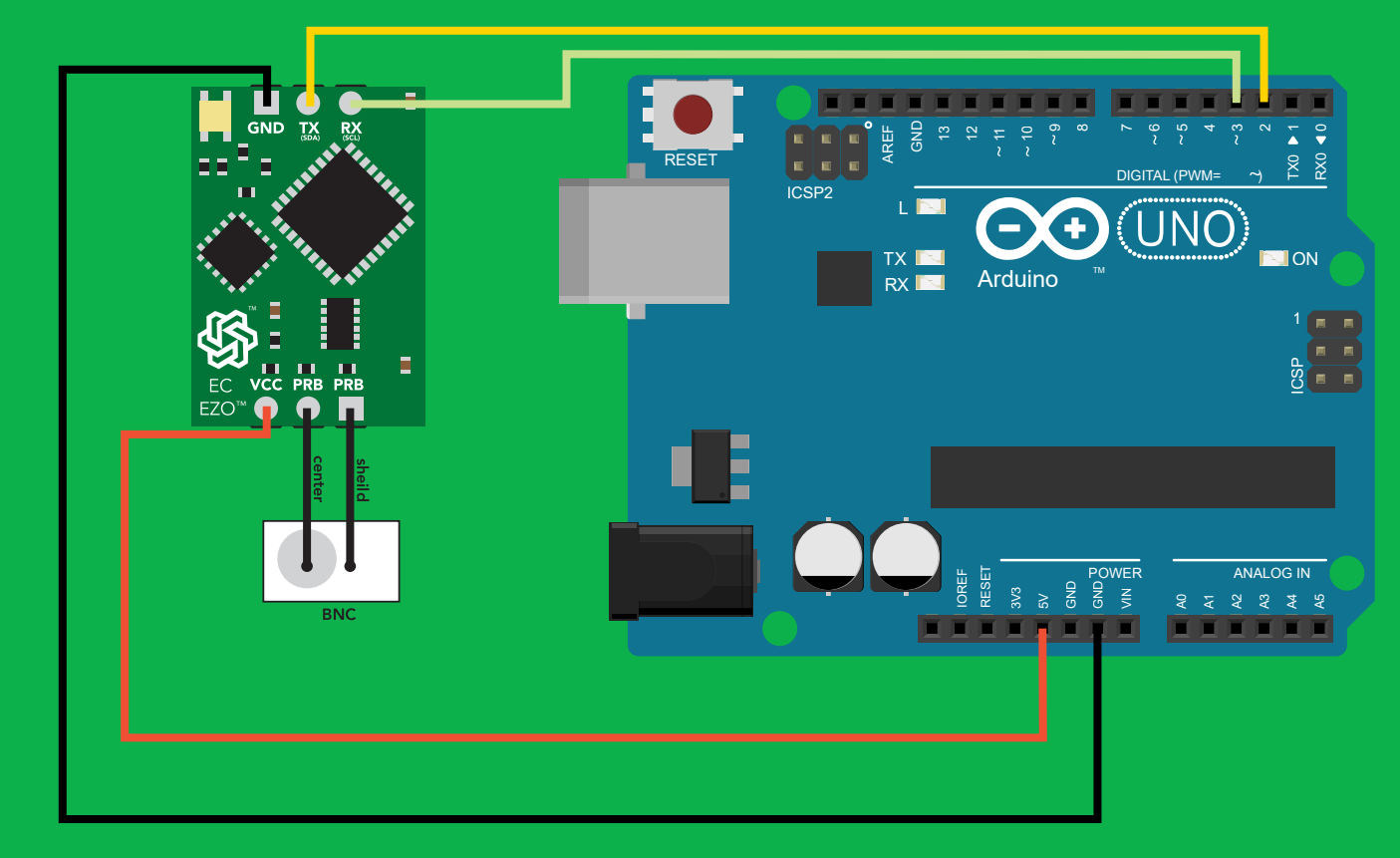
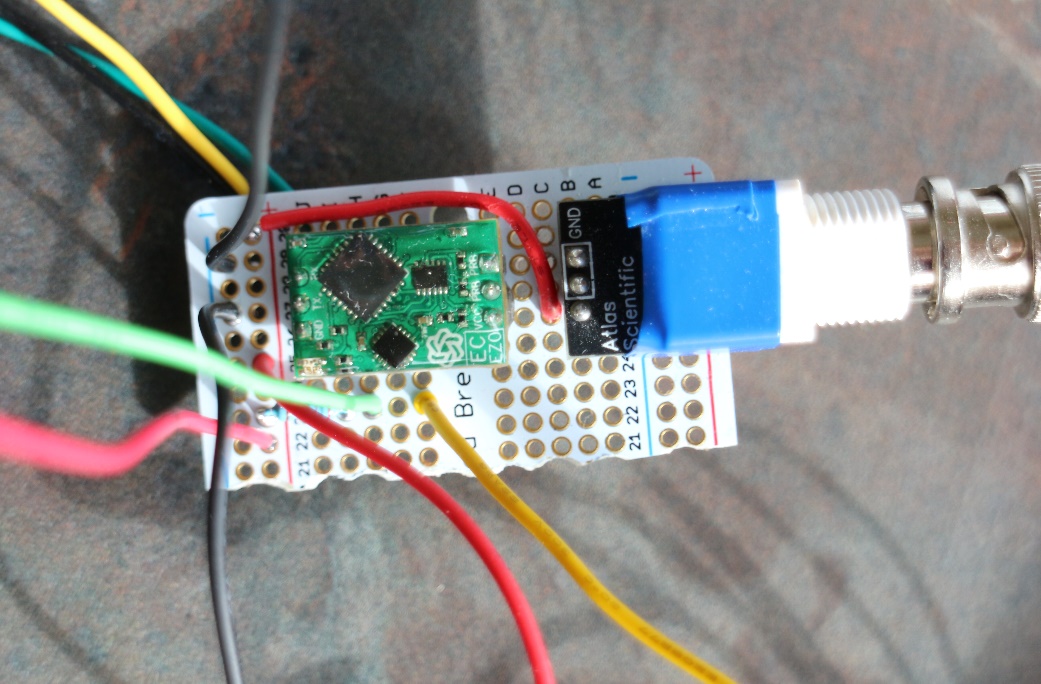
The response time of the sensor when taken out of ice water to come to room temperature. This isn’t as important to know as if they are accurate, as in the first graph, but it is a good data point to know if only to realize that you can’t move the CTD too fast through the water column if you want accurate data. They took about two minutes in air to come to come to ambient temperature. They’d be faster in water, as it conducts heat better, just don’t rush.

**Atlas Scientific EZO-EC Embedded Conductivity Circuit**

<https://www.atlas-scientific.com/product_pages/kits/ec_k0_1_kit.html>

**Explanation**

This sensor can tell you the TDS (Total Dissolved Solids), EC (Electrical Conductivity) and supposedly tell you the SAL (Salinity). However, in our case we only care about getting the EC data from the device as we will calculate the salinity based on that information ourselves.

**Wiring Salinity Sensors**

There is an important part of the wiring missing from the diagram, that is explained in the datasheet for the sensor. Both parts of the equipment need 5V power, not just the part in the green diagram. Connect it the same way as shown on right, but in a breadboard, so both pieces get 5V of power. The red bent wire close to the board is supplying power from the hot on the left to the row both of their power is in. It might not be clear right away which pin is the center and shield, so lay it out as in the photo. It’s also recommended, once you get to the soldering stage, to electrical tape around the sensor as shown (blue tape on right) to avoid any shorts.. Bench test in and once everything’s working, start calibrating!

**Calibrating Salinity**

Using the code “salinSensor\_test.ino”, follow the instructions found in the EZO-EC Conductivity Circuit Datasheet in order to calibrate the sensor using the correct liquids. Once you have calibrated your sensor *be sure to* label the sensor and components, so you always use the same sensor and board.

**MS5803-14BA Pressure Sensor Breakout Board**

<https://learn.sparkfun.com/tutorials/ms5803-14ba-pressure-sensor-hookup-guide>

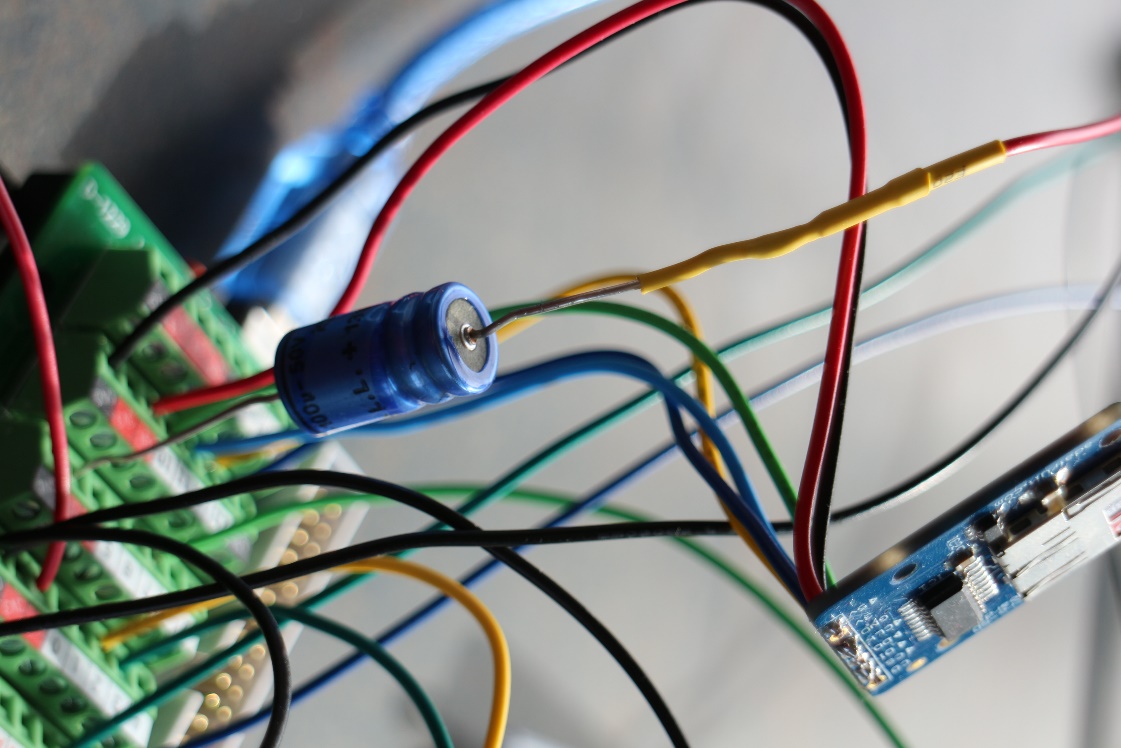
**Explanation**

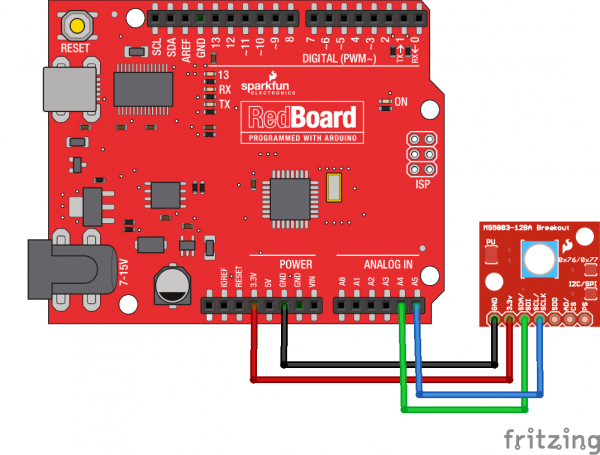
This sensor is used to tell depth, as pressure is a highly accurate way to tell the depth in a water column. Without this sensor the rest of the data would be worthless, as without depth to compare it to it is impossible to create the graphs needed to analyze the water column.

**Wiring Pressure Sensor**

This diagram is also missing a key component to make it function correctly. As stated in the manual (that’s why you look at the manual) a 100-microfarad capacitor must be placed in series with the 3.3V power supply to the board as seen below. Without this the sensor will display a semi accurate a reading; however, the reading will vary more. Use code “pressSensor\_test.ino”

The cylinder is a 100-microfarad capacitor in series (soldered) to the power wire connected to the power on the Uno

****



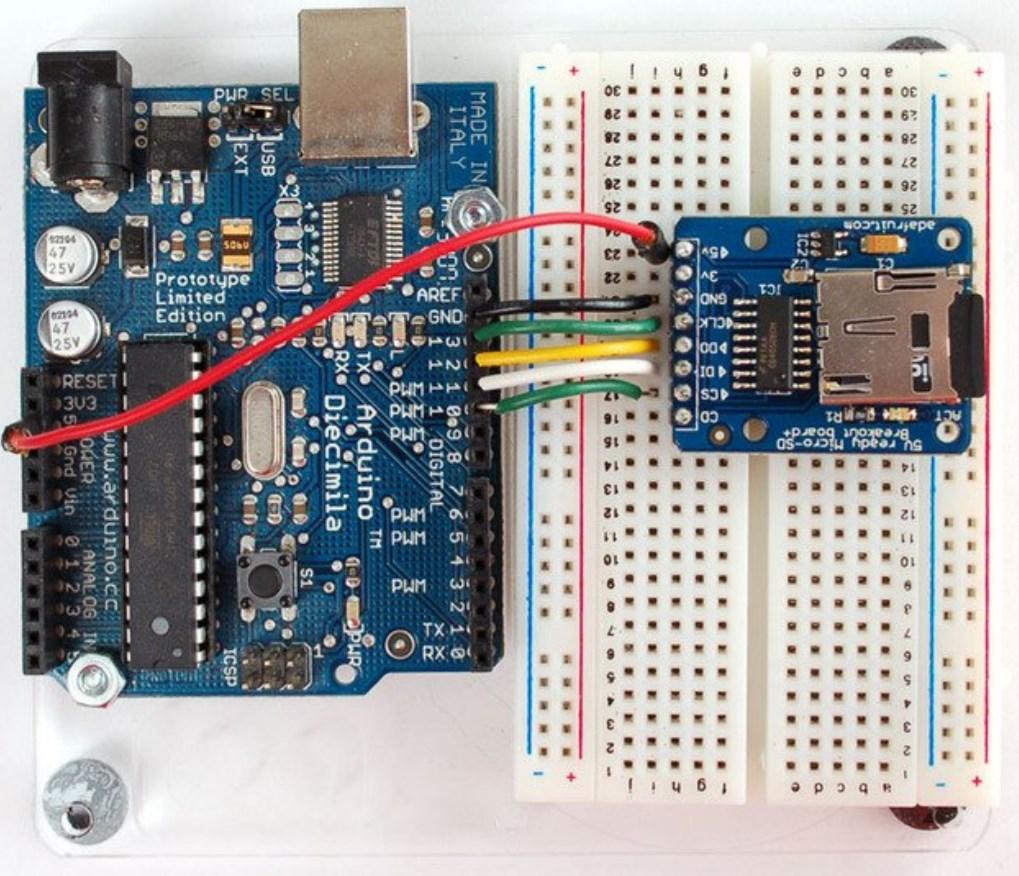
**Micro SD Card Breakout Board**

<https://www.adafruit.com/product/254>

**Explanation**

As the whole sensor package will be going underwater there must be a way for the Uno to save the sensor data locally – a microSD card. The wiring is relatively simple, but the coding is more difficult.

**Wiring SD Card**

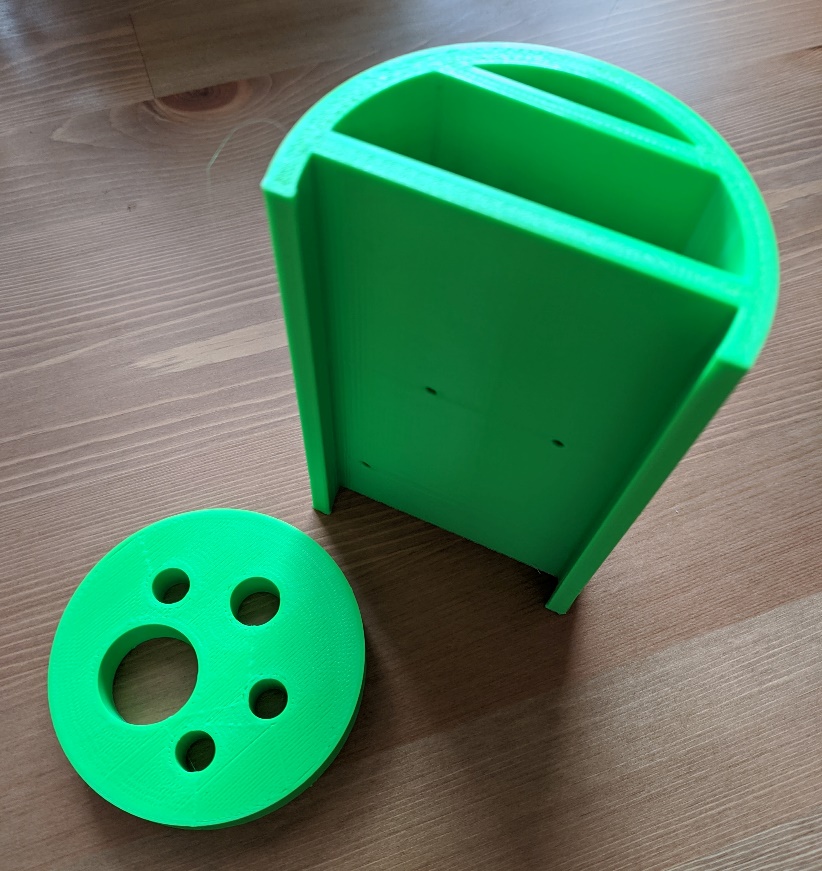
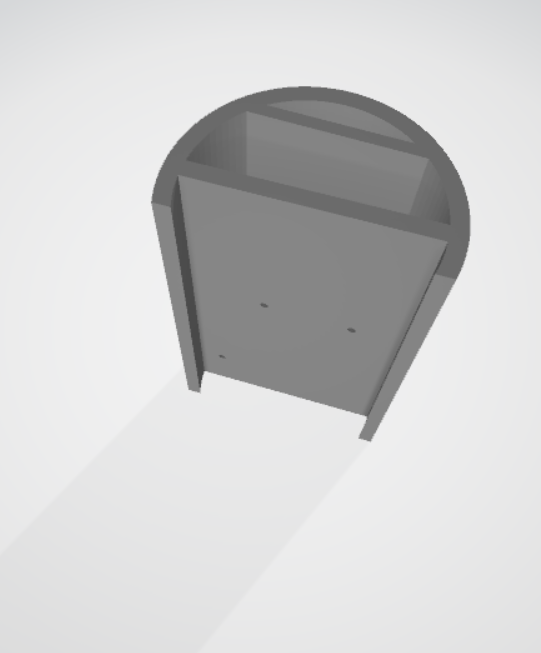


**Waterproof Housing**

While PVC is cheap and relatively easy to waterproof, 2” PVC is only rated for 280PSI. That means it’s only good to a depth of 91’. This is relatively shallow but might be good enough for your situation. Ideally, an aluminum housing would be ideal. One side with sensors, one side you can screw with a gasket.

**3D printed equipment tray**

It’s recommended, but not necessary, to 3D print a tray to slide in and out of the housing, to make working on the equipment easier. This also has the bonus of securing the equipment more securely, so it jostles less as you deploy it. See zip file for printer STL file.





Making the housing

Using 3” PVC pipe make an approximately 1.5’ section. On one end you shall have to secure the 3D printed sensor array or using a PVC end cap drill holes the correct size for the sensors. Either way it’s done, permanently secure and watertight the connection of the sensors through and it to the pipe. This end stays permanently attached. The other end is what you will open to slide the tray in and out in order to access the Arduino. A 3" PVC cleanout plug could be used as the screw able end cap.

**Arduino Programs**

See README in the zip file for explanation of what the files are.

**Takeaways and Final Thoughts**

Although I was not able to finish the project – i.e., put in a housing and put in the water – I was able to get all the equipment ready to put in a housing. I would not have been able to do this without the help of my best friend ‘Vizi’, as he is wholly responsible for the final program that creates the text files and the README file. Together we went through almost 200 revisions to the deployment code; quite simply, without him the code would not have happened.

There are a few takeaways I would mention to students attempting this in the future:

* Take detailed notes every step of the way. A week later when you return to the project you can know exactly where you left off. It also greatly helps in the coalition of documents such as these.
* Take the time to *understand* how the breadboard works and stay *consistent* with the colors of your wires. You do not want to wire everything together, have it not work, and have to start from scratch if you can’t find the one wire that’s in the wrong spot. Knowing the breadboard and being consistent makes finding the one awry wire much easier.
* Do not sit and wallow in a ‘stuck’ state. If you are stuck on something ask your teacher, fellow students, and do not give up! The answer is out there – it is not impossible. More than once I got frustrated with something not working so I would move on and try again the next day. Sleeping on it and coming back tomorrow can be helpful.