

Homebrew CTD

Building a CTD with simple to use and readily-available consumer parts

Sophie Scopazzi



Homebrew CTD

**Building a CTD with simple to use and
readily-available consumer parts**

by

Sophie Scopazzi

Instructor: A. Cifuentes-Lorenzen
Project Duration: Sept, 2020 - Dec, 2022

Cover: The CTD in the water, sampling
Style: EPFL Report Style, with modifications by Batuhan Faik Derinbay

Preface

This manual began within an Introduction to Oceanography Lab with Dr. Cifuentes-Lorenzen at California State University Maritime Academy during the Spring 2020 semester. The original iteration was the same in principle, but used different sensors¹. During the class, I got these sensors to work and the data to save to an SD card, but stopped short of a water-tight housing - everything worked as intended on the bench. However, the aforementioned sensor's compatibility for a watertight enclosure left me wanting something better.

Fast forward. I took the entire setup home during the COVID-19 stay-at-home orders and got busy with life, this project never truly escaping me. Seeing the project sit in its plastic tub on my shelf, I finally returned September 2022 to finalize the project. I decided that doing so required different sensors (detailed in this manual), so set about making it happen.

While it is of course possible to buy an off-the-shelf CTD, doing so does not afford the level of education equivalent with building own's own (nor is it as fun!). The learning and problem solving involved with coding, soldering, connectivity of sensors, building a housing, and more, are all invaluable educational experiences that are part and parcel to this project.

I make this manual to document what I have made in the hopes of creating easy to follow instructions anyone with basic tools can follow to create their own - for themselves, or in a classroom setting.

Sophie Scopazzi
Cal State Maritime, January 2024

¹Waterproof 1-Wire DS18B20 Digital temperature sensor, SparkFun Pressure Sensor Breakout - MS5803-14BA

Contents

Preface	i
1 What is a CTD?	1
2 Design Objectives	2
3 Equipment Used	3
3.1 Sensors	3
3.2 Sensor Accuracy	3
3.3 Datalogger	4
3.4 Power	4
3.5 Housing	5
3.6 Misc. Equipment	5
3.7 Overall Cost	5
4 How to: Breadboards, Soldering, Coding	6
5 Atlas Scientific EZO-EC Embedded Conductivity Circuit	8
5.1 Wiring	8
5.2 Calibration	9
6 Blue Robotics Bar30 High-Resolution 300m Depth/Pressure	10
6.1 Wiring	10
7 Adafruit MicroSD Card Breakout Board	11
7.1 Wiring	11
8 Power Supply	12
8.1 Battery Life Calculation	12
8.2 Anker PowerCore 10000 Portable Charger	12
8.3 Blue Robotics Pressure Switch	12
9 3D Printing: Trays, Stands and Holders	13
9.1 3D Printed Equipment Tray (4x)	13
9.2 3D Printed Equipment Tray Holder (2x)	14
9.3 3D Printed Housing Holder (2x)	14
9.4 3D Printed Flange Holder	15
9.5 Misc. Small 3D Printed Parts	15
10 Housing	16
10.1 Pressure Ratings of Parts	16
10.2 Building the Housing	17
10.3 Flange	17
10.4 Flange Without Machining Tools	18
11 Deploying the CTD	19
12 The Arduino Program	21
13 Conclusion	22
A Wiring Diagram	23
B Arduino Program	24
C MATLAB Program	27

1

What is a CTD?

"CTD" is an acronym that stands for **C**Conductivity, **T**emperature, and **D**epth due to what the instrument measurements. The salinity value is calculated from the conductivity of the water, as the salinity of water changes its conductivity. Using a CTD, it is possible to plot the salinity and temperature of a water column by the depth.

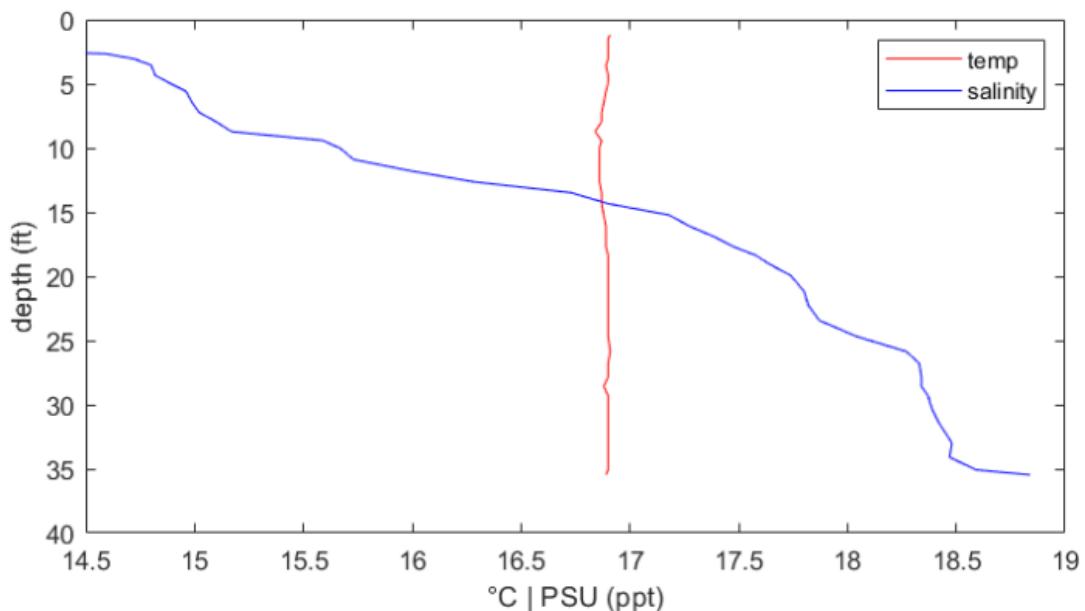


Figure 1.1: Example of what CTD data looks like using test data from the device built

This instrument, which supplies what may seem like only a few data points, can help answer a lot of questions about the water column. 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?" and, "What is the depth of the thermocline or halocline?" From this information it's possible to look further into the life living in the epipelagic zone (the top photic zone, where there is light in the water column).

Another important use for CTD data is calculating the speed of sound through the water column, as the temperature of water effects the speed of sound through the medium. This knowledge is vitally important for multi-beam sonar mapping of the seafloor.

2

Design Objectives

My goals were twofold: 1) A more cost-accessible sensor package still delivering on data fidelity requirements and 2) creating an accessible manual detailing how to build it, intended for use by individuals or teachers as part of a class, but also for anyone else who wants to make one.

1. Cheaper than pre-built existing CTDs currently on the market, ideally under \$700 USD.
2. Built with basic and accessible tools – soldering, 3D printers, basic shop tools (like those found in a college makerspace).
3. Profile to at least 150ft.
4. Profile continually for twenty-four hours and/or able to observe at least one complete tidal cycle.
5. Acceptable sensor accuracy (given overall cost constraints):
 - (a) **Pressure:** 2 inch resolution in water column (dZ)
 - (b) **Temperature:** $\pm 1^{\circ}\text{F}$
 - (c) **Salinity:** $\pm 2\text{-}3\%$

3

Equipment Used

3.1. Sensors

Part	Cost
1. Atlas Scientific Conductivity Probe K 10	\$183
2. EZO™ Conductivity Circuit	\$68
3. Blue Robotics WetLink Penetrator	\$12
4. Blue Robotics Bar30 High-Resolution 300m Depth/Pressure Sensor	\$85
Total	\$348

1. This can be bought in a kit or separate like this. Pick the one that has the best use case for you (range of values it reads, etc.).
2. Needed to connect to the Arduino.
3. Allows the sensor from the Conductivity Probe to go through the watertight hull.
4. Pressure and depth sensor in one. If your application requires greater than $\pm 1^\circ\text{C}$, get the Blue Robotics Celsius Fast-Response, $\pm 0.1^\circ\text{C}$ Temperature Sensor (I2C). I might incorporate this in the future.

3.2. Sensor Accuracy

The design objectives were selected with certain sensors in mind to keep the project cost-realistic.

- **Pressure:** 0.2mbar resolution, allowing for a depth resolution of 2mm, within our design objectives.
- **Temperature:** $\pm 1^\circ\text{C}$, outside our design objectives. The temperature sensor is integrated with the depth/pressure sensor, and as such is only accurate to $\pm 1^\circ\text{C}$. This is outside the original design objectives. If a greater temperature accuracy is required an alternative to the integrated temperature sensor is Blue Robotics' Celsius Fast-Response, $\pm 0.1^\circ\text{C}$ Temperature Sensor (I2C). If your budget allows, I'd recommend using this sensor for temperature, but if your project doesn't need anything more accurate than $\pm 1^\circ\text{C}$ don't worry about it. As it is, I accepted the trade-off of accuracy and cost.
- **Salinity:** With an accuracy of $\pm 2\%$ this sensor is within our design objectives. As stated before, you will need to decide which salinity range you wish to purchase for your use-case. It is important to note that the sensors documentation gives a 90% accurate response time in one second. This means the slower the CTD is lowered in the water column the more accurate this sensor will be (more on this later in Section 11).

3.3. Datalogger

Part	Cost
5. Arduino Uno R3	\$28
6. Screw Terminal Block Breakout Shield Module for Arduino UNO R3	\$21
7. Adafruit MicroSD card breakout board+	\$8
8. SD/MicroSD Memory Card (8 GB SDHC)	\$10
9. Blue Robotics I2C Level Converter	\$25
	Total \$92

5. Arduino Uno R3 is cheap, easily found, and has a lot of supporting documentation online. The Arduino IDE is free and relatively user friendly.
 6. Allows for ease of connecting wires. It is not strictly necessary but makes connecting power to everything especially easy.
 7. Allows the Arduino to save sensor data onto an SD card.
 8. Where the data is saved.
 9. Converts and connects the depth / pressure sensor to the Arduino.

3.4. Power

Part	Cost
10. Anker PowerCore 10000 Portable Charger	\$26
11. Blue Robotics Pressure Switch	\$20
12. Adafruit 1 Watt Cool White LED - Heatsink Mounted	\$4
Total	\$50

10. Using a normal USB power bank to power the entire package makes it easy to charge, and supplies plenty of power.
 11. The way to turn the CTD on and off that works under pressure. The only problem I've had is sometimes it is difficult to turn off (unscrew) after the CTD is at depth. It's also possible to make a magnetic switch.
 12. This is needed because the power draw is so low the USB battery bank goes into sleep mode and stops supplying power. I had this LED lying around from another project and threw it on, and it worked. The light is a good visual, especially in the water, that the CTD is on. If you have some kind of LED lying around, try it before buying this.

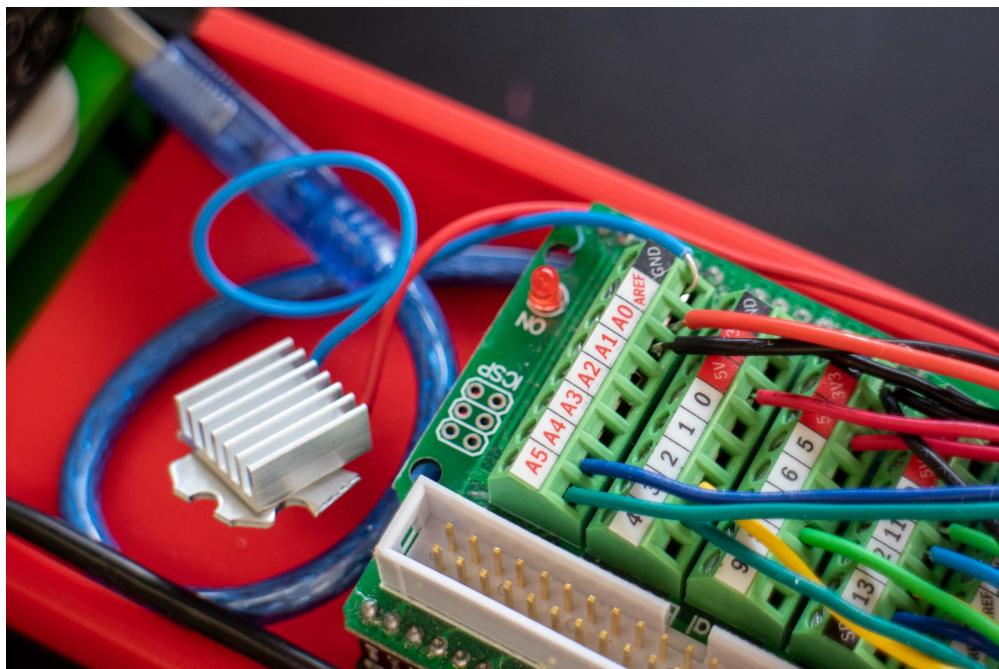


Figure 3.1: The LED is 5v and I put a small heat sink on it

3.5. Housing

Part (McMaster part #)	Cost
Thick-Wall PVC Plastic Pipe Fitting for Water (4881K218)	\$30
Water- and Steam-Resistant EPDM Gasket (1082N15)	\$5
Standard-Wall Clear Blue Rigid PVC Pipe for Water (49035K49)	\$61
Pipe Cement for Plastic Pipe (74605A14)	\$10
Steel Oval Eye Nut - for Lifting (3274T42)	\$12
Low-Pressure Steel Unthreaded Pipe Flange (68095K382)	\$42
Zinc-Plated Steel USS Washer (90108A032)	\$18
Standard-Wall PVC Pipe Fitting for Water (4880K57)	\$6
Medium-Strength Grade 5 Steel Hex Head Screw (91247A726)	\$13
Zinc Yellow-Chromate Plated Hex Head Screw (91257A720)	\$13
Medium-Strength Steel Hex Nut (95462A033)	\$30
250-Feet Hollow Braid Polypro Rope, Yellow (Amazon)	\$16
Total	\$256

This is designed for my application of at least 150ft. It has a theoretical max depth of 300ft based on the minimum specification of the parts (the clear PVC pipe). You could take this list as is or look through designing another housing – especially if there is a greater or lower max depth desired. If you design another, please reach out and let me know what you make! Lastly, the PVC doesn't have to be clear, but I felt seeing the insides was more important and didn't sacrifice depth.

3.6. Misc. Equipment

Part	Cost
Soldering iron and solder (any kind)	\$105
Heat Shrink Tubing Kit - 2:1 Shrink, 300W Mini Heat Gun	\$14
Hook-up Wire Spool Set - 22AWG Solid Core - 10 x 25ft	\$35
Wire strippers / cutters (any kind)	\$14
Standard Jumper Wires Plus Set Solderless Breadboard	\$11
Safety glasses or goggles	\$10
Total	\$189

This section is included for the benefit of anyone who does not have access to the above. Already having these parts from previous projects, or because you have access to a space that has these things, would lower the cost further.

3.7. Overall Cost

Entire overall cost: \$925

Cost of everything minus misc. equipment: \$736

This value may appear costly, but in comparison it is relatively cheap, especially if one already has access to soldering equipment and hardware for the housing (for example, in a Makerspace, machine shop, or lab space on a college campus).

**** I used a lathe to face (thin) the flange plate then a mill to drill the through-hulls and wells to hold the sensors. This is difficult to put a price on. If you do not have access to tools such as these, I recommend getting a bit more creative with the flange (see Section 10.3-10.4) ****

4

How to: Breadboards, Soldering, Coding

Breadboards

Breadboards are for temporary testing of wiring and are connected as shown in fig. 4.1.

The holes down the sides with red and blue colors go all the way down the board, to be used as power/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. Staying consistency with wire colors is incredibly helpful.

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 is not worth the potential for eye problems. Do not breath in the solder smoke as it is the flux on the solder burning. It will not kill you quickly, but is not good for you either.

General - Given the choice, be 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. Expose only about 5mm of wire. Before soldering the wires slip on the heat shrink. Put into the board and solder. It should look something like the photo shown (fig. 4.2). Clip the extra wire and inspect to be sure you did not inadvertently bridge any connections. If you did, unsolder using a vacuum gun or copper solder wick and redo.

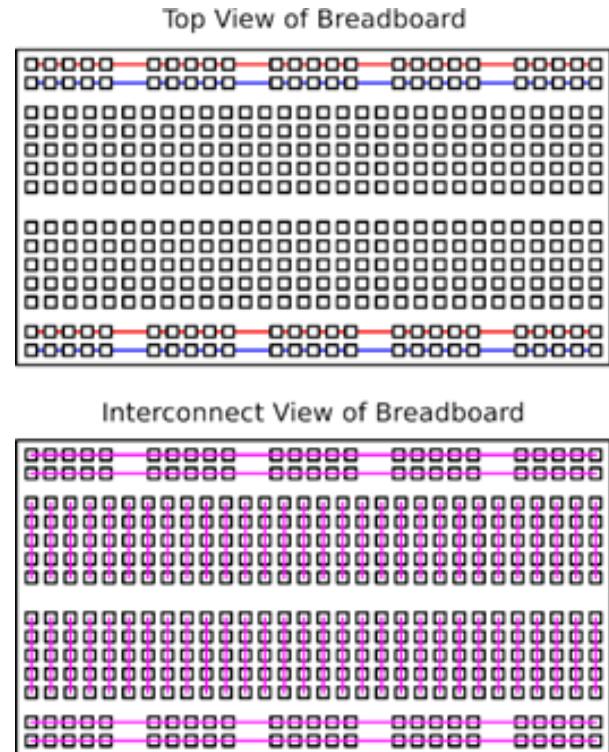


Figure 4.1: How a breadboard is connected

Software

Programming is accomplished in the *Arduino IDE software* on Windows, Mac, or in a web browser. The latter is not recommended. Use the downloaded software if at all possible. Using the version on the web can create all kinds of headaches.

Unable to connect to board - COM PORT error

1. Ensure the USB is plugged into Arduino and computer.
2. In the Arduino IDE program navigate to 'Tools' → 'Port' → select 'Arduino UNO'.
3. Try a different cable, unplug/replug, or restart the program/computer

Coding

Common errors when coding are below. Working through understanding code can be difficult. Thankfully, Arduino has many wonderful examples and *language reference guides*.

1. Not having a ";" at the end of the line.
2. The baud rate stipulated in the code is different than the baud rate selected in the serial monitor.
3. Missing a capital letter, or having one where there must not be.

Helpful commands

```
// 'Comment' out the line with two "//" to add words without having them run as code
define rx 2; //define what pin receive is going to be, 'tx' would be transmit
Serial.begin(9600); //sets baud rate for the hardware serial (select this in the serial monitor)
```

//leave concise notes in your code ensuring you can go back and know what it is doing

The Main Arduino Program and Other Code

Start with the supplied code in the appendix or within the zip file for individual sensor code and the final deployment code *available on my website*. Each sensor has its own individual code that can be used for testing on a per-sensor basis supplied by the manufacturer, but I've included them in the zip file for ease.

5

Atlas Scientific EZO-EC Embedded Conductivity Circuit

This sensor gets TDS (Total Dissolved Solids), EC (Electrical Conductivity) and the Salinity (in Practical Salinity Units, ppt).

5.1. Wiring

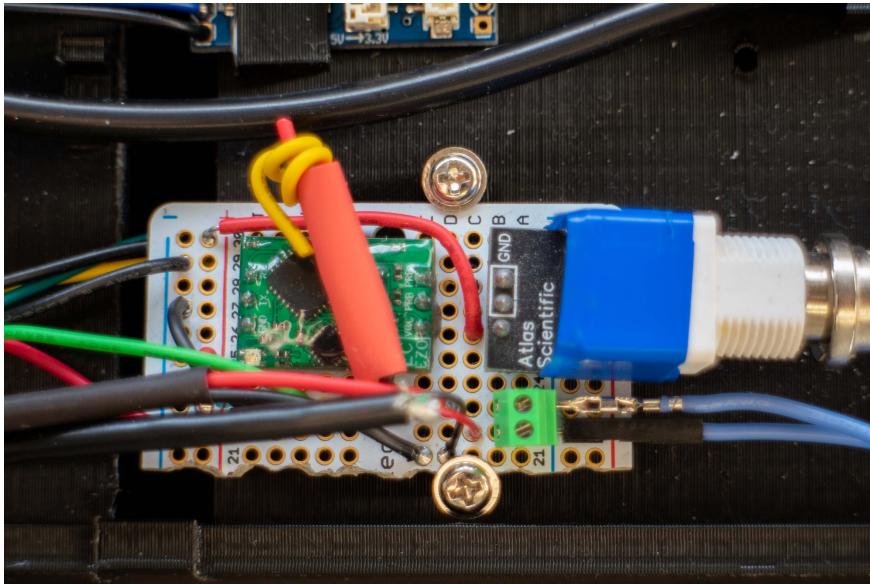


Figure 5.1: Salinity sensor power wiring. Red wire between screws.

There is an important part of the wiring missing from the diagram found in the manual, that is explained in the datasheet for the sensor. **Both parts of the equipment need 5V power**, not only the part in the green diagram. I recommend connecting it the same way as shown in figure 5.1, so both pieces get 5V 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 pins is in.

It might not be clear right away which pin is the center and shield, so I recommend laying them out in a breadboard as in the photo, and testing, *before soldering*. I also recommended, once you get to the soldering stage, to electrical tape around the sensor's BNC connector as shown in fig. 5.1 (the blue tape) to avoid any shorts or potential issues from the bottom of the board touching the perma-proto breadboard.

To get the salinity sensor's wire through the thru-hull, you will need to cut its wire. This may feel scary, but take your time, be careful and it will be okay. I recommend cutting the sensors cable about 3in from the sensor itself. Depending how confident you are with your cutting and soldering skills, you could give yourself less to work with, but I wasn't willing to take that chance with the most expensive sensor, which is why you see my cable so long in the photos. Once you have put the wire through the Blue Robotics thru-hull hardware, solder the wires with their matching color, then cover in heat shrink (don't forget to put the heat shrink out of the way on the wires before soldering).

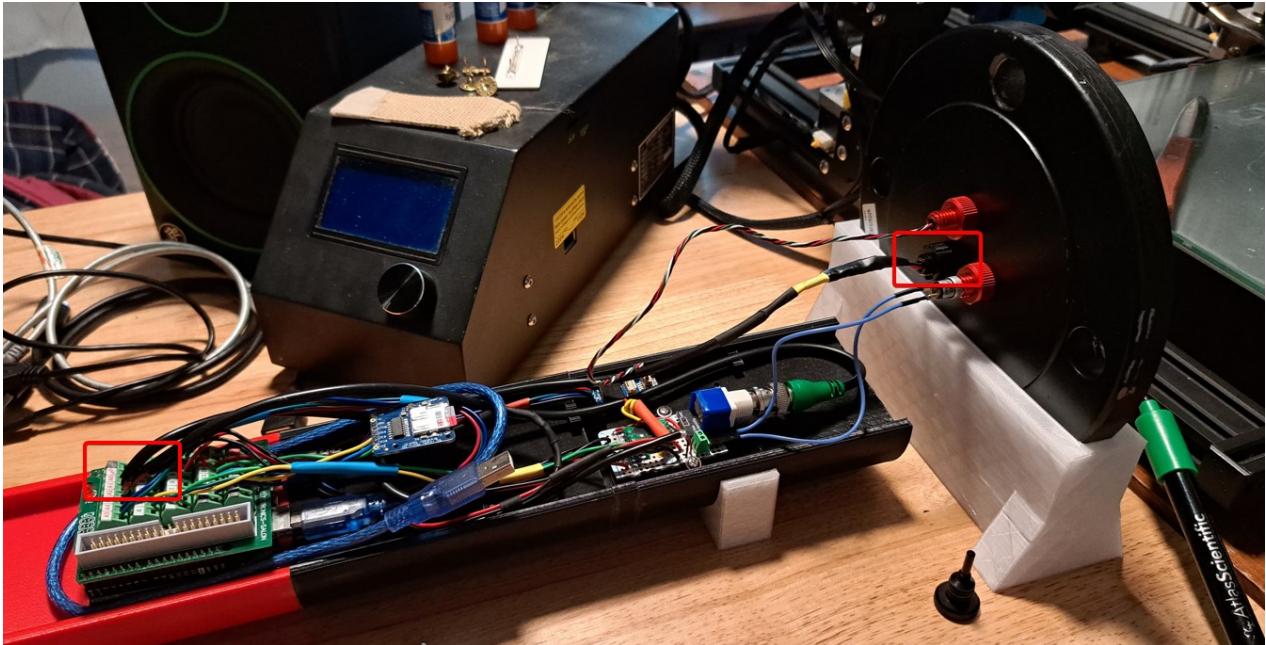


Figure 5.2: Salinity sensor wiring, parts in red boxes.

To ensure it is possible to fully remove the flange and easily access the electronics, the cable must be removable. This can be accomplished by the BNC connection, but I chose not to do so. The soldered connection holding the BNC connector to the perma-proto breadboard is not the strongest of connections and I am worried about breaking the soldering so I opted for a different solution. Using empty analog ports on the Arduino shield (A1 and A0) since they are unused terminals). I screw in the matching wires from the BNC connector and the sensor from the flange, making a temporary connection using the screw terminals. This affords the ability of easy removal without buying different parts (this is similar in idea to the power switch, as detailed later).

5.2. Calibration

Using the code `salinSensor_test.ino`, follow the instructions found in the EZO-EC Conductivity Circuit Datasheet to calibrate the sensor using the correct liquids. Once you have calibrated your sensor be sure to label the sensor and components, (if you have multiples) so you always use the same sensor and board.

6

Blue Robotics Bar30 High-Resolution 300m Depth/Pressure

This sensor gets depth (in mbar and meters, which I convert to feet), and temperature (in °C, which I convert to °F).

6.1. Wiring

Please follow the extremely well-made tutorial available on Blue Robotics' website. They've done a stellar job and there is no use repeating it here.

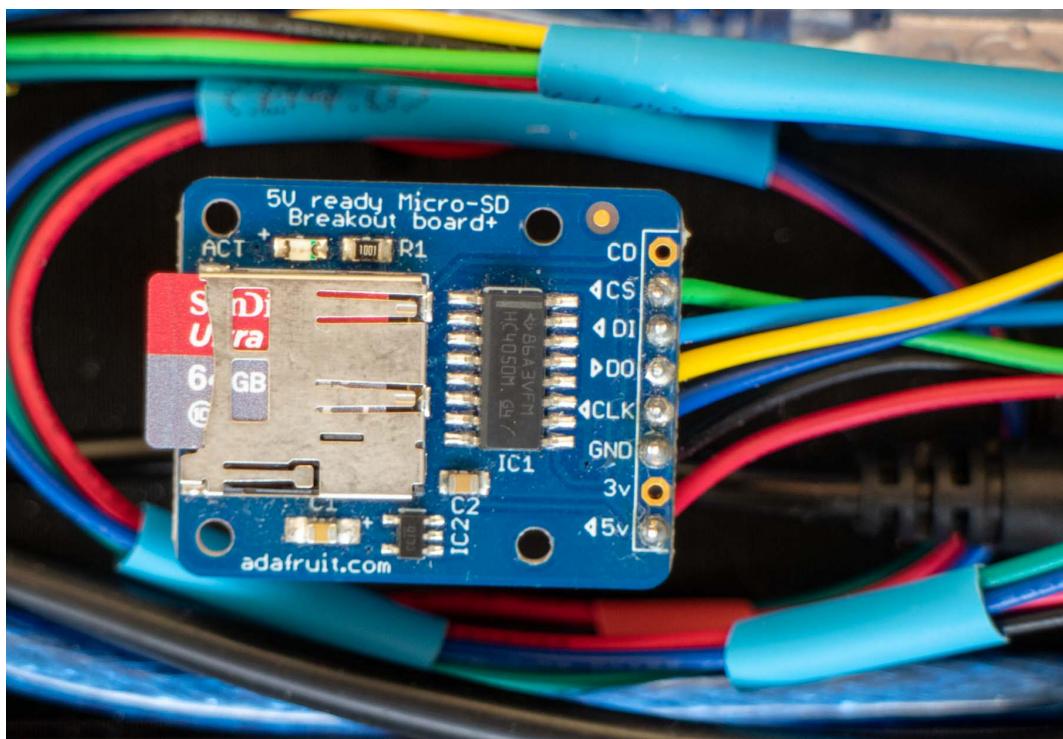
7

Adafruit MicroSD Card Breakout Board

Cheap and easy to physically connect, this allows the Arduino to save the sensor data to a micro SD card.

7.1. Wiring

Like the Blue Robotics' sensor, Adafruit has a well made tutorial found on their website. And to reiterate, make your life easier and, given the choice, do not use the same color wires for everything.



8

Power Supply

8.1. Battery Life Calculation

As designed the entire CTD package uses approximately 340mAh (0.34Ah) of power. This means, using the following formula (which assumes a perfect scenario, which is never the case), the CTD can be deployed for approximately [THIS MANY] hours (rounded down).

- **Formula:** $Time(H) = Capacity(Ah) / Current(Amps)$.
- **Plain English:** Time in hours = battery capacity in Ah divided by the current draw in amps.
- **Calculation:**

$$Time(H) = 10Ah / 0.31A$$

$$Time(H) = 29.42$$

The power draw without the 1W LED was actually so low the power supply would auto power off after thirty seconds, not sensing enough power draw to stay "awake". It uses more power, but there is still enough power to last at least twenty-four hours.

8.2. Anker PowerCore 10000 Portable Charger

Using an off-the-shelf USB power supply was selected because 1) they are cheap and readily available and 2) are easy to charge without specialized battery chargers, making them easier for folk not experienced with more specialized batteries (such as those found in hobby drones). I used a Velcro command strip to attach the battery to the tray, for easy removal if required.

This specific battery was selected for the following reasons:

1. With a capacity of 10,000mAh (10Ah), it has plenty of power to outlast the design objective.
2. This specific model fit within the PVC pipe, and even the tray, making it easy to fit.

8.3. Blue Robotics Pressure Switch

This switch allows turning the CTD on and off in the field. This not only saves battery, but also simplifies the data back-end by not having stretches of meaningless data between deployments.

The switch is a bit overkill for this situation, able to handle up to 5A of current and 120V AC or 26V DC, but it is pressure rated and quite up to the task, while remaining relatively inexpensive.

Much like the salinity sensor, to ensure the possibility of fully removing the flange, I had to get a bit creative as to how to integrate the switch into the power circuit. I settled on a small screw terminal block where I could attach both ends from the switch (see section [SOMETHING] for the wiring diagram).

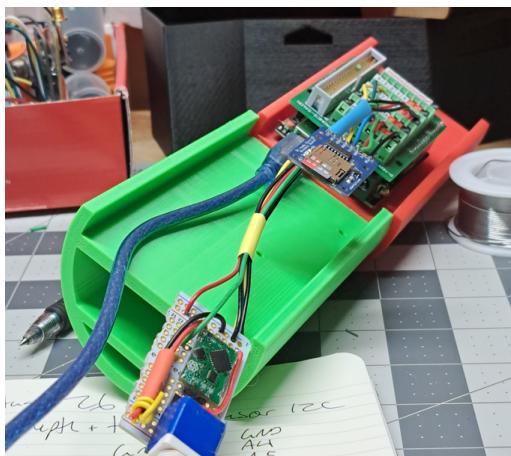
9

3D Printing: Trays, Stands and Holders

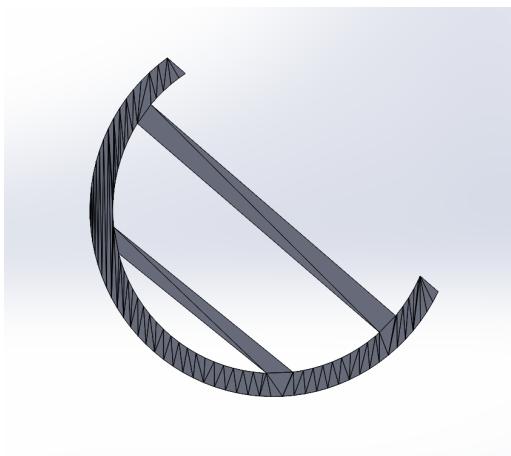
All files explained below can be found in the zip file downloadable from *my website* (<https://www.sophiescopazzi.com/projects>)

9.1. 3D Printed Equipment Tray (4x)

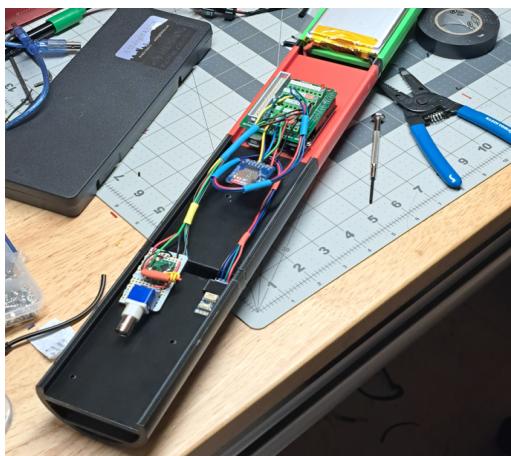
To make deployment and working on the equipment easier I recommended printing an equipment tray to secure the equipment to.



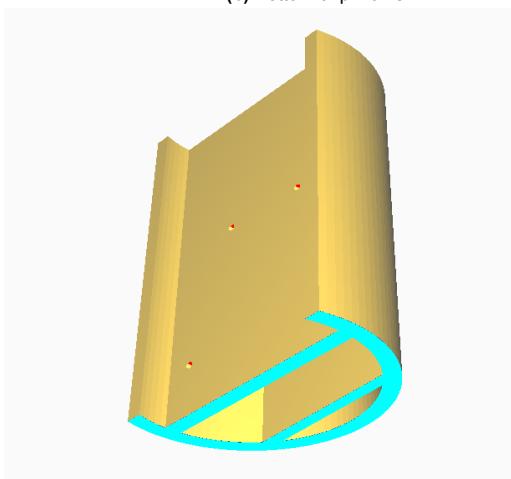
(a) First two I printed



(b) Bottom of print view



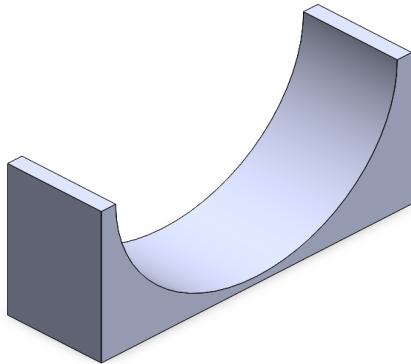
(c) Added two more (black ones)



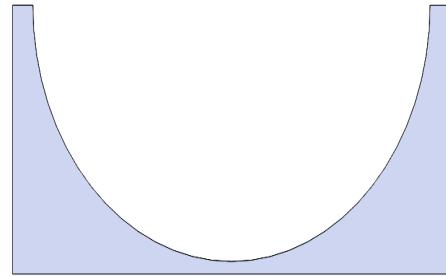
(d) From Ultimaker Cura

9.2. 3D Printed Equipment Tray Holder (2x)

For holding the equipment tray when it is not in the housing. It is much easier working on the tray when it's not rolling around (I did not use this for a bit and got fed up until I made this).



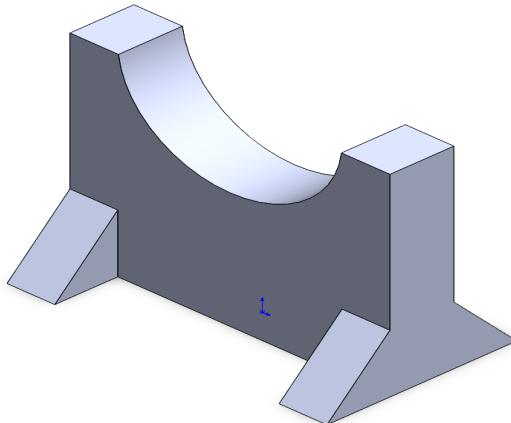
(a) Isometric



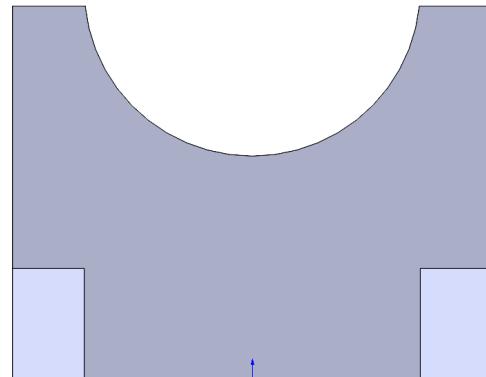
(b) Side

9.3. 3D Printed Housing Holder (2x)

Used to hold the housing while it is on the worktable. I highly recommend printing it! Same as before, I made it out of workability necessity (and it helps the CTD look good in the classroom / on display).



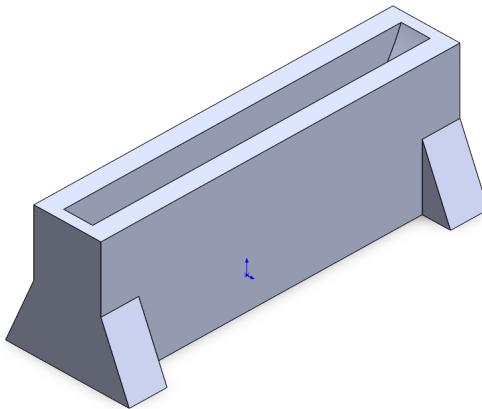
(a) Isometric



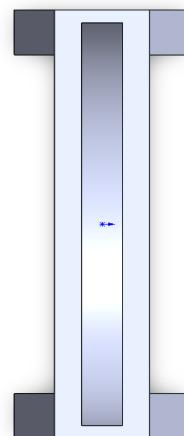
(b) Side

9.4. 3D Printed Flange Holder

I highly recommend printing this, or something like this to hold the flange and sensors safely on the table while you work on the CTD. Print this file about 104%.



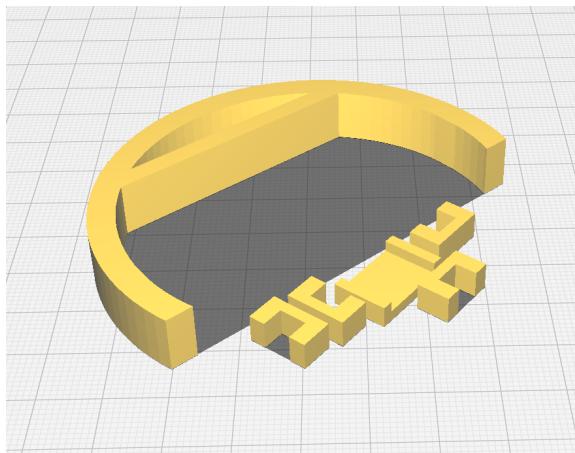
(a) Isometric



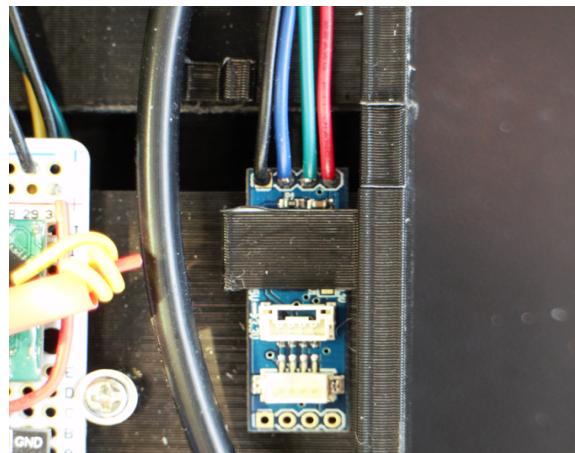
(b) Top

9.5. Misc. Small 3D Printed Parts

I 3D printed small parts to hold cables and parts in place. Depending how you choose to organize the internals, you may or may not need these, or will need to design your own.



(a) From Ultimaker Cura



(b) Holds the I2C converter down and cable (cable out of connector in photo)

10

Housing

10.1. Pressure Ratings of Parts

The minimum pressure rating for any one part is the clear PVC at 130psi, allowing for a theoretical max depth of 300ft (91.5m) (Blue Robotics' Water Pressure and Depth calculator). If greater depth is desired, or you simply do not feel the need to use a clear pipe and see the inside (clear is also more expensive), I'd recommend buying Standard-Wall Unthreaded Rigid PVC Pipe for Water (3" Pipe Size, 5 Feet Long). It is rated for 260psi, matching the PVC pipe fitting, leaving the limiting part the salinity sensor at 200psi, allowing for a theoretical deployment depth of 472ft.

I decided to use the Clear Blue Rigid PVC Pipe because 1) it met the 150ft profiling criteria and 2) it affords the ability to see the internals, creating peace of mind during deployments. While the clear PVC has a lower pressure rating compared to its opaque cousin, I felt the trade-off was acceptable.

1. **Standard-Wall Clear Blue Rigid PVC Pipe for Water - 300ft, 91m**
130 psi @ 72° F. Schedule 40.
2. Thick-Wall PVC Plastic Pipe Fitting for Water - 659ft, 200m
285 psi at 72° F for steel. Schedule 80. Pressure class 150.
3. Water and Steam Resistant EPDM Gasket - 1,850ft, 564m
800psi.
4. Low-Pressure Steel Unthreaded Pipe Flange - 659ft, 200m
285 psi @ 72° F.
5. Standard-Wall PVC Pipe Fitting for Water - 601ft, 183m
260 psi. Schedule 40.
6. Conductivity Probe K 10 - 462ft, 141m
200psi.
7. Blue Robotics Pressure Switch - 3,116ft, 950m
200psi.
8. Blue Robotics WetLink Penetrator - 3,116ft, 950m
200psi.

10.2. Building the Housing

Below are the steps to make the housing exactly like I did. If you have better ideas for how to build the housing, or want to get different parts, absolutely experiment!

1. Ensure the connection areas of the cap, tube, and plastic flange are free from oil, particulate, or other matter. Once you have cleaned the connection areas, don't touch them with your hands (will leave oil from fingers).
2. Put on disposable gloves (do not get the PVC pipe cement on your skin). Fully cover the inside of the cap with the glue, then seat it fully onto the tube.
3. Have a paper towel or other disposable rag handy. Cover the inside of the plastic flange and seat it onto the plastic tube. If there is glue dripping into the flange area (there probably will be) wipe with rag before it dries. This is to ensure the circular area of the flange that protrudes from the main body presses into the rubber gasket used to seal the housing.

With that, almost all the housing is made. The last (and what I would call the most complex) part of building the housing is the part bolted to the housing to seal the watertight container - the metal flange.

10.3. Flange

I decided to go with a steel pipe flange, as 1) I had the resources available to machine it to desired specifications and 2) I figured it would be heavy enough to help the CTD sink without having to add additional ballast.

Using the Machine Shop at CSU Maritime, and the invaluable expertise and assistance of Mr. Long, we first used a lathe to face the flange, making it thinner - original 24mm, new 15mm. This step had two purposes: 1) make the part lighter, but more importantly 2) thinner, so the Blue Robotics through-hull bolts would properly fit. We quickly realized thinning the flange to 5mm was impractical on the lathe, due to the amount of time it would take to remove that amount of material.



(a) Lathe



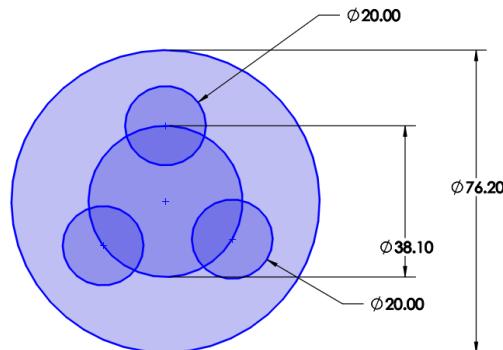
(b) Mill

Figure 10.1: Machining the flange

Instead, we moved to the mill with the idea of creating wells for the thru-hulls. In this way, we removed far less material for the same desired function. I began by using SolidWorks to model the layout for the holes and wells, ensuring everything would remain within the 3" of the PVC pipe (fig. 10.2(b)). There are many ways to accomplish a diagram such as this (hand drawing, SketchUp or any other CAD program, and I'm sure more I'm not mentioning. I was/am in the process of getting better at using SolidWorks so I used it).



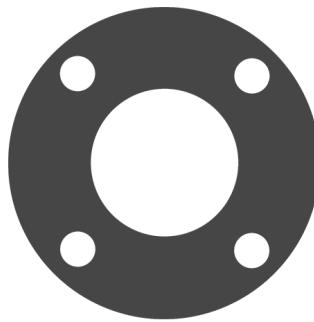
(a) Final flange



(b) Solidworks diagram

Figure 10.2: Closeup of flange holes, SolidWorks file

The holes in the flange, centered in the well, need to be made to a tight tolerance so the Blue Robotics' through-hulls will fit. They also cannot be too wide to afford a proper seat for the silicone gasket.

**Figure 10.3:** Gasket

After the first test of the CTD, to prove it did not leak, I cleaned the flange with fresh water, dried, scrubbed any rust-appearing parts, then coated it with three coats of spray paint to protect the flange from salt water and rust. I chose pink because it stands out in the water (and I like the color and had it from another project). Given the choice, I would recommend picking a bright color like yellow or pink.

10.4. Flange Without Machining Tools

If you do not have access to machining tools, I have a few suggestions, but also encourage creativity. If using plastic or thin metal, you will most likely need to ballast the CTD so it actually sinks. I was unsure exactly how it would float or sink, since I chose to not calculate prior to building, deciding to figure that out if it became an issue. As it is built, the CTD sinks flange first. I do not mind this, as it affords the sensors direct contact with the water column during its downward travel. Two ideas:

1. A steel plate of sufficient thickness for the depth rating desired, cut to size with an angle grinder, then drilled with a drill press or even by a hand drill, could work as a flat flange. This would change the bolts.
2. Laser cut, or otherwise cut in some manner, plastic of sufficient thickness for the desired depth rating. I'd be wary of the plastic strength to hold up to the stress around the bolts.
3. Pot the sensors in a PVC end cap using epoxy. This changes the entire design, but is an option.

11

Deploying the CTD

Putting the Housing Together

Using the McMaster bolts and associated parts listed earlier, close the flange securely as in figure 11.1. Using a criss-cross star pattern, as when securing a car tire, secure the nuts on the bolts as far as they will go. Once the flange is fully seated, tie a bowline (or other reputable knot, but I recommend a bowline) through both eye nuts.



(a) Overall layout



(b) Closeup

Figure 11.1: Example of knot and tying arrangement

Deploying in the Water Column

It is important to ensure the CTD falls sensor first in the water and is lowered *slowly*. The slower the better as all sensors will be more accurate as they adjust. This, so to speak, gives a higher 'resolution' (see fig. 11.2 and 11.3, these were taken *near* one another, but not in the *exact* location, which is why they appear similar).

Additionally, tie the bitter end of the rope to something you do not lose everything.

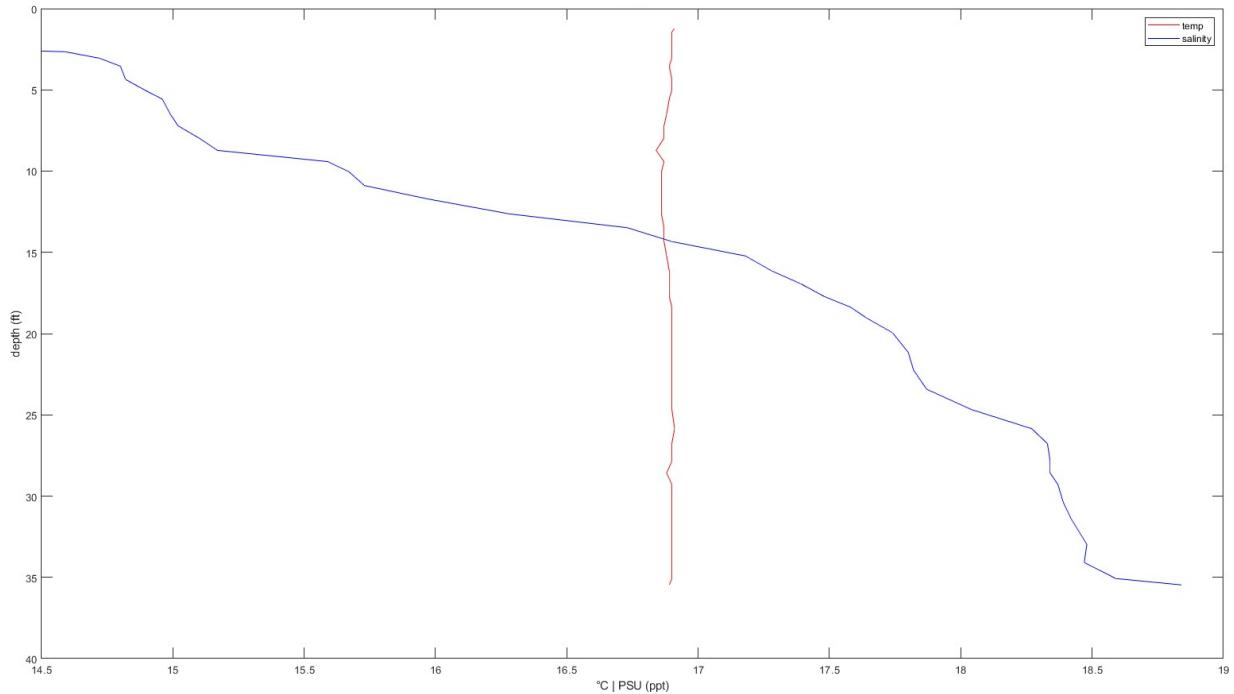


Figure 11.2: Fast drop, smoother

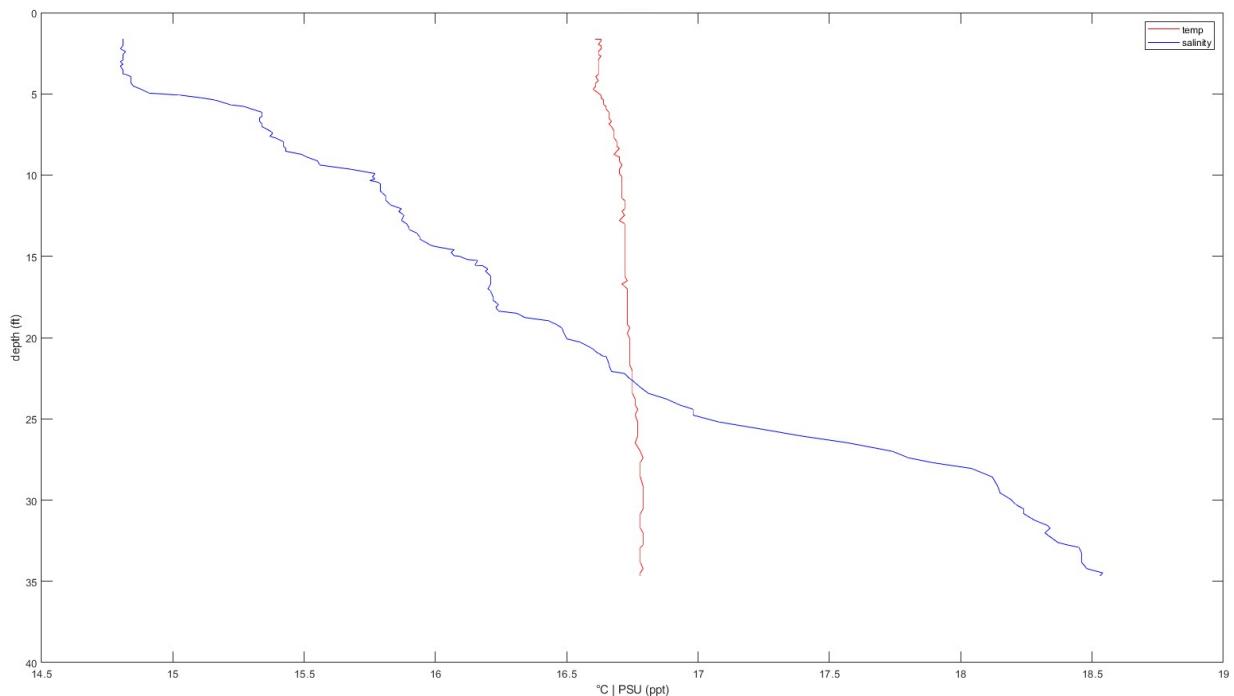


Figure 11.3: Slow drop, more granular

12

The Arduino Program

Where it (mostly) came from

The code is a combination of Blue Robotics' sample code and Atlas Scientific's sample code for their respective sensors. The SD part of the program is based off the Arduino SD Datalogger example program found in the Arduino IDE, but changed by me for my purposes.

If you find the coding aspect of this type of project intimidating, try and focus on one aspect at a time. Using a breadboard, wire and connect the SD card and make sure that works. Then add one sensor, then the other. Use the example programs provided by the manufacturer or elsewhere on the internet. Generally speaking, someone else has probably tried or done what you're doing so look around and see what help is already out there.

13

Conclusion

Takeaways and Final Thoughts

- Take detailed notes every step of the way. Comment your code. Keep a lab notebook. A week later (or sometimes a year...) 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 or when you share your work.
- 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, then start from scratch if you cannot find the one wire that is in the wrong spot. Knowing the breadboard and being consistent makes finding the one awry wire or connection much easier, especially if you take a week or two off (say over a holiday).
- Do not sit and wallow in a stuck state. If you are stuck on something, ask your teacher, fellow students, the internet, and do not give up! Maybe ask me too. The answer is out there – it is not impossible. More than once I got frustrated with something not working. I would ask for help, move on to another part of the project, or try again the next day. Sleeping on it and coming back tomorrow is helpful (maybe with a coffee or tea!), or simply take a walk around the building.

What I would do differently next time

Building the Housing

- Not face the flange on the lathe, instead only making the wells and thru-hulls. While this would increase the overall weight (ultimately not a big deal), it would decrease the steps and tools involved while not substantially changing the final product.
- I made the wells 9.5mm deep, which is 4.5mm or so too deep. As they are now, getting a wrench on the water-side of the flange to hold the Blue Robotics' thru-hulls in place to properly seat the nut on the dry side is a slight bit challenging. While you can get the wrench on it, it would simply be easier if I did not make the well as deep.

3D Printed Parts

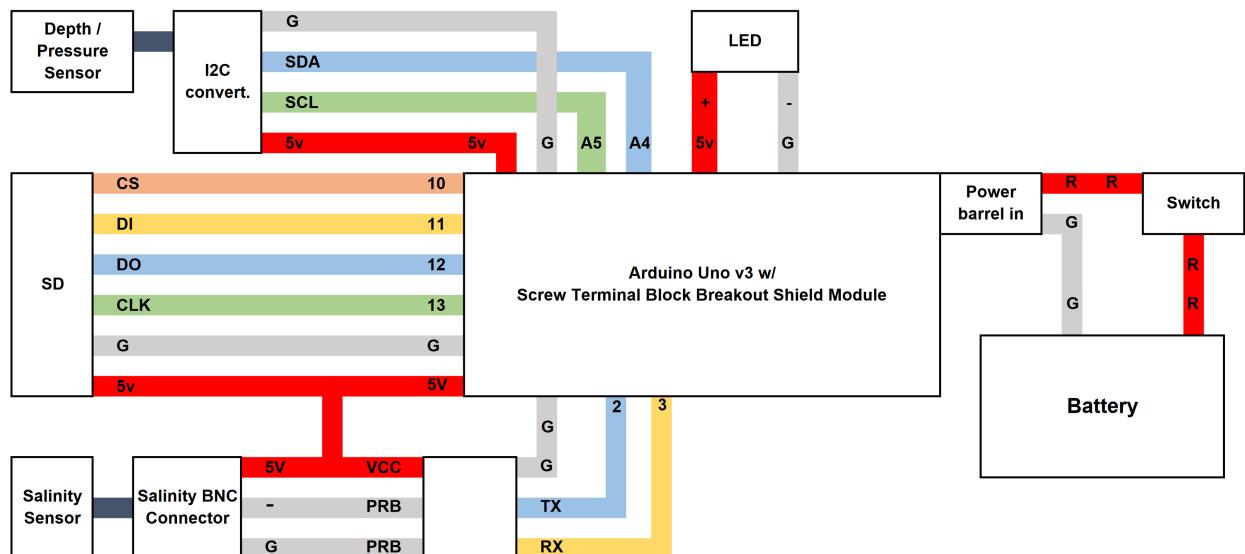
- Redesign the equipment tray to allow for greater clearance over the Arduino's shield. As designed, the wiring barely clears the PVC as it slides in.
- Redesign the flange holder's slot to properly fit the flange. When measuring I did not account for the slight protrusion on the housing side of the flange, meaning the holder is a tight fit. My print only works because of the low infill I printed it with, allowing it to flex to fit the flange. Thus, I do not recommend you make the flange exactly like I did anyhow (see section 10.3)

Wiring and Layout

- Solder pins to the I2C converter board instead of wires. This way I could put everything on a perma-proto breadboard.
- Totally redo the board with power and salinity to combined everything on one board. This would make the wiring far less messy.

A

Wiring Diagram



I do not recommend following my physical layout exactly 1:1 because there are far better ways to organize the internal sensor tray (i.e., use the wiring diagram to design your own physical wiring). The layout I ended up with formed from the organic state of starting with certain sensors, changing them partway through, and making it up as I went along while I worked through getting the CTD to operate as intended.

B

Arduino Program

```
1 // Order of CSV
2 // EC , TDS , SAL , mbar , C , meters
3
4 //This code was written in the Arduino 1.8.57.0
5
6 // TDS - Total Dissolved Solids
7 // 12,940 for 12880 23.0C      12,396 is actual thing to input
8 // 80,000 for 61601 at 22.4C    76,048 as actual thing to input
9
10 //Blue Robotics
11 #include <Wire.h>
12 #include "MS5837.h"
13 MS5837 sensor;
14
15 //EC
16 #include <SoftwareSerial.h>           //we have to include the SoftwareSerial
17     library, or else we can't use it
18 #define rx 2                          //define what pin rx is going to be
19 #define tx 3                          //define what pin tx is going to be
20 SoftwareSerial myserial(rx, tx);       //define how the soft serial port is going to
21     work
22
23 String inputstring = "";               //a string to hold incoming data from the PC
24 String sensorstring = "";             //a string to hold the data from the Atlas
25     Scientific product
26 boolean input_string_complete = false; //have we received all the data from the PC
27 boolean sensor_string_complete = false; //have we received all the data from the
28     Atlas Scientific product
29
30 //SD
31 #include <SD.h>
32 File dataFile;
33
34 void setup() {                         //set up the hardware
35     Serial.begin(9600);                 //set baud rate for the hardware serial
36     port_0 to 9600
37
38     myserial.begin(9600);               //set baud rate for the software serial port
39     to 9600
40     inputstring.reserve(10);            //set aside some bytes for receiving data
41     from the PC
42     sensorstring.reserve(30);          //set aside some bytes for receiving data
43     from Atlas Scientific product
44
45 //BLUE ROBOT
46 // Initialize pressure sensor
47 // Returns true if initialization was successful
48 // We can't continue with the rest of the program unless we can initialize the sensor
49 Wire.begin();
```

```

43 while (!sensor.init()) {
44     Serial.println("Init failed!");
45     Serial.println("Are SDA/SCL connected correctly?");
46     Serial.println("Blue Robotics Bar30: White=SDA, Green=SCL");
47     Serial.println("\n\n\n");
48     delay(5000);
49 }
50
51 sensor.setModel(MS5837::MS5837_30BA);
52 sensor.setFluidDensity(1029); //kg/m^3 (saltwater, 997 for freshwater)
53
54 //sets the output from the Arduino to the micro SD reader
55 pinMode(10, OUTPUT);
56 if (!SD.begin(10)) {
57     Serial.println("initialization bad. check card"); //if doesn't init
58     return;
59 }
60 Serial.println("initialization done."); //if does init
61
62 File dataFile = SD.open("dataFile.txt", FILE_WRITE);
63 dataFile.print("#####"); //between data when turned on/off between
64 deployments
65 dataFile.println();
66 dataFile.close();
67 }
68
69 void serialEvent() {
70     char
71     inputstring = Serial.readStringUntil(13);
72     input_string_complete = true;
73     received a completed string from the PC
74 }
75
76 void loop() {
77     if (input_string_complete == true) {
78         in its entirety
79         myserial.print(inputstring);
80         product
81         myserial.print('\r');
82         inputstring = "";
83         input_string_complete = false;
84         received a completed string from the PC
85     }
86
87     if (myserial.available() > 0) {
88         has sent a character
89         char inchar = (char)myserial.read();
90         sensorstring += inchar;
91         if (inchar == '\r') {
92             sensor_string_complete = true;
93         }
94
95         if (sensor_string_complete == true) {
96             product has been received in its entirety
97             if (isdigit(sensorstring[0]) == false) {
98                 digit
99                 Serial.println(sensorstring);
100            }
101            else
102                a digit
103                {
104                    print_EC_data();
105                }
106                sensorstring = "";
107                sensor_string_complete = false;
108                received a completed string from the Atlas Scientific product
109            }
110        }
111    }
112
113    //if the hardware serial port_0 receives a
114    //read the string until we see a <CR>
115    //set the flag used to tell if we have
116
117    //if a string from the PC has been received
118    //send that string to the Atlas Scientific
119    //add a <CR> to the end of the string
120    //clear the string
121    //reset the flag used to tell if we have
122
123    //if we see that the Atlas Scientific product
124    //get the char we just received
125    //add the char to the var called sensorstring
126    //if the incoming character is a <CR>
127    //set the flag
128
129
130    //if a string from the Atlas Scientific
131    //if the first character in the string is a
132    //send that string to the PC's serial monitor
133    //if the first character in the string is NOT
134
135    //then call this function
136
137    //clear the string
138    //reset the flag used to tell if we have
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2289
2290
2291
2292
2293
2294
2295
2295
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2389
2390
2391
2392
2393
2394
2395
2395
2396
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
24
```

```

103 void print_EC_data(void) {
104     char sensorstring_array[30];
105     char *EC;
106     char *TDS;
107     char *SAL;
108     char *GRAV;
109     float f_ec;
110     //this function will pars the string
111     //we make a char array
112     //char pointer used in string parsing
113     //char pointer used in string parsing
114     //char pointer used in string parsing
115     //char pointer used in string parsing
116     //used to hold a floating point number that
117     //is the EC
118
119     sensorstring.toCharArray(sensorstring_array, 30); //convert the string to a char array
120     EC = strtok(sensorstring_array, ",");
121     TDS = strtok(NULL, ",");
122     SAL = strtok(NULL, ",");
123     GRAV = strtok(NULL, ",");
124     //let's pars the array at each comma
125     //let's pars the array at each comma
126     //let's pars the array at each comma
127     //let's pars the array at each comma
128
129     //PRINT TO SERIAL MONITOR
130     //Serial.print("EC:"); //we now print each value we parsed separately
131     Serial.print(EC); //this is the EC value
132     Serial.print(",");
133
134     //Serial.print("TDS:"); //we now print each value we parsed separately
135     Serial.print(TDS); //this is the TDS value
136     Serial.print(",");
137
138     //Serial.print("SAL:"); //we now print each value we parsed separately
139     Serial.print(SAL); //this is the salinity value
140     Serial.print(",");
141
142     //Serial.print("GRAV:"); //we now print each value we parsed separately
143     Serial.print(GRAV); //this is the specific gravity
144     Serial.print(",");
145     //Serial.println(); //this just makes the output easier to read
146
147     //f_ec= atof(EC); //uncomment this line to convert the char to a float
148
149     //Save EC data to SD card
150     File dataFile = SD.open("dataFile.txt", FILE_WRITE);
151     dataFile.print(EC); //this is the EC value
152     dataFile.print(",");
153     dataFile.print(TDS); //this is the TDS value
154     dataFile.print(",");
155     dataFile.print(SAL); //this is the salinity value
156     dataFile.print(",");
157     //dataFile.print(GRAV); //this is the specific gravity
158     //dataFile.print(",");
159
160     //Save Blue Robot data to SD card
161     sensor.read();
162     dataFile.print(sensor.pressure()); //mbar
163     dataFile.print(",");
164     dataFile.print(sensor.temperature()); //C
165     dataFile.print(",");
166     dataFile.print(sensor.depth()); //meters
167     dataFile.println();
168     // closes the file to ensure the file was written:
169     dataFile.close();
170
171     delay(750);
172 }

```

C

MATLAB Program

```

1 %
2 %
3 %
4 %
5 %
6 %
7 % (ASCII font Big from https://www.coolgenerator.com/ascii-text-generator
8 % Homebrew CTD Data Grapher / Visualizer
9 % Author: Sophie Scopazzi
10 % Author Website: sophiescopazzi.com
11 % Date: OCT 2022
12
13 % always
14 clear all
15 close all
16
17 %% INSTRUCTIONS - HOW TO USE THIS CODE
18 % There are comments throughout code as well
19 % If you want to contact me about this code, use my website
20
21 % 1. Put this script in same folder as datafile.txt from CTD
22 % Check name of datafile is the same as in the script
23 % 2. Ensure sensors are in the correct order in the data from the Arduino
24 % script, or none of it will make sense (EC, TDS, sal, etc) in order
25 % 3. Find correct time values in this plot, record for next step
26 % Want data from downward travel, upward water column is disturbed
27 % 4. Re-comment step 3, put the values recorded into DEPLOY DATA PULLING
28 % 5. Plots or subplots as desired. Set xlims and ylims as necessary
29
30 %% 2. DATA INGEST AND CLEANING
31 % datafile from CTD, ensure name is correct, script in same folder as data
32 data = readtable('DATAFILE.TXT');
33
34 % ensure the data is labeled correctly, will depend on Arduino code
35 % get data from data table, BUT it has nans
36 EC_nans      = data(:,1); %EC data
37 TDS_nans     = data(:,2); %TDS
38 sal_nans     = data(:,3); %salinity
39 mbar_nans    = data(:,4); %mbar
40 temp_nans    = data(:,5); %temp in C
41 dmeter_nans  = data(:,6); %depth in meters
42
43 % take nans out of above data
44 EC_table      = rmmissing(EC_nans);
45 TDS_table     = rmmissing(TDS_nans);
46 sal_table     = rmmissing(sal_nans);
47 mbar_table    = rmmissing(mbar_nans);
48 temp_table    = rmmissing(temp_nans);
49 dmeter_table  = rmmissing(dmeter_nans);
50

```

```

51 % convert data from table to array for plotting / manipulation
52 EC = table2array(EC_table);
53 TDS = table2array(TDS_table);
54 sal = table2array(sal_table);
55 mbar = table2array(mbar_table);
56 temp = table2array(temp_table);
57 dmeter = table2array(dmeter_table);
58
59 % convert mbar to feet, incase you want to do that
60 % depth_feet = mbar.*0.033455256555148;
61
62 % convert meters to feet, in case you want to use the Mapping Toolbox
63 % deploy_feet = distdim(dmeter,'meters','feet'); %(needs Mapping Toolbox)
64 dfeet = dmeter.*3.28084; % I didn't want to use Mapping Toolbox
65
66 %% 3. TO FIND CORRECT START / STOP VALUES IN TIME
67 % FIND TIME VALUES IN THIS PLOT than re-comment this out
68 % look in this plot to find where depth values start and stop
69 % this is what you will put in below, the (93:135), etc
70 % uncomment below --- 72-75
71 % n = length(mbar);
72 % time = 1:n;
73 % figure;
74 % plot(time,mbar);
75
76 %% 4. DEPLOY DATA PULLING
77 % take out only parts of the data for deployments
78 % this is the values found 'FIND DEPTH VALUE PLOT'
79 % if you have more or less than four, change it
80
81 % ONE
82 deploy1_EC = EC (93:135,1);
83 deploy1_TDS = TDS (93:135,1);
84 deploy1_feet = dfeet (93:135,1);
85 deploy1_temp = temp (93:135,1);
86 deploy1_sal = sal (93:135,1);
87
88 % TWO
89 deploy2_EC = EC (609:660,1);
90 deploy2_TDS = TDS (609:660,1);
91 deploy2_feet = dfeet (609:660,1);
92 deploy2_temp = temp (609:660,1);
93 deploy2_sal = sal (609:660,1);
94
95 % THREE
96 deploy3_EC = EC (774:799,1);
97 deploy3_TDS = TDS (774:799,1);
98 deploy3_feet = dfeet (774:799,1);
99 deploy3_temp = temp (774:799,1);
100 deploy3_sal = sal (774:799,1);
101
102 % FOUR
103 deploy4_EC = EC (832:979,1);
104 deploy4_TDS = TDS (832:979,1);
105 deploy4_feet = dfeet (832:979,1);
106 deploy4_temp = temp (832:979,1);
107 deploy4_sal = sal (832:979,1);
108
109 %% 5. GRAPHS
110 % however many you did, make that many subplot graphs
111 % or you can put them on their own plots (not subplot them)
112 % these are the same with the numbers changed, 1-4
113 figure;
114 subplot(2,2,1); % subplot since I want four graphs on one figure
115 plot(deploy1_temp,deploy1_feet,'red');
116 title('One');
117 hold
118 plot(deploy1_sal,deploy1_feet,'blue');
119 set(gca,'Ydir','reverse'); % to make zero the top of Y axis
120 ylim([0 40]); % use the depth you deployed
121 xlim([14.5 19]) % use the range of your data

```

```

122 legend('temp','salinity')
123 ylabel('depth (ft)');
124 xlabel('°C | PSU (ppt)');
125
126 subplot(2,2,2);
127 plot(deploy2_temp,deploy2_feet,'red');
128 title('Two');
129 hold
130 plot(deploy2_sal,deploy2_feet,'blue');
131 set(gca,'Ydir','reverse');
132 ylim([0 40]); % use the depth you deployed
133 xlim([14.5 19]) % use the range of your data
134 legend('temp','salinity')
135 ylabel('depth (ft)');
136 xlabel('°C | PSU (ppt)');
137
138 subplot(2,2,3);
139 plot(deploy3_temp,deploy3_feet,'red');
140 title('Three');
141 hold
142 plot(deploy3_sal,deploy3_feet,'blue');
143 set(gca,'Ydir','reverse');
144 ylim([0 40]); % use the depth you deployed
145 xlim([14.5 19]) % use the range of your data
146 legend('temp','salinity')
147 ylabel('depth (ft)');
148 xlabel('°C | PSU (ppt)');
149
150 subplot(2,2,4);
151 plot(deploy4_temp,deploy4_feet,'red');
152 title('Four');
153 hold
154 plot(deploy4_sal,deploy4_feet,'blue');
155 set(gca,'Ydir','reverse');
156 ylim([0 40]); % use the depth you deployed
157 xlim([14.5 19]) % use the range of your data
158 legend('temp','salinity')
159 ylabel('depth (ft)');
160 xlabel('°C | PSU (ppt)');
161
162 %%%%%%%%%%%%%%
163
164 figure;
165 plot(deploy1_temp,deploy1_feet,'red');
166 title('Slow');
167 hold
168 plot(deploy1_sal,deploy1_feet,'blue');
169 set(gca,'Ydir','reverse'); % to make zero the top of Y axis
170 ylim([0 40]); % use the depth you deployed
171 xlim([14.5 19]) % use the range of your data
172 legend('temp','salinity')
173 ylabel('depth (ft)');
174 xlabel('°C | PSU (ppt)');
175
176 figure;
177 plot(deploy4_temp,deploy4_feet,'red');
178 title('Fast');
179 hold
180 plot(deploy4_sal,deploy4_feet,'blue');
181 set(gca,'Ydir','reverse');
182 ylim([0 40]); % use the depth you deployed
183 xlim([14.5 19]) % use the range of your data
184 legend('temp','salinity')
185 ylabel('depth (ft)');
186 xlabel('°C | PSU (ppt)');

```