Appendix A: User Manual

Team #4: RoboSub

Fall 2014 – Spring 2015

Dennis Boyd

Bjorn Campbell

Samantha Cherbonneau

Kevin Matungwa

Elliot Mudrick

The RoboSub and You

Welcome RoboSub 2015-2016 team! Normally the User Manual is intended to be a very formal, 3rd person report explaining a product to a generic theoretical user. For the RoboSub project, however, this is not the case. The intended user for the RoboSub is you, next year's group (and any group that may follow thereafter). Thus, the purpose of this manual is not to explain the sub to a generic hypothetical person, but to explain it in such a way that you can understand it and get started on it as soon as possible. It is a very complicated but thoroughly interesting project that I hope you will enjoy.

The ultimate goal of the RoboSub project is to compete in the AUVSI's Annual International RoboSub Competition. The rules for the competition vary by year, but always consist of similar tasks. Specially, the sub must be able to pass through the validation gate in order to compete at all. It must also follow paths from task to task and of course be able to maintain depth and heading. It is these basic functionalities that have been the focus of this year's team. By the time you receive the sub it should be capable of the following things:

- Maintaining depth.
- Maintaining pitch.
- Maintaining heading (yaw), thus being capable of travelling in a straight line.
- Use the front camera to identify objects of a specified color and calculate the center of them.
- Use the bottom camera to identify path pieces and calculate their angle (relative to the sub).

It will be your goal to use this foundation to make the sub achieve more tasks, with the ultimate hope of competing. The competition page can be found here: http://www.auvsifoundation.org/foundation/competitions/robosub/.

Good luck!

Table of Contents

1. System

The robosub is a rather complex system with many parts. The primary mechanical and electrical design has already been completed, so most of this should be superfluous to you. Listed here are the major components worked with by this year's team, which focused on core functionality. More information on additional attachments, such as the claw mechanism, can be found on the previous years' website: http://eng.fsu.edu/~cifreda/. Anything that is not listed here has not been used or tested for at least a year, and so its functionality cannot be promised.

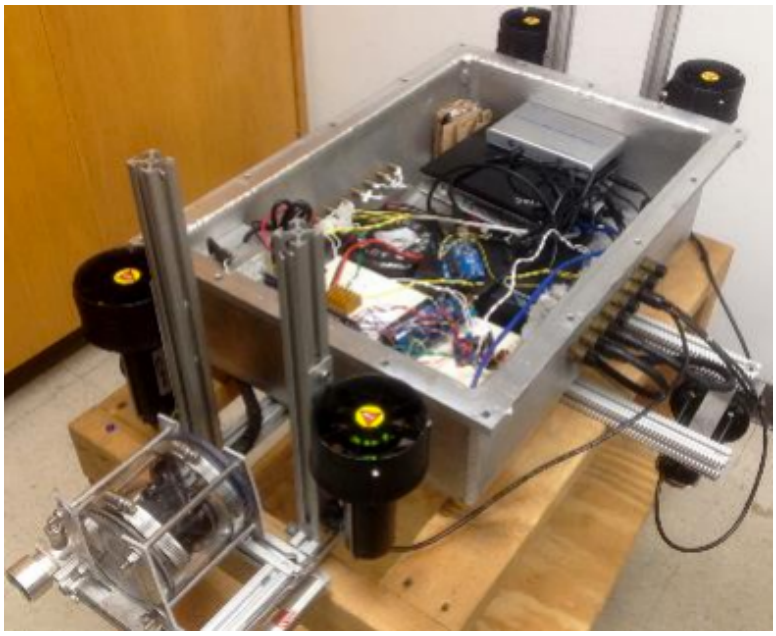

Figure 1: The RoboSub

1.1 List of Components

1.1.1 Hull, Lid, and Frame

The hull consists of a single large aluminum box. It has waterproof ports along the sides: SEACON plugs. These are used to connect external materials (such as the thrusters and cameras) to the main computer inside. Not all components use the same connector, so keep that in mind. If you change anything with these connections, be sure to take pictures so you know what you changed.

The lid is a large chunk of Plexiglas with a seal around it. It is bolted onto the sub with the provided screws, bolts, and washers.

The frame is made of 80-20 aluminum bars and is bolted onto the body.

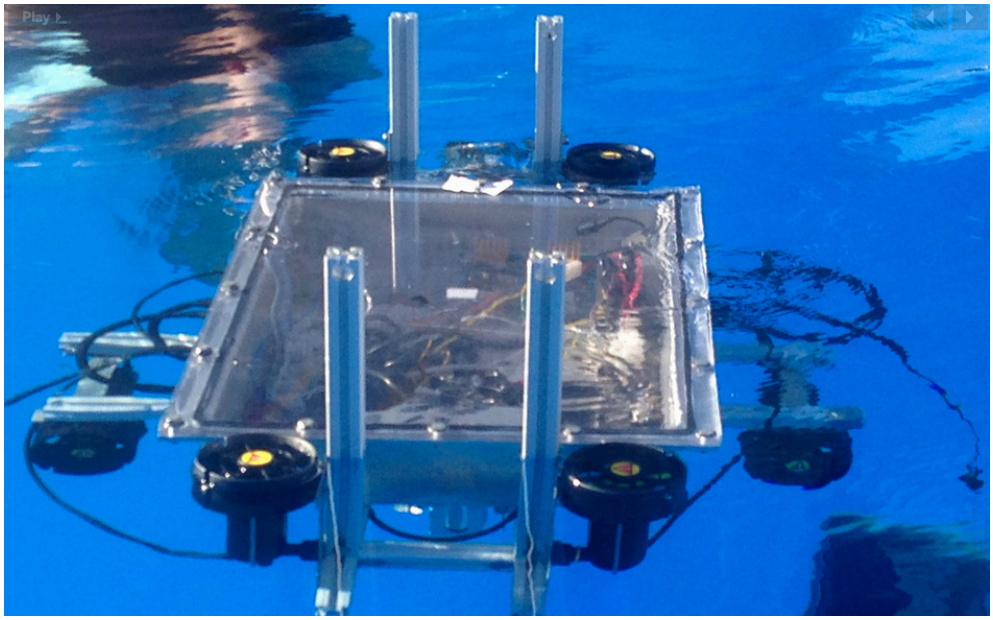

Figure 2: Sub in the water with lid on.

1.1.2 Thrusters

The sub is equipped with six Seabotix thrusters. They can be moved around on the frame to experiment with different orientations. One of the thrusters had its propeller shroud fall off. It has been glued back on, but this is worth watching out for (it was thruster 5 in Figure 20). Also, after removed from the pool, the thrusters maintain a little water, so the next time you run them in air they will spit out a little water. Thus it is recommended to run it one last time after it is removed from the pool to get the water out.

Figure 3: Thruster

1.1.3 Zotac Computer

The main processing unit (MPU) of the sub is one Zotac Intel Core i3-2330M ZBOXHD-ID82-U. The Zotac is the primary brain of the sub, interfacing with the Arduinos, IMU, and cameras. It runs Ubuntu and can be accessed remotely via Ethernet for testing, as described in detail in 3.1.4.

Figure 4: Zotac Computer

1.1.4 Arduino MEGA

An Arduino MEGA 2560 is used to interface from the Zotac to the motor controllers. While most of the equipment is a good three years old, this piece is new. It could potentially be used for other accessories as well.

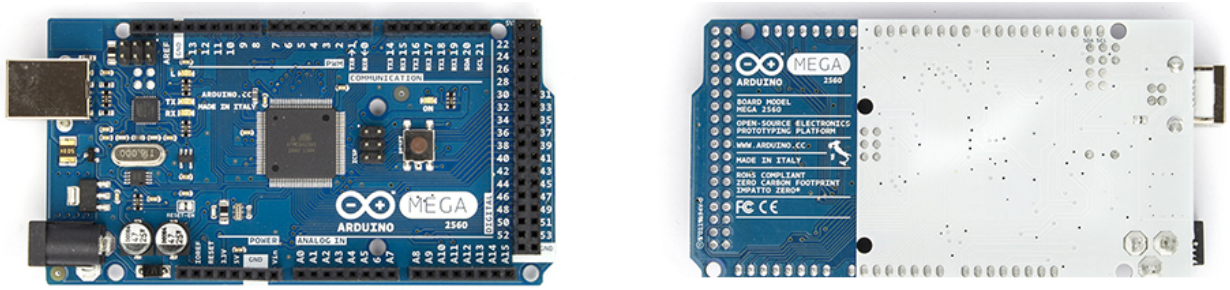

Figure 5: Arduino MEGA

1.1.5 Motor Controllers

The sub is equipped with three L298 motor controllers. There are two varieties: two CanaKit and one Solarbotics. They are functionally the same and can be seen below.

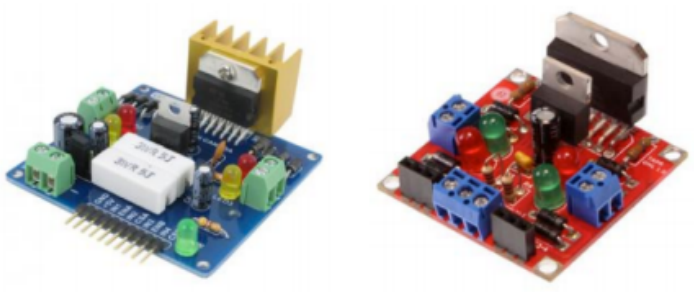

Figure 6: Motor Controllers – KanaKit on the left and Solarbotics on the right

1.1.6 Arduino UNO

An Arduino UNO is used to interface between the depth sensor and the Zotac.

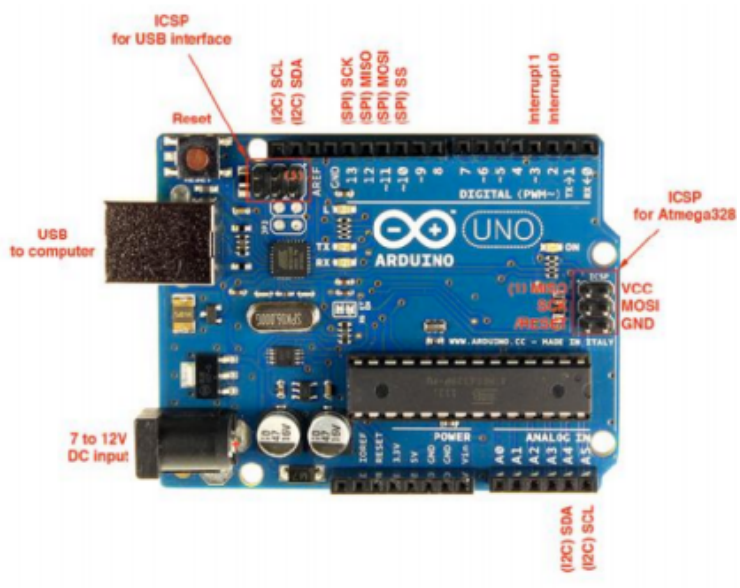

Figure 7: Arduino UNO

1.1.7 Depth Sensor

To measure depth, the sub is equipped with a Keller America Levelgage pressure sensor. It is mounted on the bottom camera with zip-ties.

Figure 8: Depth Sensor and Mounting on Bottom Camera

1.1.8 Inertial Measurement Unit

Attached to the lid via Velcro is the sub's Inertial Measurement Unit (IMU), a SparkFun 9 Degrees of Freedom – Razor IMU. This consists of three parts on a single board:

- ITG-3200 - triple-axis digital-output gyroscope

- ADXL345 - 13-bit resolution, ±16g, triple-axis accelerometer
- HMC5883L - triple-axis, digital magnetometer

It is connected to the Zotac through the FTDI Basic Breakout Board. The connecting wires are soldered and are unlikely to need to be modified.

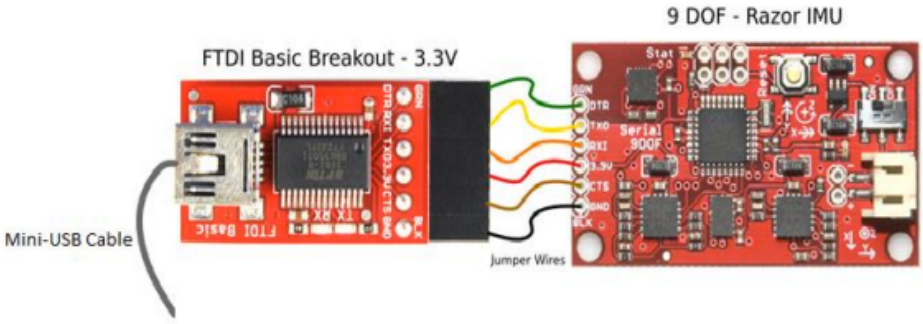

Figure 9: IMU

1.1.9 Cameras

The sub is equipped with two Logitech C615 Webcams to constitute its vision. One is mounted on the front and one on the bottom. Both are waterproofed and connected to the Zotac through the SEACON connectors.

Figure 10: Camera

1.1.10 Batteries and Chargers

The sub is equipped with three batteries. Of these, two have backups. At any given time there are three batteries in the sub, which are used as follows:

Table 1: Batteries

#	Battery	Use
1	Depth Sensor Battery	Powers the depth sensor
2	Universal Laptop Battery	Powers Zotac (through which the MEGA, IMU, and cameras are powered)
3	Lithium Ion Battery Pack	Powers thrusters

There is one new universal laptop battery and one old. The old one has connection issues and reduced battery life, but can be used as a backup, or more accurately a last resort. It should be clear which is the older one by the high degree of wear and tear present on it. There are also two lithium ion battery packs, which are both good. There is only one depth sensor battery, but it holds its charge practically forever, at least while powering only the depth sensor.

Charging the batteries is described in 4.7. The depth sensor battery has no charger right now, but has also never died.

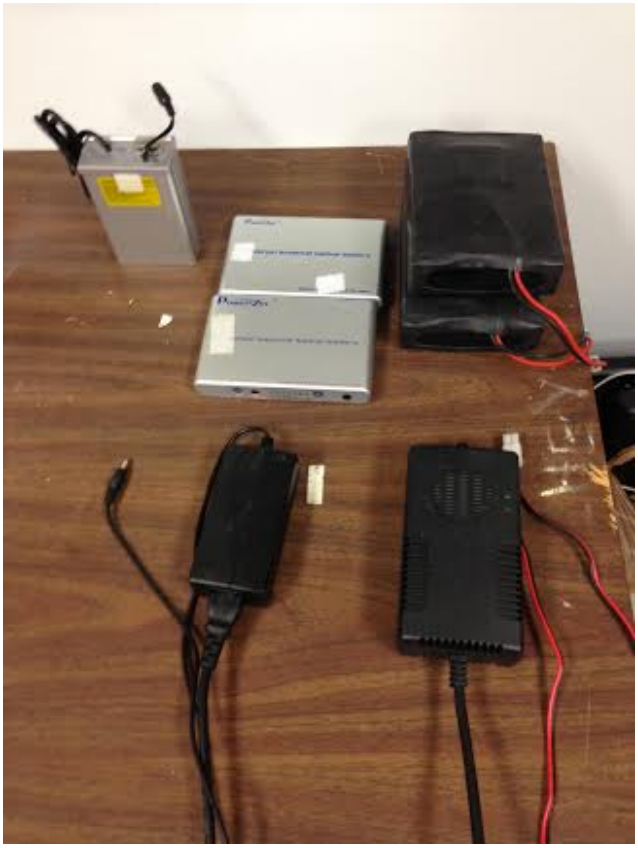

Figure 11: Batteries and Accompanying Chargers - # (from left to right) 1, 2, 3

1.1.11 Zotac Wall Adaptor

There is also a wall adaptor for the Zotac, allowing it to be powered from a socket like a normal computer. This is especially useful for testing in the lab.

Figure 12: Zotac Wall Adaptor

1.1.12 Waterproof Ethernet Cable

There is a 50-foot Ethernet cable available for water testing. One end is the Ethernet port, and the other end is a SEACON connector. This is used for water testing, as it can be used to screen share and give you access to the Zotac while it is in the water and the sub is sealed. That way you can observe debugging output as well as run/cancel code and change/recompile code. It is worth noting that the cable is actually two 25-foot cables spliced together, and the waterproofing of the splicing area is likely to hold, but is not guaranteed.

Figure 13: Ethernet Cable

1.1.13 Ethernet SEACON Substitute

When testing without the Ethernet cable attached to the sub, a stopper has been provided for the Ethernet SEACON connection. With this, you do not need to completely dry the connection after every single run, and can avoid having to lift the sub completely out, as described in 3.5.4.

Figure 14: Stopper

1.1.14 Bolts, Nuts, and Screws

There are various screws and such which hold the sub together. The screws for the lid are 7/16” and the nuts for it are 13mm. The frame uses hex screws: 3/16” for mounting on the frame. Inner frame connections, however, use 5/32” (also hex). You will need to purchase or otherwise acquire some of these wrenches and ratchets. There are also some screws you may need a flat-head or Philips-head screwdriver for.

1.1.15 Gate

A mock gate, for testing, has been built and should be available to you. Its set-up is described in 3.4.1. It can be seen in Figure 17.

1.1.16 Path

Several path segments have also been constructed. They are set up as described in 3.4.2 and can be seen in Figure 18.

1.2 Hierarchy

The overall hardware hierarchy can be described as below.

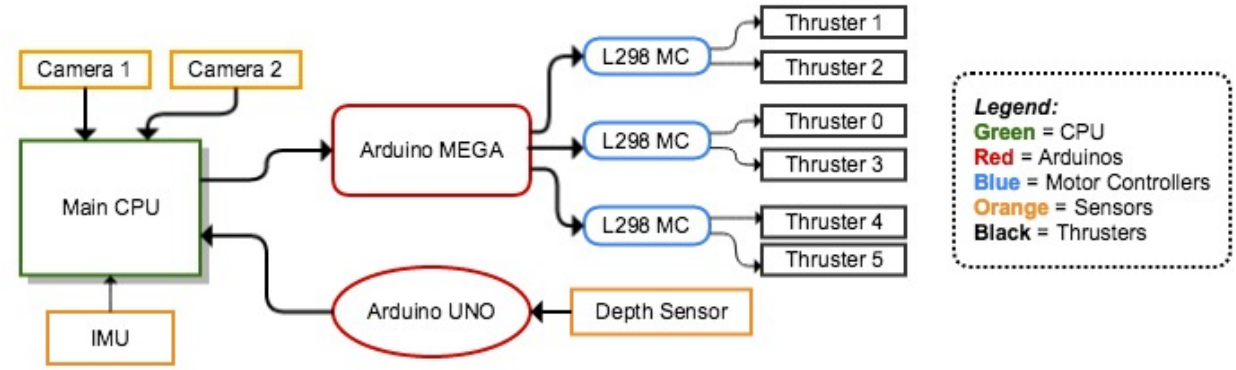

Figure 15: Hardware Overview

1.2.1 Connections

Each camera is connected to the Zotac by USB, as are both Arduinos. The IMU is also connected to the Zotac's USB through mini-USB on the Breakout board. The UNO is connected to the depth sensor by a single wire in analog pin 0. The MEGA's connections are more complicated.

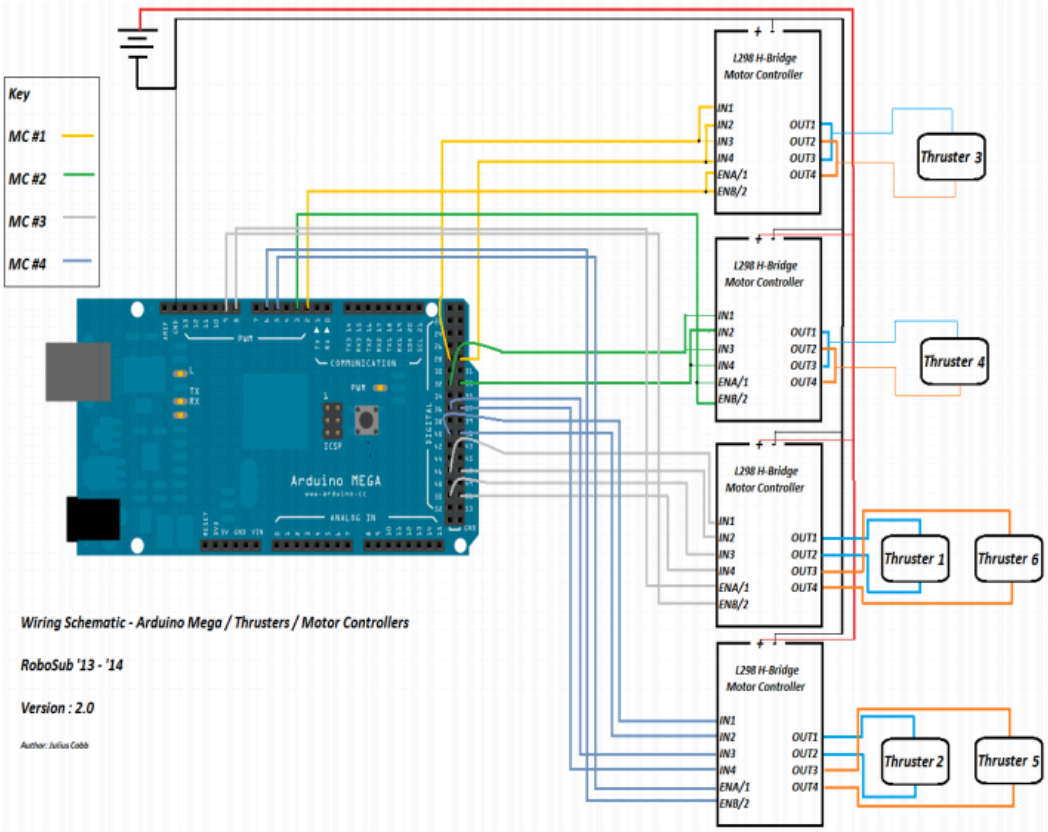

Figure 16: Motor Controller Wiring

Obviously, this is not completely up to date as there are only 3 motor controllers now (also the thrusters have been renumbered to better reflect their representation in the code). However, the only real difference is that only MCs #2-4 are connected, with the remaining thrusters wired as in MC #3 on MC #2. In theory, these connections should never need to be changed. However, stuff goes wrong, motor controllers blow, or things may need to be added, so if you need to re-wire, take good pictures to ensure proper reconnections.

Note: As the sub is made of metal, never place any motor controllers or Arduinos directly on the hull. This is what the cardboard is for.

2. Specifications

This section is mainly for programmers. If you are an Electrical, Mechanical, or other team member not planning to code, you should scan this section, but skip the parts that are irrelevant for you.

2.1 User Specifications

Programming for the RoboSub is no simple task. Just sorting through the mountain of code is quite daunting. This manual is intended to help with that, but there are certain things expected of you as a programmer. No need to worry though, as you can quickly teach yourself anything you do not already know with the help of the Internet. Some helpful pointers will be included here.

2.1.1 C++

The RoboSub is written entirely in C++. The assumption that you are familiar with C++ should be a safe one.

2.1.2 Ubuntu

It is assumed that you have basic understanding of the Ubuntu operating system and Unix terminal navigation. If you are not familiar with Ubuntu, it's pretty simple. It may just take a little time to get used to. Don't afraid to Google. As for navigating Unix terminals, if you have made it to senior year, this should be secondhand to you. Even if you're not, running the sub will be described in detail under Section 3: Operating Instructions.

2.1.3 Pthreads

The RoboSub makes heavy of POSIX threads, or Pthreads. These are taught in Operating Systems, which you may very well have not yet taken. If you have, you can probably skip this part. If not, do not worry. Pthreads are pretty easy. Threading is a concurrent programming technique, something very important for a dynamic robot like this. It needs to be able to maintain its pitch, depth, and heading all simultaneously, for example. An excellent tutorial on using Pthreads can be found here: https://computing.llnl.gov/tutorials/pthreads/.

2.1.4 SQL

The sub uses a mySQL database to store information from the cameras so that the main code can use it. The interaction functions have already been written (such as updateQuery), but if you wish to change the table at all (such as add a new value the camera calculates), some understanding of SQL is important. If you are not familiar with SQL, you should be able to simply copy the syntax of existing implementations to get by. A line-by-line example of how to view the table is in 3.5.2.

2.1.5 Arduino

The RoboSub makes use of two Arduinos as it is now, and may need even more if extra accessories are added. It is not likely you will need to modify the control code or the pressure sensor code. However, if you feel you need to and aren't familiar with Arduinos, they have lots of tutorials on their website. The most basic uploading of a sketch (an Arduino program) can be found here: http://www.dummies.com/how-to/content/how-to-upload-a-sketch-to-an-arduino.html.

2.1.6 PID Controllers

Most of the sub's orientation (such as depth, pitch, and yaw) is controlled through PID (Proportional Integral Derivative) controllers (or to be more specific most are just PD – they lack an integral). If you don't know anything about these, you should be okay if you simply reuse the existing controllers for your purposes. If you do know a lot about PID controllers, you could probably improve what's already there if you want to or design your own for new stuff. For those who don't know anything, an important aspect is determining good values for the tuning constants, called kP and kD in the sub, by testing. That should be enough to get you started anyway.

2.1.7 Diver

At least one person will need to be in the pool with the sub at all times. It is recommended to have a designated diver who will almost always be this person. This person needs to be capable

of swimming alongside the sub to trigger the killswitch if necessary. Most difficultly, he or she needs to be able to dive to the bottom of the 17 foot dive pool in case something happens and something ends up on the bottom of the pool.

2.1.8 Truck

You will need a truck (or large SUV/van) of some kind to transport the sub to the pool. You could wheel it all the way over there every single time you want to test, but this is not only extremely inconvenient, but also dangerous. The cart is vibration heavy, and that is not good for the sub. The stand the sub sits on is not actually attached to the wheeled section, meaning it can be used to place the sub in the trunk of a large vehicle. It fits snugly in the back of a Ford Explorer. This way the amount of time it actually spends being pushed on the cart is minimized.

2.2 Capabilities

As summarized in the intro, the sub has the basic capabilities to perform tasks, as well as several extra capabilities that may be useful. This section will summarize what has been done so far.

2.2.1 Z Control (Maintain Depth)

The sub can maintain its depth (or position along the Z-axis). This is done through two threads: depthSensorMain and zControllerDepth. depthSensorMain is created at start up and simply polls the UNO to get the depth. zControllerDepth then uses this value in a PID controller to maintain depth. It is created in the DMCS in the main code area, though you could potentially create it elsewhere. This thread takes a parameter, the depth you wish it to maintain. Because this is actually passed as a pointer, you can change the depth by simply changing the argument, without having to recreate the thread.

The depth is usually accurate to +/- 1 foot, relative to the value the depth sensor reads. For deep depths it will sometimes overshoot at first, going as low as 1.5 feet below the set point. This is worth keeping in mind since you don't want it to hit the bottom. For shallow depths it sometimes sits on the higher end. The value the depth sensor measures does not always seem to be exact. That is, setting the depth to 7 feet might look more like 10 feet while the sensor reads 7. That said, the sub maintains depths quite consistently, so if you give it 7 intending it to actually go to 10 it should do so fairly consistently. Since it uses a pressure sensor, these values can change from day to day as the atmosphere and temperature change, which is worth keeping in mind. Also, when setting depths, recall that the sensor is placed a good 1.7 feet or so below the top of the sub, so that is the reference point for the depth the sensor reads. Even with all of that, in a good amount of testing, the sub's ability to maintain depth was very good, even while moving.

2.2.2 Gyro-Based Pitch Control

The pitch thread, pitchController, is created in stabilizationMain, as it is always desired. The gyroscope of the IMU is used in a PID controller to allow the sub to maintain its pitch. This assumes that the sub never ends up at any extreme angles. If it ends up pointing up 60 degrees it probably won't be able to fix it, but at that point you will likely have more problems from the internal components shifting. Under normal operation, the sub should be able to maintain its pitch.

2.2.3 Gyro-Based Yaw Control (Maintain Heading)

The sub can maintain its heading using the gyroscope as well. This thread, yawController, is created whenever you want the sub to maintain its current heading. It uses angular velocity to account for drifting off-course. However, very gradual drift may not cause a big enough change in the angular velocity for it to catch it. Because of this, the sub does not go perfectly straight all the time. It does, however, go very close to straight, for significant distances, almost all of the time. You may be able to tweak this a little by changing the thresholds in the loops or the constants.

2.2.4 Forward Movement

By giving the side thrusters values, the sub can move forward (or possibly backwards). The thruster layout is described in 3.5.1. You will likely need a yaw controller to go straight though.

2.2.5 Vision-Based Yaw, Z, X Control

Over the course of the project, several vision-based PIDs were created. None of them have shown consistent success, and none have been tested very much. Thus they are "use at your own risk" if you will.

A yaw controller function (yawControllerVision) using the camera is in the code, the intention being to go through the gate based on the camera seeing it. It proved unnecessary, as the gyro-based yaw controller worked very well, and the vision-based yaw never worked quite right. You are welcome to try to get it to work if you like.

A function also exists, zController(), using the camera. This one is not a thread, but rather called a single time to attempt to submerge based on the z-center of the gate. It also never worked great (in part due to reflections as described in 2.3.9), and since the zControllerDepth works great, it was abandoned. However, you may try to use or modify it if you like.

A thread designed to adjust the sub's speed based on the camera seeing the gate also exists, called xController. It has never been properly tested. It might actually work very well, or it might be horrendous. You can try it out and modify it if you like.

2.2.6 Front Camera Vision

The front camera is capable of identifying objects. If finds them based on their color. It can box an object of the desired color, and find the center of it, uploading this data to database. If there are multiple objects of the same color, it attempts to box them all and find the center of the cluster. This was designed specifically for identifying the center of the gate (which has orange legs), but could be modified to do other things as well. However, because of this, the values it calculates are placed in the "Gate" section of the table in the database.

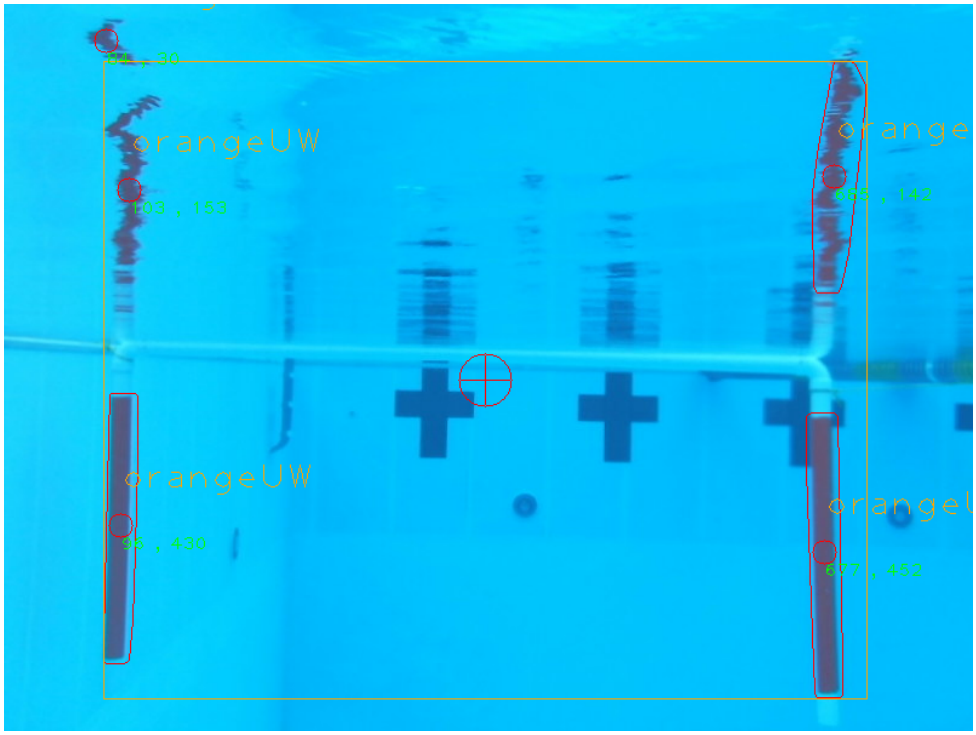

Figure 17: Front Camera Screenshot – Identifying Gate

2.2.7 Bottom Camera Vision

The bottom camera has similar capabilities to the front camera. However, it is designed to be used to identify the path segments, which will link tasks together in the competition. It still identifies objects of a certain color, but instead of trying to find the center, it calculates the object's angle. This is so that the sub can adjust its orientation to match this angle and then follow that heading. Programming the sub to make this adjustment will likely be where you want to start. This angle is placed in the "Path" section of the table.

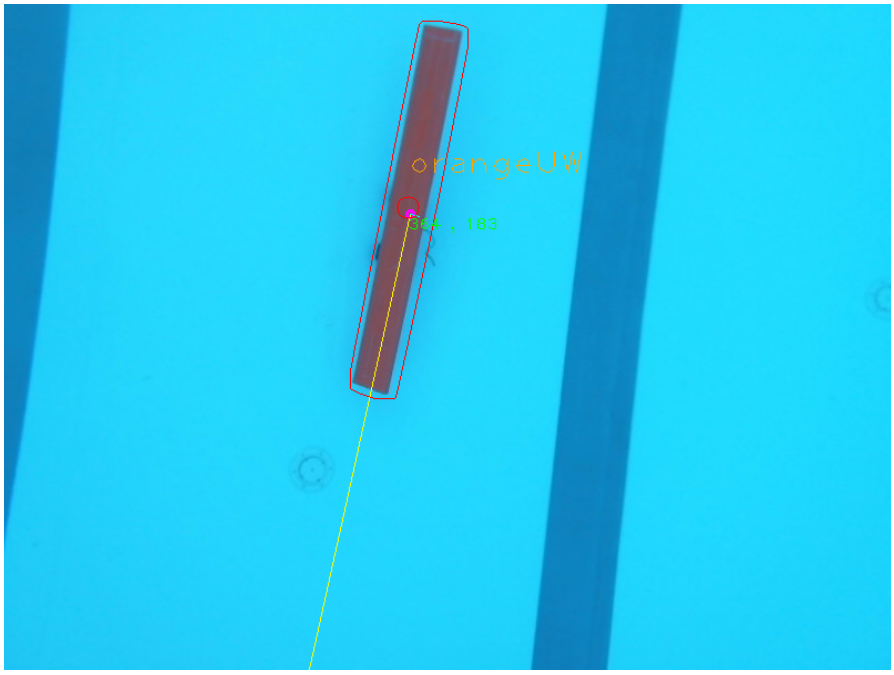

Figure 18: Bottom Camera Screenshot – Identifying Path

2.2.8 Killswitch

The sub is equipped with a killswitch. It is a single pushbutton which, when toggled, cuts the power to the thrusters. It can be seen hanging off the sub to right in Figure 2.

2.3 Limitations

The sub is subject to several noteworthy limitations and quirks, which are outlined here.

2.3.1 Magnetometer Does Not Work

As part of the IMU's 9 degrees of freedom it has a magnetometer which gives roll, pitch, and yaw angles. Originally, these were intended to be used for yaw and pitch control, as simply telling what angle the sub to maintain on a circle is much more precise than correcting angular velocity drift. However, a standard magnetometer does not work inside a metal box, full of electronic equipment, underwater. It gives nonsense values (such as a rotation of 90 degrees being seen as only 60 to the IMU), and is thus basically useless. It is unlikely any use of the IMU's roll, pitch, or yaw readings will be successful.

2.3.2 Weight

The sub is extremely heavy. Sitting somewhere around 120 pounds, it is difficult to move around or lift out of the pool (or even put in the pool depending on whether you have steps). For this reason, it has a cart to be wheeled around on. Furthermore, the maximum weight in the competition is 125 pounds, so that gives little leeway to add more weight.

2.3.3 Buoyancy

The sub is extremely buoyant, almost more so than it is heavy. This is actually one of the reasons it is so heavy: weights have been added inside the hull just try to get it to sink (and a little for balancing reasons). Even so, it needs a significant thrust to force it down, which is why four of

the six thrusters are placed vertically: to give it enough thrust to make it sink. This inherent desire to surface should be considered when programming it, though the depth controller should be able keep it where you want it. It does make it easy to recover when killed though.

2.3.4 Drift

The sub almost always wants to drift. It is in large part for this reason that going straight is difficult. To go straight, the side thrusters are not given even values. Thruster 3 (layout is described in 3.5.1) should be given a slightly larger value to compensate. For example, when going through the gate, thruster 0 has 100 while thruster 3 has 105. The yaw controller is usually able to deal if the thrusters are given even values, but it causes the sub to snake quite a bit and can cause it to end up shifted a little, relative to where it started.

2.3.5 No Y-Control

With the current thruster orientation, the sub is not capable of direct movement along the y-axis. This means it cannot move left and right. It would have to rotate and then go forward. Direct y-axis movement may be very helpful for some of the tasks. Some function prototypes exist in the code for y-based things, but none have been implemented.

2.3.6 Battery Life

The universal laptop battery has severely limited battery life. It usually lasts about an hour with no cameras on, cut to about 30 minutes with the cameras on. Do not trust the batteries' LED charge indicator, as it will claim 75% full and die in a few minutes, sometimes. It is best to keep track of how long you are usually able to run on a full charge and go by that. It does charge pretty quickly though. On the bright side, the lithium ion battery packs last a long time, and the depth sensor battery should almost never need to be charged. This is, of course, with only the things on the sub at time of writing. If more things are added, depending on where they draw their power, battery lives may change completely.

2.3.7 Incomplete Killswitch

The current killswitch is optimized for testing: it kills only the motors. The competition specs, however, require it to kill all electronics. For testing, this would require taking the sub out, unscrewing the lid, turning the Zotac back on, re-screwing the lid, and placing it back in the water every time. That's why it just toggles the motors, but before competition, it needs to be modified to meet the specifications, if they still require it to kill everything.

2.3.8 Screen Sharing only Works on Mac

Being able to share screens, or otherwise somehow remote desktop connect, is essential for testing. Right now, the sub can only share screens with a Mac (which has an Ethernet port, sorry fancy new MacBooks) as detailed in 3.1.4. It should be possible to do so with about any computer (such as by using VNC), but this has not been implemented. So if no one has an Ethernet-ready Mac, being able to connect remotely to the sub will be something you need to figure out.

2.3.9 It is Very Difficult to See Underwater

The title says it all. A large part of your headaches is almost certainly going to be trying to see things underwater, where everything is basically blue, and lighting is the difference between black and bright orange. Some attempts at dealing with the water have already been made, but are far from perfect. The strategy for handling underwater colors was, for this year, to create

color definitions for what colors look like underwater to the camera, as detailed in 3.5.5. The almost certainly better solution would have been to apply filters or something like that, but the filters that were there didn't compile with two cameras. It is up to you how to deal with this limitation. You may very well know a lot more about this sort of thing and put what is there to shame. If so, more power to ya.

Another issue, which may less obvious, is reflections. Underwater, objects can get reflected on the surface, which can through the camera off. This effect can be seen in Figure 17.

3. Operating Instructions

Here it how to make the RoboSub do stuff. It is full of quirks, but at the least, these instructions should be enough for you to easily get it to do what it does at time of writing. It is worth noting that the following considers the "front" of the sub to be where the front camera is, and "left" and "right" mean when looking at the sub from the front.

3.1 Connecting Everything

The RoboSub should still be assembled when you get it. The only plugging of things you should need to do is connecting all of the power sources and a screen (either the monitor or through screen sharing).

3.1.1 Zotac Power

The Zotac can be powered by either the universal laptop battery or the wall adaptor. It is recommended to plug it into the wall in the lab (unless perhaps you are testing how long the battery will last for), but of course, you will need the battery for the pool. When using the battery, set it to 19 Volts and turn it on. Be sure to turn it off, not just unplug it, when done. It sits directly to the right of the Zotac. The Zotac's power button is on its left. You should see a blue ring of light on the computer when it comes on.

3.1.2 Accessory Power

The lithium ion battery pack is used to power the thrusters. It is placed near the center of the sub and its plug should be in the same general area. It is the only plug of its kind, so it should be hard to miss. This battery has no power switch, so it needs to be unplugged when not in use. The depth sensor battery is located at the left front of the sub. It is generally left plugged in. For normal operation, place the power switch in the one setting (or 12 Volts). The cameras, IMU, and Arduinos all draw power from the Zotac directly.

3.1.3 Connecting the Zotac in the Lab

Outside of water, you are unlikely to need to bolt on the lid. Thus, you can control the Zotac like a normal computer: with a mouse, keyboard, and monitor. It has a USB expander hub (D-Link) plugged in already. Plug your wireless mouse (you will need your own wireless mouse) and the keyboard in here. The monitor connection is on the right side of the Zotac.

3.1.4 Connecting to the Zotac for Water Testing

Obviously, you cannot connect the monitor or keyboard while in the pool. This is why the sub is set up with the 50-foot waterproof Ethernet cable. You can share screens with it to remotely access the Zotac. The steps for doing so with your Ethernet-enable MacBook are easy:

1. Turn on the Zotac. (It assumed you left it to auto-login)
2. Plug the Ethernet cable into the sub. It should be clear where the correct SEACON is as it is the one that goes to the Ethernet cable, which is plugged into the Zotac.
3. Plug the Ethernet end of the cable into your Mac.
4. Open Finder.
5. Look for "robosub1314's remote desktop on coerobosub2012-ZBOX-ID82" in the "SHARED" category. Click on it. It may take a little while for it to show up.
6. Click "Share Screen…" in the top right.
7. If it asks, the password is robosub.

Now you should be able to navigate the Zotac with your Mac. Since it is running Ubuntu, some of the keyboard shortcuts are different (such as command+c not being copy), so keep that in mind. There is some lag, and you periodically lag out (with a little "Wired network Disconnected - you are now offline" box in the top right corner). This should not require you to reconnect though. Just wait a couple of seconds and control should be returned to you. If it works right, you theoretically never need a monitor, keyboard, or mouse.

Important: The screen sharing only works when there are no windows open on the Zotac. If there are, you will likely get a pure white screen. You can actually still do things like this, and the Zotac will see your clicks, but you cannot see anything. If you accidentally lock yourself out like this (if the cable got unplugged form your laptop, for example), simply follow the steps under 3.5.4 to reconnect without having to take the sub out of the water and open it up.

3.2 Running the Code

While it is likely most of the time you will want to perform the sealing the lid and sharing screens steps before running any code, knowing what to run to make it work is the most essential, and so was placed first for your convenience.

Disclaimer: There are places with statements such as "it is unlikely you will need to edit…" These are meant to be advice on what I expect will be important to you. By no means take this to mean you should avoid editing them at all cost or ignore them, they are just less likely to be a priority. But I don't know how you're going to do your project, so I may be totally wrong. Furthermore, the code represents at least three years of different teams' work, so I certainly don't have a perfect grasp on everything, we simply understood it well it enough to get it to do what we wanted. It is up your discretion what to change or add.

3.2.1 Logging in

The Zotac is set to auto-login, but if for some reason you need it, the primary login is robosub1314 and the password is champs2013. If, for some reason, you want to login to COE Robosub 2012, that password is robosub.

Figure 19: RoboSub Desktop (as seen through Screen Sharing)
Note: It seems that if you change the background, everything segfaults.

3.2.2 Locating the Files

Once on the desktop, you see a folder there called RoboSub14-15. This is where all of the most recent code is. If you explore, you will find all sorts of folders with various pieces of code from various pieces of the past, but that folder contains the code you want to run. You may make a copy with a different name (such as "RoboSub15-16"), rename that folder, or simply leave it alone, whatever floats your boat.

The folder RS_C contains the main routine, RoboSub_Control_v2.cpp. This will likely be the main file you will be editing. It contains all the PID controllers mentioned above along with the main routine and compiles into the primary executable for the sub.

The folder RoboSubObjectTracking contains the vision and database code. It is unlikely you will need to edit the Database files, but multipleObjectTracking.cpp may be a major point of interest for you. It contains the vision code and compiles into the ColorDetection executable. If you wish to create new colors, they can be added in RoboSubColors.cpp.

The folder Arduino contains, as it says, the Arduino .ino files in use by the sub. These are unlikely to need modification. However, you may want to tune the feetperVolt constant in PressureSensorArduino.ino if you can get it more accurate.

The folder razor-9dof-ahrs-Release-v1.4.2 contains the driver code for the IMU. The code that is flashed on the IMU is located in Arduino/Razor_AHRS/Razor_AHRS.ino and this is what you will edit if you want to re-calibrate the IMU.

The folder IMU contains the important IMU test code referenced in 3.2.6.

3.2.3 Viewing/Editing the Files

You can view or edit all of the files in terminal with your preferred text editor, such as Emacs or vim, but the recommended method of viewing and altering the files in through gedit. Simply click the gedit icon on the sidebar (it calls it "Text Editor" if you hover the curser over it). It will open a blank document. Click "Open", navigate to the file you want, and open it. Now you can edit the files with all the convenient features of your mouse! This is especially handy because, like Word, it keeps track of the most recently opened files, so you rarely have to waste your time swimming through folders. You can then save the files in gedit and make them in the terminal.

3.2.4 Compiling

Makefiles have been provided for the RoboSub_Control_v2 and ColorDetection executable. To compile your changes:

1. Open terminal (if not open).
2. Navigate to the directory of the code you changed.
3. Type make clean.
4. Type make.

Basically, the noteworthy thing is that you need to make clean. RoboSub_Control_v2 can be used with gdb already, though its usefulness for multi-threaded programs is questionable.

3.2.5 Preparing the Terminals

To run the RoboSub fully, there are two executables you will need to run simultaneously: RoboSub_Control_v2 and ColorDetection. There is a third executable you may choose to run, Logger, but I prefer standard couts for debugging so the Logger is not covered in thus manual. Anyway, simply follow these steps to get everything set up:

1. Open a new terminal from the sidebar.
2. Click "file" -> "new tab" and open **two** new tabs.
3. In the very first tab, navigate to the RS_C directory. Typing cd Desktop/RoboSub14-15/RS_C should do the trick, if you have not changed the folder at all.
4. In the second tab, navigate to the RoboSubObjectTracking directory. This time you want cd Desktop/RoboSub14-15/RoboSubObjectTracking.
5. In the third tab, navigate to the IMU directory mentioned in 3.2.2 and its subdirectory C++. You should be able to get there with cd Desktop/RoboSub14-15/IMU/C++.

Now you should have 3 tabs open in terminal, each in a different directory.

3.2.6 The IMU Dance

Perhaps the strangest quirk of the RoboSub is the ritual that is sometimes required to get the IMU to work. The strangest part is the *sometimes*. The IMU *may* work if you do none of these steps. It *may* work if you do half of them. But it will *always* work if you do all of them. In any

case, if you find the IMU giving nonsensically big values, like 3 x 10^9 or nonsensically small values such as 3 x 10^-8, this will fix it:

1. In that third tab, the one in the IMU/C++ directory, run example by typing ./example.
2. Press enter to connect to the IMU. You should see a constant stream of values, which are reasonable in size (less than 1000).
3. Press enter (to cancel the example).
4. Open the Arduino IDE (if it is not already open). It can be found on the sidebar.
5. Open the serial stream monitor (small plus on the far right of the Arduino window). You do not need to have a file open in the IDE, you just need to see the serial stream. It should be streaming some numbers.
6. Close the serial stream.
7. Go back to that 3rd tab and run example again, repeating steps 1 through 3.

Now the IMU should be giving the correct values to the main routine.

3.2.7 Run in the Lab (Out of Water)

Now you are ready to run the code! In the second tab, type ./ColorDetection. This should open the two camera windows. One may be hiding behind the other, so don't worry if you only see one at first. Now in the first tab, type ./RoboSub_Control_v2. And that's it, the RoboSub should be running! It takes a couple seconds for the thrusters to come on while everything is initialized.

3.2.8 Run in the Pool

The steps are the same (3.2.1 – 3.2.7) to run the sub in the pool but you will be performing them through screen sharing (detailed in 3.1.4) and of course will need to have the lid sealed. It is recommended that you always run the code at least once before tightening the lid, to ensure you have plugged everything in and turned all the batteries on and such.

The code that is currently compiled is designed to get the sub to go through the gate, until killed by the person sharing screens. In order for it to do so, the diver needs to aim it at the center of the gate. For best results, the diver should keep the sub aimed correctly for just a second or so once it starts descending so that its starting heading is not messed up.

3.2.9 Canceling the Code

Normally the sub is supposed to run autonomously, and thus must shut down on its own. But when testing it is often most convenient to just run until you have gained all you wanted to gain. In that case, the sub can be killed with control+C in the first tab, like a normal program. However, to stop the thrusters (if they were on), you must then run ./RoboSub_Control_v2 again and kill it quickly. This is because the last values that were sent to the Arduino are still there. You will notice one of the first that things that happens in the DMCS is stopThrusters(), which is why it is actually the second run that stops them. You could probably write your own handler for control+C using SIGINT that stops the thrusters if you want.

3.2.10 USB Ports

Probably the second biggest quirk of the RoboSub is its USB ports. The specific port that something is plugged into may or may not matter. What it comes down to is that the port something is plugged into is the same thing that the code expects. The Arduino IDE-using devices have ports that are defined at the top of RoboSub_Control.v2 as follows:

```
#define COM_PORT "/dev/ttyACM0"     // name of port the Mega is connected to
#define COM_PORT3 "/dev/ttyACM1"    // port defined for the UNO
#define COM_PORT2 "/dev/ttyUSB0"    // port IMU is connected to
```
If they get moved around and show up in the IDE as different things be sure to change that here (or get them to go back to normal).

It is also important that everything is plugged in. This likely has to do with the way serial ports are assigned or something. Basically, what it comes down to is that sometimes if the IMU is not plugged in, the cameras (or usually one of them) won't work. So even if something is not being used, it is advisable to leave it plugged in so the ports don't get confused.

3.3 Sealing the Sub

As a giant box of electrical equipment intended to be sent underwater, it is imperative that the sub remains waterproof. Preparing it for pool testing is fairly straightforward, but here are some instructions/tips anyway.

Important: It is recommended that you test the waterproofness of the seal before putting the sub in the water. It will likely have been around 4 months since it was last used by the time you begin using it. Seals degrade over time, and it may be faulty by the time you get it, though this is unlikely. Nevertheless, it is not worth risking all of the components inside, so you should probably remove all of them and the test the integrity of the seal with the sub empty before placing the whole sub in the water.

3.3.1 Attaching the Lid

For putting on and taking off the lid, there are a few important things to note. The lid is not symmetric, it is marked with electrical tape what side the front camera is on. Use two people to decrease the amount of time it takes. Get two sets of wrenches and ratchets. The size for the ratchet is 7/16" and the size for the wrenches are 13mm. After placing the lid on with all the screws through, attach the nuts to all of the screws. With one person starting on each side of the lid, start tightening the nuts. Tighten every other screw when putting on the lid. Use the ratchet for the screw and the wrench for the nut. Tighten each screw half-way when putting on the lid. After each screw is tightened half-way, tighten all screws until resistance is encountered. Tighten the screws until you cannot tighten any more. If you hear cracking, stop. Take pictures of the lid to make sure that the cracks don't grow. The seal after tightening should become black in almost all places. It is not recommended to use a power screwdriver, as it could easily over tighten and damage the lid.

3.3.2 Removing the Lid

For taking off the lid, loosen each screw slightly. Loosen every other screw, similar to when putting on the lid. After the nuts have been loosened, take all the nuts off. When lifting up the lid a high resistance can be encountered. If this happens use a flathead screwdriver to loosen the lid. When taking the lid off make sure to disconnect the IMU. Also, water gets absorbed in the seal, so be careful and dry the seal after removing the lid. There are sponges along the sides to try to and catch any water that may fall in (though they won't be any good in a case of catastrophic failure).

3.4 Testing Peripheral Setup

Since all of the tasks in the competition involve doing things with, to, or around stuff, you will need to construct your own versions to test on. Thus far, two of these peripherals have been constructed: the gate and the path. This section describes how to use them.

3.4.1 Gate

The gate consists of two vertical and two horizontal 3" PVC pipe. The vertical pipes have 90-degree elbows attached. The gate will have the horizontal pipe on the surface of the water. Use two people to build the gate. One person should get in the water. Give the vertical pipe to the person in the water, then connect a horizontal piece to it. Place the horizontal pipe on the lip of the pool. Make sure the horizontal pipe won't move. Do the same thing for the other vertical pipe. Then connect the horizontal pipes. The gate should then float on top of the water. Attach rope to the floating buoys and something on land to prevent the gate from moving. If you want the gate submerged, attach the ropes to the gate before submerging the gate. Have two people on land hold a connected vertical and horizontal piece, submerged. Then connect the horizontal parts together. Adjust the rope length to the desired submerge depth.

To take out the gate, bring it to the side of the pool. Have two people remove the gate and then disassemble it. It can come apart in the pool, but this happens rarely enough that it is not worth gluing the whole thing together, as this would make transporting it extremely difficult.

3.4.2 Path

The path segments have already been bought, holed, and grooved. Attach rope through a hole, and tie a knot at the grooved end. Place blaze orange electrical tape over the knot. The other end of the rope should be tied to a 5-10 pound weight (which you will need to acquire). Make sure that the length of rope is equal for both hole locations. When placing the path in the pool a diver is required. Dive down with the weight, and gently place the weight on the bottom of the pool. When diving down equalize your air pressure by blowing out your ears, if needed. For bringing the path up, a strong diver is required. The addition of the weight greatly increases the complexity of the dive. For our path, we used a 30lb weight. This is not recommended; the weight was very high which causes problems bringing it up.

3.5 Tutorials

This section is dedicated some brief tutorials and explanations for the stuff on the RoboSub.

3.5.1 Thruster Layout

In `RoboSub_Control_v2.cpp` you will find a six-element array called `dataBuf`. This array is how you change the values of the thrusters. Each element of the array is one of the thrusters (0 through 5) as shown below:

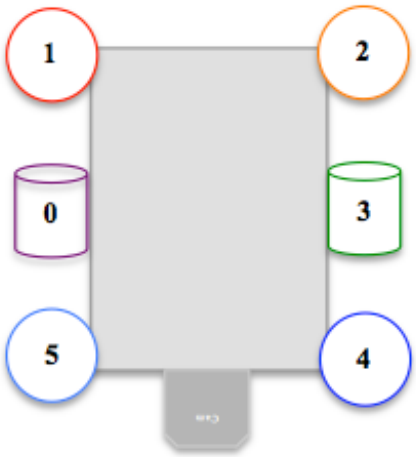

Figure 20: Thruster Layout

Table 2: Thruster Values for Movement

Type of Movement	5	4	1	2	0	3
Forwards	Off	Off	Off	Off	+ On	+ On
Reverse	Off	Off	Off	Off	-- On	-- On
Rotate left	Off	Off	Off	Off	+ On	Off
Rotate right	Off	Off	Off	Off	Off	+ On
Ascend	+ On	+ On	+ On	+ On	Off	Off
Descend	-- On	-- On	-- On	-- On	Off	Off

So if you want it to rotate right, for example, you would write:

```
dataBuf[3] = 60;
```

The exact value depends on how fast you want the movement of course, but that is the idea.
It is worth mentioning that the sub's buoyancy means that most of the time you could ascend by simply turning thrusters 5, 4, 1, and 2 off, though it will likely do so slowly.

3.5.2 Seeing the Database

The database can be found by opening a terminal and entering the following commands as you would normally do in Unix (hit enter after each command):

```
mysql -u root
show databases;
use RoboSub2014;
show tables;
SELECT * FROM Tasks_List;
```

Don't forget that you need the semi-colons in SQL. You can also edit the table from here. If you run the vision code in another tab, you can see the table being updated by periodically re-entering the last command (which you can also get to by using the up arrow, like in Unix).

3.5.3 Calibrating the IMU

The IMU is very finicky. It has been calibrated, but it was optimized for the gyroscope. If you feel you need to calibrate it again, or can do better (especially if you want to use accelerometer

data), instructions for doing so can be found here: https://github.com/ptrbrtz/razor-9dof-ahrs/wiki/Tutorial. You should be able to skip over the "Setting up the hardware" section. The Razor_AHRS.ino file it asks for is in Desktop/RoboSub14-15/ razor-9dof-ahrs-Release-v1.4.2/Arduino/Razor_AHRS/Razor_AHRS.ino as described in 3.2.2 already.

3.5.4 Testing Without the Ethernet Cable

At some point, you may want to test the sub when it is truly autonomous, without the Ethernet cable. When doing so, you can avoid having to completely lift the sub out of the water every time you want to change something by using the stopper in Figure 12. Simply share screens with the Zotac as normal, run the code you want (remember to include a sleep or something to give time to position the sub), unplug the SEACON end of the Ethernet, and put the stopper in its place. Now when you want to share screens again you can lift the sub out of the water just enough to swap the stopper for the Ethernet cable. Do be sure to dry the sides a little, as you remove the plug, as you don't want any water in the connection.

The problem that arises with this, or trying to reconnect to the Zotac in general, is that you need to be on a clean desktop for the screen share to work right. To account for this, some keyboard shortcuts have been made to make use of Ubuntu's Workspace system:

- Shift + alt + left -> workspace 1
- Shift + alt + right -> workspace 2

So if you need to share screens but the Zotac has windows open (such as when it was running the code) and you're getting the white screen, you can get back in this way:

1. Share screens. This is the one where you get the white screen.
2. Switch to workspace 2 using the keyboard shortcut. You won't see any noticeable change, but it should clear by step 4 whether or not the command went through.
3. Disconnect.
4. Share screens again. This time you should be in, on the black desktop in workspace 2.
5. Switch back over to workspace 1.

This assumes of course that you have left workspace 2 empty. If you implement a difference screen sharing technique, this may not even be a problem for you.

3.5.5 Seeing What the Camera Sees

When you run Color Detection, two windows will open (1 for each camera). They do more than just show images though; they also have some useful numbers. Particularly, you can use the mouse to find out the RGB values of the images. Simply hold it over something and the RGB values of that pixel can be seen in the bottom left. This is how the camera looks for orange objects underwater, by looking for a different color entirely: orangeUW, which was created based on the RGB values the camera sees while underwater. Colors are defined in RoboSubColors.cpp using HSV values, so you do have to translate the RGB values from the camera to HSV values. A tool to do so can be found here: http://www.rapidtables.com/convert/color/rgb-to-hsv.htm.

4. Maintenance Guide

The RoboSub should not require much routine maintenance, but like anything, it needs a little love every now and again.

4.1 Lid
The lid has internal cracks. This is a huge risk. You should consider replacing it. However, if the lid is not replaced, take pictures of the cracks and monitor their change.

4.2 Seal
The seal needs to be monitored to make sure that no cracks occur. If the seal fails, the result is catastrophic. The seal absorbs water when wet, so make sure that after each dive the seal is dried.

4.3 Screws/Nuts
The screws and nuts get worn and fragile after extensive exposure to water. Consider replacing nuts and screws that do not tighten easily. Also, after each time, dry each nut and screw to increase lifetime.

4.4 Frame
Because of vibrations from transportation and operation, the frame screws need tightening periodically. Use the prorate size hex screws to tighten. The frame should be tightened as needed, or at least weekly. The frame attachment to the aluminum body requires a Philips screwdriver.

4.5 Seabotix Thrusters
Monitor the wire coming out of each thruster. If needed, apply new liquid electrical tape.

4.6 SEACON Connectors
All SEACON connectors need to be monitored for waterproofness. If waterproof integrity is damaged, replace or apply new heat shrink tubing. Apply liquid electrical tape when needed.

4.7 Batteries
To maintain the universal laptop batteries, do not use them until they are dead (you want them to power off only when you turn them off). The batteries do not last very long if both cameras are running. For the new battery, it lasted about an hour to an hour and a half. The old battery is very sensitive. If it dies it might not come back. Only use the old battery for less than one hour. To charge the universal batteries use the charger and connect to the inn. The charger will shut off after it is done charging. The charge to use time ratio is approximately 2 to 1. For the lithium ion battery, there are two batteries. Keep both charged. Charge the batteries after each use. The charger will over charge the batteries if left alone. When charging, the charger is red and changes to green when charged. For the depth sensor battery, a new charger is required.

5. Recommendations
Here are simply some recommendations for some things that may come up or just generally make your life a little easier.

5.1 New Kart Wheels
The kart is very shaky. Its wheels are not very high quality. If you need to push it around a lot (such as because you don’t have access to a truck to place it in), you should seriously consider getting better wheels.

5.2 Replace the Lid

The lid is quite old. It has many cracks from over-tightening and just general wear and tear. It has held up for this long, but its continued integrity is somewhat questionable. Thus, it may be worth considering replacing it. If you do, you should be sure not to glue the O-ring seal on (as it has been done before), so that it can easily be replaced as it grows old. Using a different material, which is less prone to cracking from over-tightening, may be a good idea.

5.3 Add Y-Control

Some tasks would greatly benefit from the ability to slide side to side. With the current layout, this is impossible. It may be worth it for you to rearrange the thrusters so that two of them are along the y-axis (and thus perpendicular to the side thrusters). They could be placed on the bottom of the sub, as the frame can easily accommodate that. This orientation was tried at one point, but the buoyancy of the sub was a serious problem with only two z-direction thrusters. You may be able to get it to work though.

Another option would be to get two more thrusters. Add these to the bottom and you would still have four vertical thrusters to handle depth and balance. One extra motor controller would not be too hard to add. Sufficient power would be a concern though. The most difficult part, however, would likely be modifying the dense `RoboSub_Control.ino` file to handle the new thrusters, but it should be doable.