Project Albatross

Subsurface Data Acquisition
Using Semi-Autonomous Aquatic Robotics

Northeastern University
Capstone Final Report

Team Members:
Steven Seeberger
Dave Evans
Patrick Gannon
Nick Sullo
Connal West

Advisor:
Bahram Shafai

Table of Contents

I. Abstract

Project Albatross encompasses the design, construction, and testing of a semi-autonomous robotic system for profiling horizontal layers of bodies of water for various indicators of health. In response to the negative effects that climate change, extreme weather patterns, and pollution have had on the world's oceans and lakes, improved water quality monitoring systems are quickly becoming necessary. Currently, marine biologists still rely mainly on manual data capturing methods for performing water quality analyses. Consequently, their systems create sparse, vertically profiled data sets that suffer from missing data and aliasing. The solution proposed here will enable researchers to autonomously collect dense, horizontally profiled water quality measurements, thus decreasing data collection time and cost while increasing data set size and accuracy.

Project Albatross consists of three subsystems: a Ground Control Station (GCS), an unmanned surface vehicle (USV), and an autonomous underwater vehicle (AUV). Using the open-source ground control software MissionPlanner, the user plans a mission for the USV, which is sent over a telemetry radio connection to the Pixhawk flight controller that controls the USV's autonomous motion. The AUV subsystem is tethered to the USV, and is therefore towed by the USV as it completes its mission. Water quality monitoring sensors installed on the AUV (and controlled by an Arduino Uno) capture data as the mission is completed, while the AUV also attempts to autonomously remain at the water depth specified in the mission. The data parameters captured are conductivity, temperature, depth, pH, turbidity, and dissolved oxygen. Data is sent to the USV via a serial connection over the tethered wire, where it is stored locally on a Rasberry Pi and also transmitted over radio to the GCS for real-time monitoring.

To implement the system, both the USV and the AUV were constructed by the team based on their custom designs. Subsystems were tested both individually and holistically to ensure the optimized performance and minimized risk of the system's integration. Field testing was performed in various bodies of water throughout the Boston area, with Jamaica Pond being the primary testing location. The captured results indicate that the system is successfully able to autonomously follow a pre-planned mission of GPS waypoints, capture pertinent water quality measurements at the water's surface and at minor water depths, and transmit this data to the GCS for real time data monitoring. While the system is limited by the low speed of the USV and the low maximum diving depth of the AUV, the team concludes that the system's performance is still sufficient for simple water data collection and could be easily improved in the future with updated mechanical designs.

II. Introduction

With the increasingly detrimental effects of environmental issues such as climate change, extreme weather patterns, and pollution that our world is beginning to witness today, proper environmental monitoring systems are becoming crucial as scientists are attempting to better understand and develop solutions for these growing problems. In particular, methods for water quality monitoring and analysis are currently experiencing much advancement to meet a heightened need for applications like pollution detection, pollution source localization, and wildlife health examination. To perform such necessary environmental monitoring, applied robotic systems are being developed and utilized as important measurement tools.

Multiple robots have been developed for environmental monitoring purposes in both freshwater and saltwater bodies [1]. Early examples consist mainly of remotely operated vehicles (or ROVs) that required constant human interaction for successful operation. More recently, however, research has been centered around the creation of robots that can move and capture data autonomously or semi-autonomously. These robots typically fall into one of two categories: AUVs (autonomous underwater vehicles) or ASVs (autonomous surface vehicles). AUVs are fully-submersible watercrafts that are capable of capturing data at different depths within a body of water, while ASVs are floating vehicles that only capture data at the surface of the water.

Initial AUV designs that became prominent include the REMUS system [2] and the AutoSub system [3]. Both of these robots were capable of diving to substantial water depths (150 meters for REMUS and 750 meters for AutoSub), taking valid marine measurements (such as temperature, turbidity, and pH), and performing autonomous navigation tasks. Other significant AUVs include underwater "floats" such as the Argo profiling float, which have no horizontal motion capabilities other than the effects of wave forces but can sink and rise to specific depths to take important measurements throughout the water column [4]. The prime advantage of these systems is their energy efficiency, which allows them to perform missions that span multiple days. Still other AUVs such as the MantaRay Microplastic Sampler, which is currently under development, seek to quantify and identify microplastics that are present in a given water body autonomously [5]. These robots are advantageous mainly in their ability to save countless man-hours that are currently spent manually collecting and analyzing microplastics to determine pollutant characteristics and concentrations.

ASVs are traditionally used when capturing marine data at the surface or moderately below the surface of a body of water is sufficient for the water analysis being performed. Although the abilities of ASVs are thus somewhat limited when compared to those of AUVs, ASVs are still highly advantageous for their simpler control, communication, and power efficiency implementations. One significantly successful water monitoring ASV example is the Wave Glider, which uses a set of passive wings submerged a few meters below the surface vehicle to convert wave motion into thrust that drives the surface vehicle forward [6]. This energy efficient design, along with the vehicle's integrated solar panels, has allowed the ASV to conduct autonomous missions that exceed one year in length [6]. Some ASVs have become well-commercialized for marine science data acquisition as well. Multiple ASVs produced by the company ASV Global have been sold directly to large scientific consumers for water quality data acquisition purposes, including the C-Enduro which can undergo 30+ day missions and contains

a diverse set of data acquisition sensors (including conductivity, temperature, depth, and meteorological sensors) [7]. Other ASVs have found biological inspirations in their designs to help them better accomplish more specific water analysis tasks. In particular, the Envirobot, which is still under development, is an eel-inspired ASV that uses anguilliform swimming to achieve highly power efficient motion as it autonomously captures water quality data along a pre-defined path [8]. The robot is expected to be further utilized in the future for autonomous pollutant localization in various aquatic environments [8].

While there is certainly not a shortage of viable aquatic robots for scientific data acquisition and water quality analysis applications, current AUV and ASV offerings still possess numerous areas for potential improvement. Namely, one such shortcoming of the existing systems is their high development and selling costs, which presents a large adoption barrier for researchers and organizations who might consider utilizing these robots. Water robots that are labeled as "cost-effective" typically cost as low as $5,000 to build, while more sophisticated and highly commercial robots are sold for as much as $500,000 [9]. This price range is considered to be too expensive for many marine science groups, especially when multiple robots are needed to adequately perform a study. Aside from costs, most water-analyzing robots are also very difficult to deploy and operate. With bulky physical layouts and non-intuitive control interfaces, most AUVs and ASVs require specially trained personnel for proper use. Additionally, a major area of research for future AUVs and ASVs is to improve capabilities for adaptive sampling and mission planning. Adaptive sampling refers to the robot's ability to take measurements from its sensors at different rates (i.e. with different water distance gaps between them) based on the process that is attempting to be measured. Adaptive sampling allows the robot to become much less susceptible to the sample-aliasing issues that plague researchers who perform manual data capturing (which is data capturing by humans directly without the use of robotics). Adaptive mission planning refers to the robot being able to alter its path mid-mission, either autonomously or as a result of operator influence. This capability is highly sought after in newly researched systems because of its potential to allow robots to autonomously localize and map pollution concentrations in pertinent environments. Research is also being performed to determine methods for improving the reliability, safety, and endurance (mission length) of present-day water monitoring robotic systems. Here, greater endurance is key for performing longer studies that will be more useful for certain organizations. Finally, there is a substantial need for more built-in data analysis tools that give researchers more useful and potentially more complex insights into the sensor data they are collecting. While both real-time and post-mission data analysis tools are valuable, better real-time analysis software would allow researchers to more intelligently plan mission paths and objectives.

After exploring the current state of water quality monitoring robotics and understanding the issues still faced by the field, we decided to create a new semi-autonomous robot for water quality monitoring and analysis purposes. The robot is a cross between traditional ASV-type and AUV-type systems, as it is designed as a surface vehicle with a deployable subsurface vessel for recording data measurements at various water depths. The AUV submersible features a modular design capable of supporting user-customized sensors, such as temperature, conductivity, and pH sensors. The surface vehicle section is controlled using a ground control computer station located on the shoreline that is running a computer application for robot path planning, control communication, data communication, and data analysis. Although other researchers have

attempted similar designs before [10] [11], our vehicle is unique in its extremely low development cost (around $1,000), its ability to capture data at multiple depths while sampling frequently enough to avoid aliasing, and its dynamic control interface software that features easily adaptable mission planning and advanced data analysis tools (both for real-time and post-mission analysis). Our project achieves the main objective of creating a low-cost, easy-to-use, and data-insightful robotic system for water quality monitoring and research purposes.

III. Problem Formulation

Our goal for this project was to create a semi-autonomous aquatic robot that is capable of sampling bodies of water at different depths for various readings in order to paint a complete picture of the water quality in said body of water. This device includes a physical drone in the form of a surface "boat" accompanied by an underwater submersible equipped with various sensors that are able to take readings at specified depths. This system is connected wirelessly to a Ground Control Station (GCS) that dictates and monitors drone missions while processing data readings from the mission simultaneously.

There are two main physical assembly problems that needed to be solved for this project to be successful. The first of these was inherent to the environment that we were working in, which was waterproofing the bulk of our electronics. For both the surface vehicle and the underwater vehicle, it was imperative that the sensitive components were either designed to be waterproof or were isolated from all water sources.

The next problem was that of movement through the water. As our project goal required the navigation of a 3D space underwater, some method for controlling the latitude, longitude, and depth of the sensors had to be accurately implemented in order to guarantee the data captured was associated with the correct positioning.

In addition to these main problems, several smaller ones also existed, such as that of buoyancy. This problem was twofold, in that we needed to design our surface vehicle with enough buoyancy to allow it to float properly and that we needed to design our underwater vehicle with a variable buoyancy so that it could rise/sink to specified depths when desired. Modularity was another minor design problem, as a goal for this project was to enable a variety of data collection technologies that could be switched out and replaced at the user's desire. This required discovering how to build standardized modular housings for a set of non-standard sized sensors.

Furthermore, we had to consider how our robot was to perform the data collection that was imperative for accurate water quality analysis. Water data collection is usually performed either by taking physical water samples for later laboratory analysis or by using specialized sensors for measuring water characteristics directly. In discerning which data collection method we wanted to implement, we evaluated potential sensors based on their cost, ease of implementation, and data accuracy (including sensor maximum sampling rate). Additionally, we had to determine the number and types of sensors that needed to be simultaneously implemented for our system to be capable of performing a complete water quality analysis. While we always planned to make our system modular so that it could support a range of user-customized data collection technologies, we needed to understand how many and what types of methods the system had to support for it to be useful for our end users. To properly interface with and store results from these data collection technologies, we also needed to decide what platform (i.e. what microcontroller) would best integrate our various data collection modules.

As discussed above, some form of wireless communication between the water robot and the GCS was required, both for monitoring the mission and sending the recorded sensor data back to the GCS. The communication goals required a robust communication system that could achieve the

longest transmission ranges possible, and with the best data rates possible. In order to run autonomous missions, we needed to solve the problem of making the water robot able to accomplish tasks on its own. After researching these types of projects it became clear that we needed some combination of flight controller and on-land GCS in order to accomplish these tasks. To further discuss this issue, we needed to decide on the appropriate hardware for our situation, and then make it interface with the software we designed to control the drone itself.

There were a few software design problems that needed to be solved for this project to be fruitful as well. The first of which was gathering the data from the robot, which required the robot to either wirelessly transmit the data while it was running or store the data locally for later retrieval when it had completed its mission. Another design was to have a combination of both data storage methods. Once the data was gathered, we needed a real-time application to visualize the data so the operator could easily see patterns in the data. Programming this application could have been performed with various languages, each with its own advantages and disadvantages. One important element to remember is that both control and data analysis functionalities needed to be contained within a single application, meaning that user interface (UI) design was also important. The UI for the data analysis component had to be both intuitive and compact, so as to not interfere with the control side of the application. The overall software goal was to have a complete application that could control the robot and interact with the data it supplied in a fully intuitive and appealing way.

IV. Analysis

The following section outlines the analysis for the major systems and problems in our design.

A. Physical Assembly

The physical assembly and housing for all of our components is a problem that influences how the rest of the components and software issues get handled. There are currently two main approaches that are being used. One method that is used is to deploy a submersible to the body of water. This option provides direct control over the positioning of all the sensors, as all motors are directly controlling the sensors rather than passively lowering them into the water. Many submersibles also come equipped with a device for directly interacting with the environment, such as a suction hose or an arm. In our system, these options would allow us methods for collecting physical samples in addition to collecting debris.

However, a fully submersible device provides several more challenges across the scope of the project. The first of which would be a steep increase in cost. Providing full range of movement in a 3D space requires a minimum of 4 motors, with most builds relying on six motors, as seen in Figure 1 [12] [13]. This increase in motors also increases the difficulty in controlling the vehicle as it now has to navigate a 3D space on its own. The buoyancy of the device must be as close to neutral as possible as well, to prevent unintended vertical movement during measurement taking. A separate, attached housing would be required to float above the vehicle to provide GPS reading to occur and be relayed down to the vehicle. This leads to a design, not unlike the ROV from Figure 1, but connected to a buoy containing the GPS elements.

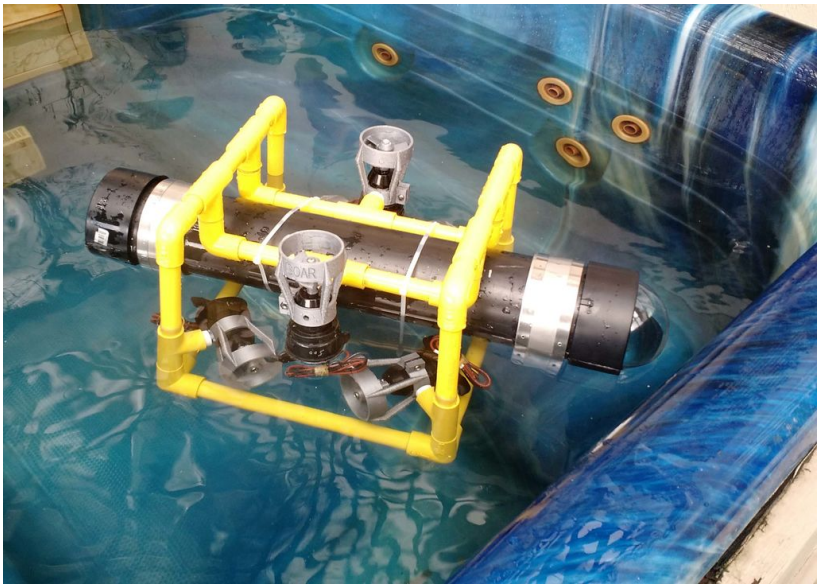

Figure 1: Example of a 6-thruster Homemade Submersible

The other primary method being used to collect data from water is to utilize a CTD system [14]. The canisters are lowered from a platform or attached to a submersible and brought to a certain depth, as seen in Figure 2. Once there, they take conductivity, temperature, and depth measurements, and in some cases light scattering readings, dissolved oxygen, or even physical

samples. Using this approach, we could lower our sensors from a platform that houses the majority of our onboard electronics and power, isolating more of the equipment in case of a breach in the submersible component. This design can use a minimum of two waterproof motors for movement as well as a third for controlling the depth of the sensors from a crane mechanism. The steering for this boat would only require navigating a 2D plane, while the sensor depth can be handled independently from the movement. Neutral buoyancy no longer becomes a concern, needing only positive buoyancy for our main boat, and negative for our sensors. The cost for this increase in simplicity is a significant loss in control over the positioning of the sensors. As they are only controlled along a single axis, they are much more susceptible to currents moving them out from underneath the primary vehicle.

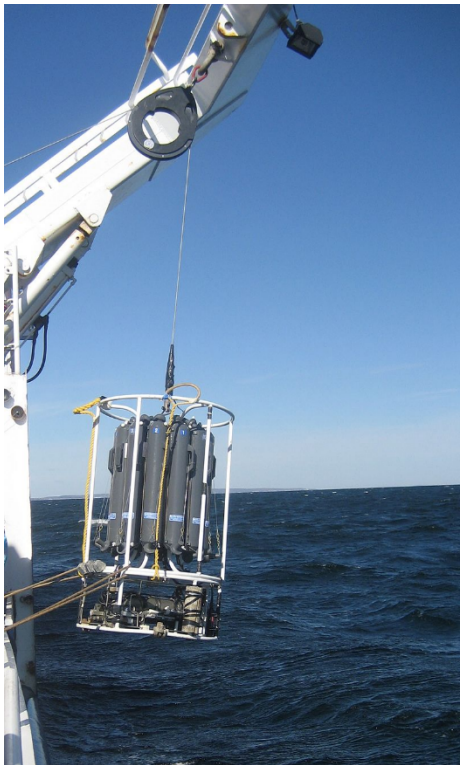

Figure 2: Niel Brown Mk3 CTD Deployment

While both methods are useful for certain applications, they both have significant drawbacks. The fully submersible vehicle allows for a wide range of movements and measurements at all feasible depths and positions. However once underwater, its range is limited by cable length as many require some connection to a radio/GPS module for communication and positioning. Those that do not require this cable need to resurface periodically to perform communication and positioning tasks, thus elongating mission times and wasting energy. It is also extremely difficult to know the GPS location of such a vessel underwater. CTD systems have a much larger range and do not face the same navigation and communication issues. However, they are limited to only being able to measure vertical profiles at sporadic coordinates. This process allows for a large degree of aliasing between vertical profiles at different locations, thus decreasing accuracy in trend measurements. These systems also typically involve much interaction from a trained crew of professional engineers and marine biologists, thus increasing mission time and cost.

B. Data Collection

There are two chief methods that could have potentially been implemented to perform the necessary data collection for analyzing water quality. The first consists of equipping the aquatic robot with a myriad of sensors capable of measuring individual water quality parameters in-situ as the robot traverses its path. The benefit of this direct measurement strategy is that it allows multiple water features to be measured easily across various locations in the water body without the need for additional human interaction. Alternatively, the second method consists of capturing physical water samples using containers on the robot so that these samples can be brought to shore and later analyzed in a laboratory to determine their relevant properties and characteristics. As this strategy requires much additional human labor in the form of laboratory analysis and results in measurements being taken with the sample well removed from its original environment, it is not well suited for the assessment of parameters such as temperature and conductivity that are more easily and accurately evaluated with local sensors. However, the technique is advantageous for the analysis of qualities such as specific dissolved nutrient compositions that cannot be readily determined using inexpensive, mobile sensors.

Considering the sensor equipment method, there are a number of sensors whose data measurement would be valuable in performing useful water quality monitoring. These sensors can be compared in terms of the quantities they measure, the diverse technologies used to accomplish these measurements, their integration capabilities (i.e. the level of additional processing they require for accurate sensing), and their average cost ranges. Table 1 below summarizes this comparison for a set of sensors that are commonly utilized in water quality assessments. Examination of the table shows that all of the common water quality analysis sensors can be obtained for relatively cost-effective prices, with the exception of the dissolved oxygen sensor which costs at least $169. Additionally, the table shows that all of the sensor types contain sensors with demonstrated Arduino microcontroller integrations. Thus, our research demonstrates that these different sensor types can be well-integrated on a single microcontroller system.

Figure 4: Dissolved Oxygen (O2) Sensor

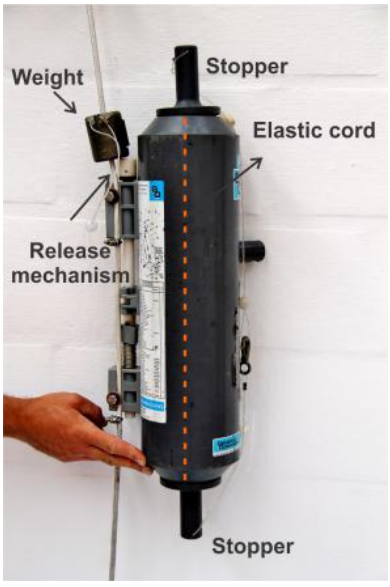

Figure 3: Example of a Niskin bottle

Sensor Type	Measurement Quantity	Measurement Technology	Integration Capabilities	Average Cost Range
Temperature	Temperature (°C)	RTD, thermistors, semiconductor	- Range of sensors with sample Arduino integrations	$6.00 - $56.00
Pressure (Depth)	Pressure (mbar)	Microelectromechanical systems capacitive	- Some sensors with sample Arduino integrations - Potential for dual pressure/temperature sensor	$55.00 - $68.00
Turbidity	NTU	Nephelometers, Absorption turbidity meters	- Only one suitable option in price range - Contains sample Arduino integration	$10.00 - $1000.00+
Conductivity (Salinity)	Conductivity (ms/cm)	Analog Electrical Conductivity meter	- Some sensors with sample Arduino integrations	$70.00 - $200.00+
pH	pH	Glass pH electrode	- Some sensors with sample Arduino integrations	$30.00 - $100.00+
O_2	Dissolved Oxygen (mg/l)	Chemiluminescent probe	- Some sensors with sample Arduino integrations	$169.00 - $250.00+

Table 1: Comparison of Common Water Quality Analysis Sensors

To capture physical water samples, there are a few existing technologies that could have been adapted for use in our project. The most popular method for capturing water samples at specified depths is through the use of Niskin bottles. These bottles each contain two open valves (one on each end of the bottle) that allow water to pass into the bottle as it sinks to the desired water depth. Once the bottle reaches the desired depth, a person uses a physical trigger (in the form of a weight) to shut the bottle's valves and contain a water sample from that depth within the bottle. [15] Multiple Niskin bottles are usually attached to the CTD systems described previously so that individual water samples can be collected at different depths. Aside from Niskin bottles or similar manual valve oriented designs, water pump designs have also been utilized for sample capturing. These systems feature either manual or electronically driven water pumps that are submerged to a desired depth and then used to pump water to a sample collection container that is located elsewhere. The collection container is usually located at the surface of the water, but can sometimes be contained in a subsurface collection chamber as well. These systems work well for the collection of samples intended to be analyzed for simple water characteristics such

as pH, but cannot be used for dissolved oxygen analysis (as the pump introduces oxygen into the sample) or bacterial analysis (as the pump is not sterile and will kill the bacteria) [16]. Finally, electromechanical fluid systems can be implemented for more autonomous control of water sample collection. These fluid systems are recently becoming more complex and featuring biomass filters that filter various organisms and materials from the water for in situ scientific analysis within an AUV [17]. However, they can more simply be used for implementing an electromechanical valve that allows water to fill a container when an electrical signal is sent to the valve. Thus, these electromechanical valves can be used in Niskin bottle-like attachments for AUVs so that the AUV can autonomously close the valves when a desired water depth (or any desired water location) is reached.

C. Control System

In order to run various surveying missions in either still bodies of water like lakes, or more moving bodies of water like oceans, we needed to consider how we would control the water robot both manually and autonomously. A great option to accomplish this task is the PX4 Autopilot Pro [18] (more commonly known as the Pixhawk controller). It is an autonomous flight controller that is used in all types of drones, from flying drones, to land rovers, to submersibles. Since our water drone is intended to travel at the surface level, the Pixhawk should be easily integratable into our design since it is already commonly used for both AUVs and ASVs.

Figure 5: Pixhawk Controller

In terms of communicating with the controller, we can interface directly with the Pixhawk controller, or use a companion computer such as a Raspberry Pi to communicate with the PX4 flight stack via the MAVlink protocol [18].

Figure 6: QGroundControl Application Example

So far, all hardware mentioned will be used on the water robot itself. However, we also needed a Ground Control Station (GCS) to monitor the drone and program autonomous missions. We had three options for this area of the system. The first was a GCS known as QgroundControl. QgroundControl was developed by Dronecode, which creates the PX4 flight stack and the MAVlink communication protocol [19]. It is used to install firmware, configure frames, and plan missions. It also has development features so it can be extended to fit the use of another project. This is one path we could have taken so that we could take advantage of QGroundControl's base features while extending them for our uses.

The second option was a GCS called MAVProxy [20]. It is a simpler application that has primarily command line interfacing, with modules for mapping and mission editing. One positive aspect of MAVProxy is that it is written in Python and is therefore easily extensible.

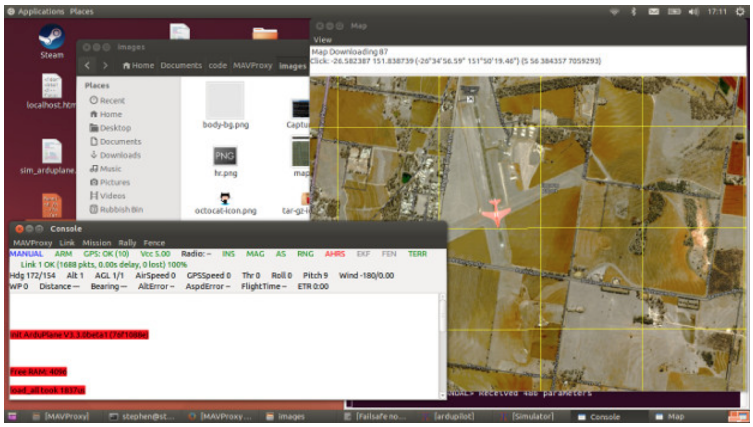

Figure 7: MAVProxy Application Example

The final option was the GCS known as Mission Planner [21]. Mission Planner is one of the most well-known, open-source ground control station softwares. It is the official GCS supported by ArduPilot and explained in all of their tutorial documentation. Aside from mission planning, Mission Planner supports firmware writing, easily accessible parameter writing and testing, the ability to plan a mission and save the planned waypoints to a file, and monitor mission measurements in real-time. Thus, Mission Planner was one of the best GCS options available to us. Its main downside is that it is a very large application, and is thus difficult to perform further development upon.

D. Data Storage

When the robot is in operation, there are two potential choices for extracting the data from the robot. When the robot is gathering information via its sensors, it can either continuously send data while capturing it or store the data locally until the end of the current mission when an operator can manually extract the information from the robot. The main advantage of sending data continuously is that the data can be viewed and process in real-time as the mission occurs. Thus, users coordinating the mission can make live mission updates and analyses as the robot continues to capture data. Also, if the robot cannot make it back safely, the data is not lost. However, this design means that the robot cannot leave the range of the GCS, which may limit its range of data gathering significantly. On the other hand, if the operator manually collects the data at the end of the mission, much less strain would be placed on the robot's power supply. Additionally, the robot will have a much wider range of operation and the captured data will be much less susceptible to data corruption errors. The negative effect of this design choice, though, is that leaving data transfer until the end of the mission will prevent any real-time viewing and analysis of the data while the mission is being performed.

E. Data Analysis

The sensor data the robot collected would ultimately be processed to notice patterns and draw scientific conclusions. To support this necessary data analysis, we needed to build a relevant data analysis software tool. In building this tool, we considered programming languages with large amounts of documentation, with 1st party or 3rd party support, and that would integrate nicely

with all the other sections of the project. Three main candidates that stood out were Java, Python, and C#. Java has the advantages of extensive documentation, a wide variety of support, and relative ease of use. Additionally, it is commonly used for robotics applications, as Java code can be exported to a variety of platforms. C#, on the other hand, is comparatively more difficult to program in. However, C#'s speed is unrivaled and the language itself is the basis of the Mission Planner GCS. Thus, a C# data analysis program could be easily integrated into the Mission Planner GCS if that GCS was chosen for the project. Python is a newer language that has seen recent support within a variety of fields. Its flexibility comes in the form of ease-of-use and the huge variety of 1st and 3rd party modules that we can import and immediately use with it. Additionally, it comparatively has the most support for data analysis and data visualization.

F. System Architecture and Communication

Some wireless communication system needed to be in place in order for the surveying robot and the GCS to communicate. This was important so that consistent status updates could be sent from the robot to the GCS on the location, progress, and internal status of the robot. This communication system would also be important if the user of the robot needs to call the robot back to its start position in order to retrieve it before the allotted survey time has elapsed. If the communication system implemented is robust and powerful enough, it would also be possible to transmit some or all of the sensor data that the robot collects back to the GCS in real-time. All of these concerns require a reliable wireless communication system that can send information across tested bodies of water. With these requirements in mind, there are a few options for wireless communication systems.

Building off of the Pixhawk controller described previously, there are radio communication platforms which can directly interface with the Pixhawk controller. One of these, the Holybro Radio V3 [22], has a maximum output power of 100 mW, resulting in ranges of ~300m. The FPV Radio Telemetry Set [23] is another option with higher output power of 500 mW, and thus a longer reported range of ~1 mile. The Readytosky 3DR 500 mW Radio Telemetry Kit [24] serves as a middle ground between the two aforementioned radio systems, as it is more cost effective than the FPV Radio Telemetry Set and exhibits a range of 1-2 km. All of these radios

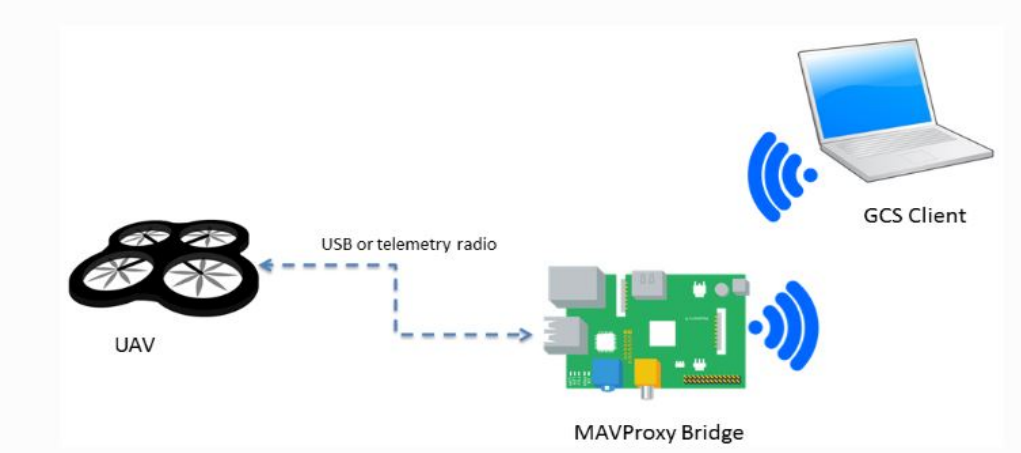

Figure 8: An overview of how the aquatic robotic platform could communicate with the GCS, with the only difference being that the Raspberry Pi depicted here would be included on the robot *[20]*

are available in 915 MHz configurations, which is within the ISM band (industrial, scientific and medical band). This is a frequency allocation that is open for unlicensed radio users in North and South America, and therefore it would be perfect for this application. Furthermore, the MAVlink protocol utilized by the Pixhawk controller for wireless communication is supported by these three radio systems. Of course, using any of these radio platforms would necessitate some peripherals on the GCS side to ensure that the radio could be connected to the ground control computer.

Another possibility was to use a Raspberry Pi on the robot to set up a WiFi network that the GCS could connect to. Examples of this implementation have been shown in guides detailing how to configure and use the Pixhawk controller. The advantages of communicating over WiFi are higher data speeds, and the elimination of a peripheral radio that needs to be connected to the ground control computer. There were concerns of the ranges that could be achieved using the WiFi antenna built into the Raspberry Pi, though, so this question was something that needed to be considered and tested. Figure 8 shows a basic overview of how such a system would operate.

V. Design Strategies and Approach

A. Physical Assembly

While the challenges and difficulties of an almost completely submerged device initially sounded intriguing, our team opted to develop a system that combined the benefits of both surface vehicle collection and underwater vehicle collection. Thus, as unmanned surface collection robots are typically referred to as unmanned surface vehicles (USVs) and unmanned underwater vehicles are typically referred to as autonomous underwater vehicles (AUVs), when their motion is autonomously controlled, we decided to build a dual USV/AUV system as shown in Figure 9. The USV portion of the design handled autonomous navigation to specified GPS positions, while also maintaining a constant communication link with the ground control station. The AUV portion of the design contained a set of sensors for water quality measurements, and also contained a buoyancy control system for diving to different depths to take measurements at these depths. A tether connected the USV to the AUV for towing, communication, and power purposes. Importantly, the AUV subsystem was capable of capturing horizontal water quality profiles, which is something a traditional CTD-based measurement system cannot accomplish.

The problem of autonomously navigating a 3D space would have required great deal more time than this 2D plane this solution offered, especially since some of the team already had experience with autonomous land vehicles (which operate on a similar planar system). The materials of our main USV consisted of non-corrosive aluminum framing, plastic flotation devices, and plastic electronics housings. The hull was styled after that of a pontoon boat, which was a non-complex design that provided a wide flat area for housing all of our electronics. The materials of the AUV were primarily PVC pipe and PVC pipe fittings. The use of PVC pipe provided the submersible with an inherent waterproofing design, as PVC cementing of joints and connections made these connections perfectly waterproof. Special PVC fittings also allowed sensors connected to the submersible to be modular, as some PVC fittings are designed to be easily removed and replaced. Finally, the use of PVC piping allowed the AUV to be naturally hydrodynamic through the thin and elongated profile of the pipe. Wings were added to the submersible for increased stability.

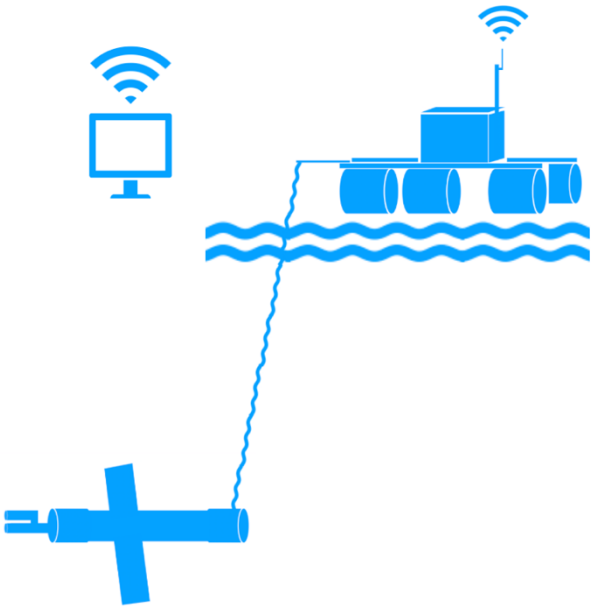

Figure 9: Diagram of Dual USV/AUV Inspired Setup

B. Data Collection

To capture the data necessary for pertinent water quality analysis, we decided to implement a deployable data collection module (our AUV) that extends from our main USV system to configurable depths of water (as explained in the previous Physical Assembly section). The AUV

submersible was designed to have multiple water quality sensors attached to it, so that these sensors could take measurements in regular intervals as the submersible navigated to different depths (via its buoyancy control system) and coordinates (through it being towed by the USV). The sensor system connected to the submersible was fully modularized through the use of custom PVC fitting housings for any sensors connected to the submersible. These PVC fitting housings could be screwed onto and off of the main submersible frame, thus allowing a variety of sensor types to be used simply by encasing any desired sensor types in a new PVC fitting housing.

Based on the sensor type analysis provided in Table 1 previously, we decided that it was feasible (both in terms of cost and of integration capabilities) and desirable to implement all sensor types listed in our data collection system. Thus, we developed our baseline construction of the submersible around being capable of supporting six sensors: a depth sensor, temperature sensor, conductivity sensor, pH sensor, dissolved oxygen sensor, and turbidity sensor. We also designed the amount of computation power we would need around this same number of sensors. Thus, we determined that we could support computations to meet the communication and signal processing requirements specified by the six identified sensors through the use of a single Arduino Uno Rev 3. In our final design, however, two Arduino Uno devices were used. The second Arduino was used to operate the submersible's buoyancy control system. As this control system was dependent upon the submersible's current depth, the depth sensor was attached to and processed by this second Arduino. The first Arduino was used to interface with all remaining sensors, as well as to keep a communication link with the USV. Thus, the second Arduino device interacted with the depth sensor to take depth measurements, used these depth measurements in the buoyancy control system, and sent the depth measurements it recorded to the first Arduino when necessary. The first Arduino, as a counterpart, interacted with the remaining sensors to record their measurements, requested a depth measurement from the second Arduino during every measurement cycle, and then sent the full set of data measurements to the USV at the end of every measurement cycle.

C. Control System

We decided to use the PX4 Pixhawk flight controller as our main source of autonomous control for the designed USV. The Pixhawk ran a modified version of the APMRover2 firmware, which is a version of the Ardupilot base drone firmware that is used for land vehicles and boats. The firmware was modified to support a custom message type intended to hold sensor data so that data could be sent from the Pixhawk to the GCS in real-time during missions. This firmware also uses a skid-steering algorithm that features left and right controls. The left control was connected to the motors on the left side of our USV while the right control was connected to the motors on the right side. The skid-steering algorithm would allow the USV to make both gradual and tight, pivot-like turns when running a mission (depending on the situation).

To relay missions to the USV, monitor these missions, and allow sensor data to flow back to land, we used the Mission Planner ground control software. However, the Mission Planner software we used was first modified to accept our custom MAVLink message type for relaying sensor data on its network. We chose Mission Planner because it is a robust application that

allowed us to create the features we wanted while utilizing its vast library of mapping and configuration features.

Autonomous control was provided by a pre-built PID controller running on the Pixhawk. Users would define an autonomous path for the USV to follow by selecting a series of GPS waypoints (sequential GPS coordinates) in the Mission Planner software to be a "mission". As the Pixhawk controller contained an attached GPS receiver and a built-in inertial measurement unit (IMU), it was able to utilize its current global coordinates, orientation, and velocity in its PID controller to determine what direction to move in and how fast to move to successfully navigate to the next waypoint. This high-level understanding of orientation and velocity would then be translated into a series of motor actuations (following the aforementioned skid-steering algorithm). The Pixhawk continued to direct the motion of the USV from waypoint to waypoint until the mission was completed.

In addition to the Pixhawk, we also chose to include a Raspberry Pi for computation on the USV. The Raspberry Pi served as a bridge between the sensor-interfacing Arduino in the AUV and the Pixhawk controller. Thus, the Raspberry Pi would receive sensor data from the AUV Arduino, package this data into a custom MAVLink message (using our defined MAVLink message type), and then send this message to the Pixhawk controller.

D. Data Storage

To meet the goal of being able to perform real-time data processing while not sacrificing data integrity, we chose to both store data locally on the robot and send it wirelessly to the ground control station.

Upon compiling all of the relevant sensor measurements for a given measurement cycle, the sensor-interfacing Arduino in the AUV would then send the sensor measurements for that trial to the Raspberry Pi on the USV. Once receiving the sensor data, the Raspberry Pi immediately saved the data to a CSV file on its local microSD card. Thus, this data storage source served to protect the integrity of the measurements collected, as it was a local storage method with a very low probability of data corruption or missed data messages.

After storing the data locally, the Raspberry Pi would then package the data into a message to be sent to the Pixhawk. The Pixhawk then transmitted this data message over the wireless radio link to the Mission Planner GCS. Here, the GCS displayed the data in real-time and also saved it to a CSV file on the ground control station computer. In this manner, each mission would produce two data CSV files: one on the Raspberry Pi and one on the ground control station computer. This redundancy helped to ensure data was never lost or mishandled. Additionally, by attempting to send all of the data to the GCS as it was recorded, we were able to perform real-time processing of the data on the GCS. As we decided to only capture data at a rate of 1 Hz, we felt confident that we could send all of the data to the GCS without causing excessive strain on the radio communication channel.

E. Data Analysis

As we decided to use the open-source Mission Planner software for our ground control station, it became a natural choice for us to also extend the Mission Planner software to perform our data analysis tasks. As such, we resolved to create a new "Data Visualization" tab within Mission Planner for performing in-mission analysis and visualization of the captured water quality measurements. As Mission Planner is a C# application, this Data Visualization tab was also written fully in C#.

F. System Architecture and Communication

To explain our full system architecture and the communication between the different subsystems, we start with an explanation of the ground control station and then work our way down the communication line.

Mission Planner was our chosen ground control station. It was retrofitted with a custom MAVLink message so that it had the functionality to receive and process sensor data from our AUV. This application was run from a laptop computer, which had a radio link running on the 915 MHz ISM band (implemented with the Readytosky 3DR 500 mW Radio Telemetry Kit) to communicate with our USV. Mission Planner tracked and recorded telemetry data from the Pixhawk over the radio link, while also sending down planned missions to be stored.

The USV was comprised of the Pixhawk flight controller and a Raspberry Pi 3. The Pixhawk was connected to the other end of the radio link, which is plugged into its Telem1 port. We edited the APMRover2 firmware to allow our custom water quality data message to be recognized by the Pixhawk and forwarded onto the GCS. A GPS module was also attached to the Pixhawk so that it could localize its position and attempt to navigate to GPS waypoints throughout a mission. During missions, the Pixhawk would output signals to the left or right side motors of the USV, allowing for the skid-steering algorithm to function properly. Attached to the Telem2 port was a custom connection to our Raspberry Pi. The Raspberry Pi acted as a communication bridge between the USV and the AUV. The USV ran a MAVLink network while the AUV communicated via a serial protocol, so the two had to integrate at single point. The Raspberry Pi ran a Python script that used the Dronekit library. This library allowed us to interface with the Pixhawk from the Raspberry Pi and generate our custom message. When the AUV transmitted data to the USV, it would be received on the Raspberry Pi's serial port, encoded and packaged into our custom message, and sent along the MAVLink network to be processed and displayed by Mission Planner.

The AUV comprised of two Arduinos communicating over a serial connection and with a series of sensors to collect data. The Arduino-to-Arduino communication was arranged in a hierarchy comprised of the sensor-interfacing Arduino on top and the buoyancy control Arduino underneath it. The sensor-interfacing Arduino held the pH sensor, conductivity sensor, turbidity sensor, and dissolved oxygen sensor. The buoyancy control Arduino held the depth/temperature sensor because it needed to take depth measurements more regularly to maintain buoyancy control. Once a second (1 Hz), the sensor-interfacing Arduino would interrupt the loop being run on the buoyancy control Arduino so it could get the current depth and temperature measurements. Then, it would add in its own pH, conductivity, turbidity, and dissolved oxygen

readings. These readings were sent up the communication line along the tether to be processed by the Raspberry Pi and sent further along to Mission Planner.

95% of communication was in the form of sensor data, which started in the AUV, went up through the USV, and was finally processed by Mission Planner. 5% of the time, a desired depth had to be sent down to the USV's buoyancy control Arduino to dictate the depth the AUV should attempt to maintain. To perform this communication, we used the Pixhawk's "home altitude" parameter because it was unused by the APMRover2 firmware (i.e. there is no need for an altitude parameter in a rover that cannot leave the ground). When the home altitude was changed in Mission Planner (and thus also on the Pixhawk after writing new mission parameters), a loop within the Raspberry Pi would be tripped. This allowed it to record the new desired depth, send it along to the sensor-interfacing Arduino on the USV, and then sit in a waiting state for message reception confirmation from the sensor-interfacing Arduino. Once the sensor-interfacing Arduino received this depth, it would record it, send confirmation back to the Raspberry Pi, send it along to the buoyancy control Arduino via serial communication, and then wait for confirmation. The final step would be for the buoyancy control Arduino to receive the desired depth, record it, and send back confirmation to the sensor-interfacing Arduino. This last step would kick all of the subsystems out of their waiting states and allow them to continue with normal data acquisition. This architecture and communication system proved robust enough for us to run missions in windy conditions and still complete planned missions while recording sufficient sensor data at various depths.

G. System Operational Block Diagram

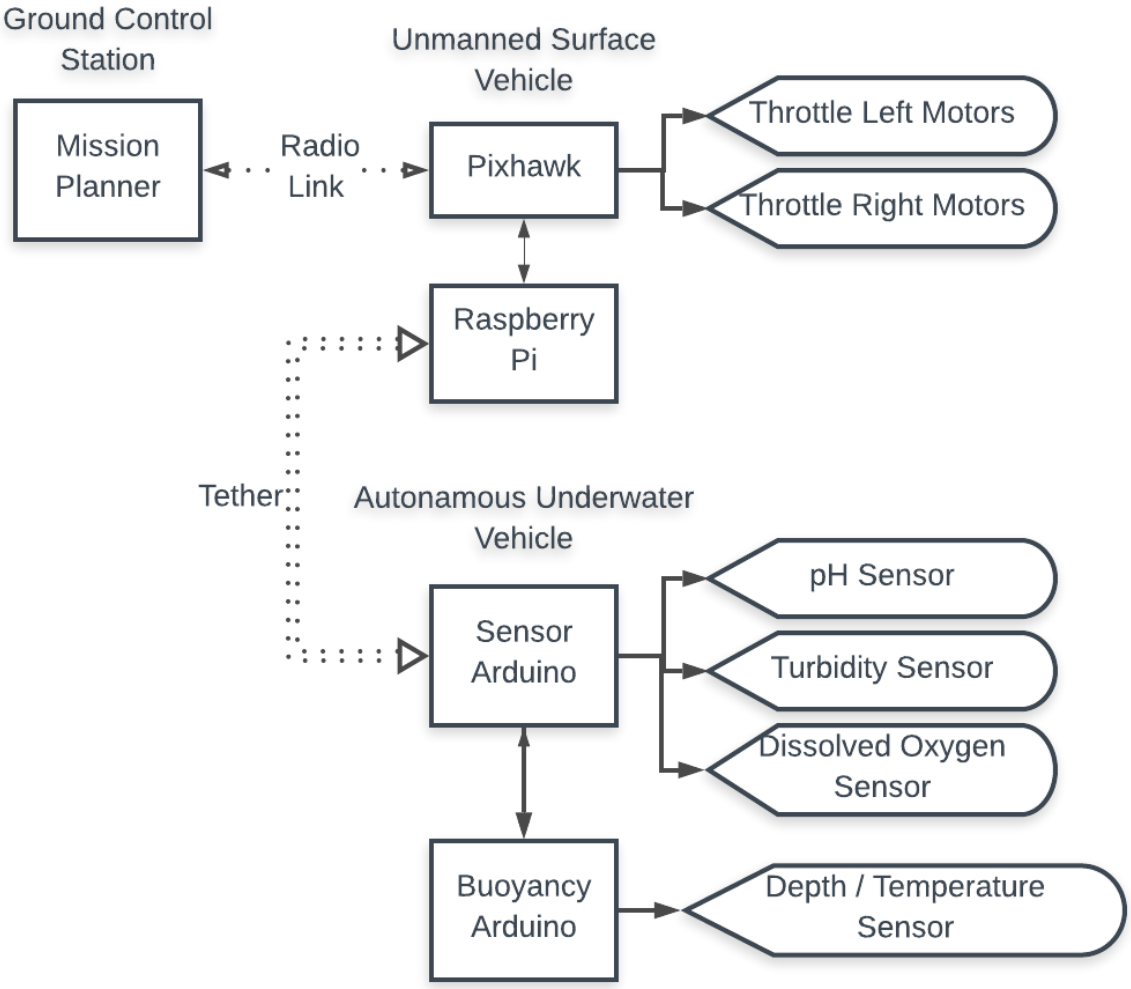

Figure 10: Block Diagram Overview of the Robotics Platform and the GCS to Interface with the Robot

VI. Parts and Implementation

A. Physical Assembly

i. Unmanned Surface Vehicle (USV)

The unmanned surface vehicle was ultimately created to meet the design presented in Figure 9. The pontoon boat structure was constructed from T-slot aluminum structural frame, which provided a great deal of adaptability in component placement as well as sufficient structural support for our vessel. 5 gallon buckets sealed with marine epoxy were utilized to enable flotation of the USV. These were secured to the aluminum frame using large hose clamps. Two electronic casings were used, one larger sealed container and a smaller one mounted atop the first. The smaller housing contained the Pixhawk flight controller, the Raspberry Pi, and the required 5V power supplies for these devices. Inside the main housing, was a number of electronics including an Arduino, 2 motor drivers, 2 relay boards for switching the required forward and reverse PWM signals, and 2 sealed lead acid batteries. These drivers and batteries provided the power for our motors. The motors, mounted to the frame below the level of the buckets, were made from modified 500 GPH bilge pumps outfitted with RC boat propellers to allow them to generate forward thrust. These were arranged with 2 motors in the rear left, 2 in the rear right, 1 in the middle left, and 1 in the middle right. This configuration was to allow for easy integration with the skid steering control algorithm that was implemented by the Pixhawk. In the skid steering method, all the motors on one side of the craft are ganged together. To move forward or backward both sides generate forward or backward thrust together. But to make a turn, one side can produce forward thrust and the other produces backwards thrust. This method achieves very sharp and quick turns. A radio antenna was also mounted to the top of the USV to communicate back to the ground control station. The frame also had a rear extensions to allow the tether to be mounted to the frame far enough to prevent any entanglement with the propellers. The completed USV can be seen in Figure 11.

Figure 11: Completed USV

ii. Autonomous Underwater Vehicle (AUV)

An open source underwater glider was used as the design inspiration for our AUV [25]. This design, seen in Figure 13, offers a much more hydrodynamic shell than the initially proposed system. The shell is constructed from 4” PVC piping in addition to several PVC fittings and caps to allow for sensors to penetrate the hull. The individual sensors have been epoxied to threaded

end caps, so that they may be swapped out depending upon the desired data to be collected. Two aluminum wings are affixed to the side of the submersible using custom 3D printed mounts. These wings help to stabilize the submersible as it is gliding, either above or below the water, as it is pulled behind the USV. An off-the-shelf 4” test cap, which can be seen in Figure 12 below, was used on the front of the submersible to provide a waterproof housing that still allowed for internal access. Over the top of this was placed an additional end cap with two locks to secure it in place and provide clamping pressure on an O-ring placed between the endcap and the rest of the shell.

Figure 12: Test Cap for Submersible Nose

The internals of the AUV can be split into two subsystems: the buoyancy engine and the sensor system. The buoyancy engine which adjusts the buoyancy of the submersible by taking in or expelling water. Three 150 mL syringes are used to contain this water, and have access to the exterior of the submersible by rubber tubing and fittings. A lead screw and fixed nut transfers the rotational output of a NEMA 23 stepper motor into lateral movement of the plungers of the syringes to actually move the water. This system is controlled using an Arduino. The sensor subsystem consists of a bank of 5 sensors including pH, dissolved oxygen, water temperature, turbidity, and depth. As stated above, these sensors are epoxied to threaded PVC end caps to allow for removal or exchanging of sensors. These sensors then have break-out boards, to translate the sensor readings to an analog value. This analog value is then read in by a second Arduino, which publishes this data to the USV using serial communication along the submersible tether. Both of these subsystems are mounted to custom 3D printed plates, which are connected to each other using threaded rods. These plates were designed to fit perfectly within the 4” PVC tubing. A full display of these internal components can be seen below in Figure 14.

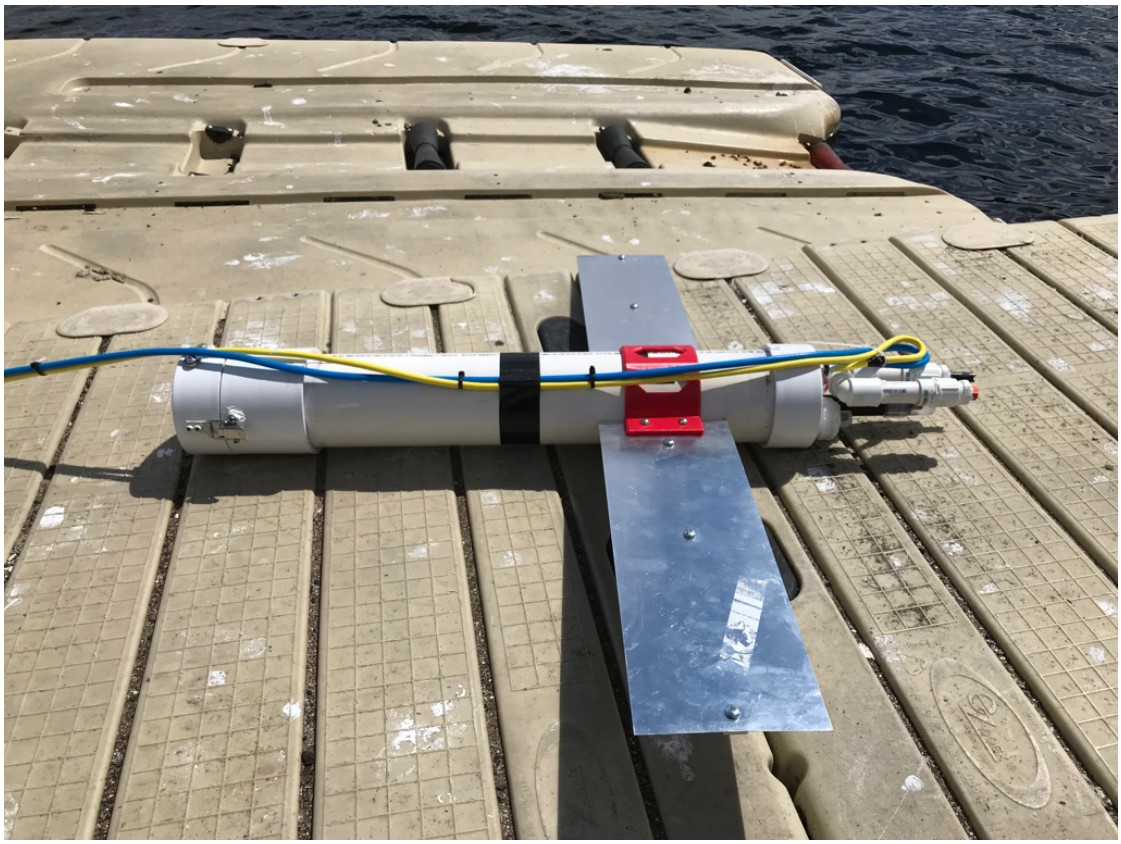

Figure 13: AUV External Design

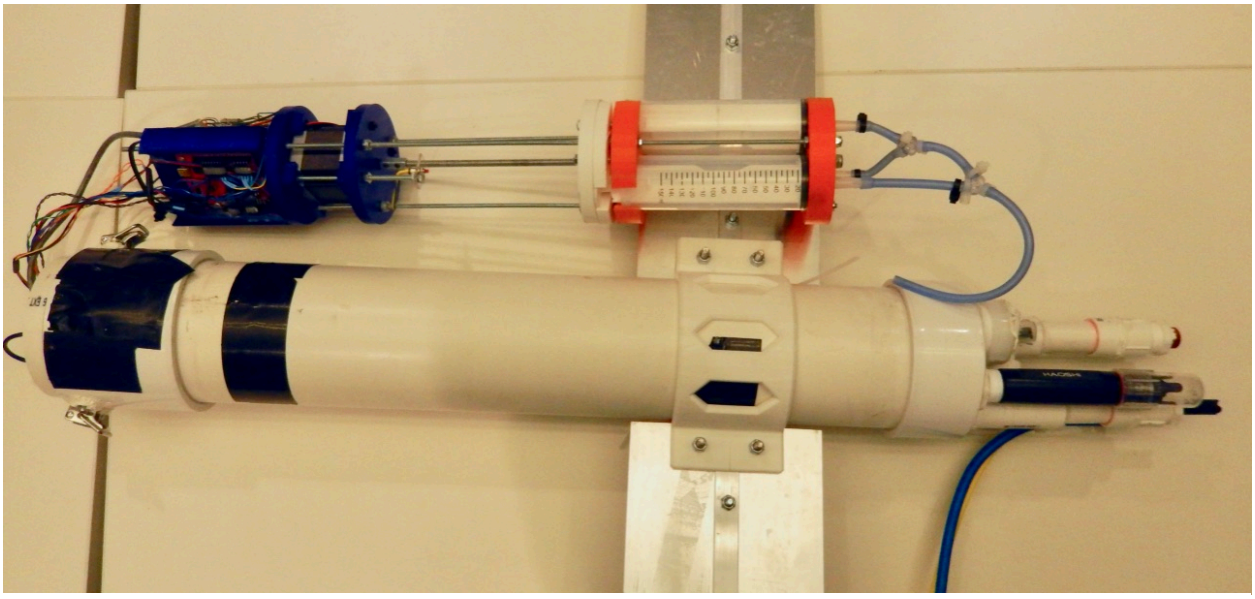

Figure 14: AUV Internal Design

B. Data Collection

To obtain the six water quality monitoring sensors that we needed for our AUV's data capturing system, we relied on two different sensor manufacturers. The first was a company based in Shanghai, China known as DFRobot. DFRobot is a well-known producer of affordable yet quality robotics parts. Amongst their inventory of robotics components, they have a dedicated set of "liquid sensors" that have experienced much attention and development, and thus were also of great interest to us. After reaching out to DFRobot to discuss our project with them, they offered to sponsor our project design by giving us one of their dissolved oxygen sensors (valued at $169) and one of their turbidity sensors (valued at $9.90) for free. Thrilled to accept this sponsorship offer from DFRobot, we also decided to purchase our conductivity sensor and our pH sensor from them. As DFRobot did not supply a depth sensor, we searched elsewhere to find one appropriate for our design. After much investigation, we settled on the Bar30 high-resolution depth sensor developed by Blue Robotics. As this sensor was actually a dual depth and temperature sensor, there was no need to purchase a separate temperature sensor.

To interface with these sensors, the sample Arduino code bases provided by both DFRobot and Blue Robotics were leveraged [26] [27] [28] [29] [30]. Building from these sample code bases that demonstrated basic methods for interacting with the sensors to receive and correct measurements, we incorporated functions for reading from each sensor into two cohesive Arduino scripts. One script would run on the second, buoyancy control system Arduino to control the buoyancy system and take depth and temperature measurements. The second script would run on the first, sensor system Arduino to read measurements from the remaining sensors, compile the measurements into a message structure, and send this message over the tether to the USV.

A large portion of the sensor integration and implementation also consisted of calibrating the sensors properly. While the depth/temperature sensor from Blue Robotics came pre-calibrated and thus did not need any further development, all remaining sensors required some form of calibration. For the dissolved oxygen sensor, a two-point calibration scheme was followed with one point for a 0% dissolved oxygen reading and a second point for a 100% dissolved oxygen reading. The 0% dissolved oxygen solution was created by mixing sodium sulfite (Na_2SO_3) with distilled water until the solution was saturated (the sodium sulfite would react with the water to absorb all of the dissolved oxygen in it). The 100% dissolved oxygen measurement was taken by wetting the sensor probe with distilled water and then exposing it directly to air. To calibrate the conductivity sensor, a single-point calibration scheme was followed using the 12.88mS/cm conductivity calibration solution sold with the sensor. The pH sensor was calibrated using a two-point calibration scheme, with one point being a controlled 7 pH solution and the second point being a controlled 4 pH solution. Finally, to simplify the turbidity sensor calibration procedure, the second order function relating turbidity sensor voltage output to a turbidity measurement (in NTU) provided by DFRobot was simply applied. While this step did not represent a true calibration of the sensor, it still produced acceptable results as the sensor itself was not rated to a high accuracy level.

C. Data Analysis

The Data Visualization tab we designed consisted of four main data analysis components. The first allowed the user to view some basic statistics about a specified incoming measurement quantity. Namely, these basic statistics were the quantity's maximum value, minimum value, and mean value. Next, the tab allowed the user to visualize temporal line charts of specified water quality quantities. Thus, the user would be able to view how different water quality parameters have been changing over time. From here, the user could also visualize a scatterplot of one water quality parameter versus another water quality parameter. Such a visualization was useful for measuring the interdependence of two different quality categories for a given water body. The final option the user was given was the ability to generate a heat map of a particular water quality parameter over the geographical coordinates the parameter was recorded at so far. Here, the user could set inputs for the heat map's individual point radius and color intensity. Thus, this visualization allowed the user to easily see how a particular water quality parameter varied over different geographical locations within a body of water. The user was able to view and update each of these plots in real-time as the mission was conducted. Additionally, the user could generate other desired graphs using the data CSV file written to the GCS and their favorite data analysis tool (such as Microsoft Excel or Python). Figure 15 shows an example of the implemented Data Visualization tab.

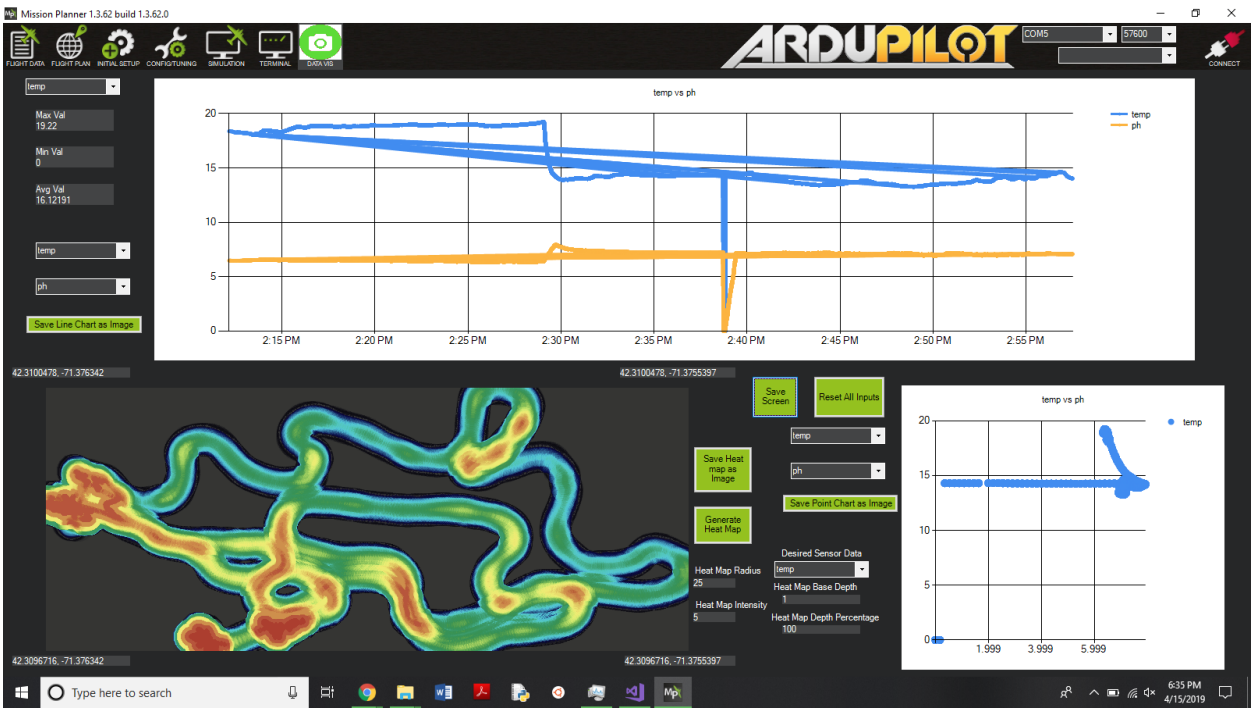

Figure 15: Created Mission Planner Data Visualization Tab

VII. Experimental Results and Analysis

Once our system was fully designed and implemented, we were able to perform some final experiments with the finished product to evaluate its performance and real-world applicability. We conducted our experiments at Cochituate State Park in Natick, Massachusetts as the park contained a deep lake with a boating dock that was perfect for launching our USV and conducting our testing. Our main testing consisted of two trials: one where the AUV was towed on the surface of the water and a second where the AUV was towed while maintaining a depth of approximately two meters below the water's surface. In both trials, sensor measurements of depth, temperature, pH, dissolved oxygen, and turbidity were captured.

Figure 16 shows the waypoint mission set up for the USV in Trial #1 (the surface trial mission) and a portion of the USV's path as it attempted to autonomously navigate to each waypoint. The tab on the bottom left-hand side of the screen also shows the sensor measurements being recorded and displayed in Mission Planner in real-time. Figure 17 shows these same results for the USV in Trial #2 (the two meter depth trial).

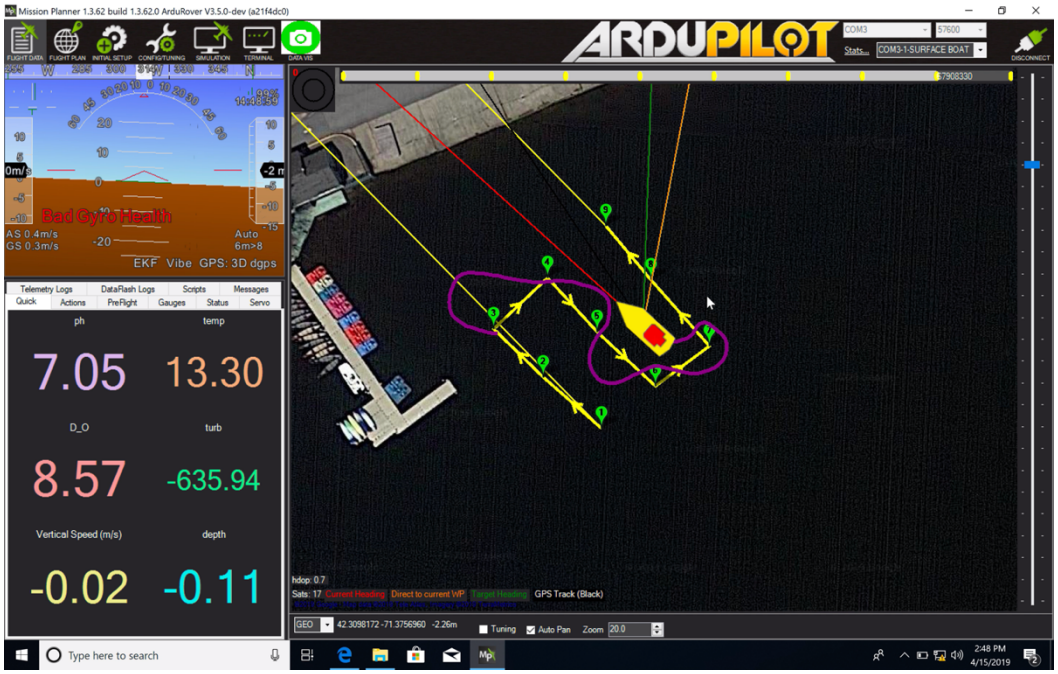

Figure 16: Mission Planner Screen View for Trial #1

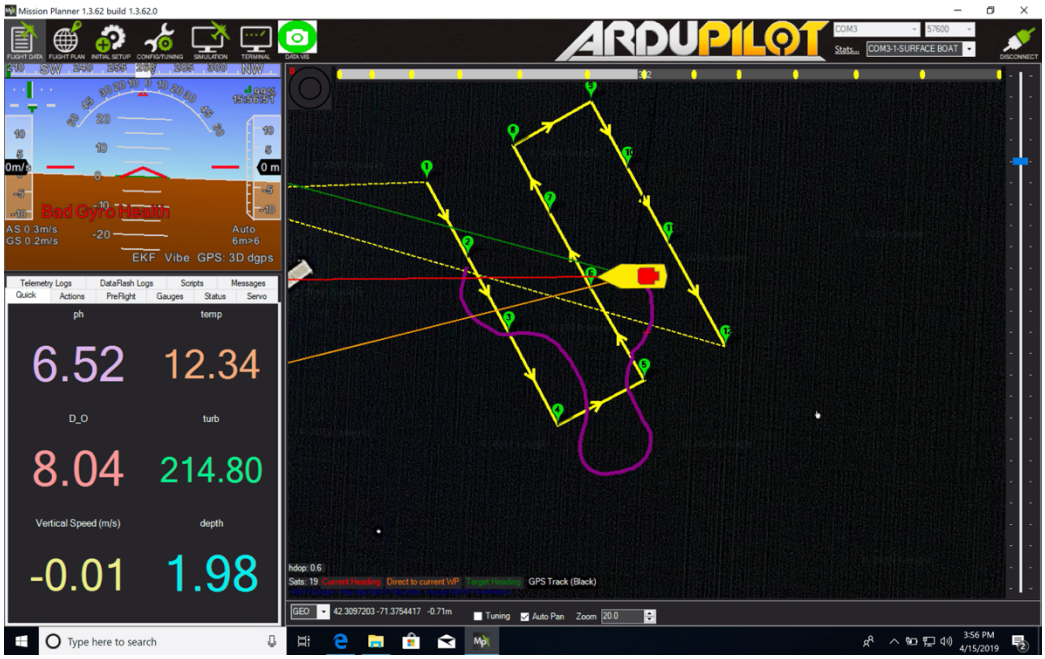

Figure 17: Mission Planner Screen View for Trial #2

In both trials, the USV was successful in autonomously navigating to all mission waypoints. As Figures 17 and 18 show, the USV was not perfect in following the paths between waypoints. Such a result is to be expected, as external factors such as currents in the lake will affect the robot's motion in unforeseeable ways. Further tuning of the USV's autonomous control PID controller could also allow the vehicle to remain closer to the set path during the missions. However, the figures do also display the USV reaching every defined waypoint within a certain (two meter) radius around the defined coordinates. Thus, as we were less concerned with the robot's path in between waypoints and were more concerned with dictating a general region for the robot to survey (in fact, having the robot move slightly off the defined path actually allows a larger region of the lake to be surveyed), we found these navigation results to be very successful and encouraging.

Additionally, both trials succeeded in sending sensor data through the full communication chain, from the Arduinos in the AUV, to the Raspberry Pi on the USV, to the Pixhawk on the USV, to the Mission Planner GCS on shore. Figures 17 and 18 show these sensor measurements being displayed in real-time, and also show that they are generally accurate to expectations (e.g. the pH measurement is around 7 pH as expected in water and the temperature measurement is between 12 and 14 degrees Celcius as expected for a lake in the northeast in April). Thus, we concluded that our data capturing system was successful in functioning as intended both for surface data collection and subsurface data collection.

Finally, we noted that our USV was capable of performing multiple missions with no signs of deterioration in its frame, pontoons, or electronics housings. Additionally, our AUV was perfectly balanced when placed in the water and thus moved without rolling when pulled by the USV. The AUV also did not experience any leaks in Trial #1 and only experienced a small leak in Trial #2. As such, we found that our physical design and assembly of both the USV and the AUV was very robust and successful for performing the necessary missions.

Figures 18 and 19 below show two temporal data plots generated for Trials #1 and #2. Figure 18 shows a plot of temperature v.s sample number for Trials #1 and #2 while Figure 19 shows a plot of pH v.s sample number for Trials #1 and #2. Figure 18 displays that the temperature for Trial #2 is consistently lower than the temperature for Trial #1, which makes sense intuitively as it is well-known that water temperature is colder at deeper depths. Additionally, the figure shows a large drop in temperature at the end of Trial #2. This temperature drop also makes great sense, as at this point in time the USV had completed its mission and was stationary. Without being pulled by the USV anymore, the AUV began to drop in depth until it reached a depth of almost three meters (the length of the tether that we used). Thus, as the AUV's depth dropped further at the end of the mission, the drop in temperature experienced is expected. Figure 19 displays that the pH measurements were very consistent in Trial #1 but were much more cyclic in Trial #2. We believe that this relationship can also be explained by small drops in speed and depth experienced in Trial #2 as the USV made its sharpest turns. While making these turns, the AUV would drop slightly in depth as more slack was present in the tether. Due to the characteristics of the lake, a lower pH value was experienced at deeper depths.

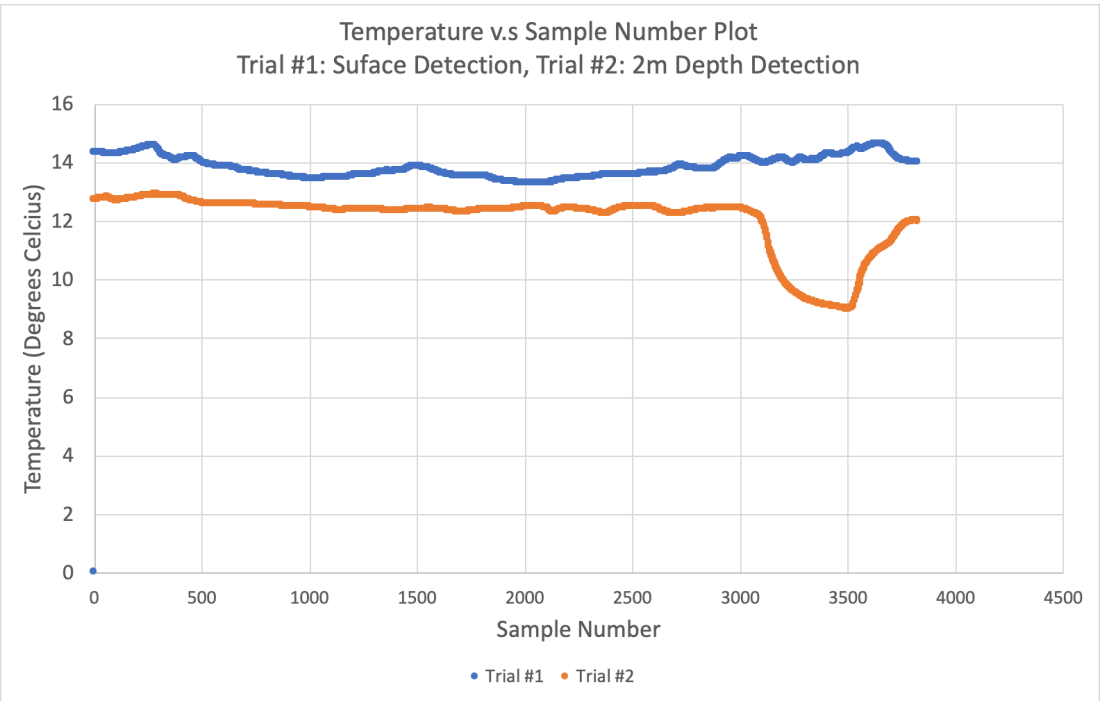

Figure 18: Temperature v.s Sample Number Data Results Plot

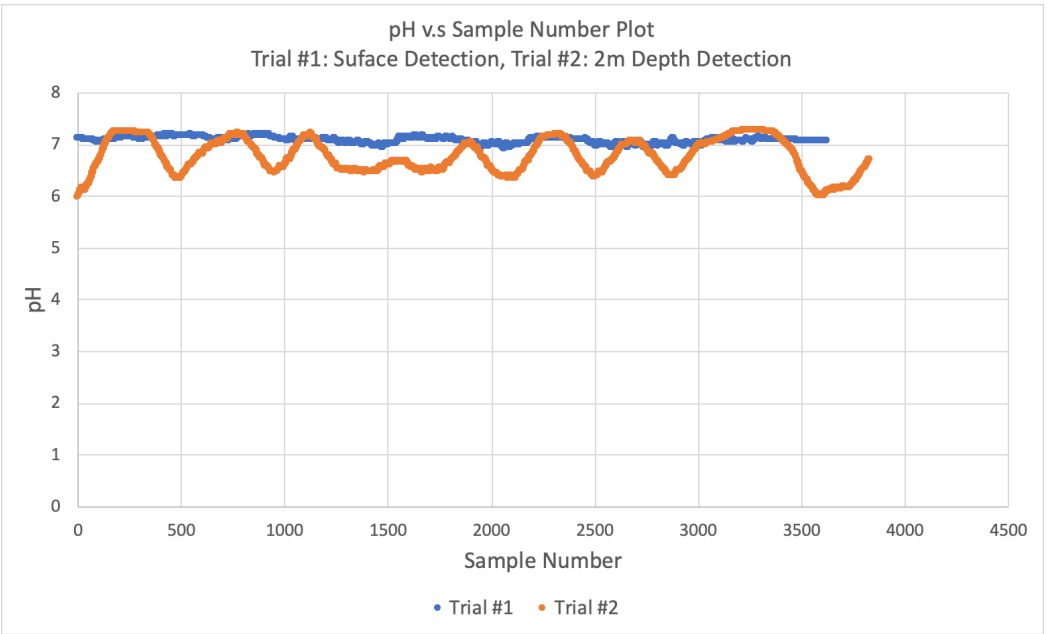

Figure 19: pH v.s Sample Number Data Results Plot

As the trends displayed by Figures 18 and 19 are both explainable and meaningful in the context of analyzing the water quality of the lake we performed our testing at, we concluded that our system was very capable of and well-suited for performing real-world water quality measurements and analyses at multiple water depths.

VIII. Cost Analysis

The following table outlines our final expenditure list for our project. Evidently, we were successfully able to remain under our allotted $1000 budget. Thus, we prove that our design is cost-effective, especially compared to other existing water quality monitoring robots on the market.

Item #	Item Name	Individual Price	Quantity	Total Item Cost
1	USV Physical Construction			
1.01	Home Depot Buckets	$ 3.25	6	$ 19.50
1.02	Home Depot Bucket Lids	$ 1.76	6	$ 10.56
1.03	Aluminum T-Slot Extrusion	0	As needed	0
1.04	Duct Clamping	$ 1.16	10	$ 11.60
1.05	Weatherproof Bin	$ 12.99	1	$ 12.99
1.06	Larger Propellers	$ 13.85	2	$ 27.70
1.07	Marine Sealant (for Buckets)	$ 27.39	1	$ 27.39
1.08	Hose Clamps	$ 0.93	3	$ 2.79
1.09	Spring Link (Carabiner)	$ 6.97	1	$ 6.97
1.1	Velcro	$ 9.47	1	$ 9.47
1.11	Zip Ties	$ 9.22	1	$ 9.22
1.12	Tether Cable Thimble	$ 5.00	1	$ 5.00
2	USV Electronics			
2.01	Motor Driver	$ 13.59	2	$ 27.18
2.02	Raspberry Pi	$ 34.98	1	$ 34.98
2.03	Pixhawk	$ 127.99	1	$ 127.99
2.04	32GB Extreme SD card	$ 22.99	1	$ 22.99
2.05	Radio Module	$ 28.99	1	$ 28.99
2.06	Relays	$ 16.99	1	$ 16.99
2.07	USB to Serial Converter	$ 8.99	1	$ 8.99
2.08	12V SLA Batteries	$ -	2	$ -
3	AUV Physical Construction			
3.01	Syringes	$ 9.49	1	$ 9.49
3.02	3D Print #1	$ 12.00	1	$ 12.00
3.03	3D Print #2	$ 10.00	1	$ 10.00
3.04	Crimping Ring	$ 0.68	6	$ 4.08
3.05	Threaded Rod	$ 1.97	1	$ 1.97
3.06	Through Rod	$ 2.21	3	$ 6.63
3.07	ROV Tether	$ 20.00	1	$ 20.00
3.08	Enclosure Vent and Plug	$ 8.00	1	$ 8.00
3.09	Loctite Marine Epoxy	$ 6.00	1	$ 6.00
3.1	M10 Cable Penetrator for 8mm Cable	$ 5.00	1	$ 5.00

3.11	O-Ring Set 4" Series	$ 3.00	1	$ 3.00
3.12	Threaded Rod	$ 2.94	3	$ 8.82
3.13	Aluminum Bar (Wing Support Bar)	$ 4.28	1	$ 4.28
3.14	Sheet Aluminum (Wings)	$ 8.97	2	$ 17.94
3.15	Plumber's Tape	$ 0.98	1	$ 0.98
3.16	PVC Cement	$ 5.40	1	$ 5.40
3.17	DWV Trap Adapter	$ 1.85	1	$ 1.85
3.18	Various PVC Fittings			$ 72.98
3.19	Loctite Marine Sealant (Flexible)	$ 8.97	1	$ 8.97
4	AUV Electronics			
4.01	Level Shifters	$ 7.49	1	$ 7.49
4.02	Stepper Motor	$ 26.00	1	$ 26.00
4.03	Stepper Motor Driver	$ 25.99	1	$ 25.99
4.04	Pressure/Depth Sensor	$ 34.10	1	$ 34.10
4.05	Conductivity Sensor	$ 79.90	1	$ 79.90
4.06	pH Sensor	$ 56.95	1	$ 56.95
4.07	Bar30 High-Resolution 300m Depth Sensor	$ 68.00	1	$ 68.00
5	Miscellaneous			
5.01	HDMI Cable	$ 21.99	1	$ 21.99
5.02	Silicone Caulk	$ 6.79	1	$ 6.79
5.03	Rope	$ 11.74	1	$ 11.74
5.04	Wing Nuts	$ 1.18	1	$ 1.18
5.05	Machine Screws	$ 1.18	2	$ 2.36
5.06	Jumper Cables	$ 6.98	1	$ 6.98
5.07	Lock Nuts: 1/4 in	$ 5.97	1	$ 5.97
5.08	Screws: 1/4 in	$ 4.44	1	$ 4.44
5.09	Lock Nuts: 8/32	$ 4.57	1	$ 4.57
5.1	Washers 1/4 in	$ 0.12	8	$ 0.96
5.11	Machine Nuts 8/32	$ 3.92	1	$ 3.92
5.12	Lock Washers	$ 0.65	4	$ 2.60
Total $$ Spent				**$ 990.62**

Table 2: Final Project Costs

IX. Conclusion

This report outlines our methodology for constructing an autonomous subsurface data acquisition system. By utilizing autonomous navigation, this project eliminated the need for human work hours to be wasted collecting large amounts of data. The complete project also demonstrated the ability to collect a variety of water quality sensor data at multiple depths while protecting the electronics from flooding. The promising results we received through construction and testing demonstrate that it is possible to minimize time spent retrieving data measurements as well as alleviating current aliasing issues that are found in the field.

X. Appendices

A. List of Figures

B. List of Tables

XI. References

[1] M. Dunbabin and L. Marques, "Robots for Environmental Monitoring: Significant Advancements and Applications," *IEEE Robotics & Automation Magazine,* pp. 24-39, March 2012.

[2] B. Allen, R. Stokey, T. Austin, N. Forrester, R. Goldsborough, M. Purcell and C. von Alt, "REMUS: A small, low cost AUV; System Description, Field Trials and Performance Results," in *OCEANS '97. MTS/IEEE Conference Proceedings*, Halifax, 1997.

[3] G. Griffiths, N. W. Millard, S. D. McPhail, P. Stevenson, J. R. Perrett, M. Pebody and A. T. Webb, "Towards environmental monitoring with the Autosub autonomous underwater vehicle," in *Proceedings of 1998 International Symposium on Underwater Technology*, Tokyo, 1998.

[4] J. Gould, D. Roemmich, S. Wijffels, H. Freeland, M. Ignaszewsky, X. Jianping, S. Pouliquen, Y. Desaubies, U. Send, K. Radhakrishnan, K. Takeuchi, K. Kim, M. Danchenkov, P. Sutton, B. King, B. Owens and S. Riser, "Argo Profiling Floats Bring New Era of In Situ Ocean Observations," *Eos,* vol. 85, no. 19, pp. 179, 190–191, 11 May 2004.

[5] E. Edson, "Introducing the MantaRay Microplastic Sampler," MantaRay Microplastic Sampler, 2017. [Online]. Available: https://www.mantaraysampler.com/technology-1. [Accessed 27 June 2018].

[6] J. Manley and S. Willcox, "The Wave Glider: A Persistent Platform for Ocean Science," in *OCEANS 2010 IEEE - Sydney*, Sydney, 2010.

[7] ASV Global, "C-Enduro," ASV Global, 2018. [Online]. Available: https://www.asvglobal.com/product/c-enduro/. [Accessed 27 June 2018].

[8] B. Bayat, A. Crespi and A. Ijspeert, "Envirobot: A Bio-Inspired Environmental Monitoring Platform," Biorobotics laboratory (BioRob) - Ecole polytechnique federale de Lausanne (EPFL), Lausanne, 2016.

[9] D. Rodriguez, M. Franklin, C. Byrne and I. Shockey, "A Study of the Feasibility of Autonomous Surface Vehicles," Worcester Polytechnic Institute, Worcester, 2012.

[10] M. Dunbabin and A. Grinham, "Experimental Evaluation of an Autonomous Surface Vehicle for Water Quality and Greenhouse Gas Emission Monitoring," in *2010 IEEE International Conference on Robotics and Automation*, Anchorage, 2010.

[11] G. Thirunavukkarasu, L. Patrick, B. Champion, L. Chua, V. T. Huynh and M. Joordens, "Design and Development of a Low-Cost Autonomous Surface Vehicle," in *2017 12th System of Systems Engineering Conference (SoSE)*, Waikoloa, 2017.

[12] dcolemans, "DIY Submersible ROV," Autodesk, Inc, 24 May 2018. [Online]. Available: http://www.instructables.com/id/DIY-Submersible-ROV/. [Accessed 27 June 2018].

[13] US Department of Commerce, and National Oceanic and Atmospheric Administration, "NOAA Ocean Explorer: Observing Systems and Sensors: Submersible Collectors," *NOAA Ocean Exploration and Research: Annual Report 2014: Ocean Exploration Benefits NOAA and the Nation,* 5 April 2013.

[14] US Department of Commerce, and National Oceanic and Atmospheric Administration, "NOAA Ocean Explorer: Technology: Observation Tools: Sonde and CTD," *NOAA Ocean*

Exploration and Research: Annual Report 2014: Ocean Exploration Benefits NOAA and the Nation, 30 October 2012.

[15] Flanders Marine Institute, "Niskin bottle," VLIZ, 2018. [Online]. Available: http://www.vliz.be/en/Niskinbottle. [Accessed 27 June 2018].

[16] L. Green, "How to Collect Water Samples," The University of Rhode Island College of the Environment and Life Sciences, 2012.

[17] Monterey Bay Aquarium Research Institute, "The Environmental Sample Processor (ESP)," Monterey Bay Aquarium Research Institute (MBARI), 2017. [Online]. Available: https://www.mbari.org/technology/emerging-current-tools/instruments/environmental-sample-processor-esp/. [Accessed 27 June 2018].

[18] PX4 , "PX4 Autopilot User Guide," PX4, 25 June 2018. [Online]. Available: https://docs.px4.io/en/. [Accessed 27 June 2018].

[19] D. Gagne, "QGroundControl User Guide," QGroundControl , 2018. [Online]. Available: https://docs.qgroundcontrol.com/en/. [Accessed 27 June 2018].

[20] MAVProxy, "MAVProxy 1.6.2 Documentation - Modules," ArduPilot, 2018. [Online]. Available: http://ardupilot.github.io/MAVProxy/html/modules/index.html#. [Accessed 27 June 2018].

[21] ArduPilot Dev Team, "Mission Planner Home," [Online]. Available: http://ardupilot.org/planner/. [Accessed 26 April 2019].

[22] Holybro, "Holybro Radio V3," Holybro, 2018. [Online]. Available: http://www.holybro.com/product/57. [Accessed 27 June 2018].

[23] Holybro, "FPV Radio telemetry set," Holybro, 2018. [Online]. Available: http://www.holybro.com/product/15. [Accessed 27 June 2018].

[24] Readytosky, "Product Description," [Online]. Available: https://www.amazon.com/gp/product/B07B4VX5K9/ref=ask_ql_qh_dp_hza. [Accessed 26 April 2019].

[25] alexw, "OSUG: Open-Source Underwater Glider," [Online]. Available: https://hackaday.io/project/20458-osug-open-source-underwater-glider. [Accessed 26 April 2019].

[26] DFRobot, "Gravity Analog Electrical Conductivity Sensor," [Online]. Available: https://wiki.dfrobot.com/Gravity__Analog_Electrical_Conductivity_Sensor___Meter_V2__K=1__SKU_DFR0300. [Accessed 26 April 2019].

[27] DFRobot, "pH Meter," [Online]. Available: https://wiki.dfrobot.com/PH_meter_SKU__SEN0161_. [Accessed 26 April 2019].

[28] DFRobot, "Gravity Analog Dissolved Oxygen Sensor," [Online]. Available: https://wiki.dfrobot.com/Gravity__Analog_Dissolved_Oxygen_Sensor_SKU_SEN0237. [Accessed 26 April 2019].

[29] DFRobot, "Turbidity Sensor," [Online]. Available: https://wiki.dfrobot.com/Turbidity_sensor_SKU__SEN0189. [Accessed 26 April 2019].

[30] Blue Robotics, "BlueRobotics MS5837 Library," [Online]. Available: https://github.com/bluerobotics/BlueRobotics_MS5837_Library. [Accessed 26 April 2019].

[31] Plotly, "Jupyter Notebook Tutorial in Python," Plotly, 2015. [Online]. Available: https://plot.ly/python/ipython-notebook-tutorial/#introduction. [Accessed 27 June 2018].

[32] John Hunter; Darren Dale; Eric Firing; Michael Droettboom; the Matplotlib development team, Matplotlib, 5 May 2018. [Online]. Available: https://matplotlib.org/. [Accessed 27 June 2018].