Tech 797 Final Report: Remotely Operated Vehicle (ROV)

By: Timothy Holt, Kevin Johnson, Logan Yotnakparian

In collaboration with: Travis Calley (CS), Matthew Lemire (CS)

PI: Dr. May-Wein Thein

Co – PI: Dr. Yuri Rzhanov

Graduate Advisors: Allisa Dalpe, Ozzy Oruc, Alex Cook

Sea Grant
New Hampshire

Abstract

This multi year spanning senior project is one half of a larger project aimed at mapping the ocean floor using an Autonomous Surface Vehicle (ASV) equipped with its own Autonomous Underwater Vehicle (AUV). The AUV part of the project is the focus of this group commonly referred to as the “ROV Team”. The overall goal is to create a system that can autonomously map a section of seafloor. When it finds something of interest, it will deploy an AUV which will get a closer look at the object. This year, the ROV team focused on developing an effective and repeatable testing procedure to facilitate the implementation and development of advanced dynamic control systems into existing robotic platforms. At the start of the academic year the system consisted of many subsystems made up of different sensors, and the ROV itself (*BlueROV2*). These systems were set up and operating together, but in a way that limited potential progress. Immediately, efforts were focused on making these separate components operate as a cohesive, uniform system. Upon designing a testing procedure that is intuitive and convenient, the energy put into research would show immediate returns in autonomous performance, preventing frustration caused by lack of organization. The unforeseen circumstances that followed the outbreak of COVID-19 will have a limited impact on the progress of the ROV team. The following class of engineers will be able to quickly understand the components that encompass ROV operation and immediately execute testing whenever possible.

Table of Contents

List of Figures

List of Tables

Introduction

Unmanned vehicles are very useful in exploration and research because they have the ability to reach locations that are inaccessible to people, and they remove the human element from observing an environment. Only about 35% of the ocean has been explored using modern technology, which greatly impacts our understanding of the world given that two thirds of the Earth is covered in water. A big component of this is the physical limitations of the human body. However, the application of unmanned vehicles in this difficult environment will allow for more in-depth explorations and documentation that previously possible, and will improve man's understanding of the oceans.

A robotic system capable of undersea travel provides a platform onto which scientists can implement various research tools depending on the goals of each mission. Unmanned Underwater Vehicles (UUVs) often incorporate live feed cameras and lighting systems to aid in the collection of visual data. Additionally, these platforms sometimes house sonar systems for seafloor mapping or manipulators for biological and geological sampling. UUVs can be divided into two categories; Remotely Operated Vehicles (ROVs) and Autonomous Underwater Vehicles (AUV). ROVs, as their name suggests, require human control at all times, which is often done from the surface using a joystick or controller station. While the lack of autonomy may be a disadvantage in long missions, the ability for a person to have total control of the vehicle would be advantageous when exploring unknown or unique environments, and would give the user the ability to choose objects of interest to focus on. AUVs have the advantage of not requiring human interaction to complete the desired task, but these missions require more complex control algorithms and introduce the need for path planning algorithms as well.

The problem that UUVs face is that the ocean is far too massive to be explored efficiently by manned machines. If ocean exploration vehicles could perform autonomously, then the rate at which the ocean is explored can increase exponentially, and lead to groundbreaking discoveries faster than ever possibile. This is a difficult task to accomplish, however, as UUVs require complex interdisciplinary systems to handle waypoint selection, deployment and recovery, and power in addition to the navigational abilities required from the UUV. If large scale rapid ocean exploration is to be accomplished, each of these demands will have to be performed autonomously.

ASV/ROV Mission

The primary goal of the ASV/ROV teams is to create a modular multi platform system of two vehicles that can perform an ocean mapping mission. An autonomous surface vehicle (ASV) will be sent to perform a search pattern to map a section of ocean floor with sonar. When the ASV detects an object or location of interest, it will deploy an Autonomous Underwater Vehicle (AUV) to navigate close to the object and capture images of the area.

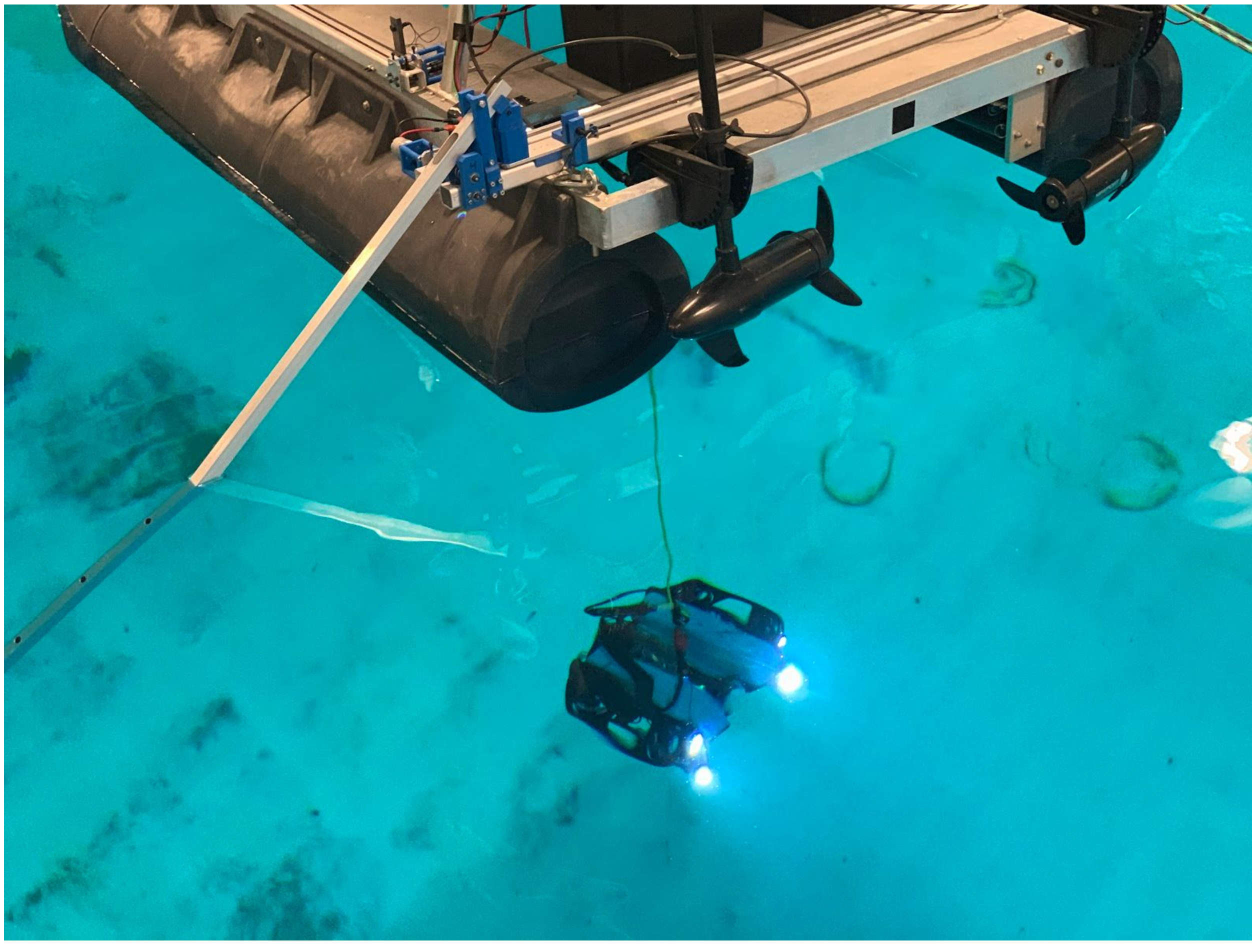


Figure 1: The AUV explores the Chase Test Tank After Being Deployed from the ASV

Objectives

- Improve the current components of the AUV system to cut down on testing time
- Test the accuracy of the AUV’s waypoint navigation
- Optimize the gains of the AUV’s PID controller
- Develop a means of sending waypoints through the ROS network and outside of the QGroundControl User Interface

Approach

In order to perform an autonomous underwater mapping mission, the AUV would have to work in tandem with an Autonomous Surface Vehicle (ASV). The ASV would be responsible for housing, deploying, and retrieving the AUV as well as selecting waypoints to send the ASV to. The ASV in turn would be tasked with being capable to perform waypoint to waypoint navigation, being capable of receiving waypoints from an external system, and being able to perform a data collecting task upon arrival of the destination.

The UUV in use will be a Blue Robotics Heavy2 ROV. This existing platform will be modified into an autonomous system to achieve the requirements mentioned above. Incorporating autonomous packages into this system will allow it to utilize a ROS communication network that features a user interface, and gives it the capabilities to communicate with a coordinate determining system. The location of the AUV will be determined using a Waterlink acoustic locating system known as underwater GPS (UGPS). The modified AUV and the Waterlink system will allow for the AUV to determine its own location as well as receive waypoints to navigate towards. The accuracy of this UGPS system will be improved by incorporating an improved acoustic transmitter on the UGPS.

The performance of the AUV will be measured by using the user interface and physical testing to determine how accurately the AUV can navigate from waypoint to waypoint. The efficiency of this testing can be improved by redesigning the wiring of the various components to utilize quick connections, therefore cutting down on set up time. These quick connections will also improve the ease of switching the AUV components from the testing platform to the full model platform.

In order for this system to be fully functional, it must be able to receive waypoints via the ROS network without the use of the User Interface. This will be accomplished with the development of a Python script that can be utilized to send latitude and longitude coordinates to the AUV in order to provide a waypoint.

The ROV

The platform used for this project is the Blue ROV2 Heavy ROV designed by Blue Robotics. This configuration features 8 Blue Robotics T200 thrusters arranged to allow for full 6 degree of freedom control (Figure 3). The body is made of HDPE plastic and buoyed by 6 R-3318 Urethane Foam blocks. The battery and electronics are kept in seperate electronics tubes composed of acrylic plastic. These tubes are sealed with stainless steel caps lined with o-rings for waterproofing, and the electronics ports are sealed with epoxy.

The electronic hardware of the ROV includes a Raspberry Pi, A Pixhawk, internal sensors, a 1080P live stream camera, and an 18Ah, 15.2 V lithium ion battery. The Pixhawk uses an autopilot software called Ardusub and is responsible for all navigation control as well as controlling the lights and the camera's servo motor. The raspberry Pi performs all computation, integration, and data processing from the camera and the various sensors throughout the robot. Additionally, it is responsible for receiving and processing data provided from the ASV computer including location, orientation, and waypoint. The system diagram for the components of the AUV can be seen in Figure 2.

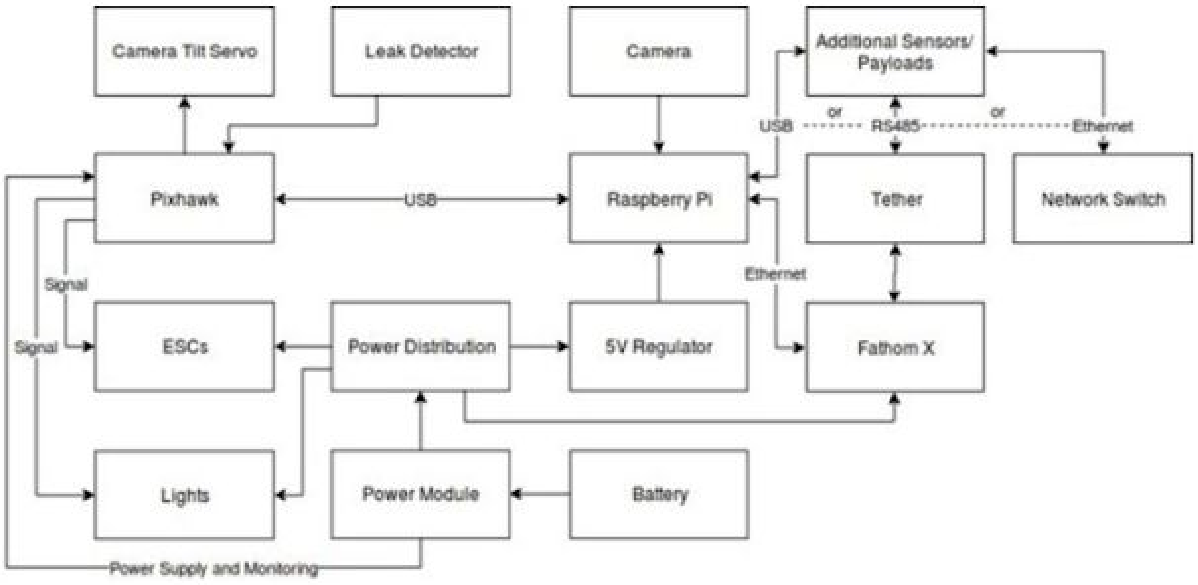


Figure 2: System Diagram for the AUV Components

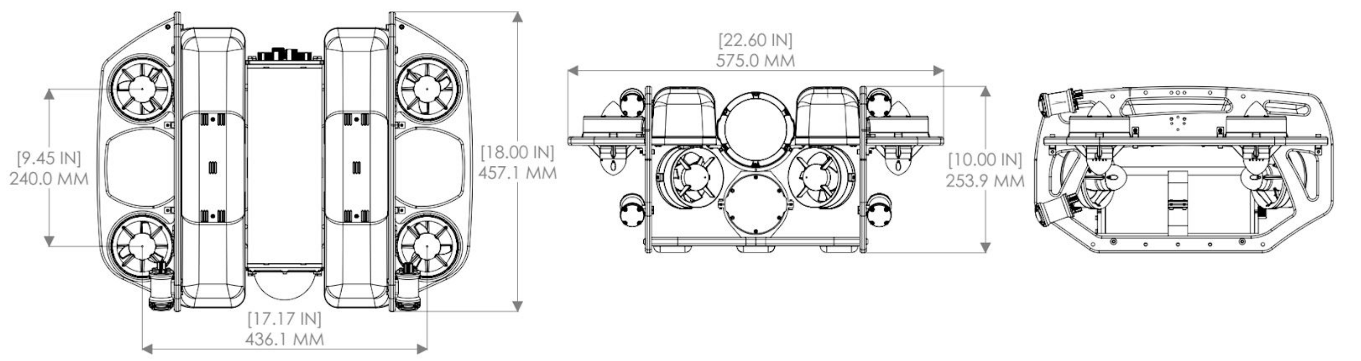


Figure 3: Dimensions of the AUV

Table 1: Blue ROV2 Heavy Specifications [1]

Weight: 26lbs	10” high, 22.6” wide, 18” long
8 thrusters (4 vectored, 4 vertical)	Battery: 18Ah, 15.2V lithium ion
Maximum rated depth: 100m	Maximum forward speed: 3kts
Battery life: ~2 hours	Control board: Pixhawk 1
Tether interface: Fathom-X	Control Software: ArduSub
1080P camera	

The Tether

The tether consisted of 6 bounded pairs of wires encased in a waterproof sheath made of high strength fibers capable of withstanding a tensile strength of 300 pounds[1]. The ROV only requires two individual wires for communication, so the other 10 would allow for the integration of future systems. Originally, the ROV was fitted with a permanent tether connection for remote operation through a computer as shown in Figure 4.The result of this was that if the ROV was being tested externally and needed to be incorporated into a full system test, all 75 meters of the tether would need to be transferred from one system to the other as well. This caused very inefficient testing and wasted significant amounts of time. Additionally, if multiple ROVs wished to be tested, they could not be interchanged, and would therefore each require their own tether.

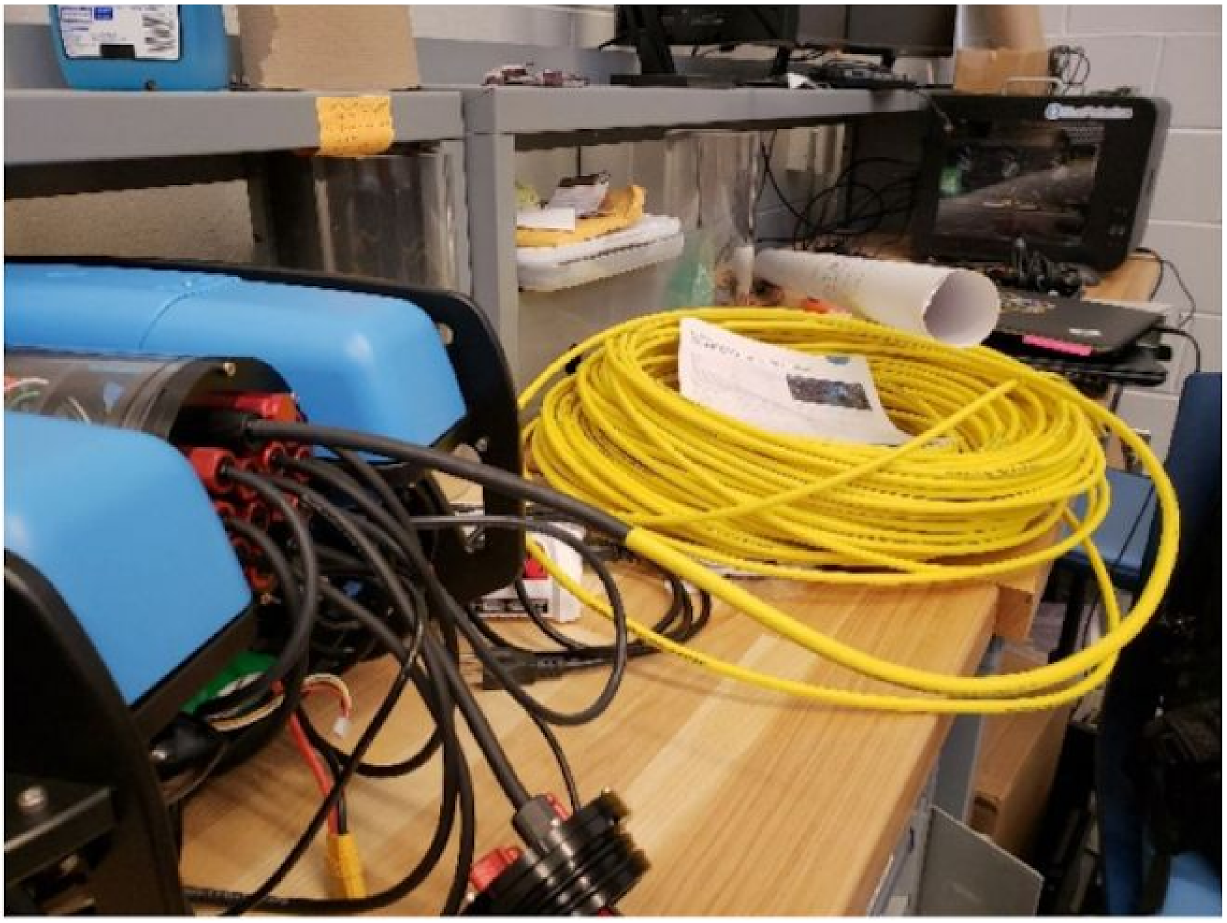


Figure 4: Original AUV Tether Connection

The tether mount also consisted of means to interface the tether with a computer and the UGPS system. The original system consisted of a cardboard “blackbox” consisting of two FathomX boards that served as interfacing nodes. The terminating end of the ROV tether was stripped to expose the wires, and the two in use were screwed into a terminal block. 5V leads were then used to connect the block to the inputs of the FathomX board (Figure 5). The FathomX boards were then connected to the computer using an ethernet cable and a USB to MicroUSB cable. The problem with this design was in operating the ROV, the tether would unspool to allow the ROV to travel longer ranges. If the interfacing system was wired while the tether was unspooling, then the wires would twist and endure increasing stress with each revolution of the reel. The result of this makeshift system was a constant threat of accidentally tearing various fragile wires in this system, and often the lead wires would fall from their ports causing a break in the connection. This required that the necessary length of tether be unspooled prior to connecting the tether to the interfacing system. If more tether was needed, or if it was time to respool the tether, the tether needed to be disconnected and reconnected for the spool to move without damaging anything.

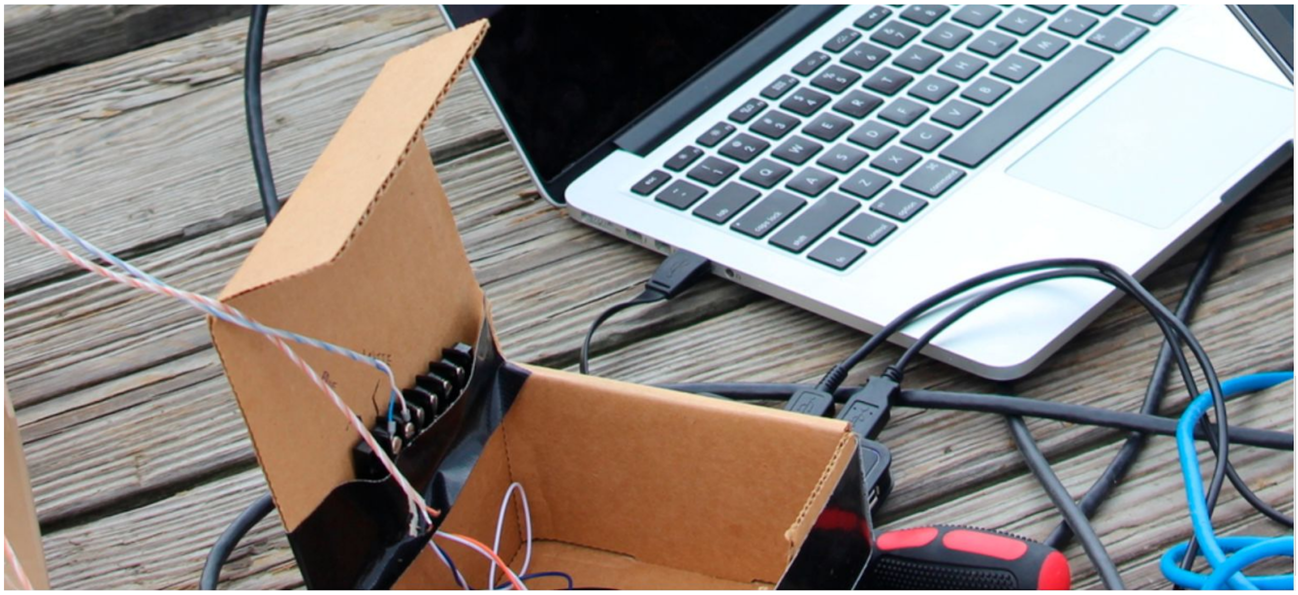


Figure 5: “Black Box” interface between the Tether, UGPS System, and Computer

A new tether reel and communication box was built using a slip ring to allow spooling and unspooling of the reel while the ROV was in operation. This also removed the need to connect or disconnect any wires between the reel and communication box. The new communication box was built from a conduit box with a lexan cover and was attached to the side of the tether reel. It holds the FathomX boards that form a network between the UGPS, base station and AUV. The communication box had plugs for the base station and UGPS and is the main hub of the entire AUV system. All of the connections are quick connectors to make it easy to assemble and disassemble. The AUV communication box is designed to be removable from the tether real and be attached to the ASV where it serves as the go-between for the ASV, UGPS and the ROV. Additionally, the hardwired tether connection was replaced with a 12 pin SubConn connection that was spliced into the tether. This allows the AUV to easily be connected and disconnected from the testing tether and the ASV’s tether, simplifying the process of transferring from one system to another

Underwater GPS

The underwater GPS system developed by Waterlink consists of multiple acoustic receivers listening for pings from an acoustic transmitter. The UGPS box shown in Figure 6 utilizes a satellite connection to determine its global location, and create a local North East Down coordinate frame. An acoustic transmitter is wired from the UGPS box onto whatever vehicle is being tracked, and gives off periodic sonic pulses. These waves are received by four acoustic receivers which use the timing of the sonar data to determine the location and heading of the transmitter.

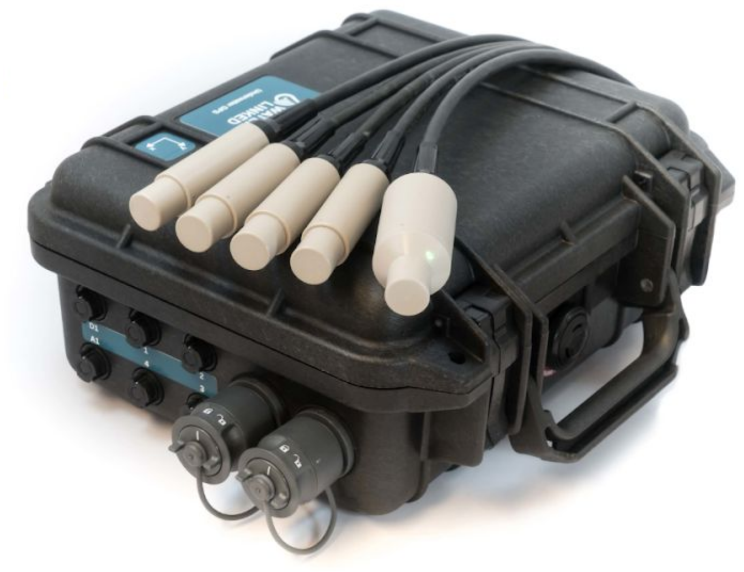


Figure 6: Waterlink UGPS System

The Waterlink UGPS system consists of a waterproof box that houses the control circuitry and a GPS module to give a reference location. The box is mounted to the top of the ASV, with the four receiver nodes mounted on retractable arms that deploy off the side. These nodes listen for the locator, which is mounted to the ROV, and use the signal to triangulate the position of the AUV. The original configuration utilized the *Waterlinked D1 Locator*. This locator requires a direct wired connection from the UGPS box to the ROV. This extra cable not only created problems for the dynamic controls of the ROV, but more so served as a hindrance to testing efficiency and repeatability. This specific locator created an additional wired connection to the surface that could not be autonomously managed/reeled, and behaved independently from the ROV tether. The *D1 Locator* is 81m in length, which created a large amount of cable to be managed and had a high risk of entanglement. Upon conducting multiple tests with the *D1*

Locator, it became evident that this device would inhibit testing and the eventual collaboration between the ROV and the ASV.

The *Waterlinked A1 Locator* was researched and selected to replace the *D1 Locator*. The *A1* is a modular adaptation of the *D1* that was specifically manufactured for platforms like the *BlueROV2*. This device is only 41mm in length, and is connected directly into the ROV's electronics tube using the ROV's own tether for communication. This connection immediately simplified the set up and execution of testing. This also simplified connections above the water. The UGPS box now only required one wire to the communication hub on the ROV tether spool. This change allowed for the construction of a platform to use the UGPS system independently of the ASV without a long setup time[4].

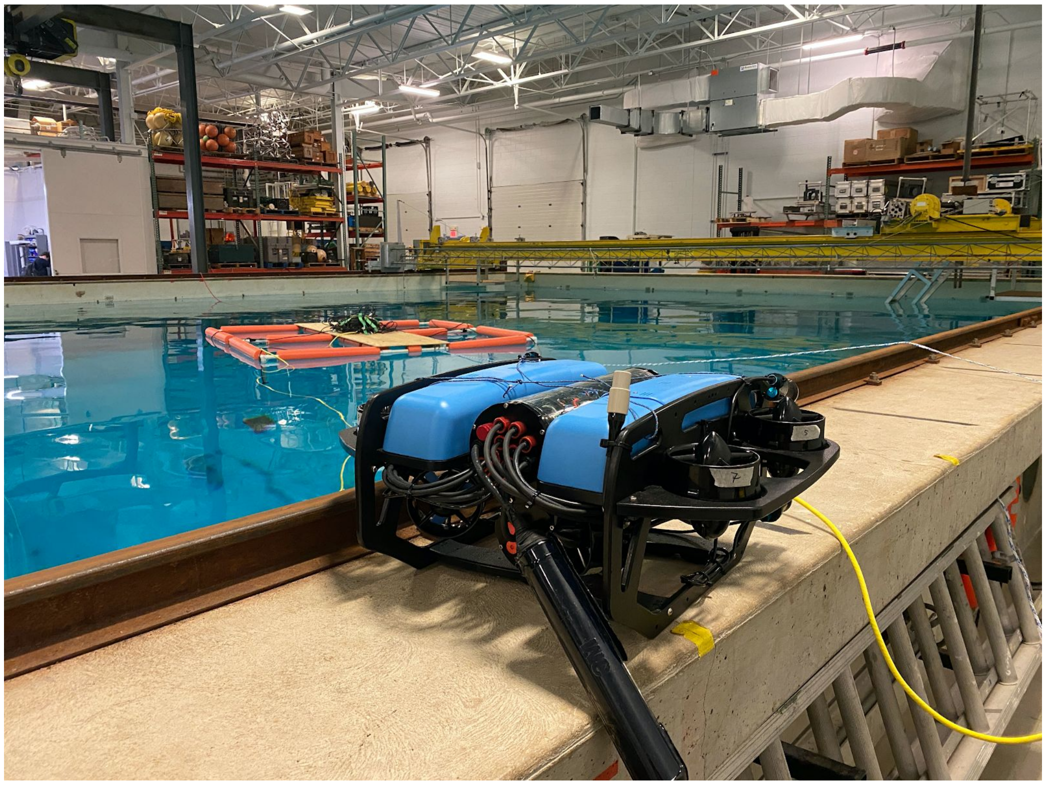


Figure 7:The A1 locator attached to the corner of the ROV and the UGPS platform deployed in the pool.

It was impractical to take the ASV out to the pool or lake every time the ROV was being tested, so a floating platform to hold the control box and receiver nodes were constructed. The platform can be seen floating in the middle of the Chase pool in Figure 7 above. This platform was made of PVC tubing, foam tubing for flotation and plywood. It disassembled for easy transport. The control box was mounted in the middle with the four receivers, each mounted to the end a section of PVC that extended down into the water. A 20' cable connected the base station to the UGPS control box. The previous setup required a triangle of wires between the UGPS box, tether box, and base station making it difficult to distance the three from each other. Now the UGPS box and base station do not have a direct connection which allows the UGPS box to be placed on the floating platform.

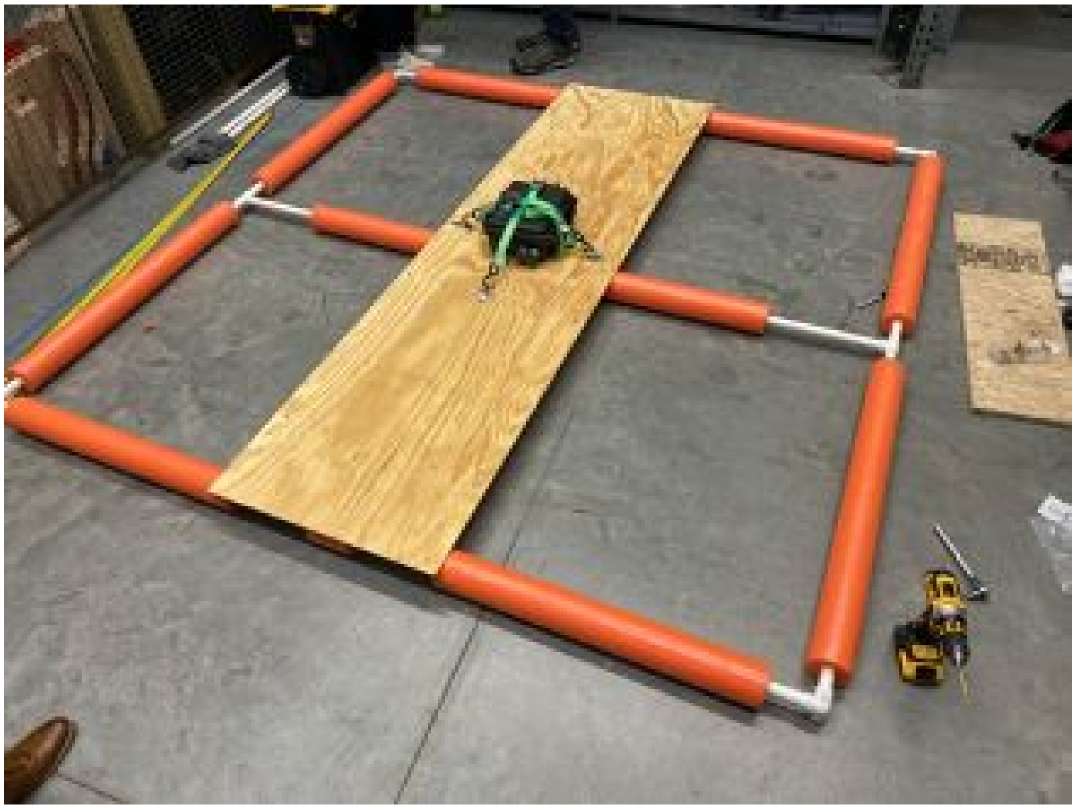


Figure 8: Initial Construction of the Testing Platform

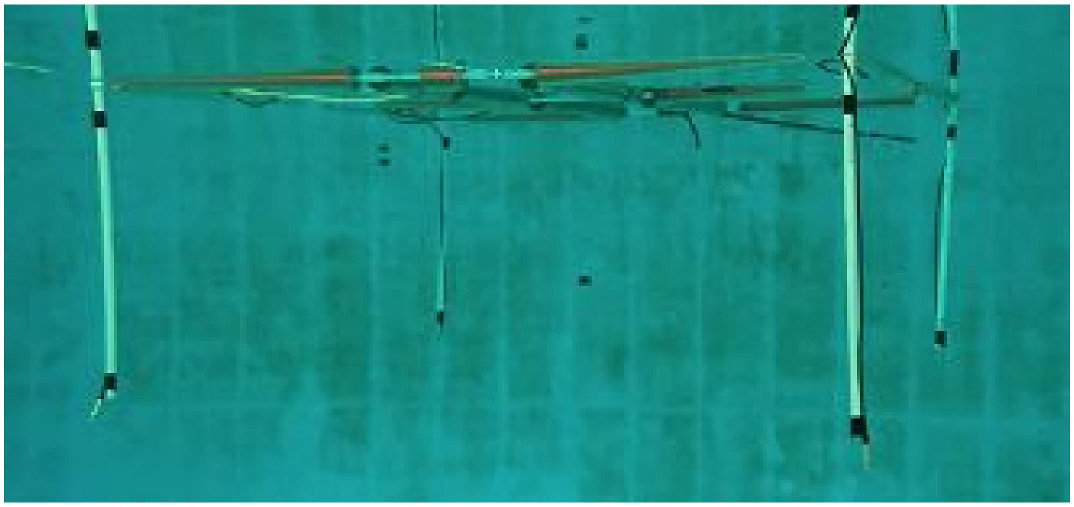


Figure 9: Underwater View of the Deployed Testing Platform

Controls

Although critical for fully independent autonomy, advanced control systems were not developed this academic year. The entire team participated in weekly supplemental lectures to better understand control system theory with the project advisors. These lectures were primarily focused on bridging the gap of understanding between complex theory and practical application. In addition these lectures helped the Computer Science team members better understand how the control systems work and why they are necessary for autonomy. The primary focus regarding controls was to establish a system that will facilitate the implementation of advanced systems by future students. Future teams will be able to focus on feedback from different control systems rather than troubleshooting issues with the testing procedures.

The challenge with controlling a complex object in 6 degrees of freedom begins with identifying the reference frames in which the motion will occur. As the AUV is only expected to travel at most approximately 50 meters from the ASV, this scenario can be simplified by using a North East Down (NED) global coordinate frame {n}. While this coordinate system assumes the Earth is flat, it will still remain very accurate due to the sheer size of the Earth compared to our functional environment. The second reference frame used is the body frame {b}, as shown in Figure 10. Frame {n} is used to track the positions and attitude of the AUV while frame {b} provides the reference for all forces and moments. The notations that will be used for the controls calculations are shown in Table 2

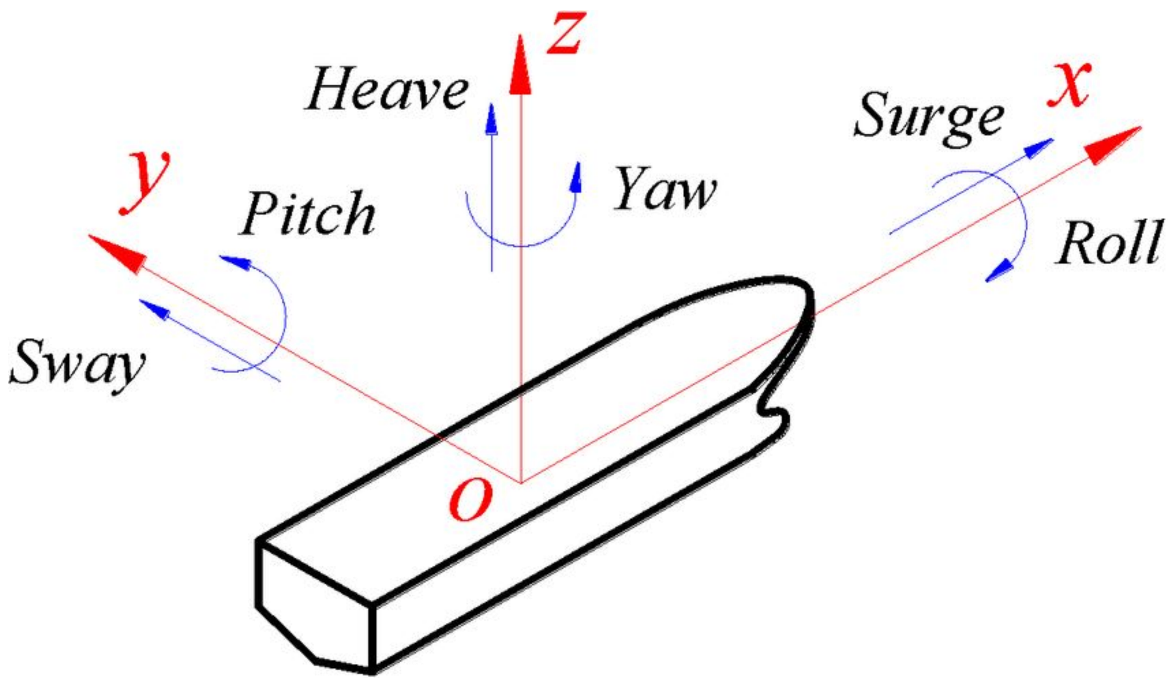


Figure 10: Body Frame Concept Utilized for Theoretical Controls

Table 2: Notation Used for Theoretical Controls [3]

Degree of Freedom	Description	Force/Moment	Position/Attitude	Velocity
1	Surge	X	N	u
2	Sway	Y	E	v
3	Heave	Z	D	w
4	Roll	K	ϕ	p
5	Pitch	M	Θ	q
6	Yaw	N	Ψ	r

The position and attitude of the {b} reference frame with respect to the {n} frame is cataloged in the matrix $\eta^n_{b/n}$, and the velocity vector of the body with respect to {n} expressed in {b} are captured in $\nu^b_{b/n}$,where

$$\eta^n_{b/n} = [N\ E\ D\ \ \Theta\ \Psi]^T \text{ and } \nu^b_{b/n} = [u\ v\ w\ p\ q\ r]^T \quad (1)$$

The velocities and angle rates of {b} with respect to {n} can be expressed as

$$\tfrac{d}{dt}\eta^{n}_{b/n} = J(\eta) * \nu^b_{b/n} \quad (2)$$

Where $J(\eta)$ is the 2x2 transformation matrix

$$J(\eta) = \begin{vmatrix} R^n_b & 0_{3x3} \\ 0_{3x3} & T_\Theta(\Theta) \end{vmatrix} \quad (3)$$

R^n_b is the rotational matrix for converting the linear velocities in {b} into the {n} frame, and $T_\theta(\theta)$ is used for converting the angular velocities. Equation (2) is the full kinematic model of the vehicle. With these matrices, the full hydrodynamics of the AUV can be expressed as

$$M_A \tfrac{d}{dt}\nu + M\tfrac{d}{dt}\nu + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau \quad (4)\ [2]$$

Where the M_A is the hydrodynamic added mass, M is the inertia matrix, C is the coriolis matrix, D is the damping matrix, and g is the hydrostatic buoyancy force. All of these values equate to Tau, which is the matrix of control forces and moments in six degrees of freedom. The compositions of matrices M_A, M, C, D, and g are constantly changing, and therefore must be calculated as part of a feedback loop controls system.

This system is simplified by making three assumptions. The first is that AUV's geometric, gravitational, and buoyancy centers are the same location. In reality, the gravitational and buoyancy centers of the AUV would not be at the same height, which is what causes the AUV to naturally float or sink, but the difference between these points is small enough we can neglect it. The second assumption is that given the design of the AUV and its current functionality, it will

not pitch or roll ($= \Theta = 0$). These two assumptions greatly simplify the Kinetic and Kinematic Equations used in the controls algorithm. The final assumption is that the AUV will travel at very slow speed (less than 1 knot). Since the thruster configuration of the Blue ROV Heavy2 has 3 planes of symmetry, this will greatly simplify the coriolis matrix C.
The control forces and moments matrix τ is a result of the forces output by the thrusters used by the ROV. These output forces can be expressed as

$$F = k * u \quad (5)$$

where *u* and *k* are matrices representing controller input and thrust coefficients, respectively. Figure 11 shows the thruster configuration of the Blue ROV Heavy2. As seen in the figure, half of the thrusters do not point in the direction of a principal axis, and they all affect multiple degrees of freedom. Matrix F only provides the magnitudes of the forces. Therefore, the T matrix needs to be introduced to relate matrix F to the τ matrix. The T matrix is the thrust configuration matrix which, when multiplied to the F matrix, will produce a matrix of the resulting effect each thruster has on the six degrees of freedom. The τ matrix can therefore be expressed as

$$\tau = t * k * u \, (6)$$

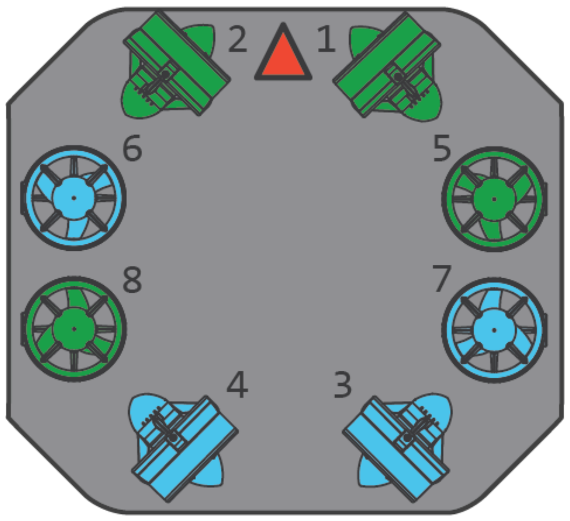


Figure 11: Thruster Configuration

The theoretical feedback control loop utilized by the AUV can be seen in the block diagram shown in Figure 12.

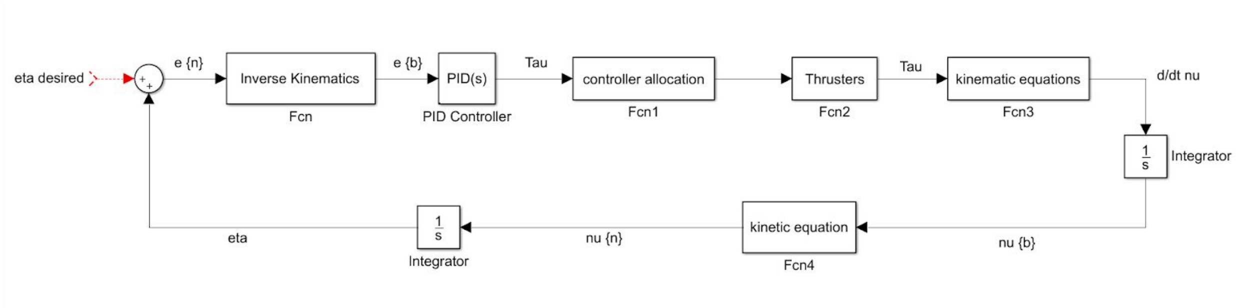


Figure 12: General 6 Degree of Freedom Control Loop

Testing

There were no formal testing procedures at the beginning of the academic year. One of the first goals was to streamline testing and develop a procedure. The first few tests required 1 hours to set up because of the old UGPS and tether system. With the new UGPS beacon, platform and tether reel, set-up was reduced to 20 minutes. A testing procedure was developed along with a procedure manual created for future teams. The new instruction manual was created with set-up and tear-down procedures for the UGPS and ROV systems. It also contains instructions for battery charging and operation of the base station, along with ROV operation.

The Testing Area

The final proof-of-concept mission will take place at the Jackson Estuary Laboratory in Durham, New Hampshire. This lab lies on the coast of the Piscataqua River, and has a boat dock and crane that will be used to deploy the ASV and ROV unit. This location is unique in that it is connected to the ocean, therefore both maritime vehicles will be exposed to saltwater. Additionally, geographic dynamics of this estuary create very fast currents which would overpower the capabilities of the vehicles, so any testing or missions in this area can only be performed during slack tide. This is additional motivation for why testing improvement was so necessary.

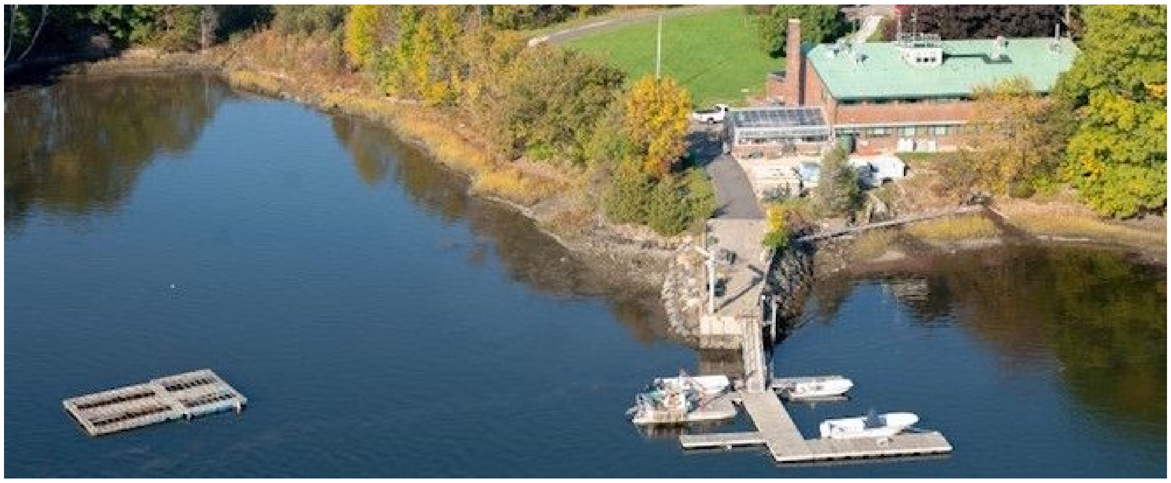


Figure 13: Jackson Estuary Laboratory

The majority of testing was performed in the Chase Ocean Engineering Lab Test Tank. The 60x40x20 foot tank provided ample room to test each component of this system. This is a saltless aquatic environment which reduces corrosion, and has access to multiple cranes and viewing platforms which can be used for deployment and monitoring.

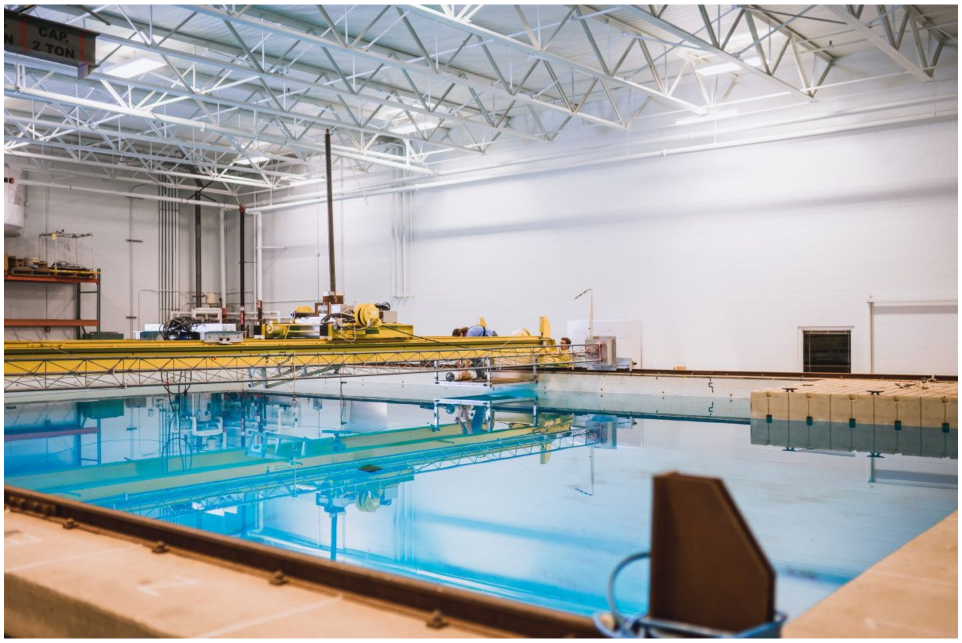


Figure 14: Chase Ocean Engineering Laboratory Testing Tank

The User Interface

To test the performance of this AUV, there would need to be a way to measure the systems ability to navigate from waypoint to waypoint. This was accomplished using the QGroundControl User Interface. This UI provided a means of manipulating the AUV both manually with an Xbox 360 controller and unmanned by integrating the UGPS system and allowing for the choosing of waypoints. To test and refine the AUVs autonomy, QGroundControl was utilized to provide waypoint instruction to the AUV. The various tasks included depth hold, single waypoint navigation, multiple waypoint navigation, and return to home. In each test, the AUV was judged on the error between the planned path and the actual path taken by the AUV.

Initial Testing

The initial rounds of testing were performed with the goal of confirming a fully functioning system before engaging the aspects that needed to be improved. Manual control of the AUV proved to be fully functioning, and the robot responded well to input from the controller. The AUV was able to receive waypoints chosen in QGroundControl, and traverse the water smoothly when given a heading. However, these first few rounds of testing did reveal a few minor issues. The first of which was that the waypoints were chosen arbitrarily with no knowledge as to where exactly they were in the tank, which made it impossible to judge the accuracy of the waypoint navigation. But the biggest problem was that the voltage surges from multiple motors being turned on would temporarily cause major drops in the perceived voltage of the battery. This in turn would trigger the low battery protocols of the AUV and cause it to stop its mission and surface. Therefore the range that the AUV could travel was greatly limited because the low battery protocol would continuously cut missions short.

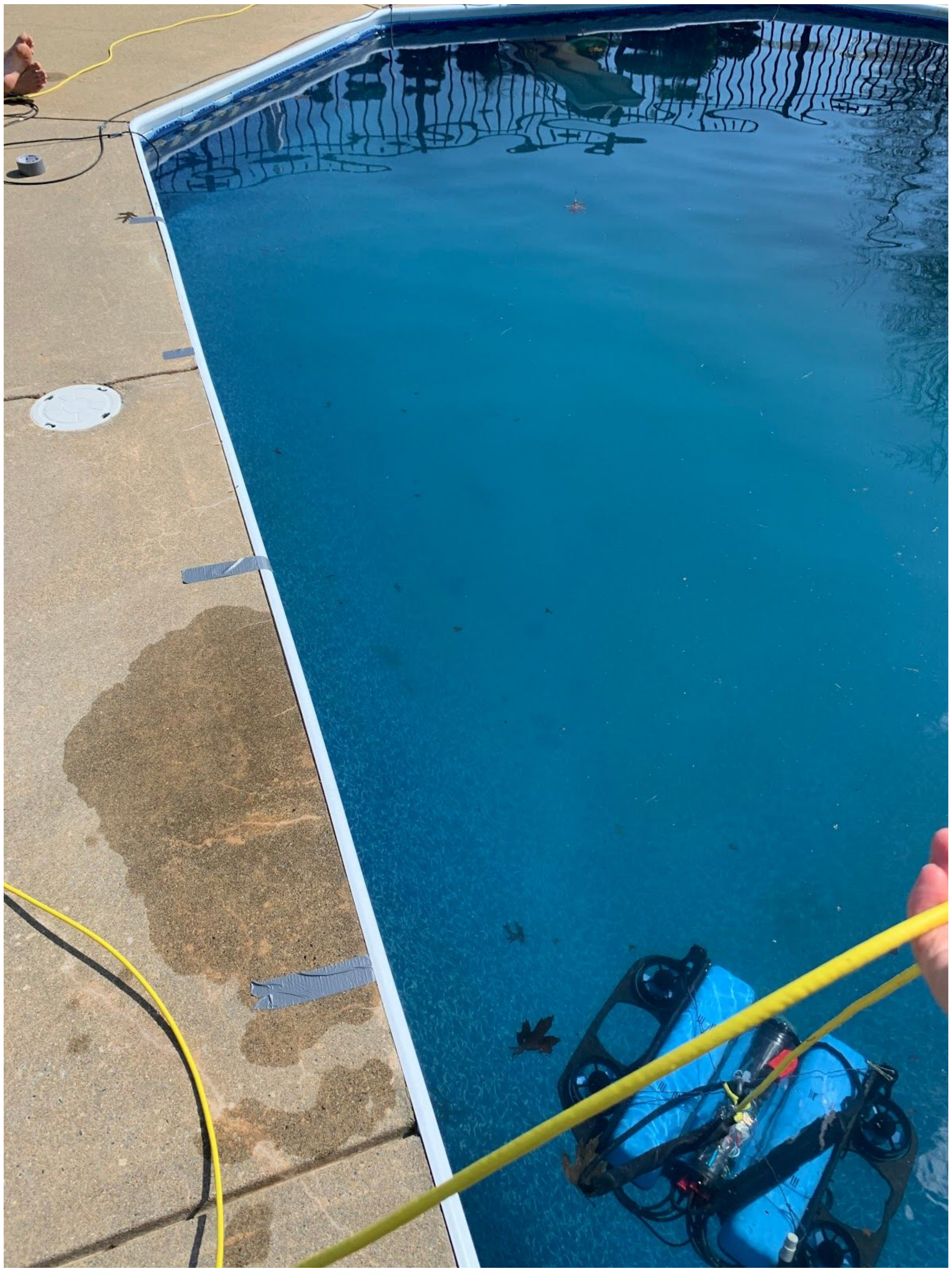


Figure 15: Attempted AUV Test in Smaller Testing Area

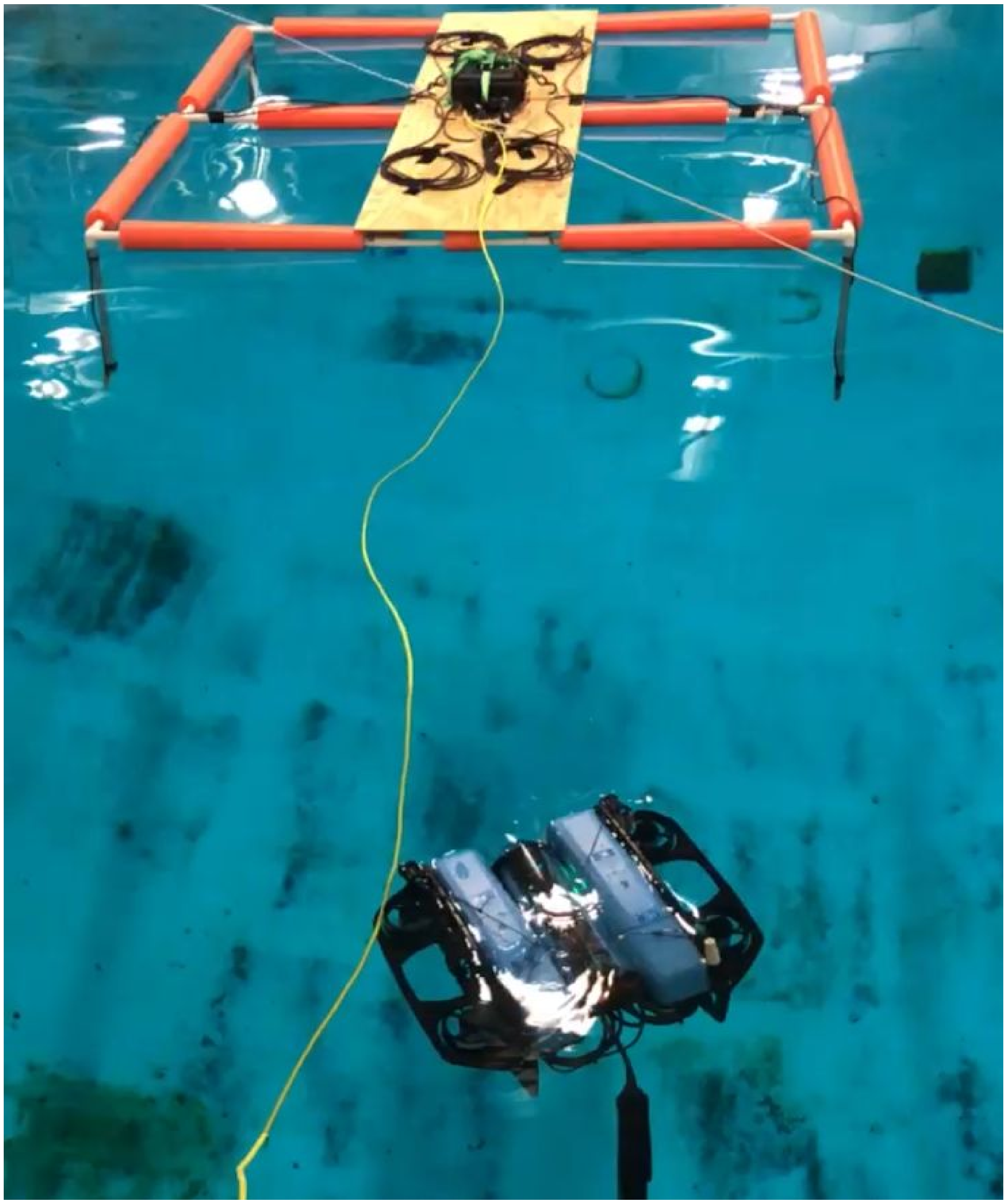


Figure 16: Waypoint Navigation Test in the Chase Test Tank

Refined Testing

The above mentioned issues were addressed in the following test sessions. The battery issue was resolved by lowering the threshold for the low battery protocol to a value unreachable by a 4 cell Lipo battery. This did require the user to be mindful of how long the AUV had been active to ensure the battery was not drained beyond rechargeability, but it allowed for the AUV to complete any mission given to it. Further exploration of QGroundControl revealed the ability to input exact latitude and longitude coordinates in decimal form. Assuming the Earth to be a perfect sphere with a radius of 6371 km, a conversion from degrees to meters can be found using the arc length formula and converting from degrees to radians

$$s = r\theta \quad (7)$$

With the ability to establish the locations of exact waypoints, the performance of the AUV could be tested.

Unfortunately the testing environments no longer became available as a result of the school being transitioned to virtual presences, and the refined testing could not be pursued. Therefore, a testing procedure for optimizing the gains of the controller was drawn up so that this test could be done quickly and efficiently when the facilities became available again. A measuring tape will be used to determine the distance from a visible landmark outside of the Chase Ocean Engineering building to the southwest corner of the tank , which would serve as the starting point. The AUV only moves in the surge, heave, and yaw directions (Figure 10), so these controller gains would be the only ones tested.

To test the surge controller, a single waypoint will be chosen a distance of 5 meters from the starting position, 1.11 meters away from the tank's east wall. When the mission is started, the AUV should travel in a straight line to the waypoint and stop at the desired position. The accuracy of the surge controller will relate to the overshoot of this position, and the how the AUV decelerates to stop at this position. Through repeating the test while varying the P,I,and D gains of the surge controller, an optimal PID controller can be found for surge
The heave controller will be tested in a similar manner to the surge controller. Evenly spaced markers will be placed vertically along the walls of the tank, and the AUV will be started at a depth of 4 meters. The AUV will be instructed to travel to a waypoint 3 meters above it. Trial and error will once again be used to optimize the heave gains based on overshoot and deceleration. The yaw accuracy will have to be tested slightly differently than the previous two. This system only has the capability to receive waypoints, so it is impossible to send it its own waypoint at a different orientation. Instead, the AUV will begin in its starting position and be given a waypoint 90 degrees to its left 5 meters away. The protocols of the AUV will cause it to yaw first before driving towards the waypoint. If the AUV overshoots on the yaw, the results of this will be seen in the AUVs ability to hold a straight path to the waypoint. Therefore, the accuracy of the yaw controller can be improved by analyzing the lateral distance from the AUV to the tank wall while it travels towards its waypoint. The yaw PID controller would be optimized when the change of this lateral distance was minimized.

Computer Science Work

Autonomy and the ROS Network

Previously, the ASV and AUV functioned off of an operating system developed by the Navy called SeaMoose. This system was not open sourced and was very difficult to integrate into a multiplatform system. Therefore, an imperative task was to change this system to utilize a ROS network. The Robotic Operating System (ROS), is an open source software platform that provides libraries and tools for creating and building robotic applications. Since it is open source, there are extensive resources on the internet to access to aid in utilizing it for the purposes of this project. The ROS network functions by identifying each controller and computer as a node and interconnecting each of them in a network. The nodes are able to communicate with each other by identifying itself as a publisher or subscriber to a communication channel. It was decided that the actual autonomy of this system would take place in the ASV computer, and the AUV's autopilot capabilities would serve as a subsystem able to be initiated using the publisher-subscriber relation of ROS. The ASV Autonomy will operate using front seat - back seat configuration where the backseat will use feedback from the systems sensors to make decisions, and then send commands to the front seat, which would then in turn operate the components of the multiplatform system.

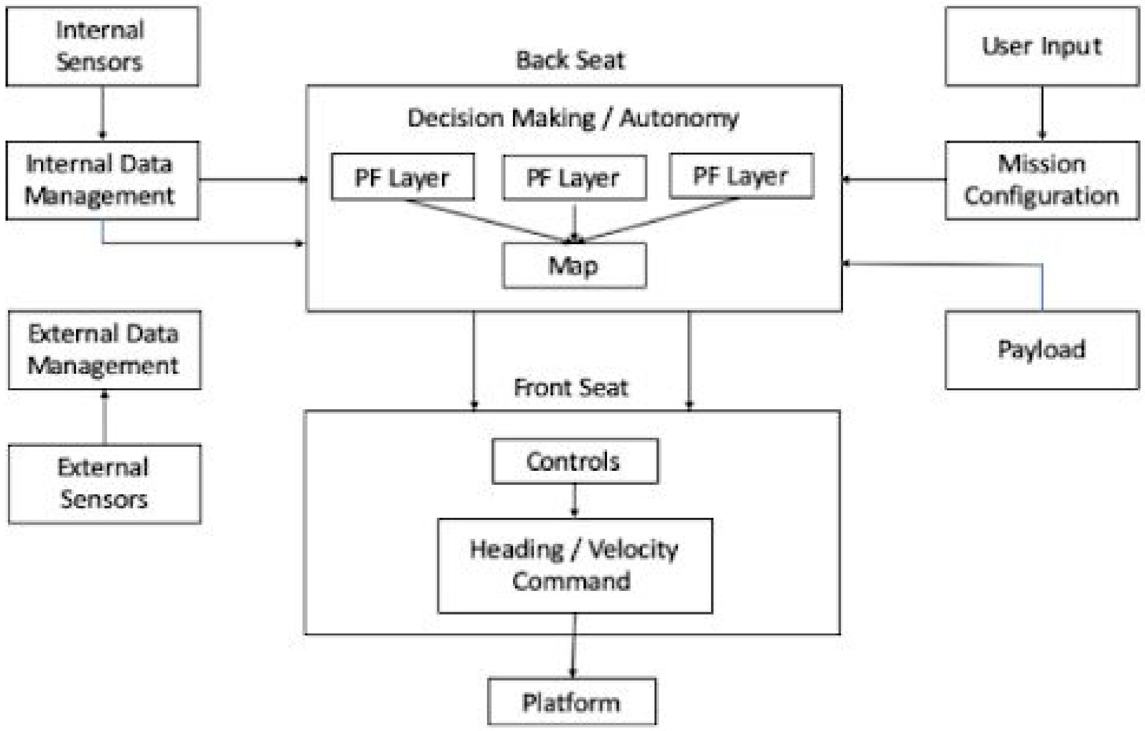


Figure 17: Map of the Front Seat - Back Seat Autonomy

As the final goal of this project is to be able to receive waypoints from the ASV, The manual waypoint selection used in the testing will not be sustainable for the life of this project. To make this goal a reality, the AUV would need to receive waypoints outside of the User Interface. Thankfully, the UI is just a simplified means of using the AUV. The actual communication is done with MAVLink. MAVLink is a library used to communicate with the autopilot systems of the AUV. This communication network can be accessed from the command line and shortcutted with a python script. This is extremely important because the ROS network would allow for the python script to be run on the shorestation and have the signal be carried all the way down to the AUV.

A Python script was developed that utilized the MAVLink library to provide waypoints to the AUV. This script allows for the user to choose waypoints in the form of longitude and latitude (in decimal form) and choose the required accuracy of the waypoint navigation. The required accuracy is represented by an acceptable radius from which the AUV may deviate from the path. This radius is a requirement of the MAVLink communication, but nothing found in analyzing the existing files showed that it was related to the gains of the controller being used by the AUV, or had any effect on the performance of the AUV. However, this script successfully transmits waypoints to the AUV, and can therefore be used to achieve the goals of autonomy.

Simulation

With the school transitioning into virtual classes, the testing resources of the Chase Engineering Lab became unavailable. Therefore, testing efforts were transferred to using computer simulation to test the performance of the AUV and the effectiveness of the python script. The ArduSub git repository includes a simulation program that allows for one to utilize the MAVLink communication network to simulate the movements of the AUV. This program was used with the python script to simulate the AUVs ability to navigate the Chase Engineering Tank. The waypoints chosen created a path where the AUV would follow the walls of the tank counterclockwise while maintaining a 1.11 meter distance from the wall and holding a depth of 1.5 meters. The measurements were converted into latitude and longitude for inputs into the Python script and the waypoints were sent to the AUV. The resulting path can be seen in Figure 18.

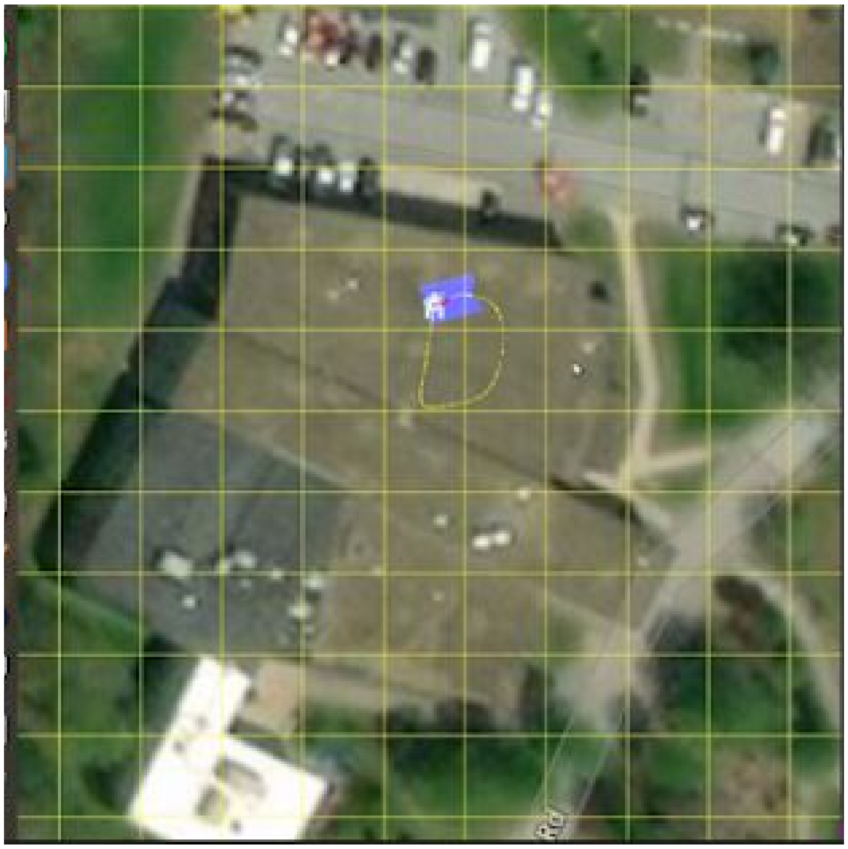


Figure 18: Path of the Simulated AUV

While the simulated AUV did follow the path, its ability to hit the desired points showed poor accuracy. It was able to successfully reach the first point, but struggled to maintain the strict path connecting to the 2nd and 3rd points and the return to position 1. This is a result of improper gains in the controllers being used by the AUV. By using the trial and error testing procedure outlined in the Refined Testing section in the simulation, the optimal gains can be found for the AUV that will maximize its ability to follow a path. However, this simulation assumes a perfect environment and system, which would not be the case in the real world. In physical testing, some turbines may provide more thrust than others, the thrust will not be impulsive, and waves and currents will affect the movement of the AUV. Additionally, the tether itself will create inconsistent and unpredictable drag on the AUV. While the simulation's optimal gains do not correspond to the optimal gains of the physical system, they will provide a sufficient starting point for when physical testing will resume.

Future Work

Control Gains Improvements

With the testing procedure refined and the plan for improving the controller gains mapped out, the next stage will be to perform the physical testing outlined in the Refined Testing section. The most optimal area for this test to take place would be at the estuary or another open water environment. While the Chase Tank provides a sufficient controlled area for testing, The solid concrete walls of the tank often can result in inaccurate data due to acoustic resonance. Having an open environment like the estuary would remove this effect and provide accurate data.

Receiving Waypoints from the ROS Network

As discussed before, the end goal of this project would be to be able to select waypoints outside of the user interface. Currently the Python script has the ability to send waypoints to a simulated AUV and have it perform a navigation mission. Future developments would include modifying this script to work with the physical system, as well as being able to receive the waypoints from another source. Compatibility with the physical AUV and the ability to receive waypoints from an external source would allow this script to be effectively run from the shore station of the AUV system and receive waypoints from a user or from collected sonar data.

Proof of Concept Modularity

A major part of this project is the concept of modularity. This system needs to be able to function across a variety of platforms without having to be excessively redesigned for that platform. This proof of concept can be achieved by either purchasing a second Blue ROV Heavy2 or by building a new AUV, and then flashing the current operating system onto this new platform. Proving modularity is important because this project is operating on a limited budget with economical components, so showing that the operating system can be moved onto a new system allows for the potential of this multi platform system to be incorporated onto a project with greater resources, and therefore can achieve results more constructive to ocean exploration

Tether Management

An important part of the AUVs navigational ability is tether management. There needs to be enough tether deployed such that the AUV can travel to its desired waypoint, but not too much as to cause unnecessary drag or an obstacle for the AUV to be tangled in. Additionally, the tether needs to be deployed in such a way that there is always enough tension in the tether to prevent it from unspooling around the reel. The solution proposed by this year's team involves using a pulley and a force gauge to track when the AUV is pulling on the tether. The system would function by passing the tether over a pulley suspended by a very stiff spring. The spring, in turn, would be connected to a force gauge that provides voltage outputs back to the Arduino that controls the tether reel. The Arduino would incorporate bang-bang control to utilize the level of voltage as a threshold for turning the winch motor on or off. This way, the winch would only be activated when there was enough tension in the line to prevent the unspooling, and when the AUV was moving.

Acknowledgements

The UNH ROV team would like to thank the Naval Engineering Education Consortium (NEEC), specifically Dr. Martin Renken and Thai Tran, from the Naval Undersea Warfare Center (NAVSEA) in Keyport, WA for their continued support and sponsorship.

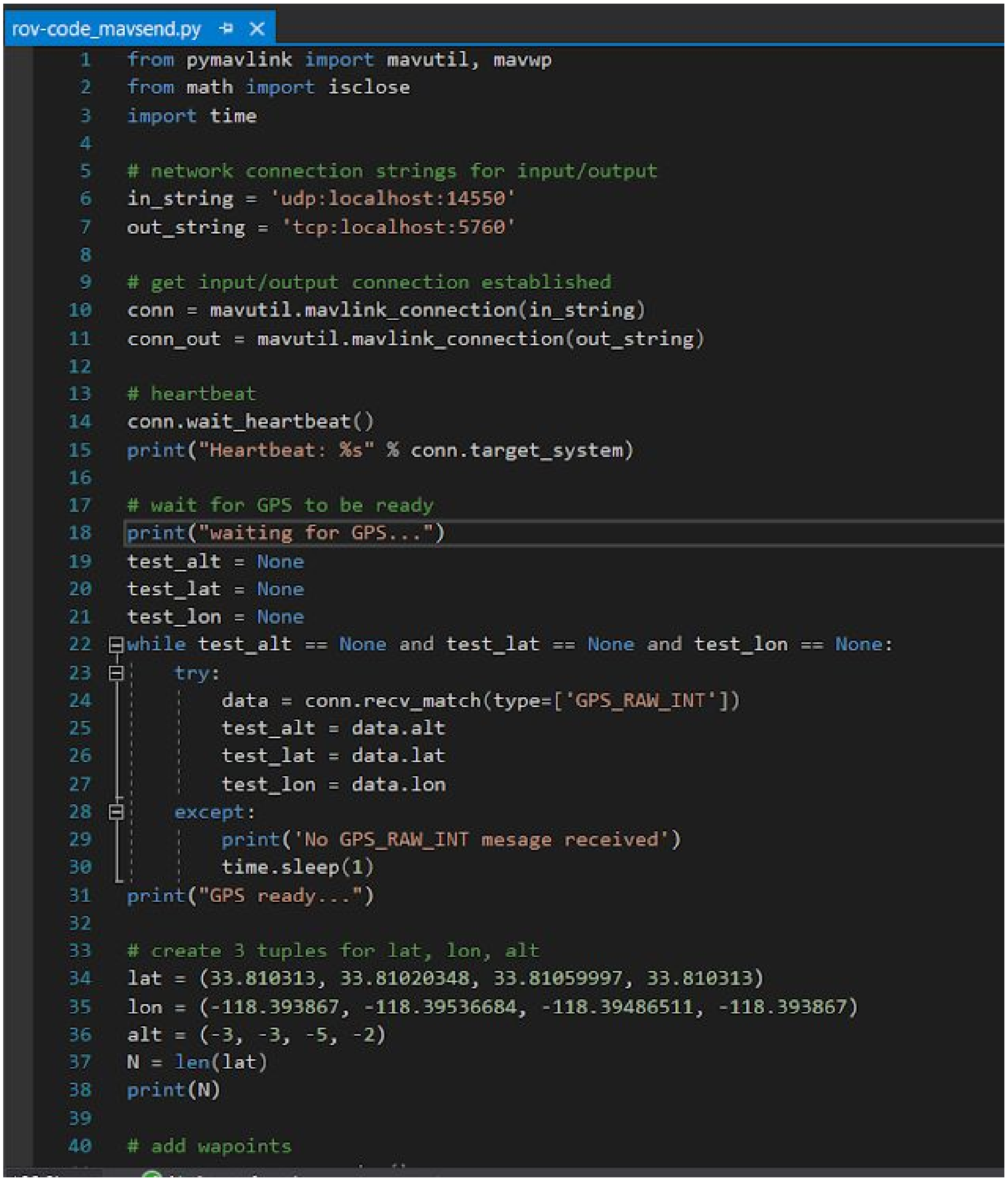
This research has been sponsored in part by the New Hampshire SeaGrant College Program through NOAA and the UNH Marine Program

Additionally we'd like to thank our professional advisor Dr. May-Win Thein and our graduate student advisors Oruc Oruc, Allisa Dalphine, and Alex Cook for their continued support and guidance.

Finally, we'd like to thank John Ahern for allowing us to use the Chase Test Tank and surrounding facilities.

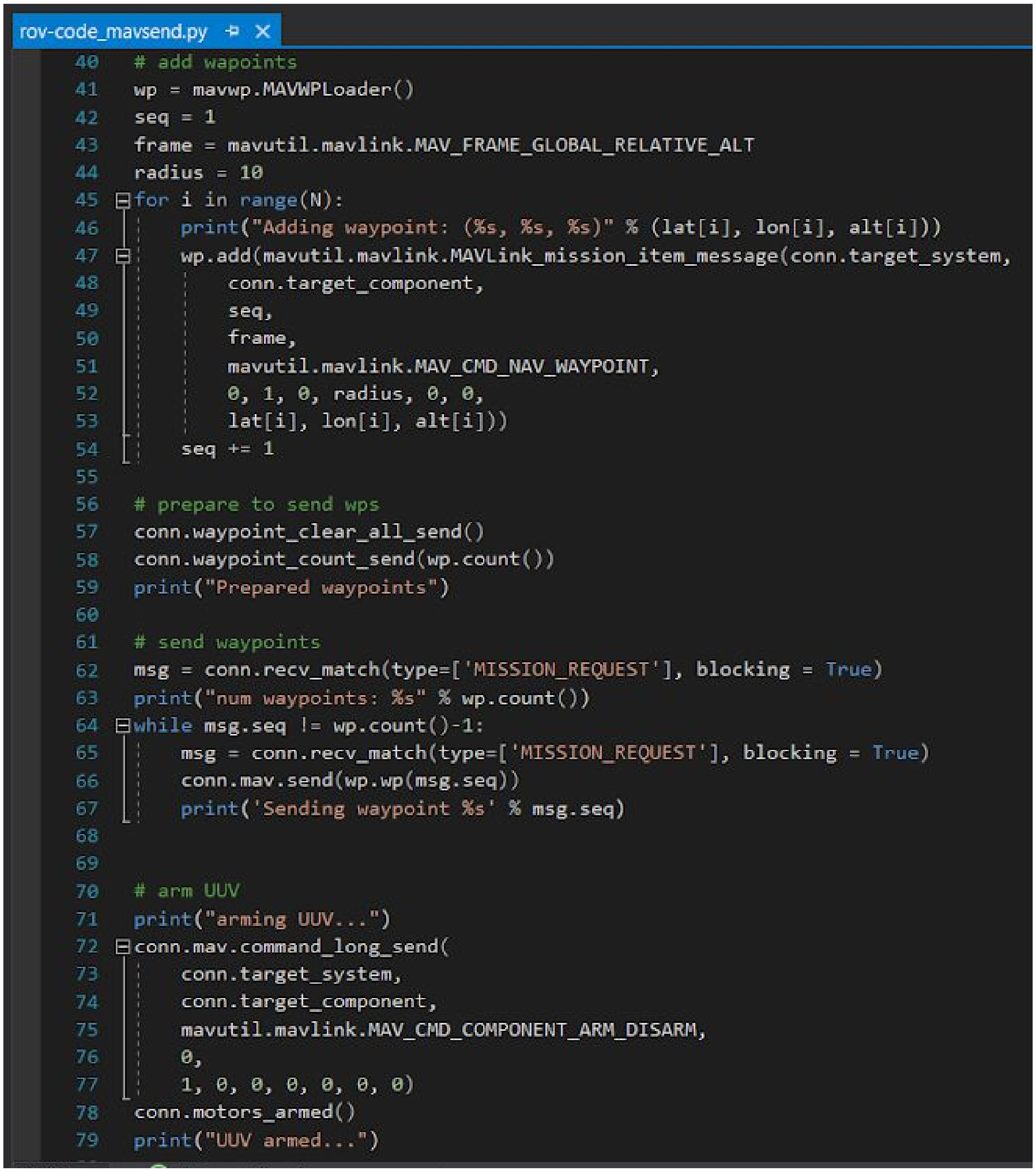
Appendix

Python Script

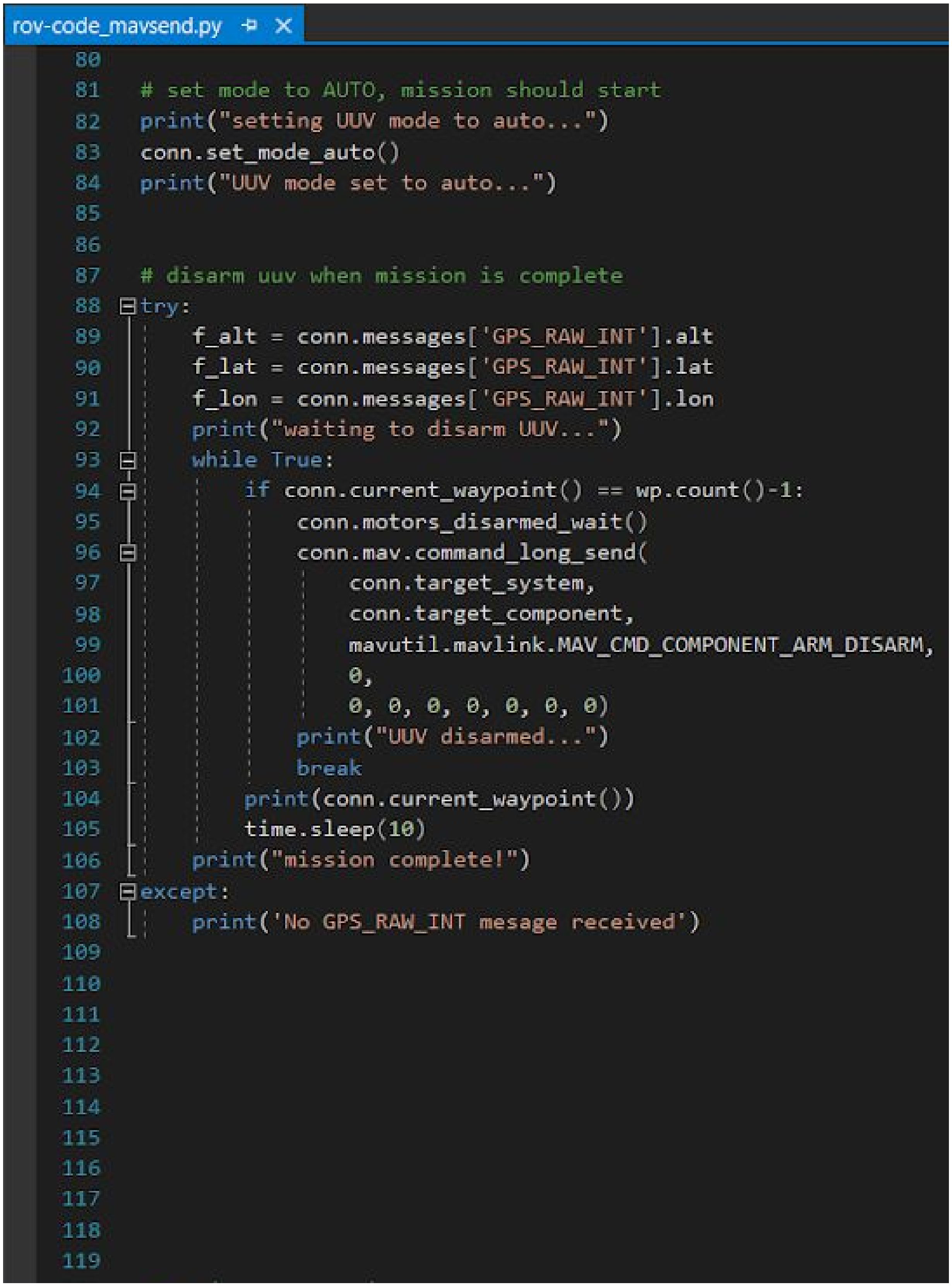


rov-code_mavsend.py

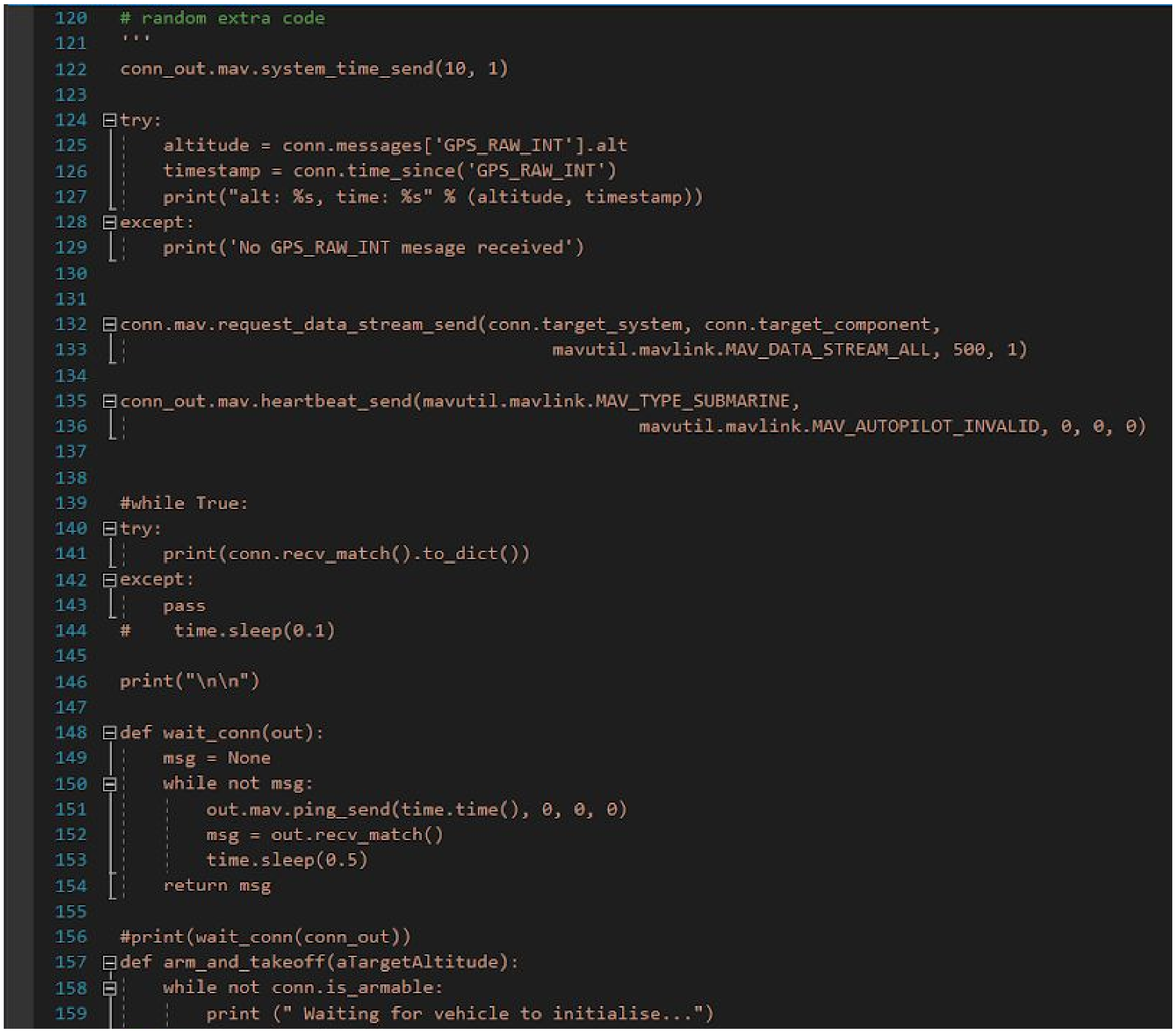
```
from pymavlink import mavutil, mavwp
from math import isclose
import time

# network connection strings for input/output
in_string = 'udp:localhost:14550'
out_string = 'tcp:localhost:5760'

# get input/output connection established
conn = mavutil.mavlink_connection(in_string)
conn_out = mavutil.mavlink_connection(out_string)

# heartbeat
conn.wait_heartbeat()
print("Heartbeat: %s" % conn.target_system)

# wait for GPS to be ready
print("waiting for GPS...")
test_alt = None
test_lat = None
test_lon = None
while test_alt == None and test_lat == None and test_lon == None:
    try:
        data = conn.recv_match(type=['GPS_RAW_INT'])
        test_alt = data.alt
        test_lat = data.lat
        test_lon = data.lon
    except:
        print('No GPS_RAW_INT mesage received')
        time.sleep(1)
print("GPS ready...")

# create 3 tuples for lat, lon, alt
lat = (33.810313, 33.81020348, 33.81059997, 33.810313)
lon = (-118.393867, -118.39536684, -118.39486511, -118.393867)
alt = (-3, -3, -5, -2)
N = len(lat)
print(N)

# add wapoints
```



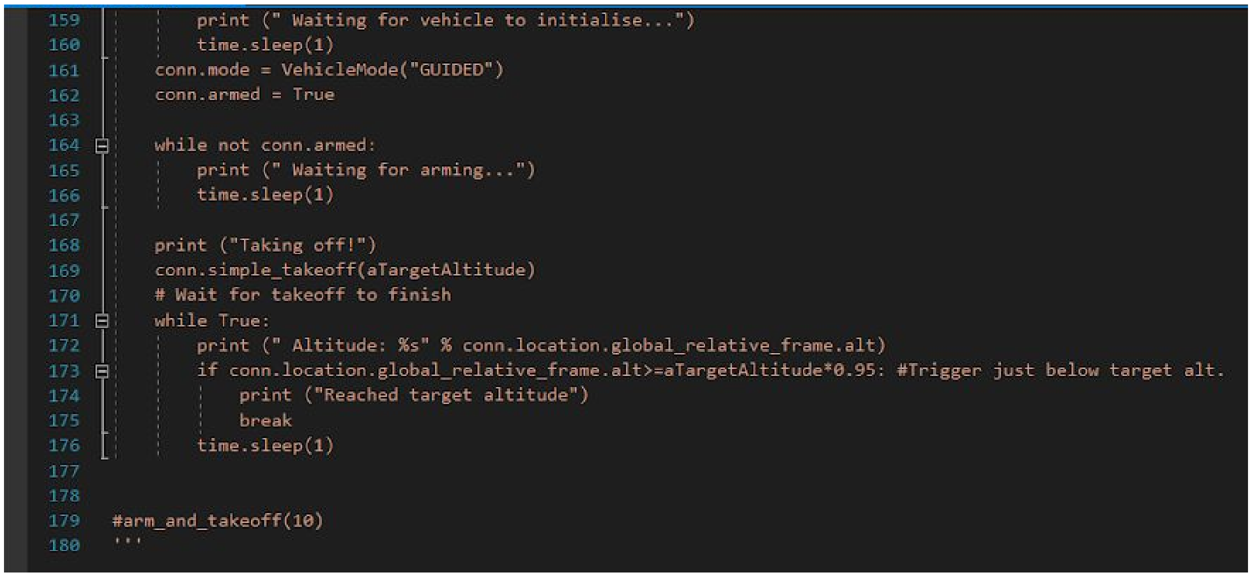
```python
# add wapoints
wp = mavwp.MAVWPLoader()
seq = 1
frame = mavutil.mavlink.MAV_FRAME_GLOBAL_RELATIVE_ALT
radius = 10
for i in range(N):
    print("Adding waypoint: (%s, %s, %s)" % (lat[i], lon[i], alt[i]))
    wp.add(mavutil.mavlink.MAVLink_mission_item_message(conn.target_system,
        conn.target_component,
        seq,
        frame,
        mavutil.mavlink.MAV_CMD_NAV_WAYPOINT,
        0, 1, 0, radius, 0, 0,
        lat[i], lon[i], alt[i]))
    seq += 1

# prepare to send wps
conn.waypoint_clear_all_send()
conn.waypoint_count_send(wp.count())
print("Prepared waypoints")

# send waypoints
msg = conn.recv_match(type=['MISSION_REQUEST'], blocking = True)
print("num waypoints: %s" % wp.count())
while msg.seq != wp.count()-1:
    msg = conn.recv_match(type=['MISSION_REQUEST'], blocking = True)
    conn.mav.send(wp.wp(msg.seq))
    print('Sending waypoint %s' % msg.seq)


# arm UUV
print("arming UUV...")
conn.mav.command_long_send(
    conn.target_system,
    conn.target_component,
    mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM,
    0,
    1, 0, 0, 0, 0, 0, 0)
conn.motors_armed()
print("UUV armed...")
```


rov-code_mavsend.py

```

# set mode to AUTO, mission should start
print("setting UUV mode to auto...")
conn.set_mode_auto()
print("UUV mode set to auto...")


# disarm uuv when mission is complete
try:
    f_alt = conn.messages['GPS_RAW_INT'].alt
    f_lat = conn.messages['GPS_RAW_INT'].lat
    f_lon = conn.messages['GPS_RAW_INT'].lon
    print("waiting to disarm UUV...")
    while True:
        if conn.current_waypoint() == wp.count()-1:
            conn.motors_disarmed_wait()
            conn.mav.command_long_send(
                conn.target_system,
                conn.target_component,
                mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM,
                0,
                0, 0, 0, 0, 0, 0, 0)
            print("UUV disarmed...")
            break
        print(conn.current_waypoint())
        time.sleep(10)
    print("mission complete!")
except:
    print('No GPS_RAW_INT mesage received')
```



```
# random extra code
'''
conn_out.mav.system_time_send(10, 1)

try:
    altitude = conn.messages['GPS_RAW_INT'].alt
    timestamp = conn.time_since('GPS_RAW_INT')
    print("alt: %s, time: %s" % (altitude, timestamp))
except:
    print('No GPS_RAW_INT mesage received')


conn.mav.request_data_stream_send(conn.target_system, conn.target_component,
                                        mavutil.mavlink.MAV_DATA_STREAM_ALL, 500, 1)

conn_out.mav.heartbeat_send(mavutil.mavlink.MAV_TYPE_SUBMARINE,
                                                mavutil.mavlink.MAV_AUTOPILOT_INVALID, 0, 0, 0)


#while True:
try:
    print(conn.recv_match().to_dict())
except:
    pass
#    time.sleep(0.1)

print("\n\n")

def wait_conn(out):
    msg = None
    while not msg:
        out.mav.ping_send(time.time(), 0, 0, 0)
        msg = out.recv_match()
        time.sleep(0.5)
    return msg

#print(wait_conn(conn_out))
def arm_and_takeoff(aTargetAltitude):
    while not conn.is_armable:
        print (" Waiting for vehicle to initialise...")
```



```
        print (" Waiting for vehicle to initialise...")
        time.sleep(1)
    conn.mode = VehicleMode("GUIDED")
    conn.armed = True

    while not conn.armed:
        print (" Waiting for arming...")
        time.sleep(1)

    print ("Taking off!")
    conn.simple_takeoff(aTargetAltitude)
    # Wait for takeoff to finish
    while True:
        print (" Altitude: %s" % conn.location.global_relative_frame.alt)
        if conn.location.global_relative_frame.alt>=aTargetAltitude*0.95: #Trigger just below target alt.
            print ("Reached target altitude")
            break
        time.sleep(1)


#arm_and_takeoff(10)
'''
```

Yearly Budget

Date	Description	Income	Expense	YTD
	Balance Forward	$805.66		$805.66
	Dean's office	$1,500.00		$2,305.66
	ME students	$600.00		$2,905.66
	CES Contribution	$1,000.00		$3,905.66
	ME Event Participation	$200.00		$4,105.66
7/15/19	Posters		$108.00	$3,997.66
7/24/19	Home Depot		$36.12	$3,961.54
9/16/19	Lowes - PVC pipe & Fittings		$35.52	$3,926.02
10/17/19	Amazon - LiPo Batteries		$217.98	$3,708.04
10/23/19	Lowes - Plywood/hardware		$73.84	$3,634.20
10/21/19	Amazon		$175.46	$3,458.74
10/21/19	Amazon		$17.78	$3,440.96
11/7/19	Mouser Electronics - slip ring		$27.94	$3,413.02
11/15/19	Blue Robotics - Locator/integration kit		$1,072.00	$2,341.02
10/7/19	Amazon -battery, pool noodles, adapter		$210.33	$2,130.69
2/14/20	Blue Robotics - battery		$303.04	$1,827.65
1/2/20	Blue Robotics - battery		$303.04	$1,524.61
4/20/20	Amazon - laptop, battery, charger		$957.00	$567.61

References

[1] https://bluerobotics.com/store/rov/bluerov2-upgrade-kits/brov2-heavy-retrofit-r1-rp/

[2]https://flex.flinders.edu.au/file/27aa0064-9de2-441c-8a17-655405d5fc2e/1/ThesisWu2018.pdf

[3]https://www.fossen.biz/wiley/ed2/Ch2.pdf

[4]https://waterlinked.com/underwater-gps/#tab-id-7