An Autonomous Underwater Lionfish Harvester

Lionfish Phase III

A Robotic Solution to
Eliminating the Invasive Species
In the Caribbean Ocean

Michael Abadjiev, Qingyuan Chen, Clark Ewen,
Nicholas Johnson, Nicholas Olgado, Harrison Saperstein,
Orion Strickland, Christopher Whimpenny

A Major Qualifying Project
Worcester Polytechnic Institute
Worcester, Massachusetts

An Autonomous Underwater Lionfish Harvester

A Major Qualifying Project

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree in Bachelor of Science

By:

Michael Abadjiev (RBE/ME)

Leo Chen (RBE/ECE)

Clark Ewen (ECE)

Nicholas Johnson (RBE/CS)

Nicholas Olgado (ECE)

Harrison Saperstein (RBE)

Orion Strickland (RBE/ME)

Christopher Whimpenny (CS/ME)

Date: May 18th, 2020

Approved:

Prof. Craig Putnam, Advisor

Bradley Miller, Advisor

Prof. William Michalson, PhD, Advisor

Abstract

Native to the Indo-Pacific region, Lionfish are an invasive species in the Caribbean Sea introduced by aquarium owners releasing their pets. The lionfish population continues to grow and threaten marine life in the region. This MQP proposes a fully autonomous robot to kill and capture the invasive species. Building off of previous teams' work, this year's focus is on developing a new harvester, refined identification algorithm, and a navigation algorithm to be integrated into an off-the-shelf ROV, creating a self-contained lionfish capturing robot. The team was able to test three identification models, develop a new harvesting system, and navigate in the WPI swimming pool, laying solid groundwork for future teams to bring this project to fruition with a fully functional robot.

Acknowledgements

First and foremost, we would like to thank our advisors for guiding us through the project and providing insight for our work:

Professor Craig Putnam *Professor Brad Miller* *Professor Bill Michalson*

We would like to extend a special thanks to *Professor Stephen Bitar* for his help on electrical circuit designs.

Finally, we would like to thank the WPI recreation building staff and lifeguards for allowing us to test our robot in the WPI pool.

Executive Summary

This document details the design and creation of an Autonomous Underwater Vehicle (AUV) for the purpose of addressing the problem of lionfish as an invasive species in the Caribbean. Native to the Indo-Pacific region, lionfish first started appearing in the Caribbean in the late 20th century, likely as a result of aquarium owners releasing their pets. Their voracious appetite and rapid reproduction are wreaking havoc on the delicate balance of the ecosystems of local reefs. This problem is being addressed in a variety of ways, from "Lionfish derbies", where divers compete to catch the most Lionfish, to specialized traps.

The goal of this AUV is to tackle the lionfish problem with a novel approach - a robot that can be deployed safely in the waters of the Caribbean. This robot should be able to navigate fully autonomously, so as to remove the need for a tether for remote control, as is used by other, similar, robots. It should have moderate storage capacity on-board for captured lionfish to be collected and brought back to the surface, to be sold for human consumption, as lionfish are quite tasty. In order to increase safety around reefs frequented by SCUBA (Self Contained Underwater Breathing Apparatus) divers, the AUV's vision system should, in addition to detecting lionfish, also positively identify and avoid divers.

For this MQP, the AUV system was split into four sub-systems, to be divided among the 8 members of the team. The navigation, identification, harvesting, and communications systems must work in harmony to accomplish the robot's mission. In order to tackle the first challenge, that of even swimming underwater, an off-the-shelf remote operated vehicle (ROV) developed by BlueRobotics was selected to be used as a base platform onto which all the systems were integrated. In addition to the major subsystems, the AUV consists of a myriad of other, more minor but equally important, components like leak detectors and battery voltage monitors. Another advantage to starting with an off-the-shelf ROV was that many of these aspects were either already addressed, or easily integrated.

The navigation system's primary sensors are two ultrasonic rangefinders, especially designed for underwater use by BlueRobotics, that are used to determine the location of objects around the AUV. With one sensor pointed forwards, and one downwards, the robot is able to avoid obstacles while maintaining a constant distance from the sea floor. For this year's MQP, it was decided to assume a relatively clean, sandy bottom, with intermittent obstructions, like that found in the deeper parts of the Caribbean. Navigating the complex terrain of an underwater coral reef is an extension of the project left for future development. The robot was able to successfully simulate navigating the seafloor during testing in WPI's pool, where the AUV was able to avoid the walls, as well as a swimming person as an obstacle.

Identifying lionfish was accomplished by training a machine learning algorithm on a large dataset, containing images of both lionfish and SCUBA divers. The dataset consists of images extracted from videos of lionfish and taken by divers in the Caribbean, where this AUV will be operating. Multiple machine learning pre-trained algorithms were trained and then tested using these images, in order to determine which would be most successful in identifying both lionfish and divers. Accurately identifying divers is nearly as important as lionfish for this robot, as humans should be avoided in order to prevent accidental harm. Some success was achieved by all of these models, with the most accurate one being MobileNet V2, with an accuracy of 75%. Future work should include collecting additional images for the dataset, thereby improving accuracy.

The harvesting mechanism developed this year consists of a pneumatically powered spear that rides in a specialized gantry that causes the spear to flip 180 degrees during its travel. This

allows for a fish to be speared in front of the robot, then rotated around and inserted into a container behind the robot, and to accomplish the motion with a single DC motor. The pneumatic system replaces a spring- or elastic-powered spear, which reduced the space needed for the spear, and greatly reduced the energy needed to be exerted while underwater. Rather than requiring another motor for some sort of cocking mechanism, potential energy is stored in the form of compressed air at the surface, and then released as needed. This system was built and tested successfully on the surface, but further testing was halted due to the COVID-19 pandemic. In order to store captured fish, a specialized lionfish containment unit was created for the back of the robot. The opening on this bucket has been adapted with a "one-way" valve for fish. When fish are inserted, the tines on the lip of the bucket prevent them from coming back out, thereby allowing the spear to slide out and leave the fish inside.

In order to maintain some communications between the robot and mothership, it was deemed necessary to develop a wireless communications module, to be used to transmit basic information and commands. This is complicated by the underwater environment, which renders radio waves useless beyond a very short distance. Instead, it was decided to develop a system based on high frequency acoustic waves. This functions similarly to radio, but uses sound waves, which propagate more freely in water. A major drawback to this approach, however, is the vastly reduced bandwidth. For this reason, the acoustic communications are limited to basic status messages, as well as a call to return home sent from the mothership to the AUV, to be used in the event of emergencies.

This year's MQP was successful in beginning the creation of an AUV that is capable of addressing the complex problem of hunting lionfish in the Caribbean. The robot developed was able to perform basic navigation functions, and identify lionfish with some accuracy. Unfortunately, due to the COVID-19 pandemic, much of the testing originally planned for the spring of 2020 was not accomplished, however the platform developed this year should provide a great starting point for future teams. The next steps involve expanding the navigational capacity, improving the identification accuracy, some refining of the harvesting mechanism, and achieving two-way underwater communications using acoustic modems.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Native to the Indo-Pacific Ocean, lionfish (*Pterois volitans* and *Pterois miles*) have become an invasive species in the Caribbean Sea and East Coast of the United States. With no natural predators in these foreign waters, lionfish have multiplied comfortably around coral reefs, which serve as a refuge in non-native locations. Lionfish primarily feed on herbivores like crustaceans and small fish – species that limit the growth of harmful algae and help sustain the health of coral reefs (Gupta, 2009). For the past fifteen years, lionfish overpopulation has induced stress on reef habitats and fish populations in the region. Studies have shown that even a single lionfish living in a coral reef can reduce native reef fishes significantly ("Impacts of Invasive Lionfish," 2020).

FIGURE 1: LIONFISH OFF OF THE COAST OF FLORIDA

Currently, the issue is being recognized by the United States and Caribbean countries, where mitigation strategies to eliminate the invasive species have been in production. The National Oceanic and Atmospheric Association (NOAA) and state agencies, primarily in Florida, Louisiana, and Texas, have removed barriers and created incentives for recreational divers to hunt and collect lionfish. The U.S., Jamaica, and other countries have also promoted the consumption of lionfish, selling expensively in local markets ("Lionfish Derbies," n.d.). Dive sites have been the primary hunting spots, but lionfish have been spotted carpeting sandy bottom seafloors at depths of over fifty meters and in other remote locations where divers don't typically explore (Whitfield et al., 2002). These factors provide an environment for an unmanned robotics solution to diminish the lionfish population in areas that aren't commonly surveyed by humans.

WPI has been addressing the lionfish overpopulation problem over the past few years and a robotics solution has been sought to be designed. Prototypes of tethered vehicles and harvesting mechanisms have been created in different stages, attempting to prove the concept of a lionfish capturing robot. This Major Qualifying Project (MQP) was tasked with building upon previous work to design and create an autonomous underwater vehicle (AUV), capable of searching for and spearing lionfish to be collected and eradicated in vulnerable areas in the western Atlantic Ocean. Utilizing the Blue Robotics BlueROV2 underwater submarine, a functional platform was created with the integration of an onboard autonomous system to execute lionfish hunting missions. As coral reefs continue to be threatened by other elements like global warming and ocean acidification, this MQP can help alleviate one of the more manageable problems that continues to threaten vulnerable marine habitats.

Chapter 2: Background

2.1 The Problem

2.1.1 Lionfish

Lionfish, specifically *Pterois volitans* and *Pterois miles*, are two species of fish indigenous to the warm waters of the Indo-Pacific region. These beautiful creatures can be identified by their maroon and white stripes that run vertically along their bodies and the spanning, venomous spines ("What is a lionfish?," n.d.). They can grow up to half of a meter in length and eat other marine life equivalent up to ⅔ of its body size (Lowe, 2016).Through a puncture wound the venomous spines distribute a mixture of neuromuscular toxin, protein, and the neurotransmitter, acetylcholine. These venomous spines make capturing and handling the fish difficult since the toxin causes severe pain along with other swelling and side effects. ("Lionfish Biology: Lionfish Discovery Story," 2020)

2.1.2 As an Invasive Species

Lionfish were first discovered along the Mid-Atlantic Coast in the mid-1980s. With a lack of predators in these foreign waters, the lionfish population has flourished due to a female's ability to lay upwards of two million eggs per year ("What is a lionfish?," n.d.). The introduction of this invasive species to the U.S. coastline is most likely due to aquarium owners releasing their lionfish into the sea. They have been spotted as far north as Cape Cod, Massachusetts and as far south as Jamaica. The invaders have escalated ecological problems, eating herbivores that maintain the health of coral reefs and reducing the food supply of other species in the area. A study conducted at Oregon State University concluded that the presence of lionfish in infested regions has reduced native reef fishes by 79% ("Impacts of Invasive Lionfish," 2020). Reefs are already being stressed from the effects of climate change and pollution, and lionfish are only compounding it. Lionfish overpopulation has also led to economic troubles for local communities that rely on fishing indigenous species.

FIGURE 2: LIONFISH SWALLOW ANYTHING THAT FITS IN THEIR MOUTH AND EAT IT WHOLE

2.2 Current Solutions

Since lionfish have become problematic, environmental organizations, governments, and marine biologists have been searching for viable and effective solutions to eradicate lionfish in the Caribbean and East Coast of the United States. The invasion has become so prevalent that the Florida Fish and Wildlife Conservation Commission holds an annual lionfish summit to discuss the current impacts, trends, and solutions to eliminate the species ("2018 Lionfish Summit Report," 2018). So far, three core eradication methods have been introduced and have gained significant traction: hunting and trapping lionfish, lionfish capturing competitions (referred to as lionfish

derbies), and promoting human consumption. Other research is being conducted on bio-controls for lionfish, such as the possibility of breeding large-bodied Caribbean Groupers, which have been found to occasionally prey on lionfish (Mumby, Harborne, & Brumbaugh, 2011).

2.2.1 Hunting and Trapping Lionfish

The most prevalent method to deal with the lionfish problem is hunting and trapping. Many Caribbean islands and the U.S. have legalized the spearfishing and capturing of lionfish without permits (Spencer, 2014). SCUBA divers are free to survey for, kill, and capture the fish with limited regulations. Many dive schools even teach this skill, eventually leading recreational dives where students can spear lionfish seen along the way. Even dive store owners are encouraged to teach their customers about lionfish and how to reduce the populations of the invasive species (Spencer, 2014).

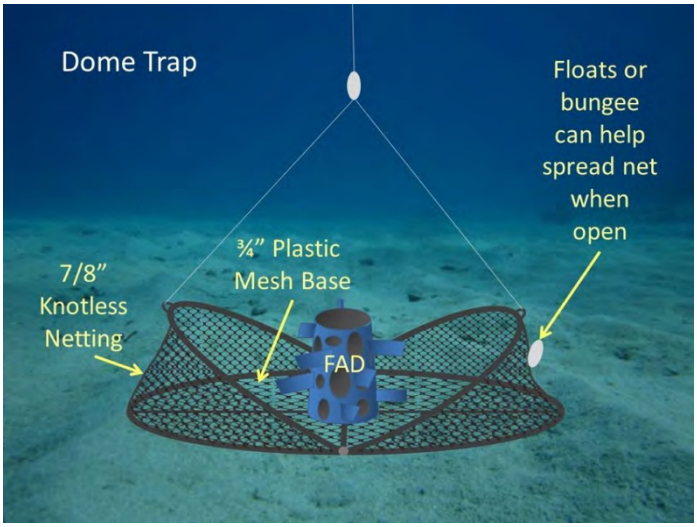


FIGURE 3: DIAGRAM OF A PARTIALLY CLOSED LIONFISH DOME TRAP

Organizations have experimented with a multitude of standing lionfish traps. Dome traps and purse traps, seen in Figure 3: Diagram of a Partially Closed Lionfish Dome Trap, are nets dropped to the sea floor, with a variation of fish attraction devices (FADs) set in the center to draw lionfish within the catching radius. These FADs are typically made to look like coral reefs or rocks, where lionfish often seek refuge (Gittings et al., n.d.). Since lionfish can swim out freely until the net is closed, these traps are inefficient.

2.2.2 Lionfish Derbies

Organizations like the Reef Environmental Education Foundation (REEF) have gone a step further to coordinate derbies, where divers can compete to catch the most lionfish in a given time. These are typically organized in areas that have seen population increases, often catching thousands of lionfish per year in significantly impacted regions. Lionfish derbies are also intended to educate the general population about the impact of lionfish and the problems they create ("Lionfish Derbies," n.d.).

2.2.3 Eating Lionfish

At the lionfish derbies, the caught fish don't go to waste. Volunteers fillet and cook the fish, providing a variety of experimental samples and dishes for the public to try ("Lionfish Derbies," n.d.). Many experts believe that the fastest way to decrease the lionfish population is for the public to develop a taste for them. Government efforts in the U.S. and Caribbean islands, such as Jamaica, are attempting to establish a market for the fish by training fishermen how to catch and prepare them. Many chefs and restaurants have also been incorporating lionfish into their menus (Spencer, 2014). In 2016, the grocery chain Whole Foods even introduced lionfish to the

consumer market at around $10 per pound (Smith, 2016). Efforts to eradicate the invasive species are being made, but the uncontrolled breeding is still creating a problem.

2.2.4 A Solution to Similar Problems on the Great Barrier Reef

A similar invasive species problem is occurring on the Great Barrier Reef (GBR). Crown-of-thorns starfish (COTS) eat away at coral reefs in the Indo-Pacific and have been a major part of the coral decline in the region. A Queensland University of Technology professor recognized that a robot can solve this issue and developed an autonomous navigation and vision system to do so, called RangerBot. After identifying a COTS with cameras (at almost 100% accuracy), the robot can kill it by injecting it with a toxic bile (Braun, 2018). RangerBot proves the feasibility of an AUV to kill an invasive species in environmentally sensitive areas such as coral reefs.

2.2.5 Work Done by Previous Major Qualifying Projects

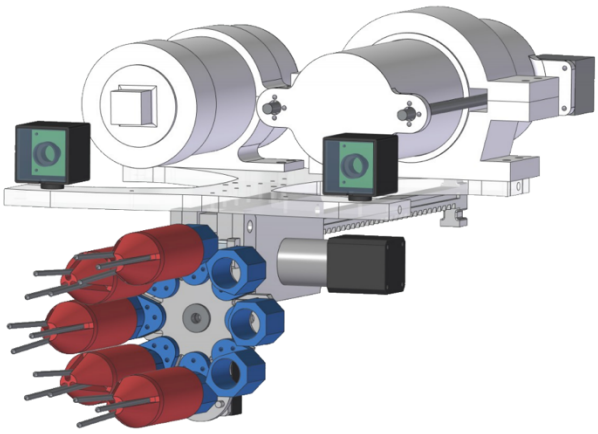


FIGURE 4: RENDERING OF PRIOR MQP HARVESTING ATTACHMENT

The first generation of the lionfish robot consisted of a lionfish hunting and capture system to eventually be attached to an AUV. The mechanism was to work completely independently of the robot, integrated with its own vision and lionfish detection system. The group was able to create a device that did not use a containment system, but rather pierced the lionfish with a buoyant spearhead that floated it to the surface to be collected. For the identification system, the group created a neural network of vision systems to determine if a lionfish was in the image (Yuzvik, Kelly, Lombardi, Uvarov, & Godsey, 2018). One of the initial concerns with this design, however, is the disposable spears. The robot loses weight and buoyancy after each spear discharge, which can disrupt the calibration and stability of the underwater vehicle. It is also limited to the number of available spears per trip (in this case, 8). Another concern is that the mechanism can contribute to ocean pollution if the spear is lost or never collected at the surface. A three-dimensional model of this device can be seen in Figure 4.

The second generation of the lionfish robot worked in conjunction with Robots in Service of the Environment (RSE) to further develop the computer vision system and harvesting mechanism for their non-lethal robot. This remotely operated robot uses an electrical shock to stun lionfish, store them, and bring them to the surface, rather than killing them. More specifically, the group created an object detection model to identify lionfish, recommended a new configuration for the electrical panels that stun the lionfish, and redesigned the lionfish intake system. The computer vision system used a ‘ssdlite mobilenet v2’ model, which can be a baseline for the identification system that is being integrated into our robot solution (Antaya, Peterson, Conroy, & Ralph, 2019).

2.3 Ethics of Autonomous Killing

As robots have grown over the years in both capability and complexity so have the ethical concerns surrounding their use. This is especially true for our Lionfish MQP, given that the purpose of the robot is to autonomously kill lionfish. Since the robot's main purpose is to take life, even though it is not human life, there are still a multitude of ethical concerns. Some of these include: is it moral to have a "soulless", mindless robot take life? What if the machine is repurposed for a malicious intent? What if the robot mis-identifies its target?

WPI offers the course RBE 3100: Social Implications of Robotics; in this course students learn about many of the ethical situations involving the robotics industry. One of the students on this year's MQP team took the course during the early stages of the project. As a result, his final paper for the class was based on the use of lethal force in the robotics industry. In this paper he proposed a general questionnaire to help people evaluate the ethics regarding their situation. The questionnaire part of this paper along with some general instruction is included in Appendix F.

2.4 The Platform for Autonomous Hunting

Autonomous lionfish hunting and harvesting pose several unique challenges that will have to be addressed by this project. These include considerations for the harvesting mechanism, as well as navigating underwater and successfully identifying lionfish, and being able to communicate with the topside boat.

2.4.1 BlueRobotics BLUE2 ROV

BlueROV2 is a commercially available Remote-Operated Vehicle (ROV) designed for inspection and research by BlueRobotics. It is an affordable ROV with many features for educational purposes. The BlueROV2 comes with six thrusters making it nimble and easily maneuverable underwater. It is equipped with many sensors such as a gyroscope, pressure and depth sensors, and leak detection. In addition, BlueRobotics offers an optional upgrade of a heavy lift configuration which we utilized. This configuration increases the thruster count to 8 and has an additional payload skid for customized equipment. The entire platform is open-source, with free to use three dimensional models and software ("BlueROV2," n.d.). These factors make it an ideal base system for building a specialized AUV on it.

2.5 Current Lionfish Harvesting Methods

The most common method for hunting lionfish by SCUBA divers is with a spring-loaded spear. These can take the form of either a harpoon, a pole spear, or some other similar mechanism. These projectiles allow divers to remain at a safe distance from the fish's venomous spines and the high stored potential energy results in an easy killing blow. For an autonomous killing solution, it is important to consider the difficulty present in drawing back and cocking such mechanisms. This requires a powerful motor and presents a high electrical load on a system with limited battery capacity. An alternative we looked in to was using a pneumatic cylinder to project the spear forward. This presents its own difficulties, as the high ambient pressures at the operating depths of this robot render traditional pneumatics systems useless.

The harvesting mechanism on this robot collects multiple fish in one dive, as to not necessitate excessive resurfacing. Many solutions exist for storing captured lionfish, but the most

prevalent, commercially available option is the ZooKeeper™ lionfish containment unit (LCU). The figure below shows an example of the ZooKeeper™ LCU14, which holds approximately fourteen pounds of lionfish, or roughly ten fish ("ZooKeeper LCU14," n.d.). The ZooKeeper™ essentially resembles a PVC pipe with a cap on one end, and a specialized "one-way valve" on the other side. This funnel shape allows a diver to push lionfish into the ZooKeeper™ on the end of a spear, and then extract the spear, trapping the fish inside. This project required a reliable containment device, and the ZooKeeper™ is the prime example we used for inspiration.

FIGURE 5: A ZOOKEEPER™ CONTAINER AND A SPECIALIZED ONE-WAY VALVE

2.5.1 Environmental Considerations for Harvesting

The deep underwater environment in which this robot is intended to operate provides its own unique set of challenges that were addressed by this project. The first consideration with these conditions is pressure. With an intended operating depth of 100 meters (300 feet), the resulting pressure on the robot is at least ten times the pressure at the surface. This means that easily compressible materials will not be useable. In addition, all watertight seals must be able to withstand these immense pressures, which will require careful manufacturing and pressure testing of all sealed chambers.

In addition to pressure, an important material consideration was corrosion resistance. Due to the nature of saltwater, any exposed parts are susceptible to corrosion. Corrosion, unlike rust, which is unique to ferrous metals, is the wearing away of metal as a result of a chemical reaction (Aluminum Handrail Direct, 2018). For this reason, plastics were mainly used instead of most metals. Stainless steel provided a solution for situations in which plastic would not be strong enough, but it is much more expensive. Other metals, like aluminum, were also considered, but would require a surface coating, such as a powder coating, to prevent major damage due to corrosion.

Some other minor considerations included water resistance, temperature, and buoyancy. The AUV needs to operate in an aquatic environment, and any components that are susceptible to water damage should be waterproofed and rated for the expected depths. Due to the large expected depths, very low temperatures were also to be expected. Materials with high thermal sensitivity, that may expand and contract significantly due to this temperature change were avoided where

dimensions are critical. Buoyancy control was considered important, as neutral or slightly positive buoyancy is generally desirable for underwater maneuverability.

2.6 Navigation

2.6.1 The Navigation Environment

The navigation environments that the AUV operates in are sandy bottom seafloors and seagrass beds. Current dive sites, especially along reefs, are typically patrolled by spearfishermen and divers with incentives to kill lionfish. Younger lionfish have been observed in shallow waters along these deep reefs and fringing reefs which are typical hunting sites where lionfish are being managed more effectively. As they get older, and as waters warm in the Atlantic, lionfish have been able to descend deeper to unreachable environments (for recreational divers), such as seafloor habitats. This provides an environment for the AUV to patrol at 50-100 meters, where sport divers aren't present. This is where our AUV is primarily most applicable and effective, especially off the coast of Florida where lionfish are seen carpeting areas of the seafloor. Underwater localization and environment mapping are also challenging tasks and are underdeveloped for three-dimensional structures like deep coral reefs.

FIGURE 6: LIONFISH 75 METERS DEEP ON THE WEST FLORIDA SHELF SEAFLOOR

2.6.2 Navigating the Environment

When navigating the environment there were two main considerations: where the AUV is and where its surroundings are. Once an AUV knows where objects are relative to itself, it can easily navigate around them. This, however, proved to be difficult and often involves error correction and cross-checking sensors. It also must know where it is relative to its launch point so it can stay within a specified area and not get lost from its operators. This was also an issue since it will not always be within the line of sight of the boat. To tackle these challenges, navigation looked at two subjects, localization & mapping, and object detection.

2.6.3 Localization and Mapping

Global Positioning System (GPS) does not function underwater. This is due to the extreme dampening of electromagnetic signals in water (Waterson, n.d.). Because of this, determining where an AUV is becomes much more difficult, but there are a few different methods that have been developed to solve this issue. One of the most common for land-based autonomous vehicles is Simultaneous Localization and Mapping (SLAM). The idea behind this technique is to use sensors around the robot to identify features, thus creating a map of the robot's surroundings in memory. At the same time, the vehicle identifies these features and cross-references them with the map it is making to identify its location within the map as seen in Figure 7.

Underwater, however, this technique runs into many different challenges, making it one of the more complex problems modern robotics is trying to solve. It becomes much harder once underwater to determine the absolute location of both the robot and its surroundings due to turbidity. This quickly leads to unreliable mapping and localization. There is often also a lack of key features to help the AUV identify where it is, and this problem is compounded with a sandy ocean floor. SLAM also becomes computationally intensive underwater, since the scale of its environment is much larger than a single room or building (Guth, Silveira, Botelho, Drews, & Ballester, 2014).

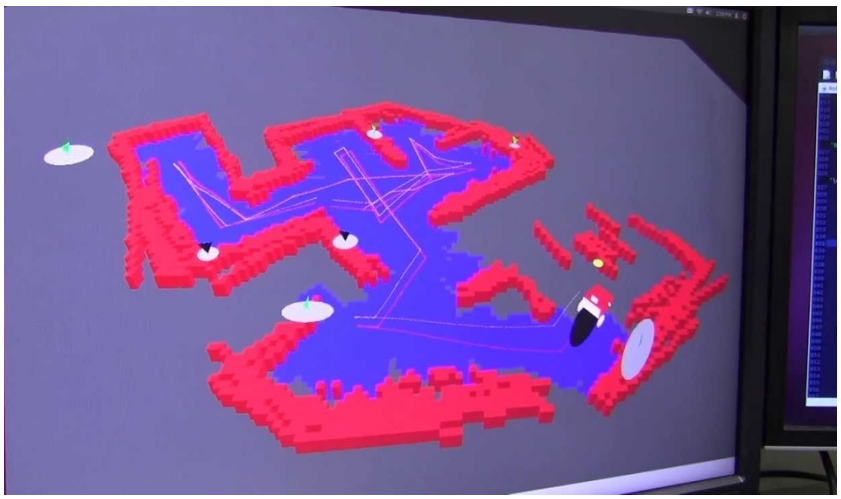


FIGURE 7: SLAM REPRESENTATION OF A ROBOT NAVIGATING A MAZE

One of the commonly used algorithms for underwater SLAM is the Extended Kalman Filter (EKF). The EKF produces an estimate of the state of the system by averaging all the predictions that the system has made and the incoming feed of the new data from sensors. It uses a weighted average that selects the most relevant data. EKF's are ideal for systems that are continuously changing or moving. Implementations typically have short processing times and don't need much memory since the system only needs to know the previous state. Many studies have also looked into an Augmented EKF (AEKF) for underwater purposes. This approach adds an "augmentation stage" to the prediction which improves overall map management and reduces the amount of error in the map. Additionally, the AUV can achieve more precise localization and mapping of objects with lower processing times (Kang, Oh, & An, 2010).

A second technique that is often used is acoustic localization. This works similar to how GPS does on land. Using known locations of sound emitters pinging at the same time, an AUV can calculate the difference in times between the pings to determine its location relative to the emitters. The Defense Advanced Research Projects Agency (DARPA) has been working on a project called POSYDON (Positioning System for Deep Ocean Navigation) which explores this concept on a much larger level (Waterson, n.d.). The main problems with this technique include the potential difference in the speed of sound through water. Depending on salinity, temperature, thermoclines, and turbulence, the speed of sound in water can vary significantly. This may lead to incorrect localization of the AUV. Also, the bouncing of sound waves off nearby objects may also lead to either interference or incorrect localization (Akyildiz, Pompili, & Melodia, 2005).

The final commonly used technique is sensor dependent systems. Mainly, these systems use a barometer to determine depth as well as an inertial measurement unit (IMU) to determine the motion of the AUV. While this system is the least dependent on the environment around the robot, it still has some technical problems. One major issue is the tendency for the sensors to drift over time. With no outside intervention, the AUV has no way to correct for these drifts, and thus the problem grows exponentially as time goes on, leading to completely incorrect localization.

2.6.4 Object Detection

Another important part of localizing and navigating the robot is detecting nearby objects. There are two main types of sensors that can be used for this purpose, cameras and acoustic sensors, often referred to as sonar. Cameras are very often used above the surface for navigation, however they become less reliable under the surface of the water. A lack of visibility can cripple a camera underwater, ruining its detection ability. In addition to this, as an AUV descends further into the ocean, the short wavelengths of light are filtered out and the overall environment becomes darker. This often hurts the camera's ability to reliably identify objects, especially at greater depths. Another consideration with cameras is that at least two are needed to detect the distance and size of objects. Using a stereoscopic system doubles the number of sensors for object detection and also increases the computational power needed to detect objects.

There are also many things to consider when using sonar. For one, most easily available sonar sensors only detect objects in a one-dimensional arc. While planar sonar systems are available they cost multiple thousands of dollars. Due to this, sonar systems need to be combined to get rough object detection in two dimensions, or only used when the risk for objects outside of the arc is low (seafloor location). Also, sonar sensors can be inaccurate due to turbidity in the water, making detection confidence more difficult (Guth et al., 2014).

A final type of sensor that was considered for object detection was a touch sensor. Similar to how the Apollo moon landers signaled that they were on the surface, a touch sensor can be used on an AUV to detect when it is near an object or the seafloor (Rogers, n.d.). The major downside to this is that the AUV must physically interact with its surroundings to detect an object. This can potentially be harmful to both itself and the environment it is interacting with. There is also no way to tell where an object may be until it is reached. Due to these considerations, touch sensors are not commonly used and should only be employed for very specific purposes.

2.6.5 Communications with the Surface

Because our AUV is in full control of a powerful hunting apparatus, there must be a way to manually intervene with the AUV in the case of an emergency. Given that the AUV operates outside of the operator’s line of sight, bi-directional wireless communication must be established for the operator to know the status of the AUV and intervene when needed. The most crucial information necessary for communication are emergency surfacing and disarming the harvesting system. Additional bandwidth can be used to relay less vital information, such as the battery status, depth, distance, or anything else. Also, these AUV status messages are very useful during development for debugging.

Acoustic modems are piezoelectric transducers which convert electrical signals into underwater sound signals (and vice versa). Acoustic modems are the best choice in most underwater situations; because using optical communication is very directional, and radio waves require very low frequencies (<300Hz) to travel at higher distances underwater (Stojanovic, 2003).

2.7 Identification

Another essential part of being fully autonomous is the ability to recognize a lionfish in front of the AUV and track its location. Machine learning algorithms and computer vision techniques allow the AUV to successfully detect and track any lionfish it may come across. While there are different types of machine learning, we focused primarily on supervised learning

techniques. In conjunction with machine learning, the AUV needs to calculate the position of the lionfish relative to itself in space so it can navigate toward the fish to aim the harvesting mechanism. This required a computer vision technique known as stereovision. Lastly, being able to process this data required an adequate processor.

2.7.1 Supervised Learning

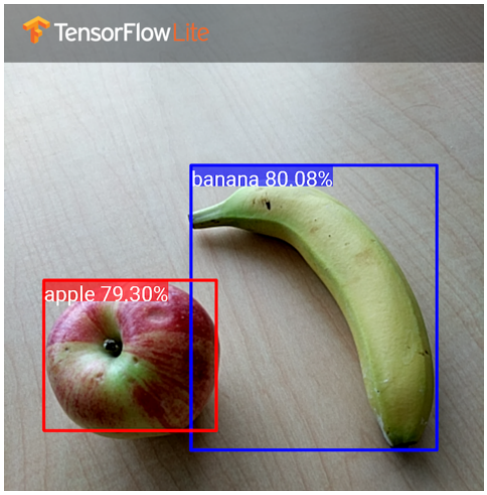
Two machine learning algorithms were considered to identify lionfish in front of the AUV. The first was image classification and the second was object detection. Image classification consists of making predictions on what class the image represents. An example might be a picture of a dog that is compared to a rabbit or hamster in Figure 8 ("Image classification | TensorFlow Lite," n.d.).

Animal type	Probability
Rabbit	0.07
Hamster	0.02
Dog	0.91

FIGURE 8: EXAMPLE OF IMAGE CLASSIFICATION WITH TENSORFLOW

When an image is passed through the image classification model, the model outputs a prediction of what 'image class' it thinks the image is a part of (i.e. dog, rabbit, hamster). This comes in the form of a probability from 0 to 1, where the sum of all the probabilities respective to each class the model knows about adds up to 1.

Object detection goes one step further than image classification. While image classification only classifies the image, object detection is able to detect where in the frame the objects are. The model output now contains both a confidence score in its guess and a bounding box around the object, shown in Figure 9. Unlike image classification, object detection can identify more than one object in an image each with their own confidence scores from 0 to 1.



Class	Score	Location
Apple	0.92	[18, 21, 57, 63]
Banana	0.88	[100, 30, 180, 150]
Strawberry	0.87	[7, 82, 89, 163]
Banana	0.23	[42, 66, 57, 83]
Apple	0.11	[6, 42, 31, 58]

FIGURE 9: EXAMPLE OF OBJECT DETECTION MODEL OUTPUT

2.7.2 Training

To train a neural network, a sizable dataset with variation is needed. This includes images in different orientations, lighting, and distance from the camera. The dataset is then randomly split into two sets, the training set, and the test set. This split can be decided by the user but a common split is 80% training and 20% test. The model never sees the test data until the model has been fully trained and is only used for determining the final score, otherwise the model will be contaminated from its "peeking" at the test set.

In the training data, input and output data is provided. The input is provided to the model for it to predict based on its current interior weights. The prediction output is then compared to the known correct output that was provided. If the prediction does not match the expected value that error is propagated back through the network to update the internal weights of the network.

The rate and effectiveness of the learning of the network is based on epochs, batch size, and the learning rate. Epochs are the number of times that the full training dataset is passed through the model and updates the internal weights. Too many or too few epochs will produce problems such as overfitting and underfitting (Sharma, 2017). Batch size is the number of images to pass into the network before the internal weights are updated. Larger batch sizes will train faster as the network's weights are updated less often, but too large of batch sizes can produce less accurate models (Shen, 2018). The learning rate is how large of an adjustment will be made to the network weights with respect to the loss gradient. A smaller learning rate will take longer to converge to the minimum, but too large of a learning rate will overshoot the minimum and can fail to converge (Zulkifli, 2018). All of these hyperparameters need to be tweaked for a successful training of the network.

Once the model has been trained, validation is needed to make sure that it has the desired accuracy and variance. There are two methods to accomplish this, train-test split, and K-folds cross-validation. The Train-Test split takes the training dataset and splits it randomly into two sets, train and validation. This split is usually 70:30 or 80:20 train-to-validation sets. This method limits how much data the model can be trained on as a significant amount is withheld. Problems such as bias can occur if the dataset is not extremely large.

The second method is K-Folds Cross Validation. K-Folds Cross Validation takes the full training dataset and divides it evenly into K-folds. The model is then fitted with K-1 folds and the remaining fold is used for the test. This process, as seen in Figure 10, repeats until all folds have been used as the validation set. All scores from each of the fittings are then averaged to provide a metric for how well the model performed. Doing this division allows for all of the data to be used for both training and validation which overcomes the limitations, such as bias from small datasets, of the Train-Test split (Krishni, 2018).

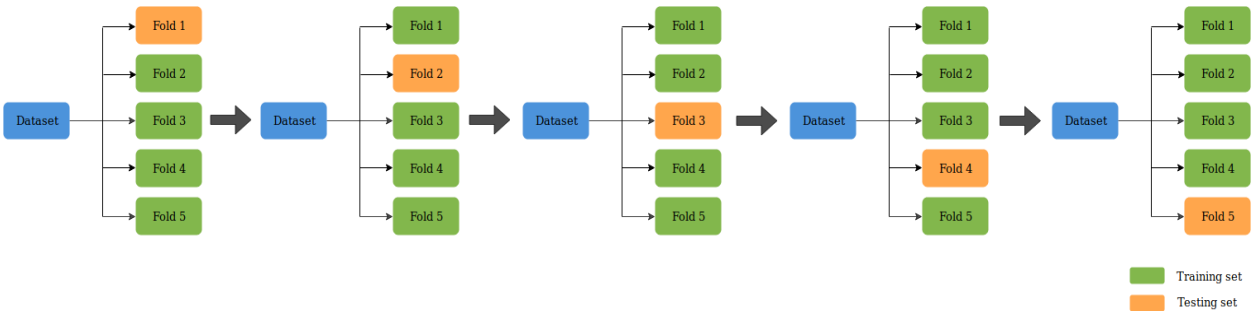


FIGURE 10: MEDIUM K-FOLD CROSS VALIDATION SYSTEM

2.7.3 Model Comparison

Once a model is trained, it needs to be evaluated and compared to other training attempts with different data and hyperparameters or other model architectures. These metrics are generated from testing the model on another separate set of data that was not used for training otherwise the score that is produced is useless as the model "peeked" at the data used for scoring.

There are two metrics of the model that will be important to the robot, speed and accuracy. To determine the speed of the model, the images in the testing set which the model has never seen before will be run through the model and timed to produce an average Frames per second (FPS). This allows for a comparison of the average framerates between the models. The second metric is accuracy, which will be measured by the percentage of the frames that the robot performed the correct action, if a Lionfish was present it indicated that, and if a Diver was present it indicated that. It will also be important to track the number of False Positives the model identified. A False Positive is where the model indicates it found a Lionfish or Diver where none existed. The speed and accuracy results are important metrics to compare the models to get the best model for the desired application.

2.7.4 Stereovision

Another component for identifying lionfish is stereovision. Stereopsis, or stereoscopic depth perception, is how our eyes and brain perceive the distance of objects in front of us. A stereo camera system does the same. The system consists of two separate camera sensors that are offset from each other by a known constant. The images from the camera sensors can be seen as two different perspectives of the same image. Then based on the perspective shift in each image, we can impose them together and create three-dimensional depth in the image. Depth of objects in the frames can then be calculated. This will be useful for the AUV in order to properly line up the harvesting mechanism with the lionfish.

2.7.5 Hardware for Running Neural Networks

For running the neural networks, there were many possibilities for hardware options. Given the AUV's size constraints, the following were considered as the central processing unit on the robot.

Raspberry Pi 4 and Google Coral USB Accelerator

The Raspberry Pi 4 comes equipped with a Quad-core ARM A72 chip @1.5 Ghz. It has two USB 3.0 ports and two USB 2.0 ports. Running a neural network strictly on the processor alone would be very slow, but in this configuration, it would be paired with a Google Coral USB Accelerator. The Google Coral USB accelerator is specifically designed to perform accelerated ML inferencing (Coral, n.d.). The combination of the Raspberry Pi 4 and the Coral USB accelerator would consume three amperes at five volts. For optimal operating temperature the Coral is designed to operate below 40°C and the Raspberry Pi 4 is designed to operate at 50°C or below. A limitation of the setup is the Coral USB Accelerator is only designed to run on TensorFlow Lite and cannot run other neural network software or be a general compute device. The cost of the Raspberry Pi 4 ranges from $35 for 1 GB of RAM to $55 for 4 GB of RAM and the Coral USB Accelerator cost $74.99. This option for hardware would cost $109.99 - $129.99 ("Raspberry Pi 4 Model B Specifications," n.d.).

Nvidia Jetson Nano

The Nvidia Jetson Nano is one of Nvidia's smaller and cheaper boards designed to run neural networks. The Jetson Nano comes equipped with a Quad-core ARM A57 chip at 1.43 GHz and 4 GB of RAM. It is equipped with four USB 3.0 ports. To run it needs to be supplied with at least five volts at four amperes to run stressful workloads in its ten-watt power mode. A huge advantage of the Jetson Nano is its ability to run a wide variety of neural networks, such as TensorFlow, TensorFlow Lite, PyTorch, TensorRT, and it can be a general compute device. The cost of a Jetson Nano is $99 (NVIDIA, n.d.).

2.7.6 Robot Frame Transformations

To properly communicate the directions from the Neural Network model to the navigation system, a frame transformation must occur. A frame transformation is changing the x,y,z (and perhaps rotation) of a given coordinate system into a different orientation. The coordinate frame of the Neural Network is not the same as the coordinate frame the robot navigation system is expecting, so a frame transformation is needed. Once the frame transformation is calculated those coordinates can then be passed on to the navigation system for a movement command.

Chapter 3: Methodology & System Design

The goal of the project was to augment the current solutions to help eliminate lionfish in the Caribbean Ocean. Since they are an invasive species with exiguous hunting regulations (divers are encouraged to hunt them), we believed an autonomous robotic solution is both moral and effective. The NOAA notes that developing an effective method to target lionfish at depths of greater than 50 meters and other remote locations is critically important in decelerating their population growth. With this in mind, we constructed the following three objectives to accomplish the project goal:

1. Navigate the lionfish's environment effectively and thoroughly while avoiding obstacles and fragile reef habitats.
2. Identify lionfish with greater than 90% accuracy and pursue them safely, without alarming them, and with the same intent as objective one.
3. Spear and harvest lionfish with a capacity of five fish minimum.

The group ultimately split into sub-teams each focused on an objective: navigation, identification, and harvesting. A fourth task was also defined: wireless communication, to design a system that can interface with a topside computer to monitor AUV status and take control if needed. Once these objectives were set, a flowchart was created to plan the AUV's mission and provide a comprehensive understanding of the AUV's operations, seen in Appendix A. To start, we discuss how the AUV accomplishes navigation.

3.1 Navigation

Before the AUV identifies, spears, and captures lionfish, it needs to be able to navigate its environment. To do this, it must identify obstacles and waypoints (such as lionfish), while being able to localize itself in relation to them and create paths of travel. With very limited light at depths greater than fifty meters, the AUV would be navigating through the water virtually blind. Instead of using a vision system, a sonar system was used to detect obstacles in the AUV's way and update the system when it was getting too close. We planned to have a mapping system where an algorithm initially determines the best course to take to a waypoint with the information that it is given. This was created in a simulation but was not physically tested in the pool. Obstacles can be mapped with a sonar scan from the boat before the mission, or as the robot moves to the waypoint, it identifies obstacles with the onboard sonar and updates the path. To localize itself, the AUV uses both the positions of obstacles and waypoints, as well as inertial measurement sensors (gyroscopes, accelerometers, magnetometers) to record its movement from a known starting position. Our plan for accomplishing these capabilities relied on the abilities of the sensors and algorithms. We were able to simulate the mapping and path planning, but did not test these on the physical robot. More rudimentary navigation like maneuvering and obstacle avoidance were tested underwater. This section discusses in more detail how the AUV achieved this and how it was tested.

3.1.1 Sensors

The most important feature of the AUV navigation is its ability to identify and locate obstacles in its path. Since it operates in sensitive and endangered habitats, the AUV needs to be

accurate when locating obstacles, simply because it cannot hit or damage anything. To execute these abilities, we implemented a few sensors into the system.

There are a few methods that are used for obstacle avoidance, but the most prevalent is sonar, which we used in our design. Sonar is most feasible since sound waves propagate farther through water than radar and light waves do. Another option we considered was the use of cameras to detect obstacles, however, it was out of the scope of the project. Luckily, BlueROV offers an open source Ping Sonar echo-sounder which integrates easily into the existing system and is rated within our constraints. The Ping sonar is a single-beam echo sounder with a range of 0.5 meters to thirty meters and a 30-degree beam angle. It returns the relative distance away from objects within its "line of sight" as well as a confidence level of these readings. The confidence level was used to filter out faulty or "unsure" obstacles so that the map can be more manageable. Using a Ping sonar device facing forward, the AUV is able to detect obstacles in its direct path for the high-level navigation system to make decisions with. Another Ping device faces downward to keep the AUV at a constant distance from the sea floor, between 0.5 and one meter. This is a desirable feature since it limits opportunities to run up on the seafloor or hit small, protruding plants or obstacles. Lionfish also typically stay close to the seafloor and projections like reefs and rocks, so keeping the AUV along the seafloor increases the chance of identifying and capturing the invasive species. These sensors interface with an Arduino Mega, where some processing and filtering takes place before relevant data is sent via USB to the navigation software on the Jetson Nano.

FIGURE 11: BLUEROBOTICS PING SONAR ECHO SOUNDER

As the AUV creates a map of the obstacles in its path, it also needs to localize itself within the map. To accomplish this task, the AUV uses the existing IMU system on the Pixhawk, the open-source hardware system used by BlueRobotics. The IMU tracks displacement from a starting point and the direction of the distance traveled. With a distance traveled and a heading, the AUV knows its location within the map and can store the path it took to get there. The heading measurements could also be used with obstacle mapping, as the sonar only returns a distance. Some challenges arose, however, when relying on an IMU to retrieve position. Accelerometers tend to lose position due to bias error and can throw off the calibration of the entire map. With an error bias of 50 milli-g's, for the accelerometer on the Pixhawk, the AUV can potentially lose its position by approximately one meter every twenty seconds. While this may not seem like a large error, it quickly propagates since this is roughly 0.5m/s^2 and due to the squaring factor in this term, it exponentially grows as time goes on. With the help of the acoustic localization, the hope was to reset the accelerometer position frequently to keep its reliability.

The last critical sensor is the depth sensor on the Pixhawk. A depth sensor uses pressure to determine how far under the water the sensor is, since water pressure is linearly correlated with increasing depths. This provide the AUV with a Z-axis position to locate itself in three dimensions. With all of these sensors, the navigation system was able to identify obstacles and maneuver in different ways. From this data, the AUV could also process travel paths to waypoints while avoiding obstacles.

3.1.2 Mapping

With sonar, the AUV can create a virtual three-dimensional map of its surroundings, similar to how whales and bats depict their environment. When creating the map, there were a few considerations when determining the resolution, including memory capacity and sonar limitations. We ultimately chose 0.5 meter resolution because it is both relatively similar to the size of our AUV and is the low end range for readable data on our sonar sensors. We planned on having two separate systems each with their own ping sensor. The first was be pointed towards the seafloor, allowing us to accurately navigate above it at a set distance. The second, faced forward allowing us to detect fish and obstacles.

Since the sonar sensors only gives the AUV a distance reading of the nearest obstacle somewhere within the sensing cone, we needed to mark the whole cone at that distance as an obstacle. To help negate this when we receive sensor data, we also clear data points within the cone. For example, if the AUV previously detected an object at five meters away, the whole cone must be marked since we don't know the exact location of the obstacle. If the obstacle was at one extreme of the cone, if the AUV rotates away from it and pings again, the two cones will overlap but the nearest obstacle will be further away. In this case the marked locations are cleared since we know that the current closest object is now a different distance away. The total map size consists of half meter data points, creating a map size of 400x400x200 half meters. In order to be able to accurately create and execute search patterns, the map must be maintained otherwise a large amount of drift can occur.

In early iterations of our AUV this will not be implemented, focusing on more immediate concerns such as not hitting nearby obstacles and being able to navigate a constant distance above the seafloor. This covers our full operating range of 100 meters to each side of the boat (operating constraint for our wireless communication range), and 100 meters in depth (BlueROV2 maximum operating depth). Currently, this will take up approximately 64kb of memory but can be reduced in the future as testing permits.

3.1.3 Path Planning

When the AUV descends directly downwards from the boat and reaches a meter off of the seafloor, the 3D location is set as the center of the map and is considered the starting point, thus allowing the AUV to always know its relative to the boat. An assumption must be made that the boat is in a fixed position. By taking in the mapping data of obstacles and waypoints, our path planning algorithm can decide the best course to take to any waypoint location that we specify. This can be broken down into two parts: the cost function and the actual path planning algorithm. The cost function essentially analyzes the quality of a path, where it weighs different criteria and gives the path a score, or cost. The path planning algorithm uses the cost function to minimize risk and ultimately chooses a path to take.

Cost Function

The cost function considers the following criteria when determining the quality of a proposed path:

1. The safety of the path. It will not plan paths that get closer than 1 meter to any object and punishes those that come within 2 meters.
2. The angular distance, or how much the AUV must turn from one intermediate waypoint to another. This helps to smooth out the path and discourages the robot from making 180 degree turns.
3. Distance from the halfway point. The function rewards setting waypoints close to halfway between the starting and final locations which can be coupled and iterated over to create a full waypoint path to the final destination.

The cost function produces a cost for a path quickly and efficiently, which is then utilized in the path planning algorithm. Path costs are relative to one another, but in theory, range from 1 to infinity depending on the complexity of the environment and path.

Planning Algorithm

We decided to use particle swarm optimization for our path planning algorithm since it is very flexible and also can quickly come to optimal solutions, especially in changing environments. The system works by creating a swarm of particles, each searching for the minimum cost. Each particle's velocity is determined by its previous inertia, its personal best, and the swarm's best. Each of these factors can be individually increased and decreased as we decide. Using the algorithm, we ran preliminary simulations in a random 2D map environment, where the algorithm created two planned paths shown in Figure 12. The faded, blue dots represent obstacles, and the green and yellow dots are waypoints from the path planning algorithm that navigate to the final destination. A red line has been laid over the waypoints to better illustrate the path of travel.

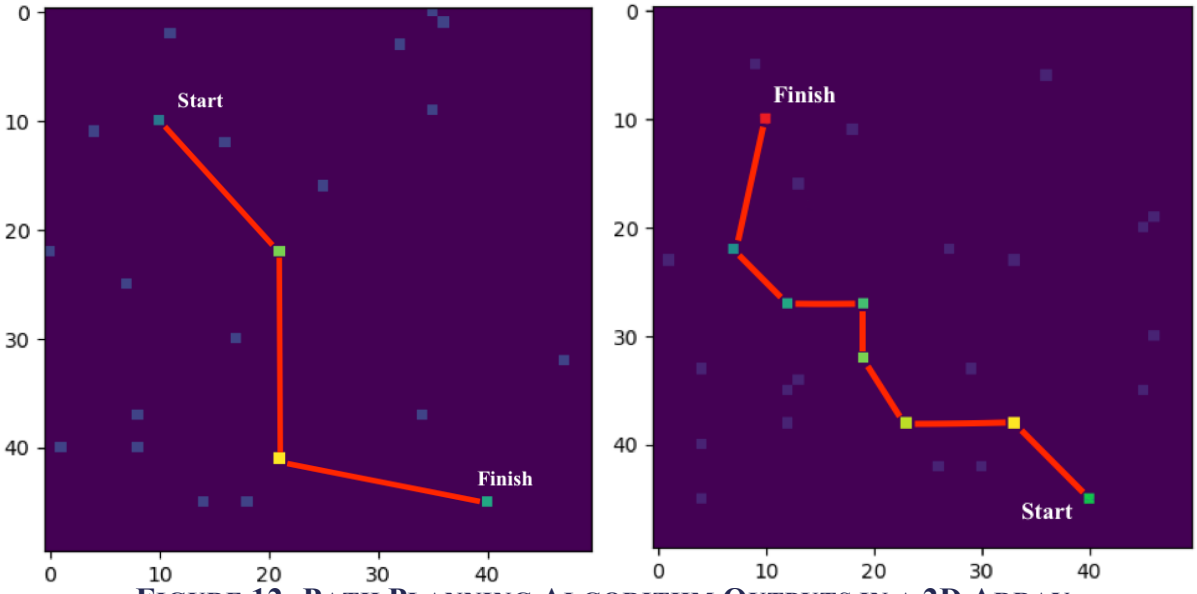


FIGURE 12: PATH PLANNING ALGORITHM OUTPUTS IN A 2D ARRAY

Once the path is planned, the path is translated into movement commands using the ArduSub protocol. Since all of this would be computed on the Nano, we would need to send messages via ethernet to the existing BlueROV system, essentially replacing the tether connection that currently exists.

3.1.4 Testing Navigation

The overall test of the navigation system was to be completed in an obstacle course laid out in the WPI pool. The obstacles would imitate rocks, large, protruding sea fans, and reef projections. We were also going to set waypoints in the pool to test different turns and directions, and change obstacle locations to test different scenarios. With limited time, we were able to carry out tests to see the operations of the robots. Much of the initial testing was looking at the different maneuvers and how to combine them with sensor inputs to make decisions.

The Building Blocks

Our process to begin testing navigation and ultimately implement the mapping and obstacle avoidance relied on the creation of what we called "navigation building blocks". These were essentially singular movements or functions that could have the possibility of being implemented into a state machine or in an ordered succession to build a more advanced navigation system. We were able to tap into the existing BlueROV2 platform and manipulate it to fit the lionfish harvester's needs. The existing system had many sensors and data feeds to subscribe to via the established Robot Operating System (ROS) framework, but for our initial purposes, we accessed the data from the magnetometer and depth sensor. In order to produce a functional overall system, we decided to create basic movements and functions, which are the building blocks to the navigation software:

- **Move Forwards**: The AUV holds its current depth and travels forward for a given time input and at a given throttle percentage.
- **Move Backwards:** The robot holds its current depth and travels backwards for a given time input and at a given throttle percentage.
- **Turn to an Angle:** The AUV remains stationary and at its current depth, and executes a pivot turn to a given relative angle (in degrees) and at a given throttle percentage. A negative angle input is a left turn and a positive angle input is a right turn. This function utilizes the data from the magnetometer.
- **Dive:** The AUV remains stationary and ascends or descends to a given depth at a given throttle percentage. A depth of 0 indicates the surface, while any negative value is the depth, in meters, that the robot will ascend or descend to. This function utilizes the data from the depth sensor.

We were also able to call commands that were already created for the robot by BlueRobotics. These controls were very important as they provided the robot with necessary stabilization techniques that allowed it to execute more precise movements. These were:

- **Stabilize Mode:** The AUV stabilizes roll, pitch, and yaw to level and will hold its orientation unless commanded to turn.

- **Depth Hold:** The AUV maintains its current depth as it travels in any horizontal direction. The algorithm controls the vertical motors to counter any changes in depth feedback given by the depth sensor. It also enables stabilize mode allowing for orientation to be held**.**

Both of these processes could be run simultaneously with any other commands or controls given. For example, a 'forward' command could be run alongside 'depth hold' to simply keep its current depth as the robot travels forward. The four building block movements, however, cannot be run simultaneously with one another. These controls were not extremely necessary during pool testing; however, in an ocean environment, stability and depth hold would be critical in maintaining control and precision. With all of these building block movements in place, we created a simple input process to call these movements directly from the command prompt of the connected computer via the tether. This enabled more feasible testing scenarios and allowed us to test, update the software, and iterate through our changes. Combined with sonar, these maneuvers were utilized for obstacle avoidance and search pattern execution that would efficiently explore ocean environments and identify lionfish.

3.2 Identification

As the AUV is navigating its environment, it searches for lionfish using cameras. In order to do this, an identification software runs continuously on the camera frames to check if there are lionfish in its view. The goal of the identification subsystem is to not only target lionfish, but also identify divers to disarm the spear and steer clear of them. The first step of identification was selecting the appropriate hardware that was capable of processing the data. After the hardware was selected, many images of lionfish, divers, and the ocean floor were collected and labeled. The next step was to train and validate machine learning models for comparison and compatibility with.

3.2.1 Hardware Selection

Hardware needed to be selected to be used as the main component in processing the machine learning algorithm, as well as other system tasks. This hardware must be quite powerful to run object detection on two cameras, but also power efficient as it is being run off of batteries. The two options laid out in the background section were the Raspberry Pi with a Google Coral accelerator, and a NVIDIA Jetson Nano.

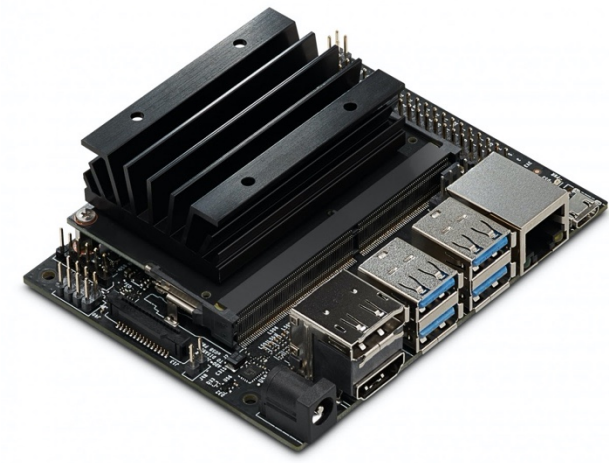


FIGURE 13: NVIDIA JETSON NANO

Comparing the two options, the Raspberry Pi has a slightly lower power draw at fifteen watts than the Jetson at twenty watts. For the clock speed of the processors, they are very similar with the Raspberry Pi having a 1.5 GHz and the Jetson having a 1.43 GHz clock. A differentiating feature of the Jetson is its ability to run more than just TensorFlow Lite that the Raspberry Pi is limited to. Due to the addition of more batteries in the design, the power difference wasn't an issue and the ability to run more than just TensorFlow Lite was the reason we have decided to use the Nvidia Jetson Nano (Figure 13) for our onboard computer.

The cameras used for the video feeds also has to be selected. This was a simple selection, as we already had multiple Low-Light HD USB Cameras from BlueRobotics. The cameras are 1080p resolution with a 64^{o} vertical field of view and an 80^{o} horizontal field of view.

3.2.2 Data Collection

After the processing hardware was selected, we collected images from a multitude of locations. A significant dataset was from the Lionfish Phase 2 team who previously collected many images. Images were also pulled from Google images, as well as personal photos team members and Advisors have collected on diving expeditions. Finally, image frames were extracted from YouTube videos of people diving in the Caribbean and Florida. All of these sources added to the collection of our dataset that were needed to train the model and ultimately detect lionfish and divers. In the end our dataset consisted of over 17,000 images, which contained over 14,000 instances of lionfish and over 5,000 instances of divers.

3.2.3 Data Labeling

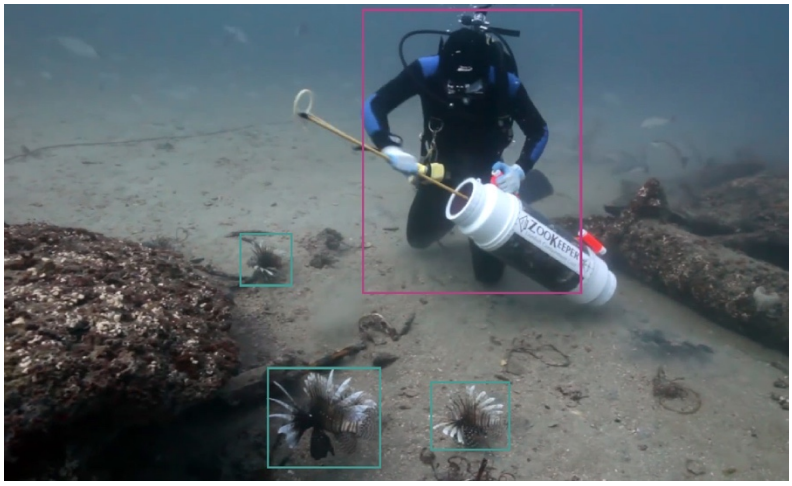


FIGURE 14: LABELED IMAGE OF DIVER AND 3 LIONFISH

Once the images were collected, they needed to be labeled so that the model could be trained. The model needs to be told what is present in the frame as well as where in the frame an object is located. To accomplish this, a time-consuming manual image labeling process was done. A webservice called *Supervise.ly* was used for this labeling process. The website allows for a dataset to be uploaded and then divided as jobs for team members to label the images. For an image, a team member classified it as one with a lionfish and/or a diver, and drew a box around the object with a corresponding tag. If there was neither a lionfish or diver the image was marked as a negative. This process made it easy to divide the images, label them and export them to be preprocessed prior to model training with the data that included what objects are in each image and where in the images they are. An example of the boxing of objects in an image can be seen in Figure 14.

3.2.4 Data Pre-Processing

After all the images were labeled, they were downloaded as PNG and JSON files. The PNG file was the image, while the JSON file included the data about the image objects, such as the bounding boxes of the objects and what objects they were. These JSON files were then parsed and the data written into a single CSV file. The CSV file allowed us to streamline all the information prior to converting it to a format that was more useful to the TensorFlow training pipeline. This format is called a *tfrecord*. Simply, this record stores all the information of all the image files in a singular file after serializing the data. The data is serialized using *tf.example*. A *tf.example* first

allows us to create the serialized string of data pertaining to the image. The specific data includes image height, image width, filename, image format, image encoding, a single object's class, and bounding box coordinates. A new string could then be serialized and written to the *tfrecord* for each individual object that was labeled.

3.2.5 Initial Pre-Trained Model Selection

Once the images were selected and appropriately labeled, training a machine learning model to identify lionfish and divers was the next process. TensorFlow1 was the selected framework for the training and deploying of our model. While TensorFlow2 was considered, it excluded the object detection API that was required for model training, which TensorFlow1 did have support for.

We decided to build off of a pre-trained model, rather than build one from scratch. Building off of a pre-trained model allowed us to utilize the existing structure and use the pre-existing weights to easily alter for the new classes. There are many pre-trained models available for use as can be seen in Appendix B. Three models were chosen which included SSD-Mobilenet v2, SSD-Inception v2, and SSD-Resnet. The reason for choosing these models was single shot detection (SSD), diversity, and a balance of speed and accuracy. SSD is necessary since it detects multiple different objects in a single pass of an image through the model. This is efficient at detection and can identify both a lionfish and diver at the same time. Diversity in model types is also necessary as each is built slightly differently, where we will determine which type of model is best in our environment. Lastly, speed and accuracy (mAP) were balanced accordingly to find the best tradeoff between the two. The models we have chosen fit those criteria the best in our situation and are shown in Table 1.

TABLE 1: OUR THREE CHOSEN PRE-TRAINED MODELS

MODEL NAME	SPEED (MS)	COCO: MAP[1]	OUTPUTS
SSD_MOBILENET_V2_COCO	31	22	Boxes
SSD_INCEPTION_V2_COCO	42	24	Boxes
SSD_RESNET_50_FPN_COCO	76	35	Boxes

1 MSCOCO evaluation protocol - http://cocodataset.org/#detection-eval

3.2.6 Model Training

Once the pre-trained models were selected and the training data had been converted to the *tfrecord* format, training could begin. Google Cloud Servers were used to run the training process due to their support for TensorFlow, as well as the compute power available. To run on Google Cloud, a local environment with all the necessary packages had to first be set up. The process of getting all the dependencies correct took some effort and trial and error. In this process it was determined that the object detection library that was the basis of our training model had been removed from TensorFlow 2. In addition to this only select versions of TensorFlow were supported on the Google Cloud. These issues were remedied by reverting back to TensorFlow 1.15 for training, and then running on TensorFlow 2 with the supplemental object detection package. Once the environment was configured with all the necessary packages for training, the pre-trained model files, and the pipeline config file were all uploaded to a Google Cloud bucket to be run. While the models were training, progress was tracked using TensorBoard. All three model types were trained until TensorBoard indicated that accuracy on the validation set was no longer increasing.

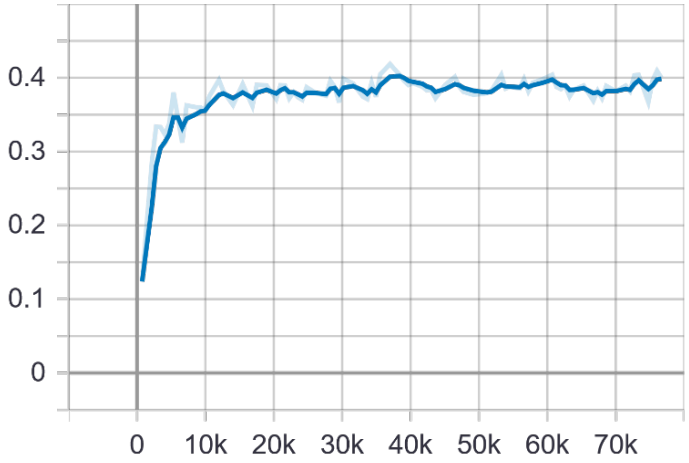


FIGURE 15: TRAINING DETECTION BOXES RECALL

3.2.7 Model Transformation

After retraining the models, they needed to be converted to a format that was optimized for running on smaller devices such as the Nvidia Jetson and Raspberry Pi. Two formats were considered. The first was TensorRT, this optimizes the model to utilize Nvidia CUDA cores. This method is specific for the Nvidia Jetson and could not be run on the Raspberry Pi. The other format was TensorFlow Lite (TFLite). TFLite was also developed by TensorFlow to be used in mobile computing, such as our AUV. The TFLite converter is a provided tool that allows for the conversion from the original TensorFlow model to the TFLite model. Using the CLI commands, the input shape, format and arrays, as well as output format and arrays, among several other arguments, the model could be converted. Once the model was converted, we can test the model to determine if the conversion had been done properly. Testing the new model consists of running inference on frames of a video and analyzing the output of the model's inference. The output of the model consists of 10 object bounding boxes, 10 confidence scores and 10 class labels. By looking at this output we will be able to determine if the TFLite model is still accurately detecting our Lionfish and Diver objects.

3.2.8 Stereo Vision

Once a lionfish has been detected by the neural network, targeting data is needed for the robot to accomplish its mission. The location in the image frame (x,y) can provide a two-

dimensional location, but to accurately approach the fish for a kill, the depth in the z direction is needed. Using two cameras for stereo vision, a depth map can be achieved using OpenCV.

The first step in the depth map process is to get calibration data for the cameras. To calibrate the cameras, a known object with known dimensions needs to be captured by both cameras to determine their relative distortion. A checkered pattern with known dimensions, as seen in Figure 16, was held up in front of both cameras and a Python script captured the images and processed them.

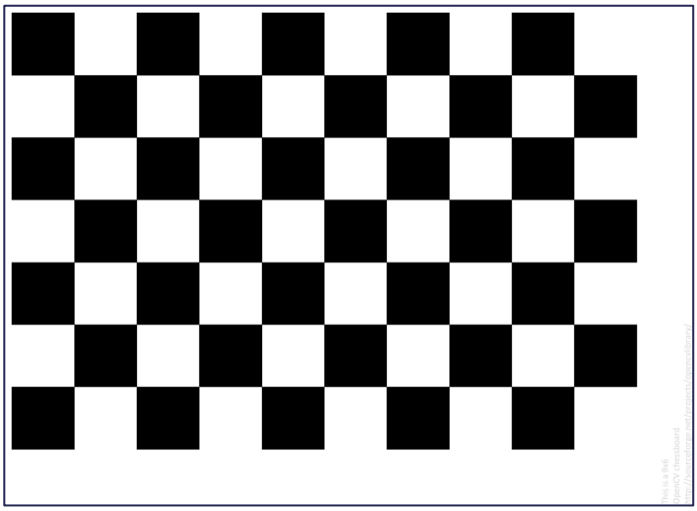


FIGURE 16: CHECKERED CALIBRATION IMAGE

Once the captured images had been processed for the distortion maps, the cameras were ready for stereo use. For stereo vision, OpenCV provides *StereoBM_create* and *StereoSGBM_create*. For our use, *StereoSGBM* provided better results. For cleaning up the depth map that *StereoSGBM* provides, there are many parameters that needed to be tweaked. To work on understanding the best values for these parameters a test bench was set up in the lab, the parameter were tweaked, and an initial set of settings were found. The next step after that was to move to the pool and gather underwater calibration data and repeat these steps for an underwater depth map.

3.2.9 Communicating the Frame Transformations

After the transformations are computed the data needs to be sent to the navigation process to maneuver to the lionfish. The relative coordinates of the Lionfish to the main robot frame will be sent to the navigation software, where it will be set as a waypoint that the AUV can plan a course to in the same process as described above. This waypoint will be continuously updated with new information as the AUV moves toward the Lionfish. At the same time, the identification process will write a pin high on the Arduino Mega to arm the spear to prepare it for a strike. The spear will move into the frame of the camera so that the lionfish can be aligned with it and the hunting can be captured on video. Once the AUV arrives within 0.5 meters of the lionfish, both sonar devices will be turned off since the sensors cannot return accurate data under this threshold. The stereo vision program will send movement commands for the AUV to maneuver itself until the lionfish is correctly oriented in the camera frame at appropriate distance from the spear. The identification process will then write another pin high on the Arduino Mega to actuate the pneumatic system to fire the spear and strike the lionfish.

3.3 Harvesting

Once the AUV is able to identify, navigate to, and align itself relative to a lionfish, it can capture and store the lionfish in a container. Since the goal of this harvester is to fatally injure lionfish, rather than attempt to capture them live, two killing methods were considered: a traditional spearing mechanism or an approach involving electrocution. The electrocution approach, as implemented on the RSE robot, was dismissed early on because of its focus on

capturing live fish. With the decision to use a spear-based approach, two design concepts evolved and were compared: the arc spear and the spin-spear.

3.3.1 Arc Spear Design

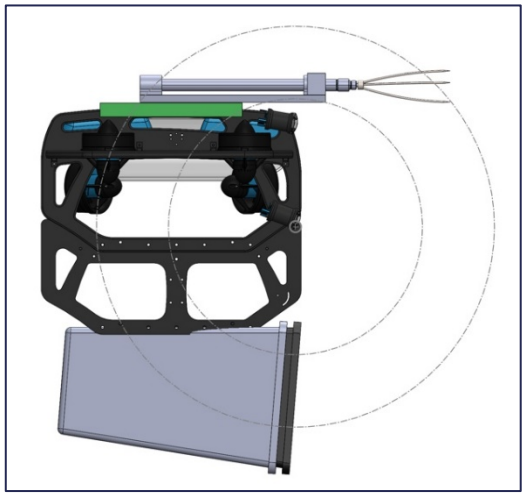


FIGURE 17: ARC-SPEAR DESIGN

The first harvester conceptual idea was based on the motion divers make when filling a container with lionfish. Usually, divers have a special container with a one-way valve that allows a spear and lionfish to go in and clears the lionfish off of the spear when it is removed. The motion of a diver's arm typically travels in an arc while the spear travels tangentially to this path. This motion allows the containment unit and the spear to be far away from each other and still be effective. The way this would integrate with the robot is a rigid platform would hold both the spear, cameras and any other electronics together. This platform would then be mounted onto a metallic arm that would swing in an arc from the top to the bottom of the robot where the lionfish containment device would be mounted as seen in Figure 17. The method behind driving the arms could vary between a pneumatic actuator or driven by a motor with a gearbox. With the design shown in Figure 17, the containment unit would be on the bottom of the robot, allowing for a quick swap of containers.

3.3.2 Spin-Spear

The second spearing concept involves a spear mounted on a linear motion carriage. As the spear moves linearly, a rack and pinion gear cause the carriage to spin, flipping the spear over in order to insert it into a container situated behind the rails. Figure 18 shows the process of the actuation in six steps:

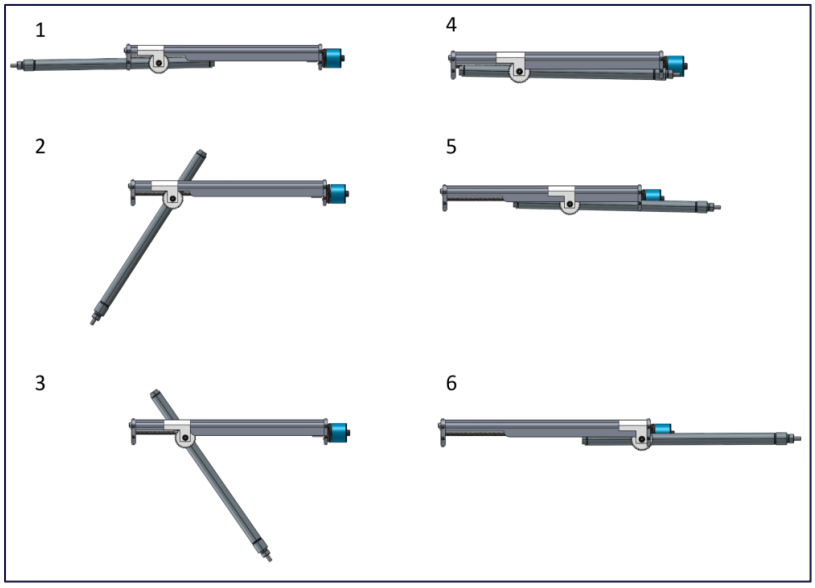


FIGURE 18: SPIN-SPEAR ACTUATION PROCESS

In order to drive this system, a simple lead screw with two linear guide rails will be employed. The linear rails with Teflon coated bushings, will provide a rigid support for the carriage, while the leadscrew, driven by an underwater brushless DC motor (the same as the thruster motors), will produce the linear motion. With this spearing mechanism mounted facing forward on the bottom of the AUV and a container behind it, the process of spearing a fish and storing it becomes a single fluid motion, minimizing the number of actuators and sensors needed.

3.3.3 Spear Design Comparison

In order to compare the two spearing mechanisms, a comparison matrix, shown in Table 2, was created to determine their respective strengths and weaknesses. Each of the metrics were scored with a value of 1-7.

TABLE 2: COMPARISON MATRIX TO DECIDE SPEAR MECHANISM DESIGN

METRIC	SPIN-SPEAR	ARC-SPEAR
SIZE (OPERATION ZONE)	7	3
WEIGHT	6	3
POWER REQUIREMENTS	5	4
COST (ROUGH ESTIMATE)	5	5
EASE OF RELOADING CONTAINER	5	7
FORM FACTOR (WHILE SWIMMING)	5	6
FREE SPACE (FOR COMPONENTS)	4	7
TOTAL SCORE:	**37**	**35**

Both of the spearing options presented different pros and cons. The arc spear scored particularly well for the free space and ease of reloading the lionfish container, but would be slightly heavier and occupies more space. In addition, the actuation required to execute the full arc would be more complicated than the linear slide used by the spin-spear. In the end, it was decided that the spin-spear was more suitable for this application.

3.3.1 Pneumatics

Both spearing designs utilized a pneumatic cylinder actuator to fire the spear. Traditional spearfishing weapons used by SCUBA divers involve elastic bands or spring-loaded mechanisms to fire the spear at a lionfish. The drawback to these systems is that they require a significant amount of extra space and power to operate. Harpoons need to be cocked back and held in place, which typically have larger frames and require extensive actuation. A pneumatic cylinder addresses both of these issues. When sitting idle, a pneumatic cylinder requires no energy to hold in place and by using a pre-charged compressed air system, the only power consumption comes from a solenoid valve to control the cylinder. Additionally, a double acting cylinder fits in a much smaller form factor than a spring-loaded mechanism with the same stroke.

3.3.5 Containment

After spearing the lionfish, the team thought it would be best to store the fish in a separate containment unit. This way the firing mechanism and camera feeds would be free of any obstructions caused by the fish carcass and it would help mitigate losses caused by fish floating away from the robot. There were a few designs, based on what lionfish hunters currently use, that the team evaluated. They were the spear-shaft, ZooKeeper ™, and catch bag.

The team evaluated these containment methods based on how well they would integrate with the existing platform and weighed that against the unknowns and complexities involved in using such methods such as free-floating carcasses, shark interferences, amorphous containers, drag, buoyancy, etc.

Spear-Shaft

Spear-shaft is a method of lionfish hunting in which the hunter uses a long metallic spear gun shaft to hold the lionfish, like a kebab. Usually, the shaft has a special tip to prevent the fish from sliding off the spear and a stopper mechanism to keep the fish's venomous spines away from the hunter's hand as seen in Figure 19. The benefit for hunting lionfish in this manner is the lack of drag caused by the spear and fish. Unlike other containment methods the spear-shaft has minimal surface area and weight making mechanical integration with our robot incredibly easy. However, this design does have flaws. One being that the fish are exposed and left on the spear. This leads to a variety of issues such as fish breaking off of the spear and floating away, fish bodies obstructing the view of the cameras, fish stuck in mechanical parts such as the thrusters, and attracting predators. This method also poses a high risk of injury to any human interacting with the robot since the venomous spines of the lionfish will be exposed. This

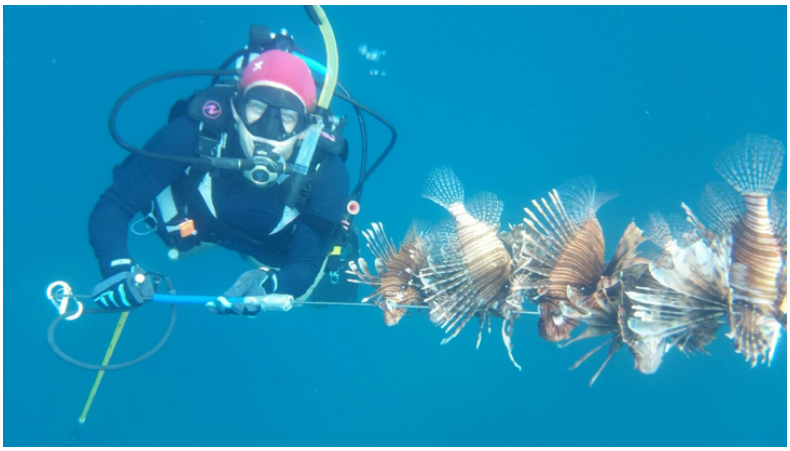


FIGURE 19: LIONFISH HUNTRESS USING A SPEAR-SHAFT

would hamper anyone trying to remove the spear-shaft from the robot making it difficult to swap spear-shafts and collect the fish.

Zookeeper™ LCU

The Zookeeper™ lionfish containment unit (LCU) is the most preferred method of hunting lionfish by divers. As seen in Figure 20, the LCU has a tube-like structure with a special one-way valve that prevents fish from escaping. These products are highly customizable and can be equipped with spear holders, quick surface floatation devices, etc. However, this method does have a lot of drag, but given the pros of the method such as diver safety, lionfish yield, and customizations, it is understandable that divers prefer using one or making a homemade version.

FIGURE 20: A ZOOKEEPER™ IN USE AND A HOMEMADE ONE

Catch Bag

Catch bags are a hybrid blend between keeping fish contained and mobility. The catch bags designed for lionfish are made out of a thick lining that prevent the lionfish's venomous spines from poking through and injuring anyone handling the product. Depending on the design of the catch bag most have a one-way valve at the opening that allows a diver to insert a spear with a lionfish on the end and when the spear is removed the lionfish will stay within the bag. The bag is not rigid and is free to drag behind and expand to hold more fish. This is favorable since it allows for a high lionfish yield, but the bag's semi-amorphous nature poses potential risks to getting caught in mechanical components or dragging along the seafloor, unless another guiding apparatus is designed.

FIGURE 21: CATCH BAG FOR LIONFISH CONTAINMENT

3.4 Communication

There are two main communication methods being utilized on the AUV. It is easiest to split them up into internal and external communications. Internal communications encompass the wired ethernet network that connects each of the onboard processing units like the Jetson Nano and Raspberry Pi. External communication considers the wireless connection made between the AUV and the user at the surface.

3.4.1 Internal Communications

The BlueROV2 has an existing internal static IP network configuration. As the system is relatively easy to extend, we utilized the base system and expanded it. We added a network switch to connect the three main processing boards - the existing Raspberry Pi, Jetson Nano, and external communication board which is discussed later. The switch also has an available port where we can connect the tether for testing and shadowing purposes. The flowchart below illustrates the communication flow between software and processing boards. The Jetson Nano is the main device, running both identification and navigation simultaneously. Identification sends three-dimensional locations of a lionfish or diver to the navigation process to be set as a waypoint or obstacle respectively in the map. Identification also sends the arm/disarm and fire message for the spear mechanism, which is controlled with a PWM signal from the Arduino Mega. The navigation software receives information from the Arduino Mega which includes sonar data to add obstacles to the map, as well as emergency messages like low battery and leak detection when necessary. The navigation process receives positioning data from the existing BlueROV system like location, depth, and heading. Once the navigation software determines a path to take from all of the data it has received, it then sends translated movement commands for the Pixhawk to execute and physically move the AUV. A flowchart of the communications is shown in Figure 22.

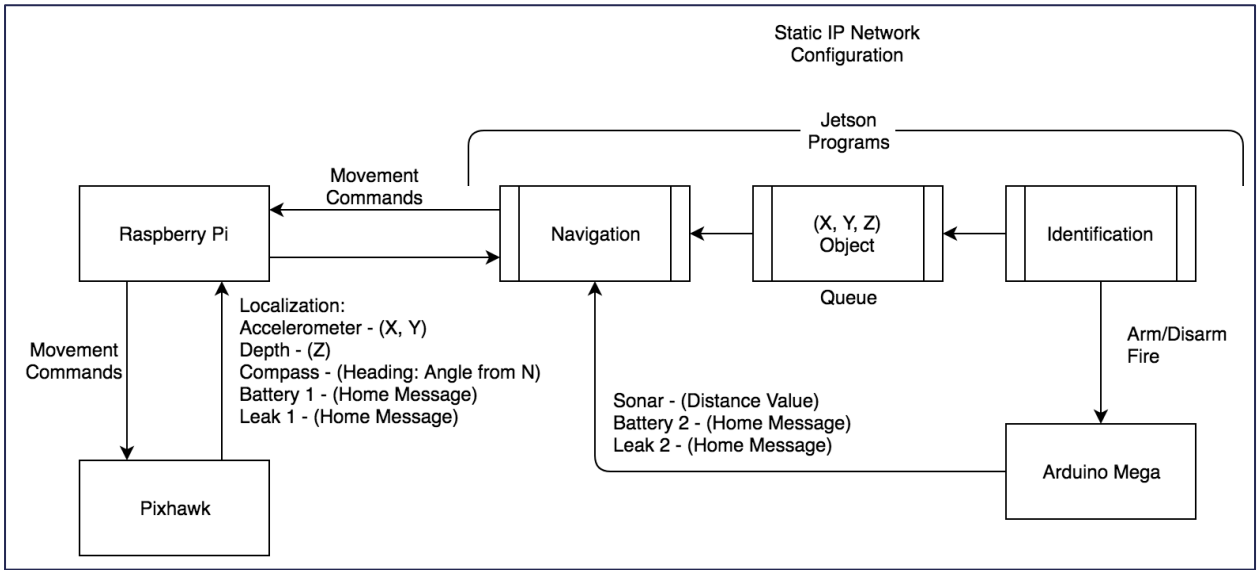


FIGURE 22: ONBOARD COMMUNICATION SYSTEM FLOWCHART

3.4.2 External Communications

Wireless communication is a supplementary, yet key feature for the AUV to be fully autonomous. To ensure the safety of nearby divers, the robot features user intervention to disarm the robot remotely. The AUV also has the ability to receive "abort mission commands" from a user at the surface, for example, when unexpected weather is approaching. Acoustic modems are typically the device of choice for underwater communications, since sound waves travel so well in this environment. The modems only need to relay and receive the AUV's telemetry and commands from the boat. Due to the high price of acoustic modems (most are used in marine and military applications), the team decided to design a low bandwidth, inexpensive version from scratch.

Sound Level

Acoustic transducers use piezoelectric properties to convert between acoustic waves and electric signals as the piezoelectric material can both be excited from the driving circuit, or from the acoustic pressure. This means only two transducers are needed to establish a communication between the boat and the AUV.

When designing wireless communication systems, there are few important characteristics that affect the abilities of an acoustic transducer: transmitting power, receiving sensitivity, and frequency. Generally, the following equation is used to describe the needed transmitting power (SL, or source level) in decibels at one microPascal of sound pressure to communicate, related to the signal to noise ratio (SNR), transmission loss (TL), and noise level (NL).

$$\boldsymbol{SNR = SL - TL - NL}$$

EQUATION 1

In order to find the source level needed, the other three variables need to be calculated. The SNR is related to the capabilities of the receiver side to actually capture the transmission. The overall transmission loss for frequency range from 3kHz - 500kHz was calculated by the equation below. This equation considers the many factors that affect transmission loss including attenuation and spreading loss (Akyildiz et al., 2005).

$$\beta = 8.68 \cdot 10^3 \left(\frac{SAf_T f^2}{f_T^2 + f^2} + \frac{Bf^2}{f_T} \right) (1 - 6.54 \cdot 10^{-4} P) \text{ [dB/km]}$$

EQUATION 2

Where:

$$f_T = 21.9 \cdot 10^{6 - 1520/(T + 273)}$$

EQUATION 3

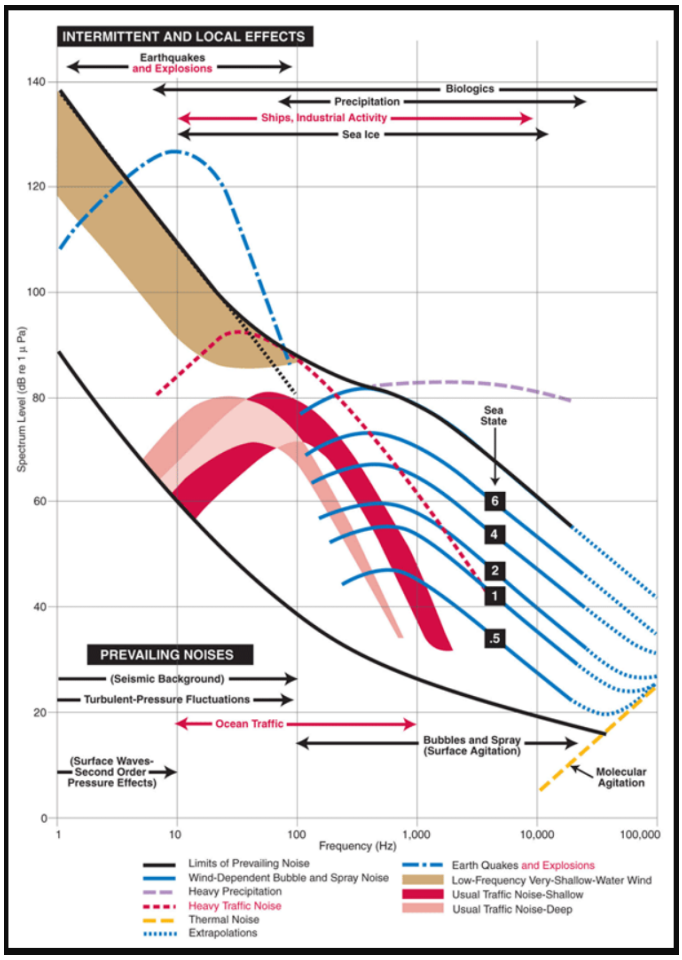


FIGURE 23: TYPICAL NOISE FREQUENCIES IN OCEAN ENVIRONMENTS

The noise level is an estimated or measured value based on previous external research. Noise level is the background noise present in the ocean at different frequencies, caused from wind and waves, marine animals, ship navigation systems, and more. Figure 23 is a ~~conclusive~~ plot of noise in the ocean, with the most extensive amount coming from wind at the surface, which is below sixty decibels and in the frequency range that the acoustic modem will be communicating at (Benson et al., 2010).

The acoustic transducer needed for communication requires a hemispherical beam since the robot could be at any orientation. There are very few transducers available on the market, and their prices could easily exceed our budget. Other options for cheap transducers are potting our own transducer or use cheap but narrow beam fish finders. We decided to use a fish finder for the purpose of testing and developing a communication system. Before actual deployment, a different transducer could be fitted for better performance.

Transducer Driver

With the source level known, the transducer could be potted accordingly. Driving the underwater transducer, however, is the same as driving a high impedance speaker. The driver voltage was approximately 200 volts peak-to-peak, enough to account for the transmission loss and reach a desired signal to noise ratioFigure 24. Vsig+ and Vsig- were two square waves with a 180-degree phase shift to drive the two N-channel MOSFET's. The MOSFET's created an alternating current on the transducer, which was essentially the communication signal. With a 1:10 ratio winding on the transformer, the resulting voltage is ten times higher on the transducer side, reaching the 200 volts peak-to-peak requirement.

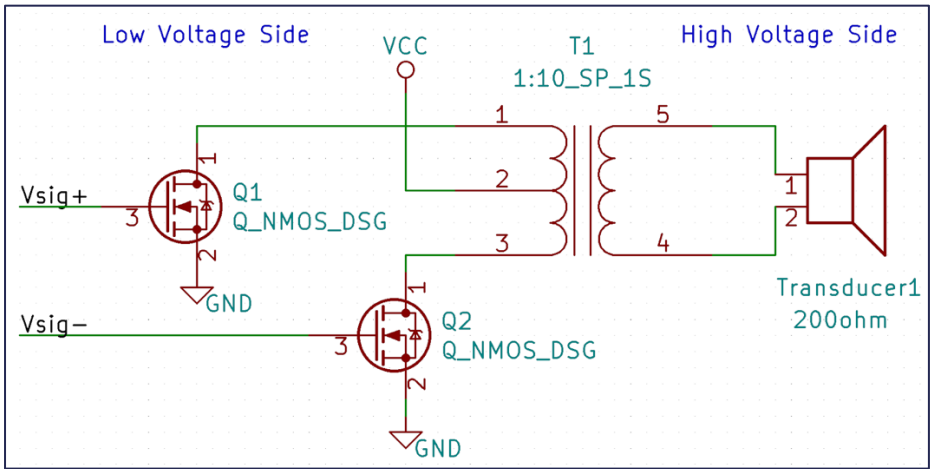


FIGURE 24: TRANSDUCER DRIVER CIRCUIT

The IRF540 MOSFET is commonly available at a low price and is the device we used to drive the transducer. It meets the power requirement of eighteen volts for input battery voltage, has an on-resistance of 0.077Ω, and has a power loss 0.5 watts at two amperes. Although the gate-source threshold voltage for the IRF540 is only four volts, most modern microcontrollers output 3.3 volts instead of five volts, making it hard to directly drive the IRF540. Since the IRF540 can handle up to twenty volts, driving it with the BlueROV2 battery voltage will have less on-resistance and energy loss. A gate driver chip was also considered as a source to easily drive the IRF540 with the battery voltage. Compared to a discrete gate driver design, a dedicated chip has better specifications, fewer parts, and less development time. In addition, the cost of a dedicated gate driver is negligible. Thus, the dedicated gate driver TC427 was used instead of discrete components.

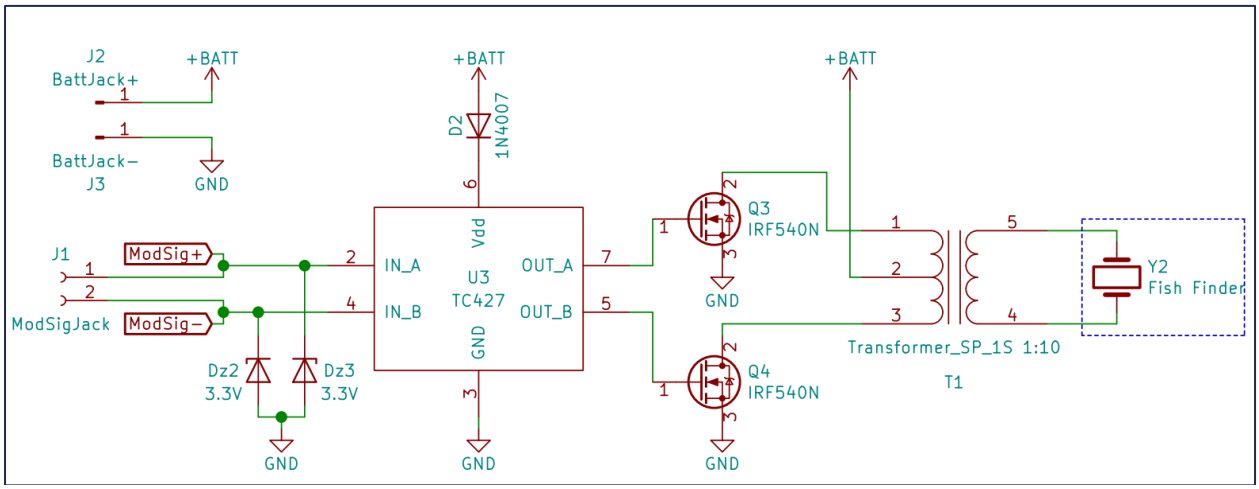


FIGURE 25: TRANSDUCER DRIVER CIRCUIT WITH INTEGRATED GATE DRIVER

The initial design of the transducer driver is shown in Figure 25. The modulated signal drives the IRF540 through the gate driver, producing a high-power signal on the transformer of a fish finder for testing. The 3.3 volt Zener diodes (Dz2 & Dz3) are used for protecting the microcontroller's input pins.

Originally, the fish finder was driving its transducer at 200kHz and therefore is the carrier frequency for the transducer driver. Our first testing results are seen in Figure 26, where the voltage on the primary and secondary side are shown when driving the circuit with five volts and 200kHz. The waveform is highly deformed and contains a lot of high frequency peaks at around 7.5MHz. When the load is changed to a 10kΩ resistor (**FIGURE 27**), the same high frequency peaks still exist, but is a much cleaner waveform than in Figure 26. This is likely due to the result of impedance mismatching, which we worked to resolve next.

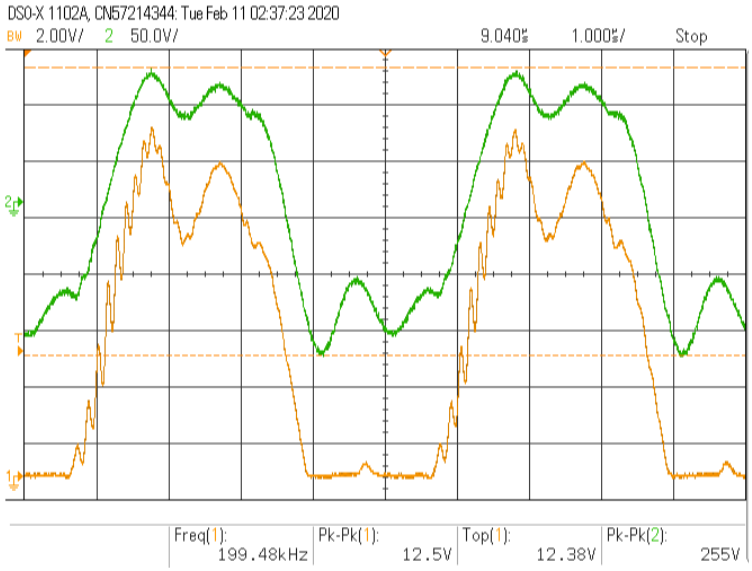


FIGURE 26: DRIVING TRANSDUCER AT 200KHZ, PRIMARY SIDE (YELLOW/BOTTOM), SECONDARY SIDE (GREEN/TOP)

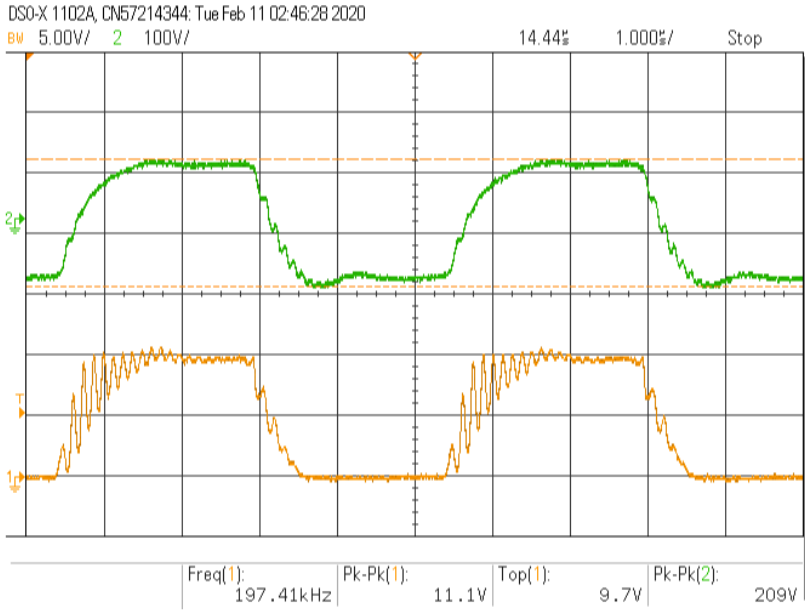


Figure 27: Driving a 10Kohm Load on Secondary Side, Primary Side (Yellow/Bottom), Secondary Side (Green/Top)

We needed to find the impedance of both the transformer and transducer to execute impedance matching. In order to find the impedances of both, a frequency sweep was performed. Figure 28 showed the test circuit setup for both the transducer and transformer. Channel 1 measured the voltage across the transducer and transformer, while channel 2 was connected to a differential probe across a shunt resistor to measure the current. Figure 29 is an example waveform that was captured.

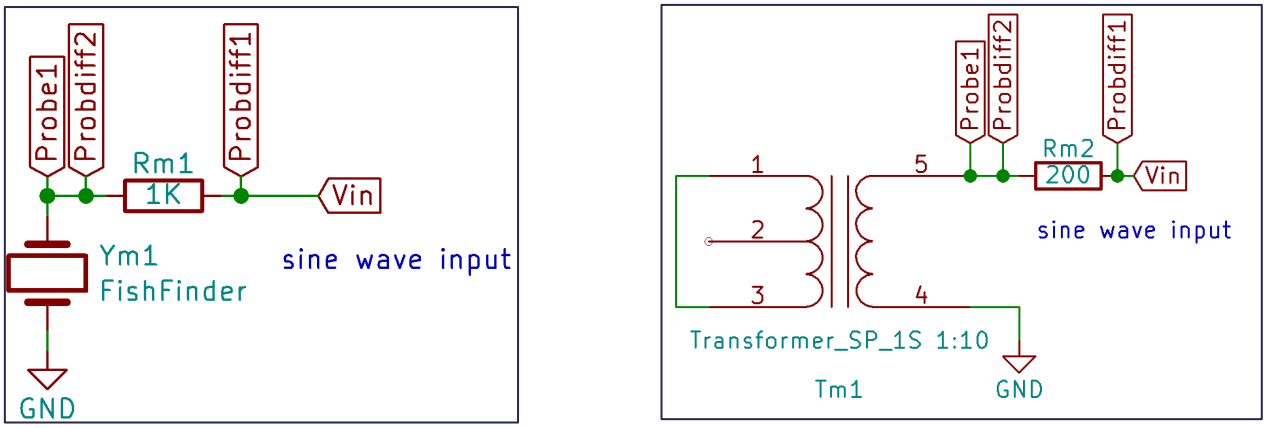


FIGURE 28: IMPEDANCE MEASURING CIRCUIT FOR FISH FINDER TRANSDUCER (LEFT) & TRANSFORMER (RIGHT)

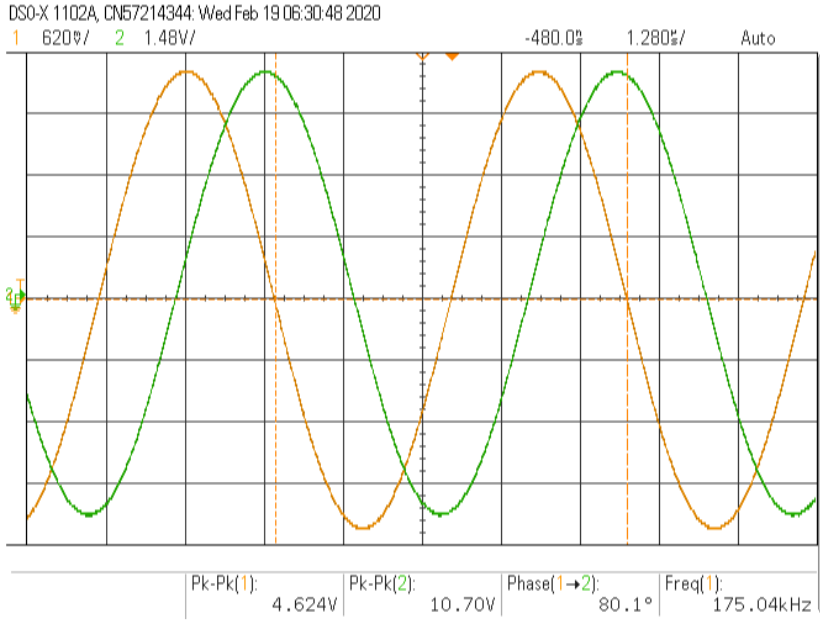


FIGURE 29: INPUT (ORANGE/LEFT) AND OUTPUT (GREEN/RIGHT) WAVEFORMS WHEN MEASURING IMPEDANCE

The resulting impedance sweeps are shown in Figure 30 and Figure 31. It is noticed that in Figure 32, for fish finder transducers, the phase difference of the current and voltage are mostly negative, however, between 76.7kHz to 80kHz, the phase is positive. This means the transducer has a resonant frequency at around 80kHz and 76.7kHz, not 200kHz.

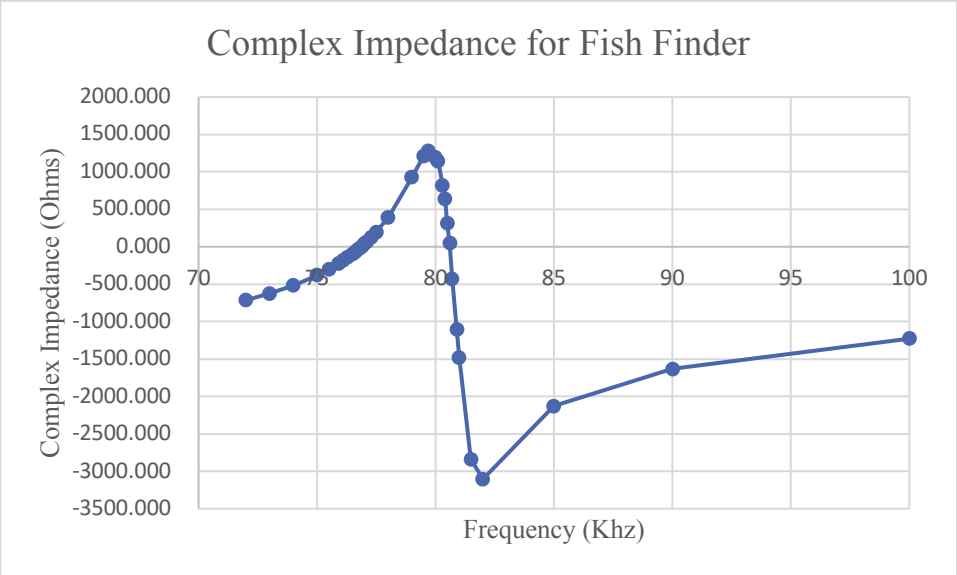


FIGURE 30: THE FISH FINDER'S IMPEDANCE DUE TO VARYING FREQUENCY

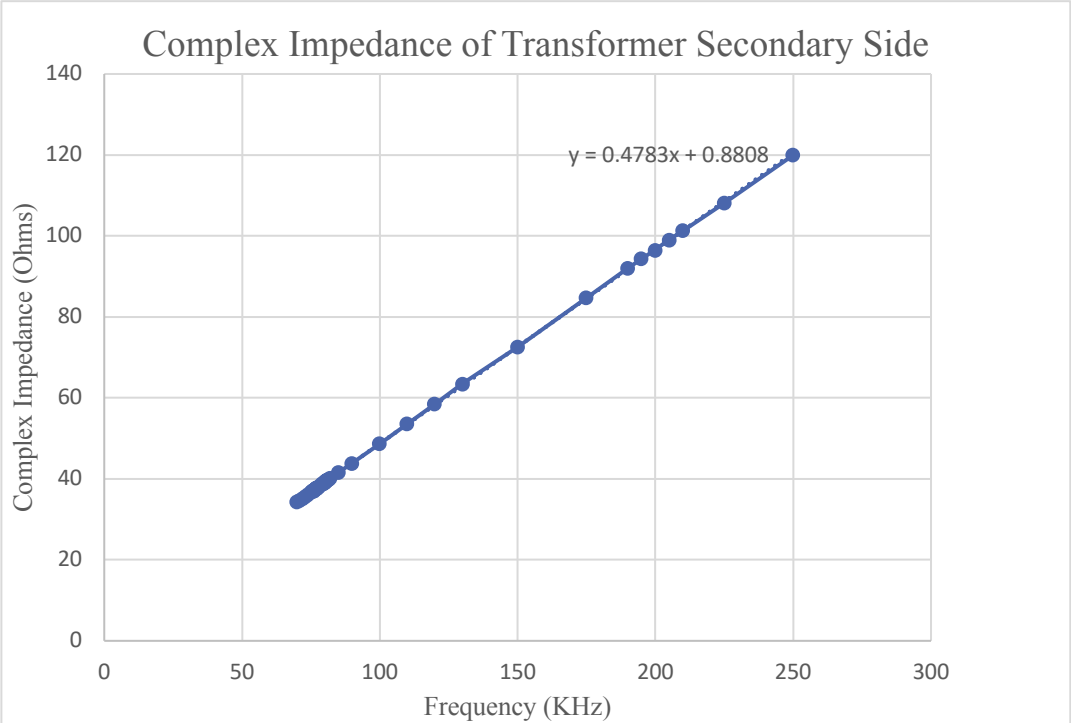


FIGURE 31: THE TRANSFORMER SECONDARY IMPEDANCE DUE TO VARYING FREQUENCY

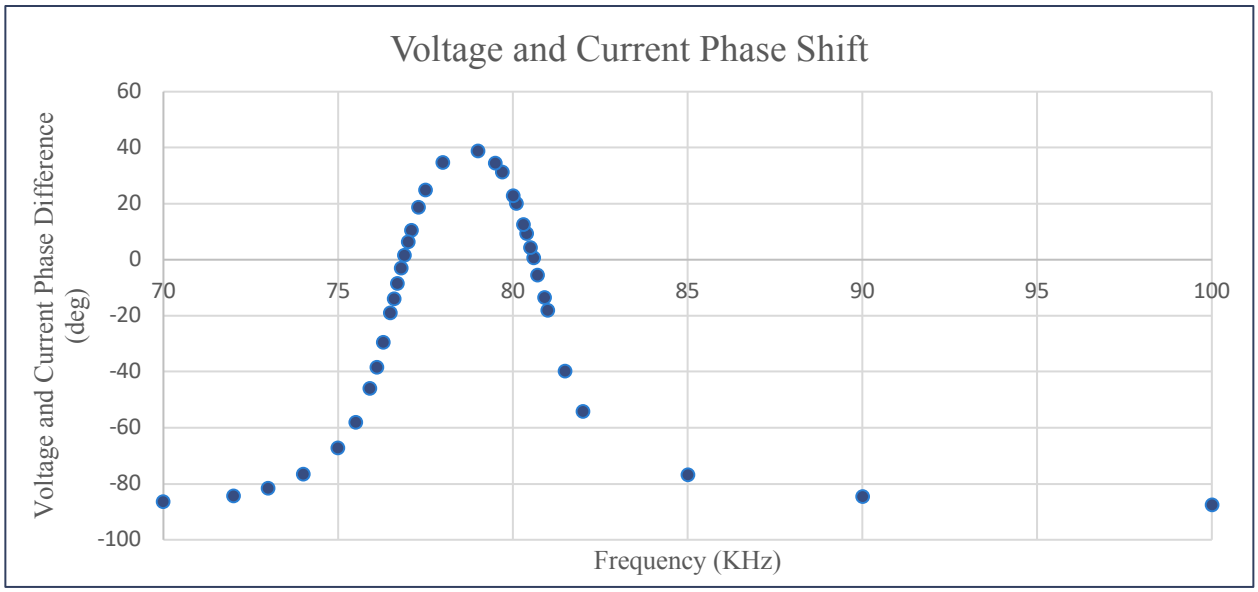


FIGURE 32: TRANSDUCER VOLTAGE-CURRENT PHASE DIFFERENCE DUE TO VARYING FREQUENCY

Once we knew the impedances of both, we were able to design a circuit to improve the mismatching shown in Figure 33. The matching inductance or capacitance values for the configuration are shown in the table in Appendix C. Although the transducer resonates at 80kHz, the use of capacitance at this high voltage does not imply the need for a high-voltage rated capacitor. To avoid this, the frequency chosen was slightly shifted. 81.5 kHz was eventually decided as a possibility since the microcontroller can output it and a realistic inductor value of 5.6mH could be used. Figure 34 shows the addition of the impedance matching configuration to the transducer driver circuit.

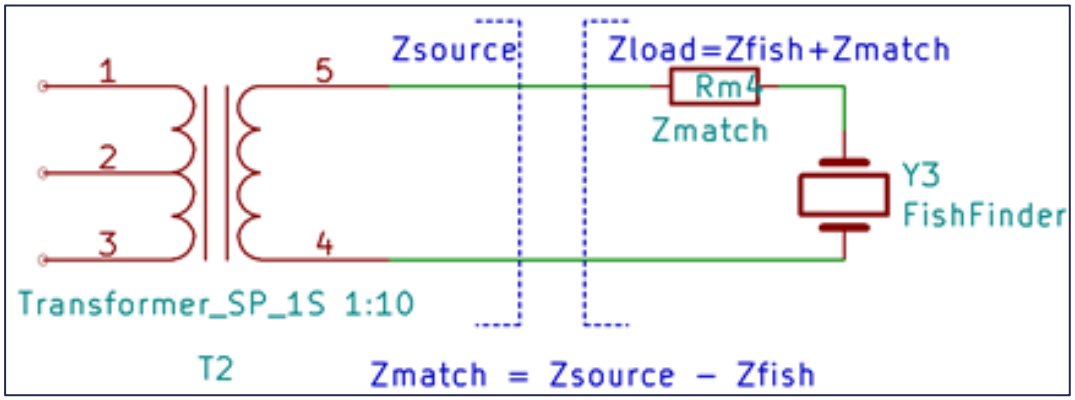


FIGURE 33: SERIES IMPEDANCE MATCHING CIRCUIT

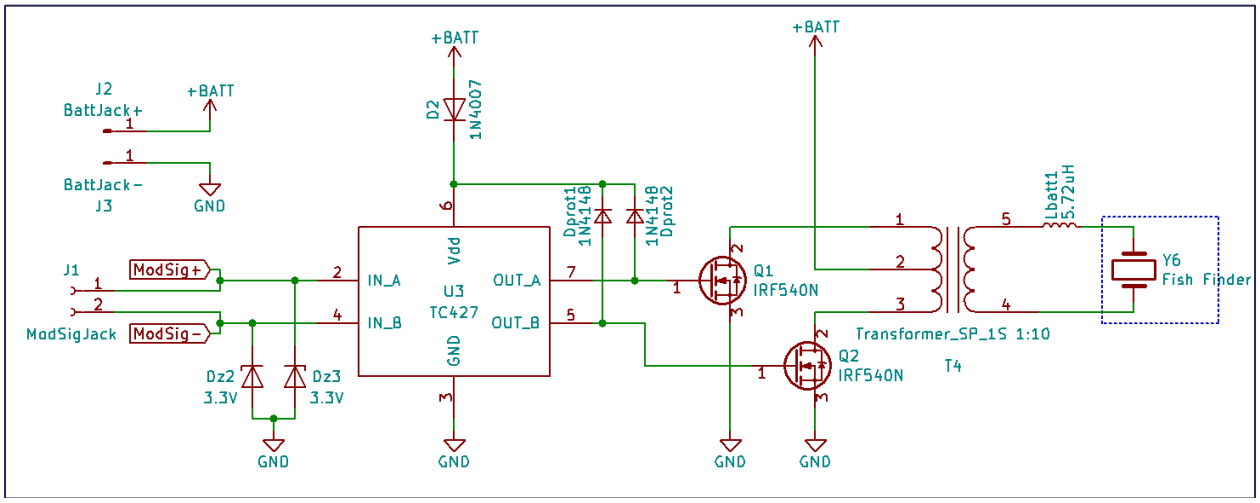


FIGURE 34: TRANSDUCER DRIVING CIRCUIT WITH IMPEDANCE MATCHING

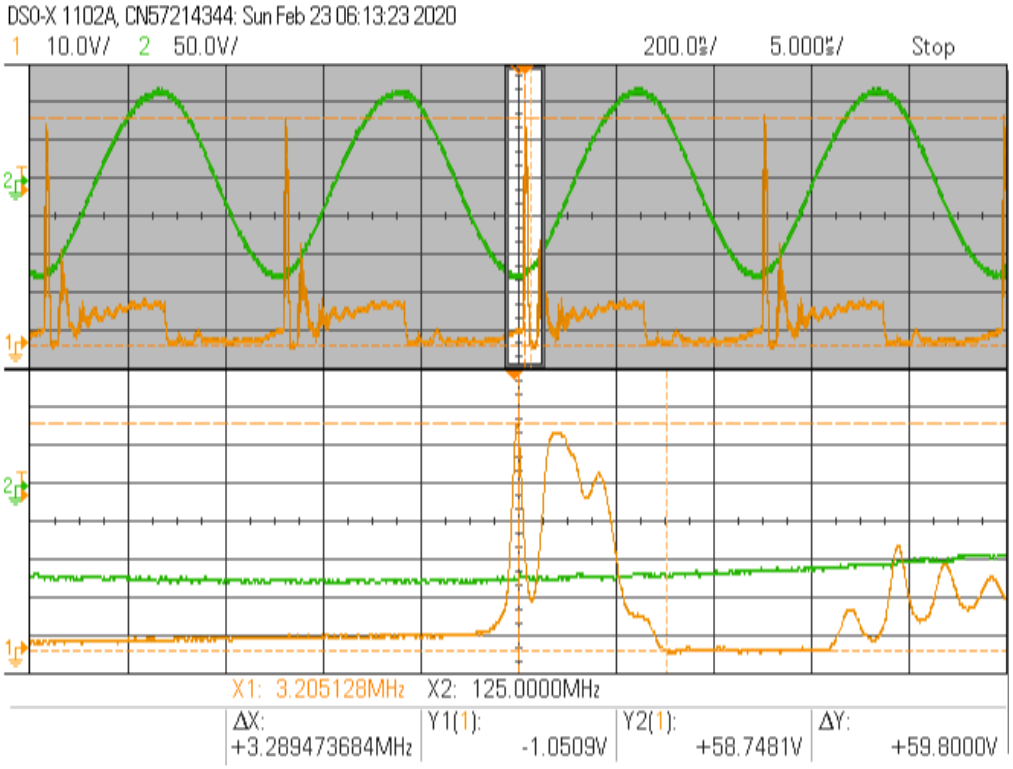


FIGURE 35: ZOOM IN ON PRIMARY SIDE FLY BACK PEEKS (CHANNEL 1)

Another problem that showed up after fixing the impedance matching on the secondary side is the primary fly back voltage. The primary side produces a huge voltage spike when the MOSFET is turned on. This is due to the rapid change in current across the transformer, similar to a large inductor. In addition, these voltage spikes caused damage to the TC427 gate driver. To help limit the voltage spike, capacitors Cr1 and Cr2 were added, and a resistor of 0.01Ω is put in series with the transformer to limit the current. In addition, the TC427 does not like the output ports to be back driven when the voltage is higher than rail. Two 1N4148 diodes were used to prevent this. The modified circuit is shown in Figure 36.

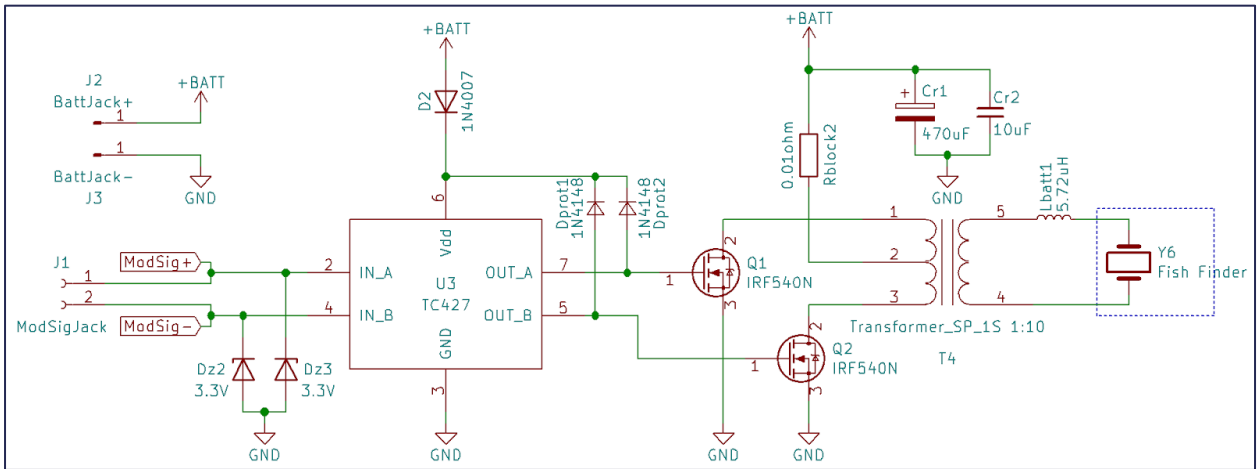


FIGURE 36: IMPROVED CIRCUIT DESIGN WITH POWER RAIL CAPACITOR

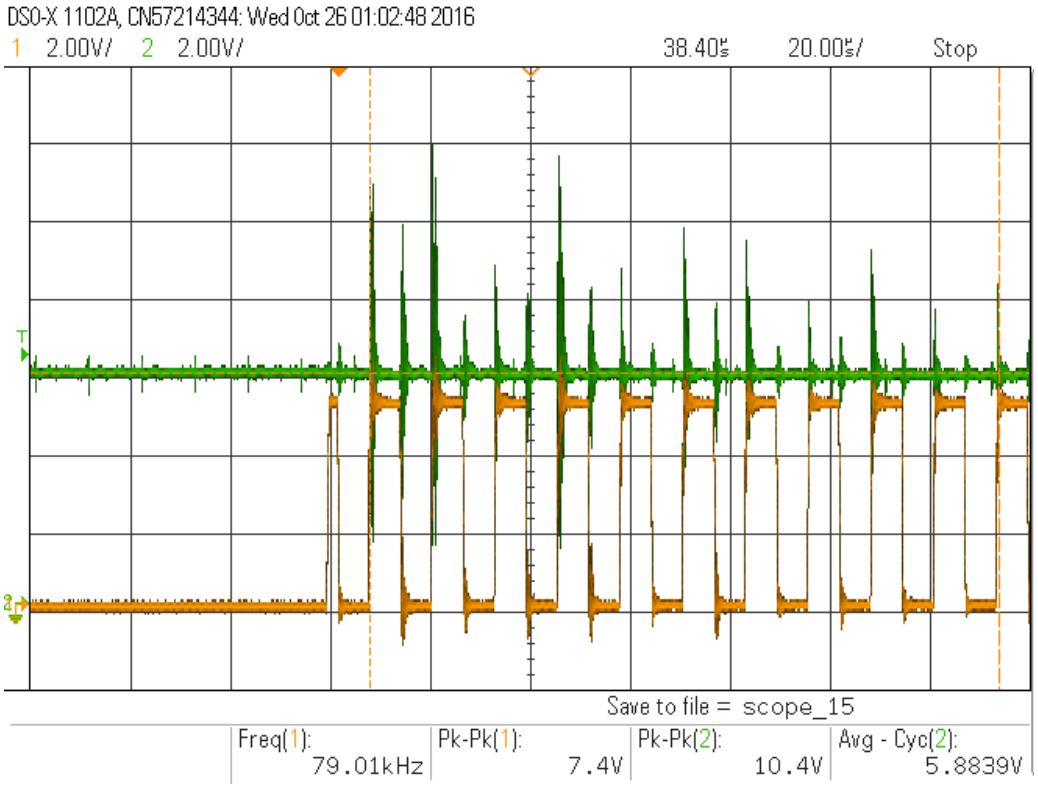


FIGURE 37: VOLTAGE ACROSS TRANSFORMER AND +BATT

However, there were still some large spikes on the power rail after these modifications (shown in Figure 37). Not only were there high voltage spikes, there were also some huge dips in battery voltages as well. This caused the battery voltage to temporarily go as low as three volts. When zoomed in on the measurement, it was found that the peak and dips are about 0.4µs in duration. This dip in voltage was likely due to the transformer acting as a short circuit at the instance of MOSFET turn on. Thus, the power rail voltage was pulled to ground. With enough capacitance, this ripple in voltage could be ironed out. Assuming, at turn on, the transformer was shorted, the only resistance in the circuit was Rshut1 (Figure 36). The instant current was 600

amperes if the capacitor needs to hold the voltage ripple within 1V. According to the Equation 5, the capacitance needed was *600μF*.

$$i(t) = C * \frac{dV}{dt}$$

EQUATION 4

Rearranged:

$$C = 600A * \frac{0.4\mu F}{1V}$$

EQUATION 5

Now that we had a functioning circuit for the transducer driver, it could be tested.

Signal Modulation

Whatever message is chosen to be sent from either node, the acoustic signal must be modulated in order to be able to be decoded. Our messaging protocol takes raw binary messages and uses it as a bitstream to determine the modulation needed to transmit. This signal can then be demodulated at the receiver side for further processing. There are many practical forms of modulation the team could use for underwater communication, but two in particular were focused on because of their simplicity, and effectiveness.

Frequency Shift Keying (FSK) is a form of signal modulation representing bits as a frequency change by a predetermined offset either above or below the base frequency. FSK modulation produces a signal that is always active, as in there is not dip in signal power or amplitude when transmitting. Though simple to detect and demodulate, the modulation would require the transducer to be amplified one hundred percent of the time it is transmitting, thus drawing more current from the on-board battery than desired.

On Off Keying (OOK), is another common signal modulation, where each bit is represented as either on or off. The one bit is represented as a pre-specified frequency for the period of a bit, and a zero-bit represented as no transmission for the period of that bit. Because the transmission is not continuous during the full transmission, edge triggered detection of a signal will not result in receiving a complete message. Each node of the communication system must have a prespecified indicator of a start of a message. The true advantage of OOK over FSK is power. OOK may reduce the power needed to transmit by a big factor, depending on the number of times the transducer may need to be excited as well as the proportion of the message that is zero bits. OOK will be essential in ensuring the AUV's wireless communication is using minimal battery charge.

Receiver Front End

The received signal from the transducer is AC coupled first. Most wireless receiver front ends for radio will have a few stages of filtering, however, a piezo-acoustic transducer only resonates at a very narrow frequency. This means no filtering is needed. The level of received signal voltage will generally need 80-100 decibels of amplification.

The signal voltage will also change as the distance between robot and boat change. This effect calls for an amplifier with AGC (automatic gain control). After searching the market, it was not realistic to have controllable high gain at 80 kHz bandwidth. Thus, the analog amplifier stage will have a constant gain of 80dB. For a shorter range, the amplifier saturates at rail voltage. For a

far range, when the output voltage after amplification is not reaching the rail, code is in place to set a corresponding threshold voltage base on the signal received.

The general design of the receiver front end was a high gain amplifier, followed by an envelope detector to demodulate the signal. An eighty decibel gain at 80kHz will ask for 64MHz of unity gain bandwidth of an operational amplifier (op-amp). Few op-amps can generate this much gain in one stage. The design achieves the amplification into two stages, with each stage amplifying the voltage for forty decibels and staying away from the bandwidth limit to allow for some phase margin.

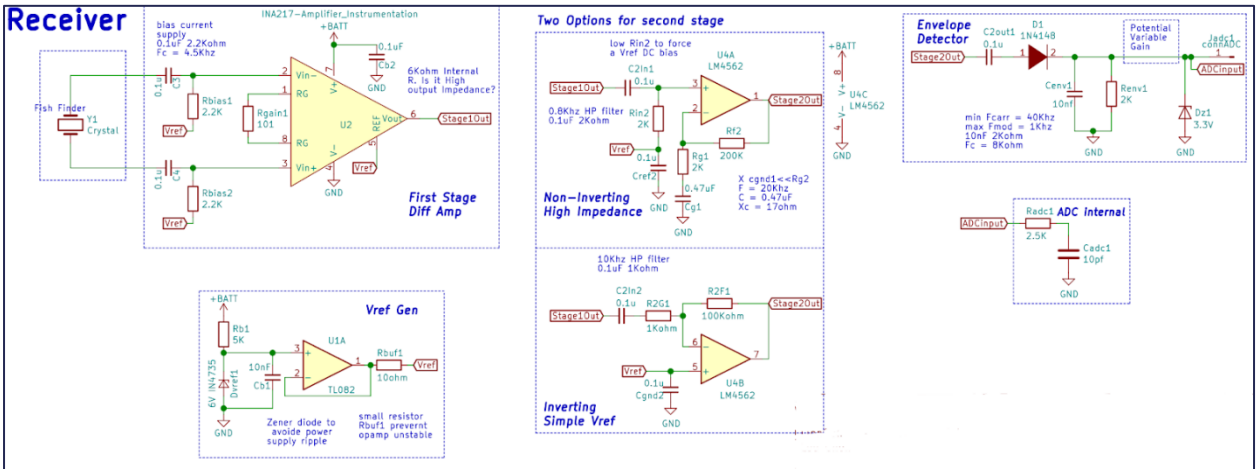


FIGURE 38: ACOUSTIC RECEIVER CIRCUIT DESIGN

The initial design of the circuit is shown in Figure 38. Since this circuit is powered by a battery and no negative rail is available, all of the amplifiers were configured to work on a single rail with the AC output voltage biased at roughly half the battery voltage to maximize the output voltage swing. The voltage reference could be generated with any unity gain general purpose op-amp that fit the voltage range. An IN4735 Zener diode was used to generate the reference voltage for the op-amp. Using a Zener diode instead of a voltage divider eliminated the power supply noise. Since most op-amps will become unstable when driving a capacitive load, a small resistor, Rbuf1, of ten ohms is used to limit the current.

The first stage is an instrumentation amplifier since it can easily take in a differential voltage, amplify it, and output with a DC bias. The INA217 instrumentation amplifier was used for its high bandwidth even at 100V/V gain. Since the datasheet mentioned that this amplifier uses current feedback technology, the resistors Rbias1 and Rbias2 are added to fulfill the input bias current.

Since the voltage is already single ended at the output of the first stage, the second stage could be built using general purpose op amps. LM4562 is picked for the voltage range and the bandwidth. There are two possible configurations for the second stage: non-inverting and inverting. The advantage of the non-inverting design is the high input impedance. The advantage for the inverting design is the simplicity of the design, using fewer parts. Since the input for the second stage comes from the differential amplifier in the first stage, we don't need to worry about the low input impedance, and the inverting design is chosen.

The classic envelope detector works by clipping the negative current, storing the positive peak voltage in the capacitor, Cenv1, and never discharging it. Then, a separate high value resistor is used to discharge the capacitor to be ready for the next modulated signal. In this circuit, the AC waveform is DC biased. To remove this DC bias, an AC coupled capacitor was used before the diode. However, this caused a separate voltage to build up between the diode and AC coupling capacitor. This circuit is acting as a clamping circuit, where no voltage shows up on the cathode of the diode. The envelope detector is tested using the setup in Figure 39. The result in Figure 40 showed this process of the negative bias voltage build up.

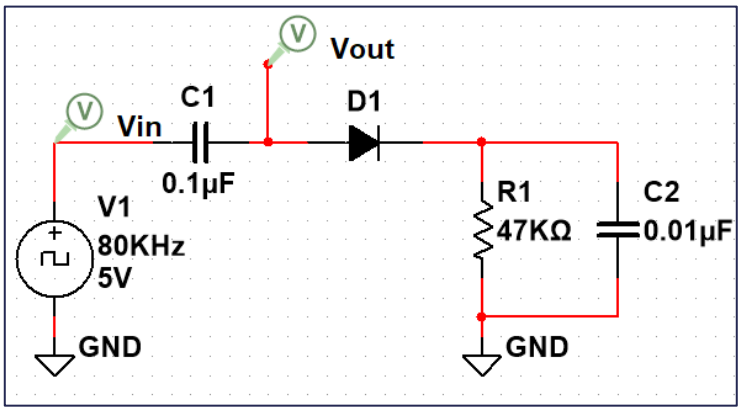


FIGURE 39: ENVELOPE DETECTOR CIRCUIT

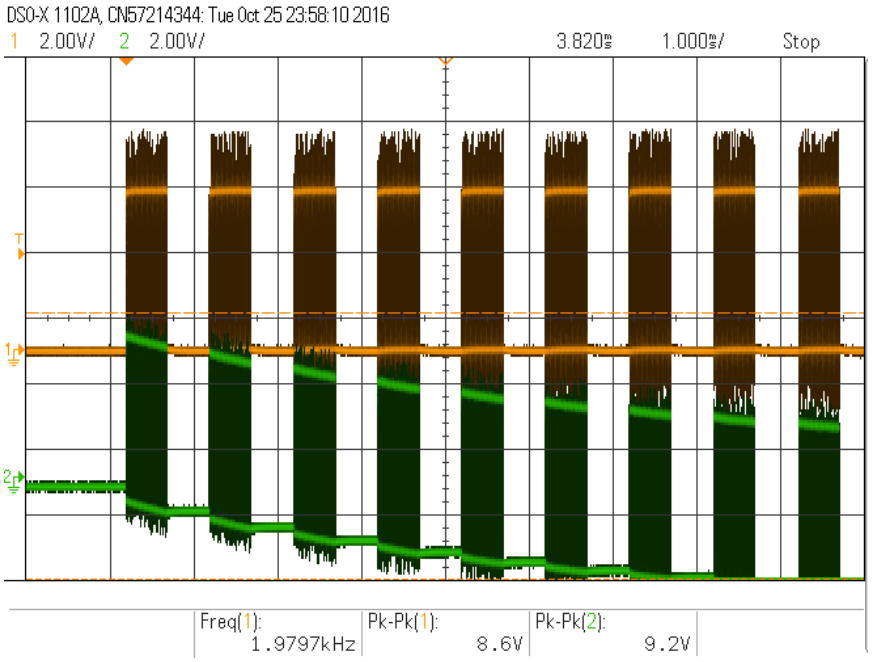


FIGURE 40: ENVELOPE DETECTOR TEST, VIN (ORANGE), VOUT (GREEN)

To solve the problem, the anode of the diode had a resistor connected to ground. This resistor drains any voltage build up on this point. This configuration stopped the bias voltage from building up with the tradeoff of a very small voltage drop on the output. However, the resulting waveform is clean and stable. On the output of the envelope detector, a unity gain buffer was added to prevent the ADC of the microcontroller loading down the envelope detector too much. This configuration is shown in Figure 41.

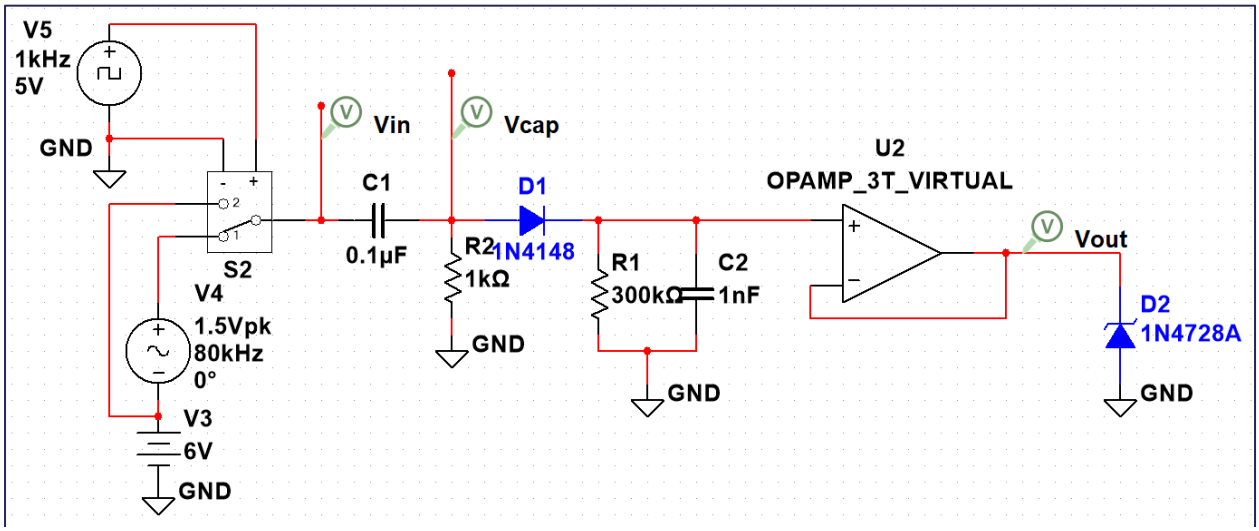


FIGURE 41: REVISED ENVELOPE DETECTOR CIRCUIT

Communication Protocol

Because of the problems and challenges that the transducer research brought, the communication protocol and specifications weren't specified until quite later. Because OOK modulation was a clear winner in terms of power management, the acoustic communication link had to have clear specification as to the starting bits, how to process the overall message, and how to respond properly. Decoding OOK and FSK modulated signals are very similar to decoding asynchronous serial protocol. A special key, or start bits, at the beginning of each transmission ended up being the indicator for a transmission. To end the transmission, each byte has a CRC error detection included.

The team had hopes to test the transducers for range, signal noise, and other measurements at an earlier point, to then decide on what desired specifications the communication protocol needed. Because of difficulties finding a commercial transducer, and designing a custom one, a lot of the early work was done on sampling of the ADC. Functions were put in place for the analog to digital converter to sample voltage levels and, with a threshold, determine whether the system is sampling a signal, or not, thus read in bits of a message. Because the wiring of the microcontroller to the transducer was unclear, critical features like automatic threshold control, two-way communication, receiving and sending were difficult to implement and test. The communication protocol was difficult to specify until much later, and a structure for one was temporarily put in place to move forward. The team knew the protocol would be primarily one way, with the AUV receiving information from the support boat. Because of this, the team knew a message packet structure needed to be implemented to specify between type of messages being sent. Messages that were more important, like recalling, or disarming the AUV, would need to be clearly identified from messages of lesser importance. The common practice is to specify and ID packet frame, or section of the message specifically for an ID indicator of that message. The packet structure would also need a payload, or raw message packet frame. The payload would designate to the raw information needing to be send or received. The protocol packet structure would also need start bits, the special key mentioned above that will determine the start of a message from other noise.

Transmitting Messages

Transmitting packets is done using the PWM module. Using pulse width modulation, we can create a square wave gate for the transducer to transmit our messages. By sequential switching on or off the PWM module, the piezoelectric transducer will transmit a signal for a constant interval of time. This was verified by coding the PWM module to send a square wave example message packet through a GPIO pin. After a look through the oscilloscope probed on the configured GPIO pin, the result was an accurate representation of the packet with the signal. Because of this verification, the assumption is that the piezoelectric transducer can transmit and receive messages, allowing full acoustic underwater communication.

Receiving Messages

Using an on-board ADC, each node of the communication system can sample the GPIO pins which are wired to the nodes piezoelectric transducer. This and the analog circuitry mentioned in the Receiver Amplifier section above, will result in reading the voltage levels of the transducer at the appropriate dynamic range specified by the ADC. When excited by an incoming signal. These voltage readings are then stored to be processed. The on-board ADC has 12 bits of resolution, meaning the ADC can read a value from 0 - 4096 which represents zero through 3.3V, which is the board's operating voltage. During the sampling, our source code lays out a function called *process_adc* inside of the *process_rx.c* file, which contains all of the acoustic link's receiving logic. When called, the function will loop through the ADC samples and look for appropriate length messages with the correct start frame to them. Once this is done, the message is put through CRC error detection and, if passed, will continue to be processed. The *process_packet* function, when called inside of *process_adc*, will strip the packet and process its ID and payload frames, which are vital to processing, and respond with the appropriate actions for every ID possible.

3.5 Operational Considerations

3.5.1 Power Considerations

The stock battery for BlueROV2 is a four cell, 18Ah lithium-ion battery with a voltage between 12V and 18V. The battery is capable of 90A continuous current and 130A burst current. According to the BlueRobotics' estimation, each battery is capable of running the stock robot for one hour, which estimates an 18A continuous current during the operation of the robot. Considering that the Jetson Nano, spear carriage motor, and acoustic link all drain a decent amount of power, we decided to put two batteries into the robot. One battery powering the stock BlueROV2 components in the main bay and one battery powering electronics in the additional bay. This way, the power for the existing robot drive train is separated from other components.

Many of the added components (Jetson Nano, Arduino Mega, ethernet switch), however, require 5VDC to operate, which the batteries cannot provide. A simple 5V step down converter would ultimately work, but there was a consideration to add an external switch to power on/off the AUV. Initially, in order to turn power on or off to the robot, the battery chamber had to be opened to plug or unplug the battery. Opening the chamber is a time-consuming task, since the seal needs to be tested for fifteen minutes every time before the AUV can reenter the water. In order to accomplish both the 5V step down and the on/off control, we first decided to use a power switch relay controlled by a reed switch. However, none of the DC relays on the market have power capability for 90A current, and the AC relays would not be able to kill current arcs while switching

DC voltages. One alternative we also looked into is the use of a MOSFET. One of the main concerns, however, is that the on-resistance, Rds(on), would draw substantial current and result in power loss due to heating. The most applicable NMOS transistor is the IPT004N03L with an on resistance of 0.4m**Ω**. Even with such a small resistance, there is still some power lost in both continuous and burst current scenarios, shown in the equations below. Due to these problems, we decided not to use a switching device on the battery.

$$P = I^2 * R = 18^2A * 0.4m\Omega = \mathbf{0.13\ W}$$
$$P = I^2 * R = 90^2A * 0.4m\Omega = \mathbf{3.24W}$$

With the 5VDC requirement, we also looked into DC-DC voltage regulators as a viable option. Table 3 below is a small list of possible voltage regulators. Among them, the NID100-5 has the best price to power ratio while meeting the performance requirements. The NID65-5 was a close competitor, however, the total current needed was at least 8A. Ultimately, the NID100-5 was chosen and also has a remote on-off function which can be used to create an external battery switch. The tradeoffs can be seen in Table 3. Using a magnetic reed switch connected to the on-off pin of the regulator, all of the 5VDC components could be power cycled by inducing a magnetic field through the chamber wall to close the reed switch.

TABLE 3: POWER CONVERTER CHOICES

Part #	*Output V-A*	*Price $*	*General Efficiency*	*Note:*
RPM5.0-6.0-CT	5V-6A	$7.40	91% and above	BLGA package
NID65-5	5V-6.5A	$14.00	93% for 24Vin	DIP
NID100-5	**5V-11A**	**$15.65**	**93% for 24Vin**	**DIP**
PI3302-00-LGIZ	5V-10A	$15.00	90%	BLGA package
SKM30A-05	5V-6A	$29.00	88%	PC-pin Isolated

This switch method was applied to both batteries powering the existing and added processing units that all require 5VDC. Other than these 5V computers, only the transducer and spear motor will draw raw power. The transducer also has its own regulation device, and since the spear motor is the same as the AUV's thrusters, it can be connected to the raw battery power as the existing ones do.

3.5.2 Mechanical Considerations

Materials

Material considerations pose the most significant mechanical challenge for this project. The salt water marine environment means that most metallic materials will corrode, unless they have some protective coating. Most steels are useless, with only stainless steel being usable in salt water. Aluminum is slightly more corrosion resistant, as the top layer oxidizes and provides a thin protective layer, however the high corrosion of the ocean can still be problematic. To combat this, any aluminum parts need a protective coating of either anodizing or powder coating, which provides a barrier to corrosion (Aluminum Handrail Direct, 2018). Additionally, any parts that were able to be made from plastic were utilized. Either 3D printed or laser cut materials worked

for our robot, which do not have corrosion problems. They do pose problems with strength, where they are not nearly as strong as aluminum or stainless-steel parts.

Buoyancy

Another mechanical consideration was buoyancy. The existing BlueROV2 was slightly positively buoyant to eventually rise to the surface in the case of a complete operational shutdown. Since we added an additional electronics chamber to house added components, the buoyancy was significantly affected and the AUV needed to be adjusted to maintain the slight positive buoyancy.

Two properties affect the stability of the UAV underwater: overall buoyant force and the location of the center of buoyancy. In order to conserve energy and not have to constantly drive the thrusters, the AUV needed to maintain virtually neutral buoyancy underwater. This was relatively easy to accomplish by designing it with a slight positive buoyancy, where we added weights as necessary to keep the robot underwater. If needed, Blue Robotics also sells buoyancy foam that we could have used to create more buoyant force.

The second aspect of buoyancy is the location of the center of buoyancy. This is a theoretical point, similar to the center of mass, that indicates the center of all the buoyant forces acting on an object. This is important because the relationship between the center of buoyancy and the center of mass dictates the orientation in which the AUV settles, as the net upward force created by the center of buoyancy tends to end up above the center of mass. This indicates that it is important to locate the center of mass below the center of buoyancy to make sure the robot remains upright underwater. This was accomplished by locating the most buoyant components as high up as possible, and the heavier, less buoyant components farther down. SolidWorks was used to estimate the location of the center of mass and we roughly determined the center of buoyancy by testing underwater and visualizing how the robot settles.

3.5.3 Emergency Situations

There are a few onboard emergency situations that were addressed within our design. Currently, the existing BlueROV2 chamber contains leak detection, pressure detection, battery monitoring, and internal temperature sensing. We duplicated all of these sensor configurations in the new enclosure. In the existing enclosure, the Raspberry Pi manages emergency messages from the sensors and alerts the navigation software, where the AUV ascends directly to the surface and then travels back to the known location of the boat. Similarly, the Arduino Mega will manage emergency sensors in the additional enclosure.

Leak

One of the potential issues in the enclosures was a leak due to faulty seals. Currently, BlueROV offers the SOS leak sensor, which uses four sponge probes to detect moisture. When a leak is detected on one of the sponges, the signal is pulled high to VCC (5V). This sensor was easily integrated on the Arduino Mega as a digital input, where an emergency message is relayed to the navigation system.

Pressure and Temperature

Both pressure and temperature are managed on the processing boards in the existing enclosure. The Pixhawk has an onboard pressure sensor that detects internal pressure changes inside the chamber. The Raspberry Pi has an onboard temperature sensor that triggers an

emergency message for the navigation software. In the added enclosure, the Jetson Nano has a temperature sensor that was utilized in the same way.

Battery Monitoring

Currently, the BlueROV2 has a power sense module (PSM) connected to the battery. The PSM provides both analog current and voltage values to manage battery level and current consumption. The lithium-ion battery suggests not discharging it lower than 12V (3V per cell), so the AUV navigation system will get a "return to home" message when the PSM reads a threshold value (ex. 12.25V or lower). We also were able to view power consumption as we do testing and easily change this threshold as we see fit. The existing battery PSM already interfaces with the RaspberryPi, and the added battery PSM was managed by the Arduino Mega as an analog input.

3.6 Conclusion

With the integration of all of these systems, the AUV is able to accomplish the three objectives and fulfill the entire mission flowchart in the future. A full electronics system layout can also be seen in Appendix D, illustrating the wired connections between existing and added components.

Chapter 4: Results

After focusing our methods and designs, the team began assembling and testing different components and processes. Most prototypes were tested out of the water to begin with, ensuring reliability and intended functionality before placing the robot in risky situations underwater. Once a component or process was deemed necessary for pool testing, we put the robot through a lengthy procedure, checking for proper and reliable chamber sealing, pressure, buoyancy, battery capacity, communication, and physical electronic connections (See Appendix E for preflight checklist in the user guide). Pool testing was limited to open water hours at the university's swimming pool, so ensuring all systems were functioning beforehand was critical.

4.1 System Development

4.1.1 Integration of Onboard Communications

The first major component that needed to be assembled and tested before much else was the communication system. We extended the existing internal static IP network configuration, adding a network switch to provide communication between the existing Raspberry Pi, Jetson Nano, and external computer via the tether. Each device was given a static IP address of 92.168.2.X, where X is a unique integer from the other devices. The tether was kept connected for easier testing and debugging while the robot was in the pool, but the hope was that the tether ethernet connection would be replaced with a wireless communication system using acoustics. Two additional ethernet ports are also available in the switch for additional devices.

Another addition to our communication system that was critical for testing and debugging was the use of secure shell (SSH) protocol with the Jetson Nano. When testing, multiple iterations of the software could be updated from the topside computer. This allowed us to develop the Jetson Nano software without opening up the chamber between each test. With a robust communication network established, it was easier for the team to debug software and develop the AUV system architecture.

4.1.2 Chamber Layout

One of the issues we faced was the limited space for hardware components within the AUV. Since we were adding many computers and electronics, we decided to mount an eight-inch diameter cylinder chamber on the AUV, the largest that BlueRobotics offered. This gave us some flexibility to iterating our design and adding unexpected components.

Following the wiring diagram that was initially created, we addressed the physical layout of the chamber. Initially, we created an acrylic structure that could house three batteries and include a platform where the electronic components would sit. Our first iteration caused a few issues. The platform was too far from the centerline of the chamber, so there was limited airspace above the components for wires to span, where they would get caught and become unplugged from components. Furthermore, the electronics were not strategically arranged and connections were not easily accessible when the endplates were disassembled. The second iteration saw much improvement. Since the third battery was going to be initially used for ballast and was eventually not needed for actually powering devices, we created a structure that would only house two. This allowed for the platform to move down to the centerline of the chamber, giving more space for the

components and wiring. We also rearranged the electronics so that the Jetson and Arduino Mega could be easily accessed since they are home to many of the peripheral connections. Figure 42, shows the final design of the chamber assembly. Once the chamber was assembled and a frame insertion technique was established, navigation methods could begin implementation and testing.

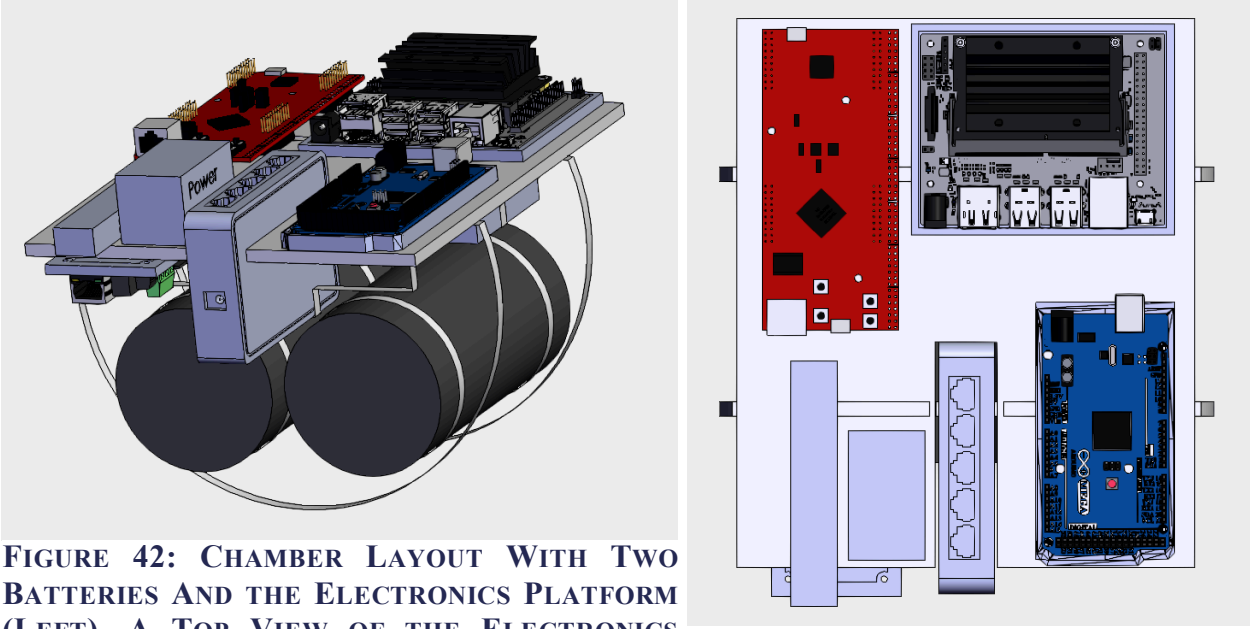


FIGURE 42: CHAMBER LAYOUT WITH TWO BATTERIES AND THE ELECTRONICS PLATFORM (LEFT), A TOP VIEW OF THE ELECTRONICS LAYOUT (RIGHT)

4.2 Navigation

4.2.1 Testing the Building Blocks

Before placing the AUV in water, we were able to test the movements in the lab. The AUV was assembled exactly how it would be in water, and we commanded each of the building block movements, identifying that the motors were rotating in the correct direction by feeling the air flow with our hands. This was a simple way to test forward, backward, ascend, and descend. To test 'turn to an angle', we placed the AUV on a cart, executed the command, and pivoted the cart until the correct motors shut off. The software read in the magnetometer data and would run the turn command until the heading was equal to the given angle. This simple test allowed us to approximate if the AUV was reading in the heading data correctly and shutting off the motors when the AUV was turned to the given angle.

Once we determined that the software was executing commands properly, we performed in-water testing in the WPI swimming pool. We went through each movement and tested its execution and precision. Forward and backward were simple to test, as we enabled depth hold on both and witnessed whether or not the AUV executed its intended action. We timed the motion in order to confirm that the motors were running for the given time. We also noted that changing the throttle percentage was increasing or decreasing the speed appropriately. After iterating the testing, we determined that the proper throttle is about 40% - 50%, allowing the AUV to maintain control and for the cameras to capture clear images to ultimately identify lionfish in the frame.

After forwards and backwards movement were finalized, we tested our turning process in water. Testing this process was more involved since the magnetometer data needed to be accessed to track the relative movement. The motor controls were already established for a proper pivot turn, so we implemented those controls into our overall turn process. We created a command procedure to input the throttle percentage, but now incorporated a relative angle to turn to instead of a time to execute. The algorithm reads in the current heading and begins the pivot turn, continuously reading in the heading and subtracting it from the initial data point until the absolute value is equal to the given relative angle.

When testing in the pool, the compass drifted approximately a degree every five seconds, so it was essential that we utilized a relative turn. In our program we also created a buffer of two degrees in either direction to account for the sensor drift and environmental conditions such as currents. The turn procedure executed correctly in most conditions, however, our algorithm was flawed. The compass measurements overlap at 0 and 360 degrees, which affected the algorithm mathematics in this specific condition. The AUV would continue to turn another 360 degrees until it reached that appropriate heading again. By the time the process is executed, the magnetometer drifts a few degrees, so the AUV does not land exactly on 0/360 degrees again and can then terminate within the buffer location. We implemented a few changes to limit the probability of this condition happening, but the algorithm was never changed to account for this. The condition happened few enough times where it did not halt the AUV's improvements, and the result is not a critical emergency if it happens: the AUV just takes longer to complete the turn.

To test ascend and descend maneuvers, we created an algorithm where a throttle and depth position were needed as inputs. The surface of the water is considered depth zero while any negative depth input is relative to the surface. If the current depth was lower than the desired depth, the vertical motors would spin clockwise at the given throttle until the desired depth was reached. Likewise, the motors would spin counterclockwise until descending to the desired depth. The depth sensor rating is accurate up to a few millimeters so we were confident in the measurements. It should be noted that this was based on measurements based on dense, smooth objects like the floor of the pool.

With this testing came many issues. One of the major problems with the magnetometer was the drift that occurred. Although we considered the magnitude of the drift was dependent on the fact that we were inside of a metal building, we agreed that the sensor calibration would likely be much better outside of a building. Testing in the pool was challenging, but we would only test maneuvers for short periods of time where the drift wouldn't be too significant. It was also important that we utilized current relative motion, rather than motion from a preset absolute heading, to maintain more accuracy as the AUV maneuvered through the pool. This posed a new issue, however, where the AUV does not actually know where it is relative to the start point. Another issue was overshoot. When the AUV completed a motion, the motors would shut off immediately. The AUV's inertia, however, would carry the robot past it's intended target. Although the overshoot wasn't critical in the pool, it is important that future groups incorporate gradual stops or some type of motion control feedback system to execute maneuvers with greater precision. Overall, the motion building blocks were a great starting point to incorporate more advanced navigation like obstacle avoidance and implementing search patterns.

4.2.2 Implementing and Testing Sonar for Obstacle Avoidance

After establishing reliable movement commands, we worked to implement BlueRobotic's sonar devices onto the AUV. The single beam echosounder detects large objects, returns a relative

distance, and provides a confidence level depending on the strength of the echo. We decided to have one device facing forward to detect objects in front of the AUV, since it will primarily move in the forward direction, and another to face downward to detect the seafloor. The devices were designed to communicate through an open-source binary message format called "ping protocol" and were interfaced with an Arduino Mega microcontroller. To test the devices and the communications with the Arduino, we created a simple program to read in the distance and confidence level data on the serial monitor of a computer. Obviously, the data was extremely inaccurate since the sensors are calibrated to be in water, but we were able to establish and test the interface with the microcontroller.

An important consideration with the sonar is that these echosounders only transmit sound waves at a frequency of 115kHz and are not able to be changed. With two operating at the same time, there was a possibility of interference among them. In order to alleviate some of the cross talk, we incorporated a process that alternated pings between the two. When testing, however, we discovered that because the two devices are orthogonal to each other, there is very limited interference anyways. We still decided to include the alternating pings to minimize the possibility.

Once the sonar was able to be viewed on the serial monitor on our testing computer, we brought the same configuration to the pool. We needed to test the accuracy and range of the sonar to understand its capabilities. We conducted a simple test and had one of our group members stand in the pool at measured distances from the echosounder. We then read in the data on the serial monitor and compared it to the actual measurement. We concluded that the readings would fluctuate by a few centimeters of the actual distance and therefore we should account for this variation when writing our high-level control program. We also concluded that the further an object was from the echosounder, the more challenging it was to capture a specific object and differentiate it from the background. Although the ping sonar is rated for a range of 30 meters, we discovered that it became unreliable for detecting distinct objects at around 15 meters. This was not critical since the AUV only needs to detect objects a few meters in front of it; we therefore implemented a filtering process that disregards any measurements over 10 meters to get rid of this unnecessary data. We also utilized the confidence level to complete more filtering, only using sonar data with confidence levels higher that 80%. Our process allowed the high-level computing to only read in relevant, accurate data needed to make decisions. Once the sonar could be read reliably, we began combining it with the movement commands to establish the AUV's basis for navigation.

4.2.3 Implementing More Advanced Navigation

The next implementation for the AUV was obstacle avoidance. Two commands using the sonar were created and tested to establish some type of obstacle avoidance. These are:

- **Ping:** Returns sonar ping data for specific sensor (bottom or forward facing). This allows us to debug and see what the sensors are picking up while the AUV is in the water.
- **Bottom Hold:** Uses the bottom facing sensor to hold the AUV off of the bottom at a given depth. The throttle percentage for depth changes and distance from the bottom are both inputs to the function. This can be used on slopes and flat bottoms in order to keep the AUV a constant distance off the bottom as the terrain changes while searching for lionfish.

After integrating the sonar into the overall AUV system, the 'ping' command was simple to implement, where the sonar readings could be accessed on screen to ensure their functionality.

Second, ‘Bottom Hold’ was created as a process that keeps the AUV near the seafloor. If we used depth hold, the AUV holds its position relative to the surface and could easily hit the varying seafloor. Using the downfacing sonar, we implemented a program called ‘Bottom Hold’ to maintain a given distance off of the seafloor. Lionfish spend most of their time along the seafloor or around obstacles like rocks and reefs, so ensuring that the AUV maintains close proximity to these is critical is efficiently navigating and capturing the invasive species. Testing the process entailed improving the mathematics and filtering the irrelevant data in order to be more robust. This program was intended to run alongside another search pattern, similar to how ‘stabilize mode’ and ‘depth hold’ can operate together.

4.2.4 Search Patterns

After establishing ‘Bottom Hold’, a few search patterns were created to test the functionality of the AUV’s system. The first simple pattern that we tested was:

- **Square:** The AUV autonomously travels in a preprogrammed square. Essentially, it is a forward drive command followed by a 90 degree right turn for four consecutive sequences.

This demonstrates the self-navigation abilities and ability to preprogram search patterns into our framework. We continued to run this process to determine the precision of the movements. For most of the executions, the AUV finished the program up to two meters from the starting position. The AUV navigated different sizes of rectangles ranging from one-meter lengths to twenty-meter lengths. As the length of travel increased, so did the error. This is essentially a limitation of the sensors onboard and would be minimized with more accurate sensors. This confirmed more of our assumptions about the overshoot that occurs after the motors shut off and the inertia carries the AUV further than its intended target. After testing ‘Square’ and other sequences of movement commands, we created a more advanced program that would work well in a pool environment. This program was:

- **Roomba:** The AUV will drive forward until it detects an object within a meter in front of it. Then, it will turn to a relative heading of 95 degrees in either direction and proceed forward again, repeating this cycle. A 90 degree turn results in the possibility of the AUV simply following the wall, instead of covering the surface area of the pool. The obstacle distance, throttle percentage, and run time are all inputs that can be tested and adjusted easily. It is similar to the navigation of early generation IRobot Roombas, where it gets its namesake.

The algorithm was somewhat simple to begin with, reading the front facing sonar data while executing a forward command. Once the sonar data is less than a preset threshold value (one meter for most of our testing), the forward command terminates, and the AUV makes a pivot turn. Our first run of the program worked as intended, and from there we tweaked the inputs to make it more efficient. This was a great search pattern to visualize the AUV’s functionality in a pool. Further search patterns that would perform efficiently in a coral reef or seafloor environment were designed theoretically (see path planning in methodology) and would be a substantial implementation for future MQP groups.

4.2.5 Process Improvement and Overall Testing

Our first version of the control software was little more than a test bed for these functions. However, as we began to develop more intensive and involved commands we ran into issues. One of the major issues that we had to deal with was threading together processes in a constructive way. The test bed version of the software would just spawn off a new process anytime a command was called to run that process and many times these were never terminated for simplicity. This would result in processes fighting over each other for control of the AUV. For example, during one of our pool tests we were testing the Roomba function using depth hold at one meter below the surface. When we attempted to run a bottom hold function, the depth hold was consistently overwriting any of the change depth functions that were attempting to be sent to the motors in order to maintain the AUV at one meter below the surface. 'Bottom hold' and 'depth hold' are two functions that should not be run simultaneously and our system allowed it to happen. We also encountered stuttering due to commands from multiple processes being given to the motors, thus diluting the commands that we actually wanted to run. In addition to these issues, there was no way to kill the processes. This would be an issue in the future since it limited our emergency procedure capabilities.

With these in mind we began to develop a brand-new software platform that uses state machines. We began by separating navigation-related systems into two main processes, navigation and depth control. In addition to these there are also processes to integrate with identification software as well as sensors interfacing with the Arduino. When a command is called, the state machine inside the process advances in order to run the function that the command tells it to. When a new command is issued, the state is changed once again and the AUV completes its new commands all in the same processes. This system also allows us to have an effective kill switch which terminates all of the processes sending commands to the robot, which we can issue at any time if needed in an emergency. All of the functions that are listed above were turned into different states in order to ensure that we have the same functionality on a more robust software platform. In order to allow the separate processes to communicate with each other and the user, we set up multiple queues that they can push and pull from in order to reduce synchronization problems. In addition to this we also parse user inputs as they come in and only push the relevant data to the processes that need it, allowing for better control over the AUV and reducing useless information going through processes.

4.2.6 Emergency Situations

Another part of navigation that we considered was the fact that things may go wrong during a mission, so the AUV has to be able to recover from that. Even if there are no emergencies during a mission, we still need to be checking data that could end a mission such as remaining battery life.

Our new software was laid out in a way where we can easily end any actions we are currently taking should an emergency happen. While we did not have time to fully implement this, the building blocks are all there. The first emergency situation that we were considering was a chamber leak. BlueROV has leak detectors in the main chamber already and we added two to the new larger chamber. These detect moisture and send an alarm to the Arduino if they detect water. BlueROV also already had a battery monitor in the stock version of their hardware package, but since we were using two batteries this needed to be modified. Other emergency situations that we did not have time to begin implementing are sensor (or camera failures), loss of communication to

the surface, and crashing of the identification program. These can and should be added in the future to ensure safe missions for the AUV and the reef and divers around it.

All of our navigation functions had parameters at which the commands should end. For example, in 'move forwards' an amount of time is an input, while in 'pivot to angle', the heading is the input. This ensures that the AUV will never just go off on its own until it runs out of battery. Checks over the wireless communication system (the acoustic link) could also tap into this to reset the timers, thus allowing the robot to stop if connection is lost to the boat.

4.2.7 Combining Navigation with Identification

It is important that the identification sub-system communicates with the navigation sub-system. The identification system can tell the navigation system when to stop the search if a lionfish is found, or a diver needs to be avoided. It also tells the navigation system where the lionfish is in order to navigate closer to the fish. With the framework discussed above, both systems run as separate processes that have their own states and functions. When they need to communicate they use a queue system which ensures that we do not run into synchronization issues. They can pass the necessary information to each other without interference, solving the communication issues in a simple and elegant way.

4.3 Identification

4.3.1 Scoring/Model Comparison

Once the training of all three models was completed, they were compared. To determine which model would best suit the needs of the robot, the decision was based on which model meets our minimum fps baseline, greater than 15fps and performs the best in accuracy. To determine the comparative accuracy of the models a test set was used. The test set consisted of 1,141 images that the models had not seen before. The accuracy was calculated on the basis of behavior, if at least one Diver was located in the frame did it detect at least one diver, and if there was at least one Lionfish in the image did it detect at least one Lionfish. This process was determined as if one Diver was in the frame the robot would perform the same action if multiple divers were in the frame, as an indicator to the robot to operate safely around the divers. In the case of Lionfish, if there was at least one Lionfish in the frame the robot would hunt it, it would not matter if more than one Lionfish was in the frame at a given time as it can only target one Lionfish at a time. The two accuracy percentages were then averaged for the final recall percentage. In terms of scoring the speed of the model, due to time limitations and complications converting the model, these tests were performed on a desktop Nvidia GPU. While the actual FPS would not be the same, the relative performance should scale roughly the same to running on the Jetson Nano.

The Mobilenet v2 model is the lightest model in terms of compute power required that we chose, and it scored an accuracy of 0.755 while having an average fps of 20.6. The Inception v2 model is slightly more intense to run and scored an accuracy of 0.729 while having an average fps of 17.19. The Resnet model was the most difficult to run on our system and scored an accuracy of 0.48 while maintaining an average fps of 7.5. These scores can be seen in Figure 43 for comparison.

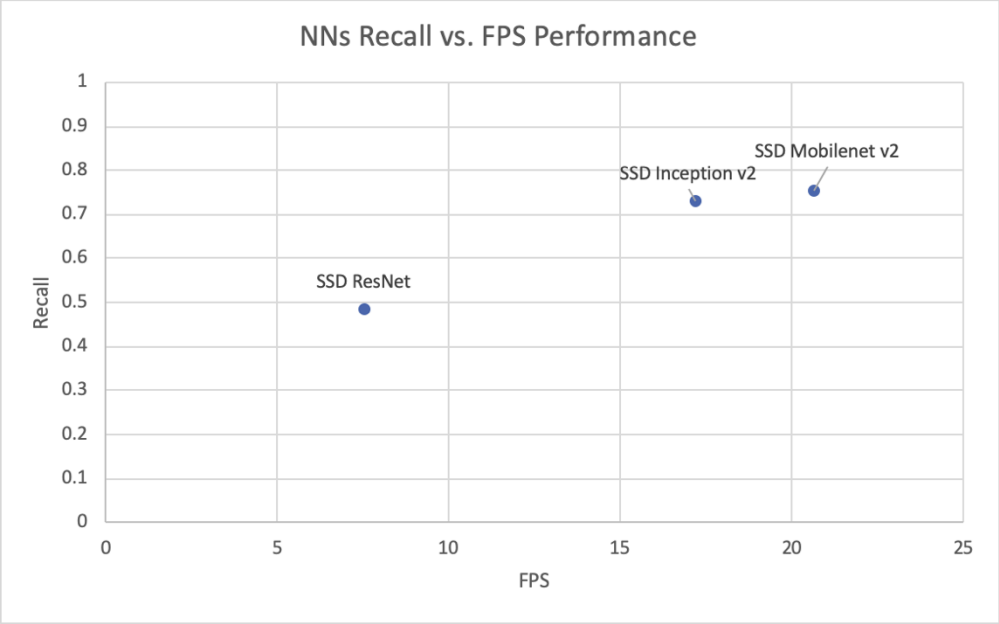


FIGURE 43: FPS VS. RECALL FOR EACH OF THE DIFFERENT MODELS

In the testing of the models some interesting datapoints were discovered. As can be seen Mobilenet had the highest recall score followed by Inception, and then Resnet. We had expected the more complex models to score better than the simpler Mobilenet model at the expense of fps. The fps drop was experienced the more complex a model was, but the recall of the more complex models did not improve. While recall did not seem to improve, the False Positive rate did improve. The Mobilenet model had 265 False Positives, Inception had 189 False Positives, and Resnet had only 152 False Positives. This shows that the more complex models did improve in the incorrect detections, but at the expense of the number of True Positives.

We noticed that the models performed worse when detecting Divers than detecting Lionfish. ResNet suffered the most, 0.26 Diver detection vs. 0.705 Lionfish detection. To improve the performance of the models' performance more images of Divers would need to be collected, especially Divers in many orientations to better generalize the models.

4.3.2 Distance Sensing

After working with OpenCV's StereoSGBM function for a long time in the lab, parameters for the function were tuned enough to produce a depth map in front of the robot that was useable for determining the distance to an object. Once the lab proof-of-concept was completed, calibrating for use in the pool was next. Unfortunately, we had minimal time in the pool to work on calibration of the stereo system. The time we did have in the pool was taken up by the issues in holding the robot still for good calibration data, as well as communicating to the swimmer in the pool as to where to hold the calibration board so the cameras could properly see it. Our team was unable to calibrate in the pool properly and depth sensing in the pool was not completed. Next year's team can expand upon the proof-of-concept lab demonstration and develop a better pool calibration system to enable distance sensing in the water.

4.4 Harvester

4.4.1 Pneumatic System

The pneumatic system is a relatively simple circuit consisting of a double acting cylinder controlled by a 12V DC solenoid with 5 ports and 2 positions. The system features two chambers, one for the supply of compressed air, and one for the exhausted air. The spring-return solenoid is triggered by a relay in the electronics chamber to switch to its second position, releasing air into the back side of the cylinder and extending the piston. Once the fish has been speared, the spring returns the solenoid, and the cylinder retracts. Figure 44 shows a diagram representing this pneumatic system.

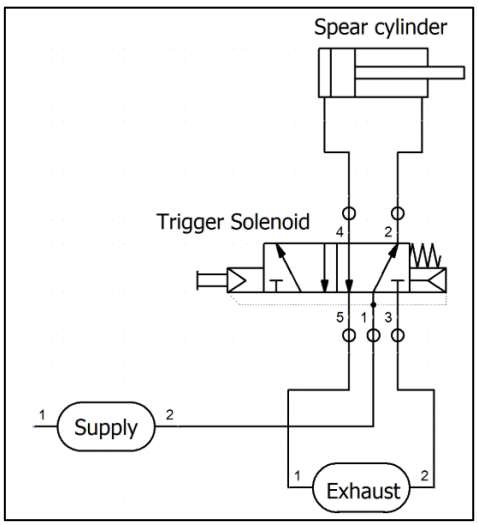


FIGURE 44: PNEUMATIC SYSTEM DIAGRAM

The second chamber was used in place of venting the exhaust ports into the "atmosphere". This is because, at the AUV's operating depth of 100 meters, the atmospheric pressure from the surrounding water is nearly 150psi, while the operating pressure of the pneumatic system is merely 100psi, meaning that the exhaust ports would not be able to vent into the atmosphere, in fact, water would flood in instead. To resolve this, the system vents into an enclosed chamber that can be evacuated between shots.

The primary issue encountered when testing this system was with the exhaust chamber. It was discovered that venting into an enclosed space, resulted in a significant damping effect on the firing of the cylinder, rendering it far too slow to kill a fish. As the exhaust chamber was filled by consecutive shots this effect was magnified. The solution to this issue was to evacuate the exhaust chamber between shots, rather than attempting to continue firing and filling it up.

In order to accomplish the evacuating of the exhaust chamber, a small compressor pump, of the type used to inflate car tires, was placed inside of the exhaust chamber. When activated, the pump pumps air from inside the chamber into the atmosphere outside, thereby decreasing the pressure inside the vessel. The pump selected operates on 12V DC, with a maximum draw of 3 amperes, and is capable of compressing over 150psi, which is required at the maximum operating depth of this AUV. With the use of the compressor pump, the pneumatic system was able to operate as intended and achieved the desired 3 m/s velocity for the speartip, as determined necessary by a previous MQP team.

4.4.2 Spin Spear Design

The spin spear design, as described in the methodology section, features a single brushless DC motor to actuate the entire system, thereby minimizing the need for electrical connections and simplifying the system in the challenging underwater environment. The carriage is supported by teflon-coated bushing riding on anodized aluminum rails. The leadscrew selected for this application was a 0.125" lead, 0.25" outer diameter screw, using a Thomson Linear supernut, meaning that for every revolution of the screw, the carriage travels 0.125 inches. The motor selected was a BlueRobotics M100 brushless underwater motor, which is a 540kv brushless motor capable of producing 2.5 in-lbs of torque. During initial testing, it was determined that the motor did not have enough torque at the slow speeds required for turning the leadscrew. The equations

below were used to determine an appropriate gear reduction for this motor, resulting in a desired 1:7 gear reduction.

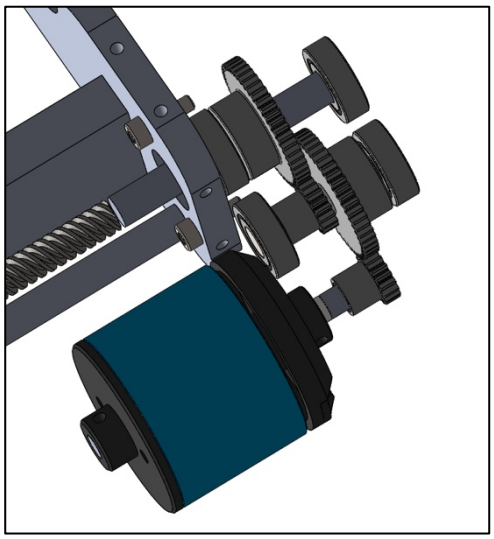


FIGURE 45: 1:7 GEARBOX FOR SPEAR MOTOR

$$motor\ speed = 540\frac{RPM}{V} * 12V = 6{,}500\ RPM$$
$$travel\ distance = 20\ in.$$
$$time\ to\ flip = 20\ sec$$
$$\frac{20\ in.}{20\ sec} * \frac{1\ rev.}{0.125\ in.} * \frac{60\ sec}{1\ min} = 960\ RPM$$
$$desired\ reduction = \frac{960\ RPM}{6500\ RPM} = 0.14\ or\ 1{:}7$$

In order to find appropriate gears, McMaster Carr was used, with the intent of using plastic gears. A 20-degree pressure angle, 48 pitch gear was selected, in order to minimize the size of the gears for ease of packaging. With these constraints, the McMaster Carr catalog was used to find appropriate gears. In order to accomplish the desired reduction, a two-stage gearbox, consisting of two 18:48 gear reductions was designed, which can be seen in Figure 45 and Figure 46. The resultant reduction is a reasonable approximation of the 1:7 reduction desired.

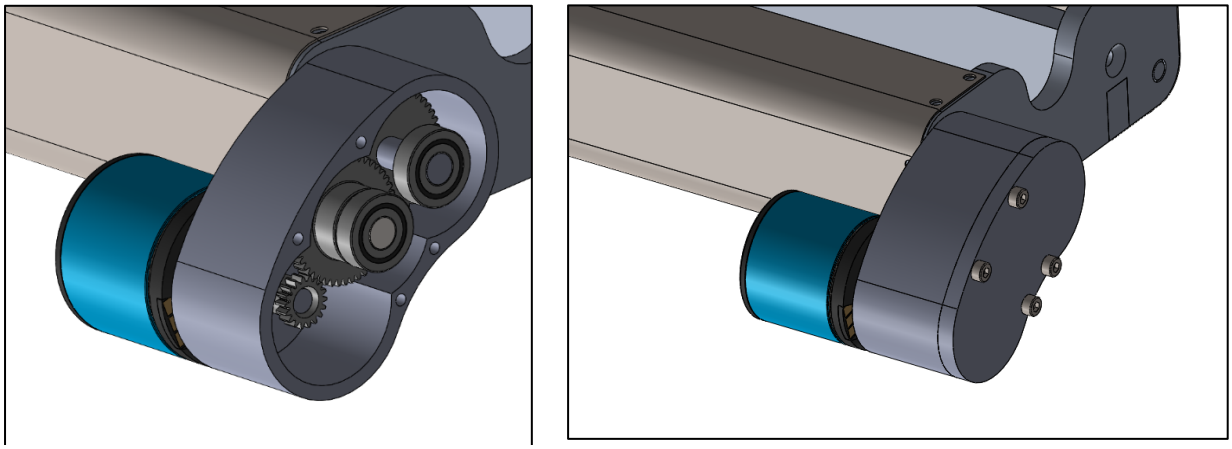


FIGURE 46: 1:7 GEARBOX FOR MOTOR WITHOUT COVER (LEFT), WITH COVER (RIGHT)

For the flipping mechanism, the important factor was the diameter of the gear to ride on the pinion rather than a ratio. In Figure 47 below, it can be seen that the radius of the pinion gear on the carriage is directly proportional to the force required to flip the spear, because the linear force provided by the leadscrew acts on the outer diameter of the gear, and a larger diameter will result in a higher torque to spin the spear. On the other hand, a larger diameter would also result in a large portion of the linear motion being used for flipping the spear, and less linear throw. The selection of an appropriate gear required these two competing factors to be considered, and it was decided to use a 1.5" diameter gear.

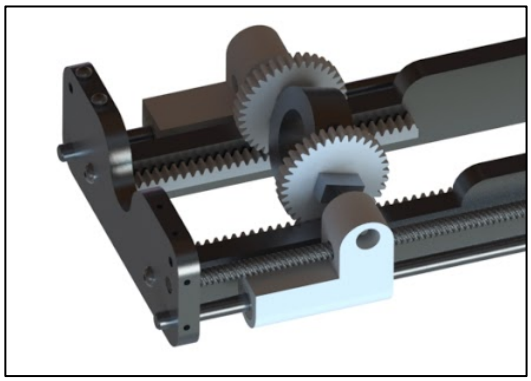


FIGURE 47: RACK AND PINION MECHANISM FOR FLIPPING SPEAR

Once the spear is flipped over by the rack and pinion, the flat spots on the carriage are captured by the flat part of the gantry rails. The friction between the aluminum of the carriage and the aluminum gantries was initially a concern, however through testing it was found to be negligible, and to not have a significant impact on the motion of the carriage. On the other hand, one issue encountered during testing was the droop of the pneumatic cylinder during the linear motion part of the travel. The cylinder is not centered in the carriage, in order to reduce the space needed when flipping the spear. However, also causes a slight torque on the axis about which the entire carriage rotates. This torque in turn causes the carriage to droop slightly when riding along the linear portion of its motion, due to some compliance in the system. This droop can be seen in Figure 48, and causes intermittent issues when attempting to insert fish into the container.

FIGURE 48: SPEAR DROOP PROBLEM

This issue of droop in the spear can be addressed in a few ways for future iterations, of which the simplest would be to use steel rails for the sliders rather than aluminum. One cause of the droop is flex in the rails which allows the carriage to not ride perfectly flat on the gantry rails. While aluminum was originally selected for its corrosion resistance, stainless steel rails would be a better solution, as they would eliminate this flex, and should be used to replace the aluminum ones if the budget allows. Alternatively, or if steel rails do not sufficiently address the issue, a redesigned gantry that captures the key on the carriage on both sides, thereby reducing the room for error, would also address this issue, but would require more design work and fabrication.

One more minor issue that was encountered was that the original design of the gantry, which allowed for 14” of horizontal travel, did not have enough throw to fully insert a captured fish into the container. This is easily resolved by extending the length of the gantry, which, due to the equation-based modeling of the SolidWorks design, required only changing a few variables. The only parts that have to be changed or remade to achieve this extra throw are the gantry rails, as well as the linear rods on which the carriage rides. In order to successfully insert a fish into the container, at least an additional 7-10” of travel would be desirable. In order to address this solution temporarily, we were able to achieve success by simply firing the cylinder a second time to insert the fish, and have already updated the SolidWorks models as appropriate.

The final component of this system are the sensors used to determine where the carriage is in its travel. This problem seems simple at first, however the underwater environment in which this AUV must operate adds some complexity, as simple limit switches to indicate the two ends of travel will not work. Instead, magnetically-operated switches can be used, which are triggered

when a magnetic field approaches the switch. By placing a magnet on the carriage and a switch on either end of the gantry, it is easy to determine when the spear has reached the end of its desired travel. Unfortunately, this functionality was not able to be tested due to the COVID-19 pandemic and the lack of access to the MQP lab, however they would have easily been implemented.

4.4.3 Lionfish Containment Unit (LCU)

Four different designs of Lionfish Containment Units were tested. These designs were based on LCUs that are commonly used by lionfish hunters and modified to work with the robot. The overall results of each design can be seen in Appendix G but more details will follow in the subsequent sections. Although it seems as though there is a best choice based on the results we cannot fully recommend any design. This is due to the many important factors that we could not account for, did not know about, or were uncertain about the importance of factors relative to others. There are also many features and aspects of the designs that could be modified easily and drastically improve their results, such as a change in material or adding a handle.

Design 1: Home Depot Bucket

FIGURE 49: HOME DEPOT BUCKET DESIGN

The first design as seen in Figure 49 resembles a ZooKeeper™ LCU using a five-gallon bucket and lid; both the main body and one-way fish valve are constructed by cutting the plastic of the bucket. The bucket is then attached to the bottom of the robot using two 3D printed stabilizer pieces and pins that go into the bucket. These pins are not attached or anchored to anything and can be removed as easily as they were put in. This results in the bucket being installed and removed very easily, but it could also mean the bucket can unintentionally detach with minimal effort. However, this can be easily modified by modifying the design to use a cotter pin instead.

The main body of the bucket is cut with a few thin rectangular columns and holes to allow for better water drainage and decrease the drag force produced. The bucket comes with a handle that is convenient for a human user as this allows someone to move or empty the bucket while keeping some distance from any venomous Lionfish spines that poke through the bucket's openings.

The lid of the bucket is cut into eight slices and serves as the one-way fish valve for the system. Since the lid is made of a less rigid plastic than the bucket it allowed for our foam test fishes to enter easily, but struggled to clear them from the spear every time. The lid flaps would also be occasionally pulled out of the bucket with the spear. The lid is manufactured for the specific bucket and attaches or detaches with a decent amount of applied force.

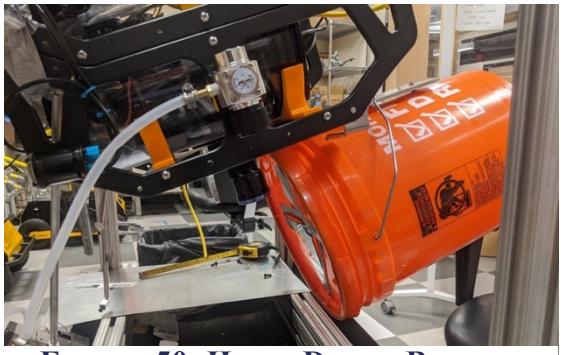


FIGURE 50: HOME DEPOT BUCKET MOUNTED ON AUV

The rigid plastic of the bucket makes the design overall very durable. However, it is due to this rigid plastic and size of the bucket that contributes to a large drag coefficient. Due to this drag the bucket made the robot almost undrivable in the forwards and backwards direction. The strafing drive command of the robot was also greatly inhibited. The yaw rotational motion and vertical ascending or descending motion of the robot were not dampened as much but still affected. However, it is believed that the forward drive commands can be changed to account for the bucket and drastically improve the results. Altering the drive commands should improve the results for all the proposed LCU designs. If altering the commands is not a viable option for the bucket design, one could potentially replace the bottom of the bucket with a strong mesh material to make forward and reverse driving more feasible.

Design 2: Catch Bag with Bucket Frame

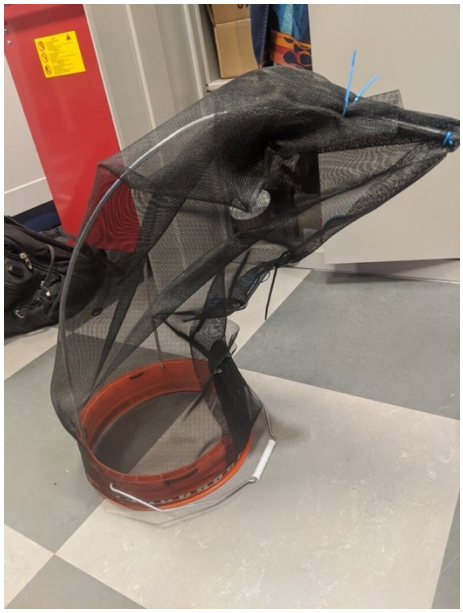


FIGURE 51: CATCH BAG WITH BUCKET FRAME DESIGN

As seen in Figure 51 and Figure 52the Catch Bag design is a modified version of the Bucket LCU. This design essentially takes the top part of the bucket, the lid and two 36 inch aluminum rods to act as a frame for a window screen mesh and form a bag-like LCU. The design is then attached to the metal tailpiece using some screws and nuts attached to a thin plate on the bucket frame. This method of using screws and bolts to attach the LCU caused issues when in the semi-still water of the WPI pool. Although we could not test this it is assumed that in an ocean environment this method of attachment would be almost impossible to manage. However, it is easy to design and make parts to improve or alter this system. There was just little time left in the term for us to make parts for an alternative clamping system.

By using window screen mesh for a majority of the container the drag coefficient on this design is greatly reduced compared to the others. This open design resulted in very little to no impairment in the motion of the robot's basic drive functions. However, this comes with the sacrifice of durability and safety for the handler. Although we could not test this it is believed that the window screen can be easily damaged by either the lionfish's spines or various rocks on the ocean floor. Thus, allowing fish put into the bag to potentially fall out. This can be improved upon though by using stronger mesh or using a similar material such as chicken wire.

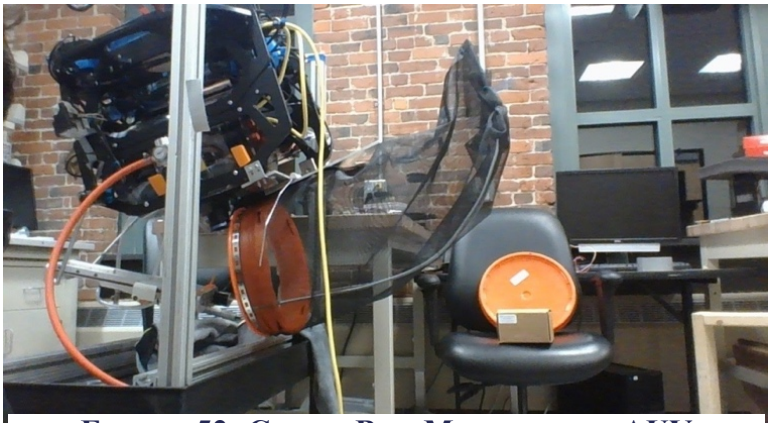


FIGURE 52: CATCH BAG MOUNTED ON AUV

This design performed really well on the robot. However, there are still various things that we could not test that may sway the overall performance of the design such as driving ability when the container is full, how the weight of the fish affects the robot's alignment, etc.

Design 3: Trash Can

FIGURE 53: TRASH CAN DESIGN NEXT TO HOME DEPOT BUCKET

The following design is quite simply a trash can with a lid and holes cut in various places. The trash bin can hold up to seven gallons and is made of a flexible plastic unlike the bucket. As seen in Figure 53, the lid that acts as a one-way fish valve was not made for this specific trash can. The lid used was made for a twenty quart Tupperware™ container and modified to attach to the trash can with screws and bolts. This lid is made of a very thick and strong plastic that is resistant to bending easily. This made it difficult to shove the test foam fish we had into the container, but it guaranteed that the fish would be cleared from the spear tip when removing it from the trash can.

This lid although durable and performs well, does have a large surface area. This combined with the bottom area of the trash can impaired forward motion. To resolve this issue the bottom of the can was removed and replaced with a mesh material (Figure 54). This reduced drag on forward motion of the robot significantly, but due to durability and strength of the mesh this feature was removed in the next iteration of the design. As seen in Figure 55, the final version of the trash can featured holes all around to reduce drag

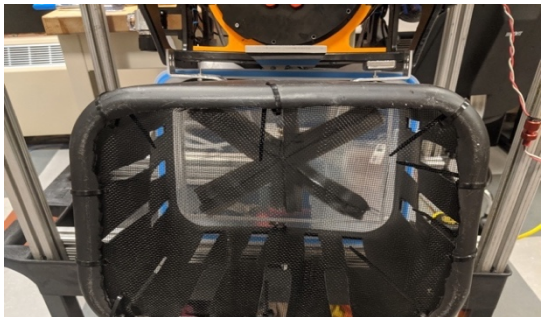


FIGURE 54: TRASH CAN DESIGN WITH MESH BOTTOM

FIGURE 55: FINALIZED TRASH CAN WITH MESH DESIGN

and had a carrying handle installed. The forward motion of the robot was still affected by the lid and bottom area of the can, but still performed much better than the bucket. This is most likely due to the flexible plastic of the trash can, since it bends and flexes under load instead of forcing the robot to move in weird ways.

The method of attachment to the robot is similar to the catch bag. The user slides the trash can onto the bottom of the robot, through a looped zip tie and then aligns a screw with a hole on the tail and then puts a nut on the screw. This method is not the easiest to manage and sometimes the nut is dropped. This is not a problem in a 14ft pool, but in the ocean or anywhere else it would be almost impossible to retrieve the nut. Along with a semi difficult attachment method the holes in the container do increase the risk of being poked by the venomous fish spines. However, this can be reduced by applying a strong mesh material to the outside of the container.

Design 4: Mesh Covered Trash Can

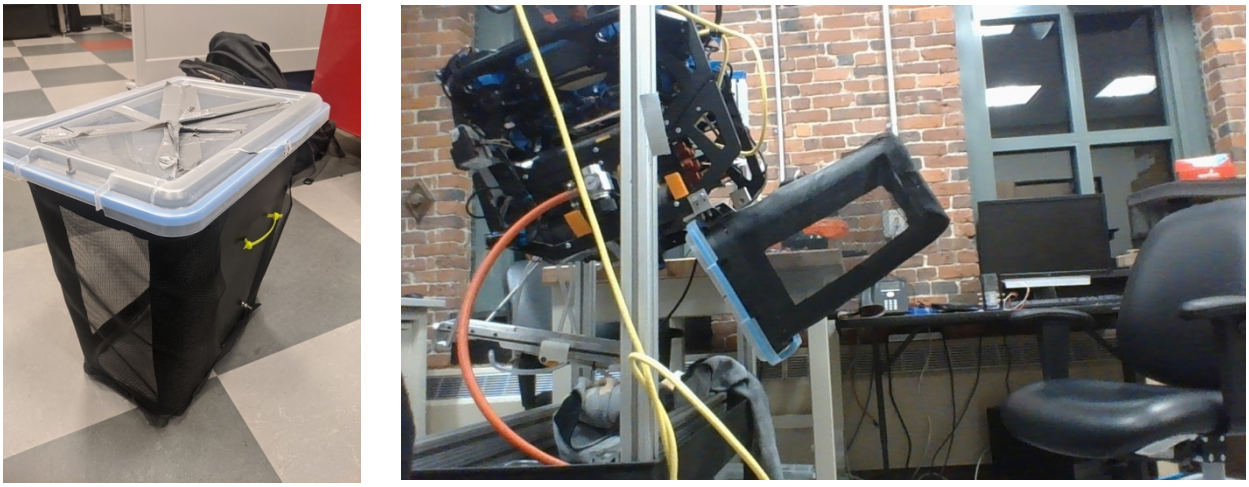


FIGURE 56: TRASH CAN FRAME WITH MESH COVERING DESIGN

The next design is the same as the trash can, but modified so that the window screen mesh surrounds the whole container and almost all of the side and bottom areas are removed as seen in Figure 56. This trash can generally perform the same as the other one, but the basic driving control shows little to no impairment. However, like the catch bag design in terms of durability and handler safety this design scores very low. As with the catch bag even though the default driving functions were hardly impaired, they were not tested with full containers.

4.4.4 Maintaining Proper Buoyancy

The buoyancy of the robot was greatly affected by the addition of the large chamber to house the batteries and electronic boards. As well the side-to-side balance was affected by the pneumatic chambers on the side of the robot as well as the spin spear and LCU in the front and back. To counteract the majority of the buoyancy caused by the main chamber approximately

seven pounds of lead dive weights were inserted in between the batteries to weigh down the system. Along with this we added 3D printed pieces to hold the chamber stationary to prevent it from sliding forwards or backwards. To counteract the sideways tilting alignment of the robot small seven-ounce dive weights were attached to various places until the robot was naturally positively buoyant and level.

4.5 Acoustic Link

4.5.1 Transducer Driver

The acoustic link has seen many iterative changes. A lot of early focus went into researching commercial side underwater acoustic transducers, which on average are fairly expensive. Keeping it within our price range would be difficult to do and because of such, it is more feasible to design a custom transducer. Along with this, the uncertainty of a core protocol or means to control the transducer made initializing the code base and deciding the microcontroller or solution to execute the code difficult.

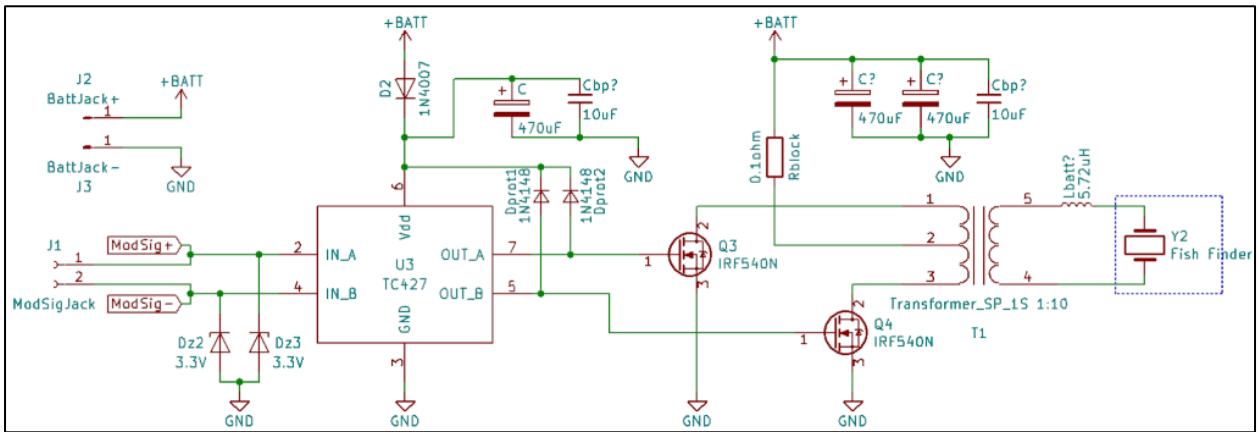


FIGURE 57: FINALIZED TRANSDUCER DRIVER CIRCUIT

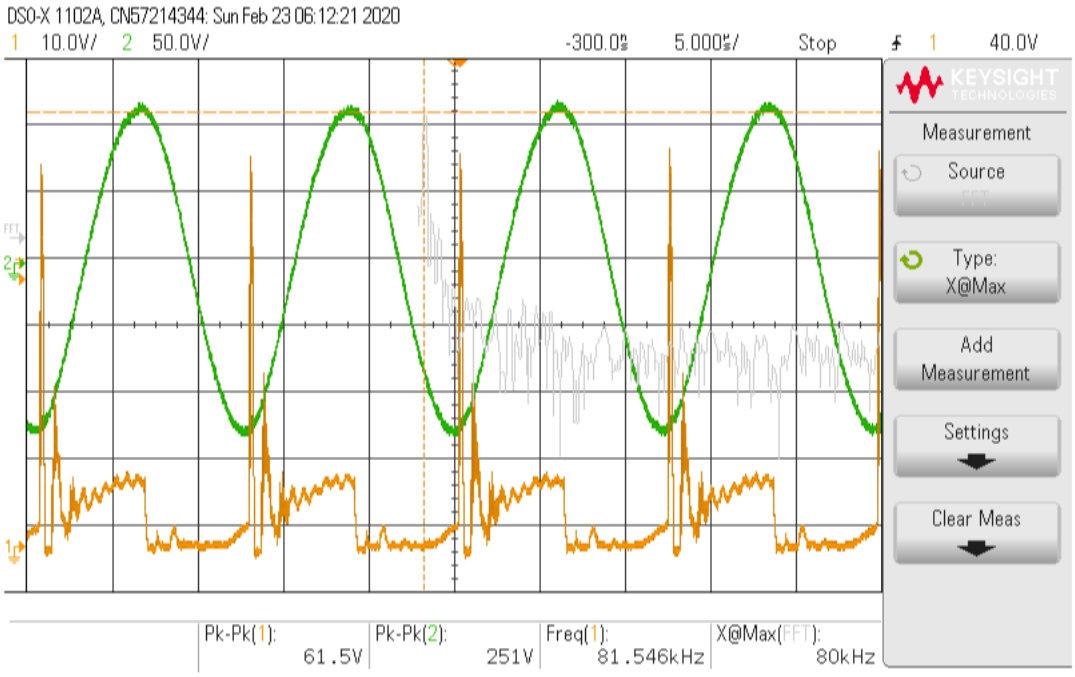


FIGURE 58: WAVEFORM CAPTURE ON TWO SIDES OF TRANSFORMER, CHANNEL 1 - PRIMARY SIDE (ORANGE/BOTTOM), CHANNEL 2 - SECONDARY SIDE (GREEN/TOP)

The finalized transducer drive circuit is shown in Figure 57. Figure 58 showed that with an impedance matching network, the secondary side voltage becomes a very smooth sine wave. This helps to minimize the energy lost in other frequencies which the transducer is not resonating or transmitting.

There is a problem with the gate driver. Because of the high voltage spike and dips on the power rail, the TC427 gate driver has been damaged due to overvoltage quite a few times during the development of the circuit. Also, when the gate driver is damaged, it is not open short, instead, the input pins will be short to power. This means when TC427 is damaged, it is likely to damage the microcontroller as well. In fact, one of the PWM channels is no longer functioning during the test due to this problem. With a stabilized battery voltage, this should not appear again. However, there are also many other power-hungry parts running directly off of the battery, such as propellers and pumps. To ensure a robust operation, it is necessary to implement over-voltage protection or choose a gate driver with such feature for the next revision of the design.

4.5.2 Receiver Amplifier

The final design of the receiver amplifier circuit is shown in Figure 59. After adding the pulldown resistor on the anode of the diode, the envelope detector is now working. The first and second stage of the amplifier work as well. When testing at 100V/V, 80MHz, the phase shift between input and output of each amplifying stage is well below 90°. This means when two stage are combined, there are plenty of phase margin before oscillation (180° phase shift). The output also has the desired DC bias which is the same as Vref (Figure 60). When the two stages are connected, there are no oscillation on the output, and the DC bias from first stage is filtered and doesn't get picked up by the second stage.

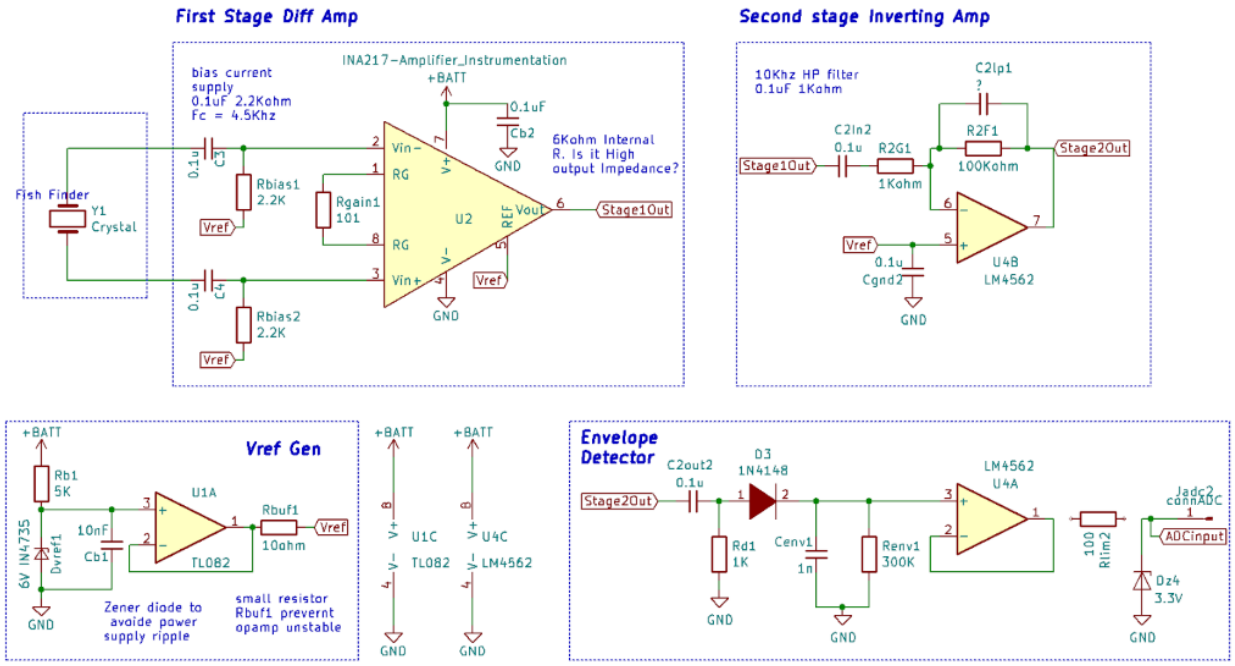


FIGURE 59: RECEIVER AMPLIFIER CIRCUIT

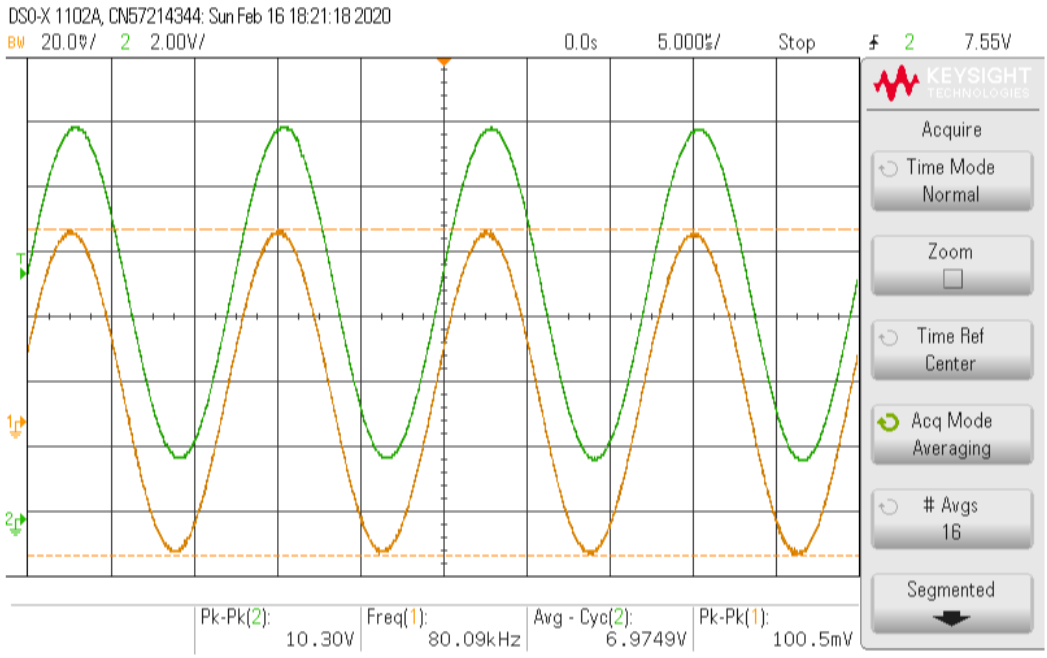


FIGURE 60: FIRST AMPLIFIER STAGE WORKING WITH OUTPUT BIAS

In theory, the fish finder transducer has a very narrow band; it should not produce anything apart from the received 80Khz signal. However, when both stages are connected, and the fish finder transducer is used as the input, there is quite a bit of noise on the output of the first stage.

This noise also gets amplified by 40dB by the next stage and thus produces a fully saturated output from the second stage. This problem is shown in Figure 61. With the human eye, it is possible to separate the modulated signal since it is saturated while the amplified noise is not. However, this result will cause the envelope detector to be saturated for all time, thus not usable by the software.

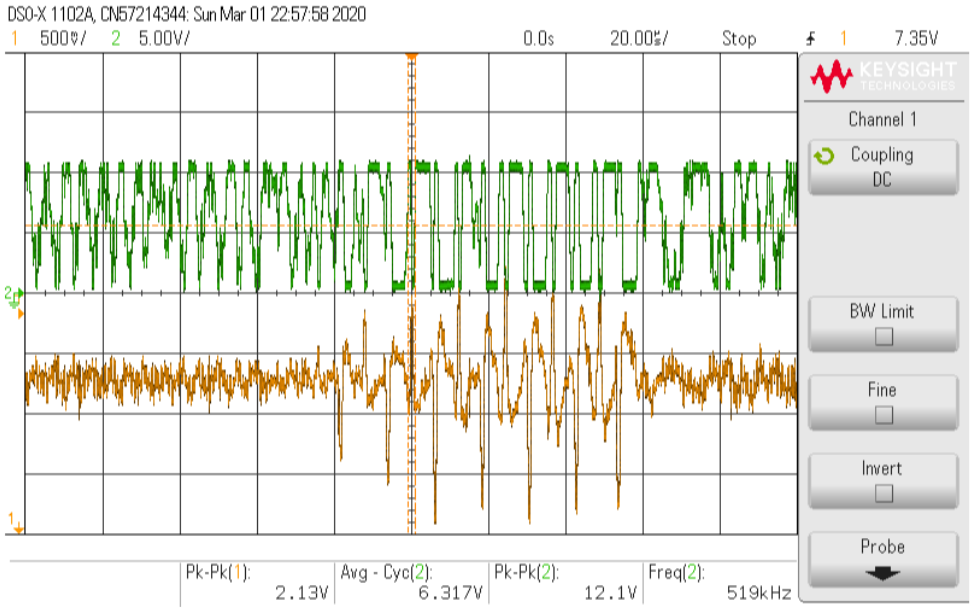


FIGURE 61: AMPLIFIED NOISE RESULTS IN UNUSABLE OUTPUT, CHANNEL 1 (ORANGE/BOTTOM) - OUTPUT OF FIRST STAGE, CHANNEL 2 (GREEN/TOP) - OUTPUT OF SECOND STAGE

There are multiple possibilities for the cause. The noise could be wirelessly coupled into the circuit from another circuit next to the amplifier. Since the test is conducted in a small bucket, this could also be the long-lasting echo of the previous transducer ping which also gets picked up by the amplifier. Also, the transducer is likely to have a second resonance point at a much higher frequency. When the transducer driver does not have the impedance matching network, there is a high content of 6Mhz signal on the secondary side. Since the test is done in the bucket, that 6Mhz noise might also get transmitted through. Due to the limited resource and time, the frequency content of this signal cannot be measured. Thus, this will need to be examined closely in the future.

4.5.3 Implementation of Communication Protocols

The team elected to use a symmetrical code base system where the support boat can execute on the same code that will be on the AUV node. This is helpful because the amount of work is significantly reduced. With a symmetrical system, the acoustic link can propagate its received and sent information with the AUV or boat users in the exact same way, and the true difference is reduced to what each node will do with information Acoustic Link passes on.

For the support boat at sea level, this information is purely for display, and the users can send a command status, or a request for more information. In the case of the AUV, the acoustic link will pipe all of its received information internally through Ethernet to the Jetson Nano, which will handle commands accordingly.

In order to ensure each message is being sent and received properly, the same CRC error detection routine was implemented on both sides. CRC, or Cyclical Redundancy Check is an error

checking algorithm that implements polynomial math and modulo division to process a message for any errors. Using a specified CRC generator polynomial, each node can check each message for errors constantly. CRC packet information was added to the least significant eight bits of our protocol's packet structure (Figure 62).

The packet structure of our communication systems protocol is very similar to common serial communication and IP protocols with error detection, message acknowledgements and message identification frames. Because this communication system is very sparse, and only used for very vital commands, the protocol was adjusted for more appropriate specifications.

The data buffer is a set of variables which are stored on both the master boat and the autonomous AUV. In theory, this will be passed along to the rest of the AUV through the Jetson Nano, which will then decide how to respond in the case of a command status, info, or request message. We choose this structure because of the simplicity to parse and obtain the correct information. We also choose this because it allows for more info data points if needed, and more vital command status in the future. A simple ACK packet signifies that the opposite node in the communication network has received the message previously sent. The ACK variable in the data_buffer will contain the ID of the information it receives. On the next page is a flowchart of the procedure each node should follow when operating (Figure 63) and an example communication protocol showing the case of a corruption (Figure 64).

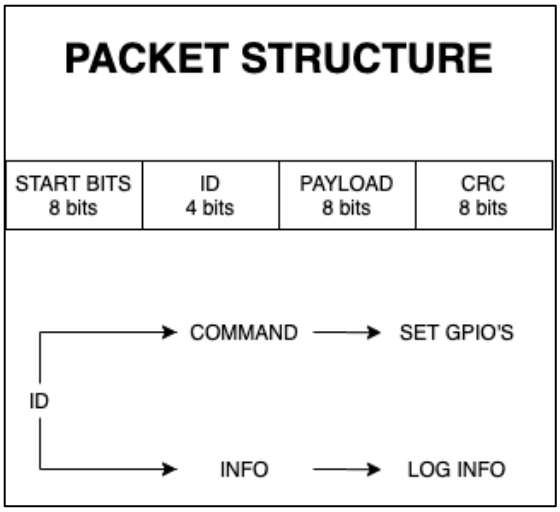


FIGURE 62: WIRELESS COMMUNICATIONS PACKET STRUCTURE

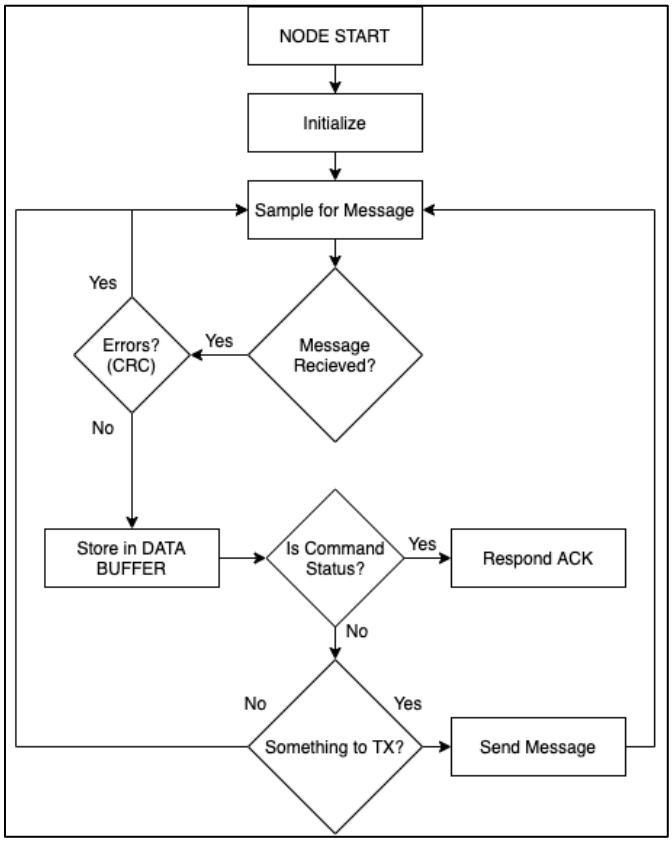


FIGURE 63: COMMUNICATION PROTOCOL FLOWCHART

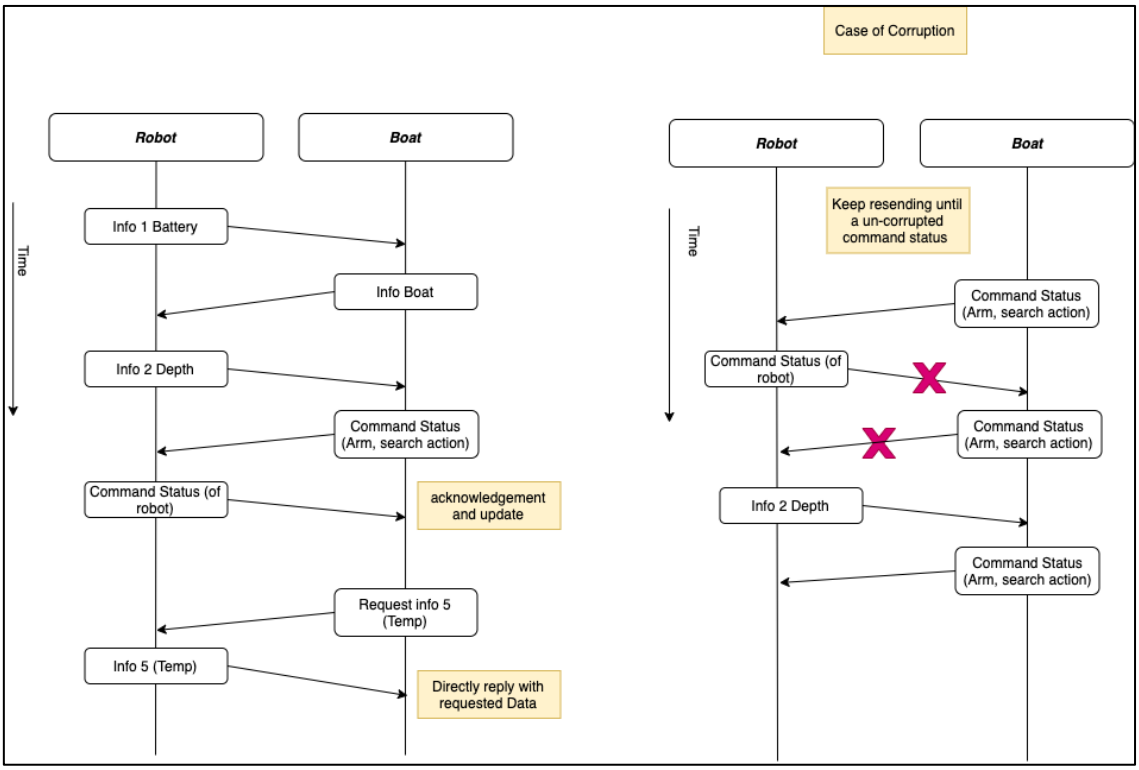


FIGURE 64: EXAMPLE COMMUNICATION PROTOCOL WITH POSSIBLE CORRUPTION

4.5.4 Embedded Code

Our embedded code is based off of code that was used for the Tiva Series TM4C129XL from Texas Instruments. This board features General Purpose Input and Output (GPIO) , Ethernet, Pulse Width Modulation (PWM) modules, Analog to Digital (ADC) converters, and many more useful libraries in C. The board from Texas Instruments is powered by a 120MHz 32-bit ARM Cortex-M4 CPU. Taking the template clean project from one of our prior classes, Real-Time Embedded System, we added and modified the template into our source code. A development board is needed on both ends of the communication system to properly excite the transducers, as well as process incoming and outgoing messages. The use of the Tiva Series board is necessary because without it, the AUV's external communications code base would need to live on one of the other boards already used for the other various subsystems, which would eat into the performance of the other subsystems.

Chapter 5: Discussion

With the conclusion of our testing and results for the lionfish harvester, it is important to discuss the direction of the project from the state in which we have left it. Due to the coronavirus pandemic of 2020, our project was cut short, leaving some of our plans for the AUV unfinished. However, the team believes we have created a solid platform for future MQP teams to augment and enhance in a multitude of ways. In light of this, we have left future groups a 'Lionfish Harvester User Guide, which discusses how to use the AUV's software and hardware, the structure of the system, and our processes for testing the robot. The following section dives deeper into what we had plans for and implementations that are critical for the success of the lionfish harvester.

5.1 AUV Concerns & Future Implementations

Across the different subgroups, there are many implementations for future groups to embark on. Some of these include functionality that we did not get to and applications that were out of scope for our project. Each of our recommendations can be found by sub-system below.

5.1.1 Navigation

Currently, we have implemented the basic building blocks of movement as well as an advanced search pattern that combines these together, but there is much more to do. In the future it is important for the team to add more advanced movement, especially that with higher intelligence navigating the known surrounding area. There are multiple ways to do path planning, especially in an underwater environment, but these are essential in order to effectively search for and eliminate lionfish on a reef. In addition, it would be smart for a future team to finish integrating as well as expanding our emergency situation handling, allowing the AUV to recover from anomalous behavior. This includes loss of communication, leak detection, program crashes, and more. With our current software, integration should not be too difficult due to the separation and control of processes that is provided, but it is imperative that more emergency and error handling be added to ensure the safe operation of the AUV.

One of the issues with navigation is the overshoot that occurs. To complete more precise control in its maneuvers, the AUV should have more precise control systems implemented. The existing maneuvers hard stop the motors which causes the AUV to overshoot its destination point. Introducing deceleration techniques could be a point of attack for a future group. Likewise, in an ocean environment, there are inevitable currents that will cause the robot to overshoot its target. Implementing PID control systems, or a more robust 'stabilize' mode would be an essential task before putting the AUV in a large body of water.

Since the sensor network that we developed is somewhat elementary, one of the recommendations would be to implement more external sensors for better localization and object detection. Using more sonar devices or a better combination of them could enhance the overall navigation of the AUV. Another possibility is the use of the cameras for more extensive object detection. With the initial development of a depth map created by the cameras, it would be promising to extend the camera use into obstacle detection, not just lionfish and diver detection. The acoustic link could also be a source for localization. Utilizing a triangulation technique, the transducer could be used to determine the location of the AUV relative to the boat in a local GPS

system. There are many opportunities for the enhancement of the navigation software and components.

5.1.2 Identification

Future iterations of the project could make several improvements on the identification system. First, the depth map system works on the lab table, but we did not have enough time for calibration and testing in the water. A future team can take the existing proof-of-concept code and work with it for water calibration and distancing. This would involve a better system for calibration in the water as our team had difficulty in stabilizing the robot and swimmer with the calibration board.

An improvement to the neural network model could also be made. The model that our team trained did not perform as well as we wanted in identifying divers, so more diver images are needed. As scuba divers can be in many orientations and lighting conditions we believe more images of divers are needed to perform better at identifying them. Also, for the model, our team converted to Tensorflow Lite for the sake of time but converting the model to TensorRT would yield better performance out of the Jetson Nano board.

To target a Lionfish, the identification system must identify one, as well as pass that data along to the navigation team. A future iteration of the project needs to compute frame transformations, including both translations and rotations, from the identification location within an image to the navigation coordinate frame to properly pass the targeting data to the navigation program.

Another aspect of identification to be considered is object avoidance. This would be similar to how other autonomous vehicles work by using cameras and machine learning algorithms to detect objects around the vehicle and plan their paths accordingly. Object avoidance will have its own obstacles to overcome, including a possibly stronger processor and extra cameras placed around the AUV, as well as the data collected for training a model.

5.1.3 Harvester

The harvesting would mostly benefit from continued testing and refinement. The overall system worked well, and the gantry was able to successfully flip the spear and slide it backwards in the desired linear motion, however some elements could be improved. Adding a gearbox, extending the length of the linear travel, and adding sensors are all important elements that should be incorporated in future years. The pneumatic system works as a self-contained system, successfully venting exhaust air into the atmosphere around the robot, but has yet to be tested, due to the unfortunate circumstances in the spring of 2020. Multiple containment units were constructed, and further testing should be performed in order to make a final decision on the best design for this robot.

In order to successfully store captured fish in the container behind the robot, the gantry needs to have more distance for linear travel, in order to be able to fully insert the fish into the container behind it. Furthermore, the motor selected for driving the gantry was not as strong as expected, and should be paired with an appropriate gearbox in order to provide more torque at a slower speed for flipping the spear. Both changes have already been implemented in the SolidWorks model of the robot, and, by the modular, equation-driven design of the gantry, are

easy to change. In order to better control the motion of the spear, it would also be beneficial to implement sensors on both ends of the travel to indicate when the carriage has finished moving. This is most easily done by placing a magnetic reed switch on either end, and a magnet on the carriage. When the carriage reaches the end of travel, the switch is activated by the magnet, similarly to how a push-button switch would activate, but more suitable for the unique underwater environment.

The pneumatic system was successfully tested at the surface, and operates as a functionally self-contained system, using compressed air from a canister, and venting into another enclosed chamber, which is evacuated in between shots. The operation is currently all manual, and control should be added in the form of a pressure transducer in the exhaust chamber to control the vacuum pump, and a microcontroller to control the firing action of the solenoid valve.

Multiple lionfish containment units were created, using a variety of different containers, as well as different end caps. These were tested for a variety of parameters, such as maneuverability underwater and capacity, and a matrix was made to compare. This can be used to determine which container is most suitable for the applications of this robot for future use. This matrix can be found in Appendix G

5.1.4 Acoustic Link

The wireless communication system can be improved by future groups. The driver of the transducer should function without much problem. However, as mentioned in the Methodology and Results sections, the power rail filtering might need some improving to prevent the voltage drop when the transformer first turns on causing other devices to restart. When the robot is deployed under water, any component failure will cause a communication loss. Currently the MOSFET gate driver fails quite often due to unstable rail voltage. Thus, it is important to put in more robust protection on that circuit.

On the receiver side, the noise between the two amplifier stages is yet to be solved. The noise could be from the environment. However, the amplification picking up the tiny bit of echo in close range operation might be another factor. Lastly, this might also be the noise from the first stage output saturating. When saturated, the output sine wave gets clipped and the sharp corners created other frequency content. This leads to the other point of variable gain control. In the ideal case, the amplified signal reaching the microcontroller should always be switching between 0 and 3.3V. Currently, this is achieved by simply clipping the high voltage. Since the received signal level change happens gradually as the robot move further away, the microcontroller could use the average voltage level received to control the amplifier gain for the next receiving cycle.

As far as two-way communication is concerned, the building blocks are in place. Currently, with wire communication across the GPIO pin for sampling incoming messages, and a temporary debug pin set as the PWM module, communication is at one-way at best. With additional time to debug and trouble shoot, two-way communication with wires is the first step to full underwater acoustic communication with piezo electric transducers.

Chapter 6: Conclusion

Lionfish are a major environmental issue in the Caribbean, that has been addressed in many different ways, so far with limited successes. This year's project laid a very solid groundwork for future years to continue development of a fully autonomous underwater vehicle as a novel approach to eradicating this pest from the waters it has invaded. The large scope of this project was divided amongst the team into the four major subsystems of identification, navigation, harvesting and communications, and significant progress was achieved in each element.

The identification system's focus was on using a convolutional neural network and machine learning to consistently identify lionfish in the wild. In addition, it is equally important to accurately identify SCUBA divers, in order to avoid them and ensure safety to humans. The team classified over 10,000 images of divers and lionfish, and compared multiple different pre-trained models to determine which would work best in this application. The results are a solid foundation that can be augmented with additional data in the future to create a more accurate model.

The navigation of an underwater environment is a challenging problem, that was reduced to a simpler task for this year's project. The focus on a flat, sandy bottom with some protruding obstacles was simulated in the WPI swimming pool, with people acting as the obstacles to be avoided. The robot was able to successfully follow a simple obstacle avoidance program to cover area in the pool. Future work can be done especially in localizing the AUV within its environment, in order to create a map to follow when searching for lionfish. Additional sensors, such as potentially using the identification cameras for navigation as well, can be used to allow the robot to navigate the more complex environment of an underwater reef.

This year's harvesting mechanism featured a complete re-working of previous ones, utilizing a pneumatic cylinder and unique gantry mechanism to spear fish and store them in an on-board lionfish containment unit. The device was built and some preliminary testing was accomplished with very promising results, but further development was unfortunately halted due to the COVID-19 pandemic. Future teams can implement some refinements that have already been designed, and simply need to be manufactured.

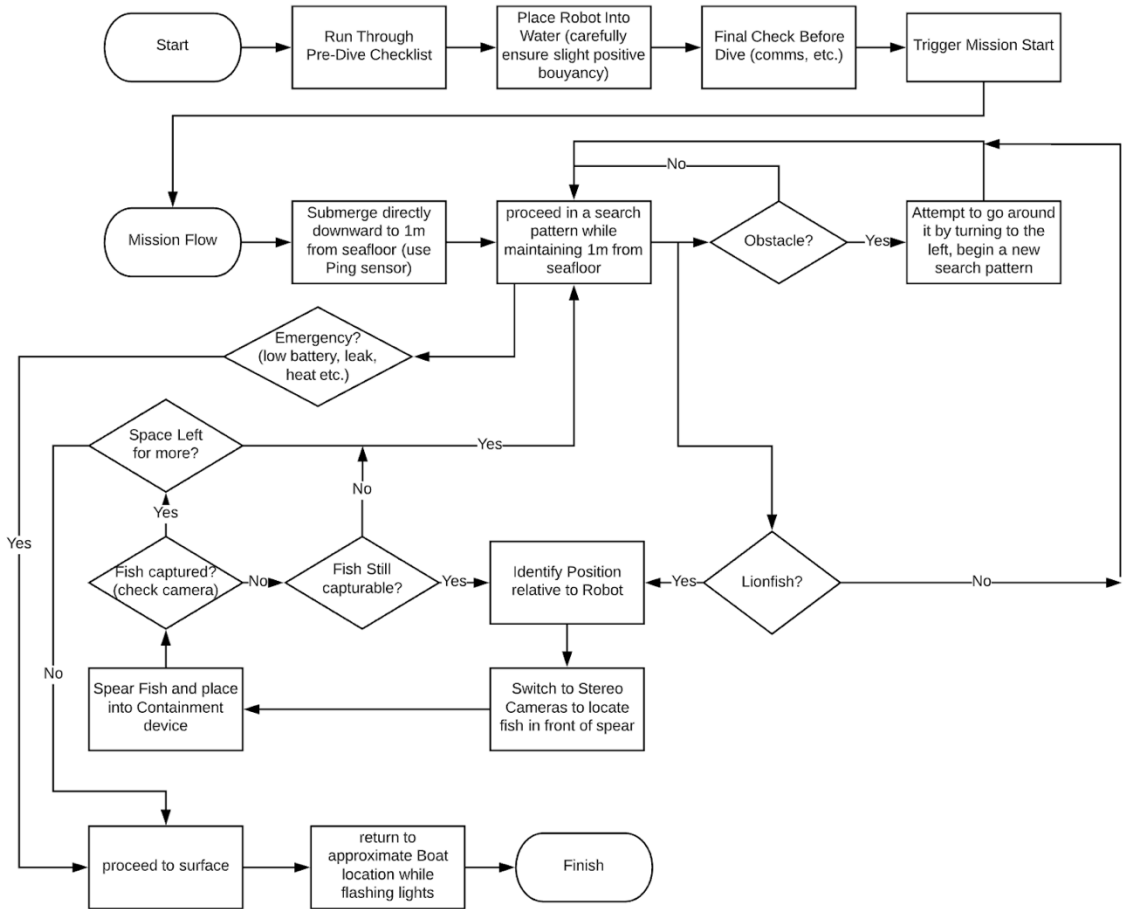
Perhaps the most overlooked portion of this project is the acoustic link communication device. This is critical in establishing a link between a mothership and the AUV in order to send basic status updates and commands, like an emergency call to return home, sent from the ship to the robot. The team was able to develop a protocol and create the transducers needed for the communication system, but further testing couldn't be completed due to the COVID-19 pandemic.

Future teams should focus on developing more advanced navigation algorithms to take advantage of stereo vision depth mapping, as well as other additional sensors. Additional work can also be done in testing the harvesting mechanism and acoustic link in order to develop refinements and ensure that they will be fully operational in the extreme conditions in the ocean. Finally, the identification system would benefit from additional images, especially of SCUBA divers, more development of the depth mapping, and locating lionfish relative to the robot in real space.

Overall, given the circumstances, the AUV was able to operate in rudimentary ways and be tested in a pool environment. With the implementation of basic movements and search patterns, a chosen neural network for lionfish identification, a harvesting unit, and the design of an acoustic communication system, we feel we have created a solid platform for future groups to augment and continue with. In the third year of the project, there has been significant improvement towards the goal of testing the AUV in ocean environments. The lionfish harvester can hopefully continue on its journey to help solve an accelerating environmental issue in the Caribbean Ocean.

Appendix A

A generic mission protocol flowchart for the AUV. It includes decision points, feedback, and safety checks.



Appendix B

The following table is a list of the possible pre-trained neural networks that we could've used for object identification and classification purposes. It includes the model name, it's processing speed, a measure of speed-accuracy balance, and the output type (Kulz, n.d.).

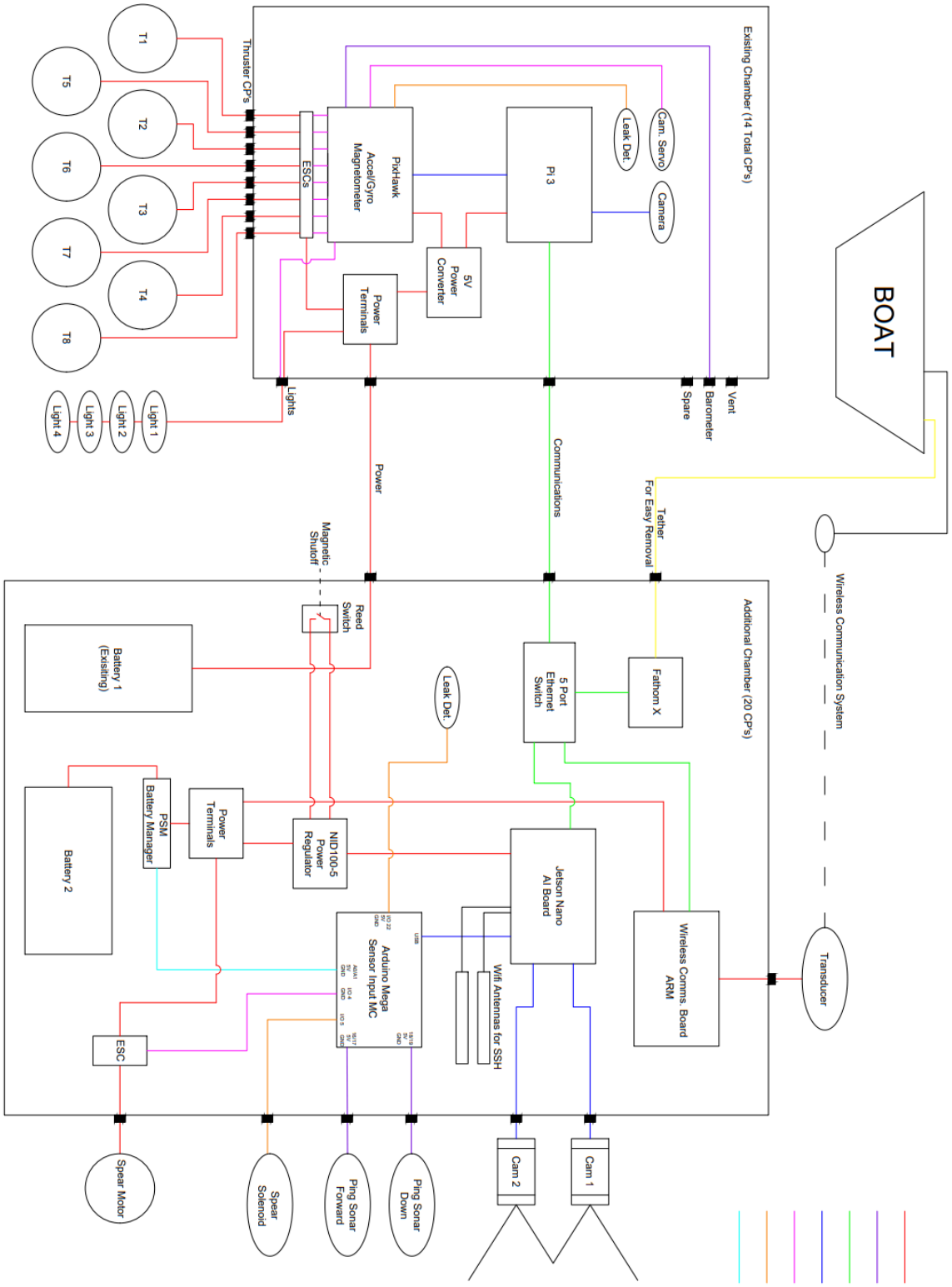
Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes

faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes
faster_rcnn_nas_lowproposals_coco	540		Boxes
mask_rcnn_inception_resnet_v2_atrous_coco	771	36	Masks
mask_rcnn_inception_v2_coco	79	25	Masks
mask_rcnn_resnet101_atrous_coco	470	33	Masks
mask_rcnn_resnet50_atrous_coco	343	29	Masks

Appendix C

Freq.	Transducer Impedance	Transformer Impedance	Z match	Inductor L $\frac{Z}{2\pi f}$	Capacitor C $\frac{1}{2\pi f Z}$
kHz	Im{z}	Im{z}	Im{z}	(mH)	(μF)
100	-1226.784602	48.71574593	1275.50	2.0300218	
90	-1630.713043	43.88976178	1674.60	2.9613479	
85	-2126.625123	41.62627599	2168.25	4.0598579	
82	-3103.700447	40.18061775	3143.88	6.1020025	
81.5	-2838.915436	39.9757117	2878.89	5.6219602	
81	-1476.393263	39.68115078	1516.07	2.9788979	
80.9	-1103.432362	39.72216596	1143.15	2.2489331	
80.7	-428.3547206	39.58774454	467.94	0.9228668	
80.6	45.73565264	39.5646998	-6.17		0.319987389
80.5	311.9705186	39.46191791	-272.51		0.007255111
80.4	642.6064596	39.38825024	-603.22		0.00328163
80.3	821.451631	39.33660194	-782.12		0.00253416
80.1	1146.801378	39.16688586	-1107.63		0.001793871
80	1199.004361	39.19804665	-1159.81		0.001715318
79.7	1282.51762	39.01132516	-1243.51		0.001605883
79.5	1212.041545	38.81736514	-1173.22		0.001706365
79	929.0677121	38.7249726	-890.34		0.002262746
78	389.9198565	38.19561513	-351.72		0.005801272
77.5	199.2939428	37.8036145	-161.49		0.012716626
77.3	129.7450344	37.71306557	-92.03		0.022371851
77.1	64.35936898	37.77359786	-26.59		0.077645536
77	37.61936713	37.71679755	0.10	0.0002013	
76.9	8.99275503	37.66650848	28.67	0.0593442	
76.8	-16.50576098	37.53902909	54.04	0.1119986	

Appendix D


BOAT
Wireless Communication System
Transducer
Existing Chamber (14 Total CP's)
Thruster CP's
T1
T2
T3
T4
T5
T6
T7
T8
ESCs
PixHawk
Accel/Gyro
Magnetometer
Pi 3
Leak Det.
Cam. Servo
Camera
5V
Power
Converter
Power
Terminals
Lights
Light 1
Light 2
Light 3
Light 4
Spare
Barometer
Vent
Power
Communications
Tether
For Easy Removal
Magnetic
Shutoff
Reed
Switch
Additional Chamber (20 CP's)
Battery 1
(Existing)
Battery 2
PSM
Battery Manager
Power
Terminals
NID100-5
Power
Regulator
Leak Det.
5 Port
Ethernet
Switch
Fathom X
Jetson Nano
AI Board
Arduino Mega
Sensor Input MC
Wifi Antennas for SSH
Wireless Comms Board
ARM
ESC
Spear Motor
Spear
Solenoid
Ping Sonar
Forward
Ping Sonar
Down
Cam 2
Cam 1
Power
Serial
Ethernet
USB
PWM
Digital I/O
Analog Input

Appendix E

Pre-Flight Checklist

Hopefully you have a good understanding of the current system and are able to go to the pool and test it out. Before pool testing though, or any underwater testing, it is important to check for all of the following in order to maintain your sanity and the safety of the AUV. We had a few close calls with water flooding the chambers and had our fair share of connection and communication issues. This is the way we did our testing, but hopefully you can build off of it and make it more efficient and easier to manage.

1. **Charge the batteries.** They take longer than you think to charge, so make sure they are at a capacity that works for the testing you're doing that day. Obviously, the motors use the most power, so if you're doing navigation/motion heavy testing, then have the batteries charged to full capacity. BlueRobotics recommends:
    - Avoid discharging below 3.0 V/cell (12.0 V)
    - Never discharge below 2.5V/cell (10.0 V)
    - Never charge over 4.20 V/cell (16.8 V)
2. **Set-up the chamber components.** We created a frame that can easily slide in and out of the large chamber. A lot of the components are glued onto the frame and are strategically placed to make wired connections. Make sure that the devices and and connections are secured - they tend to move when the AUV is in motion and can cause connection and communication problems.
3. **Insert chamber frame and components.** Insert the frame first and make connections to the penetrators on the appropriate endplates. The overall wiring diagram can be seen in Appendix A. Don't screw the end plates on until you test communications (which is the next step). Most of the connections are solid, however the ping sensor wires caused a lot of issues. Make sure that you connect the proper wires to each other and tape them. The connections are color coded using zip ties. Also insert the 2 batteries and plug them into the appropriate XT90 connectors. We decided to have one battery power all of the existing components on the BlueROV2 and another to power all of our added components. This kept the two systems ipower independent of each other.
4. **Test communications.** Run *process.py* or *test_run2.py* from your command line once connecting to the network via the tether. The *process.py* program works for most of the motion controls and grabbing ping data, however, we got cut off by COVID-19 so we are unsure of how far along we were since we weren't able to test. You may have better luck using *test_run2.py* which is our first edition of the software. It's great for testing different commands, but it isn't as robust and shouldn't be used for an overall mission system. Once you can run the program, pull some ping sonar data, try a few motion commands, and run anything you plan on testing in the pool. Make sure everything seems up to par.
5. **Attach both chamber endplates.** GREASE the endplates and any penetrators that were added or tampered with from the last testing. Make sure the screws are tight and the penetrators are snug. This is critical because pressure testing takes a long time to confirm that there are no leaks.
6. **Pressure test both chambers.** There are 2 vacuum pumps. Hand pump to approximately 10-15 inHg and monitor for about 10-15 minutes, making sure that the pressure does not decrease by more than 1 inHg. If decreasing rapidly, double check any new penetrators, use more grease, check if hairs, wires, or debris are on the O-rings.

7. **Re-Test communications.** While pressure testing, test the communications again. Wires can easily disconnect and connections can become loose when moving things around and putting end plates on. If communications and pressure testing passes, then you should be done in the lab.
8. **Transport to the pool.** We had a lot of issues, but ultimately it comes down to having a car that can carry the cart and robot without disassembling anything. Remember to bring everything possible: screwdrivers, tools, vacuum pumps, towels, gopro/camera for video evidence, battery charger, tether/associated materials, backup sensors and wires.
9. **Pressure test again at the pool.** This is not as critical, but just to be safe.
10. **Conduct your pool testing.** Drop the robot in the pool. If you've added components, the weights and buoyancy foam may need to be adjusted to maintain a slightly positive buoyancy. It is also helpful to have someone in the pool who can wrangle the robot during testing. Take notes and videos of everything you do. We also wanted to mention that connections and components get jostled when in the water. If you lose connection with the robot, try power cycling the components: on the side of the large chamber, there is a small, clear reed switch that powers down the computers for a second or two. Put a strong magnet (there are a few lying around in the lab) to the reed switch and you should see the LED on the 5V power regulator shut off and turn back on. This indicates that the Jetson has been power cycled and you can terminate whatever you're running and restart it. This typically allows us to regain connection with the robot. This happens a lot when debugging and updating code, so the reed switch allowed us to quickly reboot everything without opening the chamber up.
11. **Wash off AUV in the showers.** When done testing, pull the AUV out of the water (watch the harvesting mechanism and ping sensor). Chlorine is damaging to a lot of the materials, so it just helps to preserve the AUV as much as possible. Dry off after use too.
12. **Bring back to the lab and disassemble.** Make sure the AUV is dry before pulling out any electronics. Unplug the batteries and store everything safely.

Appendix F

The Questionnaire

The questionnaire is as follows and the actual questions are bolded for ease of reading. Anything not in bold is either an example or a potential answer to the question.

1. **What is the intended purpose of the robot?**
    a. **Is its intended use morally justifiable?**
    b. **If the robot is intended to kill something is it moral to kill whatever you are killing?**
2. **If the robot is intended to kill, is it necessary to kill the subject?**
    i. Example: farming, self-defense, military dominance, etc.
    a. **If the subject is not killed immediately what then happens to the specimen?**
        i. Example: Live capture then released.
        ii. Example: Live capture for specimen identification then killed by human.
        iii. Example: Wounded and left for natural selection to decide one's fate.
3. **Is lethal force simply a byproduct of the design?**
    a. Intentional, semi-intentional, or not intentional
        i. Example: Giant robot arm, swings and kills person. (not intentional)
        ii. Example: Robot aims at and shoots human (intentional)
        iii. Example: Robot's mission is to disable vehicle, but people are injured in the process. (semi-intentional)
4. **Is lethal force the primary method of achieving the desired task?**
    i. Example: Security Robot – detain first, but lethal if need be. (Secondary method)

ii. Example: Farming/ Food source preparation (Primary method)

5. **If intended to kill, is the methodology of killing humane/ non-torturous?**
    i. Example: Suffocation, drowning, blood loss, etc. (Not Humane)
    ii. Example: Sedated, numbed, severed brain stem, etc. (Humane)
6. **If the robot has lethal capabilities what/who is doing the killing? Who is in control of the robot and determines where and how it is used? Does the end user have or need special licensing/training to own the product? Is it legal for them to possess the robot?**
    a. Robot operates autonomously and makes its own decision.
        i. Example: Fully autonomous car
    b. Robot is operated by a human.
        i. Example: Remote controlled drone.
    c. Robot functions mostly on its own, but a human is what engages it (co-operative control).
        i. Example: Targeting computer on a gun.
    d. Robot's intended customer is a government military, industry worker, etc.
    i. Example: Driver's license for a car, or gun permit
7. **Is the Robot/system/operator able to differentiate/ identify its intended target within an acceptable accuracy?**
    a. **What if the system causes the death of an unintended target, what might that target be and what are the repercussions?**
8. **If applicable what are the consequences of killing the intended target(s)? Are these ultimately good or bad?**

i. Example: Killing adults of a species may be detrimental to survival of infants of species, therefore damaging the population further than intended.
ii. Example: Ecosystem damage, eliminating another species food source.
iii. Example: Displaced workers/ damage to the economy.
iv. Example: More food for people in an area.
v. Example: Less disease and improved population health.
vi. Example: Mental trauma to survivors in area of use.

9. **Can the robot be repurposed for a malicious use via minimal adjustments?**
    i. Example: Haughwout Drones (Easily repurposed)
    ii. Example: Assembly robot for a certain product (Harder to repurpose)
        a. **If yes and the new use is morally unjust to an extreme or far from the intended use, re-evaluate the design for the new use, and attempt to redesign to prevent such malicious use.**

Questionnaire Instructions

This next section is to briefly go over why the questions in the questionnaire exist and how the sections are intended to be answered.

1st Question Set

The first set of questions asks about the intended purpose for designing a robot that potentially could be integrated with a means of distributing harm. These questions simply serve to make the answerer think about why they are designing a lethal robot and whether the reason is moral. This section also asks whether it is moral to kill an intended target whatever it may be as it could be immoral to do so in a certain situation.

2nd Question Set

This question set serves to ensure that engineers think about the task at hand and whether it is necessary to kill in order to achieve a desired outcome. It also asks the designer to think about

what would happen if they did not kill an intended target immediately and acknowledge what may occur if they do not kill after their involvement.

3rd Question Set

Third question is to check to see if the robot's lethality was intentionally designed into the system or just a byproduct of the design. One might agree that there is a difference between intentionally and unintentionally designing something that can kill a person hence why the question is in the questionnaire.

4th Question Set

The fourth question is to check to see if lethal force is even the main method for achieving a desired outcome. The example given shows how a robot may have lethal force capabilities for protection of others or itself, like a security robot, but that it is not necessary for achieving its main task.

5th Question Set

The fifth question is to ensure that the method of killing is considered humane since an inhumane method would be considered cruel and immoral.

6th Question Set

The sixth set of questions aims at the potential end user of the robot whether this be a person, government, customer, or the robot itself. This question serves to make the answerer think about who will be using their product since morals vary by person and organization. The question set also serves to ask about any potential legalities involved with possession of the robot.

7th Question Set

This set of questions if crucial for autonomous systems as there is no human identifying and determining potential targets for the robot. These questions aim at exposing the identification capabilities of the system and whether they are reliable enough to be making a life or death decision. That is not all however, the second question in the set serves to make the answerer

identify any high risk/ unwanted targets that may be mistaken by the system for intended targets and list the consequences of killing that target.

8th Question Set

This question serves to list out all the pros and cons of killing an intended target and determine if this is ultimately good or bad outcome. Essentially, this question is designed to ensure that the engineers think about the impact their creation will have on surrounding ecosystems, economies, populations, etc.

9th Question Set

This is the redo question set since depending on the answer one may have to redesign their robot. The gist of this question is to make engineers think about how someone can misuse their creation and whether that misuse is enough to constitute a redesign of the robot. For example, a lawn mower that can operate without the user's hands on the bar may be incredibly dangerous, especially if people decide to use the lawn mower to trim hedges by lifting it above their heads. That may be dangerous and far enough from the intended use to constitute adding features to prevent this kind of usage. Such as a handlebar safety grip and wheel sensor that require hands on the push bar and a certain number of wheels on the ground for operation of the lawn mower.

WPI MQP Lionfish Robot

(I will only list the answers to each question here to save space. For the question prompts refer to The Questionnaire section)

1. The intended purpose of the robot is to autonomously kill and harvest lionfish by going to depths inaccessible to divers.
    a. Yes, this method is essentially robotic aided fishing.
    b. The lionfish are an invasive fish species with no natural predators and are doing a massive amount of damage to the reef systems. Morality would be determined by the individual.
2. Given the high reproductive rate of the fish it is necessary to kill them in order to decrease the population and the damage they are dealing.
3. Lethal force is intentionally designed into the robot

4. Yes, lethal force is the primary method of achieving the desired task.
5. The method is the same used by commercial fisherman today.
6. In this case the robot is autonomous and acts on its own. However, the robot is not self-aware or sentient therefore, it only operates and functions where a human deploys it. That being said the intent is for this to be a consumer product used in the lionfish industry. In regard to licensing it would vary by region, but more than likely a fishing license would be required if any.
7. This is to still be determined when the robot is further developed. Given that the robot is autonomous this system needs to function perfectly for proper operation.
    a. If the neural network that operates the robot does not function properly then the robot can potentially kill the wrong species of fish. This could further damage the current reef problems.
8. The benefits of killing the lionfish are that there is more food, economic growth as this is a developing industry, and the native species of fish in the areas will increase. There is very little consequence to killing the intended target given that they have no natural predators in the area.
9. Technically yes, however some serious adjustments will have to be made to the code and mechanical hunting spear in order to do serious harm to a human.
    a. If the system is designed to prevent someone from accessing and changing the code, then it should operate as intended.

Appendix G

LCU Decision Matrix

EvaluationCriteria \ Design Option	Importance/ Weight	Home Depot Bucket	Catch Bag (Bucket Frame)	Trash Can	Mesh Covered Trash Can
General Design					
Internal Volume (gal)	2	5	10	7	7
Weight (lbs)	2	1.8	2	1	0.5
Positively Bouyant (Yes/ No)		Yes	No	Yes	Yes
Cost (Base Main Materials) ($)		$5.24	$9.91	$9.46	$14.78
Cost (Estimated Additional Hardware) ($)		$5.00	$15.00	$6.00	$6.00
Cost (Total Estimate) ($)	1	$10.24	$24.91	$15.46	$20.78
Bucket Durability & Sususceptible to Damage (1:Weak 7: Strong)	3	7	3	7	5
Lid Durability (1:Weak 7: Strong)	3	4	4	6	6
Lid Performance (1: Bad 7: Good)	3	4	4	6	6
Stability/Connection to Robot (1: Loose 7:Stab	2	5	4	7	7
User Interaction					
Ease of Attachment (1:Difficult 7:Easy)	2	7	3	4	4
Ease of Dettachment (1:Difficult 7:Easy)	2	7	6	6	6
Likelyhood of Unitentional Detachment (1: Likely 7: Unlikeyly)	5	3	7	7	7
Saftey For Handler (1:Danger 7:Safe)	6	5	1	4	2
Fish Removal (1: Difficult 7:Easy)	1	6	5	6	6
Water Drainage Rate (1: Slow 7:Fast)	2	4	7	5	7
Affected Robot Performance (Empty LCU)					
Forward Drive (1: Uncontrolable 7: Drivable)	3	2	7	4	7
Reverse Drive (1: Uncontrolable 7: Drivable)	3	3	7	5	7
Strafe Driving (1: Uncontrolable 7: Drivable)	3	1	6	6	7
Rotation Yaw (1: Uncontrolable 7: Drivable)	3	5	6	4	6
Vertical Motion (1: Uncontrolable 7: Drivable)	3	4	5	3	6
Affected Robot Performance (Full LCU)					
Forward Drive (1: Uncontrolable 7: Drivable)	1				
Reverse Drive (1: Uncontrolable 7: Drivable)	1				
Strafe Driving (1: Uncontrolable 7: Drivable)	1				
Rotation Yaw (1: Uncontrolable 7: Drivable)	1				
Vertical Motion (1: Uncontrolable 7: Drivable)	1				
Total Score		200	229	254	275

Bibliography

2018 Lionfish Summit Report. (2018). In *Florida Fist & Wildlife Conservation Commission*. Retrieved from https://myfwc.com/fishing/saltwater/recreational/lionfish/

Akyildiz, I. F., Pompili, D., & Melodia, T. (2005). Underwater acoustic sensor networks: research challenges. *Ad Hoc Networks*, *3*(3), 257–279. https://doi.org/10.1016/J.ADHOC.2005.01.004

Aluminum Handrail Direct. (2018). Effects of Salt Water on Aluminum. Retrieved August 10, 2019, from https://www.aluminumhandraildirect.com/effects-of-salt-water-on-aluminum/

Antaya, A. A., Peterson, E. H., Conroy, K. F., & Ralph, T. V. (2019). *Lionfish Bot 2.0*. Worcester.

Benson, B., Faunce, B., Kastner, R., Li, Y., Domond, K., Kimball, D., & Schurgers, C. (2010). Design of a Low-Cost Underwater Acoustic Modem. *IEEE Embedded Systems*. Retrieved from https://www.researchgate.net/publication/220413512_Design_of_a_Low-Cost_Underwater_Acoustic_Modem

BlueROV2. (n.d.). Retrieved from Blue Robotics website: https://bluerobotics.com/store/rov/bluerov2/

Braun, A. (2018). *RangerBot: Programmed to Kill*. Retrieved from https://www.hakaimagazine.com/article-long/sea-cucumbers-vanishing-act

Coral. (n.d.). USB Accelerator datasheet | Coral. Retrieved October 9, 2019, from https://coral.withgoogle.com/docs/accelerator/datasheet/

Gittings, S., Fogg, A., Frank, S., Hart, J., Clark, A., Clark, B., … Fortner, L. (n.d.). *Going Deep for Lionfish: Designs for two new traps for capturing lionfish in deep water*. Retrieved from http://www.sanctuaries.noaa.gov

Gupta, A. (2009). Invasion of the Lionfish. Retrieved from Smithsonian Magazine website: https://www.smithsonianmag.com/science-nature/invasion-of-the-lionfish-131647135/

Guth, F., Silveira, L., Botelho, S., Drews, P., & Ballester, P. (2014). Underwater SLAM: Challenges, state of the art, algorithms and a new biologically-inspired approach. *5th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, 981–986. https://doi.org/10.1109/BIOROB.2014.6913908

Image classification | TensorFlow Lite. (n.d.). Retrieved May 15, 2020, from https://www.tensorflow.org/lite/models/image_classification/overview

Impacts of Invasive Lionfish. (2020). Retrieved October 9, 2019, from National Oceanic and Atmospheric Administration website: https://www.fisheries.noaa.gov/feature-story/impacts-invasive-lionfish

Kang, J.-G., Oh, S.-Y., & An, S.-Y. (2010). Augmented EKF based SLAM method for improving the accuracy of the feature map. *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Retrieved from https://www.researchgate.net/publication/221065880_Augmented_EKF_based_SLAM_method_for_improving_the_accuracy_of_the_feature_map

Krishni. (2018). K-Fold Cross Validation. Retrieved October 10, 2019, from Towards Data Science website: https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833

Kulz, P. (n.d.). TensorFlow Detection Model Zoo. Retrieved from GitHub website: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

Lionfish Biology: Lionfish Discovery Story. (2020). Retrieved May 16, 2020, from https://oceanservice.noaa.gov/education/stories/lionfish/lion04_biology.html
Lionfish Derbies. (n.d.). Retrieved October 9, 2019, from Reef Environmental Education Foundation website: https://www.reef.org/lionfish-derbies
Lowe, A. (2016). Why Are Lionfish a Problem? Retrieved October 9, 2019, from https://lionfish.co/why-are-lionfish-a-problem/
Mumby, P. J., Harborne, A. R., & Brumbaugh, D. R. (2011). Grouper as a natural biocontrol of invasive Lionfish. *PLoS ONE*, *6*(6). https://doi.org/10.1371/journal.pone.0021510
NVIDIA. (n.d.). NVIDIA Jetson Nano Developer Kit | NVIDIA Developer. Retrieved October 9, 2019, from https://developer.nvidia.com/embedded/jetson-nano-developer-kit
Raspberry Pi 4 Model B Specifications. (n.d.). Retrieved October 9, 2019, from RaspberryPi website: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/
Rogers, W. F. (n.d.). *APOLLO LUNAR MODULE LANDING GEAR*.
Sharma, S. (2017). Epoch vs Batch Size vs Iterations. Retrieved October 10, 2019, from Towards Data Science website: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9
Shen, K. (2018). Effect of Batch Size on Training Dynamics. Retrieved October 10, 2019, from Mini Distill website: https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e
Smith, A. (2016). Invasive Lionfish Now on Whole Foods Menu. *CNN*. Retrieved from https://money.cnn.com/2016/05/27/news/companies/whole-foods-lionfish-invasive-species/?section=money_topstories
Spencer, E. (2014). Lionfish Crash Course | Reef Environmental Education Foundation. Retrieved October 9, 2019, from https://www.reef.org/news/blogs/lionfish-blog/lionfish-crash-course
Stojanovic, M. (2003). Acoustic (Underwater) Communications. In *Wiley Encyclopedia of Telecommunications*. https://doi.org/10.1002/0471219282.eot110
Waterson, J. (n.d.). Positioning System for Deep Ocean Navigation (POSYDON). Retrieved October 9, 2019, from https://www.darpa.mil/program/positioning-system-for-deep-ocean-navigation
What is a lionfish? (n.d.). Retrieved October 9, 2019, from National Oceanic and Atmospheric Administration website: https://oceanservice.noaa.gov/facts/lionfish-facts.html
Whitfield, P., Gardner, T., Vives, S., Gilligan, M., Courtenay Jr., W., Ray, G. C., & Hare, J. (2002). Biological invasion of the Indo-Pacific lionfish Pterois volitans along the Atlantic coast ofNorth America. *Marine Ecology Progress Series*. Retrieved from https://www.researchgate.net/publication/242670509_Biological_invasion_of_the_Indo-Pacific_lionfish_Pterois_volitans_along_the_Atlantic_coast_of_North_America
Yuzvik, A., Kelly, B. R., Lombardi, J. P., Uvarov, N. A., & Godsey, W. G. (2018). *Autonomous Lionfish Harvester*. Retrieved from https://digitalcommons.wpi.edu/mqp-all/2499
ZooKeeper LCU14. (n.d.).
Zulkifli, H. (2018). Understanding Learning Rates and How It Improves Performance in Deep Learning. Retrieved October 10, 2019, from Towards Data Science website: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10