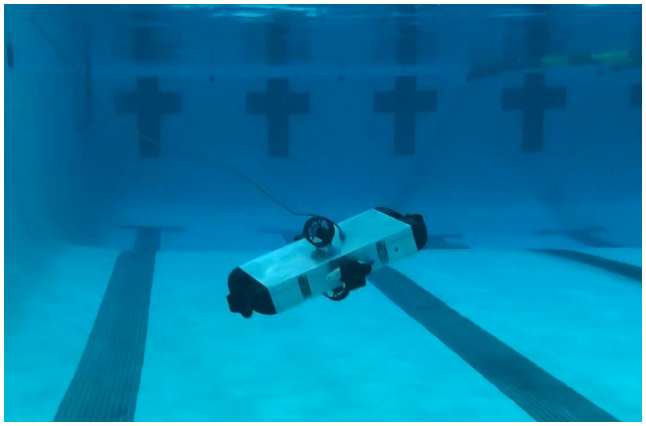

# RoboSub Team Killick 2017-2018

End of Semester Report
Spring Semester 2018


By
Nick Baron
Marta Camacho
Ross Dobitz
Jake Hannafious
Ryan Harty
Moeez Malik
Daniel Mansfield
Billy Phillips
Oren Pierce
Jeremy Naeve
Colin McDaniel

O2P/VIP Members:
Al Alothman
Angel Sanchez
Katie Wood
Elijah Atchley

Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, Colorado 80523


Project advisor(s): _Dr. Anthony Maciejewski, Megan Emmons, Chris Robbiano
Approved by: _____Megan Emmons_____

## Abstract

The RoboSub project goal is to design and build a fully autonomous underwater vehicle (AUV) to compete in the United States Navy RoboSub competition in San Diego, California. Historically, the top-placing teams have been from universities and combine students from multiple engineering disciplines. Although the project is expressly for a competition, it offers students the opportunity to develop an autonomous vehicle, an area many leaders in industry are aggressively pursuing.

The project, and team, has been broken down into three and a half subteams or categories with the half team being controls which included members from two of the other subteams. The three full teams are mechanical, power and propulsion, and vision and sensors. All are key areas of focus with specific tasks to develop a vehicle, with controls integrating sensory input and deciding the appropriate motor output for the developed mechanical system. The project as a whole implements a wide variety of technology: from image processing and recognition, data acquisition from inertial measurement units, thermistor arrays, pressure transducers, and current measurement - all of which are required to correctly control the motors and ensure the safety of the AUV's systems. The RoboSub project has an additional challenge in that it is a Vertically Integrated Project (VIP) so it includes underclassmen as well as seniors with the goal of continually evolving the project involving underclassmen in earlier stages to build institutional knowledge.

The project is advancing out of the beginning stages and one of the principal findings so far is that designs and methods must be constantly evaluated. Due to the competition based nature of the project as well as the evolutionary nature of a VIP team, the team has learned that documentation explaining decisions and progress will act as a foundation for future teams to build off of and develop state of the art AUV's that are competitive. The team has also come to appreciate the value of rapid prototyping and frequent testing of revisions, as the complex nature of the challenge makes foreseeing challenges in design difficult. Each hour of in-water testing time produced enormous amounts of useful data to continue the development of the AUV, each testing failure helping to push the design further.

This year's team focused on rapid prototyping and testing to proof the major design and philosophy changes. The propulsion system was completed Fall '17 and is able to take instructions from both a gaming controller as well as a Raspberry Pi. During the spring semester the focus was on creating an environmental monitoring system to ensure all systems were operating within specifications. A new chassis design philosophy was prototyped and tested underwater with the propulsion system in the fall semester. While the general concept was found to be promising, there were issues with the construction of the chassis and the seal was inadequately dynamic and could not compensate for changing pressures. Spring semester a second revision of the chassis was completed using CNC machine to reduce error and a dynamic and adaptive seal system was designed to compensate for changing pressures. Computer vision has made great headway in shape and color detection that will prove essential in achieving autonomy. Collecting data from the Inertial Measurement Unit (IMU) and the pressure transducers has been completed to better control the thrusters and supplement data gathered from computer vision. As a whole, the team is continually progressing towards the ultimate goal of building a robust platform for future teams to use to compete in the RoboSub competition.

# Table of Contents


Title i
Abstract ii
Table of Contents iii
1. List of Tables and Figures 5
2. Introduction 6
3. Review of Previous Work 8
    a. Mechanical Subteam 8
    b. Power and Propulsion Subteam 8
    c. Vision and Sensors Subteam 9
    d. Controls Subteam 11
4. Mechanical Subteam 12
    a. Design Objectives and Constraints 12
        i. Housing 12
        ii. Torpedo Launcher 12
        iii. Ballast System 13
        iv. Manipulator Arm 13
    b. Subsystem Breakdown and Processes 14
        i. Chassis and Electronic Housing 14
        ii. Chassis Cable Glands 17
        iii. Chassis Threaded Inserts 17
        iv. Electronics Mounting Boards 17
        v. Chassis Lid Seal 17
        vi. Lid Seal Holders 17
        vii. Latches 18
        viii. Hydrodynamic Fairings 18
        ix. Ballast and Safety System 20
    c. Testing and Design Verification 22
        i. Watertightness 22
5. Propulsion Subteam 23
    a. Objectives and Constraints 23
        i. Design Constraints 23
        ii. Design Objectives 23
    b. Design Summary 24
        i. Motivation 24
        ii. Power Distribution 25
        iii. Motor Actuation 25
        iv. Environmental Measurement System 25
    c. Testing 25
        i. Procedures 25

ii. Results 26
1. Approach 26
2. Current Data 28
3. Temperature Data 29
6. Vision and Sensors Subteam 30
a. Vision Hardware 30
i. CUDA 31
ii. Cameras and Interconnects 31
b. Computer Vision 32
c. Inertial Measurement Unit 34
d. Pressure Transducer 35
e. Testing and Verification 37
7. Controls Subteam 38
a. Objectives and Constraints 38
b. Decisions and Accomplishments 39
c. Results 42
8. Conclusions and Future Work 43
a. Mechanical Subteam 43
i. Final Design Summary 43
ii. Problems and Proposed Solutions 43
iii. Future Work 44
b. Power and Propulsion Subteam 44
c. Vision and Sensors Subteam 45
d. Controls Subteam 45

References ?
Bibliography ?
Appendix A – Abbreviations A-1
Appendix B – Budget B-II
Appendix C – Project Plan Evolution (required) C-1

# List of Tables and Figures


| Figure Number | Figure Title | Page |
|---|---|---|
| 3.1 | RoboSub Version 3 during pool testing | 12 |
| 3.2 | RoboSub Version 3 SolidWorks model exploded view | 15 |
| 3.3 | Aluminum chassis components including: tongue and groove, pressure transducer housing, threaded screw backing, latch inlay | 16 |
| 3.4 | Individual fairing cast and fairing cast to motor mount | 18 |
| 3.5 | Fairings being cast to current motor mount version | 20 |
| 3.6 | Pictures of weld porosity and leak sources during pressure testing | 22 |
| 4.1 | Propulsion Power Distribution Diagram | 24 |
| 4.2 | Hall Effect Test Circuit Set Up | 26 |
| 4.3 | Matlab Program Flow Diagram | 26 |
| 4.4 | Current Data During Pool Testing | 28 |
| 4.5 | Temperature Gradient Inside of Chassis | 29 |
| 5.1 | Jetson TX2 Board | 30 |
| 5.2 | Jetson and Joule specs and prices | 31 |
| 5.3 | Circle mask | 32 |
| 5.4 | Environment | 32 |
| 5.5 | Preprocessed Environment | 33 |
| 5.6 | Boxed Object | 33 |
| 5.7 | Flow diagram IMU process | 35 |
| 5.8 | Voltage pressure calibration curve | 36 |
| 5.9 | False Statement | 38 |
| 5.10 | True and False Statement | 38 |
| Table 6.1 | Commands Mapping | 40 |
| 6.1 | PID Feedback Diagram | 40 |
| 6.2 | Control Scheme | 42 |

# Chapter 1: Introduction

The RoboSub project is a competition put on by the U.S. Navy and RoboNation and is held annually at the TRANSDEC facility in San Diego California. The goal is to design a fully autonomous underwater vehicle (AUV) to traverse an obstacle course and complete a wide variety of tasks, all in 15 minutes or less. Additionally the vehicle must be under 125 lbs and battery powered with the open circuit voltage of each battery less than 60V DC. The vehicle can only compete once, then the following years team must design an entirely new vehicle.

The obstacle course changes from year to year with different themes. There have been consistencies from year to year which include the following: line tracking, passing through gates, buoy identification, hitting targets with a torpedo, and manipulation of objects with a robotic arm. Additionally, the weight restriction and time to complete the course have continually awarded more points for lower values on both. This has been the primary focus of both last year's team as well as this year's.

Currently, the RoboSub team is comprised of 11 seniors. The breakdown is as follows: 7 electrical engineering majors, 3 mechanical engineering majors, and one computer engineering major. Additionally, the project has 4 underclassmen who are part of the vertically integrated project (VIP) program, of which the Robosub project is. This allows the project to incorporate underclassmen to gain experience working on the team as well as join the project as seniors with a well developed understanding of project requirements. The role of these students this year is to validate and test the designs the seniors create to help the team create a more robust system. Having undergraduate students also helps the project continue to develop and progress. The hope is that this project will consistently be offered for years to come and actively compete and place well, if not win, the RoboSub competition.

This year's team is split into three and a half subteams which are power and propulsion, vision and sensors, mechanical, and controls. Controls in a new subteam introduced this year and is considered a half team because it is integrated into both power and propulsion and vision and sensors subteams. The primary role of the sensors subteam is to create the "brain" of the AUV by turning vision and sensors data into motor commands. Power and propulsion is responsible for taking instructions in the form of a serial string and turn those into pulse width modulated (PWM) signals which are sent to the electronic speed controllers (ESC's) to control motor speed. The vision and sensors subteam generates data from an inertial measurement unit (IMU), pressure transducers, and cameras to collect data to give to controls to determine where the AUV is located and what it needs to do. The mechanical subteam works on the chassis, torpedo launchers and mechanical arms as well as any extremities on the chassis. With one year of experience, and a few returning members of the original team, this year's team was able to work even more with other subteams to begin the critical testing phase.

The team set out with the goal to have the vehicle in the water that would track and follow an orange ball. This was a very ambitious task given that many facets of the design would be reevaluated and eventually changed. What was accomplished this semester was water testing of the second iteration of the chassis design to attain hydrodynamic, and thermal data. Additionally, the propulsion system was tested using manual input from a gaming controller to find out how the chassis and thrusters behave in water, and how much current the system draws. The team also successfully implemented an accelerometer to perform correctional movements to keep the vehicle balanced as it moved through the water.

This report serves to describe the work done during the Spring 2018 semester and outlines what is needed to continue this project in future years. Design choices as well as part selection will also be described with the purpose of aiding future teams by explaining the team's decision making process as well as sharing mistakes and what was learned from those mistakes.

The second chapter of this paper will provide a review of the work done by the 2016-17 team and work completed during the Fall 2017 semester. Chapter 3 will describe, in detail, the work done by the mechanical subteam during the Fall 2017 semester as well as address design changes and decisions that were made after testing in the Fall 2017 semester. The power and propulsion subteam, vision and sensors, and controls subteams will follow the same format in chapters 4, 5, and 6. Chapter 7 will describe future work that is remaining for future teams.

# Chapter 2 - Review of Previous Work

## Mechanical Sub Team

To begin the Fall 2017 semester the Mechanical Subteam conducted a range of tests on the previous year's AUV design. These tests set out to attain thermal, and hydrodynamic data as well as analyze the usable internal volume. The results of theses tests showed that the internal temperature of the submarine rose to levels that caused component failure even when there was active cooling from circulating tides. The hydrodynamic tests showed that it took more force to push the submarine through the water at the desired speed of 1 m/s than the motors could exert, and further simulation showed that vortex generation in the frame of the submarine was severely inhibiting it's motion. When the internal volume was analyzed, the frames that supported the sliding drawers took away from the overall usable space, and the heavy cap took several people and several attempts to install properly. The Mechanical Subteam concluded that major revisions for the submarine's body needed to take place.

Throughout the remainder of the Fall 2017 semester the Mechanical Subteam prototyped a torpedo launcher and tested its accuracy with different barrel lengths, determining that for this application, the barrel length was statistically insignificant. The body of the submarine was prototyped with two revisions of the lid and seal. Testing on the body prototype showed that the shape and size were viable, however the design needed revision to prevent small leaks and fabrication errors.

## Propulsion Sub Team

During the first year of Robosub the Propulsion subteam set primary objectives to gain an understanding of competition regulations for the electrical systems, and use that to develop a basis for power distribution and chassis mobility. Research and calculations were done in order to make decisions for battery, motors, electronic speed controllers and the battery management system (BMS) selection. Once parts were selected they were tested and tailored to withstand marine exposure as well as move a large chassis. The motors selected were 925 Watt drone motors that were waterproofed by coating the coils in machinable wax. Load tests were done on the motors to characterize their power draw under load which came out to an acclimating 30 Amps. In light of this discovery, a BMS was being developed that would be able to withstand the maximum possible current draw of the 60 Amp motors. Moving forward, the future of propulsion would be spent trying to move a large chassis design without overheating while cementing a reliable BMS design.

For the Fall of 2017 the Propulson subteam began working alongside the Mechanical subteam and the Controls subteam to tackle challenges such as chassis mobility, motor control and actuation respectively. The propulsion systems is generalized by two subsystems, data logging and motor control. The microcontroller selected for data logging is an Arduino Uno, which was programmed to monitor current and temperature. An Arduino Mega was selected to send PWM signals to the ESC's that will drive the motors  The combination of the two controllers spearheaded development for environment monitoring and motor driving,

Since the chassis had reduced in size, low power ESCs and motors were selected and as a result there was no longer a need for the BMS. The motors are 350 Watt T200 Blue Series Thruster motors and

are driven by 30A ESCs. Blue Series motors were preferred because they are off-the-shelf hydromotors designed specifically for AUV projects. Their specific application eliminated the need to waterproof motors and develop hydrodynamic shrouds and propellers. By this decision, the power consumption was reduced significantly and the BMS was completely phased out. However, battery monitoring and overheating of components while driving the AUV was still a concern that needed to be addressed.

Methods for current sensing, temperature sensing, and voltage sensing were explored and developed. Hall effect sensors were selected for sensing how much current is being drawn by the motors from the battery. To monitor chassis temperature, a thermistor string was manufactured and placed in each corner of the chassis to monitor heat being generated by the ESCs. Quantizing current and temperature values was done by designing a circuit using these sensors that would produce a voltage. The voltage was read into an analog port of the Arduino Uno and processed to produce a corresponding current or temperature value. In tandem, motor actuation code was developed and programmed onto the Arduino Mega. The Mega would respond to a motor state (arm/disarm) and/or supplied power to the motors so long as the temperatures or currents detected were below a certain threshold. This method of motor control is what began bridging the communication gap from the Sensors/Vision subsystem.

The Controls and Propulsion subteam began developing serial communication between a Raspberry Pi and the Arduino Mega. This method of communication will be the transfer of commands interpreting decisions that need to be made based on Vision and Sensors data. In addition the aforementioned motor state is determined by the Raspberry Pi which is constantly checking for error flags that could potentially come from the Uno during data logging. Ultimately, the goal was to be able to send motor states and PWM signals from the Raspberry Pi to the Mega and achieve motor control while monitoring current and temperature.

The Fall 2017 semester ended with a pool test where the motors were driven by a PlayStation controller, and current data was recorded with a Data-Logging Shield that was later analyzed to look at the accuracy and feasibility of current measurement. While pool testing was an accomplishment, data logging was not very successful. Communication between the Raspberry Pi to the Arduino was not implemented due to underdevelopment. The overall propulsion system setup was not efficient and posed as a difficult task when prepping for pool testing. Moving forward, work still needed to be done to refine the propulsion system layout, improve sensing and data logging to be more reliable and continue troubleshooting serial communications.

## Sensors Sub Team

In order to keep up with the rigorous goals of the sensors team, the integration of hardware needed to happen as fast as possible. The prioritized characteristics for all of the hardware are simple, fast and reliable. The previous year's team had purchased an Intel Joule 570X for vision processing, but for unknown reasons the board did not function as expected. The first step fall of 2017 semester was to research a new vision processing board. Several factors were taken into account, one such factor was the type of code that would run on the board. In the case of video processing it was decided that the code that would compute the operations could easily be parallelized for efficiency of execution. With this in mind, the team agreed upon obtaining an Nvidia Jetson TX2.

In order for an autonomous vehicle to perform tasks without human interaction, different types of sensors are needed to provide the vehicle with spatial awareness, information about its current state, and

the ability to observe its surroundings and make decisions. The submarine's vision was implemented using OPENCV software and Python on the Nvidia Jetson board using the operating system Ubuntu. OPENCV is an open source computer vision package that can be used with any language, most commonly with Python or C++. It is widely used in the industry in machine learning applications. Much of the first year of the project was spent setting up code repositories on Github and getting all of the necessary software installed onto individual systems. It was important to develop the vision code on the same platform that it would be running on so the software could be seamlessly transferred from development to running code on the Jetson board. Once the repositories and development platforms were set up, bits of code were written using OPENCV functions in order to study their functionality and potential applications. The goal was to identify functions that would best pertain to the missions of the submarine, to include object and line detection, classification, and tracking.

The beginning of the Fall 2017 semester was spent updating the repository and vetting the code that did not prove to be useful. At this point, the tasks that the submarine would have to accomplish for the competition were released so new ideas of how to accomplish these tasks were developed. Since the code development was still in its infancy, it was necessary to scale back from the overall competition and pick one thing to really focus on. Object tracking was chosen to be the primary focus because it implemented many of the concepts that needed further research such as detection, classification and tracking. Due to the complex nature of computer vision and machine learning concepts, comparisons were done between how humans see things and how a computer might see things. Along with this, the code needed to be written in a way that would be optimal for an underwater situation. For example, since the submarine would not be moving very fast, so a high camera frame rate would not be necessary, allowing more computational power to be used on each frame. This was essential for object detection, classification, and tracking to be done on multiple objects in each frame. Smaller functions were written that could be used in overall functionality like average color of a frame, object detection based on color, and object detection using shape cross correlation. Shape cross correlation uses a mask of a shape that needed to be found and compares it against a preprocessed environment.

IMUs are greatly affected by drift, which is the result of manufacturing, and how modern IMUs detect acceleration. There are better accelerometers and gyroscopes on the market but, are very expensive and often use completely different methods of measuring the acceleration and rotation. Most of these instruments would be infeasible due to size and cost greater than $100,000. This is why the previous team sought out a relatively small IMU with development board (Sparton AHRS-8). The IMU and development board were graciously donated to the team by Sparton electronics, spring 2017. The IMU and development board came several different forms of communication including serial over USB which most modern computers have. Making this form of serial communication ideal. In addition to all of the development board's communication pathways, it also came with several different commands and built in filters. All of these filters, however, incorporate the magnetometer data which will be heavily corrupted by the high speed motors surrounding the AUV and therefore cannot be used on the final design. This is why the previous team and one team member this year worked several libraries in MATLAB and in python that only operate on three dimensions of acceleration and three dimensions of rotation. Removing any bias caused by the motors.

The IMU that was used for this project as already mentioned, outputs three dimensions of acceleration and gyroscope. The accelerometer and gyroscope data was combined to describe three space orientation using Mayhony's AHRS algorithm. There are two standard algorithms that can do this, the other being Madgwick AHRS algorithm. Mayhony's algorithm was chosen for this project as it is less computationally intensive and will be easier to implement onto a raspberry pi. The basic operation of Mayhony's algorithm is the following:

- the input acceleration data provides a “best guess” of the effect of gravity and the gyroscope data is used to measure any changes in rotation
- the output of the algorithm is a quaternion, which is one of the best methods of describing a rotation
- as it encodes direction, even in the case of a 180 degree rotation, it is converted into a rotation matrix.

The rotation matrix obtained from Mayhony's algorithm contained all of the information that is necessary to keep the AUV upright. The source code for the algorithm was initially in MATLAB and was then converted over into python after extensive testing during the spring semester 2018.

## Controls Sub Team

Given the immaturity of the project, the controls subteam had a large task set ahead of them. This project having only started last year is still in its infancy. That being said, last year there was not a controls subteam. This left the controls team with nothing to start with this academic year. Last semester two members of the team mainly focused on getting the Sensors and Propulsions subteams systems to a point where the team could get down to business. One member of controls worked heavily with the Sensors subteam to create a plan for a hierarchy of sensor inputs and develop reliable interfaces within the computer for control programs to access the Sensors Team’s systems. Meanwhile another team member worked with the Propulsion subteam to create a scheme to control the motors of the AUV while maintaining safety for the user and AUV itself.

# Chapter 3 - Mechanical Subteam

*Figure 3.1: RoboSub Version 3 during pool testing*

## Objectives and Constraints

### Housing

1. The housing shall be watertight to withstand pressure at a depth of 20 ft. from the surface for a minimum of 20 minutes.
2. The housing shall weigh no more than 50 lbs. when dry.
3. The housing shall dissipate heat in order to maintain an internal temperature of no more than 60 degrees C.
4. The housing shall withstand impacts with solid surfaces when at a speed of 1 m/s.
5. The housing shall fit all electronics such that they are visible and easily accessible when the housing open.
6. The housing shall be constructed to hold or connect to all submarine systems.

### Torpedo Launcher

1. The torpedo launcher shall launch the projectile a minimum of 36 in.
2. The torpedo launcher shall launch the projectile in a straight line within a 4 in. radius of the center axis of the launcher.

3. The torpedo launcher shall generate a recoil that pushes the submarine no more than 6 in. in any direction.
4. The torpedo launcher shall withstand an internal pressure of at least 60 psi.

### Ballast System

1. The ballast system shall create neutral buoyancy within a margin of 0.1%.
2. The ballast system shall cause the submarine to sink.
3. The ballast system shall be watertight.
4. The ballast system shall fill its volume with water 100%.
5. The ballast system shall drain its volume a minimum of 90%.
6. The ballast system shall act as the safety system, producing +0.5% buoyancy when disengaged.

### Manipulator Arm

1. The manipulator shall grasp an object without definite edges and a handle.
2. The manipulator shall grasp and manipulate golf balls.
3. The manipulator shall sense when it has grasp of an object.*
4. The manipulator shall store golf balls in a predictable way on its extremity. OR
    a. The manipulator shall place golf balls in a predictable way in a storage device.
5. The manipulator shall retrieve a specific color of golf ball without needing to manipulate another golf ball first.
6. The manipulator shall have the structural integrity to push and pull levers with a maximum applied force of 15 lbs.
7. The manipulator shall have joints that have a maximum increase of friction force due to load of 20%, summed over all joints.
8. The manipulator shall have the modularity to grab, store, retrieve, and place golf balls as well as pull and push levers and plates, without the aid of attitude adjustments to the submarine.
9. The manipulator shall have the modularity to push and pull tasks with minimal aid by the submarine position, maximum of 1 foot of submarine housing displacement.
10. The manipulator shall have the modularity to place golf balls in bins with their brims breaching the surface of the water with a maximum aid in attitude adjustment of the submarine housing of 90 degrees about any axis.
11. The manipulator shall be long enough to breach the water and place golf balls in bins that breach the water.
12. The manipulator shall collapse in a way that it does not extend more than a maximum of 2 inches away from the bottom of the component housing.
13. The manipulator shall collapse in a way that it stays within the area defined by the length of the bottom of the component housing and 8 inches centered on the width of the bottom of the component housing.
14. The manipulator shall collapse in a way that does not impede the function of the bottom housing propeller.

## Subsystem Breakdown and Processes

### Chassis and Electronics Housing

Initial planning for the chassis/housing of the submarine intended for it to serve both purposes of structural chassis and component container while still meeting requirements for hydrodynamic flow, weight, and thermal dissipation. Balancing the system to be low weight and thermally conductive means that there is a point of equilibrium between internal volume, buoyancy, weight, and thermal conductivity. Since the chassis was chosen to be made of aluminum, which has a lower density than other metals, the internal volume must be relatively smaller to keep buoyant forces from being uncontrollably large. Every additional cubic inch corresponds to another roughly .04 lbs of buoyant force. This meant that the components needed to fit into the housing efficiently, otherwise the size of the housing would be increased and then the submarine would need to be weighed down with dense material adding unnecessary mass. To meet the requirements defined after testing the first version of the chassis and housing, it was determined the housing should be made small and square to fit components easily, be streamlined after adding fairings, designed for ease of manufacturing, and be made of aluminum to be light and dissipate heat well. The aluminum body allows for easy attachment points as tabs could be welded on for peripheral attachments.

To increase the ease of manufacturing and decrease cost the second version of the chassis (Fall 2017) was designed to make use of aluminum C channel, L channel, plate, and aluminum MIG welding which was available to the team through Distinctive Welding, a local welding and fabrication shop. The chassis would consist of two sections, the component housing and lid. The component housing was made using C channel sides with a plate bottom, and the lid was designed to use L channel sides and a plate top.This used an overlapping lid where the edges of the L channel sat over the the C channel sides. This base design allowed for ease of manufacturing as the machine time was limited to cutting, drilling and welding fairly simple components.

The second version functioned by attaching necessary peripheral components flush to the sides of the chassis with screws and providing gland holes for wiring that needed to enter the housing. The necessary peripheral components (components needed for the chassis to close and move) included lid latches and motors. This required welded on aluminum screw-hole backing, which was drilled and tapped so that stainless steel threaded inserts could be inserted allowing for repetitive mounting and dismounting of these components. This completed the second version of the chassis as a rapid prototype and proof of concept. Upon testing and integrating systems, errors in version two design were:

- Slow leaks that were not apparent in initial water tightness testing.
    - Leaks originating in the screw backing welds
        - Due to the thin C channel walls and that redundant welding not being possible
        - There were also 28 holes that backing was welded into increasing the likelihood of error.
        - Density of welds increased thermal stresses and likelihood of cracks forming
    - Leaks originating in corners
        - Thin metal susceptible to fatigue
        - Corner angles tended to be inconsistent and hard to weld cleanly

- Components fit well but needed a thought out mounting method that is both structural sound so the chassis can be inverted and also easy to mount to in a efficient way.
- The chassis is still relatively heavy, but submerging still requires added weight.

These errors in the design required a second iteration of the chassis which focused on using CNC machining to better meet previously defined requirements and decrease the possibility of error in the manufacturing process to ultimately have a reliable and reusable chassis that could see competition. The third version of the chassis (Spring 2018) used the same concepts as the version two design except that all components would be CNC cut to take away excess material, leave backing where needed, streamline the design, and reduce the probability of errors during welding/manufacturing. The sides of the chassis use a tongue and groove fit to maintain square, have a chamfer and thick material for robust welding, and have component specific cuts for motor mounts and pressure transducers. The bottom plate of the submarine has internal and external screw backing. Externally the backing is used for motor mounting, and internally the backing is used for mounting component boards which are modular for any components, electronic or not, to be mounted to it.

*Figure 3.2: RoboSub Version 3 SolidWorks model exploded view*

The third version of the chassis was iterated many times in its Solidworks model to accommodate as many components as possible before being manufactured. The chassis was simulated in Solidworks for its response to the pressure experienced at 16 feet of depth in water. This simulation showed that the only areas of concerning stress were on the top and bottom plates of the chassis which would experience the most deformation of a few tenths of an inch each. This deformation was considered negligible and the safety factor for the top and bottom plates relative to their yield stress was over 2, which verified their structural integrity. Simulation and modelling allowed for verification of the structure and of the

placement of necessary components including the ballast system, electronic component mounting boards, microcontrollers, and sensors. It also allowed for some level of verification of the size chosen for the chassis. Once in the manufacturing process the chassis sides were CNC'd out of *1x3* inch aluminum solid stock. The left and right sides of the chassis are identical including inlays for the latches, backing for the motor screw inserts, access for pressure transducer to the water, cutaway of excess material, and finally an inlay for the lid so that the lid sides are flush with the chassis sides. The front and back sides of the chassis were also identical, both having backing for motor mount inserts, cutaway of excess material, and inlay for the lid. The front and back sides did not include any special conditions for components. The bottom plate of the chassis was CNC'd out of a *12x20x½* inch solid plate to be ⅛ inch tall except where screw backing was needed where ½ inch tall ½ inch diameter cylinders were left from the original block. The lid of the chassis was made from *12x20x1* inch solid aluminum plate where material was removed to leave ⅛ inch thick shell walls on all sides, inlay for the latch and latch receiving ends, and motor mount insert backing.

*Figure 3.3: Aluminum chassis components including: tongue and groove, pressure transducer housing, threaded screw backing, latch inlay.*

The chassis was intended to be manufactured by setting all sides into their respective tongue and grooves, placing them on the bottom plate, placing the lid on the sides in its inlay, squaring up the seams, then clamping and tacking the seams with TIG welds. This proved to be troublesome due to inaccuracy in the CNC process and allowances for tolerance. The tongues and lid inlays of the sides needed to be reduced in size by using a manual mill and with low grit grinder passes. Once the components fit squarely the welding process began using TIG performed by a professional welder instead of MIG by a group member to have higher probability of watertight welds. Testing and error repair of the chassis is covered in the testing section of this chapter.

### Chassis Cable Glands

The glands were used for wires that needed to penetrate the chassis. At a minimum for a gland to qualify for use on the chassis it must be IP68 and able to withstand an external pressure from water at a depth of 16 feet plus a safety margin 50 percent without the use of epoxy for potting so wires could be taken in and out or replaced. This depth translates to an external pressure of 10.5 psi. The glands also needed to be variable enough to fit slightly different diameter wire for motor, camera, or controller wire while being relatively small. The glands that were chosen fit the size and wire diameter requirements with a rating of IP68 up to 1 bar or 14.5 psi, which fit the pressure requirement.

### Chassis Threaded Inserts

Due to the use of aluminum for the body, anywhere that components are required to be screwed to the chassis also required the use of stainless steel threaded inserts. These inserts were screwed in to threaded aluminum backing designed into the parts of the chassis to not be removed since aluminum quickly galls and fractures. The inserts were also chosen as stainless steel so they do not corrode in the water.

### Electronics Mounting Boards

The electronic boards are used to mount all of the electronic components. They are designed to be cut out of polycarbonate using a waterjet, because polycarbonate is not conductive and is not too brittle to be cut using the waterjet. The current electronic boards were plasma cut from steel and painted to reduce conductivity. The current boards were cut after the wrong files were sent to the waterjet and the resulting boards were too big. The steel boards were made quickly without the lead time and coordination that the waterjet demanded. Currently the steel boards aid in reducing the buoyancy of the submarine and nylon spacers are used to ensure that the electronics are insulated from the boards in the event that the paint fails.

### Chassis Lid Seal

The lid seal is the largest seal on the submarine. It was cut from ½” soft neoprene closed cell foam. The designation of “soft” means that it takes 11 psi to compress the foam 25% of its original thickness. This hardness designation ensures that latches compress the seal at least 25%, but do not plastically deform the seal during the process. Plastic deformation would mean that the seal’s original thickness has permanently decreased and would need to be replaced to ensure a watertight seal. The maximum width around the body base was used when designing the seal so that the seal would cover the maximum possible surface area. This perimeter has varying widths due to the body’s recessed latches. To precisely account for these varying widths during manufacturing, the seal was cut using a waterjet.

### Lid Seal Holders

The lid seal holders were manufactured to keep the seal from moving out of place as the lid was brought down and latched shut. The holders extend ⅛ inch above the top of the chassis housing edges and are mounted to the electronic mounting boards around the perimeter of the housing so that the ⅛ inch edge of the holders is in contact with seal and prevents it from falling into the housing on any side. This

method keeps the seal tight against the holders and in its designated area on the bottom half of the chassis housing to ensure a good seal when the lid is latched shut. The holders were designed and added after an improperly installed seal caused the inside of the chassis to flood.

### Latches

The latches used to compress the seal and hold the lid of the submarine to the base each have a 400 lb load capacity. They are stainless steel to prevent corrosion and have internal springs to negate any effects from vibration.

### Hydrodynamic Fairings

*Figure 3.4: Individual fairing cast and fairing cast to motor mount.*

A top speed of >2 m/s was listed as a requirement for the chassis, and any increase in movement efficiency is desirable to increase battery life and maximize time constraint bonus points. CFD analysis showed a top speed of the chassis with motors attached as 2.5 m/s and a cruising speed at 50% thrust of 1.2 m/s. In-water testing verified the 1.2 m/s cruising speed. However CFD analysis also revealed the fairly obvious inefficiencies of a fully squared off design. Thus hydrodynamic fairings were deemed an important add-on to the chassis subsystem. The design was started by creating fairings with the "ideal" hydrodynamic profile with no regard for size or weight of fairings, just focusing on creating the most efficient profile. These fairings extended 23 cm each beyond the front and rear of the AUV and dorsal and distal fairings were designed to cover the top and bottom motors. This design resulted in a top speed of 3.2 m/s and a coefficient of drag of approximately 0.29. However this fairing design would weigh almost 40 lbs once cast from the polyurethane resin that a materials study deemed to be the best material for the job. Polyurethane resin was chose due to its density being similar to water, its ease of casting, durability, energy absorption, and non-reactivity.

40 lbs of fairings would outweigh the chassis even with all electronics on board, and would be exceedingly difficult and expensive to manufacture, therefore design refinements were required. The next version omitted the dorsal and distal fairings and focused on the minimal possible front and rear fairings.

Analysis of CFD flow diagrams revealed that the flat surface to the left and right of the front motor and the corners produced the most vorticity, and therefore rounding out those surfaces would make the largest impact on the flow of the AUV. Two independent mirrored front fairings weighing exactly one pound each were designed and mounted to the front end in the CAD model, and then CFD was run again. The top speed was determined to be approximately 2.8 m/s. The size was determined by polyurethane shipments coming in one pound tubs, which allowed each fairing to be cast from exactly one tub of resin. The next problem area was the low pressure/low velocity zone at the rear of the AUV caused by the squared off rear end. Analysis of the flow diagrams showed a large amount of vorticity around the tail of the AUV. The front fairings were subsequently mirrored onto the rear of the AUV and CFD was run again. The top speed was determined to be approximately 3 m/s, almost as fast as the "ideal" fairing design from earlier, and the fairings only added 4 lbs to the overall weight. The team suspected adding any more fairings onto the chassis would yield diminishing returns as the front and rear minimal fairing design was so similar to the "ideal" design. To ensure hydrodynamic benefits were not being left untapped similar one pound fairings were designed for the top and bottom motors, like in the "ideal" design. CFD results showed a less than 10% improvement in top speed, but mass analysis of the CAD showed there would be an additional 4lbs of mass added to the AUV. As the drag increases exponentially with speed anything lower than top speed would see less than a 10% improvement. Thus it was deemed that the loss in acceleration due to the extra mass would be more impactful than the 10% improvement to top speed and upper and lower fairings were abandoned.

Manufacturing the fairings proved to be an exercise in trial and error. Following the advice of a practicing engineer a test mold was 3D printed and then resin was poured into it. The resin found every single imperfection in the mold and was impossible to remove. So a mold with parting lines was designed and 3D printed. But when the resin was poured it managed to seep through the parting lines and stick the mold to the table. This revision of the mold was able to be removed and demonstrated that while the execution still needed refinement the concept of 3D printed molds for resin casting was viable. From here different methods of mold release were tested until Plasti-Dip was discovered to have a teflon-like reaction with polyurethane resin. The procedure for fairing casing was developed as such as a result:

1. Clamp mold parts together tightly
2. Spray with 7-9 layers of Plasti-Dip
3. Pour polyurethane
4. Pull mold apart after 3 hour demold time specified by material data sheet
5. Allow fairing to cure for remaining 21 hours of cure time

*Figure 3.5: Fairings being cast to current motor mount version.*

Originally fairings were cast one at a time with the intent to figure out mounting systems once the fairings were cast. After several off the shelf motor mounts cracked from fatigue it became necessary for the team to design and 3D print new motor mounts at which point the idea of designing motor mounts to integrate fairing mounts were developed. The eventual design of the motor mount had arms that reached out along the face of the AUV with branches to provide ample surface area to which the polyurethane could adhere. The mold was then designed to hold the motor mount in position such that both front fairings could be poured at the same time while simultaneously mounting to the motor mount. The dimensions of the motor mount and mold ensured that the outer edges of the fairings precisely lined up with the sides of the AUV, and the inner edges of the fairing just touched the sides of the front motor. These motor mount fairing combo assemblies proved to be simple to manufacture once tooling was designed and printed, and remarkably robust and clean once installed. Assembly was also fairly streamlined by the new motor mount design. Testing revealed that the new mounts were significantly stronger than the off-the-shelf components.

### Ballast and Safety System

The ballast system is an essential part of an underwater vehicle as it gives the vehicle a dedicated system for sinking and surfacing by controlling the AUV's buoyancy. A ballast system controls buoyancy by drawing water into a sealed area inside the body of the submarine to reduce the internal air volume and add weight. The competition also requires that there be a safety system that makes the AUV 0.5% buoyant upon electrical failure. This safety system is designed as part of the ballast system since the ballast is already responsible for the control of buoyancy. There are six types of ballast systems for typical

AUV use. Two of the systems were able to meet the size and capacity needs that were set by the AUV's size and weight. Ultimately a piston style ballast system was selected for its precision control and the ability to scale its size and capacity quickly.

To reach the needed capacity two identical piston systems straddle the centerlines of the submarine. The two pistons are actuated by a threaded rod the way vice jaws move. The threaded rod is pushed and pulled by gears with nuts embedded in them. The gear train is designed to be turned by a DC motor. A set pin is used to attach the threaded rod to the piston and a t-shaped o-ring is used to seal the piston to the interior of the cylinder. Water is drawn into the cylinder through the floor of the AUV and pushed out of the cylinder through the same opening in the way that a syringe works with the piston acting as a plunger. The whole system rests on small feet that are anchored to the floor of the AUV. To act as the emergency system a spring is placed behind the piston. In the case of electrical failure the motor releases the gears and the spring pushes the pistons forward to evacuate the water and bring the submarine to the surface. The ballast is designed to have a maximum capacity of 1.2 lbs with each individual cylinder drawing in a little more than a half pound of water. This accounts for more than 3% of the submarines total weight when ballast is filled. While the initial design for the ballast system is complete, a full prototype has not been completed or tested.

This semester the mechanical team began prototyping parts of the designed ballast system. The cylinder is 3 inch diameter polycarbonate with ⅛ inch wall thickness. The piston is acetal turned down from 3" diameter stock. Acetal is a plastic that can hold tight tolerances over time and temperature changes. This material was chosen to have the best chance against preventing leaks developing in the ballast system during repeated use. The threaded rod for the prototype is steel ACME rod. ACME rod has a special thread cut to move the piston very precisely. While this rod is ideal for actuating the piston back and forth, it will likely bind when the spring is added as part of the emergency system. ACME rod was selected for the prototype for its precision movement and low cost. The gears that turn the threaded rod were designed with 12 teeth to sit 1.73 inches apart at the center and prototyped using 3D printing. 1.73 inches ensured that the tubes nested as closely together as possible, but required that the gears be custom designed. The prototyping of the gears ensured that the teeth meshed properly.

## Testing and Design Verification

### Watertightness

Testing for leaks in version two and three of the chassis design took similar, but increasingly robust procedures. Initially, without easy access to deep water, version two of the chassis was tested by placing it latched and sealed into a tub of water for extended periods of time up to 12 hours. This would allow for any slow leaks to be apparent and hopefully mimic any issues that would be seen from increased pressure from depth due to the prolonged time. Initial water testing showed that the version two of the chassis was water tight in low pressure. This testing proved to be unsatisfactory as the chassis leaked in the conditions experienced in pool testing where the chassis experienced up to seven feet of depth which is an increase of pressure of roughly 3 psi. It was not apparent, or easily testable, at what depth the chassis failed, but it was apparent that the corners of the version two chassis, where the most cuts and welding had been done, was the source of the failure. These findings inspired the chassis iteration and also showed a need for a robust testing method.

The third version of the chassis was water tested with a similar procedure as version two, but with the addition of using a vacuum pump to evacuate air from the chassis, and recording what happened internally with a GoPro camera. The air was vacuumed out while it the chassis was submerged through the gland hole for vision/controller. This method allowed for the chassis to be tested to the requirements set for depth (20 feet for 20 minutes) since vacuuming out the air from the chassis creates a net external pressure on the chassis. The water testing was done periodically through the last manufacturing processes of the chassis to limit the number of areas where failures could occur. Initially only one gland hole for the vacuum tube was drilled and a gland, detailed in the chassis component section, was attached for watertight cable penetration into the chassis. To test for leaks the GoPro and a light source were ran inside the chassis and then the chassis was latched shut and placed into the water. Air was vacuumed out to the equivalent depth of 22 feet (9.54 psi), and at different depths of equivalent pressure (for example 5 feet of water) the chassis was prodded to produce a sound to correlate the video with current external pressure. This was done so that leaks could be correlated with the pressure level where they occurred.

*Figure 3.6: Pictures of weld porosity and leak sources during pressure testing.*

It was apparent that increased external pressure caused significantly more leaks than low pressure as water could quickly make its way through microscopic porosity in the welds or heat affected zones. Once leaks were identified through the use of the GoPro video, they were either touched up by re-welding them, or using marine grade epoxy and vacuuming out the air in the chassis again to pull the epoxy through the porosity. Once apparent leaks were sealed the manufacturing process could continue by adding more gland holes and redoing the vacuum testing to ensure the addition of the glands or their holes did not compromise the reliability of the water tight chassis. This iteration of testing proved the chassis watertight and verified the selection of modular wire glands.

# Chapter 4 - Propulsion Subteam

## Objectives and Constraints

### Design Constraints

- The internal volume of the Mark II of the chassis - 10.62 Liters
- Maximum 60VDC open circuit voltage
- Chassis must be battery powered with all batteries sealed to reduce hazard from acid or caustic electrolytes
- The vehicle must contain a clearly marked “Kill Switch” - A physical disconnect from power to the motors

### Design Objectives

- Design and Test an Environment Measurement System (EMS) - A system that tracks current, battery voltage and temperature of the AUV
- Solidify the design layout of components for motor control and the Environment Measurement System
- Use the EMS during pool testing to record data over time and characterize motor behavior and internal tempera
- Create a “cleaner” system set up with the use of reliable connectors and reducing individual pin connections
- Select and install a Kill Switch

## Design Summary

**Propulsion Subteam - Subsystem Break Down**

*Figure 4.1: Propulsion Power Distribution Diagram*

### Motivation

The propulsion subteam reflected on the first pool test and came up with the objectives listed above. A new timeline was developed that would highlight and prioritize having reliable datalogging and creating an easier and quicker propulsion layout. Moving forward the propulsion subsystems will be referred to as Motor Actuation and the Environment Measurement System (EMS).

### Power Distribution

Two 14.8V LiPo batteries were used to create to individual power rails. “LiPo1” is strictly powering the ESCs and Motors. “LiPo2” is strictly powering the processing hardware devices such as the Raspberry Pi, the Arduino Uno, and any sensors and vision hardware added in the future. Some examples of future hardware are cameras, and the Jetson image processing board. Between the two batteries there is only a total voltage of 29.6V powering the hardware, which does not exceed the maximum 60V constraint. The two power rails are separate to ensure a mechanical disconnect from the power to motors does not disrupt the decision making capabilities of the “smart” hardware components.

### Motor Actuation

Initially, the motors were going to be driven by an Arduino Mega that was receiving motor states and PWM values from a Raspberry Pi. Serial communication between the Pi and Mega was unsuccessful, as a lot of time was spent trying to get the Raspberry Pi to send a complete string to the Mega. In the interest of time, it was deemed more practical to remove the Mega and use the Raspberry Pi to send PWM signals to the motors. Some tradeoffs of making this decision are that Raspberry Pis tend to have noisy PWM channels, so to compensate for this a Raspberry Pi - HAT, Hardware Attached on Top, was purchased. The HAT is a stackable board that supports an autoconfiguration system which allows automatic General Purpose Input/Output, GPIO, setup and driver set up. The Pi would still need to periodically check on the motor current, temperature values being detected by the EMS as a precaution during autonomy.

### Environment Measurement System

The Environment Measurement System, EMS, is a data logging system that is meant to monitor and record current drawn from all six motors, temperature distribution inside of the chassis and battery voltage. A few things the EMS looks for are overcurrent, extremely high temperatures inside of the chassis and low battery voltage. The total number of sensors being used is 23; 6 hall effect sensors, 16 thermistors and 1 voltage measurement from the first cell of the Lipo battery. The Arduino Uno does not have enough analog ports available to handle all 23 sensors so a multiplexer, MUX shield was implemented that created access to a total of 48 analog digital input output, I/O, ports. Ultimately, the EMS was used to record live data during pool testing and store those values to an Excel file for later analysis.

## Testing

### Procedures

1. Hall Effect Validation Procedure
    a. Connect hall effect sensor in series with multimeter
    b. Connect analog pin, $V_{CC}$, and GND from hall effect to Arduino Uno
    c. Once connected, connect to power supply and increase voltage
    d. Compare the results from multimeter to Arduino uno serial monitor readings
        i. If no reading is recorded on Serial monitor or if multimeter is unable to record a current value when voltage is increased, the hall effect sensor may be defective.

*Figure 4.2: Hall Effect Test Circuit Set Up*

2. Battery Cell Voltage Measurement Procedure
    a. Connect multimeter to battery measurement terminals and record value
    b. Connect analog pin from Arduino Uno to battery measurement terminal
    c. Validate value on serial monitor is as recorded from multimeter
3. Thermistor String Measurement Procedure
    a. Set up thermistor strings to corresponding analog pins on the Arduino Uno
    b. Observe temperature values on the Serial Monitor from Arduino IDE
    c. If temperature reading was read at ambient temperature 23℃, the thermistor string was recording correct temperature
        i. If value recorded was imprecise, there is most likely a hardware issue with the thermistor string. This issue was often a short in the thermistor and can be detected using the continuation setting of the multimeter

## Results

### *Approach:*

*Figure 4.3: Matlab Program Flow Diagram*

Data Processing was designed to quickly plot test data in an easy to review fashion using matlab figures, images, and videos. All data sets from all test days can be systematically processed for comparison, review, verification, and report. The program deals with pressure, temperature, current, voltage, and IMU data at its current state but can be easily modified to include new data types. The plotting of pressure, voltage, and current is basic and just applies formatting. The interpretation of temperature and IMU data is more complex.

The temperature plotting uses 3 dimensional linear interpolation between the 16 temperature sensors to create a 3 dimensional array of temperature data at all points between the sensors, to a specified resolution. This allows cross sectional planes to be chosen from that array of temperature data corresponding to areas of interest. These planes are plotted and then the plots used to make the frames of a video which can be reviewed and compared to events during testing to assess where heat is generated in the submarine and its magnitude. Analysis of this type serves two purposes: to assess the layout of the components, where the submarine is coldest and if components must be moved; verification that no part of the interior of the submarine reaches or surpasses the requirement set for maximum temperature during any event.

*Figure 4.4: Current Data During Pool Testing*

From the figure above, propulsion was able to analyze how much current was being drawn when the AUV was in motion. In the figure above at around sample 400, the y-axis motors are hardly drawing any current through them, whereas, the z-axis and x-axis motors are pulling large amounts of current through them. During one of the pool tests, the AUV was placed inside the water at an awkward angle. This forced the PID control to turn on and force the AUV into its natural stable state. Moreover, at sample 600 and higher, there is very little current activity through the z-axis, whereas the y-axis and x-axis motors are drawing a considerable amount of current. From this, the propulsion team was able to infer that the AUV was being driven in a straight line and trying avoid turning upside down. The y-axis motors moved the AUV in a forward direction, whereas the x-axis motors reduced the amount of roll the AUV experienced.

*Temperature Data*

*Figure 4.5: Temperature Gradient Inside of Chassis*

Thermal data was used to create video plots shown as screenshots in Figure 4.5. These plots allow for inference of the thermal properties of the AUV and its components during pool testing. After a short period of testing it can be seen in the left picture of Figure 4.5 that heat did not build up in the entire AUV, and was concentrated to the electronic speed controllers driving the forward motion of the AUV. After extensive loading and testing, heat generation was primarily concentrated to the electronic speed controllers, but also generated by the battery as its voltage dropped due to current draw from the motors. The battery heat is shown by heated volume in AUV's back right corner, or the upper left side of the right picture of Figure 4.5. The larger area of heat in the right picture of Figure 4.5, near the front edge of the sub, correlate to the forward driving electronic speed controllers (ESC) which were under the most load. Three ESC's were placed on the floor of the front and three were placed on the floor in the back of the AUV, but it is apparent which generate more heat under intense forward motion with attitude control. This aspect of the analysis shows the usefulness of the thermal data in this manner. After putting intensive load on the motors and testing for periods of time equal to or longer than competition length of 15 minutes, the thermal data verified that the AUV can dissipate heat fast enough to remain under the required maximum temperature of 60 ℃. The maximum temperature inside the AUV after all testing was roughly 32.4 ℃.

# Chapter 5 - Vision and Sensors Subteam

## Vision Hardware

Researching a new processing board took careful consideration with many factors such as processing power, power consumption, and future relevance. Ultimately, the Nvidia Jetson TX2 was chosen as the new processing board. Figure 5.1 below shows an image of the TX2 board along with all the relevant expansion ports, and features labeled.

*Figure 5.1: Jetson TX2 Board*

*Figure 5.2: Jetson and Joule specs and prices*
A table comparing the specs of the Jetson and Joule is shown in figure 5.2.

## CUDA

Recent developments in technology have pushed many applications to run on a GPU, graphics processing unit, because the complexity of the problem is too much for a CPU, computer processing unit, to handle. One of the main reasons this board was chosen was because of a technology called CUDA. From Nvidia's website: "In GPU-accelerated applications, the sequential part of the workload runs on the CPU – which is optimized for single-threaded performance – while the compute intensive portion of the application runs on thouimized for single-threaded performance – while the compute intensive portion of the application runs on thousands of GPU cores in parallel." Essentially this means that Nvidia has developed hardware called a CUDA core that is designed so that complex problems can be executed in a divide and conquer manner. Apart from parallelism being a huge advantage, the fact that a CUDA core runs according to a specific ISA is also a huge advantage. The CPU included in the Jetson board called project Denver is a dual core ARM processor. The main purpose of this processor is to interface directly with the PCIE express lanes. When the GPU finishes a computation, it writes the data to the onboard SRAM (Static random access memory). This memory is connected to a memory buffer and is bussed out through the PCIE express lanes. The Denver CPU will directly read this buffer to get the processed data. This process bypasses the main memory subsystem which saves tens of thousands of clock cycles. The CUDA ISA makes this entire process possible and significantly speeds up performance. Exploiting the power of the GPU to run intense image processing algorithms was the main task of this semester. Because the main purpose of the sensors team is to gather information about the submarines surroundings, the image processing will be done on images taken by a camera mounted on the submarines chassis.

## Cameras and Interconnects

The goal of the camera subsystem is as follows: ensure a uniform standard around all camera interconnections, and expandability. The quality of the image is less important because during processing, the image will be down scaled to a smaller resolution. Scaling down the image is critical because the processing board will be run on a battery and the goal is to maximize performance while using as little power as possible. If the image is scaled down, the processor has less pixels to process, which in turn leads to lower power consumption. The first task when selecting the cameras was to research a uniform interface. There is an interconnect called CSI, or Camera serial interface. This is a 15 pin connector that typically attaches directly to a logic board and communicates directly with the cameras hardware instead of using software to control it. The Jetson TX2 has the ability to interface with 6 different cameras at the same time using a special expansion board. The Auvidea J20 expansion board is a CSI expander board specifically designed for the Jetson TX2.

## Computer Vision

For computer vision, a thought experiment was considered to compare how a person would detect, classify and track objects versus how a computer would do the same. If a human being is shown a red circle, the person would be able to identify that it something red and it has the shape of a circle because they would have seen many of these in their lifetime. With this information, the person could classify it as a red circle. If the red circle was to move around within their field of view, the person would be able to notice that the object is moving. In the case that the red circle moves out of the field of view, the person would realize that they could no longer see the object, but would remember that they they had seen the object in the past.

A computer is not a sapient being by default. It must be taught to do a certain task, and once the task is taught correctly, the computer will carry this task out flawlessly every time. Consider a frame taken in by a camera. In order to make it so a computer can detect something in a frame, it needs to know that it is trying to detect. This is done by means of a mask. A mask is a basic outline of the object that needs to be detected. For a circle, a mask would look like something pictured in figure 5.3.

*Figure 5.3: Circle mask*

To a human, this is obviously a circle. A computer must break it down into sub-mages and compare them against a frame taken in by a camera, called an environment. This environment must also be broken down into sub-images for the sub-images of the mask to be compared to. Basically, the object that needs to be detected in the frame needs to match the mask. A function was created that could do this. Figure 5.4 is an example of an environment that could be taken in from the camera, with the orange ball being the object that needs to be identified.

*Figure 5.4: Environment*

It can be noticed that the mask in figure 5.3 looks nothing like the orange ball.

In order to make the orange ball look as close to the mask as possible, image preprocessing was required. This was done by using functions developed in the Fall 2017 semester and simple image processing functions in OPENCV. After image processing, the environment can be represented by figure 5.5.

*Figure 5.5: Preprocessed Environment*

When this modified environment is compared against the mask, it now looks much more similar. The mask is essentially run across the environment until there is a high probability that the mask matches something in the environment. When this happens, there can be a reasonable certainty that the desired object was detected. Once an object was detected, a visual confirmation was needed to show that the object was detected. This was accomplished by creating a function that could draw a bounding box around the detected object as well as create a region of interest around the object. This will be discussed later. The boxed object can be seen in figure 5.6.

*Figure 5.6: Boxed Object*

The box around the ball is noticeably off center. This is because the center of the box is where the detection algorithm believes the perfect mask/object match is. This margin of error was acceptable because the information that needs to be extracted could still be found. Using the region of interested (mentioned previously) that was made, information such as the center point location and color of the object can be found using the functions created in the fall semester. These bits of information include the

color of the object and the center point location. It seemed as though the object detection and tracking algorithm was complete at this point. However, the algorithm was only developed for a single environment.

The submarine will be taking in a video, which is a continuous stream of images. Each one of these images could contain information that can be used for the overall functionality of the submarine. If there is not an object in the frame, then nothing should not be detected. As soon as an object comes into the frame it, needs to be detected. This was accomplished by running the detection algorithm every certain amount of frames. Once an object is detected, the bounding box is drawn. Using the bounding box, there are algorithms in OPENCV that can track an object and retain information about where the object is and where it is going, essentially updating the center point location based of where it should be in the next environment (frame) in a video. The functions can also prevent the loss of sight of an object when the object is covered/obscured, something that detection would not be able to compensate. The tracking function can also detect tracking failures, which can happen when an object moves out of the line of sight. Running detection every certain amount of frames was also effective because it could re-detect objects as they re-entered the environment.

It was important to know that the code needed to be written in a way that other shapes, colors, and ultimately objects could be detected, classified and tracked. The foundation was laid so code developers in the projects future could tailor the code to accomplish the tasks given by the competition in future years.

## Inertial Measurement Unit

Dead reckoning is knowing where the team is with respect to where the team has started. This was one of the desired auxiliary functions of the IMU for the sub. It will only be used for obtaining more points as it will keep a map of optional tasks that the sub can come back to if time permits. There are several different sensors that can be used to obtain this “map,” however, using an IMU becomes very complex when due to drift which is a nonzero output with zero input signal. The effects of drift and noise will need to be removed from the output of the IMU in all measurements and from every integration of the data. If the drift was not removed, any velocity or positional data obtained from the IMU will be useless. Especially for positional data as any noise or drift will be a function of time increasing every second. The flow

chart for removing the drift and noise and obtaining positional data is shown below in Figure 5.7

*Figure 5.7: Flow diagram IMU process*

This task was done by first changing the acceleration axis into the world axis with respect to gravity using the rotational matrix from Mayhony's AHRS algorithm. This algorithm may need to be changed in the future to Madgwick AHRS algorithm as it can be more accurate. The corrected acceleration is then filtered and integrated to obtain velocity. The code used to do this was based upon xioTechnologies MATLAB open source code and extended to function with live data.

Although MATLAB is incredibly powerful programming language but, it does not work on every device. Which is why it was converted to python. The structure of the library was made to be incredibly modular, and be able to work with any IMU and multiple ones at the same time. Each line in the MATLAB script needed to be translated into python one at a time. This was quite difficult as the functions that were used in MATLAB had no one to one translation into python. In addition to this, python does not handle array math as easily as MATLAB does.

## Pressure Transducers

Pressure transducers will be incorporated into this project for several reasons. The main reason is to determine depth since surfacing too soon would disqualify the attempted run during competition. The

goal is to have 6-inch resolution in depth, determined by averaging the two pressure transducers. The pressure transducers will need to be able to withstand the pressures and not leak. They will also, need to be able to be removed from the sub as they will need new calibration cures from time to time.

Two pressure transducers were donated to the team at the beginning of the Fall semester in 2017 by Gene Fatton. These pressure transducers were chosen due to their design and ease of implementation. There are several other types of pressure transducers on the market but, they tend to more expensive and harder to attach to the sub comparatively. These specific pressure transducers output 0 to 100 mV absolute pressure which is why high precision analog to digital converter was purchased. The output from the pressure transducers is in mV and is needed to be correlated to actual pressure and ultimately depth. The was done by making a calibration curve using the universal gas law shown below.

$$pv = nRT$$

Where p, v, n, R, and T is pressure, volume, number of moles, universal gas constant, and temperature respectively. The syringe was attached to the pressure transducer and confirmed to be air tight by compressing the air in the syringe near to 100mV and kept there for 60 seconds. There was no observable decrease in voltage over this time period. The initial temperature, volume and pressure were known based on empirical measures and the altitude of Fort Collins. The temperature was measured thought the experiment to ensure the accuracy of the calibration curve. The pressure was changed by changing the volume of the syringe and at each volume step the voltage output was recorded crating the calibration cure as seen below.

F*igure 5.8: Voltage pressure calibration curve*

The relationship between pressure and voltage has the form mX+b and has an $R^2$ of near unity. Within the pressure range expected to be seen at the competition which is between 1 to 2 atmospheres. This process was repeated for both pressure transducers, as every transducer is slightly different.

The code framework that reads the voltage output from the pressure transducers was formatted in a way that will allow for any and as many transducers to be connected at one time as physically possible. The child class for these specific pressure transducers takes in the ‘m’ and ‘b’ parameters determined by

the calibration curve and outputs the pressure directly. The framework also has its own data logging portion that will write a csv file continuously until measurements have stopped. This will allow for any data gathered to be stored for further analysis and processing.

## Testing and Verification

With the increasing size of the project and the code included, it became important to thoroughly test and verify the functionality of the different sensors code. Three major steps were taken in the approach to confirm the accuracy and functionality of the code: the first was creating test cases, the second creating test files for the test cases, and finally using a software called coverage.py to monitor which lines of code were active when running the test cases. The purpose of the test cases is to exercise the different functions within the main code as to isolate the amount of code being run to a smaller and more manageable size. This allows for more in depth testing of a certain section of the main computational code and narrows down potential error and bugs. Now that the code is broken down into more manageable portions for testing, the next step is to pass specific sets of input data called test files. The test case dictates a general form of input data such as an array of values or an image, but test files can be variations of this general form of input. An example would be a test case designed to take an image and highlight the largest rectangle within that image. Possible test files could include an image of different sizes of rectangles, differing numbers of rectangles, adding different shapes and changing the colors of these shapes in an attempt to find an input that the tested code cannot properly handle. Specifically, it is beneficial to pass test files that a code being tested was not designed to interpret when it was designed, such as sending a circle to the example instead of a rectangle and examining how the code processes this data.

A tool that is helpful in analyzing how the code can interpret an input is by using a debugger to procedurally check each line as the code runs. However if there are numerous loops or conditionals, this can become tedious. Another tool that helps speed up this process is to use a coverage utility, in this case coverage.py was used. Coverage.py functions by monitoring which lines of the code were used in running a test file with a test case. This becomes useful when checking conditional statements as well as loops because coverage.py states if line was ever used, not used or partially used. Figure 5.9 False Statement shows a simple “if else” statement where the statement is never true because the variable “number” is always 0,1,2 or 3. As such line 5 is never used and is marked as “missing” and the line 4 marked as “partial” (Line 4 only ever jumps to line 6, not both line 5 and 6). Figure 5.10 True and False Statement shows the same code but the condition is true half of the time. Both lines 5 and 6 are used at some point during the runtime, and as such are marked as “run”. As such, a quick run with coverage.py can reveal if a condition is ever met instead of traversing the code line by line. These three methods of testing are helpful in discovering lapses in the code’s capabilities and exposing bugs as well as potential errors.

*Figure 5.9 False Statement*

*Figure 5.10 True and False Statement*

# Chapter 6 - Controls Subteam

## Objectives and Constraints

In past semesters objectives had never been concretely defined for the Controls subteam. Sure there were loose goals that pertain to the project as a whole, but none of the other subteams systems were to a point where Controls had a set endpoint. This semester Controls worked hard with the rest of the team to set an end goal for the semester. This whole team goal was then split up for the Controls team itself.

Our main objective as a team was to have the AUV in the water and operating alone while tracking an orange ball. This was coined, “the orange ball test.” In order for this to happen, the Controls team had to allow a Raspberry Pi to control the Motors on the AUV, Stabilize the AUV while it was in the water, and receive camera information to make decisions on the AUV’s surroundings.

Early on in the first semester the Robosub team set several constraints for the subteams. On an organizational level the Controls subteam was split up between the jurisdiction of the Propulsion and Sensors subteams. Each of the Controls members were delegated to work with the Propulsion and Sensors teams and as a consequence, finances were handled as if each Controls team member were part of the delegated team. On a technical level all of the team's permanent code was to be made in python. This was done to allow all subteams to have compatible code. Python was chosen because of Sensors pre-existing use of the OpenCV vision processing packages.

Just like last year's team, the first semester this year was focused on pursuing proof of concept ideas. This allowed the team to develop a rough hierarchy of what inputs and outputs each system was going to require. Allowing the Controls subteam to focus on designing code to represent these IO constraints. Early development was performed in part using a Parallax Robot. This tool allowed the team to test basic control systems coding on a known, and well documented, platform. These systems would then be implemented in the Spring semester.

Being a part of both the Propulsion and Sensors subteam gave the Controls subteam a better overview of how the project was developing. This allowed the team to gain operational knowledge of both the software and hardware of the project. This insight was fundamental to assisting with communication between the subteams. The Controls team integration into the other subteams allowed a fundamental understanding of how the other subteams were progressing. This handshake proved to be a great asset for the communication between the subteams.

## Decisions and Accomplishments

At the beginning of the semester the team started by tackling the challenge of getting a Raspberry Pi to control the motors of the AUV. In the previous year, an arduino was used to control the ESC for testing. So, this year, the team had decided to use this in conjunction the Raspberry Pi. The thought was to communicate with the Arduino over USB serially. This proved to be very difficult. After a lot of time the communication proved to be too unstable for a mission critical system. Eventually it was decided to use an Adafruit 12 bit PWM hat. This hardware was well documented and proven to work reliably.

To interface with the PWM hat a class had to be written. A pulse width modulated (PWM) signal used to control electronic speed controllers (ESC) has several characteristics specific to the project. The ESC expects a signal of 50Hz with a time high between 1000us and 2000us. The dead band, when there is no motor movement, is 1500us time high. Any value higher than 1500us moves the motor forward and anything less moves the motor in reverse. A control scheme was decided upon between Controls and Propulsion. This worked with an armed and disarmed state. In the disarmed state no signals are being sent to the motors that means that the PWM signal has a time high of zero. The ESC reacts to this state by not moving and making the Motors emit an audible beeping. This feature allows the operators of the sub to know that the AUV is safe to approach without fear of injury. In the armed state the PWM hat is emitting a signal with 1500us time high and if commanded to will change the motor speeds so that the time high changes bounded by 1000us to 2000us.

One design challenge that had to be overcome when creating this interface was translating the timehigh commands into bit based commands. The PWM hat operates by setting register values on the board. For the operator to control the time high a 12 bit value is used equating to a number between 0 and 4095. This means that theoretically each bit number is equal to:

$$50Hz = 50\ ocilations/1period = 20000us/1period$$
$$20000us/period * 1period/4096bits = 4.88us/bit$$

This should result in an easy translation from a 1000us to 2000us command but in actuality there was considerable error. To overcome this a mapping function was used with the actual bit values. Using a guess and check method the team found the actual bit values that caused the PWM hat to generate a 1000us, 1500us, and 2000us signal. Using these values the team then mapped the commands in the following fassion.

| Commands | Bit Value = command * 1bit/4.88us | Actual Found Value | Result |
|---|---|---|---|
| 1000 | 205 | 213 | Full Reverse |
| 1001-1499 | 205-307 | 213-320 | Reverse |
| 1500 | 307 | 320 | Full Stop |
| 1501-1999 | 308-410 | 320-4255 | Foreword |
| 2000 | 410 | 425 | Full Forward |

*Table 6.1: Commands Mapping*

This mapping method allowed for accurate reproduction of motor control.

The next hurdle the Controls team had to jump through was stabilizing the AUV. A PID controller was chosen to be used for stabilization. This is a more traditional closed feedback controller typically used by drones. Using the diagram below the team programmed a controller in python.

*Figure 6.1: PID Feedback Diagram*

*Source:* https://commons.wikimedia.org/wiki/File:Pid-feedback-nct-int-correct.png

The class was designed to be as general as possible. That way it could be expanded for any stabilization application that would be required by the project. As it stands the controller is only being used to stabilize pitch and roll. The PID controller is threaded so that the user only has to start the controller and it will update in the background. Then its values can be called when the shell program is ready to read from the PID. Recently it was reported to the team that this method might cause some instability. Future teams might look into replacing this with more stable updating method.

The controller was difficult to test. The team struggled to find a way to simulate the open loop function of the AUV's chassis in water. As a result the team ended up relying on pool testing to fine tune the PID controller. At the end of the final pool test the AUV was only marginally stable.

As the semester progressed the team identified that they needed a way for other subteams to test their systems. In the past year one team member created an Arduino program that allowed a game controller to operate the motors oven a cable. This program was great for testing the Mechanical and Propulsion team's systems but it was unable to allow the Controls and Sensors teams to test their work. It also was unable to control the sub fully in 3d space. As a result the team designed a game controller class to work on the Raspberry Pi that could control all of the subs functions.

There are several different libraries that were evaluated to use to make the Controller Class. These were the pygames controller library, Approximate Engineerings python game controller library, and Inputs. Ultimately inputs was chose because of its documentation and low resource usage. It uses xinput driver that easily detected the xbox one controller being used. There are several issue that the team ran into with the Inputs methods. One was that Inputs was interrupt based. This made the class easy to program but caused the controller to not detect sudden, violent movements. In the water the AUV would become uncontrollable at times. A button was allocated to make the controller's commands causing the AUV to come back under control. Another issue happened when the controller was disconnected. It would cause all controller commands to remain the same as the instant before the controller disconnected, resulting in the sub being stuck with no way to clear the command. To avoid any serious injury because of this issue the AUV disarms itself when the controller disconnects.

*Figure 6.2: Control Scheme*
*Source:* https://exmus506210.wordpress.com/2013/11/12/week-3-player-inputs/

The resulting controller code allowed the Controls subteam to test the PID functionality of the AUV. Unfortunately there was not enough time during the 2017-2018 academic year to fully integrate the vision and pressure sensing parts of the Sensors team work. It is up to the next team to work on those portions.

## Results

At the beginning of the year Controls started with a seemingly overwhelming amount of work and little direction from the previous years as to how to proceed. In the previous semester the team worked hard to help other subteams get their systems to a point so that Controls could get progress done. As everything stands at the end of the 2017-2018 academic year the Controls team is leaving behind a reliable way to control the AUV's motors. The stabilization controller is in a functional state but needs work before it

will stabilize the AUV to acceptable ranges. And finally the team leave behind a controller class that allows the teams to test their systems while in development. There is early development of tracking code that can be modified in the future to track more than just an orange ball. Once the characteristics of different objects are defined and passed onto the Controls subteam, the system will be able to track any object.

# Chapter 7 - Conclusions and Future Work

## Mechanical Subteam

### Final Design Summary

The mechanical designs produced over the 2017-2018 were tailored to meet weight, temperature, flow, and space requirements primarily while keeping cost low and manufacturability high. The design was created to keep modularity in mind and provide a reliable and sound chassis for further progress to be made on in later semesters. The aluminum chassis provides a sound structural base and allows for components to be attached to it easily, while remaining light and being thermally conductive. The chassis was designed to be square to maximize the efficiency of the layout of components and reduce wasted space. The chassis also has a small front surface area to minimize drag which is further reduced in its forward motion with the addition of hydrodynamic fairings directing the flow over the sub and reducing the high pressure zone in front of it. Electronic components are mounted internally on modular boards for easy access and efficient installation while staying tight to the chassis during operation. The minimized size of the submarine allows for it to have manageable buoyancy and a ballast system is under development which will provide buoyant force control and a safety system. Further models will integrate the torpedo launcher and mechanical arm which have been developed to some degree and will be added to the chassis via welding and shielded from poor flow by additional fairings.

### Problems and Proposed Solutions

A major area of importance to the successful function of the submarine is the maintenance of its watertight systems. Some problems that have been noted through full system testing are:

- The lid seal as not as easy to put on as it needs to be. It currently requires two skilled team members to ensure correct contact and is a point of stress in knowing if the seal collapsed correctly.
- The lid seal material will start to deform and lose its shape over repeated tests under high pressure.
- Under high pressure the lid will hit its mechanical stop on the bottom half of the chassis and cause the latches to be slightly loose.
- The wire glands should be replaced after so many uses from the wear on the wire gripping o-ring and wear on the plastic.

These issues can be overcome with some changes in the size and some standard operating procedures. Reducing the size of the seal slightly would allow it to be stretched over the seal holders tightly and keep it away from the overlapping edge of the top and bottom sections of the chassis. This

way the lid could be latched down without the worry of the seal moving in any way or getting caught between the overlapping edge of the two pieces. For standard operating procedures, both the seal and the glands will need a tally of uses so they can be replaced after they have reached a certain number of loading cycles. Lastly the latches can either be tightened by increasing the height of the mechanical stop or by increasing the density of the lid sealing material, or an external component can be added which ensures that the latches remain locked, even if loose, until they are unlocked. The latches are not prone to come undone on their own when they are relatively loose, but they are prone to coming undone if some force unexpectedly acts on them.

### Future Work

With the completion of a reliably water tight and fully functional submarine chassis, future work can be focussed on completing the external and internal mechanical subsystems. This includes: the mechanical arm, the ballast system, and the torpedo launcher. These systems have been under some amount of development, but need the attention of at least one student, preferably two, per system for a year. This would ensure that the systems are robust and then are packaged into the final design as streamlined as they were initially intended to be.

The ballast system has the most work completed in modelling and integration with the other systems. In the future it needs to be manufactured and prototyped multiple times with its electrical driving system so the control, water tight reliability, and size can be dialed in. The ballast system will function as the safety device of the submarine, and is an imperative device for the successful autonomous functionality.

The mechanical arm was initially modelled to be relatively simple, and to meet requirements that were tailored for the 2017 and 2018 competition tasks. These requirements should be reviewed and many of them deprecated. New requirements need to be written detailing the size constraints of the manipulator, but the best option and most easily implementable mechanical arm would be a 3 rotational joint arm with a 3 joint end effector. The arm should be streamlined and use fairings to maintain good flow while it is not in use. This allows for the modularity to complete varied tasks and this arm is a commonly studied and implemented configuration.

The torpedo launcher needs to be reformulated to be packaged in a streamline manner to the submarine, also making use of fairings. Since the controlling mechanism for the launcher is relatively simple, the majority of the effort spent on this extremenity should be focused on the torpedo design itself for accurate flight, and the actual packing of the launcher in or on the submarine.

## Propulsion

After the last pool test, it was concluded that the propulsion systems performed successfully while logging temperature and current. The EMS can readily accept more values such as pressure, and IMU data. The battery ran for the maximum duration of 15 minutes, and the motors and ESCs performed at 50% and 100% capacity without overheating inside of the chassis. Moving forward, the EMS code will need some minor development as the Controls Subteam brings the AUV closer to autonomy. At which rate, that would be defining the absolute maximum chassis temperature and current and recognizing when

the battery is too low. There are no recommendation to continue the project with a “Propulsion” team for future iterations. A few things that could be refined would be:

- Design a simple printed circuit board for the thermistor strings to reference a single resistor as opposed to having a ton of running wire through the sub going to the multiplexor
- Reduce the number of thermistors to just 6-8 for monitoring high risk hardware items
- Get a connector with ribbon cable attachments to replace hand soldered connector and wires
- Heat shrink and conformal coat everything for waterproofing
- Make sure there is no exposed wire to reduce possible shorts

## Sensors

There will always be a lot to do for sensors as there are advancements in sensors and computers. They are becoming more powerful, sensitive and smaller. Currently the AUV needs more vision coding so it is able to follow a line and recognize more objects. A lot of the base code is there, it just needs to be expanded. The dead dead reckoning from obtained solely from the IMU still need some work. Specifically in the area of filtering. As already mentioned the code that hands all of the data is already there, the focus should be getting python filters to remove drift from the acceleration and gyroscope. This should not be priority as there are other possible methods of obtaining a dead reckoning map that could be more accurate and easier to implement. One method would be to use a hydrophone array in combination with pressure transducers. The hydrophone array would be able to accurately determine the change in AUVs x and y coordinates and the and the pressure transducers can determine z component of position relative to the surface. With that being said, there have been many advancements this semester. Vision is able to draw a bounding box around a ball and track it. OpenCV has been installed onto several different computers that can be used in the AUV including a Raspberry pi. A PID has be created that takes in orientation information from an IMU, makes judgments and controls the motors. Several pressure transducers have been calibrated and install onto the AUV that will be able to determine dept of the AUV. As previously mentioned, there will always be more work for the sensors sub team and should be comprised of electrical engineers and computer scientists.

## Controls

Future teams will have quite the task ahead of them but the 2017-2018 Controls team has some recommendations that will hopefully give them more guidance than they had. The first bit of advice is on an organizational level. Make sure that all of controls subteam stuff is seperate. The controls team is an independant subteam. The 2017-2018 team spent a lot of time going to meetings that didn't really affect their subteam. The controls team should not have more than one meeting per week. Whether that is going to another sub-teams meeting, meeting as a Controls team, or going to a whole team meeting. An ideal situation for Controls would be to have the whole team work on a three week rotation. One week go to the whole team meeting, the next week Controls would meet as a team, and the last Controls might split up, having each member meet with the other subteams. Depending on the whole team’s subteam composition, it is important that every Controls member has an idea of what all other sub-teams are working on, including the mechanical team.

On a technical level, future Controls teams need to work on stabilization. Then, move on to integrating the sensors and vision. Having a reliably stable chassis is important to have all other

subsystems work properly. Once the AUV is stable then Controls can start working on navigation, choosing a method as to how the AUV will make decisions for movement.

# Appendix-A

AUV - Autonomous Underwater Vehicle
IO - Input/Output
ESC - Electronic Speed Control
PID - Proportional Integral Derivative
PWM - Pulse Width Modulation
USB - Universal Serial Bus
IMU - Inertial Measurement Unit

# Appendix-B - Budget and Finance

| | | | |
|---|---|---|---|
| **Total Funds** | $15,331.00 | NOTE: Total includes AY17-18 beginning balance + $2500 from HP | |
| **Competition Savings** | $4,331.00 | | |
| **Total Sub Funds** | $11,000.00 | | |
| | | | |
| | **Spent** | **Budget** | |
| **Mechanical** | $ 1,788.99 | $ 3,300.00 | |
| **Propulsion** | $ 2,604.79 | $ 4,400.00 | |
| **Sensors** | $ 1,185.97 | $ 3,300.00 | |

| | | | |
|---|---|---|---|
| **Total** | $ 5,579.75 | $ 11,000.00 | 51% |
| | Note: Total Spent only includes items purchased using P-Card | | |
| | | | |
| **Total Remaining:** | $9751.25 | | |
| | | | |

| **Mechanical** | | | | |
|---|---|---|---|---|
| Item | Budget | Spent | Remaining | % Remaining |
| Chassis | $ 1,500.00 | $ 1,461.81 | $ 38.19 | 3% |
| Ballast | $ 500.00 | $ 66.67 | $ 433.33 | 87% |
| Fail Safe | $ 500.00 | $ - | $ 500.00 | 100% |
| Misc | $ 500.00 | $ 149.06 | $ 350.94 | 70% |
| Mechanical Shipping | $ 500.00 | $ 111.45 | $ 388.55 | 78% |
| | | | | |
| Total | $ 3,500.00 | $ 1,788.99 | $ 1,711.01 | 49% |

| **Propulsion** | | | | |
|---|---|---|---|---|
| | | | | |
| | Budget | Spent | Remaining | % Remaining |
| Motors | $ 2,000.00 | $ 1,053.99 | $ 946.01 | 47% |
| Motor Control | $ 600.00 | $ 402.74 | $ 197.26 | 33% |
| Power Supply | $ 800.00 | $ 611.74 | $ 188.26 | 24% |
| Propulsion misc/wiring | $ 500.00 | $ 249.09 | $ 250.91 | 50% |
| Propulsion Shipping | $ 500.00 | $ 287.23 | $ 212.77 | 43% |
| | | | | |
| Total | $ 4,400.00 | $ 2,604.79 | $ 1,795.21 | 41% |

| **Sensors** | | | | |
|---|---|---|---|---|
| | | | | |
| | Available | Spent | Remaining | % Remaining |
| Measurement Units | $ 1,000.00 | $ 101.39 | $ 898.61 | 90% |
| Cameras | $ 600.00 | $ 510.77 | $ 89.23 | 15% |
| Processing Units | $ 700.00 | $ 404.04 | $ 295.96 | 42% |
| Sensors misc | $ 500.00 | $ 96.06 | $ 403.94 | 81% |
| Sensors Shipping | $ 500.00 | $ 73.71 | $ 426.29 | 85% |
| | | | | |
| Total | $ 3,300.00 | $ 1,185.97 | $ 2,114.03 | 64% |

**Appendix-C**

Mechanical Timeline
Fall 2017

| Tasks | Subtasks | Est Start | Est Finish | Actual Start | Actual Finish | Resource | Percent Complete |
|---|---|---|---|---|---|---|---|
| **Prelim Field Testing** | | **9/5/2017** | **9/15/2017** | **9/5/2017** | **9/15/2017** | | **100** |
| | Drag Pull Test | 9/10/2017 | 9/10/2017 | 9/10/2017 | 9/10/2017 | ME's | |
| | Thermal Test | 9/10/2017 | 9/10/2017 | 9/10/2017 | 9/10/2017 | Marta, Jake, Ryan, Daniel | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | CFD Comparison | 9/12/2017 | 9/14/2017 | 9/12/2017 | 9/12/2017 | Ryan | 100 |
| | Test Report Written | 9/10/2017 | 9/14/2017 | 9/10/2017 | 9/15/2017 | Jake | 100 |
| | | | | | | | |
| **CAD V1** | | **9/20/2017** | **9/28/2017** | | | | |
| | Chassis | 9/20/2017 | 9/26/2017 | 9/19/2017 | 9/22/2017 | Everyone | 100 |
| | Motors | 9/22/2017 | 9/26/2017 | 9/21/2017 | 9/21/2017 | Ryan | 100 |
| | Lid/Upper | 9/20/2017 | 9/26/2017 | 9/19/2017 | 9/22/2017 | Everyone | 100 |
| | Fairings | 9/22/2017 | 9/26/2017 | 9/4/2017 | | Everyone | 75 |
| | Camera | 9/22/2017 | 9/26/2017 | | | | |
| | Brackets | 9/22/2017 | 9/26/2017 | 9/22/2017 | 9/22/2017 | Everyone | 100 |
| | Weldments | 9/21/2017 | 9/26/2017 | 9/19/2017 | 9/21/2017 | Everyone | 100 |
| | Seals | 9/20/2017 | 9/26/2017 | 9/21/2017 | 9/28/2017 | Everyone | 100 |
| | Torpedoes | 9/24/2017 | 9/26/2017 | 9/18/2017 | 9/18/2017 | Ryan | 100 |
| | Arm Analog | 9/26/2017 | 9/26/2017 | | | Jake Ryan | 25 |
| | Ballast | 9/22/2017 | 9/26/2018 | | | Daniel | 25 |
| | Assembly | 9/26/2017 | 9/28/2017 | 9/19/2017 | | Everyone | 80 |
| **Computational Fluid Dynamics (CFD) and Finite Element Analysis (FEA)** | | **9/26/2017** | **10/12/2017** | | | | |
| | Hydro Flow | 9/26/2017 | 10/12/2017 | 9/28/2017 | 9/28/2017 | Everyone | 100 |
| | Material Stress | 9/26/2017 | 10/12/2017 | | | Everyone | 0 |
| | Thermal Dissipation | 9/26/2017 | 10/12/2017 | 10/3/2017 | | Everyone | 10 |
| **CAD V2** | | **9/28/2017** | **10/13/2017** | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Strength based updates | 9/28/2017 | 10/13/2017 | | | Everyone | 0 |
| | Hydrodynamics based updates | 9/28/2017 | 10/13/2017 | 9/28/2017 | | Everyone | 100 |
| | Manufacturability based updates | 9/28/2017 | 10/13/2017 | 10/7/2017 | 10/7/2017 | Everyone | 100 |
| | Thermal dissipation based updates | 9/28/2017 | 10/13/2017 | | | | 0 |
| | Mass and Buoyancy updates | | | 9/28/2017 | 10/5/2017 | Everyone | 100 |
| | | | | | | | |
| **Manufacturing** | | **10/12/2017** | **11/17/2017** | | | | |
| | Chassis Frame | 10/12/2017 | 10/19/2017 | 10/7/2017 | 10/7/2017 | Jake | 100 |
| | Lid Frame | 10/12/2017 | 10/19/2017 | 10/7/2017 | 10/7/2017 | Jake | 100 |
| | Lid Plexiglass | 10/19/2017 | 10/26/2017 | 10/10/2017 | 10/10/2017 | Everyone | 100 |
| | Motor Mounts (Holes) | 10/19/2017 | 10/26/2017 | | | | 0 |
| | Seals | 10/19/2017 | 10/26/2017 | 10/10/2017 | 10/10/2017 | Everyone | 100 |
| | Latches | 10/19/2017 | 10/26/2017 | 10/10/2017 | | | 25 |
| | Fairings | 10/26/2017 | 11/9/17 | | | | 0 |
| | Motor Shrouds | 43034 | 43048 | | | | |
| | Torpedoes | 11/9/2017 | 11/17/2017 | 9/18/2017 | 9/18/2017 | Ryan | 75 |
| | Arm Analog | 11/9/2017 | 11/17/2017 | | | | 0 |
| | Ballast | 10/26/2017 | 11/9/2017 | | | | 0 |
| | Wiring Glands | 10/12/2017 | 11/17/2017 | | | | 0 |
| | Camera Mounts | 11/9/2017 | 11/9/2017 | | | | 0 |
| **Physical Testing** | | **11/27/2017** | **12/7/2017** | | | | |
| | Watertightness | | | 10/10/2017 | 10/12/2017 | Everyone | 100 |
| | Hydroflow | | | | | | 0 |
| | Dummy motor test | | | | | | 0 |
| | Thermal dissipation | | | 10/12/2017 | | Everyone | 25 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Impact | | | | | | 0 |
| | Buoyancy | | | 10/10/2017 | | | 10 |
| | Weight | | | 10/10/2017 | | Everyone | 10 |
| | | | | | | | |
| **Milestones** | | | | | | | |
| | Design Decision | 9/20/2017 | | | | | |
| | Fabrication Started | 10/7/2017 | | | | | |
| | Movement Test (No comms, just motors and an input signal) | 11/4/2017 | | | | | |
| | Fabrication Completed | 11/17/2017 | | | | | |

Spring 2018

| **Tasks** | **Subtasks** | **Est Start** | **Est Finish** | **Actual Start** | **Actual Finish** | **Resource** | **Percent Complete** |
|---|---|---|---|---|---|---|---|
| MkII CAD | | 1/15/18 | 2/23/18 | 1/15/18 | | All | 75 |
| | Chassis | 1/15/18 | 2/9/18 | 1/15/18 | 2/9/18 | Jake and Ryan | 100 |
| | Ballast | 1/15/18 | 2/24/18 | 1/15/18 | | Daniel | 50 |
| | Electronics | 1/22/18 | 2/16/18 | 1/22/18 | | Jake | 80 |
| | Fairings | 1/22/18 | 2/9/18 | 1/22/18 | 2/9/18 | Ryan | 100 |
| | | | | | | | |
| Chassis Construction | | 2/8/18 | 2/23/18 | | | All | 0 |
| | CNCing | 2/14/18 | 2/19/18 | | | Jake and Ryan | 0 |
| | Welding | 2/20/18 | 2/21/18 | | | Jake | 0 |
| | Mounting | 2/21/18 | 2/22/18 | | | All | 0 |
| | Seal Cutting | 2/8/18 | 2/23/18 | | | Daniel | 0 |
| | | | | | | | |
| Ballast System Construction | | 2/19/18 | 3/16/18 | | | | 0 |
| | Endcap Machining | 2/19/18 | 2/26/18 | | | Daniel | 0 |
| | Tube Machining | 2/19/18 | 2/26/18 | | | Daniel | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Mechanism Construction | 2/19/18 | 2/26/18 | | | Daniel | 0 |
| | Acme turning | 2/19/18 | 2/26/18 | | | Daniel | 0 |
| | Gear printing | 2/19/18 | 2/26/18 | | | Daniel | 0 |
| | Motor Mounting | 2/23/18 | 3/16/18 | | | Daniel | 0 |
| | Piston Assembly | 2/23/18 | 3/16/18 | | | Daniel | 0 |
| | | | | | | | |
| Fairing Construction | | 2/12/18 | 2/23/18 | | | Ryan | 0 |
| | Mold CAD | 2/12/18 | 2/14/18 | | | Ryan | 0 |
| | Mold Printing | 2/14/18 | 2/16/18 | | | Ryan | 0 |
| | Casting | 2/15/18 | 2/20/18 | | | Ryan | 0 |
| | Tab Construction | 2/19/18 | 2/22/18 | | | Ryan | 0 |
| | Mounting | 2/23/18 | 2/23/18 | | | Ryan | 0 |
| | | | | | | | |
| Testing | | 2/23/18 | 3/22/18 | | | | 0 |
| | Waterproofness | 2/23/18 | 2/23/18 | | | All | 0 |
| | General Movement | 2/27/18 | 3/2/18 | | | All | 0 |
| | Thermal Dissipation | 2/27/18 | 3/2/18 | | | All | 0 |
| | Ballast | 3/22/18 | 3/22/18 | | | All | 0 |
| | Speed Response | 3/15/18 | 3/15/18 | | | All | 0 |

Propulsion Timeline
Fall 2017

| **Tasks** | **Subtasks** | **Est Start** | **Est Finish** | **Actual Start** | **Actual Finish** | **Resource** | **Percent Complete** |
|---|---|---|---|---|---|---|---|
| **Investigating Systems** | Select Motors to achieve desired speed of chassis design | 9/1/2017 | 9/20/2017 | 9/7/2017 | 9/20/2017 | Moeez, Jordan, Marta | 100% |
| | Select a battery to meet desired power needs of motors and other hardware components | 9/1/2017 | 9/20/2017 | 9/7/2017 | | Moeez, Jordan, Marta | 50% |
| | Research hardware interfacing | 9/1/2 | 9/25/ | 9/4/2 | | Nick | 90% |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | methods | 017 | 2017 | 017 | | | |
| | Propulsion to Sensors Comms | | | 9/1/2017 | | Nick and Jeremy | 20% |
| **Design** | Battery Management System Design | 9/1/2017 | 11/1/2017 | 2/1/2017 | 9/20/2017 | Moeez and Jordan | 100% |
| | Printed Circuit Board Design (PCB) | 1/30/2018 | 4/15/2018 | 2/1/2017 | | Moeez, Marta | 20% |
| | Emulate a Sensors signal for motor control testing | 9/1/2017 | 11/1/2017 | | | Nick and Jeremy | 0% |
| | Complete Eagle File for BMS | 8/28/2017 | 9/1/2017 | 8/28/2017 | | Marta | 90% |
| | Motor Control Development (Mock sensors signal to motors) | 11/1/2017 | 11/14/2017 | 10/9/2017 | | Jordan | 85% |
| | IMU to Propulsion Response | 11/1/2017 | 12/1/2017 | | | Nick and Jeremy | 0% |
| **Testing/Design Validation** | Bread board and test Battery Management System | 11/1/2017 | 11/14/2017 | | | Moeez, Jordan and Marta | 0% |
| | Arduino ESC Controls | | | | | Marta | 70% |
| | Arduino - Pi interface | | | | | Nick and Marta | 70% |
| | Hall Effects Sensing Code | 10/16/2017 | 10/23/2017 | 10/23/2017 | | Moeez | 90% |
| | Hall Effects Sensing circuit | 10/23/2017 | 11/3/2017 | 10/25/2017 | | Moeez | 80% |
| | Motor Control Testing | | | | | Jordan | 40% |
| | Mobility Testing/.No Communications (Motors mounted to Chassis) | 11/21/2017 | 11/25/2017 | | | Ryan, Marta, Jordan, Nick, Jeremy | 0% |
| **Manufacture** | Final BMS Design | 11/30/2017 | 1/30/2018 | | | Jordan, Moeez | 0% |
| | Power Systems | 12/1/2017 | 1/30/2018 | | | Jordan, Moeez, Marta | 0% |
| | Propulsion Controls and Feedback | 12/2/2017 | 1/30/2018 | | | Nick, Marta, Moeez, Jordan | 0% |
| | Propulsion Drive | 12/3/2017 | 1/30/2018 | | | Marta, Jordan | 0% |
| | PCB | 4/1/2017 | 5/1/2018 | | | Marta, Moeez | 0% |
| **Milestones** | | | | | | | |
| | Test BMS Design | 11/14/2017 | | | | | |
| | Test Motors on Chassis | 11/25/2017 | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Manufacture BMS | 12/30/2017 | | | | | |
| | Test Motor Control | 11/14/2017 | | | | | |
| | Complete PCB Design | 4/15/2018 | | | | | |
| | Manufacture PCB | 5/1/2018 | | | | | |

Spring 2018

| Tasks | Subtasks | Sub-subtasks | Notes | Est Duration | Est Finish | Actual Start | Actual Finish | Resource | Percent Complete |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| **System Definition** | Identify necessary systems | | | 1 day | | | | | |
| | Define Need/Importance | | | | | | | | |
| | Research system design | | | | | | | | |
| | | | | | | | | | |
| **System Design** | Draw System Block-diagram | Break down the system into subsystems | | 5 days | | | | | |
| | Draw schematic Layout | Draw separate schematic of each subsystem | | | | | | | |
| | Research necessary parts/components | | | | | | | | |
| | Order necessary parts or components | | | | | | | | |
| | Design test circuit for each subsystem | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Write necessary code for functionality | | | | | | | | |
| | | | | | | | | | |
| **System Tests** | Identify System Test Points | Quantify test point values | | 7 days | | | | | |
| | Test I | Identify the "load" for each test circuit | | | | | | | |
| | | Validate outputs | | | | | | | |
| | | Record observations and test results | | | | | | | |
| | | Refine subsystem based on test results | | | | | | | |
| | Test II | Attach actual load to Subsystem | | | | | | | |
| | | Validate outputs | | | | | | | |
| | | Record test point values | | | | | | | |
| | | Refine based on results | | | | | | | |

Sensors Timeline
Fall 2017

| **Tasks** | **Subtasks** | **Est Start** | **Est Finish** | **Actual Start** | **Actual Finish** | **Resource** | **Percent Complete** |
|---|---|---|---|---|---|---|---|
| **Plot sparton live filtered IMU data** | | **3/15/2017** | **11/1/2017** | **3/15/2017** | 4/12/2018 | Billy | 95 |
| | get live data | 3/15/2017 | 3/22/2017 | 3/15/2017 | 3/22/2017 | Billy | 100 |
| | save live data | 3/15/2017 | 3/22/2017 | 3/15/2017 | 3/22/2017 | Billy | 100 |
| | plot saved data | 3/22/2017 | 3/23/2017 | 3/22/2017 | 3/23/2017 | Billy | 100 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | filter saved data (using and comparing different filters) | 3/23/2017 | 11/1/2017 | 3/23/2017 | 11/04/2017 | Billy | 100 |
| | feed in saved data as live data to a filtering function (python) | 3/28/2017 | 3/30/2017 | 3/28/2017 | 3/30/2017 | Billy | 100 |
| | feed in saved data as live data to a filtering function (matlab) | 8/30/2017 | 9/13/2017 | 8/30/2017 | 9/13/2017 | Billy | 100 |
| | plot live filtered accel data | 8/30/2017 | 9/20/2017 | 8/30/2017 | 9/20/2017 | Billy | 100 |
| | plot live filtered 3d accel data | 8/30/2017 | 11/1/2017 | 8/30/2017 | 11/1/2017 | Billy | 95 |
| | plot live filtered3d v data | 8/30/2017 | 11/1/2017 | 8/30/2017 | 11/8/2017 | Billy | 95 |
| | plot live filtered 3d p data | 8/30/2017 | 11/1/2017 | 8/30/2017 | 11/8/2017 | Billy | 95 |
| | plot live 3d data (matlab) | 8/30/2017 | 11/1/2017 | 8/30/2017 | 11/8/2017 | Billy | 90 |
| | plot live data (python) | 3/15/2017 | 10/13/2017 | 3/15/2017 | 4/10/2018 | Billy | 100 |
| | Final Filter implemented into python | 10/17/2017 | 11/1/2017 | 10/17/2017 | | Billy | 85 |
| | fine tune | 10/13/2017 | 10/27/2017 | 10/11/2017 | | Billy | 5 |
| | Pressure Transducer | 11/8/2017 | 12/8/2017 | 3/10/2018 | 4/1/2018 | Billy | 95 |
| **Video Processing** | | | | | | | |
| | Select new vision processor unit | 9/6/2017 | 9/27/2017 | 9/6/2017 | 9/27/2017 | Ross | 100 |
| | learn image filtering techniques | 9/20/2017 | 10/20/2017 | 9/20/2017 | | Oren | 100 |
| | Learn Cuda acceleration for processor stress relief | 9/24/2017 | 10/24/2017 | 9/24/2017 | | Ross | 5 |
| | Detect Object (general case, single image) | 10/2/2017 | 12/10/2017 | 10/2/2017 | | Oren | 100 |
| | Classify Object (single image) | 10/2/2017 | 12/10/2017 | 10/2/2017 | | Oren | 75 |
| | Track Object (video) | 10/2/2017 | 12/10/2017 | 10/2/2017 | | Oren | 100 |
| | Detect and Track Object (video) | 11/15/2017 | 12/10/2017 | | | Oren | 95 |
| | Pass Object Info to Controls | 11/15/2017 | 12/10/2017 | | | Oren/Billy | 50 |
| | Objection Detection in MATLAB (arbitraty case) | 10/2/2017 | 11/1/2017 | 10/2/2017 | 10/27/2017 | Jake | 100 |
| | Parallelization of Object Detection code in MATLAB | 11/1/2017 | 11/7/2017 | 10/27 | 11/4/2017 | Jake | 100 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Port Object Detector to Python | 11/7/2017 | 12/10/2017 | 11/4/2017 | | Jake | 100 |
| | Define size | 11/20/2017 | | 11/8/2017 | | Oren | 10 |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| **Camera Integration** | | | | | | | |
| | Get video feed from raspberry pi camera with TX2 | 11/10/2017 | 11/24/2017 | | | Ross | 0 |
| | Get video feed from GoPro with raspberry pi board using HDMI to CSI 2 bridge | 11/10/2017 | 11/30/2017 | | | Ross | 0 |
| | Write driver interfacing GoPro to Jetson TX2 board | 11/30/2017 | 1/10/2017 | | | Ross | 0 |
| | | | | | | | |
| **Pressure Transducers** | | | | | | | |
| | Write pressure transducer skelton code | 10/18/2017 | 10/25/2017 | 10/18/2017 | 11/1/2017 | Ross | 100 |
| | Get voltage values from pressure transducers | 10/31/2017 | 11/29/2017 | 11/1/2017 | | Ross | 100 |
| | Get third pressure transducer | 11/20/2017 | 12/10/2017 | | | Ross | 0 |
| | Modify the A/D board to add another 5 volt input | 11/30/2017 | 12/10/2017 | | | Ross | 0 |
| | Write driver for raspberry pi for pressure transducers | | | | | Ross | 5 |
| | Output correct Data to controls | 9/18/2017 | 11/24/2017 | 9/18/2017 | | Ross and Jeremy | 0 |
| **Controls** | | | | | | | |
| Develop decision trees for algorithms | | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 12 |
| | Define different algorithms for different processes | 8/30/2017 | 10/11/2017 | 10/14/2017 | | Jeremy | 50 |
| | Interpret data from IMU | 8/30/2017 | 10/25/2017 | 10/14/2017 | | Jeremy | 20 |
| | Interpret data from Cameras | 8/30/2017 | 11/30/2017 | 10/14/2017 | | Jeremy | 5 |
| | Interpret data from hydrophone array | 9/15/2017 | 12/10/2017 | | | Jeremy | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Interpret data from pressure transducer | 10/15/2017 | 11/15/2017 | | | Jeremy | 0 |
| Target search - Cameras | | | | | | | 5 |
| | Integrate Line following commands | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | Integrate Buoy finding commands | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | Integrate Gate finding commands | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | Integrate torpedo target finding command | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | | | | | | | 5 |

Spring 2018

| **Tasks** | **Subtasks** | **Est Start** | **Est Finish** | **Actual Start** | **Actual Finish** | **Resource** | **Percent Complete** |
|---|---|---|---|---|---|---|---|
| **Plot sparton live filtered IMU data** | | **3/15/2017** | **11/1/2017** | **3/15/2017** | | Billy | |
| | get live data | 3/15/2017 | 3/22/2017 | 3/15/2017 | 3/22/2017 | Billy | 100 |
| | save live data | 3/15/2017 | 3/22/2017 | 3/15/2017 | 3/22/2017 | Billy | 100 |
| | plot saved data | 3/22/2017 | 3/23/2017 | 3/22/2017 | 3/23/2017 | Billy | 100 |
| | filter saved data (using and comparing different filters) | 3/23/2017 | 11/1/2017 | 3/23/2017 | 11/04/2017 | Billy | 100 |
| | feed in saved data as live data to a filtering function (python) | 3/28/2017 | 3/30/2017 | 3/28/2017 | 3/30/2017 | Billy | 100 |
| | feed in saved data as live data to a filtering function (matlab) | 8/30/2017 | 9/13/2017 | 8/30/2017 | 9/13/2017 | Billy | 100 |
| | plot live filtered accel data | 8/30/2017 | 9/20/2017 | 8/30/2017 | 9/20/2017 | Billy | 100 |
| | plot live filtered 3d accel data | 8/30/2017 | 11/1/2017 | 8/30/2017 | 11/1/2017 | Billy | 100 |
| | plot live filtered 3d v data | 8/30/2017 | 11/1/2017 | 8/30/2017 | 11/8/2017 | Billy | 100 |
| | plot live filtered 3d p data | 8/30/2017 | 11/1/2017 | 8/30/2017 | | Billy | 99 |
| | plot live 3d data (matlab) | 8/30/2017 | 11/1/2017 | 8/30/2017 | | Billy | 99 |
| | plot live data (python) | 3/15/2017 | 10/13/2017 | 3/15/2017 | | Billy | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Final Filter implemented into python | 10/17/2017 | 11/1/2017 | 10/17/2017 | | Billy | 0 |
| | fine tune | 10/13/2017 | 10/27/2017 | 10/11/2017 | | Billy | 95 |
| | Pressure Transducer | 11/8/2017 | 12/8/2017 | | | Billy | |
| **Video Processing** | | | | | | | |
| | Select new vision processor unit | 9/6/2017 | 9/27/2017 | 9/6/2017 | 9/27/2017 | Ross | 100 |
| | learn image filtering techniques | 9/20/2017 | 10/20/2017 | 9/20/2017 | | Oren | 25 |
| | Learn Cuda acceleration for processor stress relief | 9/24/2017 | 10/24/2017 | 9/24/2017 | | Ross | 5 |
| | Define Line | 10/2/2017 | 12/10/2017 | 10/2/2017 | | Oren/Billy | 0 |
| | Define Buoys | 10/2/2017 | 12/10/2017 | 10/2/2017 | | Oren | 75 |
| | Create object | 10/2/2017 | 12/10/2017 | 10/2/2017 | | Oren | 0 |
| | Update Object | 11/15/2017 | 12/10/2017 | | | Oren | 60 |
| | Pass Object to Controls | 11/15/2017 | 12/10/2017 | | | Oren/Billy | 0 |
| | Objection Detection in MATLAB | 10/2/2017 | 11/1/2017 | 10/2/2017 | 10/27/2017 | Jake | 100 |
| | Parallelization of Object Detection code in MATLAB | 11/1/2017 | 11/7/2017 | 10/27 | 11/4/2017 | Jake | 100 |
| | Port Object Detector to Python | 11/7/2017 | 12/10/2017 | 11/4/2017 | | Jake | 15 |
| | Define size | 11/20/2017 | | 11/8/2017 | | Billy | 0 |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| **Camera Integration** | | | | | | | |
| | Get video feed from raspberry pi camera with TX2 | 11/10/2017 | 11/24/2017 | | | Ross | 0 |
| | Get video feed from GoPro with raspberry pi board using HDMI to CSI 2 bridge | 11/10/2017 | 11/30/2017 | | | Ross | 0 |
| | Write driver interfacing GoPro to Jetson TX2 board | 11/30/2017 | 1/10/2017 | | | Ross | 0 |
| | | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Pressure Transducers** | | | | | | | |
| | Write pressure transducer skelton code | 10/18/2017 | 10/25/2017 | 10/18/2017 | 11/1/2017 | Ross | 100 |
| | Get voltage values from pressure transducers | 10/31/2017 | 11/29/2017 | 11/1/2017 | | Ross | 100 |
| | Get third pressure transducer | 11/20/2017 | 12/10/2017 | | | Ross | 0 |
| | Modify the A/D board to add another 5 volt input | 11/30/2017 | 12/10/2017 | | | Ross | 0 |
| | Write driver for raspberry pi for pressure transducers | | | | | Ross | 5 |
| | Output correct Data to controls | 9/18/2017 | 11/24/2017 | 9/18/2017 | | Ross and Jeremy | 0 |
| **Controls** | | | | | | | |
| Develop decision trees for algorithms | | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 12 |
| | Define different algorithms for different processes | 8/30/2017 | 10/11/2017 | 10/14/2017 | | Jeremy | 50 |
| | Interpret data from IMU | 8/30/2017 | 10/25/2017 | 10/14/2017 | | Jeremy | 20 |
| | Interpret data from Cameras | 8/30/2017 | 11/30/2017 | 10/14/2017 | | Jeremy | 5 |
| | Interpret data from hydrophone array | 9/15/2017 | 12/10/2017 | | | Jeremy | 0 |
| | Interpret data from pressure transducer | 10/15/2017 | 11/15/2017 | | | Jeremy | 0 |
| Target search - Cameras | | | | | | | 5 |
| | Integrate Line following commands | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | Integrate Buoy finding commands | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | Integrate Gate finding commands | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | Integrate torpedo target finding command | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | | | | | | | 5 |

| **Tasks** | **Subtasks** | **Est Start** | **Est Finish** | **Actual Start** | **Actual Finish** | **Resource** | **Percent Complete** |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Plot sparton live filtered IMU data** | | **3/15/2017** | **11/1/2017** | **3/15/2017** | | Billy | |
| | get live data | 3/15/2017 | 3/22/2017 | 3/15/2017 | 3/22/2017 | Billy | 100 |
| | save live data | 3/15/2017 | 3/22/2017 | 3/15/2017 | 3/22/2017 | Billy | 100 |
| | plot saved data | 3/22/2017 | 3/23/2017 | 3/22/2017 | 3/23/2017 | Billy | 100 |
| | filter saved data (using and comparing different filters) | 3/23/2017 | 11/1/2017 | 3/23/2017 | 11/04/2017 | Billy | 100 |
| | feed in saved data as live data to a filtering function (python) | 3/28/2017 | 3/30/2017 | 3/28/2017 | 3/30/2017 | Billy | 100 |
| | feed in saved data as live data to a filtering function (matlab) | 8/30/2017 | 9/13/2017 | 8/30/2017 | 9/13/2017 | Billy | 100 |
| | plot live filtered accel data | 8/30/2017 | 9/20/2017 | 8/30/2017 | 9/20/2017 | Billy | 100 |
| | plot live filtered 3d accel data | 8/30/2017 | 11/1/2017 | 8/30/2017 | 11/1/2017 | Billy | 100 |
| | plot live filtered 3d v data | 8/30/2017 | 11/1/2017 | 8/30/2017 | 11/8/2017 | Billy | 100 |
| | plot live filtered 3d p data | 8/30/2017 | 11/1/2017 | 8/30/2017 | | Billy | 99 |
| | plot live 3d data (matlab) | 8/30/2017 | 11/1/2017 | 8/30/2017 | | Billy | 99 |
| | plot live data (python) | 3/15/2017 | 10/13/2017 | 3/15/2017 | | Billy | 0 |
| | Final Filter implemented into python | 10/17/2017 | 11/1/2017 | 10/17/2017 | | Billy | 0 |
| | fine tune | 10/13/2017 | 10/27/2017 | 10/11/2017 | | Billy | 95 |
| | Pressure Transducer | 11/8/2017 | 12/8/2017 | | | Billy | |
| **Video Processing** | | | | | | | |
| | Select new vision processor unit | 9/6/2017 | 9/27/2017 | 9/6/2017 | 9/27/2017 | Ross | 100 |
| | learn image filtering techniques | 9/20/2017 | 10/20/2017 | 9/20/2017 | | Oren | 25 |
| | Learn Cuda acceleration for processor stress relief | 9/24/2017 | 10/24/2017 | 9/24/2017 | | Ross | 5 |
| | Define Line | 10/2/2017 | 12/10/2017 | 10/2/2017 | | Oren/Billy | 0 |
| | Define Buoys | 10/2/2017 | 12/10/2017 | 10/2/2017 | | Oren | 75 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Create object | 10/2/2017 | 12/10/2017 | 10/2/2017 | | Oren | 0 |
| | Update Object | 11/15/2017 | 12/10/2017 | | | Oren | 60 |
| | Pass Object to Controls | 11/15/2017 | 12/10/2017 | | | Oren/Billy | 0 |
| | Objection Detection in MATLAB | 10/2/2017 | 11/1/2017 | 10/2/2017 | 10/27/2017 | Jake | 100 |
| | Parallelization of Object Detection code in MATLAB | 11/1/2017 | 11/7/2017 | 10/27 | 11/4/2017 | Jake | 100 |
| | Port Object Detector to Python | 11/7/2017 | 12/10/2017 | 11/4/2017 | | Jake | 15 |
| | Define size | 11/20/2017 | | 11/8/2017 | | Billy | 0 |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| **Camera Integration** | | | | | | | |
| | Get video feed from raspberry pi camera with TX2 | 11/10/2017 | 11/24/2017 | | | Ross | 0 |
| | Get video feed from GoPro with raspberry pi board using HDMI to CSI 2 bridge | 11/10/2017 | 11/30/2017 | | | Ross | 0 |
| | Write driver interfacing GoPro to Jetson TX2 board | 11/30/2017 | 1/10/2017 | | | Ross | 0 |
| | | | | | | | |
| **Pressure Transducers** | | | | | | | |
| | Write pressure transducer skelton code | 10/18/2017 | 10/25/2017 | 10/18/2017 | 11/1/2017 | Ross | 100 |
| | Get voltage values from pressure transducers | 10/31/2017 | 11/29/2017 | 11/1/2017 | | Ross | 100 |
| | Get third pressure transducer | 11/20/2017 | 12/10/2017 | | | Ross | 0 |
| | Modify the A/D board to add another 5 volt input | 11/30/2017 | 12/10/2017 | | | Ross | 0 |
| | Write driver for raspberry pi for pressure transducers | | | | | Ross | 5 |
| | Output correct Data to controls | 9/18/2017 | 11/24/2017 | 9/18/2017 | | Ross and Jeremy | 0 |
| **Controls** | | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Develop decision trees for algorithms | | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 12 |
| | Define different algorithms for different processes | 8/30/2017 | 10/11/2017 | 10/14/2017 | | Jeremy | 50 |
| | Interpret data from IMU | 8/30/2017 | 10/25/2017 | 10/14/2017 | | Jeremy | 20 |
| | Interpret data from Cameras | 8/30/2017 | 11/30/2017 | 10/14/2017 | | Jeremy | 5 |
| | Interpret data from hydrophone array | 9/15/2017 | 12/10/2017 | | | Jeremy | 0 |
| | Interpret data from pressure transducer | 10/15/2017 | 11/15/2017 | | | Jeremy | 0 |
| Target search - Cameras | | | | | | | 5 |
| | Integrate Line following commands | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | Integrate Buoy finding commands | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | Integrate Gate finding commands | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | Integrate torpedo target finding command | 8/31/2017 | 12/10/2017 | 10/14/2017 | | Jeremy | 5 |
| | | | | | | | 5 |