Northrop Grumman Cal Poly (NGCP) Collaboration Project:

Underwater Remotely Operated Vehicle (ROV)
Computer Vision and Control Systems

Jared Spadaro: *jaspadar@calpoly.edu*
Jeremy Passehl: *jpassehl@calpoly.edu*

Table of Contents

Introduction

Project Overview

Northrop Grumman sponsors Cal Poly Pomona and Cal Poly San Luis Obispo to work together towards accomplishing collaborative, autonomous vehicle missions. The present focus of the ROV team is to integrate the electrical hardware and software components with the mechanical frame, then test the complete system. This will involve implementing the control system with the final frame, connecting components to power, and integrating computer vision capabilities. Accomplishing this will create a strong foundation for future work on autonomous capabilities.

The vehicle should be able to avoid collisions and identify simple objects. The controls, sensor data, and computer vision results will be integrated into one usable system on the ground station laptop. All data will be sent to the vehicle and received at the ground station laptop via an ethernet cable. Sensors are connected to the vehicle for keeping status of its position in the water. Cameras will provide object detection and distance sensing capabilities. Power and electronics will also be housed in a waterproof environment within the central containment unit, while maintaining necessary external connections to the thrusters and external camera containment unit. The system will use data captured from an IMU to analyze the ROV's surroundings and perform navigation and object identification tasks.

Clients and Community Partners

Northrop Grumman and the NGCP San Luis Obispo Team are the clients for this project. The Northrop Grumman Collaboration Project (NGCP) is a joint technology demonstration between Cal Poly San Luis Obispo and Cal Poly Pomona. As an industry leader in autonomous systems, Northrop Grumman sponsors a hands-on interdisciplinary experience through collaborative designing of unmanned, autonomous vehicles.

The NGCP team has been at Cal Poly for over five years and is currently working on unmanned aerial vehicles and an underwater remote operated vehicle. The ROV team is in its second year of operation. The final version of the ROV will be delivered to the client and Cal Poly for future development.

Stakeholders

The ultimate goal of the club is to complete joint missions between schools and various land and air vehicles. Although we will be working directly with the ROV team, the NGCP team as a whole will benefit from the outcome of our project. They will be able to build upon and utilize the implementation of the new control system for the other existing vehicles and those in the future.

Within the greater community, other stakeholders include marine biologists, ecologists, oceanographers, and divers. Potential researchers could use the ROV to explore parts of oceans, seas, lakes, and other bodies of water that a diver could not be sent into. Commercial and recreational divers could use the ROV to map future dives, making note of paths and hazardous areas. The current market provides these clients with few options, as the technology is expensive and geared toward commercial and private use. The ROV could provide these stakeholders with a practical and more affordable option.

Framed Insights and Opportunities

Although Northrop Grumman is the primary client, we will be working directly with the Cal Poly ROV collaboration group. We have committed to weekly meetings with the entire ROV team. An additional work day outside of class has been established as the team's work day so that we can collaborate with the mechanical and electrical teams if necessary. We have also established several different contact groups using applications such as Slack and Trello to ensure we can direct information to the the correct people quickly and efficiently.

We were able to establish a course for the project after several meetings with the collaboration group. We worked with the NGCP team to establish a foundation of what needed

to be done. The primary requirement for us is to get all components of the system working together in a final product. Detailed documentation will be important so that we can pass the existing work and our new contributions to future groups. Although we have been given an open ended direction for our project, the ROV team would like us to implement something that can continued to be built upon for future iterations of the ROV and team missions.

Goals and Objectives

Multiple goals have been established in order to work toward the expectations of our client. Our first goal is to combine the work of the previous two computer engineering capstone group's software components. This will lead to a unified interface for the controls, sensor data, and computer vision components. The other goals are generally based on having a complete system capable of navigation and object identification.

This system should be developed as a platform capable of having the controls and computer vision results communicating to one another such that autonomous missions can be accomplished if position data can be acquired as a result of future development. A list of goals and objectives was compiled:

- Goal 1: Combine the separate software components into a single interface on the ground control station
    a. Design a system that is convenient for the user to control and view results
    b. The user should be able to see all necessary programs running on the screen at all times without opening and closing windows
- Goal 2: Combine the software and hardware components
    a. Design the system components to be modular
    b. Meet the system's waterproofing requirements
    c. Test the reliability of electrical connections
- Goal 3: Create a custom thruster configuration in the control software
    a. Understand the default thruster control configurations relative to pitch, roll, and yaw.

b. Compile and upload firmware onto Pixhawk controller

c. Label thruster wires for ease of access

d. Understand relationship of vehicle weight and thrust

- Goal 4: Test the complete system, record footage and successfully demo complete system
    a. Access a pool suitable for testing the ROV
    b. Use a reliable screen capture software
    c. Run all components of the system simultaneously
    d. Develop on schedule to present at the final demonstration
- Goal 5: Continue to build the foundation for future autonomy
    a. Communicate with leads to ensure goals are on track
    b. Leave behind well-documented work so that next year's team benefits

Outcomes and Deliverables

As a whole, the ROV will have fully functional movement controls, a stable frame, and computer vision capabilities. From the ground station laptop, the user will be able to view the live camera stream, control the ROV with a joystick controller, and view the sensor results.

The ground station user will have an option to switch between computer vision modes using the ground station keyboard. They can enable the streaming mode, object detection mode, or depth perception mode. A notification will be displayed in real time at the ground station video feed when the object has been found. Once the object is identified, another computer vision algorithm can be used to calculate the distance to the object using two cameras. This will produce a depth map at the ground station allowing the user to gauge the distance to what is in the center of both cameras. The game controller will control thrusters, allowing the user to thrust, roll, pitch, yaw, and depth of the ROV. Physical design, code implementation, tests, and use cases will be well-documented with background research to allow efficient continuation by future members.

Background

The primary source of existing information is the work done by the previous ROV teams. First development began in Fall 2015, and two capstone teams completed work on the ROV during the year until the Winter 2017 quarter. The teams have provided well documented work, as well as contact information to speak directly to them about past work.

In the first year of ROV development, the ROV team was able to design, build, and test a basic ROV. The mechanical frame and basic movement controls were largely based off of the OpenROV. OpenROV is an open source ROV community that provides forums and git repositories for the OpenROV source code.

The current year's version the ROV team has been redesigned, improved, and remanufactured from scratch. The purpose of this is to expand the frame to allow for more functionality and features to be added of the ROV, including a larger central containment unit.

This generation of development led to much research about the addition of sonar capabilities. Sonar was not integrated in the current generation of the ROV, but additional computer vision capabilities and a servo-controlled stereo-camera setup were added.

The current portion of the project is also very heavily interdisciplinary, so working with the mechanical engineers and electrical engineers will be critical to the success of compiling all parts before the project demo, which is scheduled for May 20, 2017.

Engineering Specifications

The final specifications evolved through communication with mechanical, electrical, and computer engineers. All of our sub-team's requirements were derived from the overall team's requirements. In **bold** are the requirements that apply specifically to our responsibilities:

Engineering Requirements

1. *ROV*
   - **1.1. Operate at depths of 100 feet**
   - **1.2. 30 minute run time**
   - **1.3. Self-stabilizing**
   - 1.4. Survivability- protection for fragile components (i.e. buoyancy tank and thrusters)
   - 1.5. Maximum weight of 35 pounds
   - 1.6. +1 lb buoyancy
2. *Ground Control Station*
   - **2.1. Connect to ROV while ROV at 100 feet deep**
   - **2.2. Monitor pitch, yaw, and roll of ROV**
   - **2.3. Monitor and track ROV position while vehicle in motion**
   - **2.4. System shall process computer vision algorithms at a speed capable of displaying at 15 frames / second.**
   - **2.5. Perform a search for a predetermined object (a colored ball)**
     - **i. Detect the object if it is within 15 feet of the ROV**
     - **ii. Ability to calculate the distance to the object if it is within 15 feet of ROV**

Engineering Specifications

Spec. #	Parameter Description	Requirement or Target	Tolerance	Risk	Compliance
1	Operating time	30 minutes	Min	L	A,T
2	Maximum ROV operating depth	100 ft	Max	M	A, S
3	Waterproofing	IP68	Min	H	A,T, S, I
4	Strafe, up/down, and forward/back speed	2 knots	Max	L	T, I
5	Pitch, roll, and yaw angular velocity	20°/sec	+/- 10°/sec	L	A, T
6	Operational range from base station	100 ft	Min	L	A, T
7	Visibility range of the camera	15 ft	Min	H	A, T, S
8	Video display resolution at ground station	640 x 480 pixels	Min	M	S, I
9	Bandwidth of total data transmission to ground station	20 Mb/s	Max	H	T, A, I
10	Size of yellow ball detectable at 15ft	12 inch diameter	Min	L	T, I
11	Distance capability of object being detected	15 ft	Max	L	T, I

Table 1. Engineering Specifications

Compliance Key:

- Analysis (A)
- Test (T)
- Similarity to Existing Designs (S)
- Inspection (I)

Risk Key:

- Low (L)
- Medium (M)
- High (H)

Use Cases

Use Case ID	Use Case Name
1	Manually Navigate the ROV
2	Search for a Yellow-Colored Ball
3	Determine the Distance to an Identified Object

Table 2. Summary of Use Cases

Use Case ID	1
Use Case Name	Manually Navigate the ROV
Created By	Jared Spadaro
Date Created	November 3rd 2016
Last Updated By	Jeremy Passehl
Date Last Updated	5/28/2017
Actors	ROV Pilot
Description	Pilot gives a navigation command to ROV to set a direction of travel
Preconditions	Pilot is interacting with ROV Control Station. ROV is powered and connected with the control station via ethernet.
Postconditions	ROV stops moving when command is over
Normal Flow	1. Pilot selects and holds down a navigation option on the control panel: Right, Left, Forward, Back, Roll Left, Roll Right, Ascend (Up), Dive (Down) 2. ROV move in that direction 3. Pilot releases control 4. ROV stops
Alternative Flow	1. Pilot selects and holds down a navigation option on the control panel: Right, Left, Forward, Back, Roll Left, Roll Right, Ascend (Up), Dive (Down) 2. ROV does not respond

Exceptions	System fails, ROV shuts down.

Use Case ID	2
Use Case Name	Search for a yellow-colored ball
Created By	Kelly Sio
Last Updated By	Jeremy Passehl
Date Created	11/9/2016
Date Last Updated	5/28/2017
Actors	User at the ground station
Description	The user may turn on object recognition to determine whether the system recognizes the object in the frame of the front facing cameras.
Preconditions	The ROV is underwater. The ROV is tethered to the ground station. The object is predetermined (in this case a yellow ball).The yellow ball is within 15 feet radius of from the ROV.
Postconditions	The ROV notifies the user the yellow ball is found.
Normal Flow	1. User is manually operating the ROV 2. User turns on object recognition mode. 3. User navigates through water using controls at the ground station 4. Ground computer notifies the user the object is found when the algorithm detects a yellow ball in the frame.
Alternate flows	1. User turns on object recognition mode. 2. Algorithm does not detect yellow ball, no notification is displayed 3. User continues searching for the ROV. 4. Repeat step 3 until the algorithm detects a red ball in frame and notify the user.
Exceptions	System fails, ROV shuts down.

Assumptions	The PID controller is well tuned. The water conditions are sufficient to controlling the ROV. Good lighting for the camera. The ground station controller has enough processing power.

Use Case ID	3
Use Case Name	Determine the Distance to an Identified Object
Created By	Jeremy Passehl
Last Updated By	Jeremy Passehl
Date Created	11/9/2016
Date Last Updated	5/28/2017
Actors	Pilot at the Ground Station
Description	The user at the ground station may wish determine the distance of an object on the screen. The benefits: 1) Helps the ROV avoid collisions, 2) When the target object is detected, we will be able to determine how far away it is. If a target object is not selected, but an object is close to the ROV, a warning message may be displayed so that a collision is avoided. 3) May help for future object retrieval
Preconditions	The ROV is tethered to the ground station with an ethernet cord. The ROV is in an environment that supports sufficient range of vision with a low enough signal to noise ratio.
Postconditions	The distance of the object detected is displayed on the screen for the ground station user.
Normal Flow	1. The user navigates the ROV 2. The ROV approaches an object that is visible from the camera image 3. The ground station controller clicks in the viewing frame over the object 4. The “Distance to Object” field displays the current distance to the object.

Alternate flows	1. The object is detected and highlighted in the camera ground station display (See Use Case #1) 2. The “Distance to Object” field displays the current distance to the object. 1. User controls the ROV The ROV can’t detect any points for distance calculations 2. The ground station controller clicks in the viewing frame over the object 3. The “Distance to Object” field displays the message “No Object Found” 1. The user is controlling the ROV 2. The ROV is approaching an object that is getting very close to the ROV 3. The ground station controller clicks in the viewing frame over the object 4. The “Distance to Object” field displays the message “Warning! Close Object”
Exceptions	System fails, ROV shuts down.
Assumptions	The PID controller is well tuned. The water conditions are sufficient to controlling the ROV. Good lighting for the camera. The ground station controller has enough processing power.

Final Design

I. Previous Prototype Development

Separate prototypes of each software component have been developed this year up to this point. The purpose of this section is to provide context for our work and identify areas of the components that need improvement. The software components fall primarily into one of two categories:

1. Computer Vision

The computer vision component evolved from an isolated system which recorded underwater footage or still captures into a system capable of running in real-time. The state of the code at the beginning of our work was a system capable of running the three computer vision algorithms in real-time: 1) unprocessed video streaming, 2) object detection, 3) depth perception.

The real-time system uses two cameras to capture and send synchronous images on the Raspberry Pi by utilizing multithreading, as seen in *Figure 1*. The synchronized images are continuously sent to the ground control station using UDP and ethernet, where they are then processed in one of the three algorithms.

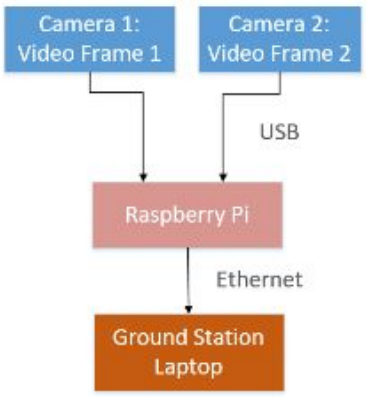

Figure 1: Data Flow From Cameras to Ground Station Laptop

One area in need of improvement is the depth perception algorithm. The initial prototype was designed with the cameras spaced 5.25 inches apart. This led to the depth perception

algorithm not being able to detect the distance to the ball until it was over 3 feet away. The chart below shows the measured relationship between disparity (directly correlated to depth) and the distance to the ball when the cameras were spaced 5.25" apart (*Figure 2*).

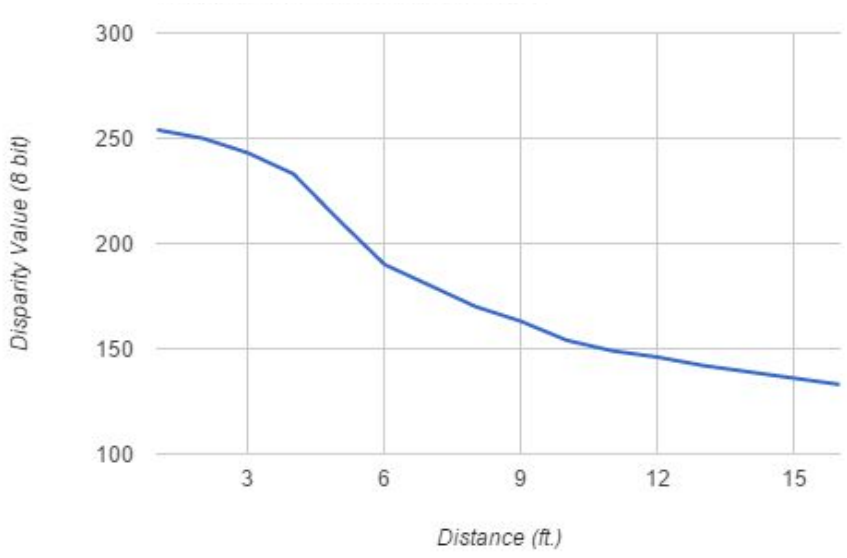

Figure 2: Depth Perception Distance (ft.) vs. Disparity (8 bit) Relationship

Notice that when the object is less than three feet away, the disparity doesn't change very much as the distance varies. One way to counteract this is to move the cameras closer together. This will reduce the disparity accuracy at over 10 feet, but it is a trade-off based on a desire to have more depth accuracy at closer ranges. The reason for this change is potential collision avoidance and object retrieval capabilities in the future.

Another area requiring improvement is a consistent segmentation fault that occurs approximately every 2-5 minutes on the ground control station image processing program. A little investigation shows that the problem is related to an OpenCV assertion failing regarding matrix dimensions. The problem may be caused by the Raspberry Pi failing to send a complete image every so often. This will need to be fixed in order to improve the reliability of the video stream.

A final area that needs work is tuning the color of the object detection algorithm since it was last tuned to detect the ball in an office-like environment. The cameras are also designed to be controlled by a servo, which has not been part of any of the prototypes developed up to this point.

2. Controls

The software in this final architecture incorporates the open source underwater vehicle navigation program *ArduSub*, which provides basic drivers and controls for underwater vehicles. This system supports various hardware and sensors and provides common utilities like depth hold (using a pressure sensor) and stabilization (through the use of Kalman filters and the IMU). This software will run on a Pixhawk, a popular hobbyist microcontroller complete with multiple sensors and easy interfacing. To control the ROV, we use the QGroundControl application. This application is the default ground control station for ArduSub and provides a way to interface with and control it. Some notable features are live sensor readings, controller input for movement control, and video streaming.

The method of communication between ArduSub and QGroundControl relies on a communication protocol called MAVLink. This protocol is designed for use with robotics and provides an API to make it easier to use. Using this protocol, the Pixhawk can send information (e.g. sensor data) to QGroundControl, which can then act based on that data. QGroundControl can also send its own commands to the Pixhawk (e.g. movement controls).

In order to test the controls software to ensure that it would be effective, the controls team retrofitted an OpenROV. OpenROV is an open source underwater vehicle designed by a group of ROV enthusiasts. This was done by replacing all of the electronics except a board which provided power distribution and ethernet to twisted pair conversion. After integrating the Raspberry Pi, containing code to supplement the flight stack, and the Pixhawk, containing the controls firmware, the OpenROV was successfully tested in a pool and the hardware/software setup was confirmed to work for our purposes.

II. Electrical Hardware Design

All hardware component decisions (cameras, Raspberry Pi, Pixhawk) were made by the previous capstone teams after weighing the specifications in their respective decision matrices. The computing hardware in the ROV consists of a Pixhawk microcontroller and a Raspberry Pi 3 microcomputer. The Pixhawk is responsible for reading sensors and controlling thrusters. The

Raspberry Pi is used to forward video data from a camera as well as interface with the Pixhawk in order to relay messages sent between the Pixhawk and the ground control station.

One of the main issues faced when designing the electrical system of the ROV was power distribution. The electronic speed controllers (ESCs), which take power and ground from the batteries and pulse width modulation (PWM) signals from the Pixhawk and convert them for use by the thrusters, take a maximum current of 30 amps each, according to the Blue Robotics[8] (the designer and seller of both our ESCs and thrusters). However, when testing their current draw, an actual maximum of 5 amps each was measured. There are six ESCs, so this is still a considerable amount of power. Luckily, the size of the containment unit on the ROV allowed us to use three high capacity multirotor batteries, each with a capacity of 10000 mAh and nominal voltages of 14.8 V. Each battery is connected to two ESCs, with a 40 amp fuse protecting the power line for the pair. Note that for the purpose of simplifying the diagram, this is not shown in *Figure 3* below. Instead, a representative fuse is shown, rather than three of them connected to each battery.

To power the control system, a fourth battery is connected to a voltage regulator which cuts down the 14.8 nominal voltage to 5 V (see *Figure 3*). This 5V signal is used to power the Raspberry Pi, which in turn powers the two Logitech C920 cameras (connected via USB), the Pixhawk (connected via USB), and the camera servo used to tilt the cameras (connected via GPIO).

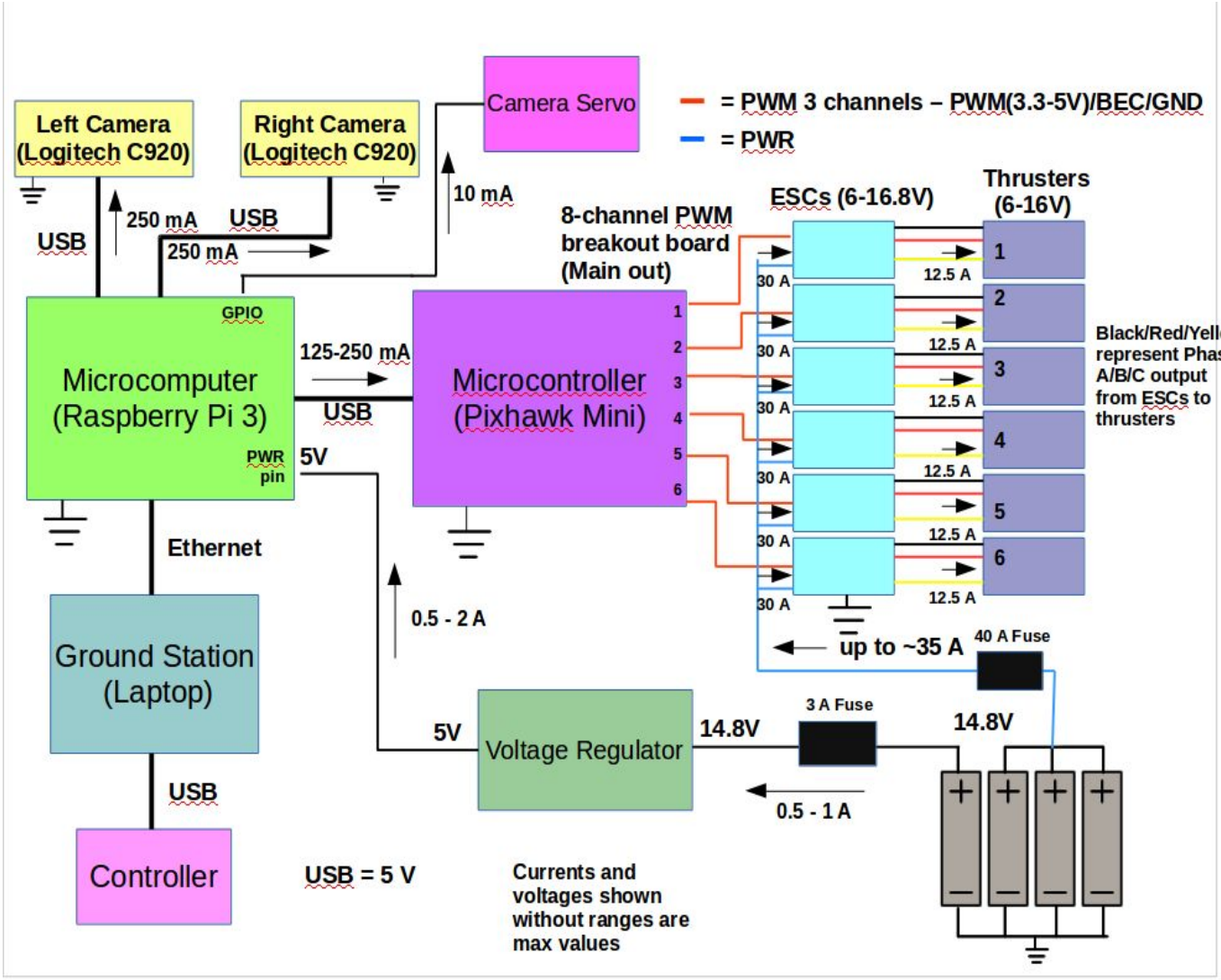

Figure 3: Electrical Hardware Schematic

III. Integrating Electronic Design with Mechanical Design

Although the mechanical engineering team designed the frame of the vehicle, the desired placement of the thrusters was a joint decision. The placement that gives optimal control from a software development standpoint and was based off of the design of the 2015-16 ROV. This is a six thruster design *(Figure 4*) which maximizes maneuverability without convoluting the software implementation. The pairs of thrusters are mounted to allow movement in all six degrees of freedom. The placement of the buoyancy tank is designed to auto-stabilize the ROV when the water is free of currents, making it ideal for pool testing.

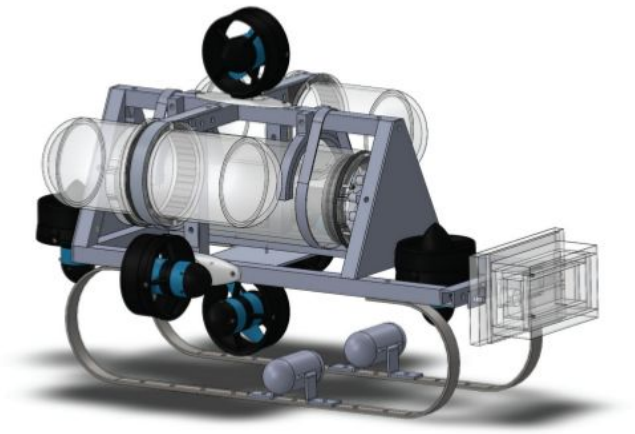

Figure 4: Example of a 6 Thruster Arrangement (2015-16 Model)

Another area requiring joint collaboration with the mechanical engineering team is in making sure all electronic components are in waterproof containment units. The mechanical engineers designed the major containment unit in a cylindrical fashion with two endcaps that are sealed with O-rings. The rear endcap was chosen as the place where all wires will enter the containment unit. The connections that require waterproofing can all be found in *Figure 5*.

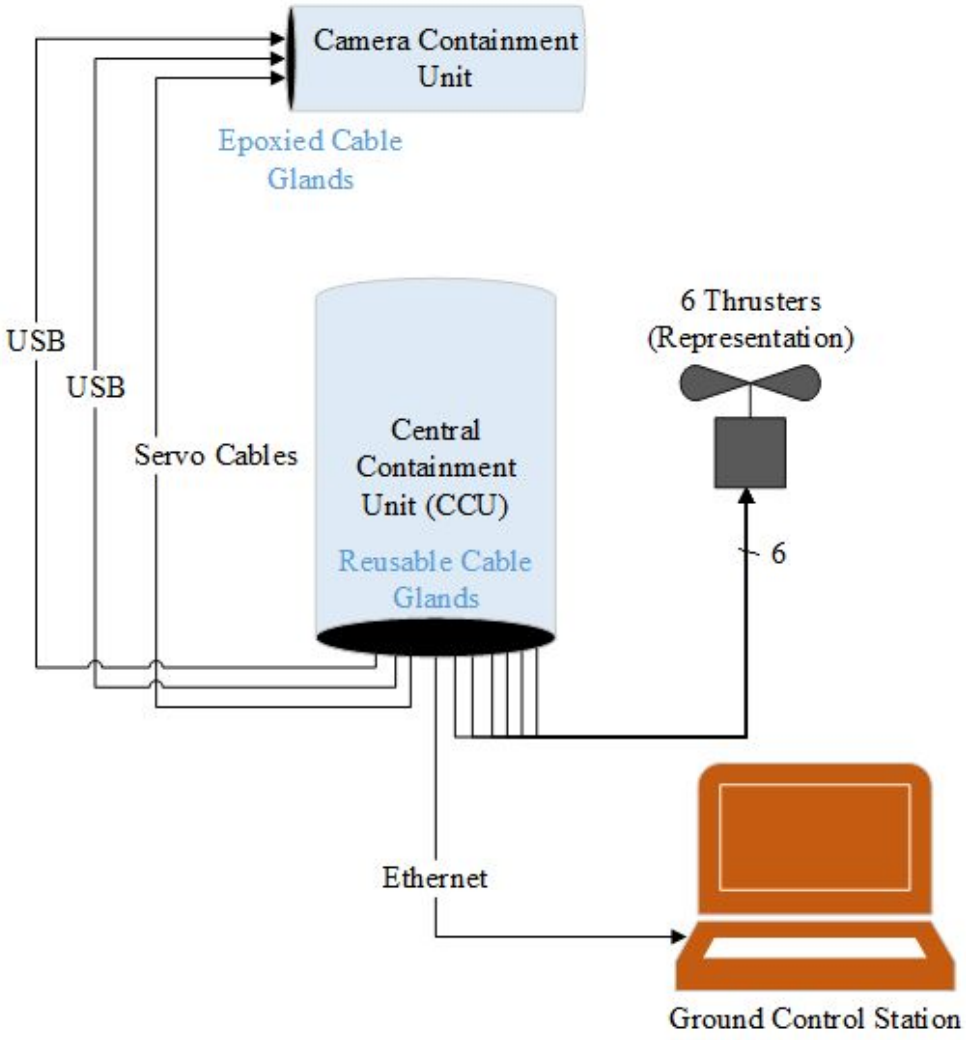

Figure 5: Required Waterproof Connections

The connection points at the Central Containment Unit endcap will all pass through adjustable cable glands which will allow tightening and loosening with the simple use of a wrench. The connection points at the Camera Containment Unit, however, require epoxy to make the connection waterproof, and are thus not "reusable" cable glands. After the initial sealing of the glands, it is not possible to disconnect the endcaps of the Camera Containment Unit and the Central Containment Unit due to the large USB ends on either side of the endcaps.

Care was taken to make the design as modular as possible. As seen in *Figure 6*, even though the endcaps will be semi-permanently (cutting and resoldering wires will be required) attached to each other from now on, the Raspberry Pi can still be easily removed from the central

containment unit and connected to the cameras, servo, and or Pixhawk at any time. All components in the camera containment unit have short male leads, while all units have female connections. The thrusters are not attached in this manner and are thus semi-permanently attached to the central containment unit endcap.

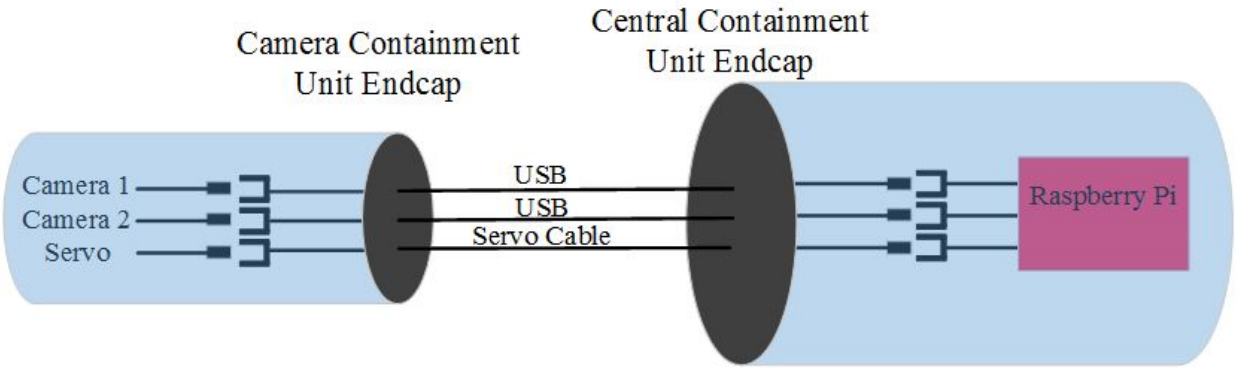

Figure 6: Modular, Waterproof Connection Design

The horizontal distance between the cameras is adjustable via manual placement. This is accomplished by using a velcro bottom as well as slots for zip ties to secure the cameras firmly in place. The mechanical engineers designed the camera viewing angle to be adjustable via servo during underwater operation (*Figure 7*). The angle of the cameras is adjustable, from looking slightly upward, to looking directly below the ROV in the range of approximately 120 degrees.

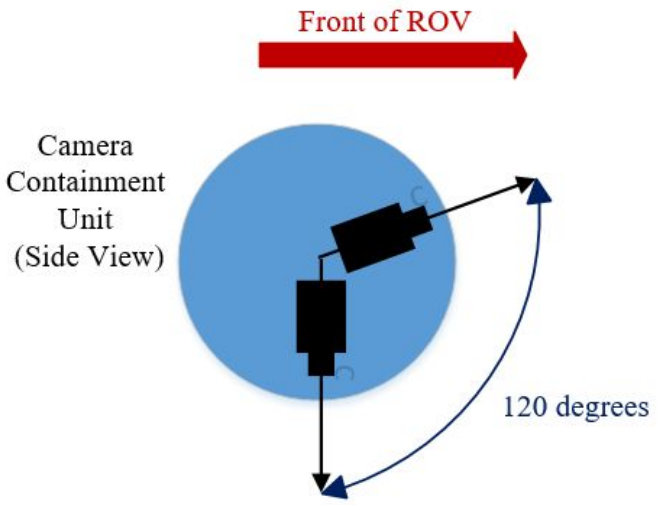

Figure 7: Camera Servo Range of Motion

IV. Software Component Integration

There are three major processes that should be running on the ground station laptop during operation. These three process are:

1. QGroundControl
2. Computer Vision Server Program
3. SSH to Raspberry Pi

QGroundControl / Configuring Vehicle in ArduSub

QGroundControl is a ground station program that provides a way to conveniently interface with the pixhawk. A controller is connected to the laptop via USB that accepts and sends commands to the pixhawk (see *Figure 8*). There are two stages of configuration when it comes to controlling the ROV.

The first is in the firmware running on the pixhawk. In the ArduSub code (which, when built, becomes the firmware for the pixhawk), there is a parameter called *frame_class* which determines the number, location, and function of the thrusters on your vehicle. Among the options for this parameters is SUB_FRAME_CUSTOM, which allows the developer to create his own custom thruster set up. This is done by providing decimal values for each thruster to

determine what it provides for the motion of the vehicle. Each thruster has six of these decimal values assigned to it: one each for roll, pitch, yaw, throttle, forward, and lateral degrees of motion, with 0 being minimum and 1 being maximum. For example, thruster 3 on our vehicle has the following assignments (in the order stated previously, with the final argument representing the motor number):

```
add_motor_raw_6dof(AP_MOTORS_MOT_3, 0, 1.0, 0, 1.0, 0, 0, 3);
```

So, thruster 3 has a 1.0 pitch factor (pitch up) and a 1.0 throttle factor (throttle up).

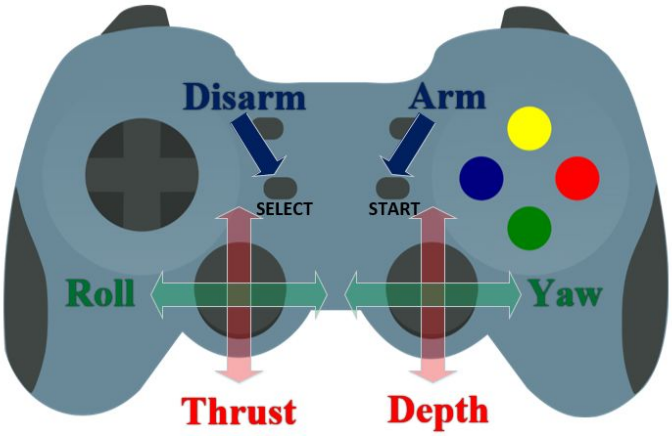

Figure 8: Controller Mapping

The second stage of configuration is in QGroundControl itself, where you have the ability to customize what each button / joystick on the controller does. In order to properly control the ROV, it is necessary to sync up the firmware and controller configurations. Otherwise it can get very confusing to know what to expect when moving the ROV around in the water.

Computer Vision Server

The Raspberry Pi will constantly be sending data from the two camera streams so that the server program on the ground control station can retrieve them whenever it is ready. There are three modes of operation of the server program the ground station laptop, as shown in *Figure 9*.

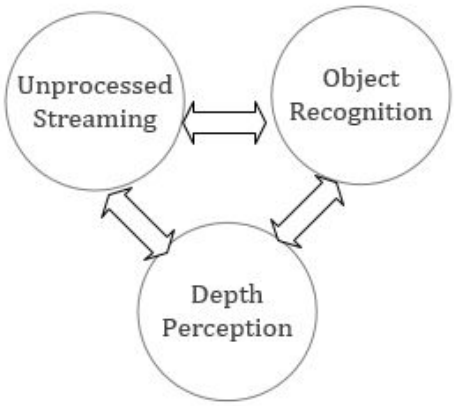

Figure 9: Computer Vision Mode Navigation

Use these keyboard commands (*Table 3*) to switch between computer vision modes.

Key Pressed	Motion Command
s	Display unprocessed streaming results
o	Display object detection results
d	Display depth perception results
e	Exit

Table 3: Computer Vision Keyboard Commands

Figure 10 below shows an example of the the interaction between the Raspberry Pi and the control station. User Datagram Protocol (UDP) is used to send synchronized images to the laptop. The laptop can then use these images to run the computer vision algorithms. The results are displayed depending on the user-selected mode.

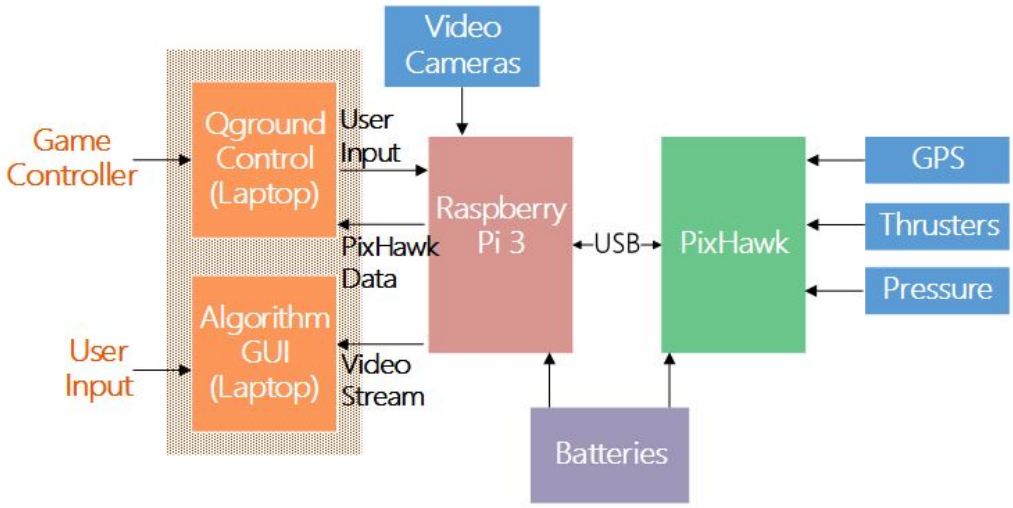

Figure 10: Major Component Block Diagram

Modifications were made to the camera streaming algorithms to greatly reduce the occurrence of segmentation faults. This segmentation fault requires the user to restart the streaming program. There is a function that checks for errors in the incoming queues capable of holding up 5 frames before throwing one away as new ones come in. First, an exception handler was added to the function and additional checks to make sure the queues are not empty before proceeding were added (line 120 of Server.cpp)[6].

It was mentioned earlier that previous prototype experience lead to a desire for better depth resolution at shorter distances. The cameras were placed closer together once the final mechanical supports for the camera finished manufacturing. The distance between the centers of the lenses was measured to be 3.375 inches, as compared to the previous 5.25 inches. Once the cameras were secured into place using velcro and zip ties, the calibration of the cameras was performed. The calibration .xml file was then saved and placed in the same directory as the Server.cpp file. The code was edited to import this new .xml file, and then recompiled (line 242 of Server.cpp)[6].

SSH to Raspberry Pi

An SSH connection is constantly maintained from the ground station to the Raspberry Pi. This is necessary for a couple of reasons. The video streaming process that streams data from the Pi to the laptop must be started through SSH. Occasionally the Pi or the Pixhawk may need to be reset remotely, and having this connection is the most convenient way to do that.

This connection is also the way to control the camera servo. The signals sent to the servo can be controlled by one GPIO pin #18 on the Raspberry Pi which is capable of sending hardware-generated PWM signals. A program called WiringPi was installed on the Raspberry Pi, allowing the user to access this GPIO pin. The following commands initialize the pin for use, and were added to a bash script on the Pi:

```
gpio -g mode 18 pwm
gpio pwm-ms
gpio pwmc 192
gpio pwmr 2000
```

This command sets the duty cycle of the pwm, allowing the user to adjust the camera angle (where 150 is the duration in milliseconds):

```
gpio -g pwm 18 150
```

System Integration and Testing

Failure Modes and Effects Analysis

The design was analyzed for potential areas of failure. The likelihood of the failure, the severity of the failure, and the ability to detect the failure were combined into one score, referred to as the Risk Priority Number (RPN).

Most of the test cases were designed to test the thrusters, sensors, and waterproofing. Some of the most critical failures listed in the FMEA (*Figure 11*) include a leak in the containment unit, batteries dying during operation of the ROV, and the ethernet tether breaking. All three of these could potentially result in permanent damage to the electronics in the ROV, so they are rated with a high severity.

A set of ROV safety standards concerning equipment, operations, and personnel has been created by the International Marine Contractors Association (IMCA). The publication is known as "IMCA R 004 Rev. 3" *Code of practice for the safe and efficient operation of remotely operated vehicles (ROVs)* [7]. Standard ROV user manuals should be studied before use and testing.

Process Function	Potential Failure Mode	Potential Effect(s) of Failure	Potential Cause(s)/ Mechanism(s) of Failure	RPN	Recommended Action(s)
Maintain Ethernet Connection	Connection is lost	Users cannot see where the ROV is navigating or control the ROV. The ROV will begin floating to the surface	1. Ethernet cord is broken/loose 2.Program crashes	48	1. Have SSH capabilities 2. Have a way to reboot the power from outside of the containment unit
Maintain Safe Operating Temperatures	Temperatures are too high due to high water temperatures	Hardware is either destroyed or goes into automatic shutdown to avoid harm	Ineffective heat dissipation / water temperatures too high	270	1. Set up automated script to warn of high temperatures. Add a temperature sensor
Rpi operating system running the necessary programs at all times	Program crashes / Rpi freezes or needs reboot	Temporary loss of communication	Frozen program / unexpected exception thrown / OS failure	120	1. Have SSH capabilities 2. Have a script that starts the programs on boot 3. Implement error handling into the code
Maintain power supply of microcontroller	Batteries get too low	Entire electronic system failure	Battery voltage drops too low	486	1. Integrate battery voltage monitor along with temperature sensor.
Maintain connection with Pixhawk	Connection is lost	Controls and sensor information are lost	Physical wires disconnect / Pixhawk system fails	128	1. Have a program that monitors the status of the connection with the Pixhawk
Maintain an optimally operating ground station laptop	Ground station laptop crashes and is unrecoverable	Ground station can't communicate with ROV	1. Ground station laptop battery dies 2. Laptop crashes unrecoverably	18	1. Have an alternate laptop capable of running all programs necessary 2. Have an extra fully charged battery
Maintain safe operating voltages for components	Voltage supply is too high	Possible damage to microcontroller and	1. Microcontroller is fried and hardware components (thrusters) are damaged.	480	1. Find a way to monitor the supply voltage to components

Figure 11: Failure Modes and Effects Analysis (FMEA)

Test Cases and Results

Once all risks were determined, test cases were designed to fulfill the engineering specifications. A list of all test cases can be found in *Figure 12*.

Test #	Priority	Test Name	Specification to Test	Method
1	High	Drive Thrusters	Verify that all six thrusters can receive power and that their speeds can be adjusted.	Test
2	Medium	Control Response	Verify the ROV responds underwater to each direction the controls give it	Test
3	High	Containment Unit Waterproof Test	Verify that the central containment unit (CCU) and camera containment unit do not leak	Test
4	High	Object Detection	The system shall be able to detect the object if it is within 15 feet from the front of the camera	Test
5	Medium	Depth Perception Range	The system shall be able to detect the distance to the object if it is within 15 feet from the front of the camera	Test

Figure 12: List of Test Cases

All test cases that have required the ROV be in water have been run in a pool environment.

Test Case: Drive thrusters
Objective: Verify all six thrusters receive power
Verify each thrusters' speed can be adjusted.
Materials: Raspberry Pi 3, Pixhawk, 6 LiPo batteries, 6 ESCs, 6 thrusters, ground station, a game controller, and a body of water large enough for an operating thruster.
Safety: Monitor battery voltage to ensure it does not drop below 14 V during operation. Keep hands away from thrusters while they are powered.
Procedure:

1. Check battery voltage to make sure it is charged to 14.8 V
2. Connect ethernet cable to ground station
3. Connect batteries to ESCs via bullet connectors
4. Wait for ESCs, Raspberry Pi, and Pixhawk to power up
5. Connect game controller to ground station

6. Launch QGroundControl
7. Arm motors in QGroundControl
8. Move both analog sticks slowly in a circle
9. (Verify each thruster begins to move)
10. Return sticks to center
11. (Verify thrusters come to stop)
12. Slowly push sticks out towards edge
13. (Verify each thruster has changed speed)

Results: One of the six thrusters appeared to get stuck. Whenever the signal was sent to the thruster in both directions, the thruster jittered back and forth. Disconnecting and manually moving the thruster in an attempt to get it "unstuck" did not fix the problem. The problem was isolated to be a thruster problem whenever an extra thruster was plugged into the same ESC socket and tested. The thruster was swapped out for a working one.

This problem would appear 2 more times during complete system testing at the pool. The thruster that got stuck the additional two times was in a different location on the ROV, meaning it probably has nothing to do with the wiring. All connections were checked for potential shorts, still, and additional electrical tape was added for safety. After the thruster was initially stuck (jittering), the next day the thruster worked without any issues.

Test Case: ROV Controls Response
Objective: Verify the ROV responds underwater to each direction the controls give it
Materials: Fully built ROV
Safety: Check that containment unit is sealed tightly to prevent a water leak. Ensure battery is charged to 14.8V. Check that wires not in containment unit are not damaged.
Procedure:

1. Connect ethernet tether to ground Station
2. Check that all other pertinent connections (e.g. any microcontroller wiring, power wiring) are connected securely
3. Launch QGroundControl
4. Move controller analog sticks to check thrusters change in speed
5. Place the ROV in the water
6. Push the analog sticks forward one direction at a time.
7. Verify the ROV performs the correct motion according to the command given to it (roll, thrust, pitch, yaw, depth)
8. Release the analog stick
9. Verify the ROV stops

Results: During the first pool test, the thrusters were unable to overcome a net buoyancy miscalculation, which kept the ROV at the top of the pool. The thrusters did not have enough thrust to bring the ROV below the surface. Pool Test #1 Footage: *https://youtu.be/X-nOXxlNkrA*

During the second pool test, the net buoyancy was overcorrected, which kept the ROV at the bottom of the pool. The thrusters did not have enough thrust to bring the ROV up from the bottom. Pool Test #2 Footage: *https://youtu.be/LullybAt6IA*

After recalculating the buoyancy, placing additional weights and buoyancy tanks on the ROV, the test was run again. This time, the ROV performed exactly as expected with a positive buoyancy of about +1 lb. The ROV was able to control depth, roll, pitch, yaw, and thrust. Pool Test #3 Footage: *https://youtu.be/VxY1e6B9kbo*

Additional Comments: Test further by using different combinations of analog stick movement. Verify that the ROV travels in the direction indicated by the controller

Test Case: Containment Unit Waterproof Test
Objective: Verify that the central containment unit (CCU) and camera containment unit do not leak
Materials: CCU, large bin filled with water
Safety: None
Procedure:

1. Lubricate the O-rings on each endcap of the containment unit with an appropriate lubricant
2. Push in the endcaps until the aluminum edge is touching the edge of the cylinder
3. Place 8 awg or larger wire in each cable gland and tighten each gland with a wrench
4. Waterproof each piece of wire by coating the ends in acrylic
5. Wait 24 hours for the acrylic to set
6. Submerge the containment unit in the water-filled bin, holding it completely under for at least 60 seconds
7. Take the containment unit out and inspect for any water inside of it
8. Dry off the outside of the containment unit and remove one of the endcaps to closely inspect the inside for any moisture

Results: Inside of containment units were dry
Additional Comments: Be careful to waterproof the wires used to plug the cable glands well, leaving no holes

Test Case: Object Detection: Max Distance
Objective: Detect the target object from up to 15 feet away
Materials: Complete ROV, tape measure, tetherball
Procedure:

1. Using the tape measure, set up markers along the pool to indicate the distance to each test point

2. Seal the containment unit with everything needed to record images from both cameras in a directory
3. Place the containment unit underwater and point it in the direction the ball will be placed
4. Use a hand signal to communicate the distance of every distance marker listed below:
   - 1 ft, 2 ft, 4 ft, 6 ft, 10 ft, 12 ft, 15 ft, 20 ft
5. Remove the containment unit and transfer the image data to the host computer
6. Run the algorithm on footage from only the left camera and count the number of frames the ball was detected and compare this to the number of frames it was expected to be found in

Results: Object was initially not detected due to a color balance calibration issue which was subsequently tuned.

Test Case: Max Depth Perception Range
Objective: Detect the distance to an object within 15 feet of the ROV
Materials: Complete ROV, tape measure, tetherball
Procedure:

1. Use the video from Test 1, run the depth perception algorithm and record the results at each distance in Step 4 of Test 1
2. Record the distance of the ball and compute the error relative to the hand signal

Results: Objective was met as a complete system, however the distances displayed on screen were not always consistent.

Test Results Summary and Analysis

The test results are summarized in *Figure 13* below. Each successive test that was performed at the pool led to improvements to the system. They also led to knowledge about the interaction of the mechanical and electrical performance, and proved that the system can function well as a whole, but can also be improved.

Although the system was not tested at depths greater than 8 feet, there is good reason to believe the system will work at 30 feet as specified in the requirements. The velocity of the ROV and the control responses have not been quantified yet to meet engineering specifications, but the performance appears to meet specifications by visual inspection. Stronger thrusters, though not absolutely necessary, could improve performance further. Once the net buoyancy problem was adjusted to be slightly positively buoyant, the T100 thrusters had sufficient force. When the net buoyancy was about 4-5 pounds (sinking the ROV) the thrusters could not move the ROV at all. Future work should consider the strength of ocean currents and how this will impact thrust capabilities.

One issue that caused repeated difficulties was making sure all of the electronics fit inside the central containment unit. After all batteries are connected, they must be strategically placed

inside the containment unit, so that the endcap can be tightened on. After the first couple of pool tests, we realized that the USB sockets of the Raspberry Pi allow the cables to easily disconnect at times when putting all cables in the containment unit, which either causes loss of connection to the Pixhawk, or power loss to the Pi. The containment units do have sufficient space, especially if the space is used wisely. The 3D printed inner battery holder can most likely be better-designed to organize the space, and any excess cables can be shortened wherever possible.

Putting cables through the cable glands in order to ensure waterproofing is a time consuming task. It requires cutting, stripping, resoldering, and epoxying many wires. This will be required any time a new thruster is swapped out, or a new cable needs to come through a cable gland. A Northrop Grumman engineer suggested investigating wireless connection capabilities through the endcap. On the morning of the demo, we noticed that a thruster had completely ripped off at a solder joint in the wire. Care should be taken to protect these joints. The break did not come because of a poor solder joint, it was the wires immediately next to the joint which appeared to have been worn down through excess movement. Evidence points to this because it was a rear thruster that was moved several times because it covers the endcap of the containment unit.

As far as the software components are concerned, they met all requirements running several processes simultaneously to operate the controls and computer vision. There is room to improve the unification of the processes and the convenience factors for the user through implementation of a GUI. The reliability of the streaming program was greatly improved since the original prototype development. One of the areas of concern is a complete operating system failure, which prevents rebooting the Pi through SSH (requiring a hard reboot). The action that causes this failure is starting the client streaming program on the Pi, when all other controls are already running (PWM signals being sent). The root of the problem is still being investigated, but it is believed to be a current limit problem caused by all programs running simultaneously on the Pi. The operating system failure was never previously experienced when the streaming program was run in isolation with both cameras connected.

	Overall Test Results				
Test #	Specification to Test	Last Date Run	Pass	Fail	Total
1	The complete ROV system can run for 30 minutes	5/19/17	1	0	1
2	Operating depth capable of up to 100 ft.	N/A	0	0	0
3	The system meets IP68 waterproofing standards	5/19/17	2	1	3
4	Strafe up/down, forward/back speeds of 2 knots	N/A	0	0	0
5	Pitch, roll, yaw angular velocity speeds	N/A	0	0	0
6	100 ft. operational range from base of ground station	5/19/17	3	0	3
7	15 ft. of underwater visibility range of the camera	5/19/17	1	0	1
8	Video display resolution at ground station is 640x480 pixels	5/19/17	3	0	3
9	Total ethernet bandwidth is less than 20 Mb/s	5/19/17	3	0	3

Figure 13: Summary of Results

Management Plan

The current work is the result of several generations of smaller prototypes used as building blocks. A major goal is to learn from the past ROV project and preserve the components that work, while also applying new technology to improve the project. There are two major principles we follow in order to achieve our goals:

- Communicate often between all interdisciplinary teams
- Apply standard design practices to provide a reusable and maintainable product

Several weekly meeting times are arranged in order to communicate with the interdisciplinary groups. There are two scheduled meeting times every week. On Thursday we meet with the entire Northrop group for an hour for a weekly update. On Saturday mornings we meet to work together on the tasks at hand. Both Cal Poly SLO, and Pomona are using a online software tool called Trello[2] to to assign and manage tasks between teams.

APPENDIX

Appendix A - Bill of Materials

BILL OF MATERIALS						
Index	**Part Description**	**Distributor**	**Part #**	**Qty.**	**Price (US $)**	**Extended Price (US $)**
1	Raspberry Pi 3	Amazon	N/A	2	35.69	71.38
2	Lights	BlueRobotics	LUMEN-LIGHT-R1	2	99.00	198.00
3	Logitech C920 camera	Amazon	Logitech C920	2	54.99	109.98
4	Yellow-colored ball	Amazon	Mikasa Yellow Tetherball	1	$21.91	21.91
5	Pixhawk	Capstone	Pixhawk mini	1	200	200
6	Controller	Capstone	Logitech	1	19.96	19.96
7	30 Amp ESC	Capstone	30 Amp ESC	6	25.00	150.00
	TOTAL					***711.23***

***Note: Purchases were made by prior capstone teams. This is an estimation based on parts that were used.**

Appendix B - Contact Information

Organization	Name	Email	Phone
Software: Computer Vision	Jeremy Passehl	jeremy.passehl67@gmail.com	970 275 5578
	Kelly Sio	ksio@calpoly.edu	415 203 0522
	Khanh Le	kle32@calpoly.edu	714 902 3651
Software - Controls	Jared Spadaro	jared458@gmail.com	-
	Sam Freed	sfreed@calpoly.edu	-
	Ryan Frawley	frawley.ryanj@gmail.com	-
	Emily Lopez	elopez21@calpoly.edu	-
Previous ROV Work	Lynne Slivovsky	lslivovs@calpoly.edu	-
	Tyler Mau	tmaudev@gmail.com	-
Mechanical Engineering	Jakob Graf	2jgraf24@gmail.com	-
	Elizabeth Hawkinson	ehawkinson32197@gmail.com	-
Electrical Engineering	Uriel Serna	UrielSerna567@gmail.com	
Lights	Dr. Paavo	bpaavo@calpoly.edu	-

Appendix C: References

1.https://books.google.com/books?id=pjW1QoX1KeIC&printsec=frontcover#v=onepage&q&f=false

2. https://trello.com/b/ZqRKb4Fm/rov

3. 2015-2016 Senior Project Report: Tyler Mau and Joseph Mahoney

4. 2016-2017 Team Discovery Capstone Final Report

5. 2016-2017 Computer Vision Capstone Final Report

6. https://github.com/NGCP/

7. http://www.imca-int.com/remote-systems-and-rov-division.aspx

8. https://www.bluerobotics.com/store/thrusters/t100-thruster/

9. Pool Test #1 Video Footage: https://youtu.be/X-nOXxlNkrA

10. Pool Test #2 Video Footage: https://youtu.be/LullybAt6IA

11. Pool Test #3 Video Footage: https://youtu.be/VxY1e6B9kbo

ROV
User Manual

Jared Spadaro
Jeremy Passehl
5/30/17

Table of Contents

Setup Guide

In order to have a fully operational vision and control system for the ROV, there are important steps to take to ensure the system will work as intended. We will begin with the setup of the Raspberry Pi microcomputer.

Raspberry Pi Setup - Controls

Download the latest ArduSub Raspberry Pi image from http://ardusub.com/firmware/#firmware. Flash this image to the microSD card that you are using for the Raspberry Pi 3 using any tool you like (e.g. Rufus, Win32DiskImager, dd, etc).

Raspberry Pi Setup - Computer Vision

Materials:

1. 32GB microSD card
2. Rapsberry Pi 3
3. Laptop with internet connection
4. Ethernet tether

Steps:

1. Download and install SDFormatter on laptop
2. Download and install Win32DiskImager on laptop
3. Download the latest image of Raspbian on laptop
4. Format the SD card using SDFormatter
5. Image the SD card with the Raspbian image, then plug it into the Rpi and boot up

Additional Required Software:

1. Firmware update:

```
sudo apt-get install rpi-update raspi-config
sudo apt-get update
sudo rpi-update
reboot
```

2. Expand filesystem
    a. `sudo raspi-config` -> Expand Filesystem
    b. `df -h` , should output approximately equal to microSD card size (32GB)
    c. Remove Wolfram engine to free up 700mb if needed , `sudo apt-get purge wolfram-engine`

3. Install dependencies
    a. `sudo apt-get install build-essential cmake pkg-config`
    b. `sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev`, image IO
    c. `sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev`
    d. `sudo apt-get install libxvidcore-dev libx264-dev`, video IO
    e. `sudo apt-get install libgtk2.0-dev`, highgui support
    f. `sudo apt-get install libatlas-base-dev gfortran`, misc dependencies

4. Download OpenCV Source
    a. `cd ~/Downloads`
    b. `wget -O opencv.zip` https://github.com/Itseez/opencv/archive/3.1.0.zip
    c. `unzip opencv.zip`
    d. `wget -O opencv_contrib.zip` https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip
    e. `unzip opencv_contrib.zip`

5. Compile and Install
    a. `cd ~/Downloads/opencv-3.1.0/`
    b. `mkdir build`
    c. `cd build`
    d. `cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D INSTALL_PYTHON_EXAMPLES=OFF -D INSTALL_C_EXAMPLES=OFF -D OPENCV_EXTRA_MODULES_PATH=~/Downloads/opencv_contrib-3.1.0/modules -D BUILD_EXAMPLES=ON ..`

```
e. ccmake ./
f. Double check flags and make sure Eigen include path is set
   for /usr/local/include/eigen3/
g. make -j4
h. sudo make install
i. sudo ldconfig
```

PC Setup

In order to communicate with the Raspberry Pi over ethernet, the ground station PC must be configured to have an IP address of 192.168.2.1. Before launching QGroundControl, connect the game controller to the PC. Start QGroundControl and navigate to the settings page. Once there, select the “Joystick” option. Calibrate the controller by following the on screen prompts. You can also remap buttons on this page if you wish to.

Pixhawk Setup

Pull the custom frame configuration firmware off of the github repository located at **http://www.github.com/NGCP/ROV2.0/firmware** and load it onto the pixhawk

Connecting Raspberry Pi and PC

Connect Raspberry Pi to portable battery pack, and connect each camera to Pi via USB. Connect Pi and PC to each other via ethernet cable. SSH to Pi with the command *ssh pi@192.168.2.2* and enter the password “raspberry.” Go to the directory containing the client executable on the pi, and run the command *./client 192.168.2.1 12345 54321*. On the host PC, go to the directory containing the server executable and run the command *./server 12345 54321*. You should see video feeds pop up on the host PC.

Running the ROV

Once the electronics have been set up, the system can now be used and tested. Follow this guide to learn how to use all parts of the system and verify its operation

Deploying the ROV

Before Deployment

1. Ensure there is no bare wire exposed.
2. Ensure all o-rings are tightly sealed.
3. Lubricate the thrusters using silicone lubricant prior to submerging the UROV.
4. Connect your ground station (laptop) to the tether using an ethernet cable and ensure you can communicate with the UROV.

Deployment

1. Gently submerge UROV in water.
2. Check for any leaks by watching for air bubbles.
3. Ensure there are no wires or other obstructions that could block the thrusters.
4. Arm the thrusters (start on the controller).
5. Complete the mission.
6. After the mission, disarm the thrusters (select on the controller)
7. Gently lift the UROV out of the water.
    a. Try not to pull the UROV in using the tether if possible.

After Deployment

1. Rinse off all parts thoroughly with fresh water to prevent corrosion.
2. Remove lithium ion batteries and store them safely.

Running Streaming / Client - Server Communication

Once the programs are running on the GCS and the RPi:

1. Use all of these keyboard commands (**Table 1**) and verify the output corresponds to the command

Key Pressed	Motion Command
s	Run unprocessed streaming
o	Run object detection
d	Run depth perception
e	Exit

Table 1: Keyboard Commands

2. After verifying all these commands in the terminal and browser the client server communication is working properly. Terminate the programs using Ctrl + C

Troubleshooting

Symptom	Issue	Solution
System does not power up	1. Batteries are low 2. Wires not connected	1. Remove the batteries and charge them 2. Ensure all connections are solid
Containment unit leaks	1. Seal is not tight enough	1. Make sure to firmly press the endcap onto the containment unit so there is no gap. 2. Lightly apply some Magic Lube to the o-ring
Thrusters spin by default	1. Control mode is not default 2. Thrusters need to have their default state set	1. Ensure the control mode on QGroundControl is Manual 2. Find the joystick positions where the motors do not move and then click the right thumbstick to lock that as the default motor state
UROV is unbalanced in water	1. Weight distribution is not centered	2. Attach weights to the UROV until it stays balanced in the water.

Useful Links

ArduSub Website
http://ardusub.com/introduction/
QGroundControl Website
http://qgroundcontrol.com/
OpenROV Operator's Manual
http://openrov.dozuki.com/Guide/OpenROV+Operators+Manual/80