Bottom Feeder – A Remote Operated Underwater Vehicle with Metal Detecting Capabilities

John Cope, T Davis, Tyler Rose, Sarah Reim

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — Living in Florida, our lives are deeply entwined with the water. Many of our hobbies, our recreations, our professions, and our stories touch the wave as some point or another. Our motivation for creating Bottom Feeder was to alleviate some of the drawbacks to our watery way of life. Jewelry, keys, and keepsakes are liable to be lost in waterways. We created a remotely operated underwater vehicle capable of locating these objects through metal detecting so they can be retrieved from the body of water's floor. Our project, Bottom Feeder, aims to de-risk underwater exploration by removing the human from the underwater environment. The contents of this paper will explore the main functions of our ROV which include the outputs, inputs, controls, power, and mechanical components of our system.**

***Index Terms* — Aquatic robots, robot programming, data communication, metallic materials, remote handling**

I. Introduction

Human underwater exploration always involves some amount of risk. Scuba divers can get stuck or tight spaces or can get their airline caught on rocks or coral. Scuba divers who surface too quickly can experience decompression sickness and have to spend an extended period of time in a hyperbaric chamber. Small personal submarines can develop catastrophic leaks, lose power, or simply run out of air. Our project aims to de-risk underwater exploration by removing the human from the underwater environment. By developing and using an underwater submersible vehicle we can perform tasks such as recording video and detecting metal objects all without risking human life.

We designed the entire system of the ROV. This includes the ROV enclosure itself all the way up to the raspberry pi, which receives and processes information from the ROV, outputs that information to the user in a live feed, and takes user inputs from a controller. We then use the laptop to relay information down to the ROV to tell it how the operator wants to move. The underwater vehicle is then responsible for recording omnidirectional video and processing it into a useable video format that is sent back up to the laptop through the tether. Stable movement is important for video recording, so the ROV is equipped with an inertial measurement system and a control program which keeps the ROV stable while underwater. Video quality is one of the most important features of the ROV, so we took special consideration when choosing our cameras, designing the placement of the cameras, and the intensity of the underwater lighting.

Our aim was to make the system intuitive and easy to control. The ROV is ready to work out of the box with no complicated setup protocols needed. The only thing the user should expect is to turn the device on and throw it into the water for exploration.

II. System Overview

A. Hardware

Bottom Feeder is a collection of features that are combined in a manner that allows the user the freedom to explore the waters in a new way. Our ROV's principal duties are locomotion and observation. Following this are task involving manipulation of the environment. In order to facilitate these goals, a physical design was conceived and built. The block diagram of our physical hardware design is shown in Fig. 1.

The hardware for this entire ROV system is broken up into a surface station and the ROV itself. These two parts are connected using a tether which allows for data and power to be transferred between the two systems. The surface station consists of a computer, a headset, a controller, and a power supply. The computer runs the code that receives controller input from the controller, sends that input through the tether to the ROV, and receives and processes the video and sensor data coming through the tether from the ROV. The surface station then takes this processed video and sensor information and displays it to the user through the headset.

The ROV consists of everything that is underwater, including the onboard computer, the thrusters, a metal detector, lights, sensors, and a power system. The onboard computer is responsible for collecting and encoding the camera feed and sending it, along with sensor information, to the surface station. The ROV takes any controller inputs that it receives and applies the correct power and direction to the appropriate motors. It is also powered solely by onboard batteries.

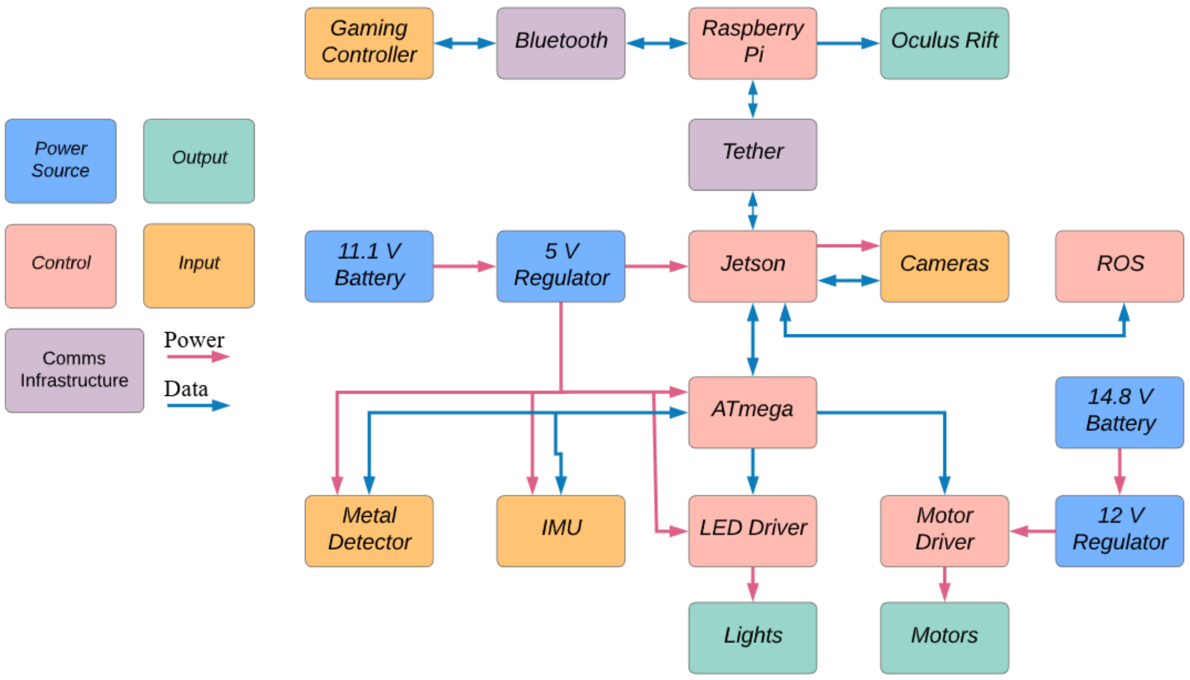

Fig. 1 Bottom Feeder hardware block diagram

B. *Software*

The software that controls our ROV is developed using the Robot Operating System (ROS) on Ubuntu Linux. The ROV initializes its cameras and sensors, connects to the surface station, and then sends video and sensor information while also receiving all controller input from the surface station. The ROV then takes that controller input and determines what motors need to be turned on, what direction they need to spin, and how much power they need to get. The surface station also connects to the headset and the controller, establishing a communication link with the ROV. It then receives video stream from the ROV, displays it on the headset, and sends all controller inputs to the ROV.

III. Output

A. *Propulsion*

ROVs need a way to move around underwater. Without it, your ROV will drift around aimlessly controlled only by the current of the water. ROVs use electric motors to spin propellers in the water. The most important things we considered in designing the propulsion system was where the motors would physically be, and how powerful they were. For our ROV, we used H-bridge motor driver integrated circuits to control the speed and direction of the motors. These motor drivers take a pulse width modulated signal from the microcontroller in order to control the speed. The direction in which the propellers spin is determined from two digital control inputs. Our ROV uses two 1100 GPH modified bilge water pumps for vertical movement and two Johnson Mayfair 1000 GPH bilge pump cartridges for forwards and backwards movement. All designs were printed with a 3D printer and tested with a homemade test bench. Clockwise and counterclockwise propellers were made for each motor pair so that the ROV stays stable in the water. The propellers have 12 mm long fins and a 35-degree pitch. Each motor produces about half a pound of thrust at full power.

B. *L6205*

We designed our motor drive system using the L6205 driver from STMicroelectronics, specifically the PowerSO package for its increased ability to radiate heat through ground planes and via stitching. These drivers are able to provide up to 3 amps per motor. Under load, these typically pull 2 amps. The package is a Dual H-Bridge, and one integrated circuit can support two independently operated thrusters.

We have placed PTC resettable fuses in series with these motor connections, to offer a fail safe in the event that a motor jams. They are rated to begin throttling current after 3.5A.

In these areas of increased current flow, vias were placed more densely, in cases where a trace needed to move from one side to another, or generally where heat dissipation was of a greater concern.

C. *Lights*

The measurement for light is called the Lumen. A lumen is, by definition, the SI unit of luminous flux, equal to the amount of light emitted per second in a unit solid angle of one steradian from a uniform source of one candela. One must take into consideration the lumen value associated with the product they are purchasing to determine brightness. Our initial target range of depth was 30 meters, so we choose an LED that packed at least 1000 lumens. Specifically, we choose the Cree XLamp XP-L-665-1 as our LED and placed two to either side of our enclosure's acrylic domes. With the amount of light being produced we can see underwater at night, with the ability to reach further depths.
The reflectors chosen for the LEDs were the 10048 Carclo Lens, which greatly helped focus our light and had an 87% efficiency rating. The AL8843Q LED driver was used for our allows for PMW brightness control as well as constant current driving. A picture of our LED driver schematics can be found in figure 2 below.

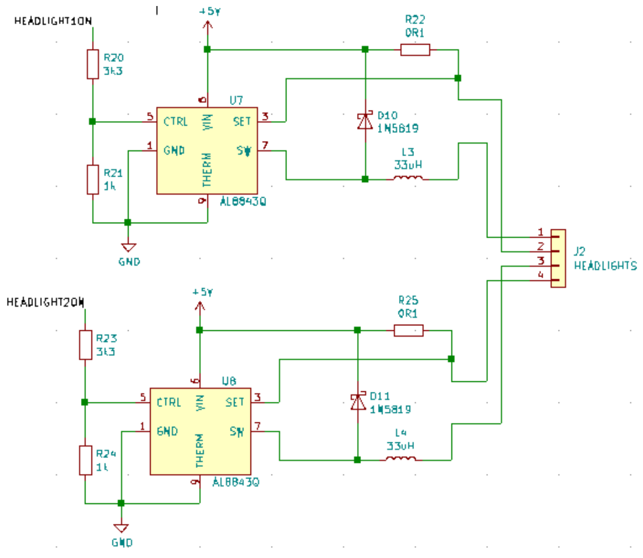

Fig. 2 LED Driver Schematics

D. *Oculus Rift*

The Oculus Rift Development Kit 2 has long since been reverse engineered and documented by the Open Head Mounted Display project. Their libraries allowed for rapid development of the virtual system for viewing the fisheye images in a comfortable viewsphere environment. In a Java application, we bind to the Rift's API, to receive orientation information about the headset, and, in response to this orientation data, we rotate the virtual world around a fixed camera. On this virtual gimbal, we can also attach heads up display elements, such as a compass heading and horizon line. To create fixed heads up display elements we render them in front of the fixed camera, without attaching them to the Rift's orientation driven gimbal.

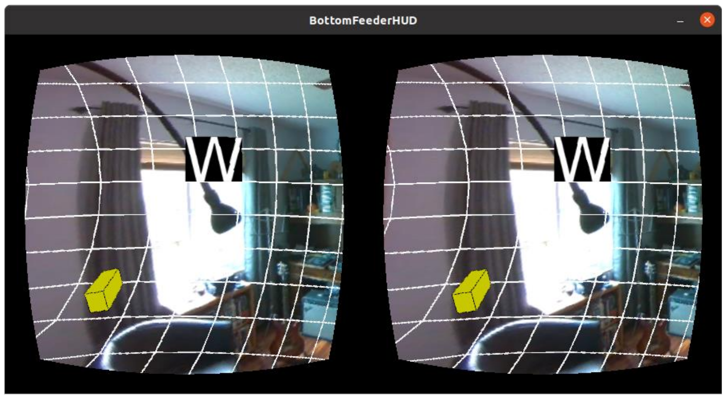

Fig. 3. Oculus View

In order to create a comfortable to view image for the operator, the Java library Camera3D is used to both generate simultaneous stereo images while creating the framebuffer, and applying the proper barrel distortion to the framebuffer so that when viewed through the Rift's lenses, the image appears natural.

IV. Input

A. *Cameras*

For our ROV's visuals we used the IMX219 sensor. It was chosen for its MIPI-CSI compatibility which is supported by the NVIDIA Jetson Nano. Its resolution is 3280 by 2464 pixels. Due to their integrated lenses, they have a 200 degree field of view. The broad field of view allows for total omnidirectional video using only two cameras. The sensor is sized to show the full circular image from the lens which is critical for our image distortion.

In order to process the two camera feeds and create an equirectangular video stream, we make use of the

NVIVIA Jetson Nano's graphics processing unit. Accessing the camera data via MIPI-CSI required using the EGLStream libraries from NVIDIA's own JetPack Camera API. These allowed for access of the data via the OpenGL Shading Language (GLSL). At full resolution, 30 fps was achievable while sampling both cameras.

In GLSL, slices of the source texture, the camera feed, can be distorted and mapped to new coordinates of the framebuffer, or the outgoing video stream. Having few, fixed translations for these texture maps allows very optimal usage of the Maxwell GPU on the Nano. Using the GStreamer API, this framebuffer can be compressed using the H265 algorithm, to limit bandwidth usage, and then streamed via UDP to the surface station for presentation to the system operator.

The camera stream is relayed back to the surface station, and live video can be viewed by the operator. This video feed could then be fed into the virtual environment and allow the operator to have an immersive experience while piloting the craft.

B. *Metal Detecting*

Our ROV is equipped with a waterproof metal detector to find lost items in water. [1] In general, metal detectors function by using an electrically charged search coil to create an electromagnetic field. When this coil is brought close to metal objects, it affects the way the electrons move inside the metal, creating electrical current. The metal then gives off its own electromagnetic field, which is recognized by the detector. There are several means of detecting metal. In our simple design, we used the law of inductance. Basically, a changing magnetic field results in an electric field that opposes this change [2]. So, when we find metal, a voltage will be created across the coil to oppose the change in current. A measure of this change is called inductance, and its unit is in Henrys. When a coil's current changes in one ampere per second, the amount that generates one volt is considered 1 Henry. We can get a rough estimate of the inductance on the coil with the use of an ATmega and a simple RLC design, a capacitor, and a diode. The coil is part of a high-pass LR filter that is given a DC square wave input from the microcontroller. Short spikes are then generated, and the pulse length of these spikes are directly related to the coil's inductance. However, because it is hard to get an accurate inductance because of the ATmega's internal clock frequency, we use a capacitor being charged instead. The capacitor is charged from the rising pulse and then a value is read from the microcontroller's analog to digital converter. Our code continuously sends out a pulse and reads the voltage of the capacitor several times. When there is a change, it lets the user know there's metal present. For our metal detecting coil, we created a 100 turn PCB coil. Three coils are used in Bottom Feeder to improve our area of detection.

C. *Gaming Controller*

The wireless controller we are using connects to the surface station via Bluetooth. To ensure that the controller was working properly, and that the Bluetooth connection was established, a test code was written to test all of the features of the controller. The test code ensured that all of the digital buttons were working, that the two joysticks gave correct values in both the x and y directions, and that the two analog triggers gave an appropriate analog reading when pressed. The controller was set up with the two analog sticks each controlling one of the forwards and backwards motors. The two up and down motors were controlled using the two trigger buttons on the top of the controller. Finally, the headlights are able to be toggled on and off using the cross button.

V. CONTROL

A. *Jetson*

The Nvidia Jetson Nano is a single board computer that was designed with enough power to run artificial intelligence projects, neural networks, and other large development projects. The Jetson Nano runs on a quad-core ARM Cortex-A57 MPCore processor with a dedicated graphical processing unit running Nvidia's Maxwell architecture with 128 NVIDIA CUDA cores putting out 0.5 teraflops. The Jetson Nano has 4 GB of 64-bit LPDDR4 RAM running at 1600Mhz, and has 16 GB of eMMC 5.1 Flash memory built in. This board runs on NVIDIA's special build of Ubuntu Linux known as Jetpack SDK. The Jetson Nano's biggest advantage is that is has two MIPI camera input ports which made using our two-camera setup very straightforward and allowed for omnidirectional video processing. The increased graphical processing power of the Jetson Nano enabled us to send higher quality video from our ROV to the surface. It sends processed video feed and sensor information to the surface station using our tether while also receiving input commands using the same tether. [3]

B. *ROS*

The Robot Operating System (ROS) is a robotics integration platform that contains a collection of tools, libraries, and conventions that help simplify the development of complex robotic behavior. The current version of ROS that is compatible with the Nvidia Jetson Nano is ROS Melodic Morenia. While it's called Robot Operating System, it doesn’t replace the operating system on the Jetson Nano, instead it is installed on top of Ubuntu Linux. The main benefit for our use of ROS is that it provided a simple message passing interface that was able to communicate with many different devices. For our project, ROS is the backbone that allowed us to transmit video and sensor data from the ROV to the surface, and then transmit bluetooth controller inputs back to the ROV in order to control the motors. The message passing interface of ROS is built on a publish and subscribe mechanism which allows all of the parts of our robot communicate with each other in a cohesive and easily manageable way.

C. Microcontroller

The single board computer is able to do a lot of complex calculations, but they are not typically designed to be used for sensor inputs or for controlling motors. The sensor inputs needed an analog to digital comparator circuit, and the motors needed pulse width modulation pins in order to control the speed and direction of the motor. The easiest way to add this functionality to our project was to add a separate microcontroller chip to the system. The ATmega328P satisfied all our needs. The chip itself has twenty general purpose input/output pins, three timers with compare modes, and internal and external interrupts. This chip has a ten-bit analog to digital converter, serial programmable USART, SPI functionality, and communication using I^2C. [4] It’s connected to the Jetson Nano using UART and has all the pins needed to send control signals to our motor drivers and LED drivers. Sensor data was collected using the analog pins, sending the information to the rest of the ROV. This microcontroller was programmed using an AVR programmer plugged directly into our printed circuit board and is powered using our 5-volt rail.

D. UART

To enable communication between the ATMEGA’s 5V UART connection and the Jetson Nano’s 3.3V UART connection, bidirectional level shifteres were implemented with P-channel MOSFETS as well as pullup resistors on the communication traces.
[BIDIRECTIONAL.PNG]
These enabled the microcontroller to speak directly to the Jetson through the Jetson’s LinuxGPIO functionality.

E. PCB

In order to draw the schematics for our project, and to lay out the printed circuit board, we chose to use KiCAD, which is a collection of EDA softwares. Of the included packages, we used Eeschema for schematic capture and PCBNew for layout and routing. The myriad design files are housed in a Git repository along with our software, and all team members had access to this cloud-based version control system.

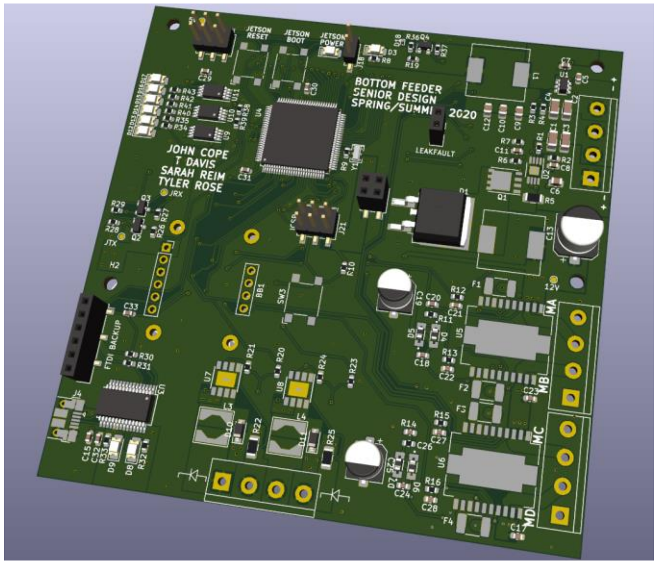

Fig. 4 Our main PCB

VI. Power

A. Batteries

We used two different types of LiPo batteries for our submerged system, The Turnigy 11.1 volt 3200 mAh and the Turnigy 14.8 volt 5000 mAh. The 11.1-volt battery was responsible for powering our ROV’s lower-powered components which included the lights, IMU, and Jetson. The 14.8-volt battery was used for powering the higher power components which consisted of the motors and the metal detector. Together, these batteries reached our desired runtime of about an hour and were the most cost-efficient power alternative.

B. Regulators

The ROV has several low power components that need to be powered by an 11.1 Volt LiPo battery. For these components to be properly powered we have to regulate the voltage coming from the battery and properly convert it to the correct voltage/current for the components.

Our low power domain makes use of the TPS565208 regulation IC, that is able to provide an output potential ranging from 0.76V to 7V. This provides an opportunity to the designer to minimize the size of the bill of materials, by reusing the same component in order to provide both 3.3V and 5V potentials to the digital components. The IC uses a reference signal provided to the voltage feedback pin, annotated in schematic as VFB. This IC creates its output based on equation 1 found below. These schematics are designed to produce no more than 4amps, as the NVIDIA Jetson Nano loaded with cameras should draw 2.5A, and associated logic ICs contribute a few dozen milliamps each to the load.

$$V_{OUT} = 0.760 \cdot \left(1 + \frac{R1}{R2}\right) \tag{1}$$

To maintain a relatively clean electrical domain for the digital electronics, we separated the thruster driving controls to a separate power supply. These two domains' grounds will be linked through an inductor to prevent ripples on the ground plane to perturb the digital circuits. The high power design elects to use an LM25085MY buck switching controller that manages the duty cycle of a P-FET rated to carry up to 70amps. Maintaining this switching creates a small ripple, smoothed by the output capacitors, and does not cause issues for the motors, that will be under their own PWM influences. When using Texas Instruments' WebBench designer, we specified that this high power circuit should be able to deliver 10 amps.

The design allows for a predictable noise on the outgoing voltage rails. In the case that water creates a short between the positive supply potential and ground the IC features an integrated thermal shutdown, should the power consumed by the aqueous component create an undue burden on the regulator.

The 5V supply in the design was not compatible with being back driven by another 5V source, and would fail in this configuration. In the final demonstration, this 5V regulator was bypassed and a prefabricated replacement was used in its place.

VII. Mechanics

The physical design of our ROV was critical in our project's success. To maintain functional electrical systems, they must be protected from contact with the surrounding environment of water. Our enclosure had to be air-tight with feedthroughs that allowed our tether to send and receive data from the surface station to our submersible.

A. *Enclosure*

The main body of our enclosure is a foot long, six inch diameter, PVC pipe. The endcaps were made out of laser cut, half inch thick acrylic, and three inch diameter domes were glued onto the endcaps to make a viewing port for our wide field of view cameras. Six threaded rods were used the hold everything together and provide mounting locations for our motors and our skids. The motor mounts and the skid mounts were all designed by us in Autodesk Fusion 360, and 3D printed in ABS plastic. The skids themselves, and the mounting locations for the top motors were made with quarter inch, laser cut acrylic. The first buoyancy test of the enclosure showed that our ROV so buoyant that half of the entire ROV was out of the water. Weights were placed on the top of the ROV to get an idea of how much weight needed to be added and large fishing weights of various were acquired. Four 12 ounce weights were added to each corner on the outside of the ROV, and over a pound of weight was added to the inside of the enclosure itself. The some small weights were added one at time afterwards until the ROV reached neutral buoyancy, with it have a slight tendency to sink to ensure that the top motors were always in the water.

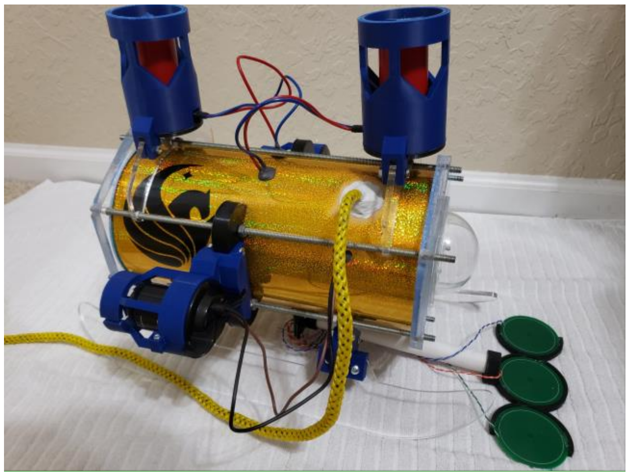
Fig. 5 ROV Enclosure

B. *Feedthroughs*

Feedthroughs are components that allow electrical connections to pass through a physical barrier. These can be rated to separate high pressure systems or vacuum systems from the atmosphere, or to allow electrical signals to penetrate a submerged vessel without water making ingress. When selecting a feed through, it is important to select one that matches the signal content and current requirements of the wiring that will connect to either side. We elected to use a dry mate RJ45 feedthrough for our tether, and a permanent compression gland feedthrough system for all onboard connections. The dry mate RJ-45 connector system allows for off the shelf Ethernet connectors and CAT6 cabling to carry data without moving to a multi-hundred-dollar solution for feedthroughs.

C. *Tether*

The tether, sometimes called the umbilical, is the cable linking the body of the submersed ROV to the tether management system on the surface. It is used to supply power and communicate with the electrical components on the underwater vehicle. In our case, we are simply using our tether as a means of communication while letting the ROV's onboard batteries be fully responsible for the power. Our tether consisted of 50 feet of category 6 ethernet wire with 1/4th inch hollow braided polypropylene rope wrapped around it to protect our cord from strain when being dragged underneath the water. The ethernet cable itself had stranded wire allowing for more flexibility so the tether was less likely to become damaged while being bent.

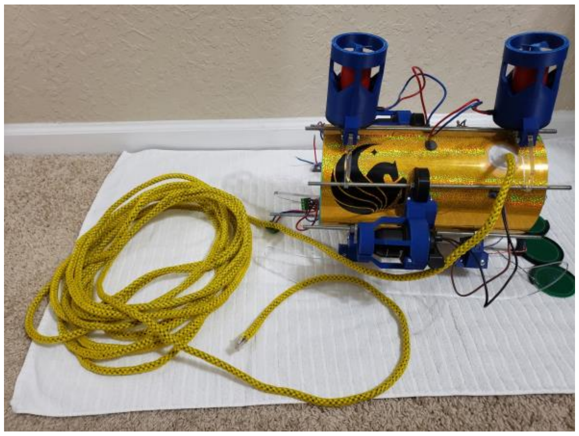
Fig. 6 Tether

VIII. Conclusion

In conclusion we were able to produce a functional ROV prototype fulfilling many of our original engineering requirements. The ability to move about in an underwater environment by means of electric thrusters was critical to our success. We achieved this early on, and extend this control of the thrusters to the operator through the connection of the Dual Shock controller, and bridge the gap between devices using the ROS environment of publishers and subscribers.

Additionally metal detecting allows the operator to do meaningful work with the device. By applying basic RLC principles we created a system that would be sensitive to metal objects in the vicinity of our device, satisfying the metal detection requirement. In some cases, the system was able to detect a cell phone up to three inches away, while smaller objects like gold coins were detectable at an inch or less of distance from the coil.

To assist in the piloting of the craft as well as the hunting for lost metallic items underwater, a camera system was implemented to immerse the operator in the underwater world. Wide angle, 200 degree fisheye lenses on 10 megapixel sensors allowed a great deal of visual information to be relayed back to the surface station. These images could be combined with an immersive virtual viewing environment, allowing the use of an Oculus Rift headset to be the pilot's link to the device.

While challenges were faced along the design and construction of our device, seeing success in individual components shows promise. Due to the COVID-19 pandemic, social distancing was required to mitigate risk of transmission between team members. Test benches and equipment had to be procured in duplicate, so that independent development could proceed in parallel. This lead to some systems being designed very well in isolation, and the restrictions on meeting constrained our team during critical integration steps.

Despite these shortcomings, Bottom Feeder has proved itself to be a promising newcomer to the field of robotic, and aquatic, exploration.

Fig. 7 Final demonstration

Acknowledgement

Our family members, for their support and patience.
Dave Casey, for his generosity and guidance
P3D Creations for their fabrication services
Tyler and Molly Jette, for the use of their pool & patio

References

[1] C. Woodford, "Metal detectors," *Explainthatstuff*. [Online]. Available:

https://www.explainthatstuff.com/metaldetectors.html. [Accessed: 19-Jul-2020].

[2] "Simple Arduino Metal Detector," *Instructables*. [Online]. Available: https://www.instructables.com/id/Simple-Arduino-Metal-Detector/. [Accessed: 10-jul-2020].

[3] “Jetson Nano,” Autonomous Machines. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/. [Accessed: 10-Mar-2020].

[4] “Atmega328P,” Microchip. [Online]. Available: https://www.microchip.com/wwwproducts/en/ATmega328P. [Accessed: 13-Apr-2020].